

Б.М.Каган

ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И СИСТЕМЫ

*Издание второе,
переработанное и дополненное*

Допущено Министерством высшего и среднего специального образования СССР в качестве учебного пособия для студентов вузов, обучающихся по специальностям: «Автоматизированные системы управления», «Прикладная математика», «Электронные вычислительные машины»



МОСКВА
ЭНЕРГОАТОМИЗДАТ
1985

ББК 32.973
К 12
УДК 681.3—523.8(075.8)

Рецензент: Кафедра ЭВМ и ВС МВТУ им. Н. Э. Баумана

Каган Б. М.

К 1 Электронные вычислительные машины и системы: Учеб. пособие для вузов. — 2-е изд., перераб. и доп. — М.: Энергоатомиздат, 1985. — 552 с., ил.
В пер.: 1 р. 40 к. 60 000 экз.

Второе издание книги переработано и дополнено с целью отражения новых идей и архитектурных решений, характерных для современной электронной вычислительной техники. Изложены основы теории и принципы организации микропроцессоров, ЭВМ, вычислительных систем и сетей, а также принципы построения их основных элементов, узлов и устройств.

Для студентов вузов, обучающихся по специальностям «Электронные вычислительные машины», «Автоматизированные системы управления» и «Прикладная математика». Представляет интерес для специалистов, работающих в области разработки и применения ЭВМ, АСУ, АСУ ТП и их математического обеспечения.

К 2405000000-012 267-85
051(01)-85

ББК 32.973

6Ф7.3

© Энергоатомиздат, 1985.

ПРЕДИСЛОВИЕ

Развитие электронной вычислительной техники и ее широкое применение в производстве, научно-исследовательских и проектно-конструкторских работах, плановых расчетах, сфере управления являются в настоящее время одним из основных направлений научно-технического прогресса.

В принятых XXVI съездом КПСС «Основных направлениях экономического и социального развития СССР на 1981 — 1985 гг. и на период до 1990 года» предусмотрено дальнейшее совершенствование вычислительной техники, ее элементной базы и программного обеспечения, средств и систем сбора, передачи и обработки информации, повышение эффективности автоматизированных систем управления, развитие сетей ЭВМ и вычислительных центров коллективного пользования, создание на основе применения ЭВМ систем автоматизации научных исследований и проектирования, широкое использование микропроцессорных средств для автоматизации машин, оборудования и приборов и построение на этой базе автоматизированных предприятий и технологических комплексов.

Для выполнения этих задач необходимы в широких масштабах подготовка и переподготовка специалистов по ЭВМ, АСУ, прикладной математике, применению ЭВМ и микропроцессоров в различных отраслях народного хозяйства.

Разнообразие проблем теории и методов проектирования объектов вычислительной техники, сложность ее элементов, устройств, машин и систем закономерно привели к тому, что из дисциплины «Теория и проектирование ЭВМ», еще недавно охватывавшей все основные аспекты этой области науки и техники, выделились самостоятельные курсы, например, по схемотехнике ЭВМ, периферийным устройствам и др.

Вместе с тем для студентов специальностей, ориентированных не на разработку ЭВМ, а на их использование, а также для широкого круга специалистов народного хозяйства необходима книга, в которой без излишней детализации рассмотрен

весь комплекс основных вопросов, относящихся к теории и принципам построения ЭВМ и систем. Этой цели служило предыдущее издание книги, опубликованное в 1979 г.

Масштабы подготовки в вузах специалистов по АСУ, прикладной математике и ЭВМ, появление за последние годы многих новых важных идей и технических решений в электронной вычислительной технике, расширяющих области применения ЭВМ и способствующих вовлечению в активную работу по использованию ЭВМ (в том числе микропроцессоров, микро- и малых ЭВМ) для управления процессами и обработки данных широкого круга инженеров, часто не имеющих специальной подготовки, вызывают необходимость выпуска второго, переработанного и дополненного издания книги.

Переработка книги проводилась в основном в следующих направлениях.

В последние годы «мир» электронных вычислительных машин значительно расширился – в нем наряду с машинами общего назначения заняли большое место малые ЭВМ, микропроцессоры, микро-ЭВМ и персональные ЭВМ. Это делает необходимым обобщение технических решений в области логической организации машин различных классов, с тем чтобы можно было с общих позиций рассматривать архитектуры микропроцессоров, микро-ЭВМ, малых ЭВМ, ЭВМ общего назначения. С этой целью в книгу включена новая глава по элементам архитектуры ЭВМ и микропроцессоров, а главы по принципам организации процессоров и систем ввода-вывода подверглись существенной переработке.

В книге отражены новые идеи и решения, характерные для ЕС ЭВМ II очереди, существенно расширен материал по малым машинам, микропроцессорам и микро-ЭВМ.

Возрастание функциональной связности аппаратурной части и операционной системы ЭВМ при реализации различных режимов и тенденция аппаратурной поддержки операционных систем потребовали изложения принципов взаимодействия аппаратурных и программных средств при управлении вычислительным процессом. Расширено также изложение вопросов организации многомашинных и многопроцессорных систем. С учетом современной тенденции развития вычислительной техники в книгу включена глава о принципах организации вычислительных (в том числе локальных) сетей.

В книге по вычислительной технике, предназначенной для будущих инженеров-системотехников и инженеров-математиков, наряду с рассмотрением принципов организации ЭВМ и систем необходимо уделить внимание теоретическим моделям протекающих в них процессов.

На основе последних результатов вероятностной теории вычислительных систем удалось придать изложению этих вопросов достаточно обобщенный характер. При принятом подходе модели разнообразных процессов функционирования вычислительных машин и систем, например таких, как функционирование каналов ввода-вывода, систем коллективного пользования, многопроцессорных систем, вычислительных сетей и других, оказываются частными случаями одной обобщенной теоретической модели. Представилось целесообразным эти вопросы, которые в предыдущем издании рассматривались лишь частично и в разных главах, сосредоточить в одной новой главе. Подверглись переработке и другие главы предыдущего издания.

Глава 17 написана автором совместно с канд. тех. наук Л. Б. Богуславским.

Автор благодарен своим коллегам и аспирантам во ВНИИЭМ и на кафедре электронных вычислительных машин МИИТ, оказавшим ему на разных этапах написания этой книги помощь в подборе и обработке материалов по отдельным вопросам.

Автор выражает глубокую признательность доктору техн. наук, профессору Ю. М. Смирнову, доцентам Л. В. Суркову и К. С. Хомякову, всему коллективу кафедры ЭВМ и ВС МВТУ им. Н. Э. Баумана, а также редактору книги доц. В. Г. Першееву за ценные советы и замечания, которые в большой степени способствовали улучшению содержания данного издания книги.

Автор будет весьма признателен всем читателям, приславшим свои замечания и пожелания по адресу: 113114, Москва, М-114, Шлюзовая наб., 10, Энергоатомиздат.

Автор

ОСНОВНЫЕ ПОНЯТИЯ

1.1. ДВА КЛАССА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

Любая форма человеческой деятельности, любой процесс функционирования технического объекта связаны с передачей и преобразованием информации.

Одно из важнейших положений кибернетики состоит в том, что без информации и ее переработки невозможны организованные системы, какими являются живые организмы и искусственные, созданные человеком технические системы.

Информацией называют сведения о тех или иных явлениях природы, событиях в общественной жизни и процессах в технических устройствах. Информация, воплощенная и зафиксированная в некоторой материальной форме, называется *сообщением*. Например, при планировании и управлении производством собираются и обрабатываются сведения (сообщения) о потребностях в той или иной продукции, ресурсах рабочей силы, сырья и материалов, производительности станков и другого оборудования и вырабатываются соответствующие управляющие решения.

Сообщения могут быть непрерывными и дискретными (цифровыми).

Непрерывное (аналоговое) сообщение представляется некоторой физической величиной (электрическим напряжением, током и др.), изменения которой во времени отображают протекание рассматриваемого процесса, например изменения температуры в нагревательной печи. Физическая величина, передающая непрерывное сообщение, может в определенном интервале принимать любые значения и изменяться в произвольные моменты времени.

Для *дискретных сообщений* характерно наличие фиксированного набора элементов, из которых в некоторые моменты

времени формируются различные последовательности. Важным является не физическая природа элементов, а то обстоятельство, что набор элементов конечен и поэтому любое дискретное сообщение конечной длины передаст конечное число значений некоторой величины.

Элементы, из которых состоит дискретное сообщение, называют *буквами* или *символами*. Набор этих букв образует *алфавит*. Здесь под буквами в отличие от обычного представления понимаются любые знаки (обычные буквы, цифры, знаки препинания, математические и прочие знаки), используемые для представления дискретных сообщений.

При дискретной форме представления информации отдельным элементам ее могут быть присвоены числовые (цифровые) значения. В таких случаях говорят о цифровой (числовой) информации. Передача и преобразования дискретной информации любой формы (например, обычного текста, содержащего обычные буквы и цифры) могут быть сведены к эквивалентным передаче и преобразованиям цифровой информации (см. гл. 2). Более того, возможно с любой необходимой степенью точности непрерывные сообщения заменять цифровыми путем квантования непрерывного сообщения по уровню и времени (см. § 13.4). Таким образом, любое сообщение может быть представлено в цифровой форме.

Электронные вычислительные машины, или компьютеры, являются преобразователями информации. В них исходные данные задачи преобразуются в результат ее решения. В соответствии с используемой формой представления информации машины делятся на два класса: *непрерывного действия* — *аналоговые* и *дискретного действия* — *цифровые*.

В силу универсальности цифровой формы представления информации цифровые электронные вычислительные машины представляют собой наиболее универсальный тип устройства обработки информации. Именно эти машины, в дальнейшем называемые сокращенно ЭВМ, составляют предмет изучения в настоящей книге.

Замечательные свойства ЭВМ — автоматизация вычислительного процесса на основе программного управления, огромная скорость выполнения арифметических и логических операций, возможность хранения большого количества различных данных, возможность решения широкого круга математических задач и задач обработки данных — делают эти машины мощным средством научно-технического прогресса.

Особое значение ЭВМ состоит в том, что впервые с их появлением человек получил орудие для автоматизации процессов обработки информации. Во многих случаях ЭВМ позво-

ляют существенно повысить эффективность умственного труда.

Внедрение ЭВМ оказало большое влияние на многие области науки и техники, вызвало процесс их математизации и компьютеризации.

1.2. ПРИНЦИП ДЕЙСТВИЯ ЭВМ

Чтобы лучше понять принцип действия ЭВМ, вспомним, как проводятся вычисления с помощью простейшей настольной счетной машины (калькулятора).

Предварительно на листе бумаги выписываются исходные данные, формулы расчета и вычерчивается таблица для занесения промежуточных и конечных результатов. В процессе вычислений с листа бумаги переносятся на регистр счетной машины числа, участвующие в очередной операции, в соответствии с расчетной формулой включается нужная операция, и полученный результат переписывается с регистра счетной машины в таблицу на листе бумаги.

Таким образом, в этом процессе счетная машина выполняет арифметические операции над числами, которые человек в нее вводит. Лист бумаги служит запоминающим устройством, хранящим программу вычислений (расчетные формулы), исходные данные, промежуточные и конечные результаты. Человек управляет процессом вычислений, в том числе переносит информацию с листа бумаги в машину и обратно, заставляет машину выполнять необходимую операцию, а также выбирает нужный вариант продолжения процесса вычислений в соответствии с результатом, полученным на данном этапе счета.

В рассматриваемом процессе практически ничего не меняет переход от механической счетной машины к использованию электронного арифметического устройства, способного с огромной скоростью производить арифметические операции. Человек будет по-прежнему слишком медленно вводить числа в арифметическое устройство, включать нужную операцию, считывать и переносить на бумагу результат операции.

Эффект, и притом принципиальный, будет достигнут, если к электронному быстродействующему арифметическому устройству добавить быстродействующую память, которая, как лист бумаги при расчете, хранит программу вычислений, исходные данные, промежуточные и конечные результаты, а также быстродействующее управляющее устройство, производящее необходимый для реализации программы вычислений обмен числами между памятью и арифметическим устройством и включающее последнее на выполнение нужной операции.

Если еще позаботиться о том, чтобы комплекс нашей аппаратуры имел средства для связи с внешним миром, т. е. устройство для ввода в память данных и программы вычислений, а также устройство вывода результатов вычислений, то придем к классической структурной схеме ЭВМ, изображенной на рис. 1.1.

ЭВМ содержит следующие основные устройства: арифметическо-логическое устройство, память, устройство управления, устройства ввода данных в машину и вывода из нее результатов расчета и пульт ручного управления.

Арифметическо-логическое устройство (АЛУ) производит арифметические и логические преобразования над поступающими в него машинными словами, т. е. кодами определенной длины, представляющими собой числа или другой вид информации.

Память хранит информацию, передаваемую из других устройств, в том числе поступающую в машину извне через устройство ввода, и выдает во все другие устройства информацию, необходимую для протекания вычислительного процесса. Память машины в большинстве случаев состоит из двух существенно отличающихся по своим характеристикам частей: быстродействующей *основной* или *оперативной* (внутренней) *памяти* (ОП) и сравнительно медленно действующей, но способной хранить значительно больший объем информации *внешней памяти* (ВП).

Оперативная память содержит некоторое число ячеек, каждая из которых служит для хранения машинного слова. Ячейки нумеруются, номер ячейки называется *адресом*.

В *запоминающих устройствах*, реализующих в ЭВМ функцию памяти, выполняются операции считывания хранимой

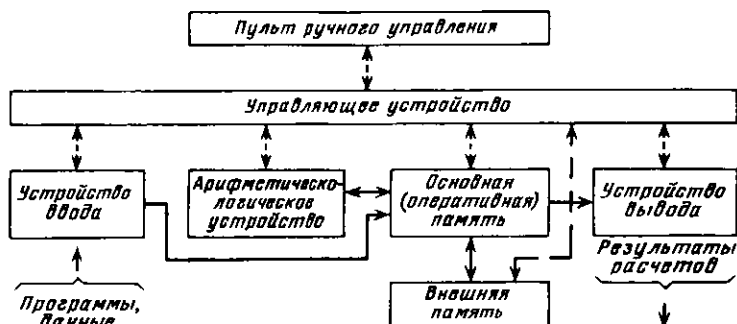


Рис. 1.1. Структура электронной цифровой вычислительной машины

информации для передачи в другие устройства и записи информации, поступающей из других устройств. При считывании слова из ячейки содержимое последней не меняется и при необходимости слово может быть снова взято из той же ячейки. При записи хранившееся в ячейке слово стирается и его место занимает новое.

Непосредственно в вычислительном процессе участвует только ОП, и лишь после окончания отдельных этапов вычислений из ВП в ОП передается информация, необходимая для следующего этапа решения задачи.

Управляющее устройство (УУ) автоматически без участия человека управляет вычислительным процессом, посылая всем другим устройствам сигналы, предписывающие им те или иные действия. В частности, УУ указывает ОП, какие слова должны быть переданы в АЛУ и в другие устройства, включает АЛУ на выполнение нужной операции и помещает полученный результат в ОП.

Алгоритмом решения задачи численным методом называют последовательность арифметических и логических операций, которые надо произвести над исходными данными и промежуточными результатами для получения решения задачи. Поэтому алгоритм можно задать указанием, какие следует произвести операции, в каком порядке и над какими словами. Описание алгоритма в форме, воспринимаемой ЭВМ, называется *программой*.

Программа состоит из отдельных *команд*. Каждая команда предписывает определенное действие и указывает, над какими словами (операндами) это действие производится. Программа представляет собой совокупность команд, записанных в определенной последовательности, обеспечивающей решение задачи на ЭВМ.

Пусть, например, нужно вычислить

$$F = \frac{a - x}{ax + c} \quad (1.1)$$

при заданных численных значениях a , c , x . Программу вычислений F можно представить следующей последовательностью команд:

- 1) вычесть из числа a число x ;
- 2) умножить число a на число x ;
- 3) прибавить к результату действия второй команды число c ;
- 4) разделить результат действия первой команды на результат действия третьей команды.

Чтобы устройство управления могло воспринять команды, они должны быть закодированы в цифровой форме.

Автоматическое управление процессом решения задачи достигается на основе *принципа программного управления*, являющегося основной особенностью ЭВМ.

Другим важнейшим принципом является *принцип хранимой в памяти программы*. Согласно этому принципу программа, закодированная в цифровом виде, хранится в памяти наравне с числами. В команде указываются не сами участвующие в операциях числа, а адреса ячеек ОП, в которых они находятся, и адрес ячейки, куда помещается результат операции.

Поскольку программа хранится в памяти, одни и те же команды могут нужное число раз извлекаться из памяти и выполняться. Более того, так как команды представляются в машине в форме чисел, то над командами как над числами машина может производить операции («модификация команд»).

Использование электронных схем, принципов программного управления и хранимой в памяти программы позволило достигнуть высокого быстродействия и сократить во много раз число команд в программах решения задач, содержащих вычислительные циклы, по сравнению с числом операций, которые производит машина при выполнении этих программ.

Команды выполняются в порядке, соответствующем их расположению в последовательных ячейках памяти, кроме команд безусловного и условного перехода, изменяющих этот порядок соответственно безусловно или только при выполнении некоторого условия, обычно задаваемого в виде равенства нулю, положительного или отрицательного результата предыдущей команды или отношения типа $>$, $=$, $<$ для указываемых командой чисел. Именно благодаря наличию команд условного перехода ЭВМ может автоматически изменять соответствующим образом ход вычислительного процесса, решать сложные логические задачи.

Перед решением задачи на ЭВМ программа и исходные данные должны быть помещены в ее память. Предварительно эта информация наносится на перфоленту, перфокарты в виде соответствующих комбинаций отверстий на них или на магнитную ленту путем соответствующего намагничивания участков ее поверхности. Затем при помощи *устройства ввода* программа и исходные данные считываются с перфокарт, перфоленты или магнитной ленты и переносятся в ОП. Информация, необходимая для решения задачи, может вводиться в ОП непосредственно с клавиатуры дисплея или электрифицированной пишущей машины.

Устройство вывода служит для выдачи из машины результатов расчета, например, путем печатания их на электрифицированных печатных устройствах или отображения на экране дисплея.

При помощи *пульта управления* оператор пускает и останавливает машину, а при необходимости может вмешиваться в процесс решения задачи.

ЭВМ обладают универсальностью, они пригодны для решения разнообразных задач. Любая необходимая точность вычислений может быть достигнута путем увеличения числа разрядов в представлении чисел в ЭВМ и соответствующего увеличения количества электронного оборудования без повышения требований к точности работы самих электронных схем.

1.3. ПОНЯТИЕ О СИСТЕМЕ ПРОГРАММНОГО (МАТЕМАТИЧЕСКОГО) ОБЕСПЕЧЕНИЯ ЭВМ

Каждая ЭВМ обладает определенными свойствами, такими, например, как возможность обрабатывать информацию той или иной формы (цифровую, алфавитно-цифровую, представленную словами постоянной или переменной длины и т. п.), возможность производить определенные арифметические и логические преобразования, операции, связанные с организацией совместной работы устройств машины, и др.

Для придания ЭВМ определенных свойств используют средства двух видов: аппаратные и программные. Последние называются также средствами *программного (математического) обеспечения*¹.

Часть свойств ЭВМ приобретает благодаря наличию в ее составе электронного или электромеханического оборудования, специально предназначенного для реализации этих свойств. Арифметическо-логическое устройство машины является примером аппаратных средств.

Ряд других свойств реализуется без специальных аппаратных средств программным путем, при этом используются имеющиеся аппаратные средства машины, работающие в предписанном порядке в соответствии с программой, обеспечивающей выполнение машиной данной функции. Машина может не иметь, например, аппаратно реализованной операции извле-

¹ В зарубежной литературе терминам «аппаратные средства» и «средства математического обеспечения» соответствуют термины *hardware* и *software*, что буквально означает «жесткий товар» и «мягкий товар».

чения квадратного корня. Однако если имеется программа извлечения квадратного корня, использующая наличные аппаратные средства машины, то с точки зрения пользователя машина обладает свойством извлекать квадратный корень.

При помощи аппаратных средств соответствующие функции выполняются значительно быстрее, чем программным путем. Поэтому одна и та же функция может реализоваться в малых моделях ЭВМ при помощи программных средств, а в больших, для которых быстродействие является одной из важнейших характеристик, — аппаратными средствами.

Часто для придания ЭВМ того или иного свойства используют комбинацию аппаратных и программных средств, что позволяет при сравнительно небольших аппаратных затратах достигнуть высокой эффективности и быстродействия при выполнении соответствующей функции.

Средства программного обеспечения и аппаратные средства являются двумя взаимосвязанными компонентами современной вычислительной техники.

С точки зрения пользователя несущественно, выполняется та или иная функция аппаратными или программными средствами. Поэтому можно говорить о виртуальной (кажущейся) ЭВМ, обладающей некоторыми свойствами, реализуемыми совокупностью аппаратных и программных средств [21].

Система программного (математического) обеспечения ЭВМ представляет собой комплекс программных средств, в котором можно выделить операционную систему, комплект программ технического обслуживания и пакеты прикладных программ (рис. 1.2).

Операционные системы являются важнейшей и центральной частью программного обеспечения ЭВМ, предназначенной для эффективного управления вычислительным процессом, планирования работы и распределения ресурсов ЭВМ, автоматизации процесса подготовки программ и организации их выполнения при различных режимах работы машины, облегчения общения оператора с машиной.

Пользователи и операторы не имеют прямого доступа к устройствам ЭВМ. Связь пользователей и операторов с ЭВМ (точнее, с ее аппаратными средствами) производится при помощи операционной системы, обеспечивающей определенный уровень общения человека с машиной.

Уровень общения в первую очередь определяется уровнем языка, на котором оно происходит. Современные операционные системы содержат трансляторы с языков различного уровня, таких, как АССЕМБЛЕР, ФОРТРАН, ПЛ-1 и другие.

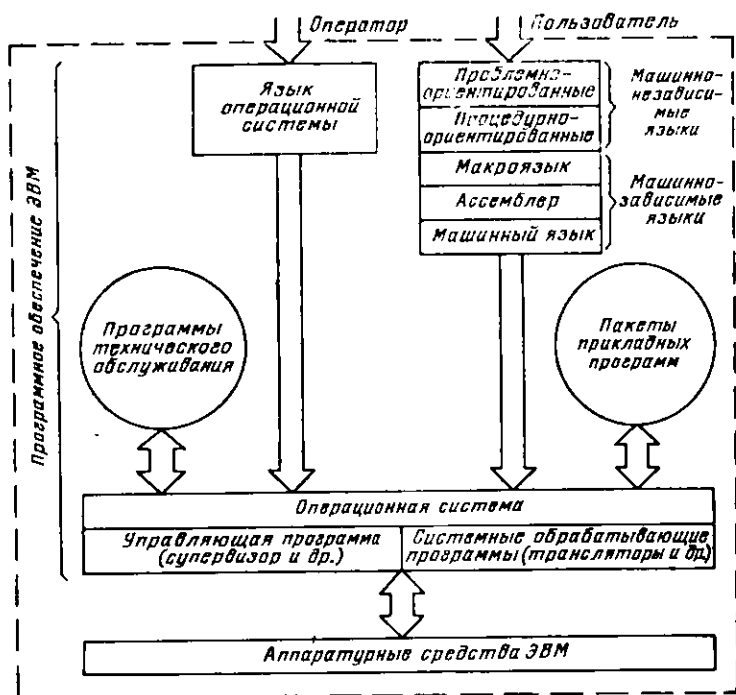


Рис. 1.2. Вычислительная система – совокупность аппаратных и программных средств ЭВМ

Комплект программ технического обслуживания, предназначенный для уменьшения трудоемкости эксплуатации ЭВМ, содержит программы проверки работоспособности ЭВМ и отдельных ее устройств, определения (диагностирования) мест неисправностей.

Пакеты прикладных программ (ППП) представляют собой структурированные комплексы программ (часто со специализированными языковыми средствами), предназначенные для решения определенных достаточно широких классов задач (научно-технических, планово-экономических и др.), а также для расширения функций операционных систем (управление базами данных, реализация режимов телеобработки данных, реального времени и др.).

Аппаратурные средства ЭВМ и система ее программного обеспечения в совокупности образуют вычислительную систему.

При создании новой ЭВМ разработка аппаратуры и разработка программного обеспечения должны производиться одновременно и взаимосвязанно.

В настоящее время средства программного обеспечения должны рассматриваться как вид промышленной продукции отрасли вычислительного машиностроения – такой же, как и сами ЭВМ. В соответствии с этим должны применяться индустриальные методы при разработке, испытаниях, проверке, размножении средств программного обеспечения. Вычислительные машины должны поставляться комплектно с системами программного обеспечения, представленными в такой форме и на таких носителях информации, чтобы использование этих средств не вызывало затруднений.

Сложность современных вычислительных машин закономерно привела к понятию *архитектуры вычислительной машины*, охватывающей комплекс общих вопросов ее построения, существенных в первую очередь для пользователя, интересующегося главным образом возможностями машины, а не деталями ее технического исполнения.

Круг вопросов, подлежащих решению при разработке архитектуры ЭВМ, можно условно разделить на вопросы общей структуры, организации вычислительного процесса и общения пользователя с машиной, вопросы логической организации представления, хранения и преобразования информации и вопросы логической организации совместной работы различных устройств, а также аппаратурных и программных средств машины.

1.4. ПОКОЛЕНИЯ ЭВМ

В середине нашего века развитие атомной физики, ракетной и космической техники потребовало решения вычислительных задач такого большого объема, что с ними нельзя было справиться при помощи имевшихся в то время клавишных или перфорационных счетных машин. Эта потребность привела к созданию на рубеже 40–50-х гг. цифровых электронных вычислительных машин.

Идея использования программного управления для построения устройства, автоматически выполняющего арифметические вычисления, была впервые высказана английским математиком Ч. Бэббиджем еще в 1833 г. Однако его попытки построить механическое вычислительное устройство с программным управлением не увенчались успехом.

Фактически эта идея была реализована лишь спустя 100 лет, когда в 1942 г. К. Цюзе в Германии и в 1944 г. Г. Айкен в США

построили на электромагнитных реле вычислительные машины с управлением от перфоленты [123].

Идея программного управления вычислительным процессом была существенно развита американским математиком Дж. фон Нейманом, который в 1945 г. сформулировал принцип хранимой в памяти программы.

Первые ЭВМ с программным управлением и с хранимой в памяти программой появились практически одновременно в Англии, США и СССР.

Фундаментальный вклад в развитие отечественной вычислительной техники внес акад. С. А. Лебедев. Под его руководством в 1949–1951 гг. в АН УССР в Киеве была построена первая в нашей стране ЭВМ – Малая Электронная Счетная Машина (МЭСМ), а в 1952–1954 гг. в ИТМ и ВТ АН СССР – Быстродействующая Электронная Счетная Машина (БЭСМ), выполнявшая 8000 операций/с и являвшаяся в то время одной из самых быстродействующих ЭВМ в мире. Последующие годы акад. С. А. Лебедев посвятил созданию оригинальных советских ЭВМ высокой производительности.

Одну из первых в стране ЭВМ построили в начале 50-х годов чл.-кор. АН СССР И. С. Брук и его сотрудники Н. Я. Матюхин и М. А. Карцев в Энергетическом институте АН СССР в Москве. Первая выпускавшаяся промышленностью ЭВМ «Стрела» была разработана научным коллективом, руководимым Ю. Я. Базилевским.

Советские ученые, в первую очередь академики С. А. Лебедев, М. В. Келдыш, В. М. Глушков, В. С. Семенихин, и их научные школы внесли крупный вклад в развитие принципов построения и теории ЭВМ и их программного обеспечения, методов использования ЭВМ в народном хозяйстве.

Проблемам развития электронной вычислительной техники большое внимание уделяет ЦК КПСС и Советское правительство. В короткий срок в стране были созданы крупные научно-конструкторские центры по разработке и мощные заводы по производству ЭВМ. Неоднократно вопросы развития техники ЭВМ и расширения ее использования в народном хозяйстве находили отражение в решениях и материалах партийных съездов и пленумов ЦК КПСС.

На протяжении более трех десятилетий электронная вычислительная техника бурно развивается. На наших глазах появились, сменяя друг друга, несколько поколений ЭВМ. Появление новых поколений ЭВМ вызывалось расширением областей и развитием методов их применения, требовавших более производительных, более дешевых и более надежных машин.

Поколение ЭВМ определяется совокупностью взаимосвязанных и взаимообусловленных существенных особенностей и характеристик используемой при построении машин конструктивно-технологической (в первую очередь элементной) базы и реализуемой в машине архитектуры (логической организации).

Первое поколение образовали ламповые ЭВМ, промышленный выпуск которых начался в начале 50-х гг. В качестве компонентов логических элементов использовались электронные лампы.

К первому поколению ЭВМ относятся созданные советскими учеными и инженерами ламповые вычислительные машины БЭСМ-2, «Стрела», М-2, М-3, «Минск-1», «Урал-1», «Урал-2», М-20, в основном ориентированные на решение научно-технических задач.

Ламповые ЭВМ потребляли большую мощность, имели большие габаритные размеры, малую емкость оперативной памяти и, что особенно важно, невысокую надежность, в первую очередь из-за частого выхода из строя электронных ламп.

В вычислительных машинах второго поколения, появившихся в конце 50-х гг., полупроводниковые приборы — транзисторы — заменили электронные лампы, что существенно повысило надежность, снизило потребление мощности, уменьшило размеры ЭВМ.

Это позволило создать ЭВМ, обладающие большими логическими возможностями и более высокой производительностью. Наряду с машинами для научных расчетов появились ЭВМ для решения планово-экономических задач (задач обработки данных) и управления производственными процессами.

В нашей стране были созданы полупроводниковые ЭВМ различного назначения: малые ЭВМ серий «Наири» и «Мир», средние ЭВМ для научных расчетов и обработки данных со скоростью работы 5—30 тыс. операций/с — «Минск-2», «Минск-22», «Минск-32», «Урал-14», «Раздан-2», «Раздан-3», БЭСМ-4, М-220 и управляющие вычислительные машины «Днепр», ВНИИЭМ-3 и др.

В рамках второго поколения академики С. А. Лебедев и В. А. Мельников создали сверхбыстродействующую ЭВМ БЭСМ-6 с производительностью 1 млн. операций/с.

Второе поколение ЭВМ позволило существенно расширить сферу использования вычислительной техники, приступить к созданию АСУ отраслями, предприятиями и технологическими процессами.

Стремление к повышению надежности, быстродействия, снижению стоимости аппаратуры привело к появлению новой

элементной базы вычислительной техники в виде интегральных микросхем, на основе которых были созданы ЭВМ третьего поколения.

ЭВМ третьего поколения появились во второй половине 60-х гг., когда фирма ИБМ (США) разработала систему машин ИБМ-360. Эта система оказала влияние на логическую организацию машин третьего поколения, разработанных в других странах.

Советский Союз и другие социалистические страны — члены СЭВ, реализуя программу социалистической интеграции, в начале 70-х годов совместно разработали и организовали серийное производство Единой Системы ЭВМ (ЕС ЭВМ) и Системы Малых ЭВМ (СМ ЭВМ) — машин третьего поколения на интегральных микросхемах.

В машинах третьего поколения значительное внимание уделено уменьшению трудоемкости подготовки программ для решения задач на ЭВМ, облегчению связи оператора с машиной, повышению эффективности использования дорогостоящего оборудования машин, облегчению эксплуатационного обслуживания ЭВМ, что достигается при помощи соответствующих операционных систем.

В последние годы появились ЭВМ и вычислительные устройства, которые следует отнести к четвертому поколению. Контуры этого поколения довольно трудно четко определить, так как в настоящее время оно представлено главным образом новыми, ранее не существовавшими типами вычислительных средств и лишь в отношении некоторых вопросов (например, замены ферритовых памяти полупроводниковыми) затронуло машины общего назначения, выполняющие основной объем вычислительных работ в разного рода вычислительных центрах.

Конструктивно-технологической основой ЭВМ четвертого поколения являются большие (БИС) и сверхбольшие (СБИС) интегральные микросхемы, содержащие тысячи и десятки тысяч транзисторов на одном кристалле. В первую очередь на БИС строят памяти ЭВМ.

К четвертому поколению относятся реализованные на СБИС такие новые средства вычислительной техники, как микропроцессоры и создаваемые на их основе микро-ЭВМ. Микропроцессоры и микро-ЭВМ нашли широкое применение в устройствах и системах автоматизации измерений, обработки данных и управления технологическими процессами, при построении различных специализированных цифровых устройств и машин.

Вычислительные возможности микро-ЭВМ оказались доста-

точными для создания на их основе в рамках ЭВМ четвертого поколения, нового по ряду эксплуатационных характеристик и способу использования типа вычислительных устройств, — *персональных ЭВМ (персональных компьютеров)*, получивших в настоящее время широкое распространение.

В ЭВМ четвертого поколения достигается дальнейшее упрощение контактов человека с ЭВМ путем повышения уровня машинного языка, значительного расширения благодаря применению микропроцессоров функций устройств (терминалов), используемых человеком для связи с ЭВМ, начинается практическая реализация голосовой связи с ЭВМ. Использование БИС позволяет аппаратурными средствами реализовывать некоторые функции программ операционных систем (аппаратурная реализация трансляторов с алгоритмических языков высокого уровня и др.), что способствует увеличению производительности машин.

Характерным для крупных ЭВМ четвертого поколения является наличие нескольких процессоров, ориентированных на выполнение определенных операций, процедур или на решение некоторых классов задач. В рамках этого поколения создаются многопроцессорные вычислительные системы с быстродействием в несколько десятков или сотен миллионов операций в секунду и многопроцессорные управляющие комплексы повышенной надежности с автоматическим изменением структуры (автоматической реконфигурацией).

Примером крупных вычислительных систем, которые следует отнести к четвертому поколению, является многопроцессорный комплекс «Эльбрус-2» с суммарным быстродействием до 100 млн. операций/с, с системой команд, приближенной к языкам высокого уровня, стековой организацией обращений к памяти.

К четвертому поколению также относится и многопроцессорная вычислительная система ПС-2000, содержащая до 64 процессоров, управляемых общим потоком команд, в которой при распараллеливании процесса выполнения программ может быть достигнуто суммарное быстродействие до 200 млн. операций/с (см. гл. 15).

В последнее время определились контуры нового, пятого поколения ЭВМ. В значительной степени этому способствовали публикации сведений о проекте ЭВМ пятого поколения, разрабатываемом ведущими японскими фирмами и научными организациями, поставившими перед собой цель захвата в 90-х годах японской промышленностью мирового лидерства в области вычислительной техники. Поэтому этот проект часто называют «японским вызовом» [55].

Согласно этому проекту ЭВМ и вычислительные системы пятого поколения помимо более высокой производительности и надежности при более низкой стоимости должны обладать следующими качественно новыми свойствами: возможностью взаимодействия с ЭВМ при помощи естественного языка, человеческой речи и графических изображений; способностью системы обучаться, производить ассоциативную обработку информации, делать логические суждения, вести «разумную» беседу с человеком в форме вопросов и ответов; способностью системы «понимать» содержимое базы данных, которая при этом превращается в «базу знаний», и использовать эти «знания» при решении задач.

Предполагается, что в ЭВМ пятого поколения быстродействие машин и емкость основной памяти составят соответственно 2 млн. операций/с и 0,5–5 Мбайт для персональных компьютеров и 1–100 млрд. операций/с и 8–160 Мбайт для сверхпроизводительных ЭВМ.

Ожидается, что в машинах пятого поколения будут использоваться сверхбольшие интегральные микросхемы, содержащие до 1–10 млн. транзисторов на кристалл. Для разработки таких схем потребуются мощные системы автоматизации проектирования.

1.5. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ЭВМ

Важнейшими эксплуатационными характеристиками ЭВМ являются ее *производительность P* и *общий коэффициент эффективности машины*:

$$\mathcal{E} = P / (C_{\text{ЭВМ}} + C_{\text{экс}}), \quad (1.2)$$

представляющий собой отношение ее производительности к сумме стоимости самой машины $C_{\text{ЭВМ}}$ и затрат на ее эксплуатацию за определенный период времени (например, период окупаемости капитальных затрат) $C_{\text{экс}}$.

Так как часто бывает трудно оценить затраты на эксплуатацию данной модели ЭВМ, то оценивают эффективность машины по упрощенной формуле

$$\mathcal{E} = P / C_{\text{ЭВМ}}. \quad (1.3)$$

Оценка и сопоставление производительности различных ЭВМ представляют собой сложную проблему, в сущности не получившую до сих пор удовлетворительного решения.

Под производительностью ЭВМ можно было бы понимать количество «вычислительной работы», или, другими словами,

количество задач, выполняемых машиной в единицу времени. Однако на производительность ЭВМ, оцениваемую по числу решенных в единицу времени задач, влияет слишком много факторов, в том числе тип задач, число тех или иных операций, выполняемых при решении задач, стиль программирования и другие особенности программ, логические возможности системы команд, особенности операционной системы, структура процессора, характеристики и организация оперативной и внешней памяти, системы ввода-вывода, состав и характеристики периферийных устройств и др.

В этих условиях оценка производительности по некоторому набору задач применяется только тогда, когда вычислительная установка предназначена для решения достаточно узкого круга задач.

В настоящее время производительность ЭВМ общего назначения оценивается упрощенно по скорости выполнения некоторых смесей команд, формируемых путем анализа частоты исполнения разного вида команд при выполнении программ решения задач некоторого достаточно широкого класса, например научно-технических задач или задач обработки данных. На основе такого анализа отдельным видам команд (операций) присваиваются определенные весовые коэффициенты.

Весовые коэффициенты отдельных видов команд для смеси команд, служащей для оценки производительности ЭВМ при решении научно-технических задач («смесь Гибсона»), приведены ниже:

Вид команды (операции)	Весовой коэффициент k_s
Сложение, вычитание с фиксированной запятой	33
Умножение с фиксированной запятой	0,6
Деление с фиксированной запятой	0,2
Сложение, вычитание с плавающей запятой	7,3
Умножение с плавающей запятой	4,0
Деление с плавающей запятой	1,6
Логические операции	1,7
Безусловные передачи управления	17,5
Условные передачи управления	6,5

При использовании смесей команд производительность ЭВМ определяется по формуле

$$P = \sum_{s=1}^h k_s / \sum_{s=1}^h k_s t_s, \quad (1.4)$$

где k_s и t_s — соответственно весовой коэффициент и продолжи-

тельность выполнения z -й команды; h — число различных команд в смеси.

К более частным характеристикам ЭВМ относятся:

число разрядов в машинном слове (влияет на точность вычислений);

скорость выполнения основных видов команд;

емкость оперативной памяти;

максимальная скорость передачи информации между ядром ЭВМ (процессор и оперативная память) и внешним (периферийным) оборудованием.

Однако все приведенные выше характеристики представляют собой лишь номинальные показатели. Они реализуются, если ЭВМ полностью работоспособна. Суждение о реальных возможностях выполнения машиной определенных функций, решения определенного комплекса задач, особенно задач управления, может быть вынесено только с учетом эксплуатационной надежности машины. Эксплуатационная надежность непосредственно влияет и на стоимость эксплуатационного обслуживания ЭВМ.

Надежность ЭВМ определяется частотой нарушения ее работоспособности из-за отказов и сбоев, затратами времени на их устранение (подробнее см. гл. 12).

Пусть K_n — комплексный коэффициент эксплуатационной надежности (комплексный коэффициент использования) машины, характеризующий потерю производительности из-за нарушения надежности функционирования ЭВМ. Тогда с учетом надежности машины формулы (1.2) и (1.3) могут вид

$$\mathcal{E} = PK_n / (C_{ЭВМ} + C_{экс}); \quad (1.5)$$

$$\mathcal{E}' = PK_n / C_{ЭВМ}. \quad (1.6)$$

Повышение надежности вычислительной машины, оцениваемое приращением ΔK_n , связано с дополнительными затратами $\Delta C_{ЭВМ}$, эффективность которых можно оценить отношением

$$\mathcal{E}_n = \Delta K_n / \Delta C_{ЭВМ}. \quad (1.7)$$

При создании новых ЭВМ должно обеспечиваться значительное возрастание отношений производительность/стоимость и надежность/стоимость.

1.6. ОСНОВНЫЕ ОБЛАСТИ ПРИМЕНЕНИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И ОСНОВНЫЕ ТИПЫ ЭВМ

Успехи в развитии вычислительной техники, в основе которых лежат достижения в области микроэлектроники, архитектуры вычислительных машин, их программного обеспечения, привели к значительному расширению сферы применения ЭВМ.

Развитие вычислительной техники и сферы и методов ее использования — процессы взаимосвязанные и взаимообусловленные. С одной стороны, потребности народного хозяйства, науки и культуры стимулируют поиски учеными и конструкторами новых путей построения ЭВМ, а с другой стороны, появление электронных вычислительных машин, систем и устройств с большими функциональными возможностями, с существенно улучшенными показателями по производительности, стоимости, габаритным размерам, надежности и т. п. создает предпосылки для непрерывного расширения областей и развития форм применения ЭВМ.

Первоначально сравнительно узкая сфера применения ЭВМ, главным образом для научных и технических расчетов, в короткий срок существенно расширилась и охватила почти все области науки, техники, планирования и управления производством, контроля и управления технологическими процессами, все области человеческой деятельности, связанные с обработкой больших объемов информации.

Разнообразие областей и форм использования ЭВМ породило широкий спектр требований к характеристикам и особенностям организации машин и систем. В результате к настоящему времени в соответствии с областями применения определились основные типы ЭВМ, которые при сохранении указанных в § 1.2 основных фундаментальных принципов существенно разнятся не только по количественным характеристикам, но и по архитектуре, электронно-технологической базе и используемым периферийным устройствам.

Основные средства современной вычислительной техники можно классифицировать следующим образом: сверхпроизводительные ЭВМ (системы), ЭВМ общего назначения, малые ЭВМ, микро-ЭВМ, микропроцессоры.

На рис. 1.3 типы машин (вычислительных средств) соотнесены с основными областями применения ЭВМ. Приведенное разделение весьма условно, границы между типами машин быстро меняются под влиянием успехов в области микроэлектроники и архитектуры ЭВМ, тем более что в ряде применений

Области применения ЭВМ

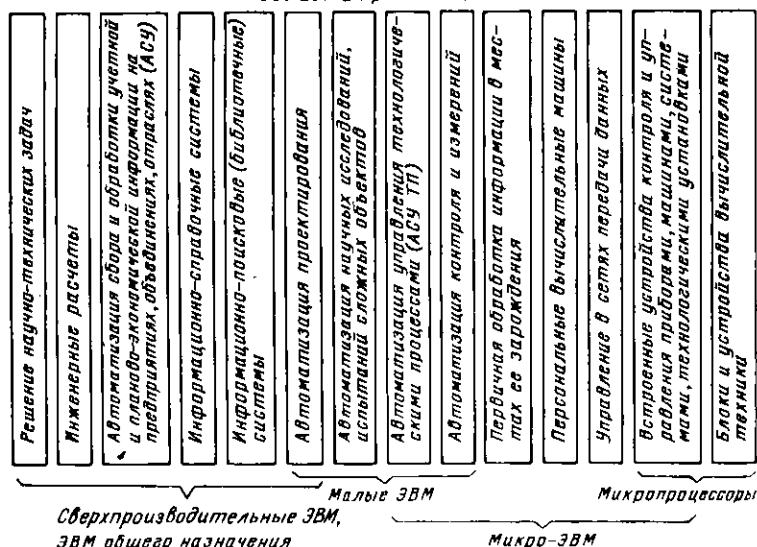


Рис. 1.3. Области применения и основные типы ЭВМ

машины разных типов объединяются в вычислительные системы и комплексы различных конфигураций.

К сверхпроизводительным машинам (системам) в настоящее время относят машины (системы), выполняющие свыше 10 млн. операций/с. Подобные машины используются для решения особенно сложных научно-технических задач, задач обработки больших объемов данных в реальном времени, поиска оптимальных решений в задачах экономического планирования и при автоматизированном проектировании. Принципы организации сверхпроизводительных вычислительных систем будут рассмотрены в гл. 15.

1.6.1. ЭВМ ОБЩЕГО НАЗНАЧЕНИЯ

Первые ЭВМ были созданы для выполнения научных и технических расчетов, для которых типичными являются возможность работы со словами фиксированной длины, относительно небольшие объемы входной (исходных данных) и выходной (результатов расчета) информации и очень большое количество вычислений, которые необходимо проделать в процессе решения задачи.

Совсем иной характер носят задачи планово-экономические, учета, статистики и т. п. Эти задачи связаны с вводом в ма-

шину и запоминанием очень большого количества исходных данных. Сама обработка данных требует выполнения сравнительно небольшого числа логических и арифметических операций. По окончании обработки выводится и печатается большое количество информации, причем результаты обработки должны печататься в отредактированной форме в виде таблиц, ведомостей и т. д.

Задачи подобного типа получили название задач обработки данных, а ЭВМ, предназначенные для их решения, часто называют *системами автоматической обработки данных*. Системы автоматической обработки данных составляют основу автоматизированных систем управления (АСУ).

Для систем обработки данных важно, чтобы имелась возможность ввода в ЭВМ, хранения, обработки и вывода из машины текстовой, т. е. алфавитно-цифровой, информации, которая представляется словами переменной длины.

Для систем обработки данных характерно наличие большого количества внешних, или *периферийных, устройств*, таких, как запоминающие устройства на магнитных дисках и лентах, способных хранить очень большой объем информации (миллионы чисел и других данных), и устройств, осуществляющих ввод и вывод данных, их регистрацию и отображение.

Лет 20 назад считалось, что ЭВМ для обработки данных существенно отличаются от машин, решающих научные и технические задачи, для которых быстродействие является важнейшей характеристикой. Современное представление о методах использования ЭВМ для научно-технических расчетов предполагает применение для программирования алгоритмических языков, ввод в машину текстов программ на этих языках и их преобразование, вывод результатов расчетов в виде оформленных таблиц с текстовыми надписями, хранение в машине больших программных массивов, в том числе разнообразных пакетов прикладных программ. В результате произошло стирание различий в структуре и способах представления информации между этими двумя типами машин — появились *ЭВМ общего назначения*, которые в настоящее время выполняют основной объем вычислительных работ и машинной обработки информации в различных ВЦ и АСУ.

Современные ЭВМ общего назначения универсальны, они могут использоваться как для решения научно-технических задач численными методами, так и в режиме автоматической обработки данных в АСУ. Такие машины имеют высокое быстродействие, память большого объема, гибкую систему команд, широкий набор периферийных устройств и способ кодирования информации, учитывающий требования обработки

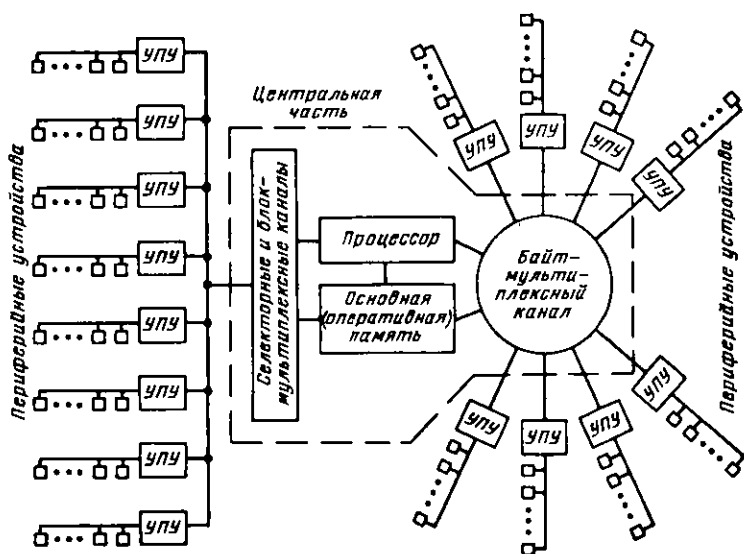


Рис. 1.4. Структура современной ЭВМ общего назначения (ЕС ЭВМ)

данных. На рис. 1.4 представлена типичная структура современной ЭВМ общего назначения. Такую структуру, например, имеют машины ЕС ЭВМ.

Собственно обработка данных производится электронным процессором, содержащим арифметическо-логическое и управляющее устройства. В ЭВМ возникает проблема организации взаимодействия быстродействующего процессора с большим числом сравнительно медленно действующих периферийных устройств (ПУ).

Для эффективного использования технических средств необходима параллельная работа во времени процессора и периферийных устройств. Такой режим в машинах общего назначения организуется при помощи специализированных процессоров ввода-вывода (каналов ввода-вывода) информации. Периферийные устройства связываются с каналами через собственные блоки управления (УПУ) и систему сопряжения, называемую интерфейсом.

Байтмультиплексный канал может одновременно обслуживать несколько сравнительно медленно действующих периферийных устройств – печатающих, ввода информации с перфолент и перфокарт, вывода информации на перфоленты и перфокарты, дисплеев и др. Селекторные и блок-мульти-

плексные каналы связывают процессор и оперативную память с периферийными устройствами, работающими с высокой скоростью передачи информации (магнитные диски, ленты и др.).

Приведем некоторые данные об основных характеристиках ЭВМ общего назначения. Длина слова 32 разряда. Скорость вычислений в разных моделях машин колеблется от нескольких десятков до нескольких сотен тысяч или миллионов арифметических операций в секунду, время обращения к памяти составляет примерно 1 мкс, емкость оперативной памяти — несколько сотен тысяч или миллионов машинных слов, скорость обмена информацией процессора и памяти с периферийными устройствами может достигать нескольких сотен тысяч слов в секунду.

1.6.2. МАЛЫЕ ЭВМ

Имеется большое число, условно говоря, «малых» применений вычислительных машин, таких, как автоматизация производственного контроля изделий, обработка данных при экспериментах, прием и обработка данных с линий связи, управление технологическими процессами, управление станками и разнообразными цифровыми терминалами (расчерчивающими устройствами и др.), малые расчетные инженерные задачи и т. д. Для этих применений ЭВМ общего назначения слишком велики и дороги.

Возникла необходимость в небольших, простых, надежных и, главное, дешевых ЭВМ, в которых обеспечиваются простота программирования и наглядность системы программного обеспечения в отличие от сложности современных операционных систем ЭВМ общего назначения, а также сравнительная простота эксплуатационного обслуживания.

Развитие технологии интегральных электронных схем позволило создать машины, удовлетворяющие указанным выше требованиям. Уменьшение объема аппаратуры и стоимости машин достигнуто, в первую очередь, за счет короткого машинного слова (12—16 разрядов вместо 32—64 в машинах общего назначения), уменьшения по сравнению с ЭВМ общего назначения количества типов обрабатываемых данных (в некоторых моделях только целые числа без знака), ограниченного набора команд, сравнительно небольшого объема оперативной памяти и небольшого набора периферийных устройств. Подобные машины за свои небольшие размеры получили название *малых* или *мини-ЭВМ*.

Для преодоления трудностей, возникающих при проектировании малых ЭВМ из-за короткого машинного слова, предло-

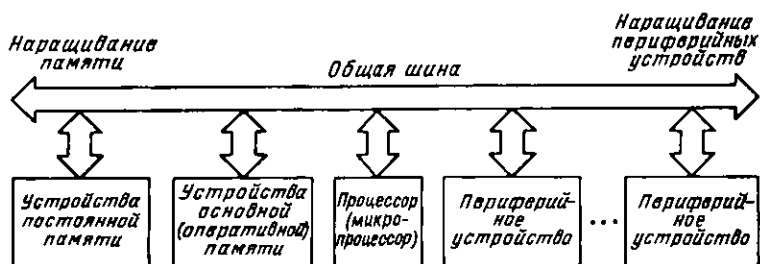


Рис. 1.5. Упрощенная структура малой и микро-ЭВМ

жен ряд решений по представлению данных, адресации, составу и структуре команд, логической структуре процессора, организации обмена информацией между устройствами ЭВМ.

Первые модели малых ЭВМ имели длину слова 12 разрядов. Впоследствии достижения интегральной микроэлектроники позволили перейти в малых машинах к слову длиной 16 разрядов, что не только повысило точность вычислений и создало возможности для построения более гибкой системы команд, но и обеспечило согласованность форматов данных с ЭВМ общего назначения.

Типичная структура малой ЭВМ показана на рис. 1.5. Малые ЭВМ отличается наряду с другими особенностями более простая, чем у машин общего назначения, весьма гибкая структура, называемая *магистрально-модульной*, основу которой составляет общая магистраль (общая шина), к которой подсоединяются в нужных номенклатуре и количествах все устройства машины, выполняемые в виде конструктивно законченных модулей. Устройства машины обмениваются информацией через общую магистраль (общую шину).

Такая структура оказывается эффективной, а система обмена данными через общую шину — достаточно динамичной лишь при сравнительно небольшом наборе периферийных устройств.

Универсальность применения при ограниченном наборе команд может быть обеспечена лишь при сравнительно высоком быстродействии машины — около 200—800 тыс. операций/с, что превышает скорость работы многих ЭВМ общего назначения. Высокое быстродействие должно позволить малым ЭВМ обслуживать технологические процессы в реальном масштабе времени, а также компенсировать замедление обработки данных, связанное с тем, что многие процедуры обработки при ограниченном объеме аппаратуры, скромном наборе команд

и отсутствии специализации машины приходится реализовать не аппаратными средствами, а соответствующими подпрограммами (например, операции арифметики с плавающей запятой).

1.6.3. МИКРОПРОЦЕССОРЫ И МИКРО-ЭВМ

Положительный опыт разработки и применения малых ЭВМ оказал влияние на направление развития интегральной электроники. При переходе от схем малой и средней степени интеграции к большим и сверхбольшим интегральным микросхемам (БИС и СБИС) возникает проблема их применимости. Большую интегральную микросхему, содержащую тысячи логических элементов, не говоря о СБИС с ее десятками тысяч и более элементов, если это не схема памяти, трудно сделать пригодной для широкого круга потребителей. Первоначально считалось, что на основе автоматизированного проектирования будут выпускаться заказные БИС и СБИС, изготавливаемые по индивидуальным требованиям заказчиков. Однако в дальнейшем оказался возможным другой путь — создание на одной или нескольких БИС или СБИС функционально законченного малоразрядного (первоначально на 2—4, а впоследствии на 8—16 разрядов и более) устройства обработки информации, управляемого хранимой в памяти программой. Это устройство (микросхему или несколько образующих его микросхем) называют *микропроцессором*, так как оно по своим логическим функциям и структуре напоминает упрощенный вариант процессора обычных ЭВМ.

Микропроцессоры по быстродействию и возможностям системы команд приближаются к процессорам малых ЭВМ. Однако из-за ограниченного числа выводов корпуса БИС (от 18 до 42) не удастся реализовать интерфейс микропроцессора с внешним оборудованием с высокой пропускной способностью.

Устройство обработки данных, имеющее в своем составе один или несколько микропроцессоров, БИС постоянной и оперативной памяти, БИС управления вводом и выводом и др., называется микро-ЭВМ. Микро-ЭВМ оснащают необходимыми периферийными устройствами (устройствами ввода-вывода, ЗУ на гибких дисках и др.) (рис. 1.5).

Электронная аппаратура микро-ЭВМ содержит несколько десятков корпусов БИС и СИС, размещаемых на одной или нескольких съемных платах.

В микро-ЭВМ сочетаются высокая скорость выполнения операций в микропроцессоре, повышенная надежность, неболь-

шая стоимость со сравнительно низкой пропускной способностью интерфейса, обусловленной ограничениями на число выводов корпуса БИС микропроцессора. Если по скорости выполнения операций микро-ЭВМ приближаются к современным малым ЭВМ, а по ряду эксплуатационных показателей (размеры, потребляемая мощность, надежность) они их превосходят, то из-за малой пропускной способности интерфейса и связанного с этим малого числа подключаемых ПУ применение микро-ЭВМ в настоящее время ограничивается системами с небольшим количеством источников и потребителей информации.

По этим же причинам затруднено использование микропроцессоров в качестве элементов при построении быстродействующих процессоров и каналов ввода-вывода ЭВМ общего назначения. Однако большие перспективы имеет применение микропроцессоров и микро-ЭВМ в периферийном оборудовании ЭВМ (устройствах управления дисками и лентами, дисплеях и других терминалах), в частности для преобразования форматов данных, контроля, перекодирования, редактирования. При этом расширяются логические функции ПУ, становится возможным создание так называемых *интеллектуальных терминалов*, выполняющих сложные процедуры предварительной обработки информации.

Подготовка и отладка программ для микропроцессорных устройств и микро-ЭВМ затруднены из-за небольшой емкости оперативной памяти, ограниченного набора ПУ. Поэтому для подготовки программ микропроцессоров и микро-ЭВМ широко применяются ЭВМ общего назначения и крупные модели малых ЭВМ («кросс-системы программирования»), а также аппаратные имитаторы микро-ЭВМ. •

Специализированные ЭВМ. До сих пор речь шла об ЭВМ универсальных в том смысле, что в пределах своих вычислительных и логических возможностей они предназначены для использования практически в любых отраслях народного хозяйства.

Однако существуют такие области применения вычислительной техники, где специфические условия работы ЭВМ или специфические требования к ее характеристикам, особенности подлежащего реализации фиксированного набора алгоритмов делают невозможным или неэффективным применение универсальных серийно выпускаемых ЭВМ.

В подобных случаях может оказаться целесообразным создание специализированных ЭВМ и цифровых устройств, конструкция, архитектура и характеристики которых ориентированы на определенную, сравнительно узкую область примене-

ния. Примером являются бортовые ЭВМ аэрокосмических систем управления.

С появлением микропроцессоров облегчилось создание и расширились области эффективного применения специализированных цифровых устройств и машин.

Уникально малые, еще недавно показавшиеся бы фантастическими размеры микропроцессоров и микро-ЭВМ при достаточно больших вычислительных и логических возможностях, их дешевизна и высокая надежность представляют собой огромный качественный скачок в развитии вычислительной техники, расширивший практически беспредельно сферу использования цифровых вычислительных устройств, сделавший возможным создание на основе микропроцессоров и микро-ЭВМ контролирующих, управляющих и обрабатывающих информацию устройств (микроконтроллеров), непосредственно встраиваемых в приборы, машины, оборудование и технологические процессы. Магистрально-модульная структура микропроцессоров и микро-ЭВМ облегчает построение разнообразных управляющих и обрабатывающих данные устройств и систем.

Микропроцессоры и микро-ЭВМ открывают принципиально новые возможности для высокоэффективной автоматизации производственных процессов, научно-исследовательских и проектно-конструкторских работ, обработки информации при планировании и управлении производством на предприятиях во всех отраслях народного хозяйства. Поэтому с полным основанием создание и применение микропроцессоров и микро-ЭВМ оцениваются как одно из важнейших направлений научно-технического прогресса.

1.7. ПОНЯТИЕ О СИСТЕМАХ ЭВМ. ЕДИНАЯ СИСТЕМА ЭВМ (ЕС ЭВМ) И СИСТЕМА МАЛЫХ ЭВМ (СМ ЭВМ) СОЦИАЛИСТИЧЕСКИХ СТРАН

Расширение сферы применения вычислительной техники и особенно ее использование в автоматизированных системах управления для обработки информации в области планирования, экономики, учета привели к включению в состав машины большого комплекса разнообразных периферийных устройств для ввода информации, ее запоминания и хранения, регистрации и отображения. Конкретные применения предъявляют различные требования к составу периферийных устройств, объему оперативной и внешней памяти и т. п.

Это привело к тому, что при проектировании вычислительной техники концепцию вычислительной машины с фиксиро-

ванным составом оборудования (где главное место занимало само устройство обработки информации) сменила концепция *агрегатированной вычислительной машины (системы)* с переменным составом оборудования, который определяется выполняемыми ею функциями. При таком подходе отдельные функциональные устройства выполняются в виде агрегатов (модулей), которые в нужных номенклатуре и количестве объединяются в ЭВМ. Например, необходимый объем оперативной памяти набирается из модулей, имеющих фиксированное число ячеек. Одним из агрегатов такой системы оказывается устройство обработки информации — процессор.

Существенное место в агрегатированных машинах (системах) занимают *унифицированные сопряжения (интерфейсы)*, обеспечивающие обмен информацией между агрегатами (модулями), входящими в состав ЭВМ, и допускающие подключение необходимого состава периферийных устройств.

Однако различные области применения вычислительной техники предъявляют разнообразные требования не только к составу и характеристикам ПУ, к объему оперативной памяти, но и к самим устройствам обработки информации — к процессорам (в отношении длины машинного слова, скорости обработки информации и т. д.). В процессе эксплуатации и развития автоматизированной системы управления некоторым объектом или вычислительного центра, выполняющего научно-технические расчеты, может возникнуть необходимость не только в расширении объема памяти ЭВМ, увеличении количества ПУ, но и в переходе к более производительной ЭВМ. В таком случае важное значение имеет возможность использования на новой ЭВМ ранее созданного пользователем программного обеспечения.

Поэтому естественным развитием концепции агрегатированной вычислительной машины с переменным составом оборудования явилось создание *рядов или систем ЭВМ*, состоящих из информационно- и программно-совместимых машин, обладающих различными характеристиками.

Информационная совместимость ЭВМ предполагает единые способы кодирования информации и форматы данных или хотя бы по меньшей мере одинаковые или кратные длины машинных слов в различных моделях.

Программная совместимость означает, что программы, составленные для одной модели, могут выполняться на других моделях ряда. Практически это достигается единой для всех машин ряда системой команд.

Единая система команд позволяет иметь общие для машин системы ЭВМ операционные системы.

Помимо программной совместимости машины, входящие в систему ЭВМ, должны обладать *аппаратурной совместимостью*, заключающейся в возможности присоединения к любой модели ЭВМ, точнее, к ее ядру, состоящему из процессора и оперативной памяти, любых ПУ, общих для всей системы ЭВМ (для ряда моделей).

Общие форматы данных и команд, единая система команд, общие операционные системы, общая номенклатура ПУ, общие методы технической и математической эксплуатации и обслуживания (эксплуатационная совместимость), единая методика построения технической документации — вот что превращает совокупность отдельных моделей ЭВМ в систему.

Рассмотренные принципы построения систем ЭВМ полностью реализованы в ЕС ЭВМ и в определенной степени в СМ ЭВМ, созданных совместно социалистическими странами, входящими в Совет Экономической Взаимопомощи.

Единая система ЭВМ представляет собой комплекс аппаратных и программных средств машин общего назначения, обеспечивающих выполнение научно-технических и планово-экономических расчетов, а также обработку данных для других применений.

Типичная конфигурация модели ЕС ЭВМ показана на рис. 1.4, а общий вид ЭВМ ЕС-1050 — на рис. 1.6.

Машины ЕС ЭВМ имеют характерные особенности:

- использование интегральных электронных схем и многослойного печатного монтажа;

- программная совместимость основных моделей;

- унифицированная организация операций ввода-вывода с помощью программно-управляемых каналов ввода-вывода и стандартного интерфейса с ПУ;

- развитая система программного обеспечения.

При разработке новых моделей ЕС ЭВМ II очереди преследовалась цель: сохранив основные принципы архитектуры I очереди ЕС ЭВМ, расширить функциональные возможности и улучшить характеристики машин.

Основными отличиями ЭВМ II очереди являются:

- более высокое значение показателей производительность/стоимость и надежность/стоимость за счет улучшения технологии, применения более быстродействующих интегральных микросхем, в том числе схем средней и большой степеней интеграции (последнее — для построения памяти), широкой автоматизации на основе применения ЭВМ процессов проектирования, изготовления и контроля;

- увеличение предельной производительности процессоров до 4–5 млн. операций/с;

Таблица 1.1. Характеристики ЭВМ

Модель ЕС ЭВМ	Производительность, тыс. операций/с		Максимальная емкость ОП, Кбайт	
	I	II	I	II
ЕС-1010 (ВНР)	3		64	
ЕС-1012 (ВНР)	6		128	
ЕС-1015 (ВНР)		12 – 16		160
ЕС-1020 (СССР, НРБ)	10 – 20		256	
ЕС-1021 (ЧССР)	20		564	
ЕС-1022 (СССР)	80 – 90		512	
ЕС-1025 (ЧССР)		60		256
ЕС-1030 (СССР, ПНР)	60		512	
ЕС-1032 (ПНР)	200		512	
ЕС-1033 (СССР)	200		512	
ЕС-1035 (СССР, НРБ)		140		512
ЕС-1040 (ГДР)	400		1024	
ЕС-1045 (СССР, ПНР)		800		4096
ЕС-1050 (СССР)	500		1024	
ЕС-1055 (ГДР)		600		2048
ЕС-1060 (СССР)		1300		8192
ЕС-1065 (СССР)		4500		16 384

Примечания: 1. Дополнительные средства: ВП – виртуальная память;
2. Производительность указана для набора операций, характерного для

повышение емкости ОП (до 16 Мбайт) и реализация виртуальной памяти;

расширение системных средств для организации многомашинных и многопроцессорных систем.

В табл. 1.1 приведены основные параметры машин ЕС ЭВМ I и II очереди, выпускаемых промышленностью стран СЭВ [49, 72].

Кроме машин общего назначения (ЕС ЭВМ) социалистические страны, подписавшие Соглашение по сотрудничеству в области вычислительной техники, создали систему малых и микро-ЭВМ. Малые и микро-ЭВМ этой системы в первую очередь предназначены для создания различных управляющих систем, работающих в реальном времени, при значительном уменьшении стоимости вычислительного оборудования, уменьшении затрат на его эксплуатацию, упрощении программного обеспечения.

Характеристики моделей малых и микро-ЭВМ, входящих в состав СМ ЭВМ, приведены в табл. 1.2.

В последних моделях СМ ЭВМ широко используются микропроцессоры для программируемых контроллеров перифе-

Единой системы I и II очередей

Мультиплексные каналы			Селекторные каналы			Дополни- тельные средства
Количество		Скорость, Кбайт/с	Количество		Максималь- ная скорость, Кбайт/с	
I	II		I	II		
1		16				ВП
1		20				
	1	20		1		
1		16	2		300	ВП
1		35	1		250	
1		80	2		500	
	1	24		1	800	
1		40	3		800	
1		40	3		400	
1		70	3		800	ВП, БМР
	1	30		4	800	
1		50	6		1250	
	1-2	40		5	1300	ВП, БМР
1		110	6		1300	
	2	40		4	1500	ВП, БМР
	2	110		6	1500	ВП, БМР
	2	200		11	1500	ВП, БМР

БМР — блок-мультиплексный режим.
научно-технических расчетов («смесь Гибсон-1»).

Таблица 1.2. Характеристики моделей СМ ЭВМ

Модель	Длина слова, бит	Производительность, тыс. операций/с	Максимальная емкость ОП, Кбайт
Малые ЭВМ:			
СМ-2М	16	230 *	256
СМ-4	16	230 *	256
Микро-ЭВМ:			
СМ-1600	8/16	400	256
СМ-1800	8	150 — 500	64
СМ-1300	16	150 — 500	64

* Для задач управления.

Таблица 1.3. Характеристики микро-ЭВМ семейства «Электроника»

Модель	Длина слова, бит	Производительность, тыс. операций/с	Максимальная емкость памяти, Кбайт
«Электроника-60»	16	250	64
«Электроника НЦ-80-01»	16	300	64

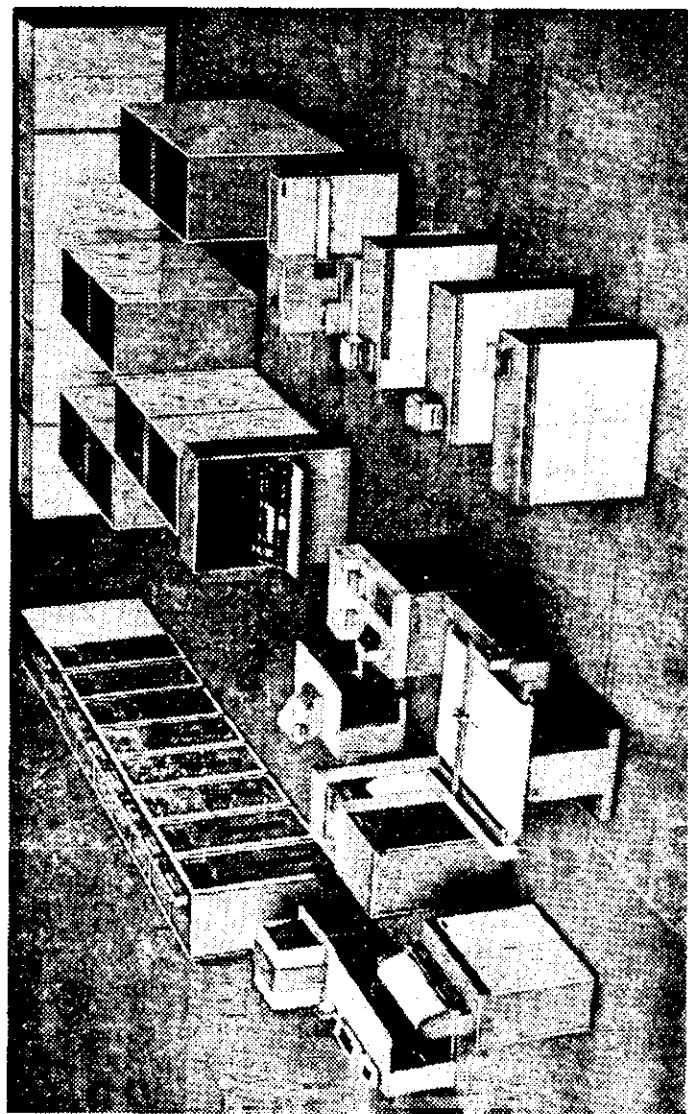


Рис. 1.6. Общий вид ЭВМ общего назначения ЕС-1050

Таблица 1.4. Характеристики микропроцессоров

Характеристика	Тип микропроцессора	
	K580	K1810
Число разрядов в слове	8	16
Скорость выполнения коротких операций (типа регистр-регистр), операций/с	500 тыс.	2 млн.
Скорость выполнения операций умножения и деления, операций/с	800 (по подпрограмме)	15 тыс.
Емкость адресуемой памяти, Кбайт	До 64	До 1024
Число общих регистров	6	14
Число входов прерывания	1 (расширяется до 64)	1 (расширяется неограниченно)
Число адресуемых портов ввода-вывода	256	32 тыс.

рийных устройств и при построении процессоров. В табл. 1.3 приведены характеристики некоторых микро-ЭВМ семейства «Электроника».

Разработка микропроцессоров и микро-ЭВМ проходила в последние годы менее упорядоченно, чем крупных машин. Поэтому выпускаемые промышленностью микропроцессоры и микро-ЭВМ не всегда образуют стройные системы моделей с разнообразными характеристиками. В табл. 1.4 приведены характеристики 8- и 16-разрядных микропроцессоров, обладающих информационной и односторонней программной совместимостью (программы микропроцессоров K580 можно использовать и в K1810). Особенности их архитектуры рассматриваются в гл. 8 и 9.

Глава вторая

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭВМ

2.1. ПОЗИЦИОННЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

Под системой счисления понимается способ представления любого числа с помощью некоторого алфавита символов, называемых цифрами. Существуют различные системы счисления. От их особенностей зависят наглядность представления числа при помощи цифр и сложность выполнения арифметических операций.

Римская система счисления является примером системы с очень сложным способом записи чисел и громоздкими правилами выполнения арифметических операций.

Наглядность представления чисел и сравнительная простота выполнения арифметических операций характерны для *позиционных систем счисления*.

Система счисления называется позиционной, если одна и та же цифра имеет различное значение, определяющееся позицией цифры в последовательности цифр, изображающей число. Это значение меняется в однозначной зависимости от позиции, занимаемой цифрой, по некоторому закону. Позиционной является десятичная система, используемая в повседневной жизни. Помимо десятичной существуют другие позиционные системы. Некоторые из них нашли применение в вычислительной технике.

Количество s различных цифр, употребляемых в позиционной системе, называется ее основанием. Эти цифры обозначают s целых чисел, обычно $0, 1, \dots, (s-1)$. В десятичной системе используются десять цифр: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$; эта система имеет основанием число десять.

В общем случае в позиционной системе с основанием s любое число x может быть представлено в виде полинома от основания s :

$$x = \varepsilon_r s^r + \varepsilon_{r-1} s^{r-1} + \dots + \varepsilon_1 s^1 + \varepsilon_0 s^0 + \varepsilon_{-1} s^{-1} + \varepsilon_{-2} s^{-2} + \dots, \quad (2.1)$$

где в качестве коэффициентов ε_i могут стоять любые из s цифр, используемых в системе счисления.

Принято представлять числа в виде соответствующей (2.1) последовательности цифр:

$$x = \varepsilon_r \varepsilon_{r-1} \dots \varepsilon_1 \varepsilon_0, \varepsilon_{-1} \varepsilon_{-2} \dots$$

В этой последовательности запятая отделяет целую часть числа от дробной (коэффициенты при положительных степенях, включая нуль, от коэффициентов при отрицательных степенях). Запятая опускается, если нет отрицательных степеней. Позиции цифр, отсчитываемые от запятой, называют разрядами. В позиционной системе счисления значение каждого разряда больше значения соседнего справа разряда в число раз, равное основанию s системы.

В ЭВМ применяют позиционные системы счисления с недесятичным основанием: двоичную, шестнадцатеричную, восьмеричную и др. В дальнейшем для обозначения используемой си-

стемы счисления будем заключать число в скобки и в индексе указывать основание системы счисления.

Наибольшее распространение в ЭВМ имеет *двоичная система счисления*. В этой системе используются только две («двоичные») цифры: 0 и 1.

В двоичной системе любое число может быть представлено последовательностью двоичных цифр

$$x = \alpha_m \alpha_{m-1} \dots \alpha_1 \alpha_0, \alpha_{-1} \alpha_{-2} \dots,$$

где α_i либо 0, либо 1.

Эта запись соответствует сумме степеней числа 2, взятых с указанными в ней коэффициентами:

$$x = \alpha_m \cdot 2^m + \alpha_{m-1} \cdot 2^{m-1} + \dots + \alpha_1 \cdot 2^1 + \alpha_0 \cdot 2^0 + \alpha_{-1} \cdot 2^{-1} + \alpha_{-2} \cdot 2^{-2} + \dots \quad (2.2)$$

Например, двоичное число

$$(10101101, 101)_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3},$$

как следует из приведенного разложения его по степеням числа 2, соответствует десятичному числу

$$(173, 625)_{10}.$$

Двоичное изображение числа требует большего (для многоразрядного числа примерно в 3,3 раза) числа разрядов, чем его десятичное представление. Тем не менее применение двоичной системы создает большие удобства для проектирования ЭВМ, так как для представления в машине разряда двоичного числа может быть использован любой простой элемент, имеющий всего два устойчивых состояния. Такими элементами, например, являются триггерные схемы и т. п. Другим важным достоинством двоичной системы является простота двоичной арифметики.

В *восьмеричной системе* употребляется восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7. Любое число в восьмеричной системе представляется последовательностью цифр

$$x = \beta_q \beta_{q-1} \dots \beta_1 \beta_0, \beta_{-1} \beta_{-2} \dots,$$

в которой β_i могут принимать значения от 0 до 7.

Например, восьмеричное число

$$(703, 04)_8 = 7 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0 + 0 \cdot 8^{-1} + 4 \cdot 8^{-2} = (451, 0625)_{10}.$$

В *шестнадцатеричной системе* для изображения чисел употребляется 16 цифр: от 0 до 15. При этом, чтобы одну цифру не изображать двумя знаками, приходится вводить специальные обозначения для цифр, больших девяти. Обозначим первые десять цифр этой системы цифрами от 0 до 9, а старшие пять цифр — латинскими буква-

ми: десять — *A*, одиннадцать — *B*, двенадцать — *C*, тринадцать — *D*, четырнадцать — *E*, пятнадцать — *F*.

Например, шестнадцатиричное число

$$(B2E, 4)_{16} = 11 \cdot 16^2 + 2 \cdot 16^1 + 14 \cdot 16^0 + 4 \cdot 16^{-1} = (2862, 25)_{10}.$$

Общие методы перевода чисел из одной системы в другую изложены в [32, 66]. Здесь ограничимся только рассмотрением правил преобразования восьмеричных и шестнадцатиричных чисел в двоичные и наоборот. Эти правила исключительно просты, поскольку основания восьмеричной и шестнадцатиричной систем есть целые степени числа два: $8 = 2^3$, $16 = 2^4$.

Для перевода восьмеричного (шестнадцатиричного) числа в двоичную форму достаточно заменить каждую цифру этого числа соответствующим трехразрядным (четырёхразрядным) двоичным числом, при этом отбрасывают ненужные нули, например

$$\begin{array}{cccc} \underbrace{3}_{011} & \underbrace{0}_{000} & \underbrace{5}_{101} & \underbrace{4}_{100} \end{array}_{16} = (11000101, 1)_2,$$

$$\begin{array}{cccc} \underbrace{7}_{0111} & \underbrace{B}_{1011} & \underbrace{2}_{0010} & \underbrace{E}_{1110} \end{array}_{16} = (11110110010, 111)_2.$$

Для перехода от двоичной к восьмеричной (или шестнадцатиричной) системе поступают следующим образом: двигаясь от запятой влево и вправо, разбивают двоичное число на группы по три (четыре) разряда, дополняя при необходимости нулями крайние левую и правую группы. Затем каждую группу из трех (четырех) разрядов заменяют соответствующей восьмеричной (шестнадцатиричной) цифрой.

Приведем примеры:

а) перевод двоичного числа 1101111001, 1101 в восьмеричное:

$$\begin{array}{cccccc} \underbrace{001}_1 & \underbrace{101}_5 & \underbrace{111}_7 & \underbrace{001}_1 & \underbrace{110}_6 & \underbrace{100}_4 \end{array} = (1571, 64)_8;$$

б) перевод двоичного числа 11111111011, 100111 в шестнадцатиричное:

$$\begin{array}{ccccc} \underbrace{0111}_7 & \underbrace{1111}_F & \underbrace{1011}_B & \underbrace{1001}_9 & \underbrace{1100}_C \end{array} = (7FB, 9C)_{16}.$$

В настоящее время в большинстве ЭВМ используются двоичная система и двоичный алфавит для представления и хранения чисел, команд и другой информации, а также при выполнении арифметических и логических операций.

Шестнадцатиричная (и восьмеричная) система применяется в текстах программ для более короткой и удобной записи двоичных кодов

команд, адресов и операндов. Кроме того, эти системы применяются в ЭВМ при некоторых формах представления чисел (см. § 2.3).

2.2. ДВОИЧНАЯ АРИФМЕТИКА

Правила выполнения арифметических действий над двоичными числами задаются таблицами двоичных сложения, вычитания и умножения:

Таблица двоичного сложения	Таблица двоичного вычитания	Таблица двоичного умножения
$0+0=0$	$0-0=0$	$0 \times 0=0$
$0+1=1$	$1-0=1$	$0 \times 1=0$
$1+0=1$	$1-1=0$	$1 \times 0=0$
$1+1=0$ + единица переноса в старший разряд	$10-1=1$	$1 \times 1=1$

Правила арифметики во всех позиционных системах аналогичны. При сложении в каждом разряде в соответствии с таблицей двоичного сложения производится сложение двух цифр слагаемых или двух этих цифр и 1, если имеется перенос из соседнего младшего разряда. В результате получается цифра соответствующего разряда суммы и, возможно, также 1 переноса в старший разряд. Приведем пример сложения двух двоичных чисел:

Переносы	1 111	1
	110111,01	55,25
+	10011,10	19,5
	1001010,11	74,75

Справа показано сложение тех же чисел, представленных в десятичной системе.

При вычитании двоичных чисел в данном разряде при необходимости занимает 1 из следующего старшего разряда. Эта занимаемая 1 равна двум 1 данного разряда. Заём производится каждый раз, когда цифра в разряде вычитаемого больше цифры в том же разряде уменьшаемого. Поясним сказанное примером:

$$\begin{array}{r} 11011,10 \\ - 1101,01 \\ \hline 1110,01 \end{array}$$

Умножение двоичных многоразрядных чисел производится путем образования частичных произведений и последующего их суммирования. В соответствии с таблицей двоичного умножения каждое частичное произведение равно 0, если в соответствующем разряде множителя стоит 0, или равно множимому, сдвинутому на соответствующее число

разрядов влево, если в разряде множителя стоит 1. Таким образом, операция умножения многоразрядных двоичных чисел сводится к операциям сдвига и сложения. Положение запятой определяется так же, как при умножении десятичных чисел. Сказанное поясняется примером:

$$1011,1 \times 101,01 = 111100,011$$

$$\begin{array}{r}
 \\
 \times 10111 \\
 \times 10101 \\
 \\
 10111 \\
 00000 \\
 + 10111 \\
 00000 \\
 10111 \\
 \hline
 111100011
 \end{array}$$

Особенности выполнения деления двоичных чисел поясняются следующим примером:

$$1100,011 : 10,01 = ?$$

$$\begin{array}{r}
 1100011 \quad | \quad 10010 \\
 \underline{10010} \\
 11011 \\
 \underline{10010} \\
 10010 \\
 \underline{10010} \\
 00000
 \end{array}$$

Благодаря простоте правил двоичного сложения, вычитания и умножения применение в ЭВМ двоичной системы счисления позволяет упростить схемы устройств, выполняющих арифметические операции.

2.3. ФОРМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ В ЭВМ

Любая информация (числа, команды, алфавитно-цифровые записи и т. п.) представляется в ЭВМ в виде двоичных кодов (двоичных слов) фиксированной или переменной длины. Отдельные элементы двоичного кода, имеющие значение 0 или 1, называют разрядами или битами. В ЭВМ слова часто разбивают на части, называемые *словами* или *байтами*. В современных ЭВМ широко используется байт, содержащий 8 бит (разрядов).

Двоичный разряд представляется в ЭВМ некоторым техническим устройством, например триггером, двум различным состояниям которого приписывают значения 0 и 1. Набор соответствующего количества таких устройств служит для представления многоразрядного двоичного числа (слова).

В ЭВМ применяют две формы представления чисел: с фиксированной запятой (точкой) и с плавающей запятой (точкой). Эти формы называют также соответственно естественной и полулогарифмической.

При представлении чисел с фиксированной запятой положение запятой фиксируется в определенном месте относительно разрядов числа. Обычно подразумевается, что запятая находится или перед стар-

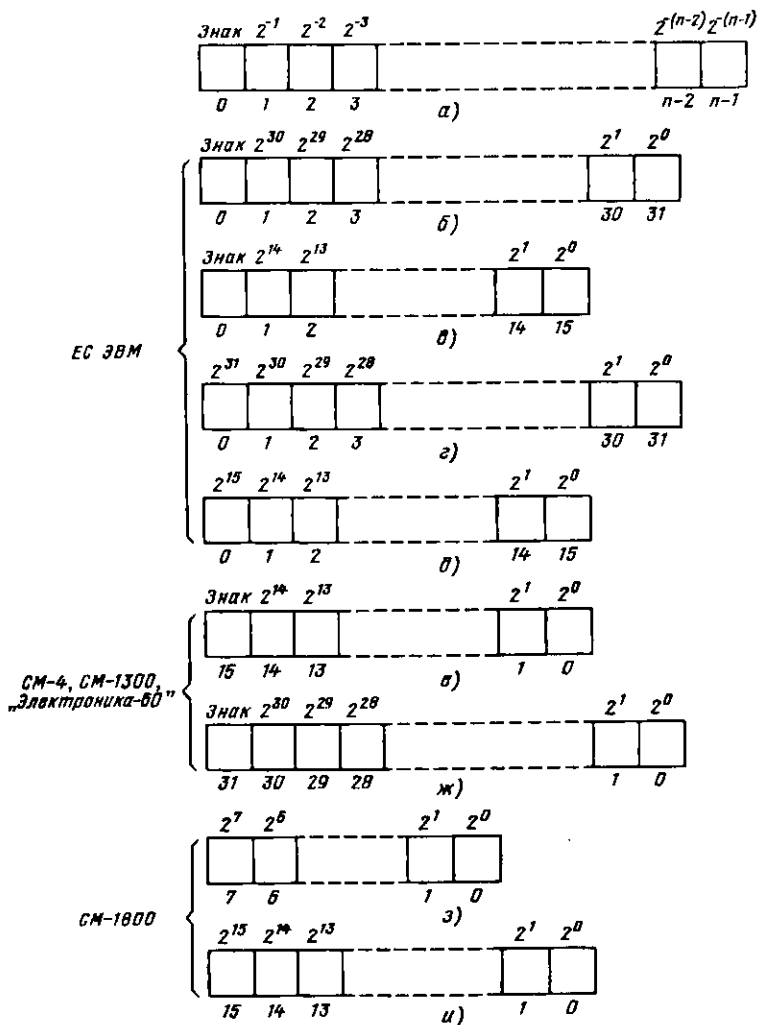


Рис. 2.1. Форматы данных для представления двоичных чисел с фиксированной запятой (точкой):

a — запятая фиксирована перед старшим разрядом; *б, в* — целые числа со знаком в формате слова и полуслова (ЕС ЭВМ); *г, д* — целые числа без знака в формате слова и полуслова (ЕС ЭВМ); *е, ж* — целые числа со знаком в формате слова и двойного слова (СМ-4, СМ-1300, «Электроника-60»); *з, и* — целые числа без знака в формате слова и двойного слова (СМ-1800)

шим разрядом, или после младшего. В первом случае могут быть представлены только числа, которые по модулю меньше 1, во втором — только целые числа.

На рис. 2.1 показаны примеры форматов данных для представления двоичных чисел с фиксированной запятой и соответствующие разрядные сетки. По сложившейся в вычислительной технике традиции нумерация разрядов (бит) в разрядной сетке в машинах общего назначения (ЕС ЭВМ) ведется слева направо, а в малых ЭВМ, микро-ЭВМ и микропроцессорах — справа налево. На разрядной сетке указаны веса разрядов.

При представлении числа со знаком для кода знака выделяется / «знаковый» разряд (обычно крайний слева). В этом разряде 0 соответствует плюсу, а 1 — минусу.

На рис. 2.1, а показан формат для чисел с запятой, фиксированной перед старшим разрядом. В этом формате могут быть с точностью до $2^{-(n-1)}$ представлены числа (правильные дроби) в диапазоне

$$2^{-(n-1)} \leq |x| \leq 1 - 2^{-(n-1)}.$$

Первые ЭВМ были машинами с фиксированной запятой, причем запятая фиксировалась перед старшим разрядом числа. В настоящее время, как правило, форму с фиксированной запятой применяют для представления целых чисел (запятая фиксирована после младшего разряда).

Используют два варианта представления целых чисел: со знаком и без знака. В последнем случае все разряды разрядной сетки служат для представления модуля числа. В ЕС ЭВМ применяются оба указанных варианта представления целых чисел, причем каждый из вариантов реализуется как в формате 32-разрядного машинного слова этих машин, так и в формате 16-разрядного полуслова (рис. 2.1, б-д).

Если запятая фиксирована справа от младшего разряда, то в n -разрядной сетке целых чисел со знаком можно представлять нуль и положительные и отрицательные целые двоичные числа, модуль которых $1 \leq |x| \leq 2^{n-1} - 1$, что при $n = 32$ соответствует диапазону абсолютных десятичных целых чисел примерно от 1 до 10^9 .

В малой ЭВМ СМ-4, а также в микро-ЭВМ «Электроника-60» и СМ-1300, оперирующих с 16-битным машинным словом, целые числа со знаком представляются в 16-разрядном формате слова и 32-разрядном формате двойного слова (рис. 2.1, е и ж). В 8-битной микро-ЭВМ целые числа представляются без знака в формате 8-разрядного слова и 16-разрядного двойного слова (рис. 2.1, з, к).

Представление чисел с фиксированной запятой используется как основное и единственное лишь в сравнительно небольших по своим вычислительным возможностям машинах, применяемых в системах передачи данных, для управления технологическими процессами и обработки измерительной информации в реальном масштабе времени.

В машинах, предназначенных для решения широкого круга вычислительных задач, основным является представление чисел с плавающей запятой, не требующее масштабирования данных. Однако в таких машинах часто наряду с этой формой представления чисел использует-

ся также и представление с фиксированной запятой для целых чисел, поскольку операции с такими числами выполняются за меньшее время. В частности, к операциям с целыми числами сводятся операции над кодами адресов (операции индексной арифметики).

Представление числа с плавающей запятой в общем случае имеет вид¹

$$x = s^p q; \quad |q| < 1, \quad (2.3)$$

где q — *мантисса* числа x ; s^p — характеристика числа x ; p — *порядок*; s — основание характеристики (обычно целая степень числа 2).

Мантисса (правильная дробь со знаком) и порядок (целое число со знаком) представляются в системе счисления с основанием, равным s (в соответствующей двоично-кодированной форме). Знак числа совпадает со знаком мантиссы.

Порядок p , который может быть положительным или отрицательным целым числом, определяет положение запятой в числе x .

На рис. 2.2 показаны примеры форматов данных для чисел с плавающей запятой. Одна часть бит формата используется для представления порядка, а другая — для мантиссы.

Особенности выполнения арифметических операций с числами с плавающей запятой рассматриваются в гл. 6. Арифметические действия над числами с плавающей запятой требуют выполнения помимо операций над мантиссами определенных операций над порядками (сравнение, вычитание и др.). Для упрощения операций над порядками их сводят к действиям над целыми положительными числами (целыми числами без знаков), применяя представление чисел с плавающей запятой со «смещенным порядком».

В случае представления числа с плавающей запятой со смещенным порядком к его порядку p прибавляется целое число — смещение $N = 2^k$, где k — число двоичных разрядов, используемых для модуля порядка.

Смещенный порядок $p_{\text{см}} = p + N$ всегда положителен. Для его представления необходимо такое же число разрядов, как и для модуля и знака порядка p .

Важная особенность смещенных порядков состоит в том, что если для порядков p' и p'' , представляющих собой целые числа со знаками, выполняется соотношение

$$p' \geq p'',$$

то и для положительных целых чисел соответствующих смещенных порядков $p'_{\text{см}}$ и $p''_{\text{см}}$ всегда

$$p'_{\text{см}} \geq p''_{\text{см}}.$$

¹ Это представление числа называют также полупологарифмическим, так как часть числа — характеристика — выражена в логарифмической форме.

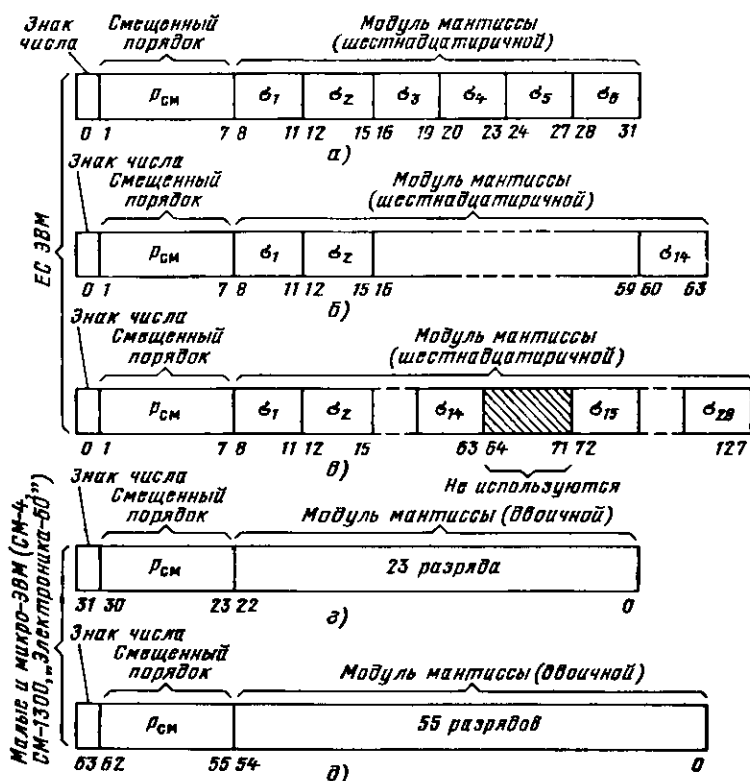


Рис. 2.2. Представление в ЭВМ чисел с плавающей запятой:
 а — в — шестнадцатичные числа со смещенным порядком в форматах соответственно коротком, длинном и расширенном (ЕС ЭВМ); г и д — двоичные числа со смещенным порядком в форматах соответственно коротком и длинном (ММЭ-4)

При фиксированном числе разрядов мантиссы любая величина представляется в машине с наибольшей возможной точностью нормализованным числом.

Число $x = s^p q$ называется *нормализованным*, если мантисса q удовлетворяет условию

$$1 > |q| \geq 1/s, \quad (2.4)$$

т. е. старший разряд мантиссы в s -ричной системе отличен от нуля.

В процессе вычислений может получаться ненормализованное число. В этом случае машина, если это предписано командой, автоматически нормализует его («нормализация результата» операции).

Пусть r старших разрядов s -ричной мантииссы равны 0. Тогда нормализация заключается в сдвиге мантииссы на r разрядов влево и уменьшении порядка на r единиц, при этом в младшие r разрядов мантииссы записывается 0. После такой операции число не меняется, а условие (2.4) выполняется. При нулевой мантииссе нормализация невозможна.

В различных ЭВМ применяются представления чисел с плавающей запятой в системах счисления с различными основаниями, но равными целой степени числа $2 (s = 2^w)$, при этом порядок p представляется целым числом, а мантиисса q — числом, в котором группы по w двоичных разрядов изображают цифры мантииссы с основанием системы счисления $s = 2^w$.

Примерами применяемых форм чисел с плавающей запятой с различными основаниями системы счисления являются

$$\begin{aligned}x &= 2^p q \quad (1 > |q| \geq 1/2); \\x &= 8^p q \quad (1 > |q| \geq 1/8); \\x &= 16^p q \quad (1 > |q| \geq 1/16).\end{aligned}$$

В скобках указаны соответствующие условия получения нормализованных чисел.

Использование для чисел с плавающей запятой недвоичного основания несколько уменьшает точность вычислений (при заданном числе разрядов мантииссы), но позволяет увеличить диапазон представляемых в машине чисел и ускорить выполнение некоторых операций, в частности нормализации, за счет того, что сдвиг может производиться сразу на несколько двоичных разрядов (на четыре разряда для $s = 16$). Кроме того, уменьшается вероятность появления ненормализованных чисел в ходе вычислений.

Диапазон представимых в машине чисел с плавающей запятой зависит от основания системы счисления и числа разрядов, выделенных для изображения порядка. Точность вычислений при плавающей запятой определяется числом разрядов мантииссы. С увеличением числа разрядов мантииссы увеличивается точность вычислений, но увеличивается и время выполнения арифметических операций.

Задачи, решаемые на ЭВМ, предъявляют различные требования к точности вычислений. Поэтому во многих машинах используется несколько форматов с плавающей запятой с различным числом разрядов мантииссы.

В ЕС ЭВМ числа с плавающей запятой представляются в шестнадцатиричной (двоично-кодированной) системе счисления. На рис. 2.2, а — а показаны три используемых в ЕС ЭВМ формата с плавающей запятой: короткий (мантиисса 6 шестнадцатиричных цифр), длинный (мантиисса 14 шестнадцатиричных цифр) и расширенный (мантиисса 28 шестнадцатиричных цифр). Во всех форматах крайний левый разряд является знаковым, а следующие семь разрядов служат для представления смещенного порядка при $N = 64$. Смещенный порядок может принимать значения от $p_{\text{см}} = 127$ (при $p = 63$) до $p_{\text{см}} = 0$ (при $p = -64$). Мантиисса q рассматривается как число, составленное из шестнадцати-

Таблица 2.1

Десятичное число	Эквивалентное шестнадцатиричное число	Мантисса нормализованного шестнадцатиричного числа	Порядок нормализованного шестнадцатиричного числа
$(16289)_{10}$	$(3FA1)_{16}$	$(0,3FA1)_{16}$	$(4)_{16}$
$(-16289)_{10}$	$(-3FA1)_{16}$	$(-0,3FA1)_{16}$	$(4)_{16}$
$(-0,01)_{10}$	$(-0,028F5C3)_{16}$	$(-0,28F5C3)_{16}$	$(-1)_{16}$
$(0,01)_{10}$	$(0,028F5C3)_{16}$	$(0,28F5C3)_{16}$	$(-1)_{16}$

ричных цифр в виде

$$q = \sum_{j=1}^k \sigma_j 16^{-j} \quad (\sigma_j = 0, 1, 2, \dots, F),$$

где k — число шестнадцатиричных разрядов мантиссы.

Шестнадцатиричное число с плавающей запятой считается нормализованным, если старшая шестнадцатиричная цифра σ_1 отлична от 0. В двоично-кодированном представлении нормализованного шестнадцатиричного числа три старшие цифры могут равняться 0. Это несколько увеличивает относительную погрешность представления чисел при фиксированном числе разрядов мантиссы.

Если r старших шестнадцатиричных разрядов мантиссы равны 0, то нормализация в этом случае состоит в сдвиге влево мантиссы на r шестнадцатиричных разрядов и соответствующем уменьшении порядка на r единиц.

Сдвиг на один шестнадцатиричный разряд выполняется как сдвиг мантиссы сразу на четыре двоичных разряда. Для случая, изображенного на рис. 2.2, а, наибольшее число сдвигов, которое может потребоваться при нормализации, равно 5. Для чисел с плавающей запятой с двоичным основанием системы счисления в формате, представленном на рис. 2.2, а наибольшее число сдвигов было бы равно 23.

В табл. 2.1 приведены примеры представления в ЕС ЭВМ чисел с плавающей запятой (в формате слова) в памяти машин. Следует обратить внимание на то, что при изображении кода в памяти шестнадцатиричными цифрами первые две старшие шестнадцатиричные цифры представляют совместно знак числа и смещенный порядок.

Представление чисел с плавающей запятой в малых и микро-ЭВМ (рис. 2.2, г и д) существенно отличается от принятого в ЕС ЭВМ. В этих машинах используется представление с двоичным основанием системы счисления. Имеются два формата: короткий и длинный, имеющие соответственно длину 32 и 64 разрядов. Для представления смещенного порядка отводится 8 разрядов (смещение $N = 128$). В коротком формате двоичная мантисса имеет длину 23 разряда, а в длинном 55. Числа в памяти всегда представляются в нормализованной форме, при которой старший разряд мантиссы всегда равен 1. Поэтому старший разряд мантиссы не запоминается, его единичное значение подра-

Смещенный порядок	Представление чисел в памяти ЭВМ двоичным словом	Представление чисел в памяти ЭВМ шестнадцатиричным словом
$(44)_{16}$	0 1000100001111111010000100000000	443FA100
$(44)_{16}$	1 1000100001111111010000100000000	C43FA100
$(3F)_{16}$	1 0111111001010001111010111000011	BF28F5C3
$(3F)_{16}$	0 0111111001010001111010111000011	3F28F5C3

зумеваются. Таким образом, в этих машинах фактически обеспечивается точность представления чисел, соответствующая длинам мантисс 24 и 56 разрядов.

Интересно отметить, что одинаковые по длине форматы чисел с плавающей запятой на рис. 2.2, а и г, использующие формы представления с 16-ричным и двоичным основаниями, имеют существенно различные диапазоны представимых чисел. В первом случае наибольшее представимое нормализованное число примерно равно 10^{76} , а во втором — лишь 10^{38} .

Использование чисел с плавающей запятой в формате, изображенном на рис. 2.2, а, соответствует вычислениям примерно с семью десятичными разрядами, что для ряда научно-технических расчетов недостаточно из-за накопления ошибок округления. Поэтому в ЕС ЭВМ для чисел с плавающей запятой предусмотрены еще два изображенных на рис. 2.2, в и г форматы: длинный и расширенный, занимающие соответственно два и четыре 32-разрядных слова (расширенный формат применяется в ЕС ЭВМ II очереди). В этих форматах не меняется число разрядов для изображения порядка и, следовательно, сохраняется диапазон представляемых чисел, а длина мантиссы увеличивается соответственно до 14 и 28 шестнадцатиричных разрядов. Использование таких представлений чисел эквивалентно выполнению вычислений соответственно примерно с 16 и 32 десятичными разрядами.

Наличие в ЭВМ нескольких длин форматов для представления чисел позволяет с учетом требований задачи выбирать или короткие форматы для чисел и тем самым экономить объем памяти, занимаемый данными, и снижать продолжительность отдельных операций ЭВМ, или использовать более длинные форматы для получения большей точности.

2.4. ПРЯМОЙ, ОБРАТНЫЙ И ДОПОЛНИТЕЛЬНЫЙ КОДЫ

В ЭВМ с целью упрощения арифметических операций применяют специальные коды для представления чисел. При помощи этих кодов упрощается определение знака результата операции, операция вычитания (или алгебраического сложения) чисел сводится к арифметическо-

му сложению их кодов, облегчается выработка признаков переполнения разрядной сетки. В результате упрощаются устройства ЭВМ, выполняющие арифметические операции.

Для представления отрицательных чисел в ЭВМ применяют прямой, обратный и дополнительный коды. Положительные числа представляются в прямом коде. Во всех этих кодах выделяются цифровые разряды и знаковый (крайний слева), представляющий знак числа, причем знак плюс кодируется цифрой 0, а знак минус цифрой 1.

Прямой код двоичного числа G с $(n-1)$ цифровыми разрядами определяется как

$$G_{\text{пр}} = \begin{cases} G & \text{при } G \geq 0; \\ A + |G| & \text{при } G \leq 0, \end{cases} \quad (2.5)$$

где A — величина, равная весу знакового разряда. Для дробных чисел $A = 1$, а для целых $A = 2^{n-1}$.

Сложение в прямом коде чисел, имеющих одинаковые знаки, выполняется достаточно просто. Числа складываются, и сумме присваивается код знака слагаемых. Значительно более сложной является операция алгебраического сложения в прямом коде чисел с различными знаками. В этом случае приходится определять большее по модулю число, производить вычитание чисел и присваивать разности знак большего по модулю числа.

Операция вычитания (алгебраического сложения) сводится к операции простого арифметического сложения при помощи обратного и дополнительного кодов, используемых для представления отрицательных чисел в машине.

Чтобы представить двоичное отрицательное число в обратном коде, нужно поставить в знаковый разряд 1, а во всех других разрядах заменить 1 нулями, а нули единицами.

Обратный код, если рассматривать его как число, является дополнением модуля исходного числа до наибольшего числа без знака, помещающегося в разрядную сетку. Для n -разрядной сетки имеем

$$G_{\text{обр}} = 2 - 2^{-(n-1)} - |G^-|, \quad (2.6)$$

если G^- — двоичная дробь, и

$$G_{\text{обр}} = 2^n - 1 - |G^-|, \quad (2.6a)$$

если G^- — целое двоичное число.

При представлении отрицательного двоичного числа в дополнительном коде ставят 1 в разряд знака, а цифровую часть числа заменяют дополнением модуля числа до 1 или 2^{n-1} соответственно для дробей и целых чисел. Дополнительный код отрицательного числа G^- определяется выражением

$$G_{\text{доп}} = 2 - |G^-|, \quad (2.7)$$

если G^- — двоичная дробь, и

$$G_{\text{доп}} = 2^n - |G^-|, \quad (2.7a)$$

если G^- — целое двоичное число.

Обратный и дополнительный коды числа можно рассматривать как двоичные числа без знаков, при этом для двоичных дробей $G_{\text{доп}}^- = G_{\text{обр}}^- + 2^{-(n-1)}$, а для двоичных целых чисел $G_{\text{доп}}^- = G_{\text{обр}}^- + 1$.

Таким образом, дополнительный код числа может быть получен из обратного путем прибавления 1 к младшему разряду обратного кода.

При выполнении расчетов на машине могут возникнуть как «положительный», так и «отрицательный» 0. Положительный 0 в прямом коде имеет вид

$$(+0)_{\text{пр}} = 000 \dots 0.$$

Отрицательный 0 изображается в прямом коде

$$(-0)_{\text{пр}} = 100 \dots 0,$$

в обратном

$$(-0)_{\text{обр}} = 111 \dots 1;$$

в дополнительном коде отрицательный 0 отсутствует.

При представлении положительных чисел прямым кодом, а отрицательных дополнительным нуль имеет единственное изображение. При применении обратного кода «положительный» и «отрицательный» 0 имеют разные изображения.

Изменению знака отрицательного числа соответствует инвертирование его кода, если число представлено в обратном коде, и инвертирование и добавление 1 младшего разряда, если отрицательное число представлено в дополнительном коде. В результате получается прямой код соответствующего положительного числа.

Сказанное следует из соотношений:

для дробей

$$-G_{\text{пр}}^- = |G^-| = 2 - 2^{-(n-1)} - G_{\text{обр}}^-; \quad (2.8)$$

$$-G_{\text{пр}}^- = |G^-| = 2 - G_{\text{доп}}^-; \quad (2.9)$$

для целых чисел

$$-G_{\text{пр}}^- = |G^-| = 2^n - 1 - G_{\text{обр}}^-; \quad (2.8a)$$

$$-G_{\text{пр}}^- = |G^-| = 2^n - G_{\text{доп}}^-; \quad (2.9a)$$

Рассмотрим применение обратного и дополнительного кодов при алгебраическом сложении n -разрядных двоичных чисел G и Q , когда одно из них или оба числа отрицательны. Могут быть сформулированы следующие правила (предполагаем, что модуль алгебраической суммы меньше 1 для дробей и меньше 2^{n-1} для целых чисел, и, следовательно, код суммы представим в n -разрядной сетке).

При алгебраическом сложении двух двоичных чисел с использованием обратного (или дополнительного) кода положительные слагаемые представляются в прямом коде, а отрицательные — в обратном (дополнительном) и производится арифметическое суммирование этих

кодов, включая разряды знаков, которые при этом рассматриваются как старшие разряды. При возникновении переноса из разряда знака единица переноса прибавляется к младшему разряду суммы кодов¹ при использовании обратного кода и отбрасывается при использовании дополнительного кода. В результате получается алгебраическая сумма в прямом коде, если эта сумма положительна, и в обратном (дополнительном), если она отрицательна.

Признак переполнения разрядной сетки. При алгебраическом сложении двух чисел, помещающихся в разрядную сетку, может возникнуть переполнение, т. е. образоваться сумма, требующая для своего представления на один цифровой разряд больше по сравнению с разрядной сеткой слагаемых. Можно сформулировать следующее правило (признак) для обнаружения переполнения разрядной сетки.

При алгебраическом сложении двух двоичных чисел с использованием прямого кода для представления положительных и дополнительного (обратного) кода для представления отрицательных чисел признаком переполнения является наличие переноса в знаковый разряд суммы при отсутствии переноса из ее знакового разряда (положительное переполнение) или наличие переноса из знакового разряда суммы при отсутствии переноса в ее знаковый разряд (отрицательное переполнение). Если и в знаковый, и из знакового разряда суммы есть переносы или нет этих переносов, то переполнение отсутствует. При положительном переполнении результат операции положительный, а при отрицательном отрицательный.

Справедливость этого утверждения доказывается, например, в [32].

2.5. КОДИРОВАНИЕ ДЕСЯТИЧНЫХ ЧИСЕЛ И АЛФАВИТНО-ЦИФРОВОЙ ИНФОРМАЦИИ

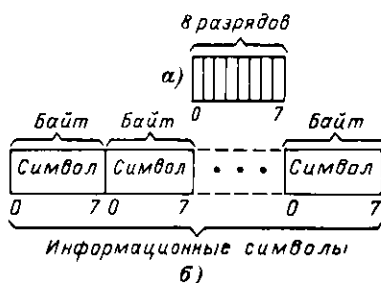
Современные ЭВМ обрабатывают не только числовую, но и текстовую, другими словами — алфавитно-цифровую, информацию, содержащую цифры, буквы, знаки препинания, математические и другие символы. Именно такой характер имеет экономическая, планово-производственная, учетная информация, а также тексты программ на алгоритмических языках и другая информация. Характер этой информации таков, что для ее представления требуются слова переменной длины.

Возможность ввода, обработки и вывода алфавитно-цифровой информации важна и для решения чисто математических задач, так как позволяет оформлять результаты вычислений в удобной форме — в виде таблиц с нужными заголовками и пояснениями.

Совокупность всех символов, используемых в вычислительной системе, представляет собой ее *алфавит*. Символу соответствует машинная единица информации *слог*. Так называют группу двоичных разрядов, служащую для представления символа в машине (двоичный код символа). Применяются различные варианты кодирования символов, использующие коды разной длины.

¹ Такой перенос называется круговым или циклическим.

Рис. 2.3. Восьмиразрядный слог (байт) (а) и размещение алфавитно-цифровых символов (б)



При выборе способа кодирования необходимо учитывать объем алфавита символов и требования, связанные с облегчением автоматической обработки данных.

Деловая информация в среднем содержит почти вдвое больше цифр, чем букв. Поэтому наряду с общей системой кодирования алфавитно-цифровых символов (десятичные цифры, буквы и другие знаки) в машинах сохраняют также отдельную систему кодирования для данных, состоящих только из десятичных цифр.

Необходимо, чтобы память машины эффективно использовалась при размещении в ней как алфавитно-цифровой, так и десятичной информации.

Наибольшее распространение получило представление алфавитно-цифровой информации с помощью 8-разрядных слогов (рис. 2.3), называемых *байтами*. С помощью байта можно кодировать 256 различных символов.

В качестве внутреннего кода для представления алфавитно-цифровых символов в памяти машины в ЕС ЭВМ применяется двоичный код для обработки информации (ДКОИ) или 8-разрядный код обмена информацией (КОИ-8), причем последний — только в ЕС ЭВМ I очереди. Указанные коды представлены на рис. 2.4.

Алфавитно-цифровая информация представляется словами переменной длины, содержащими нужное число байт-символов (рис. 2.3). Например, в ЕС ЭВМ алфавитно-цифровое слово может иметь длину от 1 до 256 байт.

Для упрощения автоматизации обработки данных применяют весовой принцип при кодировании символов. Двоичное число, соответствующее коду символа, называется его весом. При весовом кодировании веса кодов цифр последовательно возрастают, а веса кодов букв увеличиваются в алфавитном порядке. Вес кода буквы Б на 1 больше веса кода буквы А и т. д., а код пробела меньше веса кода буквы А.

Если необходимо расположить список фамилий в алфавитном порядке, то при весовом принципе кодирования эта операция может быть выполнена машиной путем сравнения двоичных чисел, соответствующих кодовым изображениям фамилий.

Рассмотрим примеры кодирования алфавитно-цифровой информации. При использовании кода ДКОИ (рис. 2.4, а) выражение «СССР 1917» будет храниться в памяти ЭВМ в виде алфавитно-цифрового

Зональная группа

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

¹ Допускается $\neg$ (логическое НЕ)

² Допускается прописная буква в

а)

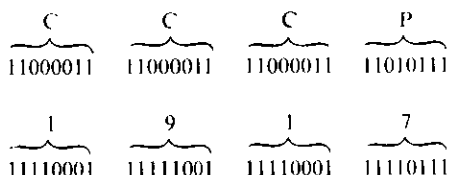
Зональная группа															
α_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
α_7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
α_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
α_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0 ЛУС	АР1	Про- дел	0	@	P	'	P	А00	Д16						
1 НЗ	СУ1	!	!	A	Q	α	α	Q	А01	Д17					
2 НТ	СУ2	"	"	2	B	R	b	γ	А02	Д18					
3 КТ	СУ3	#	#	3	C	S	C	S	А03	Д19					
4 КП	СПП	¤	¤	4	D	T	α	ε	А04	Д20					
5 КТМ	НЕТ	‰	‰	5	F	U	e	U	А05	Д21					
6 ДА	СИН	&	&	6	F	V	f	v	А06	Д22					
7 ЗВ	КБ	'	'	7	G	W	g	w	А07	Д23					
8 ВШ	АН	(	(	8	H	X	h	x	А08	Д24					
9 ГТ	КН	)	)	9	I	Y	i	y	А09	Д25					
10 ЛС	ЗМ	*	*	: J	Z	j	z	z	А10	Д26					
11 ВТ	АР2	+	+	; K	[	k	{	{	А11	Д27					
12 ЛФ	РФ	<	<	L \	\	\	/	/	А12	Д28					
13 ВК	РГ	=	=	M]	] m	] m	}	}	А13	Д29					
14 РУС	РЗ	.	.	N ^	^	^	~	~	А14	Д30					
15 ЛАТ	РЗ	/	/	? 0	-	-	0	36	А15	Д31					

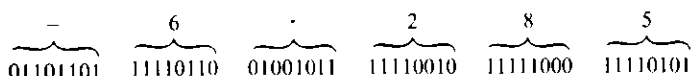
Рис. 2.4. Кодирование символов:
 a – двоичный код для обработки информации (ДКОИ);
 b – 8-битный код обмена информацией (КОИ-8)

¹ Допускается ¬ (логическое НЕ) ² Допускается прописная буква в

слова из 8 байт:



а число — 6,285 запишется в память в виде слова из 6 байт:



Заметим, что в рассматриваемом коде, как это принято в литературе на английском языке, целая часть числа отделяется от дробной точкой, а не запятой.

Для экономии объема памяти и для удобства выполнения арифметических операций над десятичными числами в машинах с байтовым представлением информации наряду с описанным выше способом представления алфавитно-цифровых символов предусматриваются специальные форматы для десятичных чисел — зонный («распакованный») (рис. 2.5, а) и уплотненный («упакованный») (рис. 2.5, б). Десятичные данные рассматриваются как десятичные числа со знаком. Числа могут иметь переменную длину. Для представления отрицательных чисел используется прямой код.

Десятичные цифры 0, 1, ..., 9 представляются в двоично-десятичной форме — коде 8421, в котором десятичная цифра изображается соответствующим 4-разрядным двоичным числом. Не используемые при этом комбинации 4-разрядных кодов (1010 — 1111) служат для кодирования знаков и служебных символов.

Код 8421 удобен для выполнения машиной (а не вручную) преобразований из десятичной системы в двоичную и обратно. Этот код аддитивен, т. е. сумма представлений двух цифр есть код их суммы. Однако использование этого кода связано с трудностями обнаружения переноса в следующий десятичный разряд и сложностью перехода к обратным и дополнительным кодам для десятичных чисел, облегчающим выполнение алгебраического сложения. Это объясняется тем, что код 8421 не является самодополняющимся, т. е. инверсия его двоичных цифр не дает кода дополнения десятичной цифры до 9.

Для зонного формата в каждом байте содержатся только одна десятичная цифра и служебный знак (зона), при этом четыре правых разряда байта служат для представления десятичной цифры в двоично-десятичном коде, а четыре левых заняты специальным 4-разрядным кодом, называемым *зоной*. Младший байт в этом формате состоит из кодов знака и младшей десятичной цифры числа.

Для знаков +, - и зоны используются разные коды в зависимости от применяемой системы кодирования символов ДКОИ или

Зона	Цифра	Зона	Цифра		Зона	Цифра	Знак	Цифра			
0	3	4	7	0	3	4	7	0	3	4	7

а)

Цифра	Цифра	Цифра	Цифра		Цифра	Цифра	Цифра	Знак			
0	3	4	7	0	3	4	7	0	3	4	7

б)

Рис. 2.5. Представление десятичной информации (ЕС ЭВМ):

а — зонный (распакованный) формат; б — упакованный формат

КОИ-8. В ДКОИ принято кодировать: плюс — 1100, минус — 1101 и зона — 1111. При работе с кодом КОИ-8 обозначают плюс — 1010, минус — 1011 и зону — 0101.

Число — 6,285 в зонном десятичном формате (при работе с кодом ДКОИ) будет иметь вид

$\begin{array}{cccc} 6 & 2 & 8 & -5 \\ \hline 11110110 & 11110010 & 11111000 & 11010101 \end{array}$

при этом информация о месте расположения десятичной запятой (точки) фиксируется в самой программе задачи.

При кодировании десятичного числа без знака в левые четыре разряда младшего байта записывается код зоны.

Важное достоинство 8-разрядного слова — байта состоит в возможности размещения в одном слове кодов двух десятичных цифр, чем достигается существенная экономия памяти. Эта возможность реализуется в упакованном десятичном формате (рис. 2.5, б). В этом формате код знака размещается в правых четырех разрядах младшего байта. Десятичное число всегда занимает целое число байтов. Если левые четыре разряда самого левого (старшего) байта оказываются свободными, они заполняются нулями.

Число — 6,285 в упакованном десятичном формате имеет вид

$\begin{array}{cccc} 0 & 6 & 2 & 8 & 5 & - \\ \hline 00000110 & 00101000 & 01011101 \\ \hline \text{Байт} & \text{Байт} & \text{Байт} \end{array}$

В ЕС ЭВМ в упакованном формате десятичное число может иметь длину от 1 до 16 байт. В этом случае максимальная длина числа — это 31 десятичная цифра плюс знак.

При выполнении операций над десятичными числами используется только упакованный формат. Результат также получается в этом формате. Числа, участвующие в операции, могут иметь неодинаковую

длину. Они рассматриваются как целые числа, выравненные по младшим разрядам. Формат с зоной используется при операциях ввода-вывода десятичных данных.

В вычислительных системах, использующих форматы, показанные на рис. 2.5, имеются команды для преобразования десятичных чисел из зонного формата в упакованный и обратно.

Глава третья

ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ И ТИПОВЫЕ УЗЛЫ

3.1. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ ФИЗИЧЕСКИМИ СИГНАЛАМИ

В предыдущей главе было показано, как информация представляется в двоичном алфавите. Физическими аналогами знаков 0 и 1 двоичного алфавита служат сигналы, способные принимать два хорошо различимых значения, например, напряжение (потенциал) высокого и низкого уровней, отсутствие и наличие электрического импульса, противоположные по знаку значения магнитной индукции и т. п.

В схемах цифровых устройств переменные и соответствующие им сигналы изменяются не непрерывно, а лишь в дискретные моменты времени, обозначаемые целыми неотрицательными числами: 0, 1, 2, ..., i ...

Временной интервал между двумя соседними моментами дискретного времени называется *тактом*.

Во многих случаях цифровые устройства содержат специальный блок, вырабатывающий синхронизирующие сигналы (СС'), отмечающие моменты дискретного времени (границы тактов).

В цифровых вычислительных устройствах обычно применяют потенциальный и импульсный способы физического представления информации.

При потенциальном способе (рис. 3.1, а) двум значениям переменной 1 и 0 соответствуют разные уровни напряжения в соответствующей точке схемы машины (*потенциальный код*). Потенциальный сигнал сохраняет постоянный уровень в течение такта, а его значение в переходные моменты не является определенным.

При импульсном способе представления информации (рис. 3.1, б) единичное и нулевое значения двоичной переменной отображаются наличием и отсутствием электрического импуль-

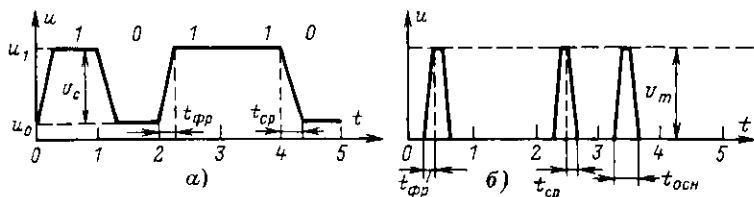


Рис. 3.1. Способы представления цифровой информации:
 а — сигналы потенциального типа; б — сигналы импульсного типа

са или разнополярными импульсами в соответствующей точке схемы (*импульсный код*).

Импульсный сигнал можно характеризовать амплитудой U_m , продолжительностью импульса по основанию $t_{осн}$, длительностью фронта $t_{фр}$ и среза $t_{ср}$ (рис. 3.1, б).

Аналогичные понятия могут быть применены к потенциальному сигналу (рис. 3.1, а). Потенциальный сигнал характеризуется, кроме того, разностью U_c верхнего и нижнего уровней напряжения. Понятия фронта и среза у потенциального сигнала всегда связаны с процессом перехода соответственно от нижнего к верхнему и от верхнего к нижнему уровню напряжений.

В соответствии с типом используемых сигналов для представления информации схемы цифровых устройств принято делить на импульсные, потенциальные и импульсно-потенциальные.

Слово может быть представлено последовательным или параллельным кодом.

При *последовательном коде* каждый временной такт предназначен для отображения одного разряда кода слова (рис. 3.2). В этом случае все разряды слова фиксируются по очереди одним и тем же элементом и проходят через одну линию передачи информации.

При *параллельном коде* все разряды двоичного кода слова представляются в одном временном такте, фиксируются отдельными элементами и проходят через отдельные линии, каждая из которых служит для представления и передачи только одного разряда слова.

При параллельной передаче информации код слова развертывается не во времени, а в пространстве, так как значения всех разрядов слова передаются по нескольким линиям одновременно (рис. 3.3).

В зависимости от применяемого кода устройства вычислительной техники называются последовательными или параллельными. При использовании последовательного кода все

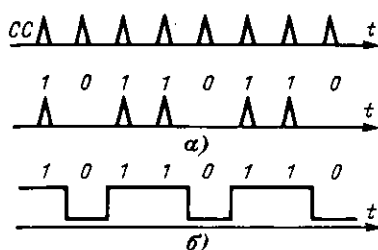


Рис. 3.2. Последовательный импульсный код (а) и последовательный потенциальный код (б)

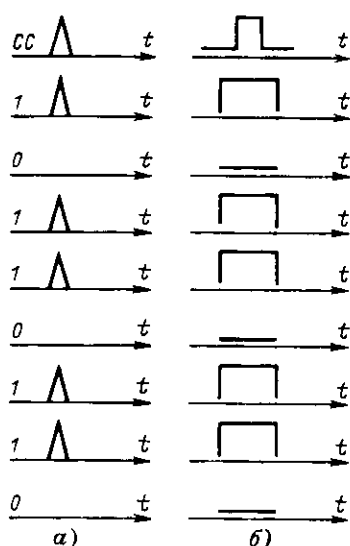


Рис. 3.3. Параллельный импульсный код (а) и параллельный потенциальный код (б)

операции, в том числе передача слов из одного узла в другой, производятся поочередно для каждого разряда слова, и поэтому последовательные устройства работают медленнее, чем параллельные. В современных ЭВМ основные устройства, участвующие в обработке информации, для достижения высокого быстродействия строятся как параллельные, хотя они и требуют большего объема аппаратуры. Для экономии оборудования в некоторых устройствах применяют последовательно-параллельный код, при котором слова разбиваются на части (слоги) и передача, а иногда и обработка производятся последовательно слог за слогом. При этом каждый слог представляется параллельным кодом.

3.2. ПОНЯТИЕ О КОМБИНАЦИОННОЙ СХЕМЕ И ЦИФРОВОМ АВТОМАТЕ

Устройство, преобразующее дискретную информацию, в общем случае имеет n входов для входных сигналов и m выходов, с которых снимаются выходные сигналы.

Каждый из входных сигналов соответствует некоторому символу (букве) входного алфавита. В свою очередь, выходные сигналы представляют собой символы (буквы) выходного алфавита. В качестве букв этих алфавитов обычно используются двоичные и, реже, десятичные цифры.

Преобразование информации в ЭВМ производится электронными устройствами (логическими схемами) двух классов: комбинационными схемами и цифровыми автоматами.

В комбинационных схемах (КС) совокупность выходных сигналов (выходное слово Y) в любой момент времени однозначно определяется входными сигналами (входным словом X), поступающими на входы в тот же момент времени (рис. 3.4, а).

Реализуемый в этих схемах способ обработки информации называется комбинационным, так как результат обработки информации зависит только от комбинации входных сигналов и вырабатывается сразу при подаче входной информации.

Закон функционирования КС определен, если задано соответствие между ее входными и выходными словами, например, в виде таблицы.

Это соответствие может быть задано и в аналитической форме с использованием булевых функций.

Другой, более сложный класс преобразователей дискретной информации составляют цифровые автоматы. Цифровой автомат в отличие от комбинационной схемы имеет некоторое конечное число различных внутренних состояний.

Под воздействием входного слова цифровой автомат переходит из одного состояния в другое и выдает выходное слово. Выходное слово на выходе цифрового автомата в такте определяется в общем случае входным словом, поступившим в этот такт на вход автомата, и внутренним состоянием автомата, которое явилось результатом воздействия на автомат входных слов в предыдущие такты.

Комбинация входного слова и текущего состояния автомата в данном такте определяет не только выходное слово, но и то состояние, в которое автомат перейдет к началу следующего такта.

Цифровой автомат содержит память, состоящую из запоминающих элементов (ЗЭ) — триггеров, элементов задержки

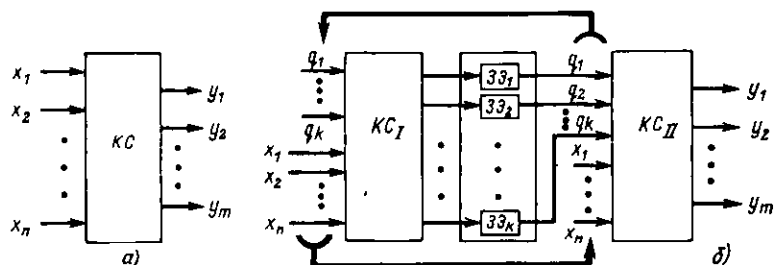


Рис. 3.4. Комбинационная схема (а) и цифровой автомат (б)

и др., фиксирующих состояние, в котором он находится. Комбинационная схема не содержит ЗЭ. Поэтому ее называют автоматом без памяти или примитивным автоматом.

Структурная схема цифрового автомата, представленная на рис. 3.4, б, содержит запоминающие элементы $ЗЭ_1 - ЗЭ_k$ и комбинационные схемы $КС_I$ и $КС_{II}$.

Состояния ЗЭ, определяющие состояние автомата, передаются в форме сигналов q_j по цепям прямой связи на входы $КС_{II}$ и по цепям обратной связи на входы $КС_I$. На входы комбинационных схем поступают также сигналы $x_1, \dots, x_n$ с входов автомата.

Выходное слово вырабатывается в $КС_{II}$, причем входными переменными для нее служат буквы входного слова и состояния ЗЭ — состояние автомата. Выходные сигналы $КС_I$ переводят автомат (его ЗЭ) в новое состояние, при этом входными переменными для этой схемы служат буквы входного слова и состояния ЗЭ. Одновременность появления новых значений входных сигналов на всех входах устройства достигается с помощью тактирующих сигналов, называемых также синхросигналами, обеспечивающих передачу информации с ЗЭ на входы комбинационной схемы одновременно с сигналами, поступающими на ее входы с других устройств.

В ряде случаев при анализе автомата его заменяют автоматом с одним эквивалентным входом и одним эквивалентным выходом и считают, что эквивалентные входной сигнал $x(t)$ и выходной сигнал $y(t)$ принимают значения из соответствующим образом преобразованных алфавитов P и S входных и выходных сигналов.

Для задания цифрового автомата должны быть указаны:

- 1) входной алфавит $P = \{p_1, p_2, \dots, p_N\}$;
- 2) выходной алфавит $S = \{s_1, s_2, \dots, s_M\}$;
- 3) алфавит состояний $Q = \{Q_0, Q_1, \dots, Q_r\}$;
- 4) начальное состояние автомата Q_0 ;

5) функции перехода $A(Q, x)$ и выходов $B(Q, x)$, однозначно определяющие зависимость соответственно состояния автомата $Q(t+1)$ в такте $(t+1)$ и выходного сигнала $y(t)$ от состояния автомата $Q(t)$ и входного сигнала $x(t)$ в такте t .

Используя функции переходов и выходов, можно поведение автомата описать выражениями

$$Q(t+1) = A[Q(t), x(t)]; \quad (3.1)$$

$$y(t) = B[Q(t), x(t)], \quad (3.2)$$

где $t = 0, 1, 2, \dots$; $Q(0) = Q_0$.

Выражениям (3.1) и (3.2) соответствует автомат, выходной сигнал которого зависит от состояния автомата и от сигнала на его входе. Такой автомат называется *автоматом Мили*.

В устройствах ЭВМ широко применяются и так называемые *автоматы Мура*, у которых выходной сигнал $y(t)$ в такте t зависит исключительно от состояния автомата $Q(t)$ в этом такте и не зависит от входного сигнала $x(t)$.

Функционирование автомата Мура описывается выражениями

$$Q(t+1) = A[Q(t), x(t)]; \quad (3.3)$$

$$y(t) = B[Q(t)], \quad (3.4)$$

где $t = 0, 1, 2, \dots$; $Q(0) = Q_0$.

Функции переходов и выходов могут задаваться различными способами, например в форме таблиц или с помощью графов. При задании в виде графов (см. гл. 7) состояния автомата представляются вершинами, а переходы из состояния в состояние — дугами. На дугах указываются значения входных сигналов, вызывающих соответствующие переходы. Выходные сигналы автомата Мура указываются рядом с вершинами графа. Выходные сигналы автомата Мили, вырабатываемые перед переходом, указываются на соответствующих дугах.

В теории автоматов вводятся понятия *полной системы переходов* и *полной системы выходов автомата*. Если для двух любых состояний Q_i и Q_j автомата имеется входной сигнал, переводящий автомат из состояния Q_i в Q_j , то такой автомат называется автоматом с полной системой переходов. Автомат Мура имеет полную систему выходов, если выходные сигналы различны для всех его состояний.

При построении узлов ЭВМ, являющихся цифровыми автоматами, в качестве запоминающих элементов (элементов памяти) используются элементарные автоматы. Элементарными автоматами служат автоматы Мура с двумя состояниями, обладающие полными системами переходов и выходов.

Для описания законов функционирования комбинационных схем используется математический аппарат булевых функций [22, 67]. Напомним, что переменные $x_1, x_2, \dots, x_n$ называются двоичными, если они могут принимать только два значения: 0 и 1, а функция от двоичных переменных $f(x_1, x_2, \dots, x_n)$ называется *булевой*, если она, так же как и ее аргументы, принимает только два значения: 0 и 1.

Система булевых функций W называется *функционально полной*, если для любой булевой функции $f(x_1, \dots, x_n)$ может

быть построена равная ей функция путем суперпозиции функций $x_1, \dots, x_n$ и функций системы W , взятых в любом конечном числе экземпляров каждая.

В математической логике доказывается, что если система булевых функций содержит функции $x_1 x_2$ (конъюнкция), $x_1 \vee x_2$ (дизъюнкция), $\bar{x}$ (инверсия), то она является функционально полной.

Из этой системы можно исключить некоторые функции без нарушения функциональной полноты. Действительно, функциональной полнотой будут обладать также система булевых функций $x_1 \vee x_2, \bar{x}$ и система $x_1 x_2, \bar{x}$.

Функционально полной является система, состоящая из одной булевой функции И — НЕ («штрих Шеффера»): $\bar{x}_1 \bar{x}_2$. Функциональной полнотой обладает и система, содержащая единственную булевую функцию ИЛИ — НЕ («стрелка Пирса»): $\overline{x_1 \vee x_2}$. Существуют и другие функционально полные системы булевых функций.

Техническим аналогом булевой функции является комбинационная схема, выполняющая соответствующее этой функции преобразование информации.

Постоянные уровни напряжения, соответствующие принятому в схеме представлению сигналов 0 и 1, могут рассматриваться как технические аналоги функции константы 0 и константы 1.

Логические операции над двоичными переменными реализуются схемами, которые называются комбинационными логическими элементами. Число входов комбинационного логического элемента соответствует числу аргументов воспроизводимых им одной или нескольких булевых функций.

Подобно тому как сложная булева функция может быть получена суперпозицией более простых функций, так и комбинационная схема строится из элементарных схем — из комбинационных логических элементов. При этом легко установить технический аналог операции суперпозиции. Последовательное соединение комбинационных схем, в том числе комбинационных логических элементов, при котором выход одной схемы соединен с входом другой схемы, соответствует подстановке в булевы функции в качестве аргументов других булевых функций. Пересоединение на входах комбинационных схем соответствует перестановке аргументов булевых функций.

Логические элементы могут быть не только комбинационными. Используются также логические элементы, реализующие функции элементарных автоматов. Подробнее о логических элементах см. в § 3.3.

Схему, показывающую связи между различными логически-

ми элементами, где сами элементы представлены условными обозначениями, называют функциональной.

При вычерчивании функциональных схем логические элементы, а часто и более крупные части схем изображают прямоугольниками, в которые вписывают символы реализуемых функций. При этом если функция реализуется при инверсных значениях некоторых входных переменных, то соответствующие входы отмечаются кружком. Выход отмечается кружком, если функция реализуется с инверсией. Примеры схем, реализующих булевы функции, даны на рис. 3.5.

Набор логических элементов для построения комбинационных схем является функционально полным, если реали-

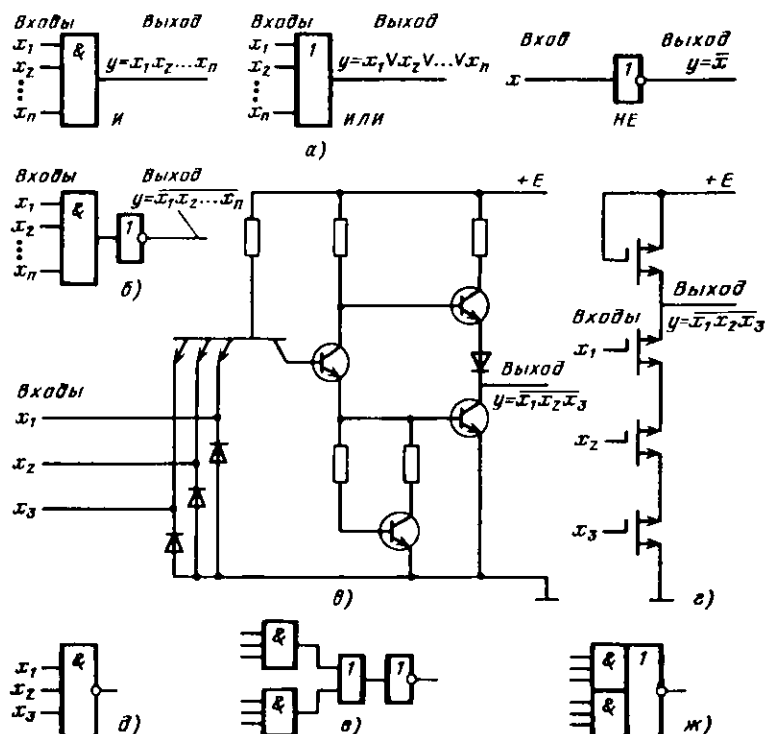


Рис. 3.5. Схемы логических элементов:

а — элементы И, ИЛИ, НЕ; б — функциональная схема элемента И — НЕ; в — принципиальная электрическая схема элемента И — НЕ транзисторно-транзисторной логики; г — принципиальная электрическая схема элемента И — НЕ на МОП-транзисторах; д — условное обозначение элемента И — НЕ; е — функциональная схема элемента И — ИЛИ — НЕ; ж — условное обозначение элемента И — ИЛИ — НЕ

зуемые этими элементами булевы функции образуют функционально полную систему функций.

Комплекс элементов обладает функциональной полнотой для построения цифровых автоматов, если он содержит функционально полный набор логических элементов для построения комбинационных схем и элементарный автомат с полной системой выходов и переходов.

В ЭВМ в качестве элементарных автоматов используются главным образом триггеры. Описание триггеров различных типов приведено в § 3.7.

Особенности построения цифровых устройств связаны со способом передачи информации между логическими элементами. Передача информации от элемента к элементу может осуществляться несинхронизируемым (асинхронным) и синхронизируемым способами, причем последний используется только при передаче информации в запоминающие элементы.

При несинхронизируемом способе передачи информации входные сигналы логических элементов преобразуются с небольшой задержкой в выходные сигналы, которые непосредственно воздействуют на входы следующих логических элементов.

При синхронизируемом способе передачи информации входные сигналы воздействуют на запоминающие логические элементы в строго определенные моменты времени, соответствующие появлению синхронизирующих сигналов.

Передача информации между запоминающими логическими элементами в общем случае требует выполнения следующего условия: она производится только после завершения передачи информации о предыдущем состоянии принимающего информацию элемента другому ЗЭ. Для выполнения этого условия в потенциальных схемах используется несколько серий синхронизирующих сигналов, сдвинутых во времени относительно друг друга так, что когда сигнал одной серии принимает значение 1, то сигнал другой серии — значение 0.

3.3. СИСТЕМЫ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

Системой (комплексом или серией) логических элементов ЭВМ называется предназначенный для построения цифровых устройств функционально полный набор логических элементов, объединяемый общими электрическими, конструктивными и технологическими параметрами, использующий одинаковый способ представления информации и одинаковый тип межэлементных связей. Система элементов чаще всего избыточна по своему функциональному составу, что позволяет строить схемы, более экономные по числу использованных элементов. Системы элементов содержат элементы для выполнения логических

операций, запоминающие элементы, элементы, реализующие функции узлов ЭВМ, а также элементы для усиления, восстановления и формирования сигналов стандартной формы.

Элементы представляют собой микроминиатюризованные интегральные электронные схемы (микросхемы), сформированные в кристалле кремния с помощью специальных технологических процессов.

В большинстве современных систем (серий) элементов имеются микросхемы малой степени интеграции (ИС), средней степени интеграции (СИС) и большой степени интеграции (БИС). Логические элементы в виде схем ИС реализуют совокупность логических операций, таких, как И, ИЛИ, И — ИЛИ, И — НЕ, ИЛИ — НЕ, И — ИЛИ — НЕ и триггеры. Логические элементы на микросхемах СИС и БИС реализуют узлы ЭВМ.

Основными параметрами системы логических элементов являются уровни питающих напряжений и уровни сигналов для представления логических 0 и 1, нагрузочная способность (коэффициент разветвления по выходу), помехоустойчивость, рассеиваемая мощность, быстродействие.

Основные, наиболее часто употребляемые типы интегральных элементов, — это потенциальные элементы транзисторно-транзисторной логики (ТТЛ), потенциальные элементы транзисторной логики с эмиттерными связями (ЭСЛ) и элементы на МОП-транзисторах.

Схемы ТТЛ относятся к интегральным логическим элементам среднего быстродействия. Время задержки сигнала в элементе ТТЛ составляет 10–30 нс. Нагрузочная способность схем ТТЛ допускает подключение к выходу элемента до десяти логических схем.

Более высоким быстродействием обладают интегральные микросхемы с эмиттерными связями (ЭСЛ), в которых транзисторы не входят в насыщение. Элементы ЭСЛ работают по принципу переключения токов при малых изменениях входных напряжений. Вследствие этого элементы ЭСЛ часто называют схемами с переключателями тока.

Время задержки у элемента ЭСЛ меньше, чем у элемента ТТЛ, и обычно имеет значение 1–5 нс.

Интегральные микросхемы на МОП (металл — окисел — полупроводник)-транзисторах являются более медленно действующими, чем элементы ТТЛ или ЭСЛ. Время задержки у элемента на МОП-транзисторах обычно не менее 50–100 нс. Однако эти элементы отличаются меньшей потребляемой мощностью, большой нагрузочной способностью и помехоустойчивостью и, что особенно важно, требуют меньшей площади на поверхности интегральной микросхемы. Схемы на МОП-транзисторах технологичны и дешевы. Поэтому они находят широкое применение, особенно в цифровых устройствах, не требующих очень высокого быстродействия, или в устройствах, для которых важна очень высокая степень интеграции, таких, как, например, устройства памяти (см. гл. 4).

МОП-транзисторы бывают *n*- и *p*-типов. Строят схемы и с одновременным использованием транзисторов *n*- и *p*-типов (дополняющие транзисторы).

Схемы с дополняющими транзисторами (К-МОП-схемы) отличаются малой потребляемой мощностью и более высоким быстродействием, так как в цепях заряда и разряда паразитных емкостей схемы оказываются включенными малые сопротивления открытых транзисторов.

3.4. ТРИГГЕРЫ

Триггер является элементом, который может находиться в одном из двух устойчивых состояний. Одному из этих состояний приписывается значение 1, а другому — 0. Состояние триггера распознается по его выходному сигналу. Под влиянием входного сигнала триггер скачкообразно переходит из одного устойчивого состояния в другое, при этом скачкообразно изменяется уровень напряжения его выходного сигнала.

Для удобства использования в схемах вычислительных устройств триггеры обычно имеют два выхода: прямой Q (называется также «выход 1») и инверсный $\bar{Q}$ («выход 0»). В единичном состоянии триггера на выходе Q высокий уровень сигнала, а в нулевом — низкий. На выходе $\bar{Q}$ наоборот.

В интегральных системах элементов триггер и схема, управляющая его входами («входная логика»), выполняются на одной кремниевой пластине, заключенной в корпус с выводами. При этом на одной пластине (в одном корпусе) может находиться несколько триггеров.

Схемы триггеров можно разделить на несколько типов: с установочными входами — RS -триггер, со счетным входом — T -триггер, а также D -триггер, JK -триггер и др.

Если хотя бы с одного входа информация в триггер заносится принудительно под воздействием синхронизирующего сигнала, то триггер называется синхронизируемым (синхронным). Если занесение информации в триггер с любого входа производится без синхронизирующего сигнала, то триггер называется несинхронизируемым (асинхронным).

Общая форма условного обозначения триггеров показана на рис. 3.6. В основном поле ставится символ T или символ TT для обозначения соответственно однократного или двухкратного триггера. Дополнительное поле может быть разделено на две части: асинхронную и синхронную. В первой представляются символы R и S входов несинхронизируемой установки триггера в 1 и 0, во второй — на местах X_1 и X_2 символы в соответствии с типом триггера. При этом пользуются следующими обозначениями для входов:

S — вход установки триггера в 1;

R — вход установки триггера в 0;

T — вход триггера со счетным входом;

D — вход D -триггера;

J — вход для синхронизируемой установки состояния 1 в JK -триггере;

K — вход для синхронизируемой установки состояния 0 в JK -триггере;

C — вход синхронизации.

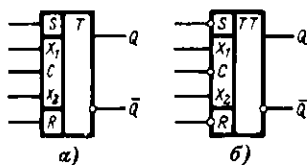


Рис. 3.6. Условные обозначения интегральных триггеров:

а — синхронизируемый одноклапчатый триггер с дополнительными входами S и R несинхронизируемой установки; б — синхронизируемый двухклапчатый триггер с дополнительными входами S и R несинхронизируемой установки

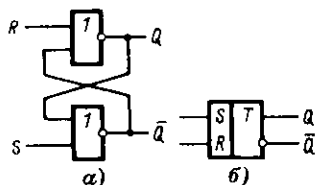


Рис. 3.7. Асинхронный RS-триггер на элементах ИЛИ — НЕ:

а — функциональная схема; б — условное обозначение

Если вход отмечен кружком, это означает, что действующее значение входного сигнала — 0.

Например, отсутствие кружка на входе C на рис. 3.6, а указывает на то, что входная информация заносится в триггер при единичном значении синхронизирующего сигнала (действующее значение синхронизирующего сигнала равно 1); кружок на входе C (рис. 3.6, б) означает, что прием информации происходит при нулевом значении синхронизирующего сигнала (действующее значение синхронизирующего сигнала равно 0).

Состояние триггера определяется сигналом Q на прямом выходе триггера (или сигналом $\bar{Q}$ на его инверсном выходе).

Законы функционирования триггеров задаются таблицами переходов с компактной записью, при которой в столбце состояний может быть указано, что новое состояние совпадает с предыдущим либо является его отрицанием.

Асинхронный RS-триггер. Асинхронный (несинхронизируемый) RS-триггер на интегральных элементах ИЛИ — НЕ показан на рис. 3.7. Триггер образован из двух комбинационных схем ИЛИ — НЕ, соединенных таким образом, что возникают положительные обратные связи, благодаря которым в устойчивом состоянии выходной транзистор одной схемы ИЛИ — НЕ закрыт, а у другой открыт. Таблица 3.1, называемая таблицей переходов, определяет закон функционирования этого триггера.

При $R = 1$ и $S = 0$ триггер устанавливается в нулевое состояние $Q = 0$; при $R = 0$ и $S = 1$ он устанавливается в единичное состояние $Q = 1$; при $R = S = 0$ триггер сохраняет состояние, в котором он находился до момента поступления на его входы нулевых сигналов. При $R = S = 1$ на прямом и инверсном выходах устанавливается нулевой сигнал. Триггерное кольцо превращается в два независимых инвертора, и при переходе к хранению ($R = S = 0$) триггер может устанавливаться в любое состояние. Поэтому такая комбинация входных сигналов запрещена.

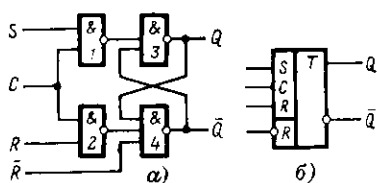
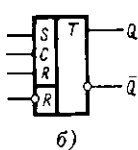


Рис. 3.8. Синхронизируемый одноктактный RS-триггер на элементах И — НЕ:

а — функциональная схема; б — условное обозначение



Функционирование RS-триггера можно описать выражением

$$Q(t+1) = S(t) \vee Q(t) \bar{R}(t), \quad (3.5)$$

причем $S(t)R(t) = 0$, а t — момент времени, предшествующий смене состояния.

Синхронизируемый одноктактный RS-триггер. На рис. 3.8 приведена схема синхронизируемого одноктактного RS-триггера на элементах И — НЕ. Здесь элементы 1 и 2 образуют схему входной логики асинхронного RS-триггера, построенного на элементах 3 и 4. Такие RS-триггеры имеют два информационных входа R и S и вход синхронизации C . Кроме того, триггер может иметь несинхронизируемые входы R и S . В этом случае функционирование триггера осуществляется либо под воздействием несинхронизируемых входов при $C = 0$, либо под воздействием синхронизируемых входов. В последнем случае на несинхронизируемых входах должны присутствовать сигналы, которые не влияют на состояние схемы.

Таблица 3.2 определяет переходы RS-триггера, построенного на элементах И — НЕ, для синхронизируемых входов R и S . Работа в соответствии с данной таблицей осуществляется при сигнале несинхронизируемого входа $R = 1$ и при $C = 1$.

Входная информация, представленная в парафазном коде, заносится в синхронизируемый одноктактный RS-триггер через элементы входной логики 1 и 2 в момент поступления сигнала синхронизации C . В отсутствие сигнала синхронизации триггер может быть установлен в состоянии 0 путем подачи на несинхронизируемый вход R сигнала $R = 0$.

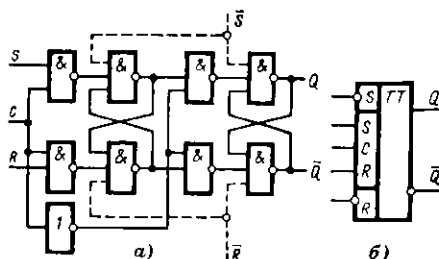
Двухтактный RS-триггер. Устойчивая работа одноктактных RS-триггеров в схеме с передачей информации между триггерами возможна только в случае, если занесение в триггер информации осуществ-

Таблица 3.1. Таблица переходов асинхронного RS-триггера на элементах ИЛИ — НЕ

R	S	Q	Примечание
0	0	Q	Хранение
0	1	1	Установка 1
1	0	0	Установка 0
1	1	—	Запрещено

Рис. 3.9. Двухтактный RS-триггер на элементах И – НЕ:

a – функциональная схема;
 $\bar{a}$ – условное обозначение



вляется после завершения передачи информации о прежнем его состоянии в другой триггер. Это достаточно просто обеспечивается при использовании двух серий находящихся в противофазе синхросигналов.

Такой принцип обмена информацией реализован в двухтактных RS-триггерах.

Простейшая схема двухвходового двухтактного RS-триггера показана на рис. 3.9. Она состоит из двух одноклапных RS-триггеров и инвертора в цепи синхронизации. При поступлении на вход RS-триггера сигнала $C = 1$ входная информация заносится в первый одноклапный RS-триггер, а второй при этом будет хранить информацию, относящуюся к предыдущему периоду представления. По окончании действия сигнала синхронизации, когда $C = 0$, а $C = 1$, первый RS-триггер перейдет в режим хранения, а второй примет то же состояние, что и первый. В результате к следующему такту на выходе двухтактного RS-триггера появится сигнал нового состояния. Таблица 3.3 задает закон функционирования такого двухтактного триггера. Такты t здесь задаются интервалами времени, в которые $C = 1$. Двухтактный триггер изменяет свои состояния только после окончания действия сигнала синхронизации $C = 1$ (переход в режим хранения информации). Поэтому из двухтактных триггеров можно строить произвольные схемы, в том числе подавать сигналы с выхода триггера на его вход.

Для установки триггера в состояние 0 или 1 без использования синхросигналов в схему вводят дополнительные входы R и S несинхронизируемой установки. Связи с этими входами показаны на рис. 3.9, a пунктиром. При подаче 0 на вход $\bar{R}(\bar{S})$ и 1 на вход $\bar{S}(\bar{R})$ оба одноклапных триггера устанавливаются в состояние 0(1). При подаче

Т а б л и ц а 3.2. Таблица переходов синхронизируемого одноклапного RS-триггера на элементах И – НЕ

R	S	Q	Примечание
1	1	Q	Хранение
1	0	1	Установка 1
0	1	0	Установка 0
0	0	—	Запрещено

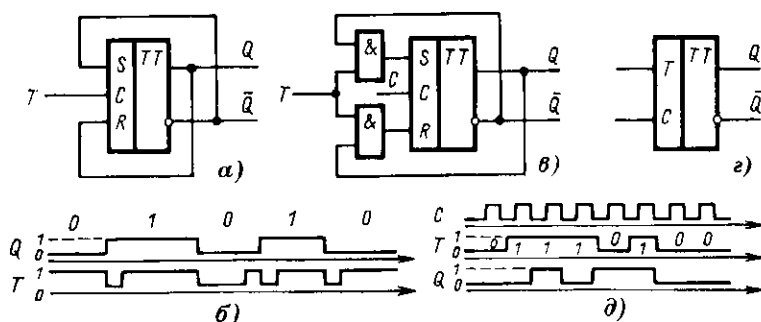


Рис. 3.10. *T*-триггер:

a – несинхронизируемый *T*-триггер; *б* – временная диаграмма работы несинхронизируемого *T*-триггера; *в* – синхронизируемый *T*-триггер; *г* – условное обозначение синхронизируемого *T*-триггера; *д* – временная диаграмма работы синхронизируемого *T*-триггера

1 на оба этих входа работа триггера осуществляется в соответствии с табл. 3.3.

Схема RS-триггеров составляет основу для построения других триггерных схем, таких, как *T*-, *D*- и *JK*-триггеры.

T-триггер. Триггер со счетным входом (*T*-триггер) в простейшем случае может быть построен с использованием двухтактного синхронизируемого RS-триггера. *T*-триггер должен реализовать функцию вида

$$Q(t+1) = Q(t)T(t) \vee \bar{Q}(t)\bar{T}(t). \quad (3.6)$$

Простейшая схема несинхронизируемого *T*-триггера представлена на рис. 3.10, *a*. В этой схеме поступление сигнала $T = 1$ приводит к записи в двухступенчатый RS-триггер состояния, противоположного ранее хранимому. При этом, так как триггер двухступенчатый, на его выходе сигнал изменится только по завершению действия сигнала $T = 1$, что исключает возникновение генерации в схеме с обратной связью. Можно считать, что в данной схеме единичный входной сигнал пред-

Таблица 3.3. Таблица переходов двухтактного RS-триггера на элементах И – НЕ

<i>t</i>		<i>t</i> + 1	Примечание
<i>R</i>	<i>S</i>	<i>Q</i>	
0	0	<i>Q</i> (<i>t</i>)	Хранение
1	0	0	Установка 0
0	1	1	Установка 1
1	1	—	Запрещено

составляется спадом сигнала $T = 1$, так как при любой продолжительности сигнала $T = 1$ изменение состояния T -триггера происходит только один раз — при снятии сигнала $T = 1$ (рис. 3.10, б).

При необходимости представлять потенциалом последовательность единиц на входе T -триггера используется синхронизируемая схема (рис. 3.10, в, г). Здесь единичный входной сигнал представляется высоким уровнем сигнала T при $C = 1$. Поэтому высоким уровнем сигнала T можно представить последовательность 1 (рис. 3.10, д). Запись в триггер происходит при $C = 1$, причем смена состояния происходит после окончания действия сигнала синхронизации $C = 1$. При $T = 1$ состояние триггера изменяется на противоположное, а при $T = 0$ состояние триггера не меняется. Временная диаграмма, поясняющая работу синхронизируемого T -триггера, показана на рис. 3.10, д.

D-триггер. Одним из самых широко употребляемых триггеров является интегральный D -триггер, который реализует функцию временной задержки. D -триггер имеет только режимы установки 1 и 0. В связи с этим несинхронизируемый D -триггер (рис. 3.11, а) не применяется, так как его выход будет просто повторять входной сигнал. Синхронизируемый одноктактный D -триггер (рис. 3.11, б) задерживает распространение входного сигнала на время паузы между синхросигналами (задержка на полпериода). Для задержки на период (на один такт) используется двухтактный D -триггер.

Вариант построения двухтактного D -триггера показан на рис. 3.11, в. Под действием синхросигнала информация, поступающая на выход D , принимается в RS -триггер, но на выходе Q появляется с задержкой на такт:

$$Q(t + 1) = D(t). \quad (3.7)$$

JK-триггер. Распространенным типом триггера в системах интегральных логических элементов является двухтактный JK -триггер, условное обозначение которого показано на рис. 3.12, а.

У рассматриваемого триггера имеются входы несинхронизируемой установки R и S , с помощью которых при $C = 0$ триггер может быть установлен в состояние 1 путем подачи $R = 1$ и $S = 0$ либо в состояние 0 путем подачи $R = 0$ и $S = 1$. При подаче сигналов $R = S = 1$,

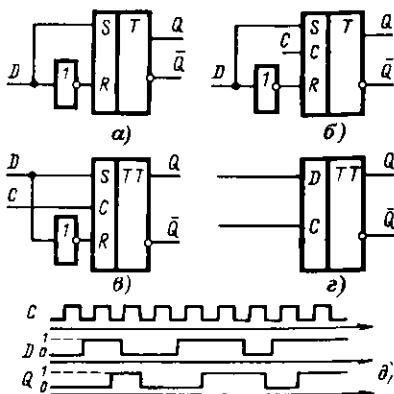


Рис. 3.11. D -триггер:

а — несинхронизируемый D -триггер; б — синхронизируемый одноктактный D -триггер; в — двухтактный D -триггер; г — условное обозначение двухтактного D -триггера; д — временная диаграмма работы двухтактного D -триггера.

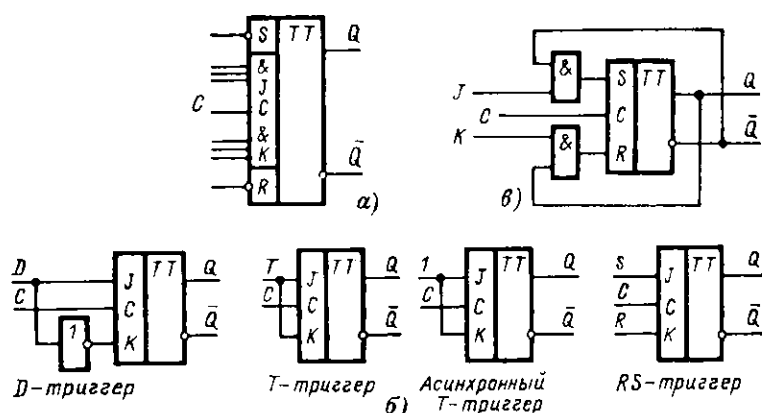


Рис. 3.12. *JK*-триггер:

a — условное обозначение двухвходового *JK*-триггера; *б* — способы использования *JK*-триггера; *в* — схема *JK*-триггера

не меняющих состояние схемы, работа триггера осуществляется под воздействием синхронизируемых входов. В этом случае функционирование триггера может быть описано табл. 3.4, причем такты t здесь, как и ранее, задаются сигналом $C = 1$.

Здесь $J = J_1 J_2 J_3$ и $K = K_1 K_2 K_3$. Входы J и K соответствуют входам установки в 1 и в 0 триггера. Однако в отличие от *RS*-триггера в *JK*-триггере сигналы 1 могут одновременно прийти на входы J и K . При этом состояние триггера изменяется на противоположное, т. е. при $J = K$ схема ведет себя как триггер со счетным входом.

Функцию переходов *JK*-триггера можно представить в виде булевой функции:

$$Q(t+1) = K(t)Q(t) \vee J(t)\bar{Q}(t) \quad (3.8)$$

при условии, что $RS = 1$.

JK-триггер удобен тем, что при различных вариантах под-

Таблица 3.4. Таблица переходов *JK*-триггера

t		$t+1$	Примечание
J	K	Q	
0	0	$Q(t)$	Хранение
0	1	0	Установка 0
1	0	1	Установка 1
1	1	$\bar{Q}(t)$	Инверсия

ключения его входов можно получить схемы, функционирующие как *RS*-, *D*- и *T*-триггеры (рис. 3.12, б). Вариант построения схемы *JK*-триггера дан на рис. 3.12, в.

3.5. РЕГИСТРЫ

Регистром называется устройство, предназначенное для запоминания слова, а также для выполнения над словом некоторых логических преобразований. Регистр представляет собой совокупность триггеров, число которых соответствует числу разрядов в слове, и вспомогательных схем, обеспечивающих выполнение некоторых операций, среди которых могут быть:

установка регистра в 0 («сброс»);

прием слова из другого устройства (регистра, сумматора и т. д.);

передача слова в другой регистр;

сдвиг слова вправо или влево на требуемое число разрядов;

преобразование последовательного кода слова в параллельный и наоборот;

поразрядные логические операции.

На рис. 3.13 показана схема приема информации в регистр. Для этого используется парафазная передача информации, при которой на одном из входов триггера обязательно присутствует сигнал 1, устанавливающий триггер в нужное состояние независимо от той информации, которая в нем хранилась.

Как правило, операция передачи кода с регистра объединяется с операцией приема этого кода в другой регистр. Это показано на рис. 3.14, где информация передается из регистра *Pz1* в регистр *Pz2* парафазным кодом под действием сигнала *ПрРz2*, управляющего приемом информации в регистр.

Операция сдвига кода — это перемещение в регистре всех

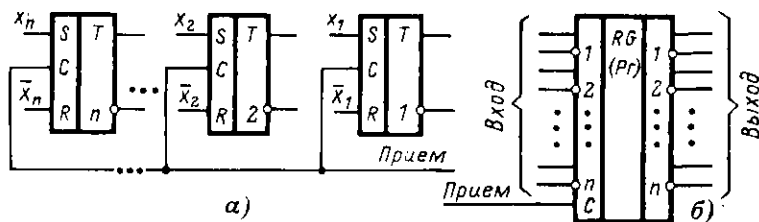


Рис. 3.13. Прием информации в регистр в парафазном коде:

а — функциональная схема регистра; б — условное обозначение регистра

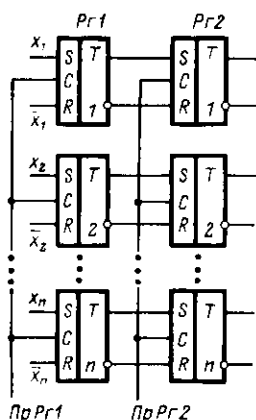


Рис. 3.14. Схема передачи слова из одного регистра в другой

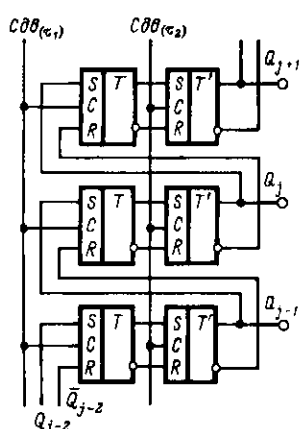


Рис. 3.15. Сдвигающий регистр на одноклапных RS-триггерах

разрядов кода слова на одинаковое число разрядов влево или вправо. В этом случае разряды слова, вышедшие из разрядной сетки регистра влево (или вправо), теряются, а в освободившиеся при сдвиге разряды регистра записываются нули.

Функциональная схема сдвигающего регистра на RS-триггерах приведена на рис. 3.15.

Сигналом $Cdv(t_1)$ код в регистре передается со сдвигом с выходных (основных) триггеров на входные (дополнительные) триггеры соседних разрядов, а потом сигналом $Cdv(t_2)$ переписывается с входных триггеров на выходные триггеры в тех же самых разрядах. Сигналы $Cdv(t_1)$ и $Cdv(t_2)$ во времени не совпадают.

3.6. ДЕШИФРАТОРЫ

Дешифратором называется комбинационная схема с несколькими входами и выходами, преобразующая код, подаваемый на входы, в сигнал на одном из выходов. Если на входы дешифратора подаются двоичные переменные, то на одном из выходов дешифратора вырабатывается сигнал 1, а на остальных выходах сохраняются сигналы 0.

В общем случае дешифратор с n входами имеет 2^n выходов (рис. 3.16), так как n -разрядный код входного слова может принимать 2^n различных значений и каждому из этих значений соответствует сигнал 1 на одном из выходов дешифратора.

На выходах дешифратора с номерами $0, 1, 2, \dots, 2^n - 1$ вырабатываются значения булевых функций соответственно

$$\begin{aligned} f_0 &= \bar{x}_n \bar{x}_{n-1} \dots \bar{x}_2 \bar{x}_1; \\ f_1 &= \bar{x}_n \bar{x}_{n-1} \dots \bar{x}_2 x_1; \\ f_2 &= \bar{x}_n \bar{x}_{n-1} \dots x_2 \bar{x}_1; \\ &\dots \dots \dots \\ f_{2^n-1} &= x_n x_{n-1} \dots x_2 x_1. \end{aligned}$$

Дешифраторы устанавливаются в схемах ЭВМ на выходах регистров или счетчиков и служат для преобразования кода слова, находящегося в регистре (в счетчике), в управляющий сигнал на одном из выходов дешифратора.

На рис. 3.16, а показан способ построения дешифратора на примере схемы дешифратора с инверсными выходами для трехразрядного входного слова. Схема представляет собой набор из восьми трехвходовых элементов И – НЕ, на входы которых поданы все возможные комбинации прямых и инверсных значений разрядов слова.

Если каждый выходной элемент дешифратора имеет вход для сигнала синхронизации, то такая схема называется деши-

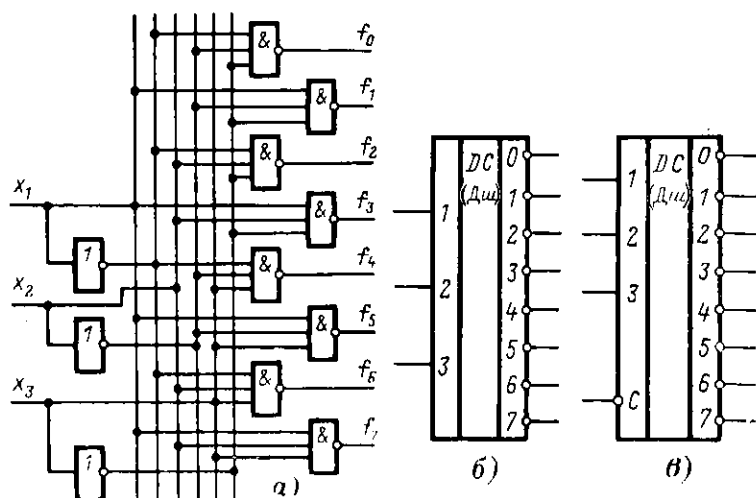


Рис. 3.16. Дешифратор с инверсными выходами:

а – функциональная схема; б – условное обозначение дешифратора; в – условное обозначение дешифратора со входом синхронизации

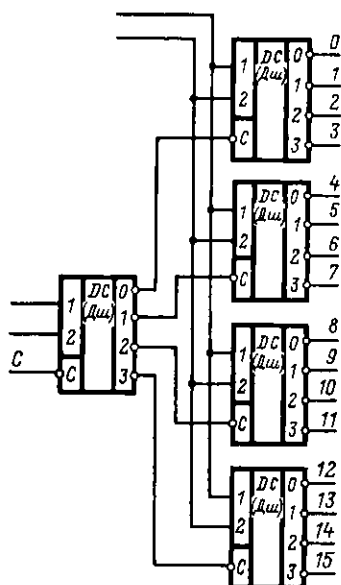


Рис. 3.17. Каскадный дешифратор

фратором со входом синхронизации (рис. 3.16, в). При $C = 1$ все выходы данного дешифратора будут иметь единичное значение, а при $C = 0$ он будет работать также, как схема на рис. 3.16, б. Такого вида схемы выпускаются в составе комплексов интегральных логических элементов.

Из логических элементов, являющихся дешифраторами, можно строить дешифраторы на большее число входов. При этом, как правило, используются дешифраторы с инверсными входами синхронизации. Каскадное включение таких схем позволяет легко наращивать число дешифрируемых переменных. Принцип построения схем нетрудно понять, обратившись к рис. 3.17. Здесь показан дешифратор на четыре входа с инверсными выходами, построенный из интегральных элементов, реализующих схемы двухвходовых дешифраторов.

3.7. МУЛЬТИПЛЕКСОРЫ

Мультиплексором называется схема, осуществляющая передачу сигналов с одной из входных линий в выходную. Выбор входной (информационной) линии производится кодом, поступающим на управляющие входы мультиплексора. Мультиплексор с k управляющими входами $x_{k-1}, x_{k-2}, \dots, x_1, x_0$ имеет 2^k информационных входов $D_0, D_1, \dots, D_{2^k-1}$.

Набор сигналов, поступающих на управляющие входы, задает двоичное число вида

$$X^* = x_{k-1} \cdot 2^{k-1} + x_{k-2} \cdot 2^{k-2} + \dots + x_1 \cdot 2^1 + x_0^*.$$

где $x_i^* = (0, 1)$ — значение входа x_i . Выходной сигнал мультиплек-

сора повторяет сигнал информационного входа D с номером X^* .

Работа мультиплексора может иллюстрироваться контактной схемой (рис. 3.18, а). Условное обозначение мультиплексора, коммутирующего 2^k линий, показано на рис. 3.18, б. Пример функциональной схемы мультиплексора с двумя управляющими входами дан на рис. 3.18, в.

Функция, реализуемая мультиплексором, может быть представлена в виде

$$y = \bar{x}_{k-1}\bar{x}_{k-2}\dots\bar{x}_0D_0 \vee \bar{x}_{k-1}\bar{x}_{k-2}\dots x_0D_1 \vee \dots \vee x_{k-1}x_{k-2}\dots x_0D_{2^k-1} = R_0D_0 \vee R_1D_1 \vee \dots \vee R_jD_j \vee \dots \vee R_{2^k-1}D_{2^k-1}, \quad (3.9)$$

где R_j — конъюнкция, равная 1 на наборе значений переменных $x_{k-1} x_{k-2} \dots x_0$ с номером j (т. е. на наборе, представляющем в двоичном виде число j).

Функция эта нелинейна, немонотонна и не самодвойственна и, согласно теореме об ослабленной функциональной полноте, позволяет при использовании булевых констант 1 и 0 путем суперпозиций реализовать любую булеву функцию [22].

Синтез комбинационных схем на мультиплексорах удобно осуществлять с помощью теоремы разложения [22]. Эта теоре-

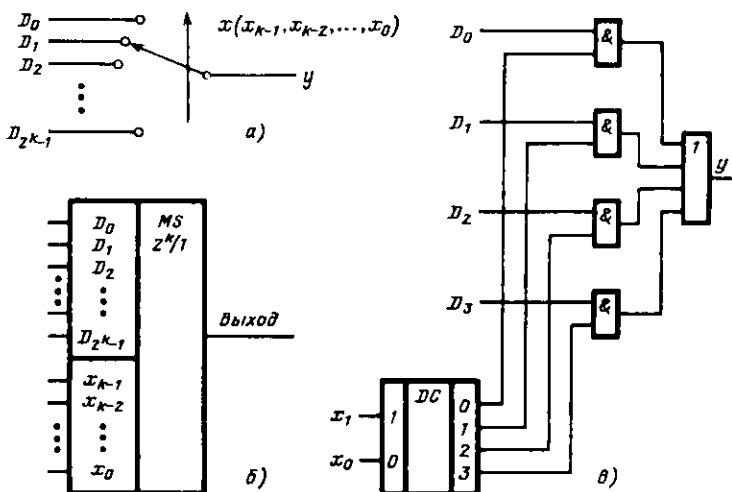


Рис. 3.18. Мультиплексор:

а — контактная модель; б — условное обозначение; в — функциональная схема

ма позволяет любую булеву функцию $F(x_{n-1}, x_{n-2}, \dots, x_1, x_0)$ разложить по i переменным $x_{n-1}, x_{n-2}, \dots, x_{n-i}$ и представить как

$$F = \bar{x}_{n-1}\bar{x}_{n-2}\dots\bar{x}_{n-i}F^0 \vee \bar{x}_{n-1}\bar{x}_{n-2}\dots x_{n-i}F^1 \vee \dots \vee x_{n-1}x_{n-2}\dots x_{n-i}F^{2^i-1} = R_0F^0 \vee R_1F^1 \vee \dots \vee R_jF^j \vee \dots \vee R_{2^i-1}F^{2^i-1}, \quad (3.10)$$

где R_j — конъюнкция, равная 1 на наборе значений переменных $x_{n-1}, x_{n-2}, \dots, x_{n-i}$ с номером j , а F^j — функция, полученная из F подстановкой в нее набора значений переменных $x_{n-1}, x_{n-2}, \dots, x_{n-i}$ с номером j .

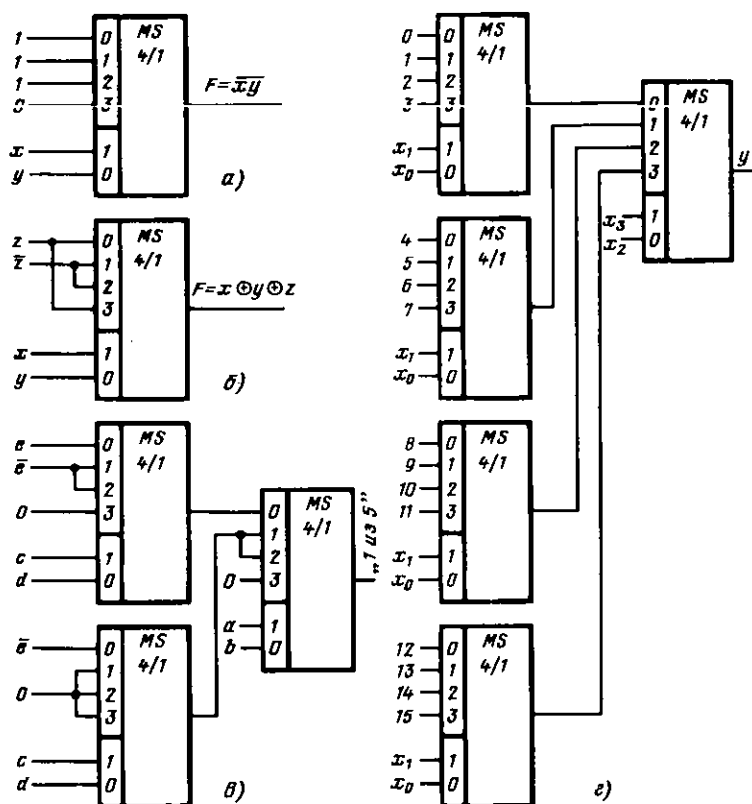


Рис. 3.19. Схемы из мультиплексоров:

а — И — НЕ; б — сложение по модулю 2; в — двухкаскадное мультиплексирование; г — многокаскадный мультиплексор.

Сравнивая (3.9) и (3.10), легко заметить, что на основе разложения булевой функции F по k переменным, можно строить реализующую ее схему на мультиплексоре. Для этого необходимо подать на входы мультиплексора $D_0, D_1, \dots, D_{2^k-1}$ сигналы, представляющие функции разложения $F^0, F^1, \dots, F^{2^k-1}$, а на управляющие входы — сигналы k переменных, по которым выполнялось разложение. Если функция F имеет $n = k$ аргументов, то функции разложения $F^0, F^1, \dots$ — константы и на вход D_j мультиплексора подается константа, равная значению F на наборе с номером j . При числе аргументов функции F , равном $(k + 1)$, функции разложения могут иметь вид x , либо $\bar{x}$, либо 0, либо 1. Подавая на входы $D_0, D_1, \dots$ мультиплексора такие сигналы, можно реализовать на одной микросхеме мультиплексора любую функцию $k + 1$ переменной. Если $n > k + 1$, то функции разложения могут зависеть от нескольких переменных и для реализации их потребуются схемы, которые можно построить, разлагая функции разложения так же, как исходную по k переменным, и т. д.

На рис. 3.19 даны примеры построенных на мультиплексорах схем, реализующих функции И — НЕ, сложения по модулю 2 и функцию выделения наборов пяти переменных, содержащих одну 1.

Корпус (кристалл) микросхемы может содержать несколько мультиплексорных схем. Выход мультиплексора может быть как прямым, так и инверсным по отношению к сигналам $D_0, D_1, \dots$; мультиплексор может быть снабжен входом выборки (синхронизации). Число управляющих входов у микросхем мультиплексоров обычно находится в пределе 1—4. При необходимости построения мультиплексоров с большим числом входов строятся каскадные схемы. Пример такой схемы дан на рис. 3.19, г.

3.8. СЧЕТЧИКИ

Счетчиком называется узел ЭВМ, предназначенный для подсчета числа входных сигналов. Счетчики используются в ЭВМ для образования последовательностей адресов команд, для подсчета числа циклов выполнения операций и т. п.

Счетчики принято подразделять на суммирующие, вычитающие и реверсивные.

На рис. 3.20 показаны схема несинхронизируемого четырехразрядного двоичного суммирующего счетчика с последовательным переносом и временная диаграмма его работы. Таблица 3.5 показывает состояния, в которых находятся триггеры счетчика при воздействии серии входных сигналов $x_{сч}$.

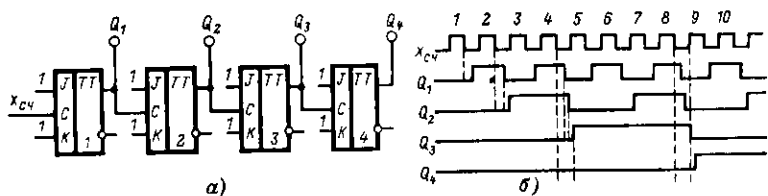


Рис. 3.20. Несинхронизируемый двоичный счетчик с последовательным переносом:

a — функциональная схема; *б* — временная диаграмма

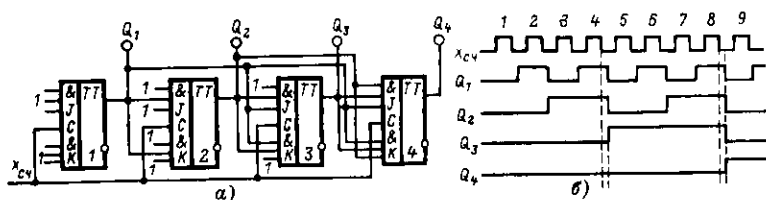


Рис. 3.21. Синхронизируемый двоичный счетчик с параллельным переносом:

a — функциональная схема; *б* — временная диаграмма

Здесь на входы J и K JK -триггеров подаются сигналы 1. Выход каждого предыдущего триггера Q_{n-1} соединен с входом синхронизации C_n последующего триггера. Каждый JK -триггер в счетчике выполняет функцию несинхронизируемого триггера со счетным входом. По спаду единичного выходного сигнала изменяется состояние триггера младшего разряда счетчика на противоположное (т. е. реализуется сложение по модулю 2

Таблица 3.5. Таблица состояний двоичного счетчика

$x_{сч}$	Q_4	Q_3	Q_2	Q_1	$x_{сч}$	Q_4	Q_3	Q_2	Q_1
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1

в этом разряде). В последующих разрядах аналогичное действие производит сигнал переноса.

Обычно счетчик имеет цепь установки в нулевое состояние (сброс триггеров в 0). Однако начальное состояние счетчика не обязательно нулевое. Начальное состояние может устанавливаться передачей в счетчик кода некоторого числа, и с него уже будет начинаться операция счета единиц. Такой режим работы счетчика необходим, например, при образовании последовательности адресов команд при заданном исходном адресе. С ростом разрядности счетчика понижается предельная частота его работы. Это объясняется тем, что с ростом разрядности счетчика n будет возрастать задержка поступления сигнала на вход C некоторого j -го разряда относительно времени поступления входного сигнала $x_{сч}$ на вход C младшего разряда счетчика. Из временной диаграммы видно, что такая задержка может привести к искажению информации в счетчике (моменты времени 4 и 8). Для повышения быстродействия счетчики выполняют с параллельным переносом.

На рис. 3.21 изображена функциональная схема счетчика с параллельным переносом. Отличительной особенностью данной схемы является то, что выходы всех предшествующих разрядов подаются на входы J и K j -го триггера. Длительность переходного процесса в таком счетчике равна длительности переключения одного разряда. Из схемы видно, что с возрастанием порядкового номера триггера увеличивается число входов в элементах И JK -триггеров. А так как число входов J и K и нагрузочная способность выходов триггеров ограничены, то и разрядность счетчика с параллельным переносом невелика и равна обычно четырем. Поэтому при числе разрядов счетчика, большем максимального числа входов J и K , счетчик разбивают на группы и внутри каждой группы строят цепи параллельного переноса.

Такой подход удобен и потому, что счетчики реализуют в виде интегральной микросхемы в отдельном корпусе. В этом случае при последовательном переносе просто осуществляется увеличение разрядности счетчика.

Вычитающие счетчики и реверсивные (настраиваемые на суммирование или вычитание) строятся аналогичным образом.

3.9. СУММАТОРЫ

Сумматором называется узел ЭВМ, выполняющий арифметическое суммирование кодов чисел. Обычно сумматор представляет собой комбинацию одноразрядных суммирующих схем.

При сложении двух чисел в каждом разряде производится сложение трех цифр: цифры данного разряда первого слагаемого, цифры данного разряда второго слагаемого и цифры (1 или 0) переноса из соседнего младшего разряда. В результате сложения для каждого разряда получаются цифра суммы для этого разряда и цифра (1 или 0) переноса в следующий старший разряд.

В табл. 3.6 приведены варианты, возникающие при сложении двух двоичных чисел.

По табл. 3.6 можно составить булевы функции для описания одноразрядного сумматора — устройства, вырабатывающего на выходе сигналы суммы и переноса при поступлении на входы двух цифр слагаемых и цифры переноса из предыдущего, младшего разряда:

$$S_i = a_i \bar{b}_i \bar{P}_i \vee \bar{a}_i b_i \bar{P}_i \vee \bar{a}_i \bar{b}_i P_i \vee a_i b_i P_i; \quad (3.11)$$

$$P_{i+1} = a_i \bar{b}_i \bar{P}_i \vee \bar{a}_i \bar{b}_i P_i \vee a_i b_i \bar{P}_i \vee a_i b_i P_i, \quad (3.12)$$

где P_i — цифра переноса из предыдущего (младшего разряда); a_i , b_i — цифры слагаемых в данном разряде; S_i — сумма; P_{i+1} — цифра переноса в старший разряд.

Выражение для цифры переноса в следующий разряд может быть приведено к более простому виду:

$$P_{i+1} = a_i b_i \vee a_i P_i \vee b_i P_i. \quad (3.13)$$

Преобразуя с помощью правил булевой алгебры выражения для цифры суммы и цифры переноса, можно получать различные соотношения, которым будут соответствовать варианты построения схем полных сумматоров.

Таблица 3.6.

Цифры переноса из предыдущего разряда P_i	Первое слагаемое a_i	Второе слагаемое b_i	Сумма S_i	Цифра переноса в старший разряд P_{i+1}
0	0	0	0	0
0	1	0	1	0
0	0	1	1	0
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1
1	0	1	0	1
1	1	1	1	1

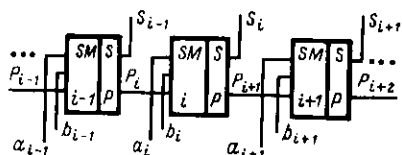
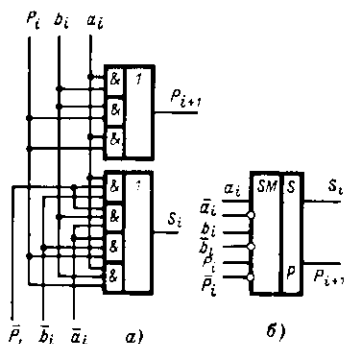


Рис. 3.23. Параллельный сумматор с последовательным переносом

Рис. 3.22. Комбинационный одно-разрядный сумматор:

a — функциональная схема; *б* — условное обозначение

Функциональная схема одnorазрядного комбинационного сумматора, реализующего соотношения (3.11) и (3.13), показана на рис. 3.22.

Параллельный (многоразрядный) сумматор может быть составлен из одnorазрядных сумматоров, число которых равно числу разрядов слагаемых, путем соединения выхода, на котором формируется сигнал переноса данного разряда, с входом для сигнала переноса соседнего старшего разряда (рис. 3.23).

После того как выработан результат сложения на выходах комбинационных схем формирования суммы, он обычно запоминается в отдельном триггерном регистре.

Быстродействие одnorазрядного комбинационного сумматора характеризуется временем установления выходных сигналов суммы и переноса после установления сигналов на входах сумматора. Наиболее важным является время распространения сигнала переноса в одnorазрядном сумматоре, так как при образовании многоразрядного сумматора из одnorазрядных схем сигнал переноса может распространяться от разряда к разряду. Это время определяется временами задержек в логических элементах и количеством последовательно включенных элементов в схеме распространения сигнала переноса.

По характеру распространения переноса различают следующие виды сумматоров: с поразрядным последовательным переносом, с параллельным (одновременным) переносом, с групповым переносом.

Сумматоры с поразрядным последовательным переносом. В сумматорах этого типа (рис. 3.23) перенос распространяется последовательно от разряда к разряду по мере образования цифры суммы в каждом отдельном разряде. При наиболее неблагоприятных условиях для распространения переноса, например при сложении чисел $11\dots11$ и $00\dots001$, произойдет «пробег» 1 переноса через весь сумматор от самого младшего разряда к самому старшему. Поэтому в худшем случае время распространения переноса

$$T_{\text{пер}} = \tau_1 n, \quad (3.14)$$

где τ_1 — время распространения переноса в одном разряде; n — число разрядов сумматора. Данный тип сумматора наиболее прост с точки зрения схемы цепей распространения переноса, но имеет сравнительно низкое быстродействие.

Сумматоры с параллельным переносом. Можно построить сумматор, в котором сложение выполняется как поразрядная операция и на распространение переноса не требуется дополнительного времени.

Затраты оборудования на построение сумматора такого типа, особенно при большом числе разрядов, настолько велики, что в чистом виде он практически не находит применения. Принцип параллельного формирования переноса используется в сумматорах с групповым переносом.

Сумматоры с групповым переносом. Сумматор разбивается на несколько групп примерно равной длины. Сигнал переноса, поступающий на вход младшего разряда группы, при наличии условий распространения переноса во всех разрядах данной группы передается на вход младшего разряда соседней, более старшей группы в обход данной группы.

Схема формирования сигнала переноса в младшем разряде каждой группы дополняется для этой цели схемой И, реализующей булеву функцию

$$P_{\text{уск}} = P_i c_i c_{i+1} \dots c_{i+k-1}, \quad (3.15)$$

где $P_{\text{уск}}$ — сигнал ускорения переноса; P_i — сигнал переноса в младший разряд группы, содержащей k разрядов, $c_i, \dots, c_{i+k-1}$ — условия распространения переноса в разрядах группы ($c_i = a_i b_i \vee a_i \bar{b}_i$).

В таком сумматоре максимальная задержка распространения переноса определяется задержкой его в младшей, старшей группах и в цепях обхода остальных групп.

Максимальная задержка сигнала переноса может быть уменьшена, если при разбиении сумматора на группы использовать параллельное (одновременное) формирование переноса внутри групп.

Современные системы элементов содержат микросхемы сумматоров и микросхемы формирования сигналов группового переноса.

Глава четвертая

ПРИНЦИПЫ ПОСТРОЕНИЯ УСТРОЙСТВ ПАМЯТИ

4.1. ОБЩИЕ СВЕДЕНИЯ И КЛАССИФИКАЦИЯ УСТРОЙСТВ ПАМЯТИ

Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации. Отдельные устройства, входящие в эту совокупность, называют

запоминающими устройствами или памятью того или иного типа.

Оба эти термина в настоящее время стали почти синонимами. Однако термин *запоминающее устройство* (ЗУ) обычно употребляют, когда речь идет о принципе построения некоторого устройства памяти (например, полупроводниковые ЗУ, ЗУ на магнитных дисках и т. д.), а термин *память* — когда хотят подчеркнуть выполняемую устройством памяти логическую функцию или место расположения в составе оборудования ЭВМ (например, оперативная память, внешняя память и т. д.).

Производительность и вычислительные возможности ЭВМ в значительной степени определяются составом и характеристиками ее ЗУ. В составе ЭВМ используется одновременно несколько типов ЗУ (несколько типов памяти), отличающихся принципом действия, характеристиками и назначением.

Основными операциями в памяти в общем случае являются занесение информации в память — *запись* и выборка информации из памяти — *считывание*. Обе эти операции называются *обращением к памяти* или, подробнее, *обращением при считывании и обращением при записи*.

При обращении к памяти производится считывание или запись некоторой единицы данных — различной для устройств разного типа. Такой единицей может быть, например, байт, машинное слово или блок данных (см. § 11.4).

Важнейшими характеристиками отдельных устройств памяти (запоминающих устройств) являются емкость памяти, удельная емкость, быстродействие.

Емкость памяти определяется максимальным количеством данных, которые могут в ней храниться. Емкость измеряется в двоичных единицах (битах), машинных словах, но большей частью в байтах (1 байт = 8 бит). При этом часто емкость памяти выражают через число $K = 1024$: Кбит (килобит), Кслов (килослов) или Кбайт (килобайт). При этом 1024 Кбайт обозначают как 1 Мбайт (мегабайт).

Удельная емкость есть отношение емкости ЗУ к его физическому объему.

Быстродействие памяти определяется продолжительностью операции обращения, т. е. временем, затрачиваемым на поиск нужной единицы информации в памяти и на ее считывание (*время обращения при считывании*), или временем на поиск места в памяти, предназначенного для хранения данной единицы информации, и на ее запись в память (*время обращения при записи*).

В некоторых устройствах памяти считывание информации сопровождается ее разрушением (стиранием). В таком случае

цикл обращения должен содержать операцию восстановления (регенерации) считанной информации на прежнем месте в памяти.

Продолжительность обращения к памяти (время цикла памяти) при считывании

$$t_{\text{обр}}^{\text{счит}} = t_{\text{дост}}^{\text{счит}} + t_{\text{счит}} + t_{\text{рег}}, \quad (4.1)$$

где $t_{\text{дост}}^{\text{счит}}$ — время доступа, определяющееся промежутком времени между моментом начала операции обращения при считывании до момента, когда становится возможным доступ к данной единице информации; $t_{\text{счит}}$ — продолжительность самого физического процесса считывания, т. е. процесса обнаружения и фиксации состояний соответствующих запоминающих элементов или участков поверхности носителя информации; $t_{\text{рег}}$ — время, затрачиваемое на регенерацию разрушенной при считывании информации.

В некоторых ЗУ реализуется считывание без разрушения информации и $t_{\text{рег}} = 0$.

Продолжительность обращения (время цикла) при записи

$$t_{\text{обр}}^{\text{зап}} = t_{\text{дост}}^{\text{зап}} + t_{\text{подг}} + t_{\text{зап}}, \quad (4.2)$$

где $t_{\text{дост}}^{\text{зап}}$ — время доступа при записи, т. е. время от момента начала обращения при записи до момента, когда становится возможным доступ к запоминающим элементам (или участкам поверхности носителя), в которые производится запись; $t_{\text{подг}}$ — время подготовки, расходуемое на приведение в исходное состояние запоминающих элементов или участков поверхности носителя информации для записи заданной единицы информации; $t_{\text{зап}}$ — время занесения информации, т. е. изменения состояния запоминающих элементов (участков поверхности носителя).

Большой частью

$$t_{\text{дост}}^{\text{счит}} = t_{\text{дост}}^{\text{зап}} = t_{\text{дост}}. \quad \text{А.}$$

В качестве продолжительности цикла обращения к памяти принимается величина

$$t_{\text{обр}} = \max(t_{\text{обр}}^{\text{счит}}, t_{\text{обр}}^{\text{зап}}). \quad (4.3)$$

По принципу действия устройства памяти можно разделить на: а) электронные (полупроводниковые), в которых в качестве запоминающих элементов используются полупроводниковые электронные элементы; б) магнитные с неподвижными

магнитными запоминающими элементами; в) магнитомеханические с движущимся магнитным носителем информации.

В зависимости от реализуемых в памяти операций обращения различают: а) память с произвольным обращением (возможны считывание и запись данных в память); б) память только для считывания информации («постоянная» или «односторонняя» память). Запись информации в постоянную память производится в процессе ее изготовления или настройки¹.

По способу организации доступа различают устройства памяти с непосредственным (произвольным), с прямым (циклическим) и последовательным доступами.

В памяти с *непосредственным (произвольным) доступом* время доступа, а поэтому и цикл обращения не зависят от места расположения участка памяти, с которого производится считывание или в который записывается информация. В большинстве случаев непосредственный доступ реализуется при помощи электронных, электрических или электромагнитных процессов коммутации в схемах ЗУ. В подобных memories цикл обращения обычно составляет 1–2 мкс или даже всего несколько сотен или десятков наносекунд. Число разрядов, считываемых или записываемых в памяти с непосредственным доступом параллельно во времени за одну операцию обращения, называется *шириной выборки*.

В двух других типах памяти используются более медленные электромеханические процессы. В устройствах *памяти с прямым доступом*, к которым относятся устройства с магнитными барабанами и дисками, благодаря непрерывному вращению носителя информации возможность обращения к некоторому участку носителя для считывания или записи циклически повторяется. В такой памяти время доступа составляет обычно от нескольких долей секунды до нескольких десятков миллисекунд.

В *памяти с последовательным доступом* производится последовательный просмотр участков носителя информации, пока нужный участок носителя не займет некоторое исходное положение. Характерным примером является ЗУ на магнитных лентах. Время доступа может в неблагоприятных случаях расположения информации достигнуть нескольких минут.

Запоминающие устройства различаются также по выполняемым в ЭВМ функциям, зависящим, в частности, от места расположения ЗУ в структуре ЭВМ.

¹ Эти типы памяти соответствуют терминам RAM (random – access memory – память с произвольным обращением) и ROM (read – only memory – память только для считывания).

Требования к емкости и быстродействию памяти являются противоречивыми. Чем больше быстродействие, тем технически труднее достигается и дороже обходится увеличение емкости памяти. Стоимость памяти составляет значительную часть общей стоимости ЭВМ. Поэтому память ЭВМ организуется в виде иерархической структуры запоминающих устройств, обладающих различными быстродействием и емкостью (рис. 4.1). В общем случае ЭВМ содержит *сверхоперативную память* (СОП) или *местную память*, *оперативную память* (ОП), *память с прямым доступом* на магнитных барабанах и на магнитных дисках, *память с последовательным доступом* на магнитных лентах. Порядок перечисления устройств соответствует убыванию их быстродействия и возрастанию емкости. Каждый уровень иерархии может содержать несколько экземпляров (модулей) соответствующих устройств для получения нужной емкости данного уровня памяти. На рис. 4.1 сплошными и штриховыми линиями показаны соответственно обычно и сравнительно редко реализуемые пути передачи данных между отдельными ступенями иерархической памяти. Иерархическая структура памяти позволяет экономически эффективно сочетать хранение больших объемов информации с быстрым доступом к информации в процессе обработки.

Оперативной или основной памятью (ОП) называют устройство, которое служит для хранения информации (данных, программ, промежуточных и конечных результатов обработки), не-

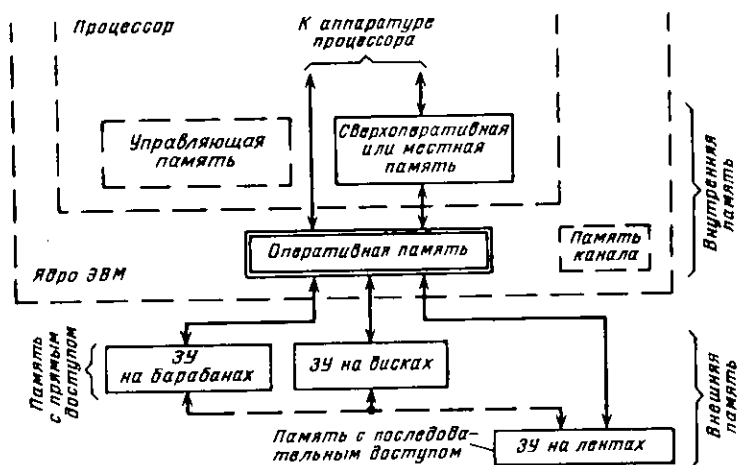


Рис. 4.1. Иерархическая структура памяти ЭВМ

посредственно используемой в процессе выполнения операций в арифметическо-логическом устройстве и устройстве управления процессора.

В процессе обработки информации осуществляется тесное взаимодействие процессора и ОП. Из ОП в процессор поступают команды программы и операнды, над которыми производятся предусмотренные командой операции, а из процессора в ОП направляются для хранения промежуточные и конечные результаты обработки.

Характеристики ОП непосредственно влияют на основные показатели ЭВМ и в первую очередь на скорость ее работы. Оперативная память высокопроизводительных ЭВМ должна иметь емкость несколько миллионов байт и цикл обращения около 1 мкс (и менее). Запоминающие устройства ОП выполняются на магнитных (ферритовых) сердечниках, тонких магнитных пленках, а в последнее время — в основном на больших интегральных микросхемах (полупроводниковые ЗУ).

В ряде случаев быстроедействие ОП оказывается недостаточным и в состав машины приходится включать СОП (на несколько сотен машинных слов) с циклом обращения порядка десятков или сотен наносекунд. Такие СОП выполняются на интегральных микросхемах разной степени интеграции. Быстродействие СОП должно соответствовать скорости работы арифметическо-логических и управляющих устройств ЭВМ. Сверхоперативная память используется для временного хранения обрабатываемых участков программы, в качестве рабочих ячеек программы, индексных регистров, для хранения служебной информации, используемой при управлении вычислительным процессом. Она выполняет роль согласующего звена между быстродействующими логическими устройствами процессора и более медленной ОП.

В качестве ОП и СОП используются быстродействующие ЗУ с произвольным обращением и непосредственным доступом.

Понятие оперативной памяти выше определялось, исходя из выполняемых ею функций. Поскольку всегда в качестве ОП (как, впрочем, и СОП) используются ЗУ с произвольным обращением и непосредственным доступом, то часто последние независимо от выполняемых функций называют оперативными памятьями (оперативными ЗУ).

Обычно емкость ОП оказывается недостаточной для хранения всех необходимых данных в ЭВМ. Поэтому ЭВМ содержит в своем составе несколько ЗУ с прямым доступом на дисках (реже — на барабанах) (емкость одного ЗУ на дисках порядка 10—300 Мбайт) и несколько ЗУ с последовательным доступом

на магнитных лентах (емкость одного ЗУ 20—40 Мбайт).

Оперативная память вместе с СОП и некоторыми другими специализированными памятьями процессора образуют *внутреннюю память ЭВМ* (рис. 4.1). Электромеханические ЗУ часто связаны с процессором не непосредственно, а через ОП. Эти устройства образуют *внешнюю память ЭВМ*, а сами они поэтому называются *внешними запоминающими устройствами* (ВЗУ).

Современные ЭВМ содержат ряд специализированных быстродействующих памяти: памяти каналов, ключей защиты памяти, отдельных типов терминалов (дисплеев и др.), различные буферные памяти для промежуточного хранения информации при обмене ею между устройствами машины, работающими с различными скоростями. Наряду с этим используются также постоянные памяти, в основном для хранения микропрограмм, а в специализированных ЭВМ — и для хранения основных программ.

Запоминающее устройство любого типа состоит из *запоминающего массива*, хранящего информацию, и блоков, служащих для поиска в массиве, записи и считывания (а в ряде случаев и для регенерации) информации.

4.2. АДРЕСНАЯ, АССОЦИАТИВНАЯ И СТЕКОВАЯ ОРГАНИЗАЦИИ ПАМЯТИ

Запоминающее устройство, как правило, содержит множество одинаковых запоминающих элементов, образующих запоминающий массив (ЗМ). Массив разделен на отдельные ячейки; каждая из них предназначена для хранения двоичного кода, число разрядов в котором определяется шириной выборки памяти (в частности, это может быть одно, половина или несколько машинных слов). Способ организации памяти зависит от методов размещения и поиска информации в запоминающем массиве. По этому признаку различают адресную, ассоциативную и стековую (магазинную) памяти.

Адресная память. В памяти с адресной организацией размещение и поиск информации в ЗМ основаны на использовании адреса хранения слова (числа, команды и т. п.), адресом служит номер ячейки ЗМ, в которой это слово размещается.

При записи (или считывании) слова в ЗМ инициирующая эту операцию команда должна указывать адрес (номер ячейки), по которому производится запись (считывание).

Типичная структура адресной памяти, показанная на рис. 4.2, содержит запоминающий массив из N n -разрядных ячеек и его аппаратное обрамление, включающее регистр адреса PzA ,

имеющий k ($k \geq \log_2 N$) разрядов, информационный регистр $P_2И$, блок адресной выборки $БАВ$, блок усилителей считывания $БУС$, блок разрядных усилителей-формирователей сигналов записи $БУЗ$ и блок управления памятью $БУП$.

По коду адреса в $P_2А$ $БАВ$ формирует в соответствующей ячейке памяти сигналы, позволяющие произвести в ячейке считывание или запись слова.

Цикл обращения к памяти инициируется поступлением в $БУП$ извне сигнала *Обращение*. Общая часть цикла обращения включает в себя прием в $P_2А$ с шины адреса $ША$ адреса обращения и прием в $БУП$ и расшифровку управляющего сигнала *Операция*, указывающего вид запрашиваемой операции (считывание или запись).

Далее при считывании $БАВ$ дешифрирует адрес, посылает сигналы считывания в заданную адресом ячейку $ЗМ$, при этом код записанного в ячейке слова считывается усилителями считывания $БУС$ и передается в $P_2И$. Затем в памяти с разрушающим считыванием (при считывании все запоминающие элементы ячейки устанавливаются в нулевое состояние) производится регенерация информации в ячейке путем записи в нее из $P_2И$ считанного слова. Операция считывания завершается выдачей слова из $P_2И$ на выходную информационную шину $ШИВх$.

При записи помимо выполнения указанной выше общей части цикла обращения производится прием записываемого слова с входной информационной шины $ШИВх$ в $P_2И$. Сама запись состоит из двух операций: очистки ячейки (сброса в 0) и собственно записи. Для этого $БАВ$ сначала производит выборку и очистку ячейки, заданной адресом в $P_2А$. Очистка выполняется сигналами считывания слова в ячейке, но при этом блокируются усилители считывания и из $БУС$ в $P_2И$ информация не поступает. Затем в выбранную $БАВ$ ячейку записывается слово из $P_2И$.

Блок управления $БУП$ генерирует необходимые последовательности управляющих сигналов, инициирующих работу отдельных узлов памяти. Цепи передачи управляющих сигналов показаны тонкими линиями на рис. 4.2.

Ассоциативная память. В памяти этого типа поиск нужной информации производится не по адресу, а по ее содержанию (по ассоциативному признаку). При этом поиск по ассоциативному признаку (или последовательно по отдельным разрядам этого признака) происходит параллельно во времени для всех ячеек запоминающего массива. Во многих случаях ассоциативный поиск позволяет существенно упростить и ускорить обработку данных. Это достигается за счет того, что в памяти этого типа

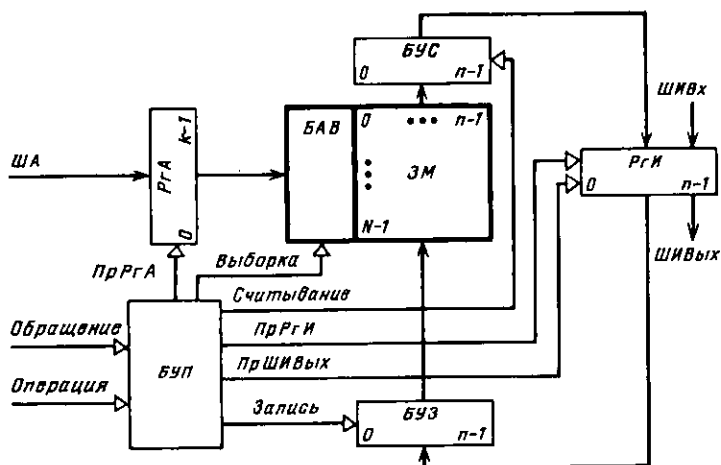


Рис. 4.2. Структура адресной памяти с произвольным обращением

операция считывания информации совмещена с выполнением ряда логических операций.

Типичная структура ассоциативной памяти представлена на рис. 4.3. Запоминающий массив содержит N $(n + 1)$ -разрядных ячеек. Для указания занятости ячейки используется служебный n -й разряд (0 — ячейка свободна, 1 — в ячейке записано слово).

По входной информационной шине ШИВх в регистр ассоциативного признака P_2AP в разряды $0 \div n - 1$ поступает n -разрядный ассоциативный запрос, а в регистр маски P_2M — код маски поиска, при этом n -й разряд P_2M устанавливается в 0. Ассоциативный поиск производится лишь для совокупности разрядов P_2AP , которым соответствуют 1 в P_2M (незамаскированные разряды P_2AP). Для слов, в которых цифры в разрядах совпали с незамаскированными разрядами P_2AP , комбинационная схема КС устанавливает 1 в соответствующие разряды регистра совпадения $P_2Cв$ и 0 в остальные разряды. Таким образом, значение j -го разряда в $P_2Cв$ определяется выражением

$$P_2Cв(j) = \bigwedge_{i=0}^{i=n-1} \{P_2AP[i] \oplus 3M[j, i] \vee P_2M[i]\} (0 \leq j \leq N - 1),$$

где $P_2AP[i]$, $P_2M[i]$ и $3M[j, i]$ — значения i -го разряда соответственно P_2AP , P_2M и j -й ячейки $3M$.

Комбинационная схема формирования результата ассоциативного обращения ФС формирует из слова, образовавшегося

в $P_2Cв$, сигналы $\alpha_0, \alpha_1, \alpha_2$, соответствующие случаям отсутствия слов в $ЗМ$, удовлетворяющих ассоциативному признаку, наличию одного и более чем одного такого слова. Для этого ΦC реализует следующие булевы функции:

$$\alpha_0 = \bigwedge_{j=0}^{N-1} \overline{P_2Cв[j]};$$

$$\begin{aligned} \alpha_1 = & P_2Cв[0] \overline{P_2Cв[1]} \dots \overline{P_2Cв[j]} \dots \overline{P_2Cв[N-1]} \vee \\ & \vee \overline{P_2Cв[0]} P_2Cв[1] \overline{P_2Cв[1]} \dots \overline{P_2Cв[j]} \dots \overline{P_2Cв[N-1]} \vee \dots \\ & \dots \vee \overline{P_2Cв[0]} \overline{P_2Cв[1]} \dots P_2Cв[j] \dots P_2Cв[N-1]; \\ \alpha_2 = & \bar{\alpha}_0 \bar{\alpha}_1. \end{aligned}$$

Формирование содержимого $P_2Cв$ и сигналов $\alpha_0, \alpha_1, \alpha_2$ по содержимому P_2AII , P_2M и $ЗМ$ называется операцией контроля ассоциации. Эта операция является составной частью операций считывания и записи, хотя она имеет и самостоятельное значение.

При считывании сначала производится контроль ассоциации по ассоциативному признаку в P_2AII . Затем при $\alpha_0 = 1$ считывание отменяется из-за отсутствия искомой информации, при $\alpha_1 = 1$ считывается в $P_2И$ найденное слово, при $\alpha_2 = 1$ в $P_2И$ считывается слово из ячейки, имеющей наименьший номер среди ячеек, отмеченных 1 в $P_2Cв$. Из $P_2И$ считанное слово выдается на $ШИВых$.

При записи сначала отыскивается свободная ячейка. Для этого выполняется операция контроля ассоциации при $P_2AII = 111\dots 10$ и $P_2M = 00\dots 01$. При этом свободные ячейки отмечаются 1 в $P_2Cв$. Для записи выбирается свободная ячейка с наименьшим номером. В нее записывается слово, поступившее с $ШИВх$ в $P_2И$.

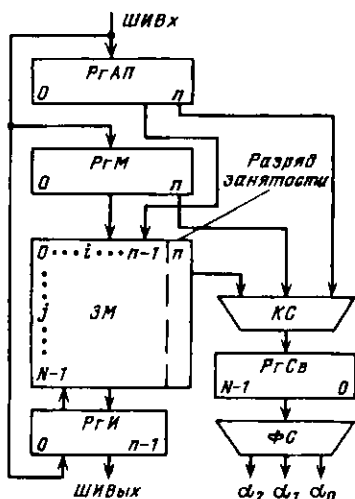


Рис. 4.3. Структура ассоциативной памяти

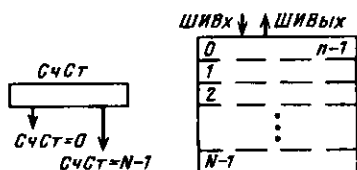


Рис. 4.4. Стековая память

С помощью операции контроля ассоциации можно, не считывая слов из памяти, определить по содержимому P_2C_6 , сколько в памяти слов, удовлетворяющих ассоциативному признаку, например реализовать запросы типа сколько студентов в группе имеют отличную оценку по данной дисциплине. При использовании соответствующих комбинационных схем в ассоциативной памяти могут выполняться достаточно сложные логические операции, такие, как поиск большего (меньшего) числа, поиск слов, заключенных в определенных границах, поиск максимального (минимального) числа и др.

Отметим, что для ассоциативной памяти необходимы запоминающие элементы, допускающие считывание без разрушения записанной в них информации. Это связано с тем, что при ассоциативном поиске считывание производится по всему ЗМ для всех незамаскированных разрядов и нигде сохранять временно разрушаемую считыванием информацию.

Стековая память, так же как и ассоциативная, является безадресной. В стековой памяти (рис. 4.4) ячейки образуют одномерный массив, в котором соседние ячейки связаны друг с другом разрядными цепями передачи слов. Запись нового слова производится в верхнюю ячейку (ячейку 0), при этом все ранее записанные слова (включая слово, находившееся в ячейке 0), сдвигаются вниз, в соседние ячейки с большими на 1 номерами. Считывание возможно только из верхней (нулевой) ячейки памяти, при этом, если производится считывание с удалением, все остальные слова в памяти сдвигаются вверх, в соседние ячейки с большими номерами. В этой памяти порядок считывания слов соответствует правилу: *последним поступил, — первым обслуживается*. В ряде устройств рассматриваемого типа предусматривается также операция простого считывания слова из нулевой ячейки (без его удаления и сдвига слова в памяти). Иногда стековая память снабжается счетчиком стека $CчСт$, показывающим количество занесенных в память слов. Сигнал $CчСт = 0$ соответствует пустому стеку, $CчСт = N - 1$ — заполненному стеку.

Часто стековую память организуют, используя адресную память (см. гл. 8). Широкое применение стековая память находит при обработке вложенных структур данных.

В последующих параграфах главы описываются различные типы ЗУ с адресной организацией. В гл. 14 рассмотрено использование ассоциативной памяти в аппаратуре динамического распределения ОП.

4.3. СТРУКТУРЫ АДРЕСНЫХ ЗУ

Техника оперативных памяти прошла большой путь развития. Большое значение имело появление в начале 50-х годов магнитных ЗУ, использующих в качестве запоминающего элемента (ЗЭ) ферритовые сердечники с прямоугольной петлей гистерезиса. Магнитные ЗУ сменяются в настоящее время полупроводниковыми, в которых в качестве ЗЭ служат триггерные схемы или МОП-транзисторы.

Тип используемых ЗЭ определенным образом влияет на структуру памяти, в результате чего существует большое разнообразие структур ЗУ. В настоящем параграфе дается систематизация структур адресных ЗУ,¹ что должно помочь в понимании принципов действия различных типов ЗУ с произвольным обращением и постоянных ЗУ.

Совокупность определенным образом соединенных ЗЭ образует запоминающую матрицу или запоминающий массив (ЗМ), где каждый ЗЭ хранит бит информации. Запоминающий элемент должен реализовать следующие режимы работы: хранение состояния, выдача сигнала состояния (считывание), запись 0 или запись 1. К ЗЭ должны поступать управляющие сигналы для задания режима работы, а также информационный сигнал при записи. А при считывании ЗЭ должен выдавать сигнал о его состоянии.

Запоминающий массив имеет систему адресных и разрядных линий (проводников). Адресные линии используются для выделения по адресу совокупности ЗЭ, которым устанавливается режим считывания или записи. Выделение отдельных разрядов осуществляется разрядными линиями, по которым передается записываемая в ЗЭ информация или информация о состоянии ЗЭ.

Запоминающие устройства строятся из специфичных ЗЭ, для которых характерно использование троичных сигналов и совмещение линий входных и выходных сигналов.

Адресные и разрядные линии носят общее название линий выборки. В зависимости от числа таких линий, соединенных с одним ЗЭ, различают двух-, трехкоординатные ЗУ и т. д., на-

¹ Систематизация структур ЗУ излагается по [41].

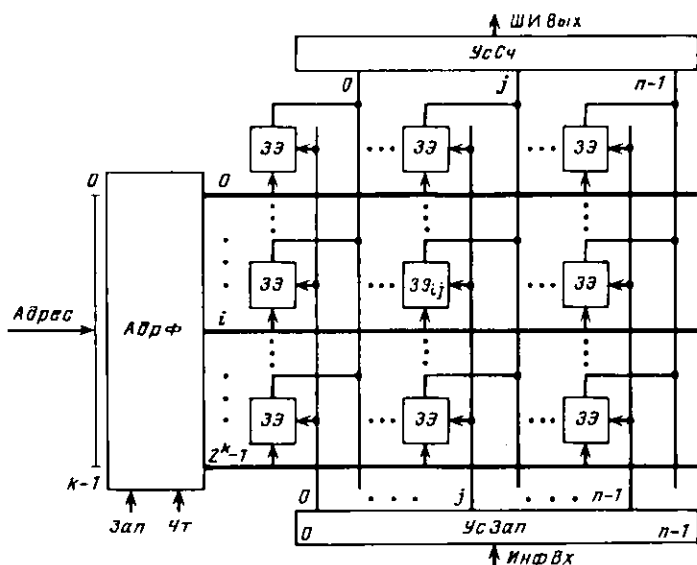


Рис. 4.5. Структура ЗУ типа 2D

зываемые ЗУ типа 2D, 3D и т. д. (от английского *dimension* – размерность). Наибольшее распространение получили ЗУ типов 2D, 3D, 2,5D и их модификации.

Запоминающие устройства типа 2D. Организация ЗУ типа 2D обеспечивает двухкоординатную выборку каждого ЗЭ ячейки памяти. Основу ЗУ составляет плоская матрица из ЗЭ, сгруппированных в 2^k ячеек по n разрядов. Обращение к ячейке задается k -разрядным адресом, выделение разрядов производится разрядными линиями записи и считывания. Структура ЗУ типа 2D приведена на рис. 4.5.

Адрес (k -разрядный) выбираемой ячейки i поступает на схему адресного формирования (АдрФ), управляемого сигналами чтения Чт и записи Зап . Основу АдрФ составляет дешифратор с 2^k выходами, который при поступлении на его входы адреса формирует сигнал для выборки линии i , при этом под воздействием сигналов Чт и Зап из АдрФ выдается сигнал либо настраивающий ЗЭ i -й линии на считывание (выдачу сигнала состояния), либо на запись. Выделение разряда j в i -м слове производится второй координатной линией. При записи по линии j от усилителя записи УсЗап поступает сигнал, устанавливающий выбранный для записи ЗЭ _{i j} в состояние 0 или 1. При считывании на усилитель считывания УсСч по линии j поступает сигнал о состоянии ЗЭ _{i j} .

Используемые здесь ЗЭ должны допускать объединение выходов для работы на общую линию с передачей сигналов только от выбранного ЗЭ. Такое свойство типично для современных ЗЭ и в дальнейшем всякий раз подразумевается.

Таким образом, каждая адресная линия выборки ячейки передает три значения сигнала: выборка при записи, выборка при считывании и отсутствие выборки. Каждая разрядная линия записи передает в ЗЭ записываемый бит информации, а разрядная линия считывания — считываемый из ЗЭ бит информации. Линии записи и считывания могут быть объединены в одну при использовании ЗЭ, допускающих соединение выхода со входом записи. Совмещение функций записи и считывания на разрядной линии широко используется в современных полупроводниковых ЗУ.

Запоминающие устройства типа 2D являются быстродействующими и достаточно удобными для реализации. Однако ЗУ типа 2D неэкономичны по объему оборудования из-за наличия в них дешифратора с 2^k выходами. В настоящее время структура типа 2D используется в основном в ЗУ небольшой емкости.

Запоминающие устройства типа 3Д. Некоторые ЗЭ имеют не один, а два конъюнктивных входа выборки. В этом случае адресная выборка осуществляется только при одновременном появлении двух сигналов. Использование таких ЗЭ позволяет строить ЗУ с трехкоординатным выделением ЗЭ.

Запоминающий массив ЗУ типа 3D представляет собой пространственную матрицу, составленную из n плоских матриц, представляющих собой 3М для отдельных разрядов ячеек памяти. Запоминающие элементы для разряда сгруппированы в квадратную матрицу из $\sqrt{2^k}$ рядов по $\sqrt{2^k}$ ЗЭ в каждом.

Структура матрицы j -го разряда в ЗУ типа 3D представлена на рис. 4.6. Для адресной выборки ЗЭ задаются две его координаты в 3М _{j} . Код адреса i -й ячейки памяти разделяется на старшую и младшую части i' и i'' , каждая из которых поступает на свой адресный формирователь. Адресный формирователь *АдрФ1* выдает сигнал выборки на линию i' , а *АдрФ2* — на линию i'' . В результате в 3М _{j} оказывается выбранным ЗЭ, находящийся на пересечении этих линий (двух координат), т. е. адресуемый кодом $i = i'/i''$. Адресные формирователи управляются сигналами $Чит$ и $Зап$ и в зависимости от них выдают сигналы выборки для считывания или для записи. При считывании сигнал о состоянии выбранного ЗЭ поступает по j -й линии считывания к УсСч (третья координата ЗЭ). При записи в выбранный ЗЭ будет занесен 0 или 1 в зависимости от сигнала записи в j -й разряд, поступающего по j -й линии от УсЗап (третья координата

в ЗМ определяется тремя координатами (две координаты для выборки и одна для выходного сигнала), а при записи в $ЗЭ_{ij}$ — двумя координатами. Считывание при этом осуществляется так же, как в ЗУ типа 3D, а запись сходна с записью в ЗУ типа 2D.

Запоминающий массив ЗУ типа 2,5D можно рассматривать как состоящий из отдельных ЗМ для каждого разряда памяти: $ЗМ_0, ЗМ_1, \dots, ЗМ_j, \dots, ЗМ_{n-1}$. Структура одноразрядного ЗУ дана на рис. 4.7, а. Код адреса i -й ячейки памяти, как и в ЗУ 3D,

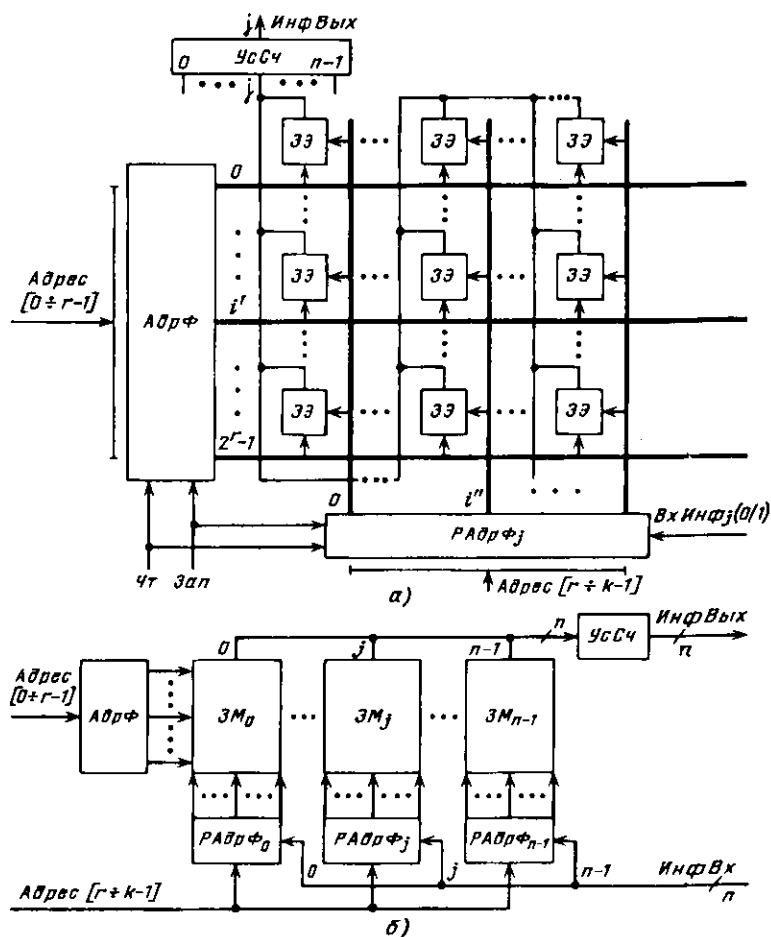


Рис. 4.7. Структура ЗУ типа 2,5 D:
а — одноразрядное ЗУ j ; б — n -разрядное ЗУ

разделяется на две части: i' и i'' , каждая из которых отдельно дешифрируется. Адресный формирователь $АдрФ$ выдает сигнал выборки на линию i' . Разрядно-адресный формирователь j -го разряда $РАдрФ_j$ выдает сигнал на линию i'' . При считывании оба сигнала, являющиеся сигналами выборки для считывания, опрашивают ЗЭ, выходной сигнал которого поступает на $УсЧ$ разряда j . Работает ЗУ в этом случае так же, как ЗУ типа 3D.

При записи $АдрФ$ выдает сигнал выборки для записи, а $РАдрФ_j$ выдает по линии i'' сигнал записи 0 или 1 в зависимости от назначения входного информационного сигнала j -го разряда $ВхИнФ_j$. На остальных линиях $РАдрФ_j$ не появляются сигналы записи и состояния всех ЗЭ, кроме ЗЭ, лежащего на пересечении линии i' и i'' , не меняются.

Из 3М отдельных разрядов формируется 3М всего ЗУ согласно схемы на рис. 4.7,б.

Наиболее экономичным по расходу оборудования ЗУ оказывается в том случае, если число выходных линий $АдрФ$ и всех $РАдрФ_j$ равно, т. е. если $r = (k - r) \log_2 n$ (рис. 4.7,б).

Недостатком ЗУ типа 2,5D является то, что сигналы на линиях $РАдрФ_j$ должны иметь четыре значения: чтение, запись 0, запись 1 и отсутствие записи (хранение). Для ЗЭ с разрушающим считыванием сигналы чтения и записи 0 совпадают и потребуются лишь три значения сигнала. В связи с этим ЗУ типа 2,5D используется лишь для ЗЭ с разрушающим считыванием. Таковыми являются, например, ферритовые сердечники.

Для построения современных полупроводниковых ЗУ из ЗЭ с неразрушающим считыванием используется структура ЗУ с двухкоординатным выделением ЗЭ и мультиплексированием выходных сигналов при считывании. Такие ЗУ будем называть ЗУ типа 2D-М.

Запоминающие устройства типа 2D-М. Запоминающие элементы таких ЗУ имеют два входа и один выход (рис. 4.8,а). При наличии хотя бы одного пустого сигнала $\sim$ на входах ЗЭ при записи находится в режиме хранения (как в ЗУ 3D). Сигнал чтения $Чт$ опрашивает состояние ЗЭ (так же как в ЗУ типа 2D). Сигналы записи $Зап$ и $Зап 0$ устанавливают ЗЭ в состояние 0, а $Зап$ и $Зап 1$ — в состояние 1 (так же, как в ЗУ типов 2D и 2,5D).

Обычно у запоминающих элементов ЗУ типа 2D-М выход объединяется со входом записи, как это показано для ЗЭ на рис. 4.8,б. Структура одноразрядного ЗУ типа 2D-М представлена на рис. 4.8,в. Как и в ЗУ типа 2,5D, код адреса i -й ячейки разделяется на две части: i' и i'' , одна из которых поступает на $АдрФ$, а другая — на разрядно-адресный коммутатор $РАдрК$. Если на $АдрФ$ и $РАдрК$ не приходит сигнал обращения к памя-

ти *Обр*, то на их выходных линиях не возникают действующие на ЗЭ сигналы и все ЗЭ находятся в режиме хранения. При наличии сигнала *Обр* выполняется считывание или запись в зависимости от значения сигнала *Чт/Зап*. При считывании *АдрФ* выдается по линии *i'* сигнал выборки для считывания, по которому со всех ЗЭ линии *i'* сигналы их состояний поступают на *РАдрК*. Коммутатор *РАдрК* мультиплексирует эти сигналы и передает на выход *ИнфВых* сигнал с линии *i''*. При записи *АдрФ* выдает по линии *i'* сигнал выборки для записи. Коммутатор *РАдрК* в зависимости от значения *ИнфВх* выдает сигнал записи 0 или 1 на линию *i''* и сигналы, не воздействующие на ЗЭ в остальные линии. В результате запись производится только в ЗЭ, лежащий на пересечении координатных линий *i'* и *i''*, причем $i'/i'' = i$.

Построив схему, аналогичную схеме на рис. 4.7, б, получим ЗУ для 2^k n -разрядных ячеек. Наиболее экономна такая схема при $r = (k - r) \log_2 n$.

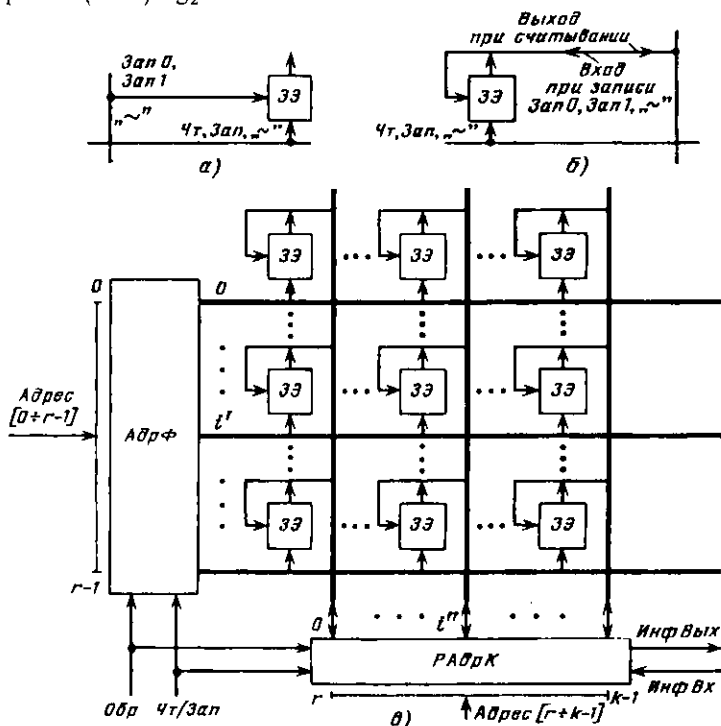


Рис. 4.8. Структура ЗУ типа 2D-M:

а — ЗЭ; б — ЗЭ с общим входом-выходом; в — структура одноразрядного ЗУ

Структура типа 2D-M наиболее удобна для построения полупроводниковых ЗУ и широко используется в настоящее время как в оперативных, так и в постоянных ЗУ.

4.4. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА С ПРОИЗВОЛЬНЫМ ОБРАЩЕНИЕМ

В вычислительной технике в качестве ЗУ с произвольным обращением, используемых в оперативных памятьях ЭВМ, широко применяются ЗУ с ЗЭ на ферритовых сердечниках и интегральные полупроводниковые ЗУ.

Принцип действия ферритового сердечника как запоминающего элемента поясняет рис. 4.9, на котором изображены сердечник с входной и выходной обмоточными обмотками и его кривая намагничивания (петля гистерезиса), имеющая форму, близкую к прямоугольной. Запоминание двоичных кодов основано на наличии у сердечников двух устойчивых состояний остаточной намагничивания противоположного знака. Состоянию, характеризующемуся остаточной индукцией $+B_r$, можно приписать значение 1, а состоянию с индукцией $-B_r$ — значение 0. Запись 1 в сердечник может быть произведена подачей в его входную обмотку импульса тока, создающего поле $+H_m$, превышающее коэрцитивную силу сердечника $+H_c$, при этом сердечник после снятия поля оказывается в состоянии $+B_r$. Для записи 0 во входную обмотку подается импульс тока, создающий поле $-H_m$, после снятия которого сердечник оказывается в состоянии $-B_r$. Для считывания во входную обмотку подается импульс тока той же полярности, как при записи 0, создающий поле, превышающее (по модулю) коэрцитивную силу H_c сердечника, при этом сердечник устанавливается в состояние 0, а если перед этим он находился в состоянии 1, то при считывании индукция в сердечнике изменяется от $+B_r$ до $-B_m$ и в выходной обмотке индуцируется ЭДС e_2 — сигнал считыва-

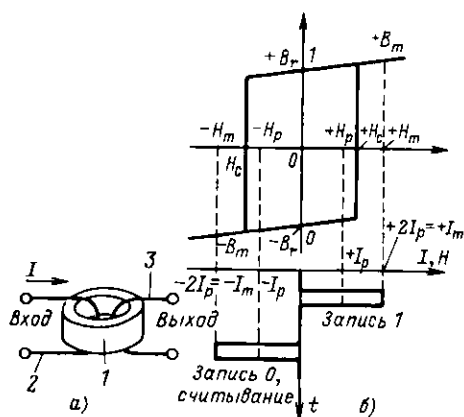


Рис. 4.9. Запоминающий элемент на ферритовом сердечнике (а) и петля гистерезиса сердечника (б):

1 — сердечник; 2 — обмотка записи; 3 — обмотка считывания

ния 1. Если сердечник находился в состоянии 0, то при считывании индукция в сердечнике меняется незначительно и индуцируемая при этом ЭДС e_0 (сигнал считывания 0) близка к 0. Считывание сопровождается стиранием информации, записанной в сердечнике. Поэтому при необходимости сохранения ранее хранившейся информации должна производиться *регенерация*, восстанавливающая информацию в сердечнике.

Ферритовые ЗУ, использующиеся в настоящее время, имеют структуру 2,5D (см. рис. 4.7). В запоминающем массиве через каждый сердечник проходят одна горизонтальная и одна вертикальная линии (соответственно от адресного и разрядно-адресного формирователей). Кроме того, все сердечники разрядной матрицы пронизываются общей обмоткой считывания, с которой снимаются выходные сигналы.

При считывании состояния сердечника в его координатные линии подаются импульсы тока $-I_p$. Ток $-I_p$ создает поле $-H_p$ такое, что $|-H_p| < |-H_c|$ и $2|-H_p| > |-H_m|$. На выбираемый сердечник воздействует поле двух токов $-I_p$, и он принимает состояние $-B_r$, генерируя в выходной обмотке сигнал 1, если сердечник ранее находился в состоянии $+B_r$, или сигнал 0, если сердечник находился в состоянии $-B_r$. Сердечники, воспринимающие ток $-I_p$ только по одной координатной линии, сохраняют свое состояние.

Чтобы считанная информация сохранилась в ЗУ, сразу после окончания считывания выполняется регенерация — запись считанной информации из информационного регистра РЗИ обратно в ЗМ.

Записи в ячейку ферритового ЗУ типа 2,5D всегда предшествует установка в состояние 0 всех сердечников ячейки, для чего выполняется операция считывания (без регенерации). При записи по адресной линии выбираемого сердечника поступает импульс тока $+I_p$. По разрядно-адресной линии поступает импульс тока $+I_p$ только при записи 1. Сердечник, в котором поля от двух импульсов $+I_p$ складываются, перемагничивается в состояние $+B_r$. Все остальные сердечники в ЗМ сохраняют состояния 0.

Успехи в технологии БИС привели к созданию *полупроводниковых интегральных ЗУ*.

По сравнению с ферритовыми ЗУ полупроводниковые имеют ряд важных достоинств: большее быстродействие, компактность, меньшую стоимость, совместимость по сигналам с логическими схемами, общие с другими электронными устройствами ЭВМ технологические и конструктивные принципы построения.

Недостатком полупроводниковых ЗУ с произвольным обращением является их энергозависимость, выражающаяся в том, что они потребляют энергию в режиме хранения информации и теряют информацию при выключении напряжения питания (потери информации можно избежать автоматическим переключением на аварийное питание от аккумуляторов).

По типу ЗЭ различают *биполярные ЗУ* с биполярными транзисторами (с ТТЛ- или ЭСЛ-схемами) и МОП-ЗУ с МОП-транзисторами.

Биполярные ЗУ. В биполярных интегральных ЗУ в качестве ЗЭ используется статический триггер на двух многоэмиттерных транзисторах с непосредственными связями (рис. 4.10).

Для записи в ЗЭ 1 или 0 с соответствующего усилителя записи на подключенный к нему информационный эмиттер подается потенциал логического 0 ($\leq 0,4$ В), а на другой информационный эмиттер продолжает поступать с его невозбужденного усилителя записи потенциал, равный примерно 1,5 В.

Если допустим, производится запись 1 в триггер, находившийся перед этим в состоянии 1 (открыт транзистор T_2), то подача потенциала низкого уровня на эмиттер 21 не меняет состояние триггера. Если до записи триггер находился в состоянии 0, то при подаче потенциала низкого уровня на эмиттер 21 (запись 1) открывается транзистор T_2 , при этом транзистор T_1 закрывается и триггер устанавливается в состояние 1.

Интегральная микросхема биполярного ЗУ представляет собой кристалл кремния, в котором образованы массивы ЗЭ (триггеров) со всеми межсоединениями, а также адресные дешифраторы, усилители-формирователи записи и считывания и другие схемы для управления адресной выборкой, записью и считыванием. Для повышения быстродействия ЗУ эти обслуживающие схемы могут быть выполнены на основе ЭСЛ-элементов, работающих в линейной области, в то время как построенные на основе ТТЛ-элементов триггеры ЗЭ работают с насыщением. В таком случае кристалл содержит схемы согласования уровней сигналов для перехода от схем ТТЛ к схемам ЭСЛ и обратно.

Полупроводниковые ЗУ размещаются в стандартных корпусах интегральных микросхем с числом выводов 14, 16, 18 и 22. Число выводов ограничивает число слов и разрядов запоминающего массива интегральной микросхемы. Для получения ЗУ с большим числом разрядов и (или) слов, чем в запоминающем массиве в корпусе схемы, применяются несколько корпусов.

В настоящее время биполярные ЗУ еще довольно дороги, поэтому они используются главным образом в качестве сверхоперативных памяти.

Динамические МОП-ЗУ сравнительно дешевы, потребляют небольшую мощность, позволяют достигнуть очень высокой плотности размещения ЗЭ на кристалле и, следовательно, большей емкости в одном корпусе микросхемы. В настоящее время динамические МОП-ЗУ широко используются для построения основной (оперативной) памяти ЭВМ [23].

В динамических ЗУ двоичные коды хранятся на «запоминающих емкостях», в качестве которых используются паразитные емкости некоторых цепей схем. Примем, что отсутствие заряда на запоминающей емкости означает состояние 0, а наличие — состояние 1. В таком случае считывание информации состоит в определении, заряжены или нет запоминающие емкости.

Запоминающая емкость может неопределенно долго сохранять состояние 0 (разряд отсутствует), но только ограниченное время из-за утечки заряда — состояние 1. Поэтому в рассматриваемых ЗУ необходимо периодически (примерно через каждые 2 мс) производить восстановление хранимой информации. Операция динамического восстановления информации называется *рефреш*. Рассматриваемые ЗУ получили название динамических.

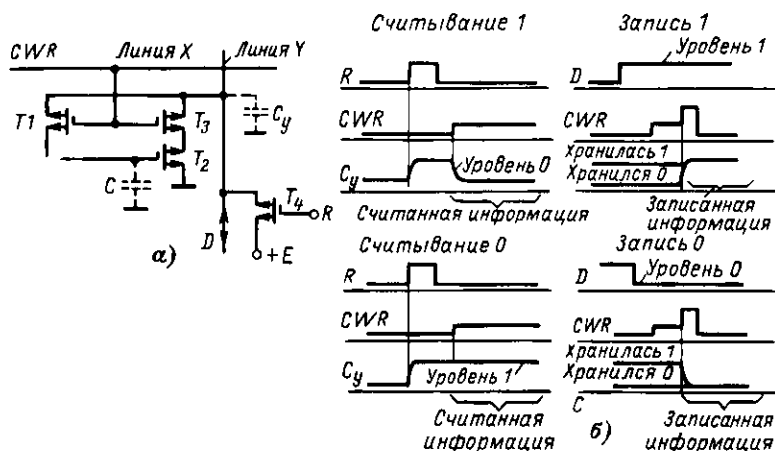


Рис. 4.11. Принципиальная электрическая схема (а) и временные диаграммы работы (б) ЗЭ динамического МОП-ЗУ

Схема и временные диаграммы работ ЗЭ динамического ЗУ на МОП-транзисторах в памяти со структурой 2D-M представлены на рис. 4.11. Запоминающей емкостью служит паразитная емкость C затвора транзистора T_2 . Линия разрядно-адресного коммутатора Y используется для ввода в ЗЭ бита информации при записи и съема его при считывании (см. рис. 4.8). Так как ЗЭ использует источник питания только при считывании, то им может служить паразитная емкость C_y линии Y .

Предварительно перед считыванием от разрядно-адресного коммутатора подается сигнал R , с помощью которого подготавливается считывание с мультиплексированием для ЗЭ, выбираемых линиями разрядно-адресного формирователя. Сигнал R открывает транзистор T_4 и емкость C_y подзарядается от источника $+E$. Затем на линию X подается от адресного формирователя сигнал считывания — промежуточный уровень сигнала CWR , который открывает транзистор T_3 , но не может открыть T_2 . Если ЗЭ хранит 1, то емкость C заряжена и открыт транзистор T_2 . В этом случае через открытые транзисторы T_3 и T_2 емкость C_y разряжается и низкий уровень (уровень 0) сигнала D на линии Y указывает, что ЗЭ хранил инверсное значение, т. е. 1. Если ЗЭ хранит 0, то емкость C разряжена, T_2 закрыт и сигнал CWR не может вызвать разряд емкости C_y . Высокий уровень сигнала D (уровень 1) указывает, что ЗЭ хранил 0. Далее сигнал D через разрядно-адресный коммутатор поступает на выход ЗУ.

При записи на линию Y поступает сигнал D , соответствующий записываемому двоичному знаку. Затем на линию X подается высокий уровень сигнала CWR , открывающий транзистор T_1 , который подключает к линии Y емкость C . В результате независимо от своего предыдущего состояния емкость оказывается заряженной, если записывается 1, и разряженной, если записывается 0.

Таблица 4.1. Параметры и области применения БИС ЗУ с произвольным обращением

Тип ЗУ	Емкость, Кбит	Время выборки, нс	Область применения			
			Большие ОП ЭВМ	Средние и небольшие ОП ЭВМ	ОП микро-ЭВМ	Сверхоперативная и буферная память
Статические ЭСЛ	4	10 – 20				+
» ТТЛ	4	25 – 80				+
» <i>n</i> -МОП	4 – 16	50 – 150		+	+	
» К-МОП	16	50 – 150		+	+	
Динамические <i>n</i> -МОП	264 – 1024	150 – 200	+	+	+	

В ЗУ периодически производится регенерация информации. При регенерации в ЗЭ записывается инверсное значение хранимого до считывания кода. После каждой четной регенерации в ЗЭ оказывается исходный код, а после каждой нечетной — его инверсия. В ЗУ имеется схема, сигнал которой указывает, какой код хранит в данный момент ЗЭ — прямой или инверсный.

В настоящее время большие оперативные памяти ЭВМ выполняют главным образом на динамических МОП-ЗУ, небольшие ОП — на МОП-ЗУ и ТТЛ-ЗУ, а сверхоперативные и буферные памяти — на ЭСЛ-ЗУ и ТТЛ-ЗУ.

В табл. 4.1 приведены характерные параметры БИС для разных типов полупроводниковых ЗУ и указаны области их использования.

4.5. ПОСТОЯННЫЕ ЗУ

Постоянные запоминающие устройства (ПЗУ) в рабочем режиме ЭВМ допускают только считывание хранимой информации. В зависимости от типа ПЗУ занесение в него информации производится или в процессе изготовления, или в эксплуатационных условиях путем настройки, предваряющей использование ПЗУ в вычислительном процессе. В последнем случае ПЗУ называются постоянными запоминающими устройствами с изменяемым в процессе эксплуатации содержимым или программируемыми постоянными запоминающими устройствами (ППЗУ).

Постоянные запоминающие устройства обычно строятся как адресные ЗУ. Функционирование ПЗУ можно рассматривать как выполнение однозначного преобразования *k*-разрядного кода адреса ячейки запоминающего массива ЗМ в *n*-разрядный код хранящегося в ней слова. При такой точке зрения ПЗУ можно считать преобразователем кодов или комбинационной схемой (автоматом без памяти) с *k*-входами и *n*-выходами.

По сравнению с ЗУ с произвольным обращением, допускающим как считывание, так и запись информации, конструкции ПЗУ значи-

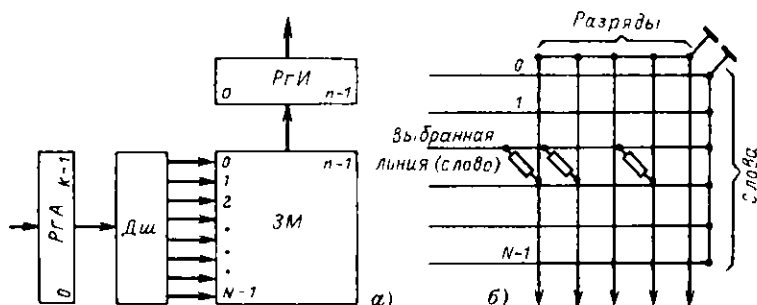


Рис. 4.12. Постоянное ЗУ типа 2D

тельно проще, их быстродействие и надежность выше, а стоимость ниже. Это объясняется большей простотой ЗЭ, отсутствием цепей для записи информации вообще или по крайней мере для оперативной записи, реализацией неразрушающего считывания, исключаяющего процедуру регенерации информации.

Одним из важнейших применений ПЗУ является хранение микропрограмм в микропрограммных управляющих устройствах ЭВМ. Для этой цели необходимы ПЗУ значительно большего, чем в ОП, быстродействия и умеренной емкости (10000 – 100000 бит).

Постоянные запоминающие устройства широко используются для хранения программ в специализированных ЭВМ, в том числе в микро-ЭВМ, предназначенных для решения определенного набора задач, для которых имеются отработанные алгоритмы и программы, например в бортовых ЭВМ самолетов, ракет и космических кораблей, в управляющих вычислительных комплексах, работающих в АСУ технологическими процессами. Такое применение ПЗУ позволяет существенно снизить требования к емкости ОП, повысить надежность и уменьшить стоимость вычислительной установки.

На рис. 4.12 приведена схема простейшего ПЗУ со структурой типа 2D. Запоминающий массив образуется системой взаимно перпендикулярных линий, в пересечениях которых устанавливаются ЗЭ, которые либо связывают (состояние 1), либо не связывают (состояние 0) между собой соответствующие горизонтальную и вертикальную линии. Поэтому часто ЗЭ в ПЗУ называют связывающими элементами. Для некоторых типов ЗЭ состояние 0 означает просто отсутствие запоминающего (связывающего) элемента в данной позиции в ЗМ.

Дешифратор Дш по коду адреса в РгА выбирает одну из горизонтальных линий (одну из ячеек ЗМ), в которую подается сигнал выборки. Выходной сигнал (сигнал 1) появляется на тех вертикальных разрядных линиях, которые имеют связь с возбужденной адресной линией (на рис. 4.12,б считывается слово 11010).

В зависимости от типа запоминающих (связывающих) элементов различают резисторные, емкостные, индуктивные (трансформаторные), полупроводниковые (интегральные) и другие ПЗУ.

В настоящее время наиболее распространенным типом являются полупроводниковые интегральные ПЗУ.

Полупроводниковые интегральные ПЗУ. Полупроводниковые ПЗУ имеют все те же достоинства, которые отмечались в предыдущем параграфе в отношении полупроводниковых ЗУ с произвольным обращением. Более того, в отличие от последних они являются энергонезависимыми. Постоянные ЗУ имеют большую емкость на одном кристалле (в одном корпусе интегральной микросхемы).

Положительным свойством интегральных ПЗУ является то, что некоторые типы этих устройств позволяют самому потребителю производить их программирование (занесение информации) в условиях эксплуатации и даже многократное перепрограммирование.

По типу ЗЭ, устанавливающих или разрывающих связь (контакт) между горизонтальными и вертикальными линиями, различают биполярные и МОП-схемы ПЗУ. Биполярные ПЗУ имеют время выборки 30–50 нс и емкость в одном кристалле (корпусе) от 256 бит до 16 Кбит. Постоянные ЗУ на МОП-схемах имеют большую емкость в одном кристалле (корпусе) — от 2 до 64 Кбит и еще большую, но и значительно меньшее быстродействие: время выборки 0,5–2 мкс.

По важнейшему признаку — способу занесения информации различают три типа интегральных полупроводниковых ПЗУ: 1) с программированием в процессе изготовления путем нанесения при помощи фотошаблонов в нужных потребителю точках контактных перемычек; 2) с программированием выжиганием перемычек или пробоем *p-n*-переходов, с помощью которых сам потребитель уже после изготовления прибора может уничтожить или образовать связи между горизонтальными и вертикальными линиями ЗМ (одноразовое программирование); 3) с электрическим перепрограммированием, при котором информация заносится в ЗМ электрическим путем, а стирание информации, необходимое для изменения содержимого ПЗУ, выполняется воздействием на ЗМ ультрафиолетового излучения или электрическим путем (многократное программирование). Время программирования составляет при ультрафиолетовом стирании 30–100 и при электрическом 0,1–1 с.

Программируемые фотошаблонами и выжиганием ПЗУ могут строиться на основе как биполярных, так и МОП-схем. Перепрограммируемые ПЗУ используют только МОП-схемы, способные хранить заряды.

Различные типы ЗЭ интегральных ПЗУ представлены на рис. 4.13. На рис. 4.13,а показан биполярный транзисторный ЗЭ с выжигаемой перемычкой, соединяющий горизонтальную и вертикальную линии. При «программировании» ПЗУ перемычки выжигаются в нужных местах серией импульсов тока с амплитудой 20–30 мА. При выборе адресным дешифратором горизонтальной линии *x* на базу транзистора ЗЭ поступает открывающий его сигнал, и при наличии перемычки (состояние 1) на вертикальной линии *y* появится потенциал коллектора транзистора +5 В.

На рис. 4.13,б изображен ЗЭ, программируемый пробиванием *p-n*-перехода. В исходном состоянии включенные встречно диоды изоли-

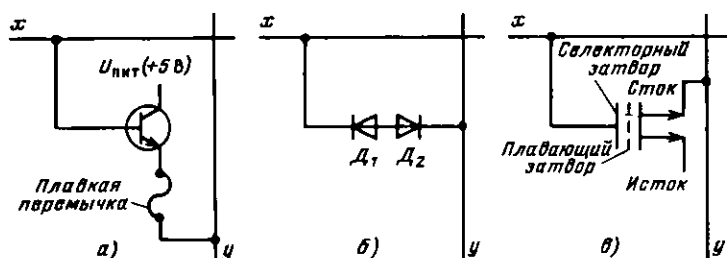


Рис. 4.13. Запоминающие (связывающие) элементы программируемых интегральных полупроводниковых постоянных ЗУ:

а — элемент с плавкой (выжигаемой) перемычкой; *б* — элемент с пробиваемым *p-n*-переходом; *в* — лавинно-инжекционный МНОП-транзистор с плавающим и селектирующим затворами

руют линии *x* и *y* (состояние 0). При подаче повышенного напряжения диод D_2 пробивается и закорачивается (состояние 1).

Более просто устроены ПЗУ с транзисторными и диодными запоминающими (связывающими) элементами, программируемые при изготовлении ПЗУ. В этом случае с помощью фотошаблонов в нужных позициях ЗМ наносятся или не наносятся контактные перемычки (вместо плавкой перемычки и вместо диода D_1 на рис. 4.13, *а* и *б* соответственно).

На рис. 4.13, *в* представлен ЗЭ в виде лавинно-инжекционного МОП-транзистора с плавающим и селектирующим затворами. Интегральные ПЗУ на таких элементах допускают многократную замену хранимой информации.

Плавающий (изолированный) затвор не имеет электрического подвода, он предназначен для хранения заряда. Селектирующий затвор подсоединен к одному из выходов дешифратора строк — горизонтальной линии, а сток — к вертикальной линии. В исходном состоянии отсутствует заряд на плавающем затворе (состояние 1), транзистор имеет очень небольшое пороговое напряжение. Выбор элемента осуществляется путем подачи на селектирующий затвор выходного напряжения адресного дешифратора, при этом включается транзистор и через цепь сток — исток протекает значительный ток. Программирование (занесение 0 в элементы) производится подачей на сток импульса напряжения 25–50 В, при этом происходит инжекция электронов, имеющих высокую энергию, через оксид на изолированный затвор, получающий отрицательный заряд (состояние 0). В результате увеличивается пороговое напряжение, и подача на селектирующий затвор выходного напряжения дешифратора не включает этот транзистор. Сообщенное элементу состояние сохраняется сколь угодно долго.

Структура программируемого ПЗУ емкостью 8 К (1 К × 8) бит на одном кристалле (в одном корпусе) изображена на рис. 4.14.

Рассматриваемое перепрограммируемое ПЗУ имеет структуру типа 2D-M. Запоминающий массив содержит 64 горизонтальных X и 128 вертикальных Y линий, на пересечении которых расположены МОП-транзисторы с плавающим и селектирующим затворами. Вертикальные линии разбиты на восемь групп по 16 в каждой. Число групп соответствует числу разрядов, хранимых в микросхеме (корпусе) слов. В качестве адресного формирователя используется дешифратор линий X , выдающий сигналы чтения. Разрядно-адресный коммутатор обра-

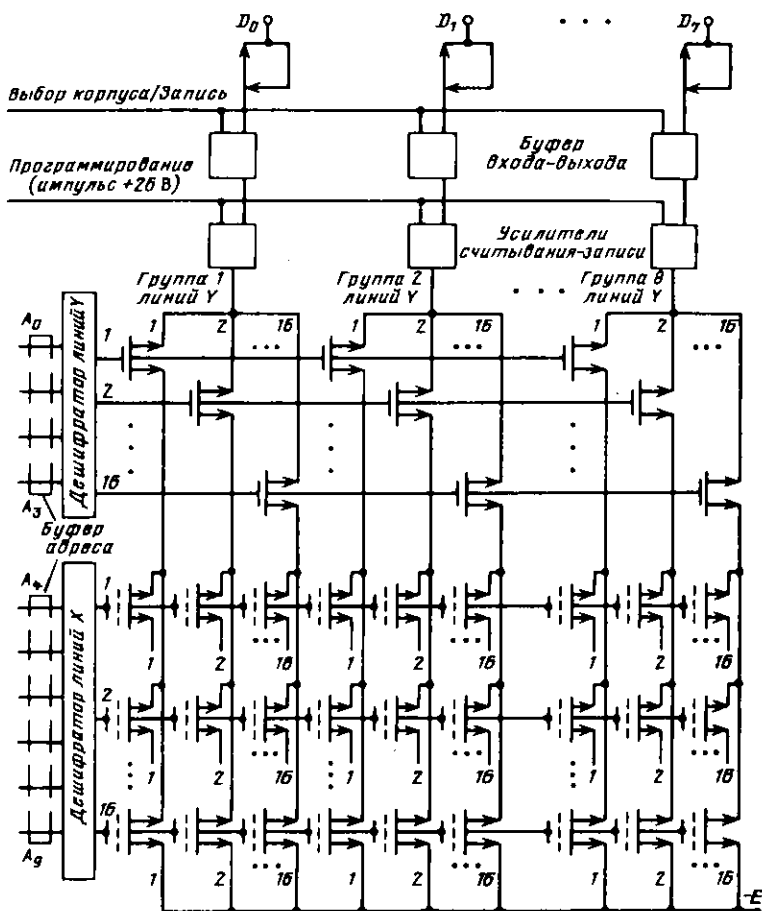


Рис. 4.14. Перепрограммируемое МОП-ПЗУ емкостью 8 К (1 К × 8) бит (в одном корпусе)

зован дешифратором линий Y , который управляет коммутирующими транзисторами, подсоединенными к разрядным усилителям считывания-записи.

При подаче младших разрядов адреса $A_3, \dots, A_0$ на дешифратор линий X выбирается одна из 64 горизонтальных линий, в результате чего управляющий потенциал поступает на присоединенные к этой линии селектирующие затворы 128 ЗЭ. В каждой группе линий по коду старших разрядов адреса $A_0 - A_3$ мультиплексируются сигналы от выбранных ЗЭ и выделяется сигнал адресуемого ЗЭ. Разрядные усилители формируют его и через буфера входа-выхода выдают на выходы $D_0, \dots, D_7$ микросхемы сигналы логических 0 и 1, соответствующие информации, находящейся в выбранной ячейке. При считывании информация в ячейке сохраняется.

Рассматриваемое ПЗУ может работать в двух режимах: 1) считывание хранимой информации и 2) программирование (запись) ПЗУ. Режим считывания устанавливается подачей сигнала выбора корпуса, который настраивает схемы буферов входа-выхода и разрядных усилителей на формирование и передачу на выходы $D_0, \dots, D_7$ сигналов, считанных с ЗЭ.

Перед записью производится стирание информации воздействием ультрафиолетового света на полупроводниковый кристалл ПЗУ. Под воздействием света заряды стекают с затворов. После стирания все ЗЭ находятся в состоянии 1.

Режим записи осуществляется при одновременной подаче сигнала *Запись* и импульса *Программирование*. На входы корпуса $A_0, \dots, A_9$ поступает адрес ячейки, в которую производится запись, а на выходы $D_0, \dots, D_7$ (в режиме записи ставшие входами) поступает записываемое 8-разрядное слово. Сигнал *Запись* настраивает буфера входа-выхода и разрядные усилители на передачу информационных сигналов с входов $D_0, \dots, D_7$ к стокам ЗЭ, при этом открываются усилители записи тех разрядов, в которых в записываемом слове находятся 0. Эти усилители пропускают в выбранные дешифратором линий Y вертикальные линии импульс *Программирование* (импульс + 26 В), который обеспечивает лавинный пробой и занесение отрицательного заряда в плавающие затворы (запись 0) только тех ЗЭ, селектирующие затворы которых присоединены к горизонтальной линии, возбужденной выходным сигналом дешифратора линий X .

Рассматриваемое устройство имеет время выборки 0,4–1 мкс, время программирования 30–100 с.

ЯЗЫК МИКРООПЕРАЦИЙ

5.1. ДЕКОМПОЗИЦИЯ ВЫЧИСЛИТЕЛЬНОГО
УСТРОЙСТВА НА ОПЕРАЦИОННЫЙ
И УПРАВЛЯЮЩИЙ БЛОКИ.

ПРИНЦИП АКАД. В. М. ГЛУШКОВА

Как показал акад. В. М. Глушков, в любом устройстве обработки цифровой информации можно выделить операционный и управляющий блоки [22]. Такой подход упрощает проектирование, а также облегчает понимание процесса функционирования вычислительного устройства. Декомпозиция цифрового вычислительного устройства поясняется на рис. 5.1.

Операционный блок состоит из регистров, сумматоров и других узлов, производящих прием из внешней среды и хранение кодов слов, их преобразование и выдачу во внешнюю среду результата преобразования, а также выдачу в управляющий блок и внешнюю среду *оповещающих сигналов*, принадлежащих к множеству

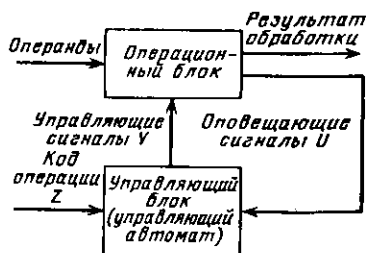
$$U = \{u_1, u_2, \dots, u_n\},$$

о знаках и особых значениях операндов, их отдельных разрядов, особых значениях промежуточных и конечных результатов операции (например, равенство нулю результата операции и др.).

Процесс функционирования во времени устройства обработки цифровой информации состоит из последовательности тактовых интервалов, в которых операционный блок производит определенные элементарные операции преобразования слов. Операционный блок выполняет некоторый набор элементарных преобразований информации, например таких, как передача слова из одного регистра в другой, взятие обратного кода, сдвиг и др. Выполнение этих элементарных операций инициируется поступлением в операционный блок соответствующих управляющих сигналов из некоторого множества сигналов

$$V = \{v_1, v_2, \dots, v_m\}.$$

Рис. 5.1. Операционный и управляющий блоки цифрового устройства



Элементарная функциональная операция (или их некоторая комбинация), выполняемая за один тактовый интервал и приводимая в действие одним управляющим сигналом v_i , называется *микрооперацией*.

В некоторые такты могут поступать несколько управляющих сигналов, вызывая параллельное во времени выполнение соответствующих микроопераций. Такая совокупность микроопераций называется *микрокомандой*. В частности, микрокоманда может состоять из одной *микрооперации*.

Управляющий блок (или *управляющий автомат*) вырабатывает распределенную во времени последовательность управляющих сигналов

$$v_{i_1}, v_{i_2}, \dots, v_{i_k} (v_{i_j} \in V),$$

порождающих в операционном блоке нужную последовательность микроопераций.

Последовательность управляющих сигналов определяется сигналами Z кода операции, поступающими в управляющий блок извне, и сигналами U , зависящими от операндов и промежуточных результатов преобразований.

Операционный блок задается его структурой, т. е. составом узлов и связями между ними, и выполняемым операционным блоком набором микроопераций.

Последовательность микрокоманд, обеспечивающая выполнение данной операции (например, операции нормализации числа с плавающей запятой), называется *микропрограммой* данной операции.

Функционирование вычислительного устройства может быть описано совокупностью реализуемых в нем микропрограмм. Это в ряде случаев удобный, хотя и не единственно возможный способ описания цифровых устройств.

5.2. ИЕРАРХИЯ ЯЗЫКОВ ОПИСАНИЯ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

Вычислительные машины и системы вместе с их программным обеспечением можно отнести к наиболее сложным системам, созданным человеком. Их сложность определяется многочисленностью входящих в их состав элементов, большим числом связей между элементами и сложностью осуществляемых законов функционирования и преобразования информации. Как и в других сложных системах, рассмотрение и описание структуры и функционирования ЭВМ и системы в зависимости от преследуемой цели производится с различной степенью детализации, определяемой уровнем, на котором анализируется процесс функционирования. Каждому уровню рассмотрения соответ-

ствуют определенные средства описания, которые по существу являются языком описания вычислительного устройства для данного уровня представления.

Сказанное можно проиллюстрировать на материале предыдущих глав. Например, работу элемента, изображенного на рис. 3.5, а, можно рассматривать с точки зрения процессов изменения электрических токов и напряжений в отдельных точках схемы. В этом случае формальным средством описания или языком описания являются дифференциальные уравнения для токов и напряжений в цепях схемы, в основе которых лежат законы физики полупроводников и законы электротехники. На таком уровне описания изучается поведение электронных схем в руководствах по электронике. Однако если такой подход применить к более сложному устройству, хотя бы, например, к комбинационному параллельному сумматору, то описание окажется весьма сложным и по нему трудно представить себе производимое схемой преобразование информации. Чтобы сделать это нагляднее, необходимо отвлечься от физических процессов, происходящих в схеме сумматора, и рассмотреть его работу на другом, более обобщенном уровне — на уровне комбинационных схем и использовать в качестве языка описания булевы функции. В этом случае в качестве входных и выходных переменных рассматриваются не электрические напряжения и токи в соответствующих точках схемы, а булевы переменные, принимающие всего два значения: 0 и 1.

Обратимся теперь к блокам, содержащим элементы памяти, например регистры, и выполняющим последовательности элементарных актов преобразования информации, называемых микрооперациями. Такие блоки в предыдущем параграфе были названы операционными. На уровне операционных блоков булевы функции уже недостаточны для построения описания.

Может показаться, что в этом случае, а тем более при рассмотрении более сложных образований — функциональных устройств, содержащих блоки памяти, несколько операционных и управляющих блоков или совокупностей взаимодействующих функциональных устройств, можно ограничиться словесным (вербальным) описанием процесса функционирования. Однако легко убедиться, что такой способ описания является и недостаточно лаконичным, и в то же время недостаточно строгим и точным, чтобы обеспечить однозначное представление о процессе функционирования устройства или машины.

Потребность в формализованных средствах описания структур и функционирования цифровых устройств машин определяется не только и не столько задачами обучения, сколько потребностями современной методологии проектирования ЭВМ. В последней существенное место занимают моделирование на ЭВМ проектных решений с целью их проверки и оптимизации и автоматизация с помощью ЭВМ конструирования вычислительных устройств и их отдельных узлов, синтеза контрольных и диагностических тестов и т. д. Для этого необходимы формализованные описания проектируемых устройств. Без формализованных описаний нельзя достигнуть соответствующего взаимопонимания между объединенными общим проектом разработчиками отдельных функциональных устройств.

Таблица 5.1. Иерархия языков формализованного описания вычислительных устройств

Уровень рассмотрения	Язык формализованного описания
I. Электронные схемы	Дифференциальные уравнения для токов и напряжений в цепях электрических схем
II. Комбинационные логические схемы	Аппарат теории булевых функций
III. Операционные узлы, узлы памяти, управляющие автоматы	Язык микроопераций
IV. Устройства вычислительной машины	Языки МОДИС [73], APL [36, 116, 118], OCC-2 и др.
V. Функционирование вычислительной машины	Языки машинных команд
VI. Вычислительный процесс	Алгоритмические языки (ФОРТРАН, ПЛ/1 и др.)
VII. Функционирование вычислительной системы в условиях действия случайных факторов	Язык моделирования дискретных стохастических систем (GPSS и др.)

Для формализованного описания устройств вычислительной техники используют различные «языки описания» в зависимости от необходимой степени детализации структуры и процесса функционирования. Различным уровням рассмотрения вычислительного устройства соответствует определенная иерархия языков описания (табл. 5.1).

Отметим, что нет четкой границы между уровнями рассмотрения, на которых можно использовать те или иные языки описания. Например, операционные блоки можно описать средствами языка описания функционирования вычислительных устройств (язык APL), но такое описание окажется во многих случаях недостаточно детализированным и наглядным. Работу устройства ЭВМ можно описать на языке микроопераций, но это описание окажется слишком детализированным и перегруженным подробностями, что будет затруднять понимание процесса функционирования. Подобным образом реализуемый ЭВМ вычислительный процесс можно описать и на алгоритмическом языке высокого уровня (например, на алгоритмическом языке ПЛ/1) и на языке команд ЭВМ. В последнем случае вычислительный процесс будет описан более подробно, все детали его выполнения однозначно определены. Такое описание пригодно для задания вычислительного процесса ЭВМ, но из-за слишком большой степени детализации и обилия подробностей воспринимается человеком несравненно хуже, чем описание вычислительного процесса на алгоритмическом языке высокого уровня.

Переход с одного уровня рассмотрения на другой, соответствующий в табл. 5.1 движению снизу вверх, сопровождается увеличением

степени детализации описания. Движение сверху вниз означает переход к описанию, позволяющему лучше понять процесс функционирования.

До сих пор речь шла о так называемых детерминированных объектах (устройствах, машинах, системах), при рассмотрении которых не учитывается действие случайных факторов. Учет случайных факторов (сбоев, отказов, потоков запросов на обработку программ и др.), необходимость в котором возрастает по мере перехода к рассмотрению более сложных вычислительных устройств и комплексов, требует привлечения методов описания и анализа дискретных стохастических систем. Для анализа подобных систем широко используется статистическое моделирование на ЭВМ процесса функционирования. Поэтому в качестве средства их описания могут служить языки моделирования дискретных стохастических систем (GPSS и др.).

5.3. ЯЗЫК МИКРООПЕРАЦИЙ

Язык микроопераций (ЯМ) предназначен для описания цифровых устройств, функционирование которых рассматривается на уровне регистров. Поэтому иногда такой язык называют регистровым или языком регистровых передач. Он имеет простые и наглядные средства описания слов и регистров, массивов и памяти, элементов и частей слов и массивов и соответственно элементов регистров и памяти, операций передачи слов и частей слов и др.

Описание слов, регистров и шин. Описание слова (числа, логического кода и др.) содержит его название — идентификатор и разрядный указатель. Идентификатором служит произвольная последовательность букв и цифр, начинающаяся с буквы. Разрядный указатель состоит из номеров старшего и младшего разрядов слова, разделенных знаком $\div$. Указатель заключается в квадратные скобки. Например, слово — число

$$X_{15} = \alpha_0 \alpha_1 \dots \alpha_n,$$

где α_i — двоичные разряды, можно представить в виде

$$X_{15} [0 \div n].$$

Разрядный указатель слова может опускаться, если это не вызывает недоразумений, в частности, если ранее слово уже полностью описано. Одноразрядное слово описывается только идентификатором без разрядного указателя.

Аналогичным образом описание регистра состоит из названия (идентификатора) регистра и разрядного указателя. Например, описание регистра команды на рис. 5.2 имеет вид: $PzK [0 \div 31]$, а его от-

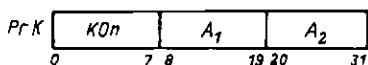


Рис. 5.2. К описанию регистра

дельных частей (подрегистров) и соответственно полей команды

$$PzK[0 \div 7] \text{ или } PzK[KOn],$$

$$PzK[20 \div 31] \text{ или } PzK[A_2],$$

т. е. в разрядном указателе подрегистра можно указывать идентификатор подрегистра (поля слова). Значение некоторого, например седьмого, разряда регистра PzK выделяется записью $PzK[7]$.

Совокупность линий (цепей), предназначенных для передачи слова или, в более общем случае, кодов и сигналов, объединенных общим функциональным назначением, называется шиной. Шина, по которой в цифровое устройство извне поступает или во внешнюю среду выдается слово, описывается, как и регистр, идентификатором шины и разрядным указателем. Например, описание шины, по которой поступает 32-разрядная команда, обозначается $ШК[0 \div 31]$.

Описание массива и памяти. Описание массива, т. е. совокупности слов одинаковой длины и памяти, содержит его (ее) название — идентификатор (например, $ОП2$ для модуля 2 оперативной памяти) и в квадратных скобках — наименьший и наибольший номера слов и нижнюю и верхнюю границу массива (наименьший и наибольший номера ячеек памяти), а также порядок нумерации разрядов в словах. Пример описания модуля памяти $ОП2$ (массива), содержащего r n -разрядных ячеек (слов): $ОП2[0 \div r - 1, 0 \div n - 1]$. В таком случае j -я n -разрядная ячейка памяти (j -е слово) и k -й разряд памяти (столбец массива) представляются записями соответственно $ОП2[j, 0 \div n - 1]$ и $ОП2[0 \div r - 1, k]$.

Описание микрооперации. Микрооперация осуществляет некоторое преобразование над данными. Это преобразование может быть логическим, т. е. выполняемым над операндом или двумя операндами поразрядно, арифметическим или функциональным. Например, это может быть установка некоторого слова в регистре, передача, взятие обратного кода, операции И и ИЛИ над двумя операндами, составление слова, сдвиг, операция счета, операции сложения и вычитания, сравнение на равенство и др. Микрооперации могут быть одностепенными или многостепенными, в них участвует соответствующее число операндов. Микрооперация описывается микрооператором и меткой — идентификатором управляющего сигнала, вызывающего выполнение микрооперации. Характерный вид описания двухместной микрооперации

$$\underbrace{G: PzA[k \div k + l]}_{\text{Метка}} = \underbrace{PzB[m \div m + l] * PzC[n \div n + l]}_{\text{Микрооператор}}; \quad (5.1)$$

Метка (идентификатор) управляющего сигнала отделяется двоеточием от микрооператора. Представление микрооператора основано на использовании «операции присваивания», обозначаемой знаком $=$. Выражение, стоящее справа от этого знака (правая часть микрооператора), называется «формулой микрооператора». Формула определяет

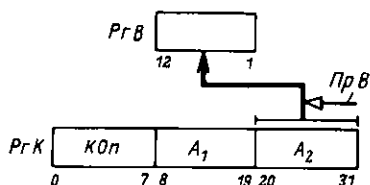


Рис. 5.3. Микрооперация: прием адреса A_2 из регистра P_2K в регистр P_2B

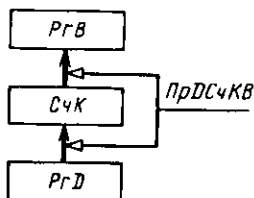


Рис. 5.4. Пример совмещенных во времени микрооператоров

преобразование [в (5.1) обозначено знаком *], производимое микрооперацией до завершения предусмотренной в ней передачи данных, и участвующих в нем операндов, или, точнее, их местоположение. Слева от знака присваивания (в левой части микрооператора) указывается регистр (или его часть), в который передается результат преобразования, описанного формулой микрооператора. Если в левой части микрооператора указана некоторая часть регистра (подрегистр), то после выполнения микрооперации остальные разряды регистра сохраняют прежнее значение.

Действие микрооператора состоит в том, что в конце такта выполнения микрооперации на регистре (шине и др.), описанном в левой части микрооператора, устанавливается слово, полученное в результате указанных в формуле микрооператора преобразований над значениями операндов в начале такта. В частном случае, если микрооперация состоит в передаче слова, формула микрооператора содержит лишь описание слова, точнее, регистра (или части регистра), из которого происходит передача. Например, микрооперацию приема адреса второго операнда A_2 из регистра команды P_2K в регистр адреса P_2B (рис. 5.3) можно представить в виде

$$ПрВ: P_2B[12 \div 1] := P_2K[20 \div 31];$$

или в более простой форме

$$ПрВ: P_2B := P_2K[A_2];$$

здесь $ПрВ$ – метка управляющего сигнала Прием в P_2B адреса A_2 из P_2K . Управляющий сигнал показан условно на рис. 5.3 тонкой стрелкой.

Управляющий сигнал микрооперации и его метка, рассматриваемая как переменная, принимают лишь два значения: 1 – микрооперация возбуждается, 0 – микрооперация не возбуждается. Один управляющий сигнал (метка) может инициировать выполнение в одном такте нескольких микрооператоров. Тогда последние записываются подряд и отделяются друг от друга запятыми. Точка с запятой разделяет микрооператоры, выполняющиеся в разных тактах.

Пусть, например, на рис. 5.4 прием из счетчика команд $CчК$ в регистр PzB и из регистра PzD в $CчК$ выполняются за один такт. Тогда можно записать

$$PrDCчKB : PzB := CчК, CчК := PzD;$$

Рассмотрим еще некоторые примеры микроопераций.

Приращение счетчика. В счетчике на рис. 5.5 определены микрооперации: установка в 0 ($УОСч$), прием кода с шины $ШИВх$ ($ПрСч$) и увеличение содержимого счетчика на 1 ($+1Сч$). Последнюю микрооперацию можно записать в виде

$$+1Сч : Cч := Cч + 1;$$

Сдвиг. Различают сдвиги арифметический, логический, циклический. В описаниях микроопераций им соответствуют обозначения $СдвА$, $СдвЛ$, $СдвЦ$. Далее за этим обозначением указываются направление сдвига (правый $П$ или левый $Л$) и в скобках число разрядов, на которое производится сдвиг. Например, описание микрооперации арифметического сдвига содержимого регистра C вправо на четыре разряда имеет вид

$$Сдв : PzC := СдвАП(4) PzC;$$

При арифметическом сдвиге знаковый разряд не сдвигается. Освобождающиеся при сдвиге разряды заполняются 0. Выдвигающиеся из регистра биты слова теряются.

При логическом сдвиге сдвигаются все разряды кода, включая знаковый разряд. При циклическом сдвиге крайние разряды регистра соединяются между собой так, что выдвигающиеся из крайнего разряда регистра биты поступают в другой его крайний разряд.

Сдвиг часто производится путем «косой передачи» слова из одного регистра в другой. Микрооперации передачи содержимого PzA в PzB со сдвигом на k разрядов вправо или влево обозначаются соответственно

$$СдвК : PzB := П(k) PzA;$$

$$СдвК : PzB := Л(k) PzA;$$

Конкатенация или составление слова. Пусть, например, в регистр B должно быть передано слово, отдельные разряды которого соответствуют содержимому некоторых разрядов регистра A , счетчика $Cч$, триггера переполнения $TzП$ и константе 0, как это показано на рис. 5.6. Описание соответствующей микрооперации имеет вид

$$PrPzB : PzB[1 \div 16] := PzA[8 \div 15] | Cч[0 \div 3] | TzП | 000;$$

Вертикальная линия является знаком операции составления слова.

Для описания требуемой последовательности выполнения микроопераций в ЯМ используется оператор перехода, который для удобства будем называть микрооператором, не связывая его выполнение с выполнением микрооперации.

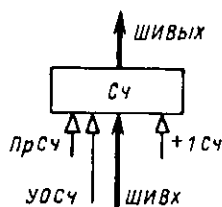


Рис. 5.5. Микрооперации счетчика

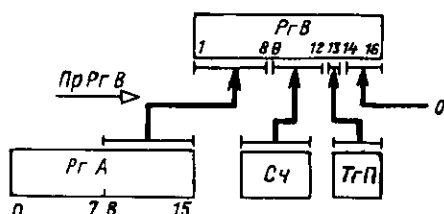


Рис. 5.6. Микрооперация составления (конкатенации) слова

Микрооператор перехода

идти к М;

позволяет задавать в микропрограмме переход к микрооператору с меткой М.

Условные микрооператоры. Часто в микропрограммах в зависимости от того, соблюдается или не соблюдается некоторое условие, должна выполняться та или иная микрооперация или, что фактически одно и то же, должна выполняться или пропускаться некоторая микрооперация. В этих случаях для записи микрооперации используются условные микрооператоры, одного из следующих типов: **если (условие) то микрооператор(ы) (или метка) иначе микрооператор(ы) (или метка);**

если (условие) то микрооператор(ы) (или метка);

Вместо микрооператора в условном микрооператоре может стоять метка, отсылающая к микрооператору, определенному в другом месте микропрограммы.

Для записи микрооперации можно воспользоваться также условным микрооператором вида (для двухместных микроопераций)

$$PzA[k \div k + I] := \text{если (условие) то } PzB[m \div m + I] * PzC[n \div n + I] \\ \text{иначе } PzD[p \div p + I] * PzE[q \div q + I];$$

Условие можно представить как равенство (или отсутствие равенства) содержимого некоторого регистра (или разряда регистра) определенной величине, например $C_m = 0$ или $PzA[0] \neq 0$, или в форме неравенства ($C_m > 0$), или в виде логической функции двоичных переменных, в качестве которых могут выступать значения определенных разрядов регистров. В последнем случае условие выполняется, если логическая функция принимает значение 1 (истинно).

Если описания микроопераций составляются не для проектирования устройства, а только для описания процесса функционирования, метки управляющих сигналов могут опускаться.

Запись микрооперации можно сопровождать пояснениями, которые должны при этом выделяться скобками <, >.

Описание микрокоманды. Описание микрокоманды производится аналогично описанию микрооперации и представляет собой метку микрокоманды и разделенную запятыми совокупность микрооператоров, выполняемых в микрокоманде, или меток соответствующих управляющих сигналов. В общем случае можно говорить о микрокоманде, так как микрооперация является частным случаем микрокоманды, содержащей только одну микрооперацию.

Обычно даже очень простые операции преобразования информации занимают несколько тактов, требуя выполнения определенной последовательности микрокоманд. Например, если на рис. 5.3 прием в P_2A адреса требует предварительной установки P_2A в 0, то прием в P_2A адреса второго операнда A_2 будет производиться последовательностью микрокоманд (т. е. микропрограммой)

1-й такт $УстО P_2A : P_2A := 0;$

2-й такт $ПрA_2 : P_2A := P_2K[A_2];$

Восприятие микропрограммы облегчается, если отдельные микрокоманды или группы микрокоманд, соответствующие определенным функциональным операциям, обозначить «метками операций». Метка операции предшествует описанию первой микрокоманды из ее группы и отделяется от этой микрокоманды знаком ::

Микропрограмма может быть изображена в виде графа, отдельные вершины которого соответствуют микрокомандам или группам последовательно выполняемых микрокоманд. Безусловные микрокоманды обозначаются прямоугольниками, а условные — ромбами с показом разветвлений. Внутри прямоугольников и ромбов записы-

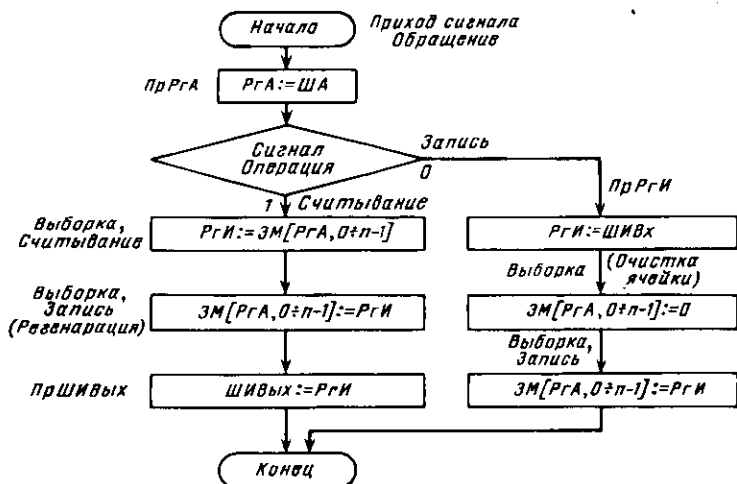


Рис. 5.7. Микропрограмма рабочего цикла памяти с произвольным обращением и с разрушающим считыванием

вают выражения для микрооператоров. Метки, если они используются, предоставляют сбоку прямоугольника.

В качестве примера на рис. 5.7 приведена представленная в виде графа микропрограмма рабочего цикла памяти с произвольным обращением и с разрушающим считыванием, словесное описание функционирования которой приводилось в § 4.2. Как видно из рис. 4.2, в блок управления памятью БУП поступают извне управляющие сигналы *Обращение* (сигнал начала рабочего цикла) и *Операция* (1 – считывание, 0 – запись); БУП генерирует в необходимой последовательности управляющие сигналы, инициирующие соответствующие микрооперации: *ПрРгА* – прием адреса с шин *ША* в *РгА*; *Выборка* – возбуждение шин адресной выборки ячейки памяти со сбрасыванием содержимого ячейки в 0; *Считывание* – открытие усилителей считывания и передача считанного кода в *РгИ*; *ПрРгИ* – прием в *РгИ* слова с ШИВх; *Запись* – возбуждение разрядных усилителей записи; *ПрШИВых* – выдача слова из *РгИ* на ШИВых.

Глава шестая

ПРИНЦИПЫ ОРГАНИЗАЦИИ АРИФМЕТИЧЕСКО-ЛОГИЧЕСКИХ УСТРОЙСТВ

6.1. ОБЩИЕ СВЕДЕНИЯ

Арифметическо-логические устройства (АЛУ) служат для выполнения арифметических и логических преобразований над словами, называемыми в этом случае операндами.

Выполняемые в АЛУ операции можно разделить на следующие группы:

- операции двоичной арифметики для чисел с фиксированной запятой;

- операции двоичной (или шестнадцатиричной) арифметики для чисел с плавающей запятой;

- операции десятичной арифметики;

- операции индексной арифметики (при модификации адресов команд);

- операции специальной арифметики;

- операции над логическими кодами (логические операции);

- операции над алфавитно-цифровыми полями.

Современные ЭВМ общего назначения обычно реализуют операции всех приведенных выше групп, а малые и микро-ЭВМ, микропроцессоры и специализированные ЭВМ часто не имеют аппаратуры арифметики чисел с плавающей запятой, десятичной арифметики и операций над алфавитно-цифровыми

полями. В этом случае эти операции выполняются специальными подпрограммами.

К арифметическим операциям относятся сложение, вычитание, вычитание модулей («короткие операции») и умножение и деление («длинные операции»). Группу логических операций составляют операции дизъюнкция (логическое ИЛИ) и конъюнкция (логическое И) над многоразрядными двоичными словами, сравнение кодов на равенство. Специальные арифметические операции включают в себя нормализацию, арифметический сдвиг (сдвигаются только цифровые разряды, знаковый разряд остается на месте), логический сдвиг (знаковый разряд сдвигается вместе с цифровыми разрядами). Обширна группа операций редактирования алфавитно-цифровой информации.

Можно привести следующую классификацию АЛУ.

По способу действия над операндами АЛУ делятся на последовательные и параллельные. В последовательных АЛУ операнды представляются в последовательном коде, а операции производятся последовательно во времени над их отдельными разрядами. В параллельных АЛУ операнды представляются параллельным кодом и операции совершаются параллельно во времени над всеми разрядами операндов.

По способу представления чисел различают АЛУ:

- 1) для чисел с фиксированной запятой;
- 2) для чисел с плавающей запятой;
- 3) для десятичных чисел.

По характеру использования элементов и узлов АЛУ делятся на блочные и многофункциональные. В блочном АЛУ операции над числами с фиксированной и плавающей запятой, десятичными числами и алфавитно-цифровыми полями выполняются в отдельных блоках, при этом повышается скорость работы, так как блоки могут параллельно выполнять соответствующие операции, но значительно возрастают затраты оборудования. В многофункциональных АЛУ операции для всех форм представления чисел выполняются одними и теми же схемами, которые коммутируются нужным образом в зависимости от требуемого режима работы.

По своему функциям АЛУ является операционным блоком (см. § 5.1), выполняющим микрооперации, обеспечивающие прием из других устройств (например, памяти) операндов, их преобразование и выдачу результатов преобразования в другие устройства. Арифметическо-логическое устройство управляется управляющим блоком, генерирующим управляющие сигналы, инициирующие выполнение в АЛУ определенных микроопераций. Генерируемая управляющим блоком последовательность

сигналов определяется кодом операции команды и оповещающими сигналами.

В заключение приведем характерные параметры для АЛУ современных ЭВМ: длина слова 32 разряда (4 байта), скорость работы 1 млн. сложений чисел с фиксированной запятой в секунду.

6.2. СТРУКТУРА И МИКРОПРОГРАММА АЛУ ДЛЯ СЛОЖЕНИЯ И ВЫЧИТАНИЯ ЧИСЕЛ С ФИКСИРОВАННОЙ ЗАПЯТОЙ

Обычно в АЛУ операция алгебраического сложения сводится к арифметическому сложению кодов чисел путем применения инверсных кодов — дополнительного или обратного для представления отрицательных чисел. Обратный код имеет два представления нуля ($+0$, -0), что затрудняет анализ результата операции. Чаще используется дополнительный код.

Алгоритмы выполнения в АЛУ арифметических операций зависят от того, в каком виде хранятся в памяти ЭВМ отрицательные числа — в прямом или дополнительном. В последнем случае сокращается время выполнения операции за счет исключения операции преобразования получаемого в АЛУ дополнительного кода отрицательного результата в прямой код, при этом несколько усложняется операция умножения.

Алгоритм алгебраического сложения двоичных чисел с фиксированной запятой при использовании дополнительного кода для представления отрицательных чисел. При выполнении операции сложения положительные слагаемые представляются в прямом коде, а отрицательные — в дополнительном. Производится сложение двоичных кодов, включая разряды знаков. Если при этом возникает перенос из знакового разряда суммы при отсутствии переноса в этот разряд или перенос в знаковый разряд при отсутствии переноса из разряда знака, то имеется переполнение разрядной сетки соответственно при отрицательной и положительной суммах. Если нет переносов из знакового разряда и в знаковый разряд суммы или есть оба эти переноса, то переполнения нет и при 0 в знаковом разряде сумма положительна и представлена в прямом коде, а при 1 в знаковом разряде сумма отрицательна и представлена в дополнительном коде.

На рис. 6.1 представлена упрощенная структурная схема АЛУ для операций сложения и вычитания n -разрядных (нулевой разряд знаковый) двоичных чисел с фиксированной запятой. Предполагается, что отрицательные числа хранятся в памяти в дополнительном коде.

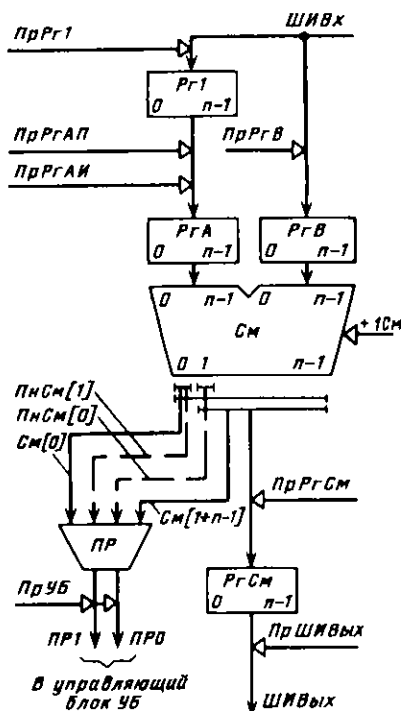


Рис. 6.1. Структурная схема АЛУ для сложения и вычитания двоичных чисел с фиксированной запятой

В состав АЛУ входят n -разрядный параллельный комбинационный сумматор $См$, регистр сумматора $PгСм$, входные регистры сумматора $PгВ$ и $PгА$, входной регистр АЛУ $Pг1$.

Из оперативной памяти по входной информационной шине **ШИВх** в АЛУ поступают операнды: положительные числа в прямом коде, а отрицательные в дополнительном. Операнды размещаются в $PгВ$ (первое слагаемое или уменьшаемое) и $Pг1$ (второе слагаемое или вычитаемое); $Pг1$ связан с $PгА$ цепями пря-

мой и инверсной передачи кода. Прямая передача используется при операции алгебраического сложения, а инверсная — при операции вычитания. Результат операции выдается из АЛУ в оперативную память по выходной информационной шине **ШИВых**.

При выполнении операции в АЛУ помимо результата операции формируется 2-разрядный код признака результата **ПР**, который принимает следующие значения:

Результат операции	Признак результата	
$= 0$	0	0
< 0	0	1
> 0	1	0
Переполнение	1	1

Примем, что код признака результата формируется комбинационной схемой **ПР**, на входы которой поступают сигналы, соответствующие значениям всех разрядов сумматора, а также сигналы переносов из знакового разряда **ПнСм[0]** и в знаковый из старшего цифрового разряда **ПнСм[1]**. Признак

переполнения ($PP = 11$) формируется, если булева функция

$$ПнСм[0] ПнСм[1] \vee \overline{ПнСм[0]} \overline{ПнСм[1]} = 1.$$

Признак нулевого значения результата $PP = 00$ формируется, если

$$\bigwedge_{l=0}^{n-1} Cм[l] = 1.$$

Условия выработки признаков положительного и отрицательного результатов имеют соответственно вид

$$\overline{Cм[0]} (ПнСм[0] ПнСм[1] \vee \overline{ПнСм[0]} \overline{ПнСм[1]});$$

$$Cм[0] (ПнСм[0] ПнСм[1] \vee \overline{ПнСм[0]} \overline{ПнСм[1]});$$

При выполнении алгебраического сложения поступившие в АЛУ коды операндов находятся на входных регистрах P_2B и P_2A сумматора. Код суммы формируется на выходах схемы $Cм$ и фиксируется в регистре $P_2Cм$.

Операция алгебраического вычитания

$$Z = X - Y = X + (-Y)$$

может быть сведена к изменению знака вычитаемого Y и операции алгебраического сложения. Изменению знака соответствует следующая процедура (см. гл. 2): принятый в P_2I код числа передается инверсно в P_2A и при сложении осуществляется подсуммирование 1 в младший разряд сумматора.

Передачи информации в регистрах АЛУ производятся отдельными микрооперациями, иницируемыми показанными на рис. 6.1 управляющими сигналами. Слово из P_2I в P_2A может быть передано в прямом (управляющий сигнал $ПрP_2AI$) или в инверсном (управляющий сигнал $ПрP_2AI$) кодах.

Ниже приведено совмещенное описание микропрограммы операций сложения и вычитания на языке ЯМ:

$ПрP_2B : P_2B := ШИВх$ <прием I операнда>;

$ПрP_2I : P_2I := ШИВх$ <прием II операнда>;

Прием: если сложение то $ПрP_2AP : P_2A := P_2I$ иначе $ПрP_2AI : P_2A := \overline{P_2I}$;

Сумма: если сложение то $ПрP_2Cм : P_2Cм := P_2A + P_2B$ иначе $ПрP_2Cм : P_2Cм := P_2A + P_2B + 1$;

$ПрУБ$: если $PP = 11$ то прерывание иначе

$ПрШИВых : ШИВых := P_2Cм$ <выдача результата>;

конец

Микрооперация $ПрУБ$ состоит в выдаче в управляющий блок кода признака и в формировании запроса прерывания при переполнении разрядной сетки.

6.3. СТРУКТУРЫ И МИКРОПРОГРАММА АЛУ ДЛЯ УМНОЖЕНИЯ ЧИСЕЛ С ФИКСИРОВАННОЙ ЗАПЯТОЙ

В ЭВМ операция умножения чисел с фиксированной запятой с помощью соответствующих алгоритмов сводится к операциям сложения и сдвига.

Произведение двух $(n - 1)$ -разрядных чисел может иметь $2(n - 1)$ значащих разрядов. Поэтому при операции умножения целых чисел необходимо предусмотреть возможность формирования в АЛУ произведения, имеющего двойную по сравнению с сомножителями длину. В ЭВМ, в которых числа с фиксированной запятой являются дробями, младшие $n - 1$ разрядов произведения часто отбрасываются (при отбрасывании может производиться операция округления произведения).

Для выполнения умножения АЛУ должно содержать регистры множимого, множителя и схемы формирования суммы частичных произведений — так называемый сумматор частичных произведений, в котором (рис. 6.2) путем соответствующей организации передач производится последовательное суммирование частичных произведений.

Операция умножения состоит из $n - 1$ [$(n - 1)$ — число цифровых разрядов множителя] циклов. В каждом цикле анализируется очередная цифра множителя, и если это 1, то к сумме частичных произведений прибавляется множимое, в противном случае прибавление не происходит. Цикл завершается сдвигом множимого относительно суммы частичных произведений либо сдвигом суммы частичных произведений относительно неподвижного множимого.

В зависимости от способа формирования суммы частичных произведений различают четыре основных метода выполнения умножения и соответственно четыре структуры АЛУ для этой операции (рис. 6.3). Для определенности сначала будем считать, что оба сомножителя — положительные числа.

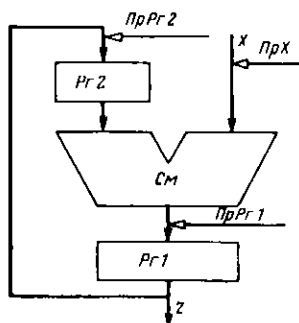


Рис. 6.2. Сумматор частичных произведений

1. Умножение, начиная с младших разрядов множителя, со сдвигом суммы частичных произведений вправо и при неподвижном множимом.

Регистр множителя и сумматор частичных произведений при этом должны иметь цепи сдвига вправо. Регистр множи-мого может не иметь цепей сдвига.

Последовательность действий в каждом цикле выполнения умножения определяется младшим разрядом регистра множи-теля, куда последовательно одна за другой поступают цифры множителя.

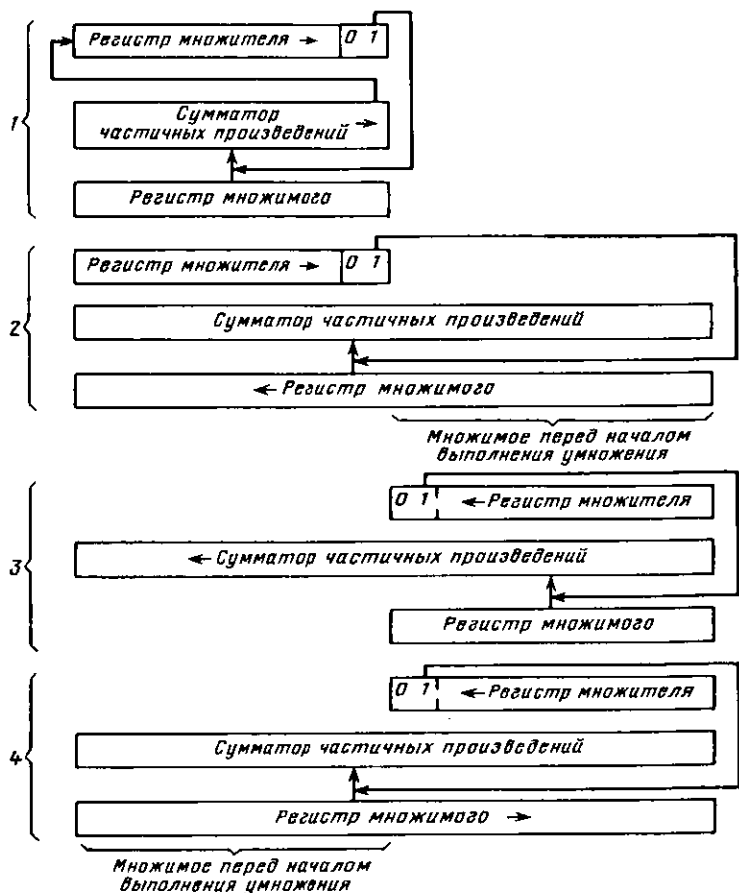


Рис. 6.3. Методы выполнения умножения

Поскольку по мере сдвига множителя вправо старшие разряды регистра множителя освобождаются, он может быть использован для хранения младших разрядов произведения, поступающих из младшего разряда сумматора частичных произведений по мере выполнения умножения. Для этого при выполнении сдвига младший разряд регистра сумматора частичных произведений соединяется со старшим разрядом регистра множителя. После выполнения умножения старшие разряды произведения находятся в регистре сумматора, младшие — в регистре множителя.

При данном методе умножения все три регистра имеют одинаковую длину, равную числу разрядов сомножителей. Этот метод умножения нашел наибольшее применение в ЭВМ.

2. Умножение, начиная с младших разрядов множителя, при сдвиге множимого влево и неподвижной сумме частичных произведений.

Регистр множителя при этом должен иметь цепи сдвига вправо, регистр множимого — цепи сдвига влево, а сумматор частичных произведений не содержит цепей сдвига.

Последовательность действий определяется, как и в первом варианте, младшим разрядом регистра множителя.

При этом методе регистр множимого и сумматор частичных произведений должны иметь двойную длину. Этот метод требует больше оборудования, но никаких преимуществ не дает, и поэтому применение его нецелесообразно.

3. Умножение, начиная со старших разрядов множителя, при сдвиге суммы частичных произведений влево и неподвижном множимом.

Регистр множителя и сумматор частичных произведений должны иметь цепи сдвига влево. Регистр множимого не имеет цепей сдвига.

Последовательность действий в каждом цикле выполнения умножения определяется старшим разрядом регистра множителя.

При этом методе сумматор частичных произведений должен иметь двойную длину. Данный метод требует дополнительного по сравнению с первым методом оборудования. Несмотря на это, он применяется в некоторых АЛУ, так как позволяет без дополнительных цепей сдвига выполнять и деление.

Для выполнения операций деления в АЛУ, реализующем первый метод умножения, необходимы дополнительные цепи сдвига влево в регистре множимого (частного) и в сумматоре частичных произведений (разностей).

4. Умножение, начиная со старших разрядов множителя, при сдвиге вправо множимого и неподвижной сумме частичных произведений.

Регистр множителя должен иметь цепи сдвига влево, регистр множимого — цепи сдвига вправо. Сумматор частичных произведений не имеет цепей сдвига. Последовательность действий на каждом шаге умножения определяется старшим разрядом регистра множителя.

При этом методе умножения и регистр множимого, и сумматор частичных произведений должны иметь двойную длину. Однако, как и третий метод, он не требует дополнительных цепей сдвига для выполнения деления.

При четвертом методе, в котором сумма частичных произведений неподвижна, можно совмещать во времени операции сдвига и сложения и за этот счет увеличить быстродействие АЛУ при выполнении умножения (а также деления).

Если необходимо образование произведения двойной длины, например, при операциях с целыми числами, наиболее экономичным является первый из рассмотренных методов умножения, так как он позволяет использовать все регистры одинарной длины.

Если в результате умножения достаточно иметь произведение одинарной длины, то целесообразно использовать либо первый, либо четвертый метод умножения. При использовании первого метода требуется введение дополнительных цепей сдвига для реализации деления, а при использовании четвертого метода необходимо удлинение сумматора. Выбор одного из этих методов умножения определяется соотношением затрат оборудования на реализацию цепей сдвига и дополнительных разрядов сумматора.

При образовании произведений одинарной длины простое отбрасывание младших разрядов вносит погрешность, которая будет накапливаться, так как произведение будет всегда вычисляться с недостатком. Поэтому для повышения точности вычислений часто производят округление результата умножения, вследствие чего погрешность становится знакопеременной.

Для округления произведения длина сумматора частичных произведений обычно увеличивается на один разряд. После образования произведения к этому дополнительному разряду прибавляется 1. Если дополнительный разряд произведения был равен 0, то произведение в основных разрядах сумматора получается с недостатком. Если дополнительный разряд был равен 1, то в результате переноса 1 из дополнительного разряда к основным разрядам сумматора добавляется единица и произведение получается с избытком, при этом максималь-

ное значение погрешности произведения равно половине 1 младшего разряда.

Отметим, что при любом методе умножения операция обычно начинается с анализа на 0 сомножителей. При равенстве нулю хотя бы одного сомножителя умножение не производится, а произведению присваивается нулевое значение.

Рассмотрим более подробно наиболее распространенный метод умножения целых чисел, начиная с младших разрядов, со сдвигом суммы частичных произведений вправо.

Алгоритм умножения чисел, представленных в прямом коде, начиная с младших разрядов, со сдвигом суммы частичных произведений вправо:

1. Берутся модули от сомножителей.
2. Исходное значение суммы частичных произведений принимается равным 0.
3. Если анализируемая цифра множителя равна 1, то к сумме частичных произведений прибавляется множимое; если эта цифра равна 0, прибавление не производится.
4. Производится сдвиг суммы частичных произведений вправо на один разряд.
5. Пункты 3 и 4 последовательно выполняются для всех цифровых разрядов множителя, начиная с младшего.
6. Произведению присваивается знак плюс, если знаки сомножителей одинаковы, в противном случае — знак минус.

Детальное рассмотрение метода умножения начнем со случая умножения целых положительных чисел, а затем перейдем к числам со знаками.

Пусть X — множимое и Y — множитель. Тогда, так как числа имеют $(n-1)$ цифровых разрядов с весами, изменяющимися от 2^0 для разряда с номером $(n-1)$ до 2^{n-2} для разряда с номером 1, произведение $Z = XY$ можно представить в виде

$$\begin{aligned} Z &= XY = X(y_1 2^{n-2} + y_2 2^{n-3} + \dots + y_{n-2} 2^1 + y_{n-1} 2^0) = \\ &= 2^{n-1} \{Xy_1 2^{-1} + y_2 2^{-2} + \dots + y_{n-2} 2^{-(n-2)} + y_{n-1} 2^{-(n-1)}\} = \\ &= 2^{n-1} \{(\dots ((0 + Xy_{n-1}) 2^{-1} + Xy_{n-2}) 2^{-1} + \dots + \\ &+ Xy_1) 2^{-1}\} = 2^{n-1} A, \end{aligned} \quad (6.1)$$

где $y_i = (0, 1)$ — значение i -го разряда множителя Y .

Для получения произведения Z производятся вычисления по приведенной формуле. При этом Xy_i равно либо 0 (при $y_i = 0$), либо X (при $y_i = 1$), а умножение на 2^{-1} осуществляется путем сдвига числа на один разряд вправо.

Вычислив A и перенес условную запятую на $(n-1)$ разряд вправо (умножив A на 2^{n-1}), получим Z .

Особенностью умножения целых чисел является необходимость представления результата умножения двух n -разрядных слов двойным словом. При этом число цифровых разрядов двойного слова $2n - 1$ на 1 больше числа $2n - 2$ цифровых разрядов произведения двух чисел, содержащих $n - 1$ цифровых разрядов. В связи с этим после получения результата в формате двойного слова необходимо дополнительно сдвинуть его цифровые разряды на один разряд вправо, чтобы правильно расположить произведение в разрядной сетке.

На рис. 6.4 представлена структурная схема АЛУ для умножения n -разрядных целых двоичных чисел. В состав АЛУ входят: входной регистр множимого $Pr1$; регистры множителя $Pr2$ и $Pr2'$, на которых с помощью косой передачи вправо $Pr2' := \Pi(1)Pr2$ и передачи $Pr2 := Pr2'$ выполняется сдвиг множителя вправо; сумматор $См$ для образования суммы частичных произведений; входные и выходные регистры сумматора соответственно PrA , PrB и $PrCm$, на которых соответ-

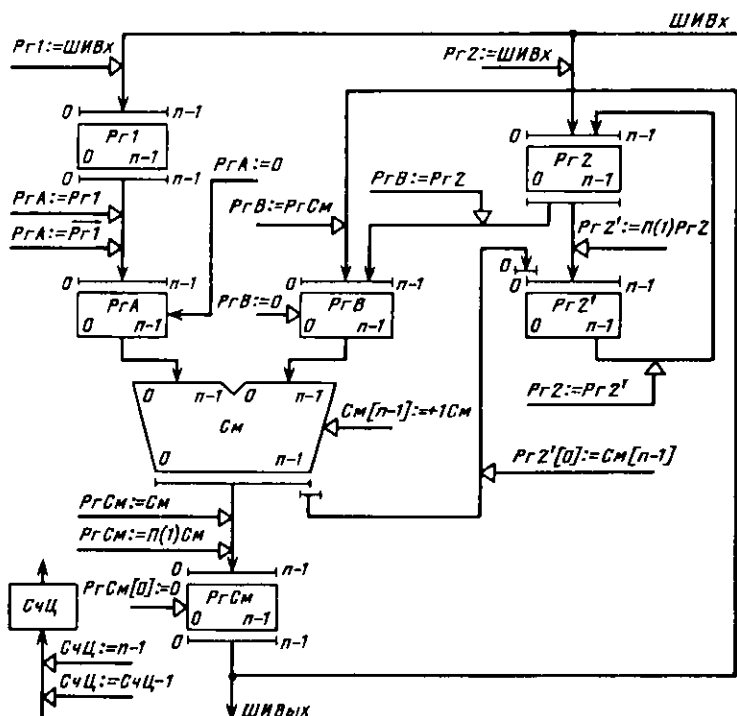


Рис. 6.4. Структурная схема АЛУ для умножения целых чисел

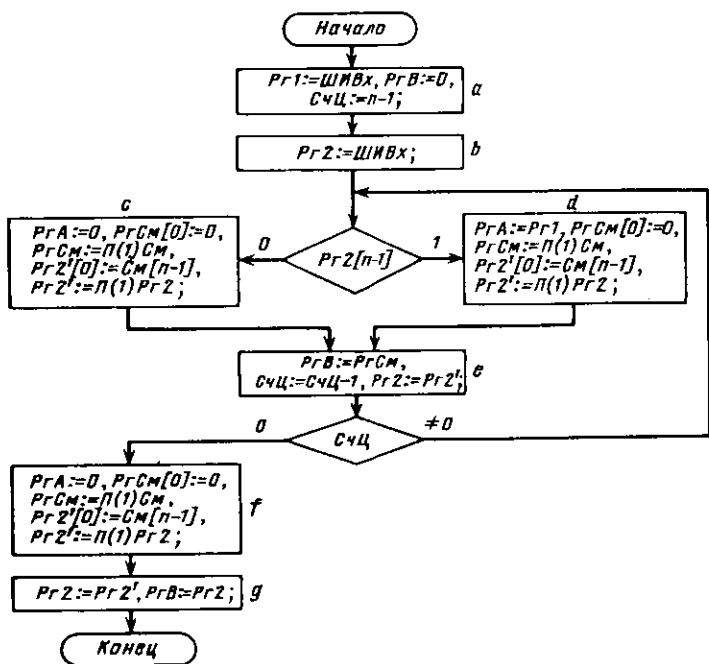


Рис. 6.5. Микропрограмма умножения целых положительных чисел

ственно хранится текущее значение и образуется новое значение суммы; счетчик циклов $C4Ц$.

Работа АЛУ при умножении целых положительных чисел описывается микропрограммой, приведенной на рис. 6.5. Первоначально на $Pz1$ поступает множимое, регистр PzB , хранящий сумму частичных произведений, обнуляется, а в счетчик циклов заносится число обрабатываемых цифровых разрядов (блок a микропрограммы). Затем на $Pz2$ поступает множитель (блок b). На этом завершается процедура начальных установок, и начинается процесс вычисления сумм частичных произведений.

В зависимости от значения 0 или 1 младшего разряда множителя к частичному произведению прибавляется либо 0, либо X , для чего соответствующее значение присваивается регистру PzA ; полученная сумма умножается на 2^{-1} путем передачи кода с выхода сумматора на $PzCм$ со сдвигом на один разряд вправо.

Одновременно множитель подготавливается к перемещению в $Pz2$ так, чтобы на месте анализируемого младшего разряда $Pz2$ оказался следующий разряд множителя. Для этого

содержимое $Pz2$ переносится в $Pz2'$ со смещением вправо на один разряд. Разряд 0 $Pz2'$ при этом остается свободным, и в него заносится младший разряд суммы, выходящий при сдвиге за пределы $PzCm$ ($Pz2'[0] := Cm[n-1]$). В следующем такте (блок e) завершается сдвиг множителя путем занесения содержимого $Pz2'$ в $Pz2$, и в PzB образуется сдвинутая сумма частичных произведений. Кроме того, в этом такте уменьшается на 1 содержимое счетчика циклов.

Когда счетчик циклов установится в 0, в $PzCm$ и $Pz2$ будут храниться соответственно старшие и младшие разряды произведения, требующие сдвига на один разряд вправо для правильного расположения в формате двойного слова. После выполнения этого сдвига (блоки f и g) результат операции из $PzCm$ и PzB поступает на $ШИВых$. Микрооперации выдачи произведения на $ШИВых$ на рисунке не показаны.

Умножение чисел со знаком можно свести к взятию модулей от сомножителей, их перемножению (см. микропрограмму на рис. 6.5) и формированию знакового разряда произведения.

При использовании для представления отрицательных чисел дополнительного кода умножение выполняется обычно непосредственно над дополнительными кодами сомножителей.

Покажем, каким образом это может быть осуществлено. Сначала рассмотрим случай $X \leq 0$ и $Y > 0$.

Формула (6.1) не меняется, но для получения отрицательного Z в дополнительном коде следует при $X < 0$ подсуммирование X производить в дополнительном коде. Это естественным образом происходит при представлении отрицательных чисел в дополнительном коде и не требует специальных схемных решений. Кроме того, следует учесть, что при умножении на 2^{-1} происходят обычный сдвиг положительных чисел и модифицированный сдвиг (с занесением 1 в старший разряд сдвигаемого числа) отрицательных чисел. В итоге $Z \geq 0$ будет представлено прямым кодом, а $Z < 0$ — дополнительным.

Рассмотрим случай $X \leq 0$ и $Y < 0$. Если $Y < 0$, то $Y_{\text{доп}} = 2^n - |Y|$. Тогда в цифровых разрядах кода Y будет представлено число $2^{n-1} - |Y|$, так как вес знакового разряда составляет 2^{n-1} . Если умножить X на цифровые разряды $Y_{\text{доп}}$ так, как умножали при $Y > 0$, то получим

$$Z' = X(2^{n-1} - |Y|) = -|Y|X + X \cdot 2^{n-1}.$$

Псевдопроизведение Z' больше истинного на $X \cdot 2^{n-1}$. Поэтому можно получать произведение, вычисляя псевдопроизведение и вычитая из него $X \cdot 2^{n-1}$.

Поправка вносится при $C4C = 0$ перед последним сдвигом, ставящим результат на свое место в двойном слове. Заметим, что после внесения этой поправки не возникает переполнения, а следовательно, $P2Cm[0]$ указывает знак результата, определяющий, какой производится сдвиг (обычный или модифицированный).

Алгоритм умножения, начиная с младших разрядов множителя, со сдвигом суммы частичных произведений и использованием дополнительного кода для отрицательных чисел:

1. Положительные сомножители представляются прямым кодом, а отрицательные — дополнительным.

2. Исходное значение суммы частичных произведений принимается равным 0.

3. Если анализируемая цифра множителя равна 1, то к сумме частичных произведений прибавляется¹ множимое в том коде, в котором оно представлено; если эта цифра равна 0, прибавление не производится.

4. Сумма частичных произведений сдвигается на один разряд вправо, при этом, если сумма отрицательна, осуществляется модифицированный сдвиг.

5. Пункты 3 и 4 последовательно выполняются для всех цифровых разрядов множителя, начиная с младшего.

6. Если множитель — положительное число, полученный результат представляет собой произведение. Если множитель отрицателен, то для получения произведения к результату прибавляется¹ множимое с обратным знаком.

Если результат размещается в двойном слове, то он предварительно сдвигается на один разряд вправо.

7. Произведение получается в прямом коде, если сомножители имеют одинаковые знаки, и в дополнительном, если сомножители имеют разные знаки.

Для выполнения такого умножения можно использовать АЛУ, представленное на рис. 6.4. Микропрограмма умножения приведена в [32].

6.4. МЕТОДЫ УСКОРЕНИЯ УМНОЖЕНИЯ

Методы ускорения умножения делятся на аппаратные и логические. Как те, так и другие требуют дополнительных затрат оборудования. При использовании аппаратных методов дополнительные затраты оборудования прямо пропорцио-

¹ Прибавление производится с выравниванием складываемых чисел по старшим разрядам.

нальны числу разрядов в операндах. Эти методы вызывают усложнение схемы АЛУ.

Дополнительные затраты оборудования при реализации логических методов ускорения умножения не зависят от разрядности операндов. Усложняется в основном схема управления АЛУ. В ЭВМ для ускорения умножения часто используются комбинации этих методов.

К аппаратурным методам ускорения умножения относятся ускорение выполнения операций сложения и сдвига, введение дополнительных цепей сдвига, позволяющих за один такт производить сдвиг информации в регистрах сразу на несколько разрядов, совмещение по времени операций сложения и сдвига, построение комбинационных схем множительных устройств, реализующих «табличное» умножение.

Среди логических методов наиболее распространены в настоящее время методы, позволяющие за один шаг умножения обработать несколько разрядов множителя.

Рассмотрим один из способов умножения на два разряда множителя, начиная с его младших разрядов. В зависимости от результата анализа пары разрядов множителя предусматриваются следующие действия. При 00 производится простой сдвиг на два разряда вправо суммы частичных произведений. При 01 к сумме частичных произведений прибавляется одинарное множимое и сумма частичных произведений сдвигается на два разряда вправо. При 10 прибавляется удвоенное множимое и сумма частичных произведений сдвигается на два разряда вправо. При 11 из суммы частичных произведений вычитается одинарное множимое и сумма частичных произведений сдвигается на два разряда вправо. Тогда в первых трех случаях результат получается правильный, а в последнем неправильный, он должен быть скорректирован на следующем шаге.

Поскольку при 11 из суммы частичных произведений вычитается одинарное множимое вместо прибавления утроенного множимого, для корректировки результата к сумме частичных произведений перед выполнением сдвига надо было бы прибавить учетверенное множимое. Но после сдвига на два разряда вправо сумма частичных произведений уменьшается в 4 раза, так что для корректировки его на следующем шаге должно быть прибавлено одинарное множимое.

Это учитывается при обработке следующей пары разрядов. Если следующая пара 00, то она обрабатывается как 01, если 01, то как 10, если 10, то как 11, если 11, то как 00, и фиксируется необходимостью коррекции при обработке следующей пары. Удвоенное множимое может быть получено его сдвигом. При-

Таблица 6.1.

Пара разрядов множителя	Признак коррекции из предыдущей пары	Признак коррекции для следующей пары	Знак действия	Кратность множимому
00	0	0		0
01	0	0	+	1
10	0	0	+	2
11	0	1	—	1
00	1	0	+	1
01	1	0	+	2
10	1	1	—	1
11	1	1		0

знак необходимости коррекции может запоминаться в отдельном триггере коррекции.

Правила обработки пар разрядов множителя с учетом признака коррекции сведены в табл. 6.1.

После обработки каждой комбинации содержимое регистра множителя и сумматора частичных произведений сдвигается на два разряда вправо.

Данный метод умножения требует корректировки результата, если старшая пара разрядов множителя 11 или 10 и состояние триггера коррекции является единичным. В этом случае к полученному произведению должно быть добавлено множимое. Аналогичным образом можно организовать умножение с обработкой за шаг большего числа разрядов множителя.

Так, например, в ЭВМ ЕС-1052 умножение выполняется с обработкой за шаг четырех разрядов множителя. На шаге умножения анализируется очередная тетрада множителя и для каждой пары разрядов тетрады определяется частичное произведение. Полученные два частичных произведения одновременно суммируются со сдвинутой на четыре разряда вправо текущей суммой частичных произведений.

Более детально с методами логического ускорения умножения можно ознакомиться в [32].

Все более широкое распространение, особенно в микропроцессорных системах, получают в настоящее время аппаратные методы ускорения умножения, основанные на использовании комбинационных схем множительных устройств. Такие схемы реализуются в виде отдельных БИС или их композиции. Так, в составе БИС серии КР1802 имеется БИС-умножитель двух 16-разрядных операндов, вырабатывающий за один такт произведение двойной длины в прямом или дополнительном коде.

6.5. СТРУКТУРА АЛУ ДЛЯ ДЕЛЕНИЯ ЧИСЕЛ С ФИКСИРОВАННОЙ ЗАПЯТОЙ

Деление в ЭВМ обычно сводится к выполнению последовательности вычитаний делителя сначала из делимого, а затем из образующихся в процессе деления частичных остатков и сдвига частичных остатков.

Алгоритмы деления аналогичны алгоритму деления при ручном счете. Рассмотрим особенности деления на примере деления целых чисел. Пусть $Z = X/Y$, где X — делимое, представляемое обычно двойным словом ($2n - 1$ цифровых разрядов), Y — делитель и Z — частное, представляемые словами, содержащими $n - 1$ цифровых разрядов.

Будем для простоты считать, что делению подвергаются целые числа, представляемые в прямом коде. Так как Z — слово, то должно выполняться неравенство $|Z| < 2^{n-1}$. Это возможно при $(|X'| - |Y|) < 0$, где $|X'| = |X| \cdot 2^{-(n-1)}$. Для получения $|X'| - |Y|$ следует вычесть из делимого $|X|$ делитель $|Y|$, выровняв их так, чтобы младший разряд $|Y|$ был под n -м разрядом. Этого можно добиться, сдвинув $|Y|$ относительно $|X|$ на $n - 1$ разряд влево.

Если результат пробного вычитания больше 0, то $|Z| \geq 2^{n-1}$ и деление невозможно, если он меньше 0, то можно выполнить деление.

Рассмотрим пример деления. Пусть делимое X и делитель Y в прямом коде есть $X_{пр} = 10101001$ и $Y_{пр} = 0111$. Тогда $|X| = 00101001$ и $|Y| = 0111$. Частное Z должно быть представлено прямым кодом с четырьмя двоичными разрядами, старший из которых представляет собой знак и в $|Z|$ должен быть равен 0. Выполним деление $|X|$ на $|Y|$ обычным способом:

$$\begin{array}{rcl}
 \text{Пробное вычитание} & \left\{ \begin{array}{l} - \begin{array}{r} 00101001 \\ 0111 \\ \hline (<0) \end{array} & \begin{array}{r} 0111 \\ 0101 \end{array} \\ - \begin{array}{r} 001010 \\ 0111 \\ \hline 00110 \end{array} & (>0) \\ - \begin{array}{r} 00110 \\ 0111 \\ \hline (<0) \end{array} & \\ - \begin{array}{r} 001101 \\ 0111 \\ \hline 0110 \end{array} & (>0). \end{array} \right. \\
 \underbrace{\hspace{10em}} & & \\
 \text{Остаток} & &
 \end{array}$$

В соответствии с правилами деления очередной цифрой частного является 1, если после вычитания делителя из остатка получается положительный результат, и 0, если отрицательный. В последнем случае восстанавливается остаток, который был до вычитания. Затем делитель смещается на разряд вправо, и процедура повторяется.

В рассматриваемом примере по отрицательному результату пробного вычитания согласно общему правилу фиксируется цифра частного $z_0 = 0$ для единообразия процедуры деления. На ее место после завершения деления модулей заносится знак результата.

Так как знаки делимого X и делителя Y в примере различны, то знак Z отрицателен, поэтому от $|Z| = 0101$ переходим к прямому коду $Z_{пр} = 1101$.

Реализовать деление можно двумя основными способами.

1. Деление с неподвижным делимым и сдвигаемым вправо делителем. Этот способ деления основан на прямом копировании действий при ручном делении. Структура АЛУ для деления имеет вид, представленный на рис. 6.6, а. Сначала делимое X заносится в P_2X , а делитель Y — в старшие разряды P_2Y . Делитель сдвигается вправо путем косой передачи из P_2Y в P_2Z и прямой передачи из P_2Y в P_1Y . Вычитание делителя выполняется подсуммированием дополнительного кода делителя. Цифры частного, определяемые по знаку частичных остатков, фиксируются в регистре P_2Z путем последовательного занесения их в младший разряд P_1Z и сдвига содержимого P_2Z с помощью косой передачи в P_2Z и прямой из P_2Z в P_1Z .

Недостатком такого АЛУ является двойная длина сумматора и его регистров.

2. Деление с неподвижным делителем и сдвигаемым влево делимым. Этот способ позволяет строить АЛУ с сумматором одинарной длины (рис. 6.6, б). Здесь неподвижный делитель Y хранится в P_2Y , а делимое X , сдвигаемое влево относительно Y , находится в двух регистрах: старшие разряды X — в P_1X , а младшие — в P_2X . Деление начинается со сдвига влево делимого X путем косой передачи его в P_2C и P_2X и соответствующих прямых передач в P_1X и P_2X . Далее на вход сумматора подается сдвинутое влево делимое, образуется частичный остаток путем подсуммирования дополнительного кода делителя, и очередная цифра частного заносится в освободившийся при сдвиге X разряд P_2X .

Арифметическо-логическое устройство рассмотренного типа широко используется для деления.

Алгоритм деления с неподвижным делителем с восстановлением остатка:

1. Берутся модули от делимого и делителя.
2. Исходное значение частичного остатка полагается равным старшим разрядам делимого.
3. Частичный остаток удваивается путем сдвига на один разряд влево, при этом в освобождающийся при сдвиге младший разряд частичного остатка заносится очередная цифра делимого.
4. Из сдвинутого частичного остатка вычитается делитель и анализируется знак результата вычитания.
5. Очередная цифра модуля частного равна 1, если результат вычитания положителен, и 0, если отрицателен. В последнем случае значение остатка восстанавливается до того, которое было до вычитания.
6. Пункты 3–5 последовательно выполняются для получения всех цифр модуля частного.
7. Знак частного плюс, если знаки делимого и делителя одинаковы, и минус в противном случае.

Рассмотренный метод деления носит название *деления с восстановлением остатка*. Недостатком этого метода является необходимость введения специального такта для восстановления остатка.

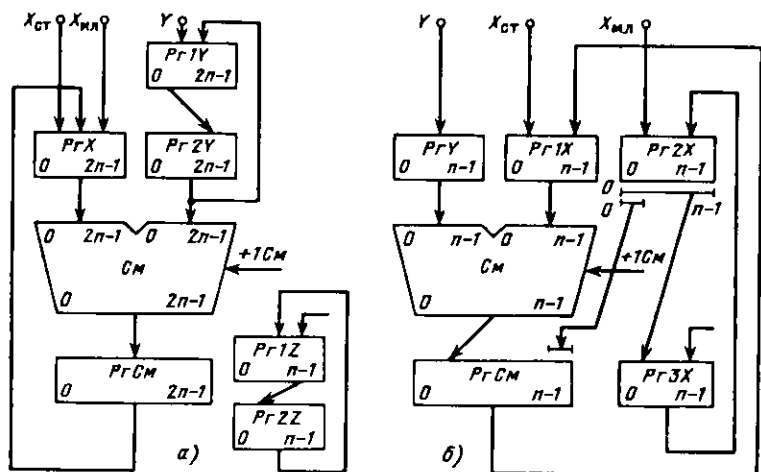


Рис. 6.6. Методы выполнения деления:

а — со сдвигаемым делителем; б — с неподвижным делителем

Обычно в ЭВМ для деления используется другой метод — *деление без восстановления остатка*.

Алгоритм деления с неподвижным делителем без восстановления остатка:

Пункты 1–3 совпадают с аналогичными пунктами алгоритма деления с восстановлением остатка.

4. Из сдвинутого частичного остатка вычитается делитель, если остаток положителен, и к сдвинутому частичному остатку прибавляется делитель, если остаток отрицателен.

5. Очередная цифра модуля частного равна 1, если результат вычитания положителен, и 0, если отрицателен.

Пункты 6, 7 совпадают с аналогичными пунктами предыдущего алгоритма.

Можно показать, что частичные остатки после выполнения сложения при делении без восстановления остатка получаются такими же, как и после сдвига восстановленного остатка при делении с восстановлением остатка.

Действительно, поскольку сдвиг частичного остатка на один разряд влево эквивалентен умножению его на два, то получим

$$2a - b = 2(a - b) + b, \quad (6.2)$$

где a — частичный остаток; b — делитель.

Деление без восстановления остатка всегда требует для получения одной цифры частного только сложения или вычитания и сдвига частичного остатка. После завершения всех циклов деления выдается результат, при этом если остаток отрицателен, то он восстанавливается путем подсуммирования Y .

Деление чисел, представленных в зависимости от знака прямым или дополнительным кодами, можно осуществлять, не переходя к модулям, при этом алгоритм деления оказывается подобным рассмотренным.

Отличия заключаются в следующем (для деления без восстановления остатка):

1. Так как делимое и делитель могут иметь разные знаки, то действия с частичным остатком (прибавление или вычитание Y) зависят от знаков остатка и делителя и определяются в соответствии с табл. 6.2.

Если знак остатка совпадает со знаком делителя, то $z_i = 1$, иначе $z_i = 0$.

2. Если $X > 0$ и $Y < 0$, частное необходимо увеличить на 1.

Если $X < 0$ и $Y > 0$, частное необходимо увеличить на 1 при остатке от деления, не равном 0.

Если $X < 0$ и $Y < 0$, частное необходимо увеличить на 1 при остатке от деления, равном 0.

Таблица 6.2

Знак остатка	Знак делителя	Действие
+	+	Вычитание Y
+	-	Прибавление Y
-	+	Прибавление Y
-	-	Вычитание Y

Деление правильных дробей выполняется так же, как деление целых чисел. Разница заключается только в том, что делимое имеет, как правило, такую же длину, как и делитель. Однако можно предположить, что делимое имеет еще n младших разрядов, равных 0. Тогда становится ясно, что алгоритм деления дробей ничем не отличается от алгоритма деления целых чисел.

Микропрограммы для операции деления приведены в [32].

6.6. УСТРОЙСТВА ДЛЯ ВЫПОЛНЕНИЯ ЛОГИЧЕСКИХ ОПЕРАЦИЙ

В состав операций, реализуемых ЭВМ, входят следующие поразрядные логические операции: суммирование по модулю 2, логическое умножение И, логическое сложение ИЛИ.

В результате поразрядной логической операции над словами X и Y формируется слово Z , в котором $Z[i] = X[i] * Y[i]$, где $*$ — символ выполняемой операции.

Для выполнения логических операций можно воспользоваться устройством, структурная схема которого приведена на рис. 6.7. Исходные операнды размещаются в регистрах P_2I и P_2J , откуда побайтно можно переносить их содержимое в P_2C и P_2D соответственно. Для реализации требуемых логических операций используется схема однобайтовых логических операций *СОЛО*, входящая в состав АЛУ и являющаяся комбинационной схемой, позволяющей реализовать поразрядные операции логического умножения И, логического сложения ИЛИ и суммирования по модулю 2 над двумя однобайтовыми операндами. Результат обработки байт фиксируется на байтовом регистре $P_2СОЛО$, из которого результат можно отправлять в выходной регистр P_2Cm ¹.

¹ Структурные схемы, рассматриваемые в гл. 6, являются фрагментами общей структурной схемы АЛУ, обеспечивающего выполнение всей совокупности операций процессора.

Помимо указанных логических операций в *СОЛО* выполняется операция сравнения двух байт, используемая, в частности, при обработке порядков в операциях сложения и вычитания над числами с плавающей запятой.

Восьмиразрядная схема однобайтовых логических операций (рис. 6.8) состоит из восьми схем поразрядной обработки СПО и схемы сравнения слов длиной 1 байт.

На выходах *СОЛО* $K_0 - K_7$ формируется поразрядная конъюнкция двух байт, поданных на схему. На выходах $M_0 - M_7$ образуется поразрядная сумма по модулю 2.

На *Вых1* и *Вых2* формируются сигналы, определяющие результат сравнения байт по численному значению в соответствии со следующим правилом:

<i>Вых1</i>	<i>Вых2</i>	Результат сравнения
1	1	$D < C$
0	1	$D > C$
0	0	$D = C$

На выходе K_i образуется конъюнкция

$$K_i = \bar{C}_i \vee \bar{D}_i = C_i D_i,$$

где C_i и D_i — значения i -х разрядов кодов, хранящихся соответ-

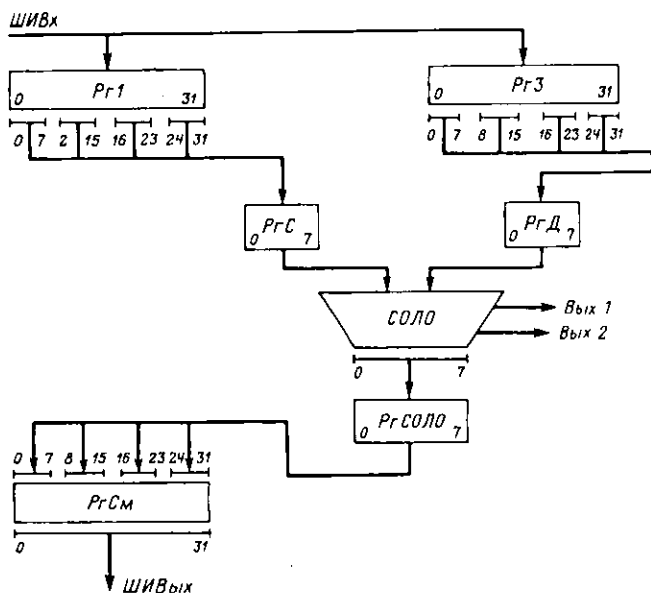
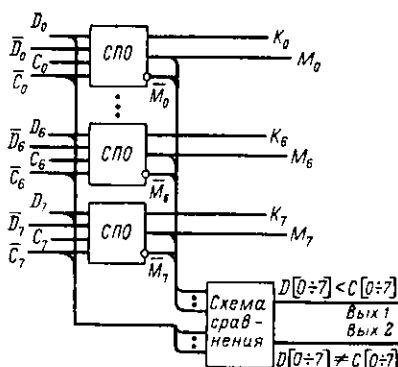


Рис. 6.7. Устройство для выполнения логических операций

Рис. 6.8. Комбинационная схема поразрядной обработки



ственно на регистрах P_2C и P_2D . На выходе M_i формируется значение суммы по модулю 2 в соответствии с выражением

$$M_i = \bar{C}_i \bar{D}_i \vee C_i D_i = C_i \oplus D_i.$$

Выходы K_i и M_i соединены со входами соответствующих разрядов регистра $P_2CОЛО$. При выполнении микрооперации $P_2CОЛО := P_2C \wedge P_2D$ в $P_2CОЛО$ заносятся значения состояний выходов K_i . При выполнении микрооперации $P_2CОЛО := P_2C \oplus P_2D$ в $P_2CОЛО$ заносятся значения состояний выходов M_i . Если одновременно выполняются две микрооперации $P_2CОЛО := P_2C \wedge P_2D$ и $P_2CОЛО := P_2C \oplus P_2D$, в $P_2CОЛО$ заносится результат поразрядной операции ИЛИ $P_2C \vee P_2D$.

6.7. ОСОБЕННОСТИ ОПЕРАЦИЙ ДЕСЯТИЧНОЙ АРИФМЕТИКИ

Арифметические операции над десятичными числами (сложение, вычитание, умножение, деление) выполняются аналогично операциям над целыми двоичными числами. Основой АЛУ десятичной арифметики является сумматор двоично-десятичных кодов. Такой сумматор, как правило, строится на основе двоичного путем добавления некоторых цепей.

Рассмотрим, каким образом можно выполнить сложение двоично-десятичных кодов. Пусть необходимо сложить модули двух двоично-десятичных чисел X и Y . Первое слагаемое X преобразуем в код с избытком 6 (обозначим X_6), получаемый путем прибавления к каждой цифре X двоичного числа 6. Переход от X к X_6 изменяет все тетрады X так, что в каждой тетраде X_6 находится число 6 – 15.

Складывая X_6 и Y по правилам двоичного сложения, получаем результат Z' . В Z' одни тетрады совпадают, а другие не совпадают с тетрадами двоично-десятичной суммы Z .

Если результат сложения в i -м разряде $X[i] + Y[i] + P[i] \geq 10$, где $P[i]$ – десятичный перенос в i -й разряд, то i -я деся-

тичная цифра $Z[i] = X[i] + Y[i] + P[i] - 10$ и $P[i+1] = 1$, где $P[i+1]$ — десятичный перенос в $(i+1)$ -й разряд.

Для $Z'[i]$ в этом случае получаем

$$Z'[i] = X6[i] + Y[i] + P[i] - 16 = 6 + \underbrace{X[i] + Y[i] + P[i]}_{\substack{> 10 \\ > 16}} - 16 = Z[i].$$

При этом возникает перенос в $(i+1)$ -ю тетраду.

Если i -я десятичная цифра $Z(i)$ должна получаться из $X[i] + Y[i] + P[i] < 10$, то $Z[i] = X[i] + Y[i] + P[i]$ и $P[i+1] = 0$.

Для $Z'(i)$ в этом случае получаем

$$Z'[i] = X6[i] + Y[i] + P[i] = 6 + \underbrace{X[i] + Y[i] + P[i]}_{\substack{< 10 \\ < 16}} = Z[i] + 6.$$

Перенос в $(i+1)$ -ю тетраду здесь не возникает ($P[i+1] = 0$), так как $Z'[i] < 16$.

Таким образом, складывая $X6$ и Y как двоичные числа, получаем Z' . В Z' тетрады, из которых возникал перенос, совпадают с тетрадами двоично-десятичного результата Z , а тетрады, из которых не было переноса при сложении, представлены с избытком 6. Для получения суммы Z необходимо откорректировать Z' путем уменьшения на 6 тех тетрад Z' , из которых не было переноса при сложении $X6$ и Y .

Вычитание 6 из тетрад, требующих коррекции, можно реализовать путем подсуммирования 10 с одновременным игнорированием переноса, возникающего при этом из тетрад. Если $Z'[i]$ нуждается в коррекции, то $Z'[i] = Z[i] + 6$. Поэтому $Z'[i] + 10 \geq 16$, значит, после прибавления 10 из тетрады возникнет перенос, т. е. в тетраде останется $(Z'[i] + 10) - 16 = Z'[i] - 6$.

Вычитание двоично-десятичных модулей $X - Y$ осуществляется следующим образом.

Все разряды Y инвертируются, что дает дополнение каждой цифры Y до 15, при этом получается обратный код двоично-десятичного Y с избытком 6, обозначенный $Y_{обр6}$. Затем, складывая $X + Y_{обр6}$ и прибавляя 1 к младшему разряду, получаем Z' . Результат Z' является положительным числом, если из старшей тетрады его возникает перенос, при этом Z' корректируется по тем же правилам, что и при сложении модулей.

Если из старшей тетрады Z' нет переноса, то получен отрицательный результат, представленный в дополнительном коде. В этом случае код Z' инвертируется и к нему прибавляется 1 младшего разряда. Новое Z' корректируется, при этом к тетрадам, из которых возникал перенос при получении $(X + Y_{\text{обр}}6 + 1)$, прибавляется 10, а к остальным не прибавляется.

Выполнение сложения и вычитания чисел со знаками сводится к выполнению сложения или вычитания модулей путем определения фактической выполняемой операции по знакам операндов и виду выполняемой операции. Знак результата определяется отдельно. Например, при $X < 0$ и $Y < 0$ вычитание $X - Y$ заменяется вычитанием $|Y| - |X|$. Затем знак результата меняется на противоположный знаку $(|Y| - |X|)$.

Двоично-десятичное умножение сводится к образованию и многократному сложению частичных двоично-десятичных произведений. Умножение двоично-десятичных чисел выполняется следующим образом:

- 1) сумма частичных произведений полагается равной нулю;
- 2) анализируется очередная цифра (тетрада) множителя, и множимое прибавляется к сумме частичных произведений столько раз, какова цифра множителя;
- 3) сумма частичных произведений сдвигается вправо на 1 тетраду, и повторяются действия, указанные в п. 2, пока все цифры множителя не будут обработаны.

Для ускорения умножения часто отдельно формируются кратные множимого $8X$, $4X$, $2X$ и $1X$, при наличии которых уменьшается число сложений при выполнении п. 2.

Двоично-десятичное деление выполняется путем многократных вычитаний, подобно тому, как это делается при обычном делении.

Двоично-десятичные АЛУ часто строятся как последовательно-параллельные, осуществляющие последовательную обработку байт.

6.8. ОПЕРАЦИИ НАД ЧИСЛАМИ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Арифметические операции над числами с плавающей запятой более сложны, чем операции над числами с фиксированной запятой.

Сложение (вычитание) чисел с плавающей запятой выполняется в соответствии с (6.3) в предположении, что $|X| \geq |Y|$:

$$Z = X \pm Y = S^p x q_x + S^p y q_y = S^p x \left(q_x \pm \frac{q_y}{S^{(p_x - p_y)}} \right) = S^p z q_z \quad (6.3)$$

Алгоритм сложения и вычитания чисел с плавающей запятой:

1. Производится выравнивание порядков чисел. Порядок меньшего (по модулю) числа принимается равным порядку большего, а мантисса меньшего числа сдвигается вправо на число S -ичных разрядов, равное разности порядков чисел.

2. Производится сложение (вычитание) мантисс, в результате чего получается мантисса суммы (разности).

3. Порядок результата принимается равным порядку большего числа.

4. Полученная сумма (разность) нормализуется.

Выравнивание порядков начинается с их сравнения. Мантисса числа с меньшим порядком при выравнивании сдвигается вправо на число разрядов, равное разности порядков. Поскольку рассматриваемые числа с плавающей запятой имеют $S = 16$, сдвиг осуществляется шестнадцатичными разрядами, т. е. каждый сдвиг производится на четыре двоичных разряда.

При сравнении порядков возможны пять случаев:

1) $p_X - p_Y > m$ (m — число разрядов мантиссы). В качестве результата суммирования сразу может быть взято первое слагаемое, так как при выравнивании порядков все разряды мантиссы второго слагаемого принимают нулевое значение;

2) $p_Y - p_X > m$. В качестве результата суммирования может быть взято второе слагаемое;

3) $p_X - p_Y = 0$. Можно приступить к суммированию мантисс;

4) $p_X - p_Y = k_1$ ($k_1 < m$). Мантисса второго слагаемого сдвигается на k_1 разрядов вправо, затем производится суммирование мантисс;

5) $p_Y - p_X = k_2$ ($k_2 < m$). Перед выполнением суммирования мантисс производится сдвиг на k_2 разрядов вправо мантиссы первого слагаемого.

За порядок результата при выполнении суммирования принимается больший из порядков операндов.

Сложение (вычитание) мантисс производится по правилам сложения (вычитания) чисел с фиксированной запятой.

Нормализация суммы (разности) производится в случае невыполнения условия $1 > |q_Z| \geq 1/s$, при этом, если $|q_Z| \geq 1$, p_Z увеличивается на 1, а мантисса q_Z сдвигается на один S -ичный разряд вправо, что дает $|q_Z| < 1$. Если $|q_Z| < 1/s$, то мантисса результата сдвигается на разряд влево при одновременном уменьшении порядка результата на 1. Эти операции производятся до тех пор, пока не станет выполняться условие $q_Z \geq 1/s$. (При $q_Z = 0$ нормализация не выполняется.)

При получении порядка $+p_z$, переполняющего разрядную сетку, должен формироваться сигнал прерывания из-за переполнения порядка. При получении порядка $-p_z$, переполняющего разрядную сетку, формируются нулевой результат и признак исчезновения порядка.

В операциях с плавающей запятой в отличие от операций с фиксированной запятой сложение и вычитание выполняются приближенно, так как при выравнивании порядков происходит потеря младших разрядов одного из слагаемых. В этом случае погрешность всегда отрицательна и может доходить до единицы младшего разряда. Чтобы уменьшить погрешность, применяют округление результата. Для этого может быть использован дополнительный разряд сумматора, в который после выполнения суммирования добавляется 1.

Умножение чисел с плавающей запятой выполняется в соответствии с формулой

$$Z = XY = S^p x q_x S^{p_y} q_y = S^{(p_x + p_y)} q_x q_y = S^{p_z} q_z. \quad (6.4)$$

При умножении чисел с плавающей запятой порядки сомножителей складываются, а мантиссы перемножаются. Произведение нормализуется, и ему присваивается знак плюс, если сомножители имеют одинаковые знаки, и знак минус, если знаки разные.

Если мантисса множимого или множителя равна 0, то произведению можно присвоить значение 0 без выполнения умножения мантисс. Если при суммировании порядков возникло переполнение и порядок отрицательный, то это означает, что произведение меньше минимального представляемого в машине числа, и в качестве результата операции может быть записан 0 без перемножения мантисс.

Если при суммировании порядков возникает переполнение и порядок положительный, может оказаться, что результат все-таки находится в диапазоне чисел, представляемых в машине, так как при умножении мантисс возможно нарушение нормализации вправо, и после нормализации мантиссы переполнение в порядке может исчезнуть.

Деление чисел с плавающей запятой выполняется в соответствии с формулой

$$Z = S^p x q_x / S^{p_y} q_y = S^{(p_x - p_y)} q_x / q_y = S^{p_z} q_z. \quad (6.5)$$

При делении чисел с плавающей запятой мантисса частного равна частному от деления мантиссы делимого на мантиссу делителя, а порядок частного — разности порядков делимого и делителя. Частное нормализуется, и ему присваивается знак

плюс, если делимое и делитель имеют одинаковые знаки, и знак минус, если разные.

Если делимое равно 0, то в частное может быть записан 0 без выполнения деления. Если при вычитании порядков образовалось переполнение с положительным знаком или если делитель равен 0, то деление не производится и формируется сигнал прерывания.

При делении нормализованных чисел с плавающей запятой может оказаться, что мантисса делимого больше мантиссы делителя, и мантисса частного образуется с переполнением. Для устранения этого явления перед делением мантисс нарушают нормализацию делителя сдвигом на разряд влево. Тогда нарушения нормализации частного влево не возникает.

Деление мантисс выполняется, как правило, методом без восстановления остатка аналогично делению целых чисел. Отличие заключается в том, что делимое берется такой же длины, как и делитель. Однако нетрудно заметить, что для дробей можно условно принять, что делимое имеет двойную длину с нулями в разрядах младшей половины числа. После сдвигов влево частичных остатков освобождающиеся разряды всегда заполняются 0 и деление можно выполнять в точности так же, как деление целых чисел.

Более детально алгоритмы и микропрограммы выполнения операций над числами с плавающей запятой изложены в [32].

6.9. МНОГОФУНКЦИОНАЛЬНОЕ АЛУ

Проектирование АЛУ включает в себя выбор кодов для представления данных, определение алгоритмов выполнения отдельных операций, структур операционных блоков и реализуемых в них наборов микроопераций. Затем производят объединение отдельных операционных блоков и соответствующих наборов микроопераций в один многофункциональный операционный блок или несколько блоков для отдельных групп операций. В многофункциональных АЛУ операции над числами с фиксированной и плавающей запятой, десятичными числами и алфавитно-цифровыми полями выполняются в основном одними и теми же схемами, коммутируемыми соответствующим образом. На рис. 6.9 приведена схема многофункционального АЛУ для выполнения совокупности рассматривавшихся арифметических и логических операций. В данной схеме в качестве отдельных фрагментов можно выделить описанные выше АЛУ, при этом для сокращения общего числа связей некоторые связи следует удалить (фактически заменить другими). Так, например, при сложении чисел с фиксированной запятой в отличие от АЛУ на рис. 6.1 в рассматриваемой схеме загрузка P_2B проис-

ходит не от ШИВх, а от P_2 ввиду того, что связь от ШИВх к P_2 и далее к P_2V должна существовать из-за необходимости реализации умножения (см. рис. 6.4). Сумма частичных произведений заносится в P_2V не непосредственно из P_2C , а через P_23 , так как загрузка P_23 необходима при выполнении сложения чисел с плавающей запятой и т. п.

Операции двоично-десятичной арифметики в данном АЛУ производятся при помощи двоично-десятичного сумматора $CмДес$ и побайтной организации обработки.

При выполнении операций над числами с плавающей запятой используются двоичный сумматор $Cм$ и схема СОЛО. При сложении (вычитании) чисел с плавающей запятой первое слагаемое (уменьшаемое) поступает на входной регистр P_21 , второе (вычитаемое) — на входной регистр P_23 . Знаки слагаемых хранятся в триггерах знаков $ТгЗн1$ и $ТгЗн2$. Смещенные поряд-

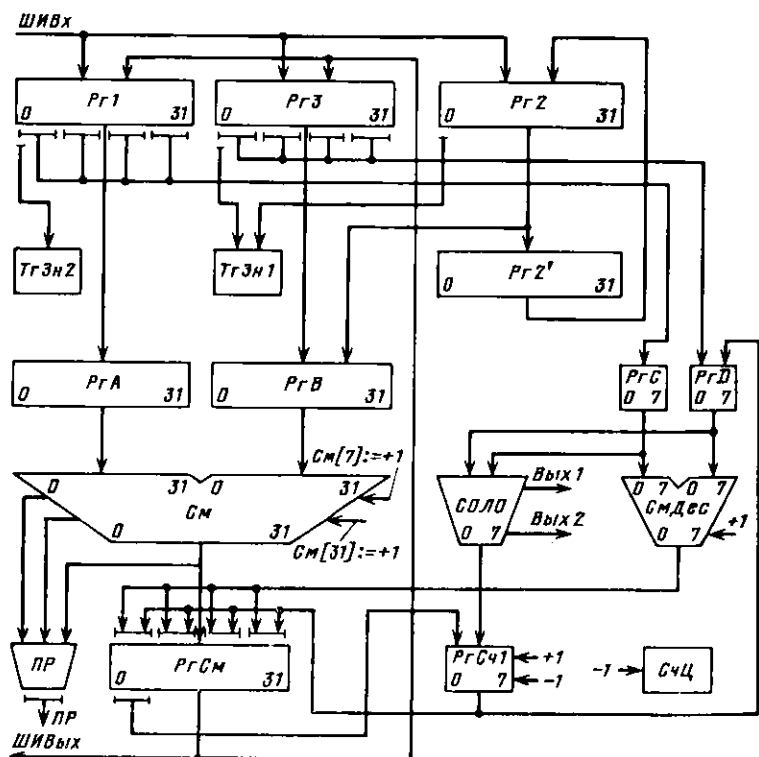


Рис. 6.9. Многофункциональное АЛУ

ки слагаемых пересылаются в регистры P_2C и P_2D . Схема СОЛО применяется для сравнения и выравнивания порядков слагаемых. Сумматор C_m , его входные регистры P_2A и P_2B и выходной регистр P_2C_m используются при сложении (вычитании) мантисс, а также при передаче мантисс со сдвигом в процедурах выравнивания порядков и нормализации результата.

Выравнивание порядков осуществляется следующим образом. Смещенный порядок числа X из P_2Z передается в регистр P_2D и в выполняющий роль $P_2СОЛО$ счетчик P_2C_41 , соединенный с выходом СОЛО. Затем в P_2C передается смещенный порядок числа Y .

После этого начинается сравнение порядков чисел X и Y на СОЛО и сдвиг мантиссы числа с меньшим порядком вправо, при этом значение смещенного порядка Y меняется до тех пор, пока он не станет равным смещенному порядку X . Порядок Z берется равным большему порядку слагаемых.

Чтобы не делать лишних сдвигов мантиссы, превратившейся в процессе выравнивания порядка в 0, на счетчике циклов $C_4Ц$ фиксируется предельное число сдвигов, равное числу шестнадцатиричных цифр мантиссы:

$$C_4Ц: = 6.$$

При выполнении сдвига на один шестнадцатиричный разряд содержимое $C_4Ц$ уменьшается на 1. При $C_4Ц = 0$ сдвиги прекращаются и в качестве результата берется большее слагаемое.

После выравнивания порядков осуществляется сложение мантисс и (при необходимости) нормализация результата.

При умножении чисел с плавающей запятой используются: сумматор C_m ; регистр P_2I для хранения множимого; регистры P_22 и P_22' для приема и сдвига множителя в процессе умножения мантисс; регистр P_2A , используемый для передачи на сумматор смещенного порядка множимого при суммировании порядков и для передачи на сумматор мантиссы множимого при умножении мантисс; регистр P_2B , служащий для передачи на сумматор смещенного порядка множителя при суммировании порядков и для хранения текущей суммы частичных произведений при умножении мантисс; выходной регистр сумматора P_2C_m , фиксирующий результаты суммирований; счетчик P_2C_41 , хранящий смещенный порядок произведения; триггеры знаков сомножителей T_23n1 и T_23n2 .

При выполнении деления чисел с плавающей запятой используются: сумматор C_m ; регистры P_2I и P_22 для приема соответственно делителя и делимого; регистры P_2A и P_2B для

хранения смещенных порядков делителя и делимого и для хранения мантиссы делителя и частичного остатка при получении мантиссы частного; счетчик $Cч1$ для хранения смещенного порядка частного; регистры P_{22} и P_{22}' для хранения цифровых разрядов мантиссы частного; триггеры знаков делимого и делителя T_{23n1} и T_{23n2} . Рассмотренное АЛУ можно считать типичным для ЭВМ средней производительности. Несколько иначе строятся АЛУ для малых ЭВМ и микропроцессоров.

6.10. ОСОБЕННОСТИ АЛУ МИКРОПРОЦЕССОРОВ

Для микро-ЭВМ и микропроцессоров типичной является такая организация, при которой их внутренние регистры используются в различных целях. Система связей у этих регистров, как правило, централизованная (магистральная), обеспечивающая возможность разнообразных межрегистровых пересылок, в том числе передач в АЛУ и из АЛУ. В связи с этим часто собственные регистры АЛУ (регистры, используемые только для выполнения арифметических и логических операций) в микропроцессорах отсутствуют. Это даст повод рассматривать АЛУ микропроцессоров как комбинационную схему, выполняющую арифметические и логические операции над операндами, находящимися в регистрах микропроцессора. Результат операции засылается в некоторый регистр микропроцессора.

Подобные АЛУ входят в состав большинства микропроцессоров: K580, K584 и др.

Примером комбинационного АЛУ может служить микросхема K155ИПЗ, имеющая пять управляющих входов, сигналы которых настраивают ее на выполнение одной из 32 арифметическо-логических одноктактных операций над двумя 4-разрядными операндами.

В процессе выполнения операций комбинационное АЛУ взаимодействует с регистрами микропроцессора, являющимися обычно источниками и приемниками операндов для такого АЛУ, при этом, как правило, один и тот же регистр может рассматриваться и как источник, и как приемник информации. Для реализации такой возможности необходимо осуществлять временное запоминание промежуточных результатов на отдельных регистрах. С этой целью используют либо регистры для кратковременного запоминания операндов, либо регистры для кратковременного запоминания результата.

На рис. 6.10,а показана схема включения комбинационного АЛУ в контур с регистрами микропроцессора для выполнения арифметических операций. В приведенной схеме имеются регистры процессора: $P_{2П}$ (регистр признака результата), $P_{2Акк}$

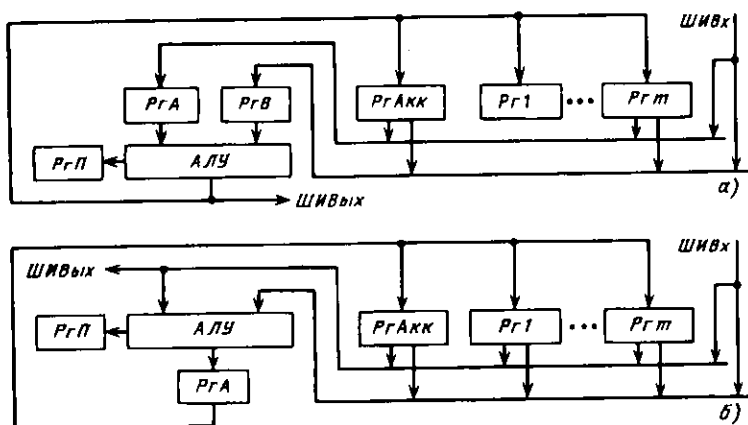


Рис. 6.10. Схемы с комбинационным АЛУ:

а — с регистрами временного запоминания операндов; б — с одним регистром временного запоминания результата

(аккумулятор), $Pr1, \dots, Prm$, которые могут использоваться произвольным образом, и регистры временного хранения операндов PrA и PrB , в которые при выполнении арифметических и логических операций загружаются операнды.

Пусть, например, выполняется операция сложения двух чисел, находящихся в регистрах процессора Prj и Pri , с засылкой результата в Prj . Эта операция потребует сначала пересылки содержимого Prj и Pri в PrA и PrB , а затем загрузки результата, сформированного АЛУ, в Prj .

Отсутствие PrA привело бы к возникновению «порочной петли», так как изменения состояний Prj влекли бы за собой новые изменения состояний Prj .

Для выполнения операции вида $Prj := Prj + Prj$ возникает необходимость в двух регистрах временного хранения операндов.

Схема на рис. 6.10, б аналогична только что рассмотренной. Отличается она лишь тем, что временному хранению подвергается в ней результат операции.

Арифметическо-логические устройства, используемые в рассматриваемых схемах, представляют собой комбинационные схемы, настраиваемые сигналами микроопераций на различные преобразования. Это может быть двоичное или двоично-десятичное сложение, вычитание, логическое умножение и т. п. При написании микропрограмм операций в АЛУ в микрокомандах задаются микрооперации, определяющие выбор источников

операндов для АЛУ, настраивающие АЛУ на выполнение различных преобразований и указывающие место занесения результата, сформированного АЛУ.

Рассматриваемые схемы являются фрагментами микропроцессора, в котором те же самые регистры используются и для других целей (см. гл. 8 и 9), что ведет к усложнению микропрограмм.

Глава седьмая

УПРАВЛЯЮЩИЕ АВТОМАТЫ

7.1. ОБЩИЕ СВЕДЕНИЯ

Как указывалось ранее, любое цифровое устройство можно рассматривать состоящим из двух блоков — операционного и управляющего.

Любая команда, операция или процедура, выполняемая в операционном блоке, описывается некоторой микропрограммой и реализуется за несколько тактов, в каждом из которых выполняется одна или несколько микроопераций.

Интервал времени, отводимый на выполнение микрооперации, называется рабочим тактом или просто тактом цифрового устройства. Если все такты имеют одну и ту же длину, то она устанавливается по самой продолжительной микрооперации.

Для реализации команды, операции или процедуры (микропрограммы) необходимо на соответствующие управляющие входы операционного блока подать определенным образом распределенную во времени последовательность управляющих функциональных сигналов. Например, для выполнения приведенной в предыдущей главе микропрограммы операции сложения и вычитания необходимо подвести к управляющим входам АЛУ следующую последовательность функциональных сигналов:

Такты

- 1 *ПрРгВ*
- 2 *ПрРг1*
- 3 *ПрРгАП* или *ПрРгАИ*
- 4 *ПрРгСм* или *ПрРгСм. + 1 См*
- 5 *ПрШИВых*

Подчеркнем, что каждый управляющий функциональный сигнал поступает в начале некоторого такта на соответствующий

щий вход АЛУ, вызывая в этом такте выполнение в АЛУ определенной микрооперации (передача слова, суммирование кодов и др.)

Часть цифрового вычислительного устройства, предназначенная для выработки последовательностей управляющих функциональных сигналов, называется *управляющим блоком* или *управляющим устройством*. Генерируемая управляющим блоком последовательность управляющих сигналов задается поступающими на входы блока кодом операции, сигналами из операционного блока, несущими информацию об особенностях операндов, промежуточных и конечного результатов операции, а также синхросигналами, задающими границы тактов.

Формально управляющий блок можно рассматривать как конечный автомат, определяемый:

а) множеством двоичных выходных сигналов

$$V = \{v_1, v_2, \dots, v_m\},$$

соответствующих множеству микроопераций операционного блока. При $v_i = 1$ возбуждается i -я микрооперация;

б) множествами входных сигналов Z и U :

$$Z = \{z_1, z_2, \dots, z_p\};$$

$$U = \{u_1, u_2, \dots, u_n\},$$

соответствующих задаваемому блоку извне двоичному коду операции (Z) и двоичным оповещающим сигналам (U);

в) множеством подлежащих реализации микропрограмм, устанавливающих в зависимости от значений входных сигналов управляющие сигналы, выдаваемые блоком в определенные такты.

По множествам входных и выходных сигналов и микропрограммам определяется множество внутренних состояний блока

$$S = \{Q_0, Q_1, \dots, Q_r\},$$

мощность которого (объем памяти управляющего блока) в процессе проектирования стараются минимизировать.

Сказанное поясняет, почему управляющие блоки называются управляющими автоматами. Поскольку эти автоматы задаются микропрограммами, они часто именуются *микропрограммными автоматами*.

Управляющий автомат может быть задан как автомат Мура:

$$\begin{aligned} Q(t+1) &= A[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)]; \\ v_1(t) &= B_1[Q(t)]; \\ &\dots \dots \dots \\ v_m(t) &= B_m[Q(t)]; \end{aligned} \quad (7.1)$$

или как автомат Мили:

$$\begin{aligned} Q(t+1) &= A[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)]; \\ v_1(t) &= B_1[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)]; \\ &\dots \dots \dots \\ v_m(t) &= B_m[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)], \end{aligned} \quad (7.1a)$$

где функция переходов A и функции выходов B_i определяются заданной микропрограммой.

Существуют два основных типа управляющих автоматов:

1. Управляющий автомат с *жесткой, или схемной, логикой*. Для каждой операции, задаваемой, например, кодом операции команды, строится набор комбинационных схем, которые в нужных тактах возбуждают соответствующие управляющие сигналы. Другими словами, строится конечный автомат, в котором необходимое множество состояний представляется состояниями k запоминающих элементов:

$$q = \{q_1, q_2, \dots, q_k\},$$

где $k = \lceil \log_2(r+1) \rceil$ при $S = \{Q_0, Q_1, \dots, Q_r\}$, а функции переходов и выходов A и B_i реализуются с помощью комбинационных схем.

2. Управляющий автомат с *хранимой в памяти логикой* (с «запоминаемой или программируемой логикой»). Каждой выполняемой в операционном устройстве операции ставится в соответствие совокупность хранимых в памяти слов — микрокоманд, содержащих каждая информацию о микрооперациях, подлежащих выполнению в течение одного машинного такта, и указание (в общем случае зависящее от значений входных сигналов), какое должно быть выбрано из памяти следующее слово (следующая микрокоманда). Таким образом, в этом случае функции переходов и выходов A и B_i управляющего автомата реализуются хранимой в памяти совокупностью микрокоманд.

Последовательность микрокоманд, выполняющих одну машинную команду или отдельную процедуру, образует микропрограмму. Обычно микропрограммы хранятся в специальной памяти микропрограмм («управляющей памяти»).

В управляющих автоматах с хранимой в памяти программой микропрограммы используются в явной форме, они программируются в кодах микрокоманд и в таком виде заносятся в память. Поэтому такой метод управления цифровым устройством называется микропрограммированием, а использующие этот метод управляющие блоки — микропрограммными управляющими устройствами.

7.2. ПРИНЦИП ДЕЙСТВИЯ УПРАВЛЯЮЩЕГО АВТОМАТА С ХРАНИМОЙ В ПАМЯТИ ЛОГИКОЙ. МИКРОПРОГРАММНОЕ УПРАВЛЕНИЕ

Одним из достоинств микропрограммных устройств управления является их наглядность, облегчающая изучение процесса функционирования ЭВМ и их эксплуатацию. Поэтому рассмотрение методов построения устройств управления начнем с микропрограммных устройств, не учитывая того обстоятельства, что они стали применяться позже управляющих устройств с жесткой (схемной) логикой. В настоящее время микропрограммное управление является наиболее распространенным методом построения управления, по крайней мере, в процессорах машин малой и средней производительности, а также в других устройствах (каналах, устройствах управления периферийными устройствами и др.). Микропрограммное управление применяется в некоторых типах микропроцессоров.

Хранимая в памяти микропрограмма должна содержать информацию о функциях переходов и выходов управляющего микропрограммного автомата.

Интерпретируя рассматриваемый ниже УА в терминах конечных автоматов, обнаружим, что он функционирует подобно автомату Мили с задержанными на такт выходными сигналами. Аргументами функций переходов и выходов автомата являются входные переменные $Z(t)$ и $U(t)$ и переменные $q(t)$, задающие своими значениями состояние автомата $Q(t)$. Набор значений аргументов удобно отождествить с адресом микрокоманды. Этот адрес заносится в регистр адреса микрокоманды во время такта. В управляющей памяти (УП) по данному адресу хранится код, задающий набор значений выходных сигналов управляющего автомата $V(t+1)$, — операционная часть микрокоманды — и набор значений переменных $q(t+1)$, представляю-

Рис. 7.1. Структурная схема простейшего варианта управляющего автомата с хранимой в памяти логикой

щий состояние $Q(t+1)$. Значения $V(t+1)$ и $Q(t+1)$ определяются функциями переходов и выходов.

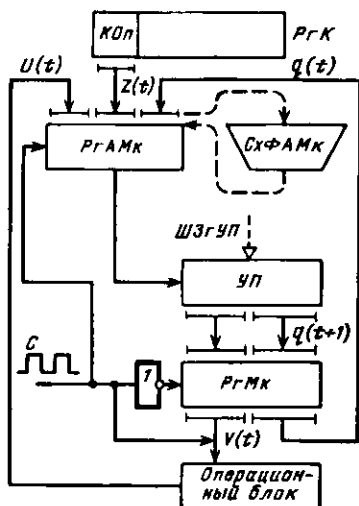
Структурная схема простейшего варианта управляющего автомата с хранимой в памяти программой приведена на рис. 7.1. Автомат работает следующим образом. Серия синхросигналов CC определяет такты работы автомата, при этом значение $CC = 1$ выделяет такт, а значение $CC = 0$ — паузу между тактами. Напомним, что значения входных и выходных сигналов и состояние автомата должны быть неизменными во время такта и могут меняться только в паузах.

Состояние автомата $Q(t)$ представляется набором значений переменных $q(t)$, в дальнейшем называемым для краткости кодом $Q(t)$.

Пусть в такте $(t-1)$ в P_2AMK занесены $U(t-1)$, $Z(t-1)$ и код $Q(t-1)$. Тогда в паузе перед тактом t при $CC = 0$ на P_2AMK эти значения сохраняются и из УП можно выбрать коды $V(t)$, которые у рассматриваемого нами автомата, как и $Q(t)$, зависят от $Q(t-1)$, $Z(t-1)$ и $U(t-1)$. Эти коды при $CC = 0$ заносятся в P_2Mk (одновременно происходит изменение значений входных сигналов). После возникновения $CC = 1$, задающего такт t , в P_2Mk хранятся сформированные коды $V(t)$ и $Q(t)$, при этом сигналы $V(t)$ используются для инициирования микроопераций, а код $Q(t)$ переносится в P_2AMK , после чего цикл работы автомата повторяется.

Воздействие управляющих сигналов $V(t)$ на операционный блок синхронизируется сигналом $CC = 1$, обеспечивающим выдачу $V(t)$ строго в такте t из P_2Mk , находящегося в режиме хранения.

Управляющая память может быть двух типов: постоянная и с произвольным обращением, т. е. допускающая как считывание, так и запись. В последнем случае загрузка УП производится пользователем со специального внешнего ЗУ по шине за-



грузки управляющей памяти ШЗУП при каждом включении машины в работу.

Анализируя рассматриваемую структурную схему, можно заметить, что число слов, хранимых в УП, очень велико из-за большой разрядности P_2AM_k , обусловленной значительным числом используемых оповещающих сигналов $U(t)$. Сократить объем УП можно, если учесть, что для каждого $Q(t)$ существенными являются значения не всех, а лишь некоторых переменных из $Z(t)$ и $U(t)$, задающих различные переходы из состояния $Q(t)$ в состояние $Q(t+1)$.

Число таких переходов в микропрограммных автоматах обычно невелико, и поэтому для каждого $Q(t)$ можно выделить группу адресов УП, хранящих коды только различных состояний перехода $Q(t+1)$. Объем УП при этом во много раз сократится, так как в УП не будут храниться многократно повторяющиеся коды, описывающие одинаковые переходы автомата.

Адрес микрокоманды при таком подходе формируется специальной комбинационной схемой формирования адреса микрокоманд $CxФAM_k$ по значениям $Q(t)$, $U(t)$ и $Z(t)$. Схема $CxФAM_k$ подключается ко входам P_2AM_k , как показано штриховыми линиями на рис. 7.1.

Адрес очередной микрокоманды можно назначить без учета значений $Z(t)$ и $U(t)$, если эта микрокоманда задает функцию перехода автомата в состоянии, имеющем единственный переход, не зависящий от значения входных сигналов. В этом случае адрес очередной микрокоманды можно указать значением отдельной группы разрядов исполняемой микрокоманды. Если очередная микрокоманда должна задавать функцию перехода автомата в состоянии, имеющем различные переходы, зависящие от значений входных сигналов, то ее адрес должен зависеть от входных сигналов.

Для получения простой схемы $CxФAM_k$ обычно используется следующий способ формирования очередного адреса. В микрокоманде выделяется помимо операционной части адресная. Адресная часть содержит несколько полей (групп разрядов): поле типа формирования адреса (ТФА) и поля формирования отдельных групп разрядов очередного адреса (ПФА). При определенных значениях поля ТФА очередной адрес формируется только из значений ПФА (в простейшем случае в P_2AM_k переносятся значения ПФА, заданные в микрокоманде).

Если адрес очередной микрокоманды должен формироваться с учетом значений входных сигналов, то в поле ТФА заносится специальный код, настраивающий $CxФAM_k$ на особую обработку ПФА. При этом содержимое некоторых ПФА по-прежнему

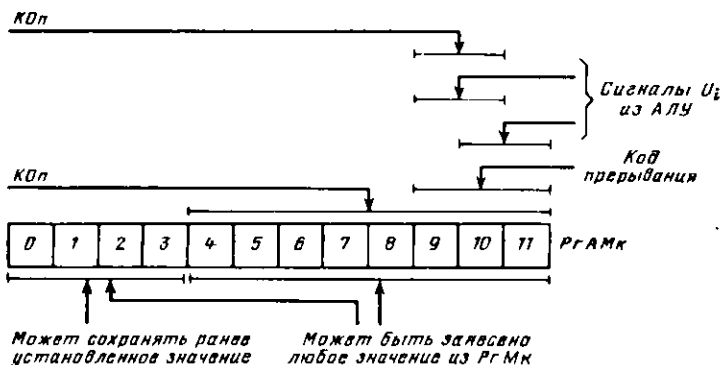


Рис. 7.2. Формат РгАМк ЭВМ ЕС-1030

переносится в РгАМк, а коды из других ПФА обеспечивают занесение в РгАМк значений, указываемых этими ПФА входных переменных. Таким образом, очередной адрес оказывается зависящим не только от ранее исполнявшейся микрокоманды (состояния автомата), но и от значений входных сигналов, влияющих на переходы автомата в новое состояние.

В качестве примера рассмотрим, как формируется содержимое РгАМк микропрограммного управления ЭВМ ЕС-1030.

Регистр РгАМк состоит из 12 разрядов (рис. 7.2). В разряды 0—3 и 4—11 РгАМк можно записывать содержимое определенных полей РгМк, что обеспечивает возможность задания любого адреса УП без учета значений входных сигналов. При выполнении микрокоманд, формирующих очередной адрес с учетом значений входных сигналов, в РгАМк можно заносить: в разряды 4—11 код операции, в разряды 9—10 отдельные разряды КОп, в разряды 9—11 код прерывания, в разряды 8, 9 и 10, 11 значения различных оповещающих сигналов. Пользуясь этим при составлении микропрограмм, можно организовывать последовательность выполнения микрокоманд, зависящую от значений входных сигналов управляющего автомата. Для этого необходимо, кодируя каждую микрокоманду, задавать способ формирования адреса следующей микрокоманды.

Рассмотренный способ формирования адреса следующей микрокоманды носит название *принудительного формирования адреса*.

Формирование адреса следующей микрокоманды иногда осуществляют *способом естественной адресации* с помощью счетчика. Безусловный переход от микрокоманды с адресом i осуществляется к микрокоманде с адресом $i + 1$, что позво-

ляет иметь в микрокоманде только операционную часть. Однако при этом в микропрограмму вводят помимо операционных адресные микрокоманды, различаемые по специальному признаку. В адресных микрокомандах отсутствует операционная часть и все разряды используются в качестве полей ТФА и ПФА, таких же, как и в адресной части рассматривавшихся выше микрокоманд. Такой способ формирования адреса микрокоманды вызывает усложнение схем дешифрирования микрокоманд и увеличение длины микропрограмм, но сокращает длину микрокоманд.

Примером использования естественной адресации микрокоманд может служить управляющий блок микропроцессора Ат 2900[11].

Управляющие автоматы с хранимой в памяти логикой различаются по способу формирования управляющих функциональных сигналов. Возможно использование горизонтального, вертикального и смешанного микропрограммирования.

Горизонтальное микропрограммирование. Каждому разряду операционной части микрокоманды ставится в соответствие определенный управляющий функциональный сигнал, т. е. определенная микрооперация. Если в разряде стоит 1, то соответствующая микрооперация выполняется независимо от значения других разрядов. При таком способе операционная часть микрокоманды содержит m разрядов, где m — общее число микроопераций. Достоинствами горизонтального микропрограммирования являются возможность одновременного выполнения в одном такте любого набора из m микроопераций и простота формирования функциональных сигналов, так как последние могут возбуждаться непосредственно от сигналов из регистра микрокоманды.

Однако оно имеет и существенный недостаток, заключающийся в том, что требуется большая длина микрокоманды, поскольку число функциональных сигналов в современном процессоре может достигать нескольких сотен. Поэтому, хотя и известны случаи практического применения горизонтального микропрограммирования, главным образом в малых машинах, где число управляющих функциональных сигналов сравнительно невелико (около 150), большее распространение получили другие методы.

При *вертикальном микропрограммировании* микрооперация определяется не состоянием одного из разрядов микрокоманды, а двоичным кодом, содержащимся в операционной части микрокоманды, при этом отдельный код задает отсутствие микрооперации.

Число разрядов операционной части микрокоманды

$$n_{o.ч} = \lceil \log_2 (m + 1) \rceil.$$

Достоинством вертикального микропрограммирования является небольшая длина микрокоманды. Однако в этом случае требуются сложные дешифраторы на большое число микроопераций, а главное — в каждой микрокоманде указывается лишь одна микрооперация, что приводит к увеличению длины микропрограмм по сравнению с их длиной при горизонтальном микропрограммировании. Вертикальное микропрограммирование часто используется в микропроцессорах, как, например, для управления центральными процессорными элементами набора K589.

В настоящее время наибольшее распространение имеет *смешанное микропрограммирование*, в котором сочетаются горизонтальное и вертикальное микропрограммирование.

При смешанном микропрограммировании множество микроопераций V разбивается на k подмножеств (или полей):

$$V = \bigcup_{i=1}^k V_i.$$

Микрооперации внутри каждого из подмножеств кодируются либо горизонтальным, либо вертикальным способом. Можно выделить два способа.

Вертикально-горизонтальное микропрограммирование. Все множество микроопераций V расчленяют на k подмножеств V_i , в каждом из которых объединяют микрооперации, наиболее часто встречающиеся вместе в одном такте. Подмножества стараются по возможности сделать равномошными. Операционная часть микрокоманды (рис. 7.3) состоит из двух полей. В первом поле, длина которого равна $\max |V_i|$, применен горизонтальный способ кодирования микроопераций, т. е. каждый разряд соответствует определенной микрооперации из подмножества V_i , а другое поле, имеющее длину $\lceil \log_2 k \rceil$, указывает, к какому из k подмножеств принадлежат микрооперации в первом поле микрокоманды.

Более гибким и часто используемым является горизонтально-вертикальный способ кодирования.

Горизонтально-вертикальное микропрограммирование (рис. 7.4). Подмножества V_i кодируются горизонтальным, а микрооперации внутри каждого из подмножеств — вертикальным способом. В этом случае каждому подмножеству V_i выделяется

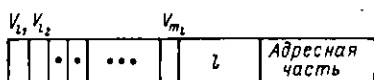
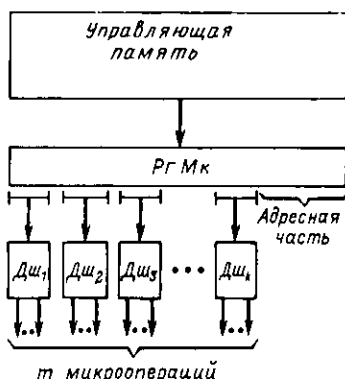


Рис. 7.3. Структура микрокоманды при вертикально-горизонтальном микропрограммировании

Рис. 7.4. Структура управляющего автомата при горизонтально-вертикальном микропрограммировании



отдельное поле в операционной части микрокоманды. Длина операционной части микрокоманды

$$n_{o.ч} = \sum_{l=1}^k \lceil \log_2(m_l + 1) \rceil,$$

где m_l — число микроопераций, представляемых в поле l .

Микрокоманды такого типа называют микрокомандами с полевой структурой. При таком способе кодирования желательно, чтобы

$$V_i \cap V_j = \emptyset \text{ при } i \neq j,$$

т. е. чтобы каждая микрооперация встречалась только в одном поле.

При прямом кодировании микрокоманд каждое поле микрокоманды несет фиксированные функции. Косвенное кодирование характеризуется наличием дополнительных полей, содержимое которых меняет смысл основных полей микрокоманды. Таким образом, интерпретация полей, формирующих управляющие сигналы, зависит от бит дополнительных полей.

Косвенное кодирование широко используется, так как позволяет уменьшить длину микрокоманды. Однако оно в некоторой степени нарушает стройность микропрограммного управления, вызывает усложнение дешифраторов и приводит к снижению скорости работы из-за потерь времени на дешифрирование дополнительных полей микрокоманды.

Различают одно- и многофазные микрокоманды. В первом случае все микрооперации, указанные в микрокоманде, выполняются одновременно в течение одного такта. Во втором — такт разбивается на части, называемые фазами или микротактами,

и указанные в команде микрооперации выполняются в различные микротакты (фазы такта). В этом случае приходится учитывать временные зависимости между отдельными микрооперациями. Однако становится возможным включать в микрокоманду взаимно исключающие микрооперации, разводя их по разным тактам. Многофазные микрокоманды часто применяются в ЭВМ с горизонтальным или горизонтально-вертикальным микропрограммированием (например, в ЕС ЭВМ).

Для повышения скорости работы управляющего автомата (УА) с хранимой в памяти логикой используются различные методы. Чаще всего осуществляется совмещение отдельных действий, задаваемых микрокомандой, что особенно существенно при использовании многофазных микрокоманд. Например, в ЭВМ ЕС-1030 двухфазные микрокоманды выполняются с совмещениями вида, указанного на рис. 7.5. В цикле i работы осуществляется выполнение микроопераций i -й микрокоманды $ВЫП Mo_i$, а одновременно с этим происходит формирование адреса очередной микрокоманды $ФА_{i+1}$ и ее выборка $ВЫБ_{i+1}$. При таком совмещении адрес $(i+1)$ -й микрокоманды формируется i -й микрокомандой с учетом значений оповещающих сигналов, представляющих собой результаты $(i-1)$ -й микрокоманды. Учет результатов i -й микрокоманды может осуществляться только $(i+1)$ -й микрокомандой при формировании адреса $(i+2)$ -й микрокоманды.

Хотя идея микропрограммирования известна с 1951 г. [125], однако еще до недавнего времени этот принцип управления не находил широкого применения в ЭВМ из-за отсутствия достаточно надежных и дешевых быстродействующих УП для хранения микропрограмм и сложности разработки текстов микропрограмм. Однако в последние годы интерес к микропрограммному принципу управления (к УА с хранимой в памяти логикой) возродился. Были созданы постоянные запоминающие устройства для УП с циклом обращения 0,25–0,5 мкс. Появились интегральные ПЗУ и ЗУ с произвольным обращением, обладающие еще большим быстродействием. Выяснилось, что

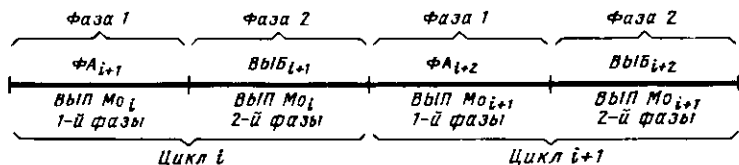


Рис. 7.5. Временная диаграмма работы управляющего автомата с хранимой в памяти логикой

эффективность методов построения УА зависит от числа и сложности команд ЭВМ и при увеличении и усложнении команд и функций процессора эффективность микропрограммного управления растет.

Использование принципа микропрограммного управления позволяет строить более регулярные схемы УА, создавать эффективные системы диагностики для автоматического поиска неисправностей, упрощает документирование алгоритмов работы УА. В последнее время микропрограммирование используется как средство для аппаратурной реализации фрагментов операционных систем, трансляторов и т. п.

Микропрограммирование широко используется для проблемной ориентации микропроцессорных устройств и систем при помощи специализированного набора команд, обеспечивающего наиболее эффективное решение определенных задач пользователя.

Все чаще в ЭВМ применяется загружаемая управляющая память, при этом в качестве первичных носителей микропрограмм используются гибкие диски или кассеты портативных магнитофонов, с которых микропрограммы загружаются в УП.

Микропрограммное управление с использованием загружаемой УП позволяет расширять или даже менять состав команд ЭВМ.

Наиболее выгодно использование УА с хранимой в памяти логикой для операционных блоков процессоров, в которых реализуются алгоритмы с относительно небольшим числом условий ветвления. Реализация в этих автоматах алгоритмов с большим числом условий ветвления ведет к значительному усложнению СхФАМк и, следовательно, к увеличению времени формирования адреса микрокоманды и в конечном счете к уменьшению быстродействия УА с хранимой в памяти логикой. В этом случае используются УА с «жесткой» логикой, обладающие большим быстродействием. В некоторых случаях в ЭВМ используются одновременно УА с хранимой в памяти логикой и УА с «жесткой» логикой.

7.3. УПРАВЛЯЮЩИЕ АВТОМАТЫ С «ЖЕСТКОЙ» ЛОГИКОЙ

Управляющие автоматы с «жесткой» логикой представляют собой логические схемы, вырабатывающие распределенные во времени управляющие функциональные сигналы. В отличие от управляющих устройств с хранимой в памяти логикой у этих автоматов можно изменить логику работы только путем переделок схем автомата.

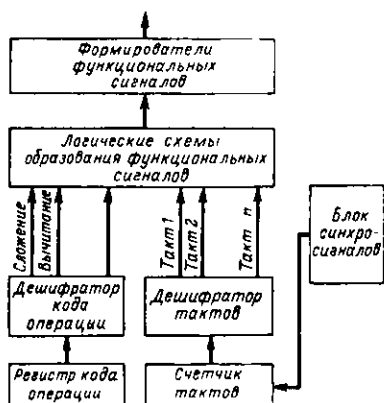


Рис. 7.6. Структура управляющего автомата с жесткой логикой

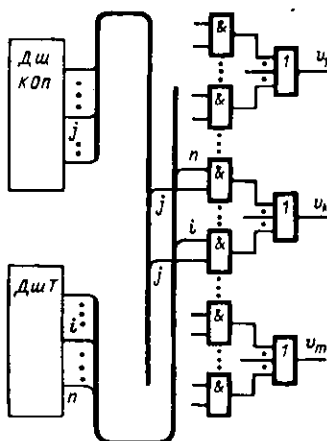


Рис. 7.7. Фрагмент схемы образования управляющих функциональных сигналов:

Дш КОп — дешифратор кода операции;
Дш Т — дешифратор тактов

Типичная структурная схема управляющего автомата с «жесткой» логикой показана на рис. 7.6. В состав схемы входят регистр кода операции, являющийся частью регистра команд, счетчик тактов, дешифратор тактов и дешифратор кода операции, а также логические схемы образования управляющих функциональных сигналов.

На счетчик тактов поступают сигналы от блока синхросигналов, и счетчик с каждым сигналом меняет свое состояние. Состояния счетчика представляют собой номера тактов, изменяющиеся от 1 до n . Дешифратор тактов формирует на i -м выходе единичный сигнал при i -м состоянии счетчика тактов, т. е. во время i -го такта.

Дешифратор кода операции вырабатывает единичный сигнал на j -м выходе, если выполняется j -я команда.

Логические схемы образования управляющих функциональных сигналов для каждой команды возбуждают формирователи функциональных сигналов для выполнения требуемых в данном такте микроопераций.

Принцип построения логических схем образования управляющих сигналов поясняется на рис. 7.7. Здесь показан фрагмент схемы, обеспечивающей выработку управляющего сигнала u_k в i -м и n -м тактах выполнения j -й команды.

В общем случае значения управляющих сигналов зависят еще и от оповещающих сигналов, отражающих ход вычисли-

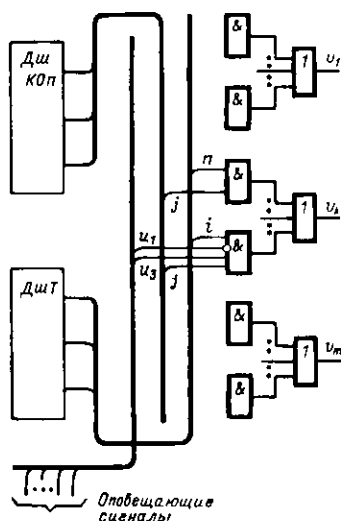


Рис. 7.8. Фрагмент схемы образования управляющих функциональных сигналов с реализацией зависимости от оповещающих сигналов

тельного процесса. Для реализации этих зависимостей элементы, представленные на рис. 7.7, берутся многовыходовыми и на них заводятся требуемые сигналы логических условий.

Если, например, необходимо, чтобы при исполнении j -й команды управляющий сигнал v_k появлялся в i -м такте только при значениях оповещающих

сигналов $u_1 = 0$ и $u_3 = 1$, а в n -м такте всегда, то схема, приведенная на рис. 7.7, изменится и примет вид, представленный на рис. 7.8.

Серьезным недостатком рассмотренных схем является одинаковое число тактов для всех команд. Это требует выравнивания числа тактов исполнения команд по наиболее «длинной» команде, что ведет к непроизводительным затратам времени. Чтобы устранить этот недостаток, схемы строят с использованием нескольких счетчиков тактов.

Схема формирования тактовых сигналов (датчик тактовых сигналов) может строиться на основе использования регистра сдвига, по которому двигается одна 1 (регистр с «бегущей единицей»), что не требует использования дешифратора.

Построение управляющих автоматов с «жесткой» логикой формализуется на основе интерпретации микропрограмм автоматами [12, 53].

Работу операционного блока можно описать микропрограммой, например, на языке микроопераций или в виде графа. По микропрограмме строится соответствующий управляющий автомат типа Мура или Мили.

Рассмотрим способ построения автомата Мура. Пусть микропрограмма работы некоторого операционного блока имеет вид, представленный на рис. 7.9, а. Блок ожидает возникновения $u_1 = 1$ и затем производит выработку управляющих функциональных сигналов $v_1 - v_6$ в определенной последовательности, зависящей от значений сигналов u_2 и u_3 .

Каждой микрокоманде, отдельно представленной на графе, ставится в соответствие отдельное состояние автомата. Состояния автомата отмечаются управляющими функциональными сигналами соответствующих микрокоманд.

Условия перехода от микрокоманды к микрокоманде представляются в виде конъюнкции входных сигналов, влияющих на переход. Каждая конъюнкция выписывается так, чтобы набор значений входных переменных, обращающих конъюнкцию в 1, соответствовал условию перехода. При безусловном переходе конъюнкция заменяется на константу 1.

Начало и конец микропрограммы отображаются начальным состоянием Q_0 автомата Мура, при этом предполагается, что автомат является циклическим, т. е. допускающим многократную повторяющуюся выработку последовательностей управляющих сигналов.

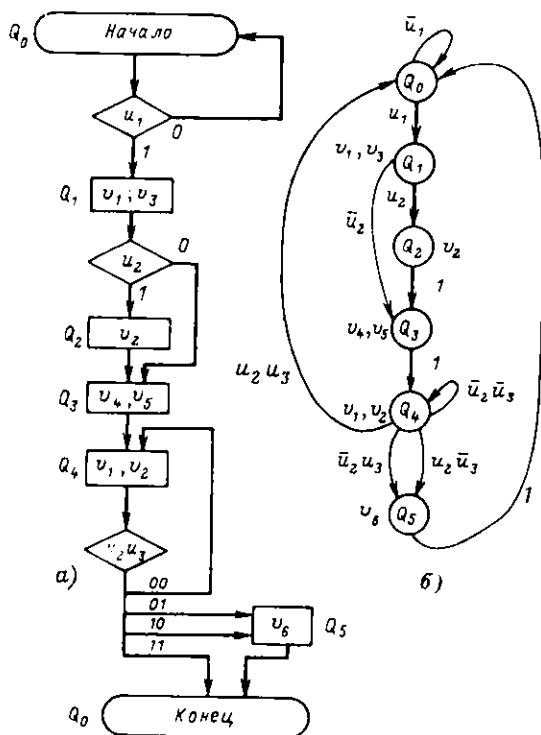


Рис. 7.9. Микропрограмма, представленная графом (а), и граф автомата Мура (б), интерпретирующий микропрограмму

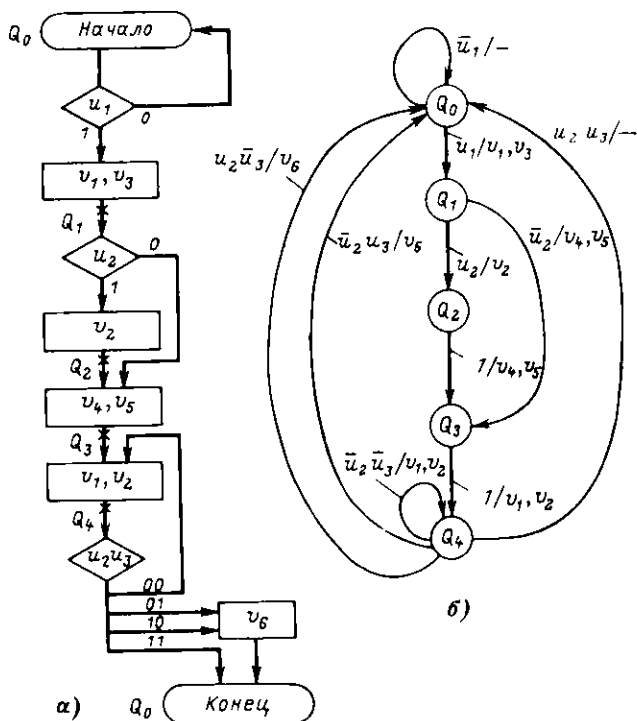


Рис. 7.10. Микропрограмма (а) и граф автомата Мили (б), интерпретирующей микропрограмму

Граф автомата Мура, построенного по описанным правилам для рассматриваемой микропрограммы, представлен на рис. 7.9, б.

Микропрограмму можно интерпретировать не только автоматом Мура, но и автоматом Мили, отличающимся тем, что значение его функции выхода в момент t зависит не только от состояния автомата, но и от набора значений входных сигналов.

Переход от микропрограммы к автомату Мили иллюстрируется на рис. 7.10, на котором показаны рассмотренный выше граф микропрограммы и граф автомата Мили, интерпретирующего ее. Начало и конец микропрограммы представляются начальным состоянием автомата Q_0 . Каждая дуга, выходящая из прямоугольника, представляющего собой микрокоманду, отмечается меткой x и символом состояния автомата. Исключение

составляют дуги, идущие к конечной или начальной вершине, которые не отмечаются. Если несколько дуг с меткой x входят в один блок графа микропрограммы, то все они отмечаются одинаковым символом состояния.

Условия перехода по микропрограмме от одной метки состояния к другой, соседней, задают функцию переходов автомата. Эти условия записываются в виде конъюнкций так же, как это делалось для автоматов Мура. Для каждого перехода, кроме того, фиксируется набор выходных переменных, принимающих при переходе единичное значение (задание функции выходов).

Автомат Мили, построенный по микропрограмме, имеет число состояний, как правило, меньшее числа состояний эквивалентного ему автомата Мура. С этой точки зрения использование автомата Мили предпочтительно. Однако применение автомата Мили в качестве управляющего автомата не всегда возможно. Объясняется это тем, что УА работает в контуре с операционным блоком ОБ (рис. 7.11). У автомата Мили переход в новое состояние осуществляется одновременно с формированием выходного сигнала. Поэтому, если ОБ вырабатывает оповещающие сигналы $u_1, \dots, u_n$ сразу при возникновении управляющих сигналов, а управляющий автомат является автоматом Мили, возможна следующая недопустимая ситуация: автомат Мили еще не сменил состояние, а на его входы пришли новые значения оповещающих сигналов, требующие выполнения иного перехода.

Для исключения возможных сбоев в работе управляющих автоматов ставятся специальные схемы задержки или, что то же самое, один из двух автоматов (управляющий либо операционный) выполняют в виде автомата Мура, который меняет выходной сигнал после смены состояния (перехода).

Рассмотрим, каким образом, имея описание управляющего автомата, можно построить схему автомата.

Зная число состояний автомата r , определяем число элементарных автоматов (двухступенчатых триггеров), требуемых для представления состояний автомата. Это число равно $k = \lceil \log_2 r \rceil$. Затем поставим в соответствие наборы значений состояний триггеров состояниям автомата. Пусть, например, нулевое состояние всех триггеров представляет собой Q_0 , набор состояний триггеров $00\dots 001$ соответствует Q_1 и т. д. Далее можно построить схему из k триггеров и дешифратора с r выходами. Единичный сигнал на одном из выходов дешифратора указывает соответствующее состояние автомата. Выбрав для определенности D -триггеры, получаем схему выделения состояний управляющего автомата, показанную на рис. 7.12.

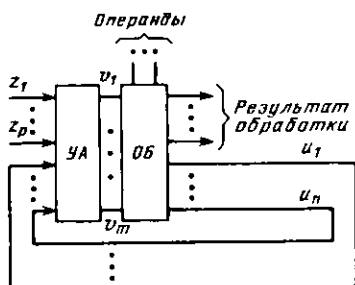


Рис. 7.11. Взаимодействие управляющего автомата и операционного блока

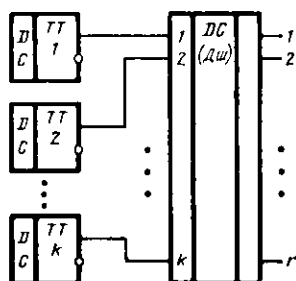


Рис. 7.12. Схема выделения состояний управляющего автомата

Для реализации схемы формирования управляющих сигналов следует выписать для каждого управляющего сигнала его булеву функцию. Для автомата Мура эти функции (функции выходов) имеют вид $v_i = \bigvee_j Q_{ij} = Q_{i1} \vee Q_{i2} \vee \dots \vee Q_{i_w}$, где Q_{ij} — состояние автомата, отмеченное v_i .

Так, для автомата, представленного на рис. 7.9, имеем следующие функции выходов:

$$v_1 = Q_1 \vee Q_4; \quad v_2 = Q_2 \vee Q_4;$$

$$v_3 = Q_1; \quad v_4 = Q_3;$$

$$v_5 = Q_3; \quad v_6 = Q_5.$$

По этим функциям легко строится схема формирования управляющих сигналов (рис. 7.13, а). Выдача выходных сигналов синхронизируется так, чтобы она происходила в то время, когда сигналы не меняются.

Для автомата Мили функции выхода имеют вид $v_i = \bigvee_j (Q_{ij} \vee K_j)$, где Q_{ij} — состояние, из которого выходит дуга, отмеченная выходным сигналом v_i , а $\bigvee_j K_j$ — дизъюнкция всех конъюнкций, записанных на дугах, выходящих из Q_{ij} и отмеченных выходным сигналом.

Для автомата, представленного на рис. 7.10, функции выходов имеют вид

$$v_1 = Q_0(u_1) \vee Q_3 \vee Q_4(\bar{u}_2\bar{u}_3);$$

$$v_2 = Q_1(u_2) \vee Q_3 \vee Q_4(\bar{u}_2\bar{u}_3); \quad v_3 = Q_0(u_1);$$

$$v_4 = Q_1(\bar{u}_2) \vee Q_2; \quad v_5 = Q_1(\bar{u}_2) \vee Q_2; \quad v_6 = Q_4(\bar{u}_2u_3 \vee u_2\bar{u}_3).$$

Преобразовав выражения с учетом общих частей формул, получим более простые зависимости:

$$\begin{aligned}v_1 &= v_3 \vee \varphi; \quad v_2 = Q_1(u_2) \vee \varphi; \quad v_3 = Q_0(u_1); \\v_4 &= v_5; \quad v_5 = Q_1(\bar{u}_2) \vee Q_2; \\v_6 &= Q_4(\bar{u}_2 u_3) \vee Q_4(u_2 \bar{u}_3); \quad \varphi = Q_3 \vee Q_4(\bar{u}_2 \bar{u}_3).\end{aligned}$$

По этим выражениям строим схему, представленную на рис. 7.13, б. Выходные сигналы здесь синхронизированы, как и в схеме на рис. 7.13, а.

Автомат содержит схему, реализующую смену состояний триггеров в соответствии с функцией переходов.

При смене состояний автомата $Q_i \rightarrow Q_j$ происходит смена состояний отдельных триггеров, так как они представляют собой состояния автомата. Следовательно, для каждого триггера

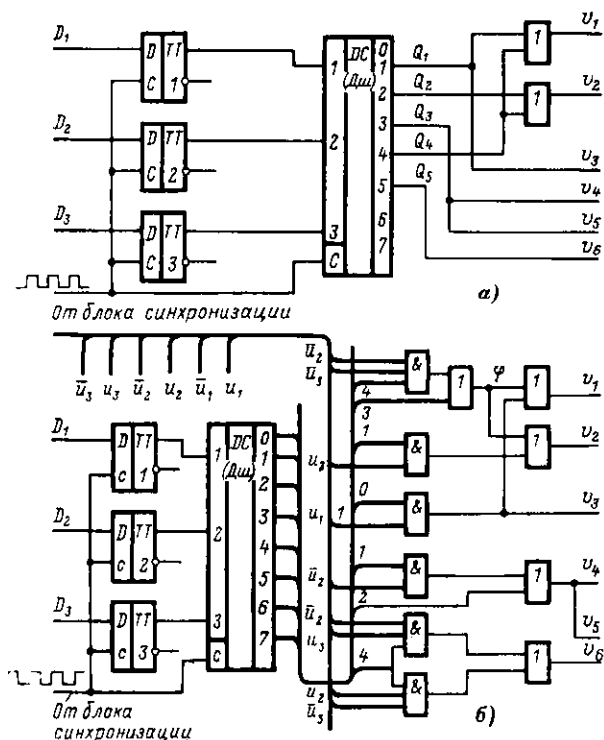


Рис. 7.13. Схема выработки управляющих сигналов для автоматов Мура (а) и Мили (б)

известно, какими должны быть изменения его состояний на всех переходах. Зная тип триггера, можно определить, какие сигналы должны быть поданы на его входы для получения требуемой смены состояний.

Для каждого перехода (указаны на графе автомата дугами) выписываются исходное состояние перехода, набор исходных состояний триггеров, набор состояний триггеров после перехода, а также конъюнкции входных сигналов, представляющие собой условие перехода.

По наборам исходных состояний и состояний после перехода выделяются отдельно для каждого триггера ситуации, при которых значения его входов должны быть единичными. Описав все случаи, требующие выработки единичного сигнала, булевыми функциями состояний автомата и входных сигналов, получаем так называемые функции возбуждения, по которым строятся схемы формирования сигналов на входах триггеров.

Поясним сказанное примером. Пусть состояния автомата, заданного графом на рис. 7.9, представлены следующими наборами состояний триггеров:

Таблица 7.1

Состояние автомата	Состояние триггеров			Состояние автомата	Состояние триггеров		
	Tz_1	Tz_2	Tz_3		Tz_1	Tz_2	Tz_3
Q_0	0	0	0	Q_3	0	1	1
Q_1	0	0	1	Q_4	1	0	0
Q_2	0	1	0	Q_5	1	0	1

Тогда функции переходов можно представить табл. 7.2, по которой строим булевы функции возбуждения D_1 , D_2 и D_3 :

$$D_1 = Q_3 \vee Q_4 \bar{u}_2 \bar{u}_3 \vee Q_4 \bar{u}_2 u_3 \vee Q_4 u_2 \bar{u}_3;$$

$$D_2 = Q_1 u_2 \vee Q_1 \bar{u}_2 \vee Q_2;$$

$$D_3 = Q_0 u_1 \vee Q_1 \bar{u}_2 \vee Q_2 \vee Q_4 \bar{u}_2 u_3 \vee Q_4 u_2 \bar{u}_3;$$

Упростив выражения, получим

$$D_1 = Q_3 \vee Q_4 \bar{u}_2 \vee Q_4 \bar{u}_3; \quad D_2 = Q_1 \vee Q_2;$$

$$D_3 = Q_0 u_1 \vee Q_1 \bar{u}_2 \vee Q_2 \vee Q_4 \bar{u}_2 u_3 \vee Q_4 u_2 \bar{u}_3.$$

По упрощенным выражениям строим схему (рис. 7.14).

Объединив схему формирования выходных сигналов (рис. 7.13, а) и схему формирования функций возбуждения (рис. 7.14), получим схему, реализующую автомат, представленный на рис. 7.9.

k/l триггеров. Пример такой схемы с $l = 2$ представлен на рис. 7.15.

Для полного выделения состояния в этом случае необходимо сигналы с выходов раздельных дешифраторов подать на элемент И. Это можно делать непосредственно в схемах формирования выходных сигналов и функций возбуждения, что дает в итоге более экономную общую схему.

При синтезе схем автоматов формальным методом приходится решать вопросы оптимального кодирования состояний, минимизации числа состояний автоматов, факторизации выражений, описывающих булевы функции возбуждения и выхода, и ряд других. Подробно методика формального синтеза схем управляющих автоматов описана в [12].

7.4. ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ МАТРИЦЫ В УПРАВЛЯЮЩИХ АВТОМАТАХ

Логическим элементом, весьма удобным для использования в схемах управляющих автоматов, является интегральная микросхема, называемая *программируемой логической матрицей* (ПЛМ). Схема ПЛМ имеет вид, представленный на рис. 7.16.

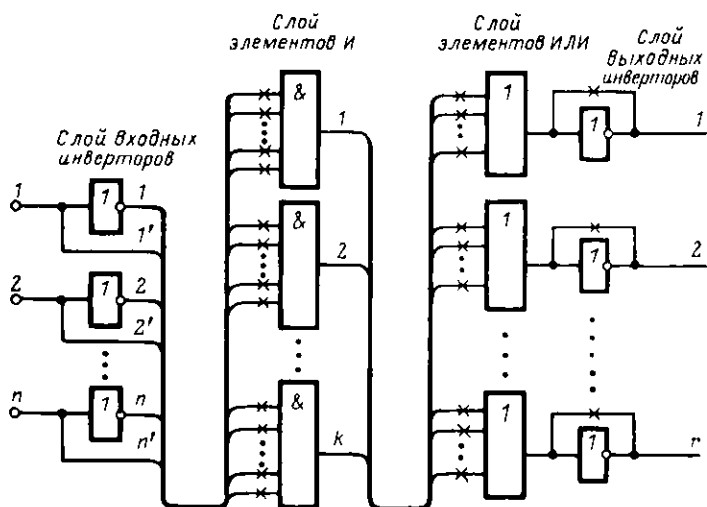


Рис. 7.16. Схема ПЛМ

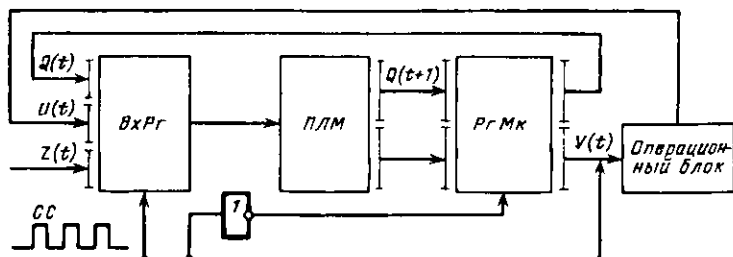


Рис. 7.17. Использование ПЛМ в управляющем автомате

При изготовлении ПЛМ образуется схема, допускающая множество вариантов обработки входных сигналов. Входные элементы позволяют иметь все входные переменные как в прямой, так и в инверсной форме. На входы любого элемента И поданы все входные переменные и их инверсии. К входам каждого элемента ИЛИ подключены выходы всех элементов И. Наконец, выходные элементы позволяют получить любую из выходных функций в прямом или инверсном виде.

Программирование матрицы состоит в устранении ненужных связей (устраняемые связи обозначены крестиками на рис. 7.16). Устранение лишних связей производится с помощью фотошаблонов или выжиганием.

Программируя ПЛМ, можно реализовывать нужные системы булевых функций. Это позволяет строить управляющие автоматы со структурой, представленной на рис. 7.17. Функционирует эта схема аналогично схеме на рис. 7.1. Применение СхФАМк здесь обычно не является необходимым из-за большого числа входов ПЛМ. Часть выходов ПЛМ используется для выработки управляющих функциональных сигналов, а часть — для выработки сигналов функций возбуждения. При этом часто оказывается необходимой совместная минимизация реализуемой ПЛМ системы булевых функций. Именно совместная минимизация дает возможность «заложить» в ПЛМ достаточно сложные функции.

Очень удобны для построения управляющих автоматов микросхемы, содержащие в одном корпусе ПЛМ с набором выходных триггеров.

ПРОЦЕССОРЫ И МИКРОПРОЦЕССОРЫ: ЭЛЕМЕНТЫ АРХИТЕКТУРЫ

8.1. ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

Выделим следующие элементы архитектуры ЭВМ:
типы обрабатываемых в ЭВМ данных и способы их представления в машине (изложены в гл. 2);
адресные структуры основной памяти;
способы адресации информации;
структуры команд;
регистровые структуры процессоров (программистские модели процессоров);
особенности системы команд ЭВМ.

В настоящей главе с единых позиций рассматриваются элементы архитектуры ЭВМ общего назначения (ЕС ЭВМ), малых ЭВМ (СМ ЭВМ), микро-ЭВМ и микропроцессоров.

8.2. АДРЕСНЫЕ СТРУКТУРЫ ОСНОВНЫХ ПАМЯТЕЙ

Основная (оперативная) память ЭВМ обычно является адресной. Это значит, что каждой хранимой в ОП единице информации (байту, слову) ставится в соответствие специальное число — адрес, определяющий место ее хранения в памяти.

В современных ЭВМ различных типов, как правило, минимальной адресуемой в памяти единицей информации является байт, т. е. 8-разрядный код (часто с дополнительным девятым контрольным разрядом — см. гл. 12). Более крупные единицы информации — основная машинная единица информации — слово и производные — полуслово, двойное слово и т. п. — образуются из целого числа байт. Обычно слово соответствует формату данных, наиболее часто встречающихся в данной машине в качестве операнда. Часто, но не обязательно, формат слова соответствует ширине выборки из основной памяти.

На рис. 8.1 представлены адресные структуры основных памяти в машинах разного типа и размещение в памяти различных единиц информации. В соответствии со сложившейся у разработчиков разных типов ЭВМ традицией в машинах общего назначения нумерация бит и байт в слове и других единицах информации производится слева направо, а в малых, микро-ЭВМ и микропроцессорах — справа налево. В обоих случаях адресом слова (и других единиц информации) служит

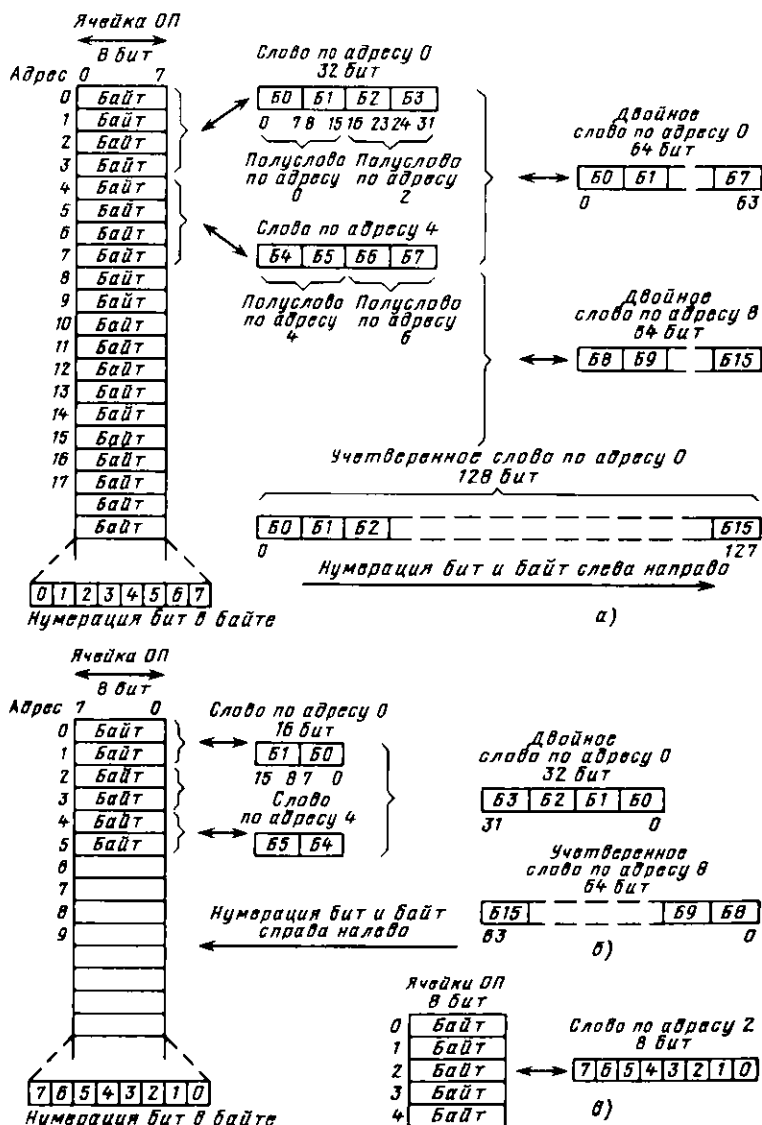


Рис. 8.1. Адресные структуры основных (оперативных) памяти в ЕС ЭВМ (а), в малых ЭВМ СМ-4 (б) и в микропроцессорах К580 (в)

адрес их байта с наименьшим номером, но в машинах общего назначения им является крайний левый (старший) байт в представлении числа (команды и т. п.), а в малых ЭВМ, микро-ЭВМ и микропроцессорах — крайний правый (младший) байт.

Число байт в адресуемой единице информации или задается в неявной форме кодом операции команды, или непосредственно указывается в коде команды.

Адрес в ЕС ЭВМ представляет собой целое 24-разрядное двоичное число, позволяющее адресовать до 16 777 216 байт. В ЕС ЭВМ используются следующие форматы информации фиксированной длины: 1) байт; 2) полуслово (два последовательно расположенных в памяти байта); 3) слово (4 байта); 4) двойное слово (8 байт); 5) учетверенное слово (16 байт). Кроме того, используются слова (или поля) переменной длины для представления десятичных чисел (в формате от 1 до 16 байт), логических кодов и алфавитно-цифровых данных (в формате от 1 до 256 байт).

Адрес слова переменной длины (адрес его левого байта) может быть любым в пределах адресной сетки машины, что обеспечивает размещение в памяти полей переменной длины без просветов.

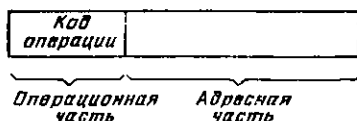
Для упрощения и убыстрения процедур выполнения команд в ряде ЭВМ на адресацию единиц информации фиксированной длины накладываются определенные ограничения (так называемые целочисленные границы). Например, в ЕС ЭВМ II очереди адреса полуслов, слов, двойных и учетверенных слов рекомендуется назначать кратными соответственно 2, 4, 8 и 16, при этом двоичные адреса указанных единиц информации содержат 0 соответственно в одном, двух, трех и четырех младших разрядах. Адрес команды может быть только четным.

8.3. ПРОБЛЕМА ВЫБОРА СТРУКТУРЫ И ФОРМАТА КОМАНД. КОДИРОВАНИЕ КОМАНД

Все возможные преобразования дискретной информации могут быть сведены к четырем основным видам: 1) передача информации в пространстве (например, из одного блока ЭВМ в другой); 2) передача информации во времени (хранение); 3) логические (поразрядные) операции; 4) арифметические операции. ЭВМ, являющаяся универсальным преобразователем дискретной информации, выполняет все указанные виды преобразований.

Обработка информации (решение задач) в ЭВМ осуществляется автоматически путем программного управления. Про-

Рис. 8.2. Общая структура команды



грамма представляет собой алгоритм обработки информации (например, решения математической задачи), записанный в виде последовательности команд, которые должны быть выполнены машиной для получения решения задачи.

Команда представляет собой код, определяющий операцию вычислительной машины и данные, участвующие в операции. Команда содержит также в явной или неявной форме информацию об адресе, по которому помещается результат операции, и об адресе следующей команды.

Процесс выполнения программы состоит из отдельных *машинных операций*. В данном случае под операцией понимаем преобразование информации, выполняемое машиной под воздействием одной команды. Содержанием машинной операции могут быть запоминание в памяти, передача, арифметическое или логическое преобразование машинных слов (операндов), а также некоторые вспомогательные процедуры.

По характеру выполняемых операций различают следующие основные группы команд: а) команды арифметических операций для чисел с фиксированной и плавающей запятой; б) команды десятичной арифметики; в) команды логических (поразрядных) операций (И, ИЛИ и др.); г) команды передачи кодов; д) команды операций ввода-вывода; е) команды управления порядком исполнения команд (команды передачи управления); ж) команды задания режима работы машины и др.

В команде, как правило, содержатся не сами операнды, а информация об адресах ячеек памяти или регистрах, в которых они находятся.

Код команды можно представить состоящим из нескольких частей или полей, имеющих определенное функциональное назначение при кодировании командной информации. Команда в общем случае состоит из операционной и адресных частей (рис. 8.2). В свою очередь, эти части, что особенно характерно для адресной части, могут состоять из нескольких полей.

Операционная часть содержит *код операции* (КОп), который задает вид операции (сложение, умножение, передача и т. д.). *Адресная часть команды* содержит информацию об адресах операндов и результата операции, а в некоторых случаях информацию об адресе следующей команды.

Структура команды определяется составом, назначением и расположением полей в команде. *Форматом команды* назы-

вают ее структуру с разметкой номеров разрядов (бит), определяющих границы отдельных полей команды, или с указанием числа бит в определенных полях.

Важной и сложной проблемой при проектировании ЭВМ является выбор структуры и формата команды, т. е. ее длины, назначения и длины отдельных ее полей. Естественно стремление разместить в команде в возможно более полной форме информацию о предписываемой командой операции. Однако в условиях, когда в современных ЭВМ значительно возросло число выполняемых различных операций и соответственно команд (система команд ЕС ЭВМ содержит около 200 команд) и значительно увеличилась емкость адресуемой основной памяти (в ЕС ЭВМ до 16 Мбайт), это приводит к недопустимо большой длине формата команды.

Действительно, число двоичных разрядов, отводимых под код операции, должно быть таким, чтобы можно было представить все выполняемые машинные операции. Если ЭВМ выполняет M различных операций, то число разрядов в коде операции

$$n_{к.о} \geq \log_2 M. \quad (8.1)$$

Например при $M = 200$, $n_{к.о} = 8$.

Если основная память содержит S адресуемых ячеек (байт), то для представления только одного адреса (например, адреса первого операнда) необходимо в команде иметь адресное поле для одного операнда с числом разрядов

$$n_A \geq \log_2 S. \quad (8.2)$$

Например, при $S = 16$ Мбайт $n_A = 24$.

Вместе с тем для упрощения аппаратуры и повышения быстродействия ЭВМ длина формата команды должна быть согласована с выбираемой из требований точности вычислений длиной обрабатываемых машиной слов (операндов), составляющей для большинства применений 16–32 бит, с тем чтобы для операндов и команд можно было эффективно использовать одни и те же память и аппаратурные средства обработки информации. Формат команды должен быть по возможности короче, укладываться в машинное слово или полу-слово, а для ЭВМ с коротким словом (8–16 бит) должен быть малократным машинному слову. Решение проблемы выбора формата команды значительно усложняется в малых и микро-ЭВМ и микропроцессорах, работающих с коротким словом.

Отмечавшиеся ранее характерные для процесса развития ЭВМ рост числа выполняемых машиной операций и увеличе-

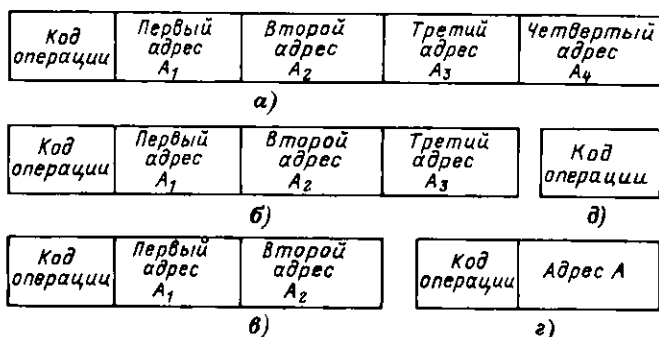


Рис. 8.3. Структуры команд:

а — четырехдресная; *б* — трехдресная; *в* — двухдресная; *г* — однодресная; *д* — безадресная

ние емкости основной памяти, а особенно создание малых и микро-ЭВМ с коротким словом потребовали разработки методов укорочения формата команды. При решении этой проблемы существенно видоизменилась структура команды, получили развитие различные способы адресации информации.

Проследим изменения классических структур команд.

Чтобы команда содержала в явном виде всю необходимую информацию о задаваемой операции, она должна, как это показано на рис. 8.3, *а*, содержать поле кода операции и четыре адреса для указания ячеек памяти, содержащих два операнда, участвующих в операции, ячейки, в которую помещается результат операции, и ячейки, содержащей следующую команду. Такой порядок выборки команд называется *принудительным*. Он использовался в некоторых первых моделях ЭВМ. Для рассмотренного выше примера с $M = 200$ и $S = 16$ Мбайт длина четырехдресной команды составила бы 80 бит. Такой формат команды окажется трудно реализуемым и неэффективным. Четырехдресные команды в настоящее время не применяются.

Можно установить, как это принято для большинства машин, что после выполнения данной команды, расположенной по адресу K (и занимающей L ячеек), выполняется команда из $(K + L)$ -й ячейки. Такой порядок выборки команд называется *естественным*. Он нарушается только специальными командами. В таком случае отпадает необходимость указывать в команде в явной форме адрес следующей команды. В трехдресной команде (рис. 8.3, *б*) первый и второй адреса указывают ячейки памяти, в которых расположены операнды, а третий определяет ячейку, в которую помещается результат операции.

Операция, описываемая трехадресной командой, может быть символически представлена в виде

$$ОП [A_3] := ОП [A_1] * ОП [A_2],$$

где знак * заменяет символ операции (+, — и т. п.).

Можно условиться, что результат операции всегда помещается на месте одного из операндов, например первого. Получим двухадресную команду (рис. 8.3, в), задающую операцию

$$ОП [A_1] := ОП [A_1] * ОП [A_2],$$

т. е. для результата используется подразумеваемый адрес.

В одноадресной команде (рис. 8.3, г) подразумеваемые адреса имеют уже и результат операции, и один из операндов. Один из операндов указывается адресом в команде, в качестве второго используется содержимое внутреннего регистра процессора, называемого в этом случае регистром результата или аккумулятором Акк. Результат операции записывается в тот же регистр:

$$Акк := Акк * ОП [A].$$

Наконец, в некоторых случаях возможно использование безадресных команд (рис. 8.3, д), когда подразумеваются адреса обоих операндов и результата операции, например при работе со стековой памятью.

С точки зрения программиста наиболее естественны и удобны трехадресные команды. Однако из-за необходимости иметь большее число разрядов для представления адресов и кода операции длина трехадресной команды становится недопустимо большой, и ее не удастся разместить в машинном слове. Следует отметить, что очень часто в качестве операндов используются результаты предыдущих операций, хранимые в триггерных регистрах машины. В этом случае выполняемая операция приобретает характер одно- или двухадресной, при этом трехадресный формат используется неэффективно. По указанным причинам в современных ЭВМ применяют, как правило, двух- и одноадресные команды и их модификации.

Способ расширения кодов операций. В машинах с коротким словом практически невозможно в одном формате команды, т. е. при фиксированном назначении ее полей, кодировать большое число различных операций и одновременно иметь гибкую форму адресации операндов. Это противоречие в машинах с коротким словом преодолевается расширением кодов операций в команде. Для задания небольшой группы основных операций (арифметических и др.) используется короткий код операции, а получаемая при этом сравнительно большая адресная

часть команды позволяет реализовать гибкую, например двух-адресную с многими модификациями, адресацию. Для задания других операций используются более длинные (*расширяемые*) коды операций, при этом сокращаемая адресная часть оставляет возможность лишь для более простой, например одноадресной, адресации операндов. В пределе расширяемый код операций занимает весь формат команды (безадресная команда).

Обычно в ЭВМ используется несколько форматов команд разной длины.

Приведенные на рис. 8.3 форматы команд достаточно схематичны. В действительности адресные поля команд большей частью содержат не сами адреса, а только информацию, позволяющую определить действительные (исполнительные) адреса операндов в соответствии с используемыми в командах способами адресации.

8.4. СПОСОБЫ АДРЕСАЦИИ

Следует различать понятия *адресный код* в команде A_k и исполнительный адрес A_d . Адресный код — это информация об адресе операнда, содержащаяся в команде. Исполнительный адрес — это номер ячейки памяти, к которой производится фактическое обращение. В современных ЭВМ адресный код, как правило, не совпадает с исполнительным адресом.

Выбор способов адресации, формирования исполнительного адреса и преобразования адресов является одним из важнейших вопросов разработки ЭВМ. Рассмотрим способы адресации, используемые в современных ЭВМ.

Подразумеваемый операнд. В команде не содержится явных указаний об адресе операнда; операнд подразумевается и фактически задается кодом операции команды. Данный метод используется не часто, однако имеется несколько важных случаев его применения. В качестве примера можно привести команды подсчета, в которых к некоторому числу (содержимому счетчика) прибавляется фиксированное приращение, часто единица младшего разряда. Один из операндов — число в счетчике — обычно адресуется явным методом, второй операнд — приращение — не адресуется, в памяти машины не содержится и является подразумеваемым.

Подразумеваемый адрес. В команде не содержится явных указаний об адресе участвующего в операции операнда или адреса, по которому помещается результат операции, но этот адрес подразумевается. Например, команда может содержать адреса обоих операндов, участвующих в операции, при этом

подразумевается, что результат операции помещается по адресу одного из операндов, или команда указывает только адрес одного операнда, а адрес второго, которым является содержимое специального регистра (называемого регистром результата или аккумулятором), подразумевается.

Непосредственная адресация. В команде содержится не адрес операнда, а непосредственно сам операнд. При непосредственной адресации не требуется обращения к памяти для выборки операнда и ячейки памяти для его хранения. Это способствует уменьшению времени выполнения программы и занимаемого ею объема памяти. Непосредственная адресация удобна для хранения различного рода констант; однако следует иметь в виду, что при этом способе адресации длина операнда короче кода команды, поскольку часть разрядов команды занята под код операции.

Прямая адресация. Исполнительный адрес совпадает с адресной частью команды. Этот способ адресации был общепринятым в первых вычислительных машинах и продолжает применяться в настоящее время в комбинации с другими способами.

Относительная адресация или базирование. Исполнительный адрес определяется суммой адресного кода команды A_k и некоторого числа A_6 , называемого *базовым адресом*:

$$A_n = A_6 + A_k.$$

Для хранения *базовых адресов* в машине могут быть предусмотрены регистры или специально выделенные для этой цели ячейки памяти (*базовые регистры*). В команде выделяется поле B для указания номера базового регистра.

Относительная адресация позволяет при меньшей длине адресного кода команды обеспечить доступ к любой ячейке памяти. Для этого число разрядов в базовом адресе выбирают таким, чтобы можно было адресовать любую ячейку ОП, а адресный код A_k самой команды используют для представления лишь сравнительно короткого «смещения» (обозначают буквой D). Смещение D определяет положение операнда относительно начала массива, задаваемого базовым адресом A_6 . Рисунок 8.4 поясняет процесс формирования исполнительного адреса.

Большей частью исполнительный адрес при базировании образуется с помощью сумматора согласно выражению (рис. 8.4, б)

$$A_n = \begin{cases} (B) + D, & \text{если } B \neq 0; \\ D, & \text{если } B = 0, \end{cases}$$

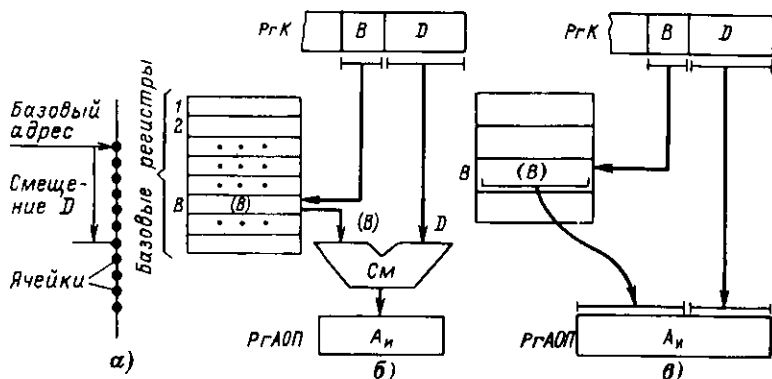


Рис. 8.4. Базирование (относительная адресация):

а — образование адреса элемента одномерного массива; б — формирование исполнительного адреса суммированием; в — формирование исполнительного адреса совмещением

где B и D — коды (числа), стоящие в соответствующих полях команды; (B) — содержимое регистра с номером B . При $B=0$ относительная адресация блокируется.

Более подробно обращение к ОП при относительной адресации можно представить в следующем виде:

если $P_2K[B] = 0$ то $P_2AOP := P_2K[D]$

иначе $P_2AOP := P_2B + P_2K[D]$;

Счит: $P_2IOП := ОП[P_2AOP]$;

Суммирование при образовании $A_{и}$ связано с потерей времени. Поэтому применяют также формирование исполнительного адреса методом совмещения (рис. 8.4, в). В этом случае базовый адрес содержит старшие, а смещение — младшие разряды исполнительного адреса, которые объединяются в P_2AOP согласно операции конкатенации слова:

$$P_2AOP := P_2B / P_2K[D];$$

Однако при совмещении базовый адрес может задавать не любую ячейку, а только те, адреса которых содержат 0 в младших разрядах, соответствующих смещению.

Относительная адресация обеспечивает так называемую перемещаемость программ, т. е. возможность передвижения программ в памяти без изменений внутри самой программы.

Относительная адресация облегчает распределение памяти при составлении несколькими программистами сложных программ. Каждый программист может располагать свою часть программы в памяти, начиная с нулевого адреса. При компо-

новке программы достаточно только задать соответствующие значения базовых адресов.

Укороченная адресация. Для уменьшения длины кода команды часто применяется так называемая укороченная адресация. Суть ее сводится к тому, что в команде задаются только младшие разряды адресов, старшие разряды при этом подразумеваются нулевыми. Такая адресация позволяет использовать только небольшую группу фиксированных ячеек с начальными (короткими) адресами и поэтому может применяться лишь совместно с другими способами адресации.

Регистровая адресация есть частный случай укороченной, когда в качестве фиксированных ячеек с короткими адресами используются регистры (ячейки сверхоперативной или местной памяти) процессора. Например, если таких регистров 16, то для адреса достаточно четырех двоичных разрядов. Регистровая адресация наряду с сокращением длины адресов операндов позволяет увеличить скорость выполнения операций, так как уменьшается число обращений к ОП.

Косвенная адресация. Адресный код команды указывает адрес ячейки памяти, в которой находится адрес операнда или команды. Таким образом, косвенная адресация может быть иначе определена как «адресация адреса».

На косвенную адресацию указывает код операции команды, а в некоторых ЭВМ в команде отводится специальный разряд (указатель адресации — УА), и цифра 0 или 1 в нем указывает, является адресная часть команды прямым адресом или косвенным. Обращение к ОП за операндом при косвенной адресации представляет собой следующую процедуру:

$P_2 AOP : = P_2 K[A];$

Счит: $P_2 IOП : = ОП[P_2 AOP];$

если $УА=0$ то идти к M иначе $P_2 AOP = P_2 IOП;$

Счит: $P_2 IOП : = ОП[P_2 AOP];$

$M : P_2 AЛУ : = P_2 IOП;$

В некоторых ЭВМ используется многоступенчатая косвенная адресация. В этом случае ячейки памяти содержат также разряд — указатель косвенной адресации (УА). Если этот разряд указывает на продолжение косвенной адресации, то машина последовательно выбирает из памяти адреса до тех пор, пока не будет найдена ячейка, в которой разряд-указатель определит прямую адресацию. Адрес из этой последней ячейки и является искомым исполнительным адресом.

Косвенная адресация широко используется в малых и микро-ЭВМ, имеющих короткое машинное слово, для преодоления ограничений короткого формата команды.

Рассмотрим широко применяемое в микропроцессорах, малых и микро-ЭВМ *совместное использование укороченной (регистровой) и косвенной адресаций* (рис. 8.5). Пусть необходимо передать число 4527 из *Р25* в ячейку ОП 1765. Длина адресных полей команды достаточна лишь для указания коротких номеров регистров, она не позволяет задать в команде полный адрес ячейки памяти. Поэтому операнд 4527 указывается регистровой прямой адресацией ($УА = 0$), а для задания адреса 1765 приходится воспользоваться регистровой косвенной адресацией ($УА = 1$), при которой в команде указывается номер регистра (в рассматриваемом примере *Р23*), предварительно загруженного полным адресом ячейки, в которую производится передача.

Автоинкрементная и автодекрементная адресации. Поскольку регистровая косвенная адресация требует предварительной загрузки регистра из ОП косвенным адресом, что связано с потерей времени, такой тип адресации особенно эффективен при обработке массива данных, если имеется механизм автоматического приращения или уменьшения содержимого регистра при каждом обращении к нему, называемый соответственно автоинкрементной и автодекрементной адресацией. В этом случае достаточно один раз загрузить в регистр адрес первого обрабатываемого элемента массива, а затем при каждом обращении к регистру в нем в результате инкрементной (декрементной) процедуры формируется адрес следующего элемента массива.

При *автоинкрементной адресации* по содержимому регистра сначала содержимое регистра используется как адрес операнда, а затем получает приращение, равное числу байт в элементе массива. При *автодекрементной адресации* сначала содержимое указанного в команде регистра уменьшается на число, равное числу байт в элементе массива, а затем используется как адрес операнда.

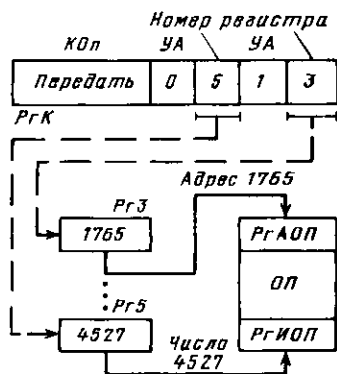


Рис. 8.5. Совместное использование регистровой прямой и регистровой косвенной адресаций для преодоления ограничений короткого слова команды

Автоинкрементная и автодекрементная адресации могут рассматриваться как упрощенный вариант индексации — весьма важного механизма преобразования адресных частей команд (см. § 8.6) и организации вычислительных циклов, поэтому их часто называют *автоиндексацией*.

Адресация слов переменной длины. Эффективность вычислительных систем, предназначенных для обработки данных (экономических, плановых и др.), повышается, если имеется возможность выполнять операции со словами переменной длины. В этом случае в машине должна быть предусмотрена адресация слов переменной длины, которая обычно реализуется путем указания в команде местоположения в памяти начала слова и его длины.

Обычно в ЭВМ одновременно используется несколько типов адресации. Тип адресации указывается либо неявно кодом операции, либо в явной форме в специальном поле адресной части команды.

8.5. СТЕКОВАЯ АДРЕСАЦИЯ

Стековая память (см. гл. 4), реализующая безадресное задание операндов, является эффективным элементом современной архитектуры ЭВМ, особенно широко используемым в микропроцессорах, малых и микро-ЭВМ. Учитывая своеобразие стековой адресации, ее рассмотрение выделено в отдельный параграф.

Стек представляет собой группу последовательно пронумерованных регистров (*аппаратурный стек*) или ячеек памяти, снабженных указателем стека (обычно регистром) (УС), в котором автоматически при записи и считывании устанавливается номер (адрес) последней занятой ячейки стека (*вершины стека*). При операции записи заносимое в стек слово помещается в следующую по порядку свободную ячейку стека, а при считывании из стека извлекается последнее поступившее в него слово. Таким образом, в стеке реализуется правило «*последний пришел — первый ушел*».

Указанное правило при обращении к стеку реализуется автоматически, и поэтому при операциях со стеком возможно безадресное задание операнда — команда не содержит адреса ячейки стека, но содержит адрес (или он подразумевается) ячейки памяти или регистра, откуда слово передается в стек или куда загружается из стека.

Механизм стековой адресации поясняется на рис. 8.6. При выполнении команды передачи в стек слова из регистра или

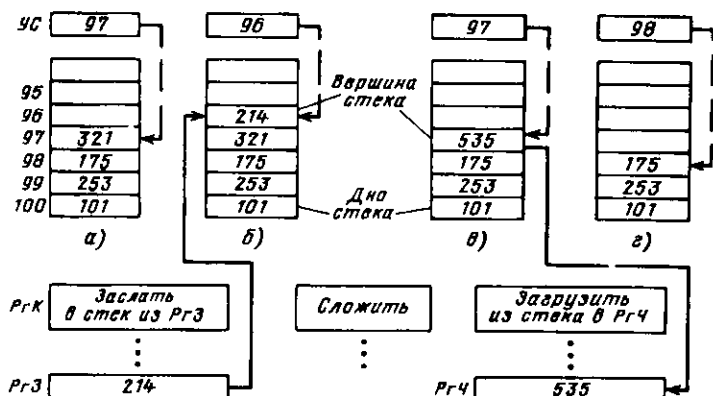


Рис. 8.6. Стековая адресация в «перевернутом стеке»:

а — исходное состояние стека; б — стек после выполнения команды «Засылать в стек из Рг3»; в — стек после выполнения команды «Сложить»; г — стек после выполнения команды «загрузить из стека в Рг4»

ячейки ОП сначала указатель стека увеличивается на 1 (в *перевернутом стеке* уменьшается на 1), а затем слово помещается в ячейку стека, указываемую УС. При команде загрузки из стека регистра или ячейки памяти сначала слово извлекается из вершины стека, а затем указатель стека уменьшается на 1 (в *перевернутом стеке* увеличивается на 1)¹. Как это ни кажется на первый взгляд удивительным, но при соответствующем расположении операндов в стеке можно вычислять выражения полностью безадресными командами, указывающими только вид операции. Такая команда извлекает из стека в соответствии с кодом операции один или два операнда, выполняет над ними предписанную операцию и заносит результат в стек.

Вычисления с использованием стековой памяти удобно описывать и программировать с помощью польской инверсной (бесскобочной) записи арифметических выражений ПОЛИЗ. Эта запись производится по следующему правилу: читаем арифметическое выражение слева направо и последовательно друг за другом выписываем встречающиеся операнды. Как только окажется, что все операнды некоторой операции выписаны, записываем знак этой операции и продолжаем выписывать операнды. Если операция имеет операндом результат не-

¹ В случае использования памяти с побайтной адресацией при занесении слова в стек и извлечении слова из стека содержимое УС изменяется на величину L , где L — количество байт в слове.

Адрес k
Адрес l
+
Адрес m
—
Адрес p
Адрес s
—
×

Рис. 8.7. Программа вычисления выражения (8.4) с использованием стековой памяти

которой предыдущей операции и знак последней выписан, то считаем этот операнд выписанным.

Например, выражение

$$(k + l - m)(p - s) \quad (8.3)$$

в ПОЛИЗ имеет вид

$$kl + m - ps - \times. \quad (8.4)$$

Выражение в ПОЛИЗ не содержит скобок, но порядок действий определяет однозначно.

При использовании стековой памяти последовательность символов в выражении ПОЛИЗ, например (8.4), может рассматриваться как программа вычисления исходного арифметического выражения (рис. 8.7), если под буквами понимать команды засылки, содержащие только адреса в ОП соответствующих операндов, засылаемых в стек, а под знаками операций — *безадресные команды*, содержащие только коды операций. Команда второго типа инициирует извлечение из стека двух (или одного) слов, выполнение над ними указанной в команде операции и засылку результата в вершину стека.

Безадресные команды на основе стековой адресации предельно сокращают формат команд, экономят память и способствуют повышению производительности ЭВМ.

Однако при такой структуре команд возникают осложнения с построением команд передачи управления и работы с периферийными устройствами.

В современной архитектуре процессоров и микропроцессоров стек и стековая адресация широко используются при организации переходов к подпрограммам и возврате от них, а также в системах прерывания (см. гл. 9). Весьма широко стеки и безадресные команды используются в вычислительном комплексе «Эльбрус» [17].

8.6. КОМАНДЫ, ПРОЦЕДУРЫ И МИКРОПРОГРАММЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ В ПРОГРАММАХ

Рассмотрим команды, управляющие порядком исполнения команд.

При естественном порядке после выполнения очередной команды выбирается команда, расположенная в следующей по порядку ячейке памяти. Обычно адрес команд хранится в специальном регистре, называемом счетчиком команды (СЧК), со-

держимое которого после выполнения каждой команды автоматически увеличивается на 1, а если память имеет побайтную адресацию, то на столько, сколько байт содержит текущая команда (*приращение адреса команды L_k*).

Выборка очередной и формирование адреса следующей команды (АСК), часто называемого «продвинутым адресом», происходит (при естественном порядке) следующим образом (рис. 8.8):

$$P_2 A O П : = C ч К ;$$

$$Счит. : P_2 И О П : = О П [P_2 A O П] ;$$

$$P_2 К : = P_2 И О П ;$$

(Выполнение текущей команды)

$$A C K_{E H} : C ч К : = C ч К + L_k ;$$

Естественный порядок выполнения команд может быть нарушен: 1) командами перехода (командами передачи управления); 2) командами замещения («Выполнение»); 3) смесой состояния программы; 4) запросами прерывания программы (см. гл. 9).

Команды перехода (передачи управления). Известны многочисленные варианты команд перехода, однако общий принцип состоит в том, что адресная часть команды перехода непосредственно или после суммирования с содержимым базового регистра передается в счетчик команд, и, следовательно, после данной команды может быть выполнена команда из произвольной ячейки памяти, номер которой определяется адресной частью команды перехода.

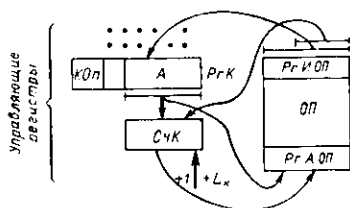


Рис. 8.8. Формирование адреса следующей команды при естественном порядке выборки команд и при его нарушении командами перехода и командой «Выполнение»

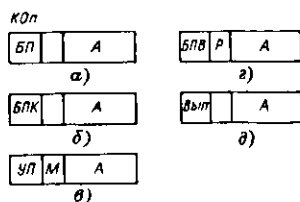


Рис. 8.9. Упрощенные структуры команд перехода безусловного (а), безусловного по косвенному адресу (б), условного (в), безусловного с возвратом (г) и команды «Выполнение» (д)

Для упрощения изложения материала рассмотрим процедуры выполнения команд переходов без относительной адресации.

Различают два основных вида команд перехода: *безусловный* переход (БП) и *условный* (УП). Упрощенные структуры этих команд показаны на рис. 8.9.

Команда БП (рис. 8.9, а) совершает переход всегда независимо от каких-либо условий. Обыкновенно следующей за командой БП выполняется команда, адрес которой указан в адресной части команды БП, т. е. (см. рис. 8.8)

$$АСК_{БП} : СчК := P_2K[A];$$

Другим вариантом команды БП является *безусловный переход по косвенному адресу* (БПК) (рис. 8.9, б). На косвенную адресацию указывает код операции команды или специальное поле в формате команды, определяющее тип адресации. Управление передается команде, расположенной в ячейке, адрес которой указан в адресной части команды БПК. В этом случае выполняется следующая процедура (рис. 8.8):

$$P_2АОП : = P_2K[A];$$

$$Счит : P_2ИОП : = ОП[P_2АОП];$$

$$АСК_{БПК} : СчК := P_2ИОП[A];$$

При *условном переходе* адрес следующей команды зависит от выполнения некоторого условия. Обычно, если условие выполняется, происходит переход к команде по адресу, указанному в адресной части команды УП. В противном случае выбирается следующая по порядку команда, адрес которой определяется, как обычно, содержимым *СчК*, увеличенным на *приращение адреса команды* L_k . Так же как и при БП, в команде УП могут использоваться относительная и косвенная адресации.

Условия перехода (например, результат предыдущей команды равен, больше или меньше 0, переполнение разрядной сетки и др.) задаются кодом операции УП или в виде отдельного поля «маска условия» *М* в команде УП (рис. 8.9, в). В этом случае следующая команда определяется по правилу (рис. 8.8)

$$\text{если } Ус[M] = 1 \text{ то } АСК_{УП} : СчК := P_2K[A] \text{ иначе}$$

$$АСК_{УП} : СчК := СчК + L_k;$$

Здесь $Ус[M]$ — логическая функция (условие), заданная (заданное) маской *М*. Если условие перехода задается кодом операции команды $Ус[КОН]$, то в приведенном выражении $Ус[M]$ надо заменить на $Ус[КОН]$.

Таблица 8.1

Результат операции	Код <i>ПР</i>		Маска условия (ЕС ЭВМ)			
	<i>ПР</i> [0]	<i>ПР</i> [1]	<i>М</i> [0]	<i>М</i> [1]	<i>М</i> [2]	<i>М</i> [3]
Равен 0	0	0	1	0	0	0
Меньше 0	0	1	0	1	0	0
Больше 0	1	0	0	0	1	0
Переполнение	1	1	0	0	0	1

Обычно выполнение машинной команды сопровождается выработкой кода признака результата *ПР*, формируемого в специальном регистре *РзПР*. Команда УП анализирует сформированный предыдущей командой *ПР*. Смысл кодов *ПР* может быть установлен различным для разных операций. Например, в ЕС ЭВМ двухразрядный признак результата *ПР* при выполнении арифметических операций принимает значения, приведенные в табл. 8.1 (*ПР*[0] и *ПР*[1] – нулевой и первый разряды *ПР*).

С помощью двухразрядной маски можно задать в качестве условия УП любое значение *ПР*.

В машинах ЕС ЭВМ используется 4-разрядная маска, и это дает возможность задавать в качестве условия перехода выполнение не только любого из *ПР*, но и их дизъюнкции (если в маске *М* несколько 1).

В этом случае имеем

$$U_c[M] = M[0] \cdot \overline{ПР[0]} \cdot ПР[1] \vee M[1] \cdot ПР[0] \cdot \overline{ПР[1]} \vee \\ \vee M[2] \cdot ПР[0] \cdot \overline{ПР[1]} \vee M[3] \cdot ПР[0] \cdot ПР[1].$$

Отметим, что при $M=1111$ команда УП выполняет безусловный переход, а при $M=0000$ (пустая команда) действует естественный порядок выборки команд. Таким образом исключается необходимость в отдельной команде безусловного перехода.

Команды условных переходов позволяют реализовать программы с разветвлениями в зависимости от промежуточных результатов вычислений или состояния машины.

Важным случаем передачи управления являются *безусловные переходы к подпрограммам*. Их особенность состоит в том, что помимо перехода они должны обеспечить по окончании подпрограммы возврат к исходной программе, к той точке ее, откуда был совершен переход. Обычно для переходов к подпрограммам используется специальная команда «Безус-

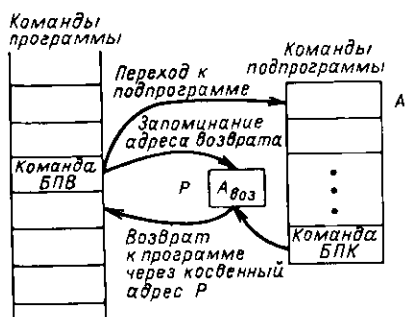


Рис. 8.10. Обращение к подпрограмме с помощью команды безусловного перехода с возвратом

ловный переход с возвратом» (БПВ). По этой команде (рис. 8.10) сначала адрес возврата $A_{\text{воз}}$, т. е. содержимое $CчК$ (увеличенное на «приращение адреса команды»¹ L_k), запоминается по

адресу P , указанному в команде БПВ², затем в счетчик команд заносится содержимое поля A команды БПВ, т. е. адрес A начала подпрограммы. В конце подпрограммы размещается команда возврата, которая представляет собой команду БПК, указывающую путем косвенной адресации ячейки, в которой находится адрес $A_{\text{воз}}$.

Формально всю эту процедуру после приема в P_2K команды БПВ можно представить в следующем виде (см. рис. 8.8 и 8.10):

$CчК := CчК + L_k$ (образование адреса возврата $A_{\text{воз}}$);

$P_2ИОП := CчК$;

$P_2АОП := P_2K[P]$;

Запись: $ОП[P_2АОП] := P_2ИОП$ (запоминание адреса возврата $A_{\text{воз}}$ в ячейке P);

$CчК := P_2K[A]$ (передача в $CчК$ адреса начала подпрограммы);

Выполнение подпрограммы

Считывание и передача в P_2K заключающей подпрограмму команды БПК, содержащей в поле A косвенный адрес P

$P_2АОП := P_2K[A]$;

Счит: $P_2ИОП := ОП[P_2АОП]$ (извлечение адреса возврата $A_{\text{воз}}$);

$АСК_{\text{БПВ}}: CчК := P_2ИОП[A]$ (возврат к основной программе — команде в ячейке $A_{\text{воз}}$);

¹ Для ЭВМ с пословной адресацией и длиной команды, равной слову, $L_k = 1$.

² Обычно это один из общих регистров.

Операция замещения, реализуемая командой «Выполнение» (*Вып*), состоит в том, что вместо очередной команды, соответствующей естественному порядку выборки команд, исполняется «замещающая» команда, указываемая адресной частью команды *Вып*, а затем, если только замещающая команда не оказалась командой перехода, восстанавливается приостановленный на время выполнения команды *Вып* естественный порядок выборки команд. Команда *Вып* должна сохранять неизменным содержимое *СчК*. Поэтому адрес замещающей команды берется не из *СчК*, а из *РзК* (см. рис. 8.8). Соответствующую процедуру после приема в *РзК* команды *Вып* можно представить в виде

$$P_2AOП := P_2K[A];$$

$$Счит := P_2ИОП := ОП[P_2AOП];$$

$$P_2K := P_2ИОП \langle \text{выборка замещающей команды} \rangle;$$

.....
Исполнение замещающей команды

$$СчК := СчК + L_k \langle \text{восстановление естественного порядка выборки команд} \rangle;$$

8.7. ИНДЕКСАЦИЯ

Характерным для реализуемых на ЭВМ методов решения математических задач и обработки данных является цикличность вычислительных процессов, при которой повторяются одни и те же процедуры, но над различными операндами (элементами информационных массивов), расположенными упорядоченно в памяти. Поскольку операнды, обрабатываемые при повторениях цикла, имеют разные адреса, можно для каждого повторения составить свою последовательность команд, отличающихся адресными частями. На рис. 8.11, а в качестве простого примера показан одномерный информационный массив, в котором выделены кружками подлежащие обработке операнды $a_0, a_1, \dots, a_{k-1}$, расположенные последовательно со сдвигом на q ячеек. Программа состоит из групп команд (рис. 8.11, б), повторяющих один и тот же цикл вычислений, причем каждая последующая отличается от предыдущей только приращением на q адреса операнда a_i . Однако при таком подходе программа решения задачи оказывается слишком длинной, а ее составление чрезмерно трудоемким.

Программирование вычислительных циклов существенно упростится, если после каждого выполнения цикла обеспечить автоматическое изменение в соответствующих командах их адресных частей согласно расположению в ОП обрабаты-

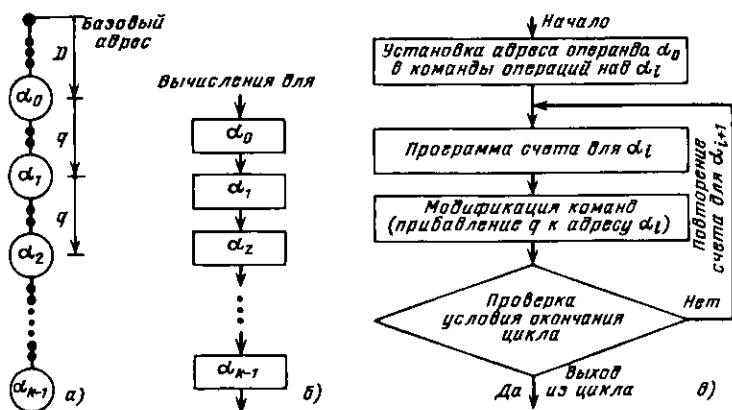


Рис. 8.11. Вычислительный цикл и автоматическая модификация адресов команд

ваемых операндов. Такой процесс называется *модификацией команд*, точнее, адресных частей команд. Модификация команд основана на возможности выполнения над кодами команд или их частями арифметических и логических операций. В рассматриваемом примере модификация команд, обрабатывающих операнд a_i , состоит в увеличении их адресной части на q после каждого цикла счета.

Благодаря автоматической модификации команд длина программы существенно сокращается, так как она содержит лишь группу команд для одного прохождения цикла и команды служебных операций, связанных с модификацией команд и управлением вычислительным циклом. Управление вычислительным циклом должно обеспечить повторение цикла нужное число раз, а затем выход из него. Сказанное поясняется схемой на рис. 8.11, в.

Контроль окончания циклических вычислений и выход из цикла могут производиться по числу проделанных повторений цикла, т. е. по содержанию некоторого счетчика, изменяемого на единицу при каждом повторении цикла, или путем задания некоторого предела для изменения адресной части (адресного кода) в модифицируемой команде.

Автоматическая модификация команд и управление вычислительными циклами в современных ЭВМ обеспечиваются механизмом *индексации*. Это понятие включает в себя специальный способ кодирования команд, командные и аппаратные средства задания и выполнения модификации команд

и управления вычислительными циклами. Упомянутые средства часто называют *индексной арифметикой*.

Как способ модификации команд индексация является развитием метода базирования (относительной адресации), с которым она часто используется совместно. Как средство управления вычислительными циклами индексация основана на использовании специализированных команд условного перехода.

Для выполнения индексации в машину вводятся так называемые *индексные регистры*¹, в качестве которых используются ячейки сверхоперативной («местной») памяти или триггерные регистры. В некоторых машинах наряду с этим при определенных дополнительных условиях в качестве индексных регистров могут служить также любые ячейки ОП. В формате команды выделяется поле X для указания номера индексного регистра. Если $(X) = 0$, то это служит указанием, что данная команда не модифицируется. Исполнительный адрес при индексации формируется путем сложения адресного кода команды (смещения) с содержимым индексного регистра (*индексом*), а при наличии базирования — и с базовым адресом. Именно этот характерный случай одновременного использования базирования и индексации показан на рис. 8.12. В схеме имеются *общие регистры*, которые программист может использовать в качестве базовых, индексных и для других целей. Пусть B и X — номера регистров, хранящих соответственно базовый адрес (B) и индекс (X).

Приведенная схема показывает формирование исполнительного адреса (при $B \neq 0$ и $X \neq 0$)

$$A_n = (B) + (X) + D,$$

которое производится в АЛУ процессора или в специальном сумматоре индексной арифметики. В последнем случае повышается быстродействие, но увеличивается объем оборудования ЭВМ.

Назначение базового адреса, смещения и индекса в определенной степени зависит от соотношения их длины с разрядностью адресной сетки машины. В ЕС ЭВМ 24-разрядный базовый адрес задает область программы или массив данных. Указываемое в команде 12-разрядное смещение может зада-

¹ Наименование индексных регистров связано с тем, что первоначально они предназначались для хранения индекса текущего элемента массива $a_i = 0, 1, 2, \dots$. Хотя назначение индексных регистров по сравнению с первоначальным значительно расширилось, термин сохранился.

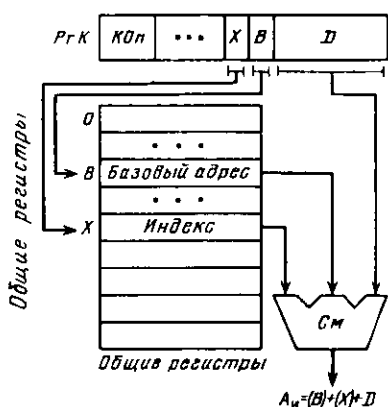


Рис. 8.12. Формирование исполнительного адреса при относительной адресации и индексации

вать отдельные группы (секции) данных в массиве, не превышающие 4095 байт; 24-разрядный индекс служит для указания отдельного элемента в массиве информации.

Индексация служит не только для организации вычислительных циклов, но в сочетании с базированием

(иногда такое сочетание называют двухуровневой индексацией), как это следует из рис. 8.11, а, является средством описания элементов информационных массивов, размещаемых в ОП.

При индексации код команды, хранящийся в ОП, остается неизменным. Следовательно, одна и та же команда может использоваться при соответствующих изменениях индекса для формирования различных исполнительных адресов.

Индексация, указываемая внутри самой команды, может быть названа внутренней, а индексные регистры, определенные таким способом, — внутренними индексными регистрами. Поскольку число разрядов в команде, которое можно выделить для указателя индексации, ограничено, число внутренних индексных регистров невелико.

Помимо внутренней может быть внешняя индексация, при которой указание о необходимости модификации и адрес индексного регистра содержатся в специальной команде «Индексация по предварительной команде», непосредственно предшествующей модифицируемой команде. В этом случае в качестве индексного регистра может быть использована любая ячейка памяти. Внешняя индексация является более медленной по сравнению с внутренней, так как требуется дополнительное время для считывания и расшифровки предварительной команды.

Для управления индексацией используются команды, задающие операции над содержимым индексных регистров, — команды индексной арифметики. Можно отметить основные виды индексных операций: а) загрузка в соответствующий индексный регистр начального значения индекса, б) изменение индекса и в) проверка окончания циклических вычислений.

Изменение индекса состоит в сложении (или вычитании) значения индекса с фиксированным приращением, производимое каждый раз после повторения цикла. Команда изменения индекса указывает номер индексного регистра, а также значение и знак (или адрес) приращения. Если, как это часто бывает, операнды располагаются в последовательных ячейках памяти, то приращение равно 1, а при побайтной адресации — числу байт L в слове. В этом случае может использоваться команда изменения с подразумеваемым операндом, т. е. приращение в команде не указывается.

Для проверки того, что требуемое число повторений цикла произведено и следует выйти из цикла, используются или обычная команда условного перехода по условию, налагаемому в этом случае на некоторую переменную, изменяющуюся в процессе вычислений, или специальные команды «Условный переход по счетчику» и «Условный переход по индексу».

Счетчиком обычно служит один из индексных регистров (общих регистров), в который перед началом вычислительного цикла загружается число повторений цикла. Команда «Условный переход по счетчику» может иметь, например, такой вид

УПСч	R_1	X_2	B_2	D_2

Эта команда уменьшает на 1 содержимое счетчика (в данном случае регистра с номером R_1), и если оно после этого не равно 0, то управление передается по указанному в команде адресу $(B_2) + D_2 + (X_2)$. В противном случае управление переходит к следующей по порядку команде (выход из цикла).

Команда «Условный переход по индексу» (если индекс меньше или равен) имеет, например, вид

УПИ	R_1	R_3	B_2	D_2

Здесь R_1 — номер индексного регистра; R_3 — номер регистра, хранящего приращение; $(B_2) + D_2$ — адрес начала цикла. В регистре с номером $R_3 + 1$ хранится предельное значение индекса. Команда производит суммирование (R_1) с (R_3) . Если новое значение индекса меньше или равно содержимому регистра $R_3 + 1$, это новое значение помещается в регистр с номером R_1 и происходит переход к команде с адресом $(B_2) + D_2$ (повторение цикла). В противном случае выполняется следующая за командой УПИ команда (выход из цикла). Аналогично выполняется проверка окончания цикла командой «Условный переход, если индекс больше».

8.8. СОПОСТАВЛЕНИЕ ПРОГРАММИСТСКИХ МОДЕЛЕЙ (РЕГИСТРОВЫХ СТРУКТУР) МАШИН ОБЩЕГО НАЗНАЧЕНИЯ, МАЛЫХ И МИКРО-ЭВМ И МИКРОПРОЦЕССОРОВ

Совокупность регистров, доступных программисту и используемых им при составлении программ на языке АССЕМБЛЕР или другом машинно-ориентированном языке, будем называть «программистской моделью ЭВМ». На рис. 8.13 представлены три характерных варианта регистровых структур процессоров. Эти варианты охватывают широкий диапазон от крупных машин общего назначения до микропроцессоров.

Обращает на себя внимание ставший стандартным в современной архитектуре ЭВМ прием организации сверхоперативной регистровой памяти в виде блока адресуемых (короткими адресами) *общих регистров*, допускающих многоцелевое использование, — для хранения операндов, результата операции, в качестве базовых, индексных регистров и указателей стеков. В машинах с коротким словом, вынуждающим прибегать к одноадресным командам, один из общих регистров выделяется в качестве *аккумулятора* — регистра, в котором находится один из операндов и в который помещается результат операции. Регистр-аккумулятор в явном виде в команде не адресуется — используется подразумеваемая адресация. В машинах, предназначенных для решения сложных вычислительных задач, регистровая адресуемая память включает в себя специальные регистры для операндов и результатов операций с плавающей запятой (*регистры чисел с плавающей запятой*)¹.

В регистровой структуре процессоров ЕС ЭВМ выделяются 16 общих 32-битных регистра и четыре 64-битных регистра с плавающей запятой. При работе с расширенным 128-битным форматом чисел регистры с плавающей запятой 0,2 и 4,6 парно объединяются.

Архитектура ЭВМ с коротким словом (малые и микро-ЭВМ) сформировалась позже архитектуры машин общего назначения с достаточно длинным (обычно 32-битным) словом. Отмеченные в предыдущих параграфах трудности, обусловленные коротким словом, стимулировали появление новых

¹ В малых ЭВМ блок регистров чисел с плавающей запятой часто поставляется лишь по дополнительному заказу. В микропроцессорах эти регистры могут входить в состав схемы отдельного кристалла (корпуса) — арифметического расширителя.

архитектурных решений – широкого использования стекового механизма, новых методов адресации. Это нашло отражение в том, что в отличие от машин общего назначения (ЕС ЭВМ) для регистровых структур малых и микро-ЭВМ и микропроцессоров характерно наличие регистров – указателей стека (ап-

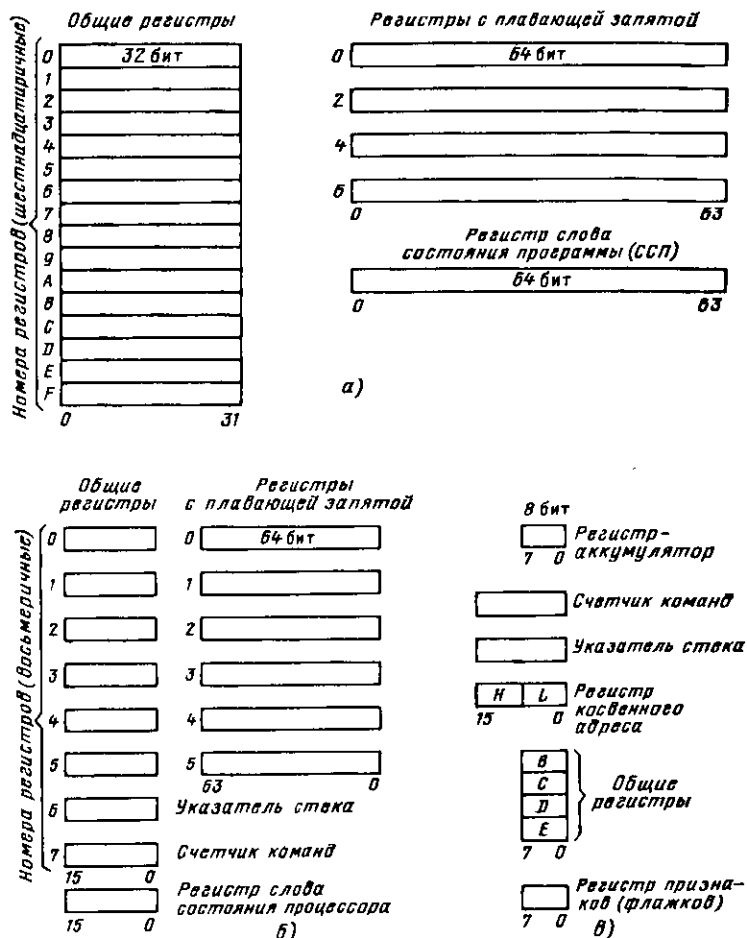


Рис. 8.13. Сопоставление программистских моделей (регистровых структур) машин различного типа:

а – машины общего назначения (ЕС ЭВМ); б – малые и микро-ЭВМ (СМ-4, «Электроника-60»); в – микро-ЭВМ и микропроцессоры (СМ-1800, К580)

паратурного средства реализации стековой памяти)¹ и программно-доступного (адресуемого) регистра счетчика команд, позволяющего реализовывать новые виды адресаций. Для полноты картины следует представленные на рис. 8.13 регистровые структуры рассматривать одновременно с соответствующими адресными структурами ОП на рис. 8.1.

8.9. СТРУКТУРА КОМАНД ЕС ЭВМ II ОЧЕРЕДИ

Рассмотрим на примере ЕС ЭВМ структуры команд современных ЭВМ общего назначения. Системы команд этих машин содержат около 200 различных команд, обеспечивающих программирование разнообразных научных, экономических и технических задач высокой степени сложности. Эти системы команд позволяют работать с различными видами и форматами информации (двоичные и десятичные числа, алфавитно-цифровые данные, числа с фиксированной и плавающей запятой, слова фиксированной и переменной длины). Используемые в ЕС ЭВМ II очереди форматы информации приведены в гл. 2.

Команды имеют длину в одно, два и три полуслова (2, 4, 6 байт). Результат операции, как правило, помещается на место первого операнда.

В ЕС ЭВМ II очереди имеется шесть основных форматов команд, представленных на рис. 8.14.

1. *Формат «регистр — регистр» RR.* В этом формате представляются короткие двухбайтные команды, выполняющие операции над содержимым общих регистров или регистров с плавающей запятой, номера которых указаны в полях R_1 (первый операнд) и R_2 (второй операнд) команды.

2. *Формат «регистр — индексиремая ячейка» RX.* Команда занимает 32-разрядное слово (4 байта). Первый операнд — содержимое общего регистра (или регистра с плавающей запятой), указанного в поле R_1 , второй — содержимое ячейки ОП. Адрес этого операнда определяется содержимым базового B_2 и индексного X_2 регистров и смещением D_2 .

3. *Формат «регистр — память» RS.* Команда также занимает слово (4 байта). Поля R_1 и R_3 определяют номера регистров, содержащих первый и третий операнды; B_2 , D_2 — адрес второго операнда в памяти. Формат RS, в частности, используется для команд групповой передачи данных из группы регистров в группу ячеек ОП и обратно. В этом случае поля R_1 и R_3 задают первый и последний регистры в группе регистров с последовательными номерами, участвующих в передаче.

¹ Стек в ЕС ЭВМ может организовываться программным путем.

4. *Формат «память — непосредственный операнд» SI.* Команда занимает слово (4 байта). Адрес первого операнда, находящегося в ОП, задают поля B_1 , D_1 , а второй операнд I_2 располагается непосредственно в самой команде. Оба операнда имеют длину 1 байт.

5. *Формат S.* Команда занимает слово (4 байта). Первый операнд или его адрес подразумевается, второй находится в ОП.

6. *Формат «память — память» SS.* Команды имеют длину три полуслова (6 байт). Они применяются для операций над словами переменной длины. Оба операнда располагаются

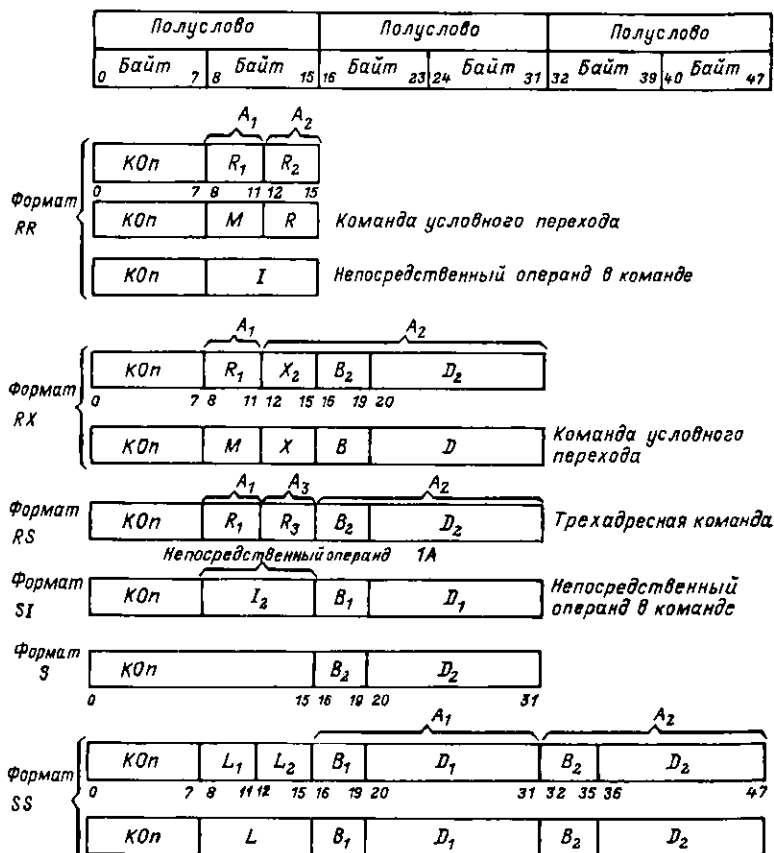


Рис. 8.14. Форматы команд ЕС ЭВМ II очереди

в ОП. Адрес первого операнда (т. е. адрес его левого байта) определяют поля B_1 , D_1 , а адрес второго операнда — B_2 , D_2 .

В командах для операций с десятичными числами (первая команда формата SS) в полях L_1 и L_2 указываются длины операндов (16-ричными цифрами). Десятичные операнды могут быть неодинаковой длины (от 1 до 16 байт).

В командах логической обработки данных, представленных словами переменной длины (вторая команда формата SS), оба операнда имеют одинаковую длину (от 1 до 256 байт), задаваемую в поле L двумя 16-ричными цифрами. Следует заметить, что в командах формата SS длины операндов на единицу превышают значения, указанные в полях L_1 , L_2 , L команды, так как эти поля указывают, сколько байт добавляется к байту, адресуемому исполнительным адресом операнда.

Адрес операнда в ОП образуется суммированием по модулю 2^{24} положительных чисел:

$$A_n = (B) + (X) + D \text{ при наличии индексации;}$$

$$A_n = (B) + D \text{ при отсутствии индексации,}$$

где (B) — базовый адрес (содержимое 24 младших разрядов общего регистра, номер которого указан в поле B команды); (X) — индекс (содержимое 24 младших разрядов общего регистра, указанного в поле X); D — смещение.

При нуле в поле B или X в качестве соответствующего компонента адреса берется 0, а не содержимое общего регистра с номером 0.

Код операции занимает 1 байт и задается двумя 16-ричными цифрами, причем первая цифра определяет группу команд для определенного типа и формата данных или управления, а вторая задает команду в этой группе. Нулевой и первый разряды в двоичном представлении KO_n задают формат команды (00 — RR , 01 — RX , 10 — $RS/SI/S$, 11 — SS).

В ЕС ЭВМ II очереди введен ряд новых функций и соответственно новых команд, в том числе команды операций с плавающей запятой повышенной точности (с числами в расширенном до 128 бит формате).

Введение этих команд позволяет достигнуть нужной точности при решении сложных вычислительных задач за счет уменьшения погрешностей округления.

Введены новые команды для операций ввода-вывода (см. гл. 11), для обмена информацией между процессорами в многопроцессорной (многомашинной) системе (см. гл. 15), для управления отсчетом времени.

8.10. ОСОБЕННОСТИ АДРЕСАЦИИ И СИСТЕМЫ КОМАНД 16-РАЗРЯДНЫХ МАЛЫХ И МИКРО-ЭВМ (СМ-4, СМ-1300, «ЭЛЕКТРОНИКА-60»)

Архитектура 16-разрядных малых ЭВМ СМ-4 и микро-ЭВМ СМ-1300 и «Электроника-60», совпадающая с архитектурой известной американской серии малых ЭВМ PDP-11, оценивается специалистами как высокоэффективная благодаря высококоразвитой системе адресации, использованию стекового механизма, многорегистровой структуре процессора, переменной длине команд и интерфейсу «общая шина». Она оказала определенное влияние на архитектуру микропроцессоров.

Форматы данных, адресная структура памяти и регистровая структура этих машин были представлены на рис. 2.1, 8.1 и 8.13. Машина выполняет операции с 16-битными словами и байтами, которые могут быть двоичными числами со знаком (отрицательные в дополнительном коде) и без знака, логическими полями, а также с 32- и 64-битными числами с плавающей запятой (при наличии дополнительной аппаратуры).

Существенной особенностью архитектуры является развитая система адресации, придающая системе команд большую гибкость и эффективность. Применяется восемь основных типов адресации и, кроме того, несколько типов, получаемых из основных благодаря адресуемости счетчика команд, а также стековая адресация при работе с подпрограммами и при прерываниях. Основные типы адресации представлены в табл. 8.2. Весьма любопытно, что возможная в рассматриваемых машинах адресация в командах счетчика команд как общего регистра порождает дополнительные типы адресации (табл. 8.3).

Основная длина команды — одно 16-битное слово. Однако при индексной и непосредственной адресациях, как это следует из табл. 8.2 и 8.3, сразу за командой в следующей ячейке (или следующих двух ячейках) памяти помещается соответственно адресное приращение или непосредственно адресуемый операнд. Поэтому можно считать, что команды имеют три длины: одно, два и три 16-битных слова (16, 32, 48 бит).

При кодировании команд употребляется расширение кода операции. Код операции может иметь длину 4, 7, 8, 10, 12, 13, 16 бит.

На рис. 8.15 представлены основные форматы команд. В отличие от ЕС ЭВМ, где результат операции замещает всегда операнд в первом адресе, в рассматриваемых ЭВМ результат помещается по второму адресу. Поэтому первое адресное поле называется «Адрес источника» или «*Источник*», а второе «Адрес приемника» или «*Приемник*».

Таблица 8.2. Основные типы адресации (малая ЭВМ СМ-4 и микро-ЭВМ «Электроника-60»)

УА	Мнемоника	Тип адресации	Описание
0	Rn	Регистровая прямая	В Rn (регистре с номером n) операнд
1	$@ Rn$	Регистровая косвенная	В Rn адрес операнда
2	$(Rn) +$	Автоинкрементная прямая	Содержимое Rn используется как адрес операнда, затем перед выполнением команды увеличивается на 1 (при операциях с байтами) или на 2 (при операциях со словами)
3	$@ (Rn) +$	Автоинкрементная косвенная	Содержимое Rn используется как адрес адреса операнда, затем перед выполнением команды содержимое Rn увеличивается на 2
4	$-(Rn)$	Автодекрементная прямая	Содержимое Rn уменьшается на 1 или 2, затем новое содержимое Rn используется как адрес операнда
5	$@ -(Rn)$	Автодекрементная косвенная	Содержимое Rn уменьшается на 2, затем новое содержимое Rn используется как адрес адреса операнда
6	$X(Rn)$	Индексная прямая	Содержимое Rn (индекс) суммируется с приращением X , находящимся в следующей за первым или вторым словом команды ячейке памяти; полученная сумма используется как адрес операнда. Содержимое $CчK$ (регистра 7) увеличивается на 2 сразу после извлечения X
7	$@ X(Rn)$	Индексная косвенная	Содержимое Rn (индекс) суммируется с приращением X , находящимся в следующей за первым или вторым словом команды ячейке памяти; полученная сумма используется как адрес адреса операнда. Содержимое $CчK$ (регистра 7) увеличивается на 2 сразу после извлечения X

Примечания: 1. Используется мнемоника ассемблера ЭВМ СМ-4; Rn — номер регистра в адресном поле.

2. При индексации значение X помещается ассемблером во второе или третье слово команды. Если индексную адресацию имеет только один операнд, применяется двухсловная команда со вторым адресным словом. Если оба операнда имеют индексную адресацию, используется трехсловная команда, причем для адресации источника используется второе, а для приемника третье (адресное) слово.

Таблица 8.3. Дополнительные типы адресации при использовании *СчК* (регистра 7) в качестве общего регистра

УА	Мнемоника	Используемый основной тип адресации	Получаемый дополнительный тип адресации	Описание
2	#A	Автоинкрементная прямая с R7	Непосредственная	Операнд A, присоединяемый в виде второго или третьего слова команды, находится в следующей за первым (или вторым) словом команды ячейке памяти, адресуемой <i>СчК</i> . После выборки A содержимое <i>СчК</i> увеличивается на 2
3	(a # A)	Автоинкрементная косвенная с R7	Абсолютная	Содержимое A адресуемой <i>СчК</i> ячейки, следующей за первым (или вторым) словом команды, является адресом адреса операнда. После выборки A содержимое <i>СчК</i> увеличивается на 2
6	A	Индексная прямая с R7	Относительная	Адресом операнда является сумма A содержимого <i>СчК</i> и содержимого ячейки, следующей за первым (или вторым) словом команды
7	(a A)	Индексная косвенная с R7	Относительная косвенная	Адресом адреса операнда является сумма A содержимого <i>СчК</i> и содержимого ячейки, следующей за первым (или вторым) словом команды

Адресные поля разбиваются на две части: в правых трех разрядах указывается номер регистра, а в левых — тип адресации (указатель адресации УА), как это показано на рис. 8.15, а, б, г. Типы адресаций источника и приемника задаются независимо и могут отличаться друг от друга. Команды в зависимости от местоположений операндов, определяющих число обращений к основной памяти, можно разбить на три типа: а) регистр — регистр, б) регистр — память, в) память — память. Следует подчеркнуть, что в команде адрес операнда в памяти не указывается, так как в этом случае используется косвенная адресация. Результат выполнения команд типа регистр — память в зависимости от кода команды может образовываться в регистре или ячейке памяти (в ЕС ЭВМ в подобном случае результат всегда помещается в регистр).

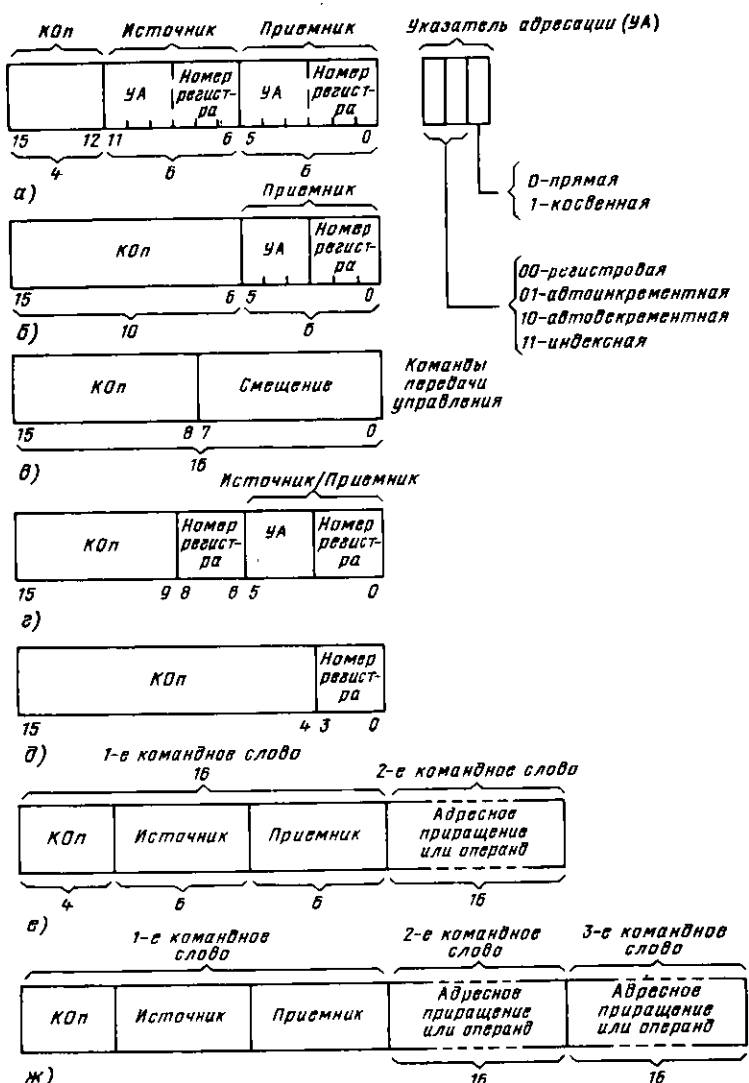


Рис. 8.15. Форматы команд 16-разрядных малых и микро-ЭВМ

Многие операции могут выполняться как со словами, так и с байтами в зависимости от значения 15-го разряда кода команды. Двухадресные команды сложения и вычитания выполняют операцию только со словами.

Среди одноадресных отметим команды «Инкремент» и «Декремент», соответственно увеличивающую и уменьшающую содержимое приемника на 1. Имеется большой набор команд условного перехода по результату предыдущей операции: переходы по равенству 0, неравенству 0, плюсу, минусу, переполнению, отсутствию переполнения, переносу, отсутствию переноса, больше или равно 0, меньше 0, больше 0, меньше и равно 0. В командах условного и безусловного перехода (рис. 8.15, в) старший байт содержит код операции, а младший — 8-разрядное смещение (число со знаком). Адрес перехода формируется следующим образом: знак смещения копируется в разрядах 8—15, полученное 16-разрядное слово умножается на два и складывается с содержимым счетчика команд. При этом оказывается возможным переход относительно *СчК* до 177_8 слов вперед и до 200_8 слов назад. Кроме упомянутой команды безусловного перехода имеется команда абсолютного безусловного перехода (имеет формат и способ задания адресации, как у одноадресных команд), позволяющая передавать управление команде в любой ячейке памяти.

Следует обратить внимание на команды и процедуры с использованием механизма стековой памяти. Команда перехода к подпрограмме, имеющая формат, показанный на рис. 8.15, г, в поле приемника задает адрес начала подпрограммы, а в разрядах 6—8 — номер регистра *R*, который будет использоваться для хранения адреса возврата к основной программе. Процедура выполнения команды начинается с уменьшения содержания указателя стека на 2 и передачи для сохранения в стек содержимого регистра *R*. Затем в регистр *R* передается содержимое *СчК* (адрес возврата), а в *СчК* поступает адрес приемника (рис. 8.15, д) указывается номер регистра, использованного при переходе к подпрограмме для адреса возврата. При выполнении команды из этого регистра адрес возврата передается в *СчК*, и исходное содержимое регистра восстанавливается из стека.

На рис. 15, е и ж представлены форматы команд, имеющие длину соответственно два и три слова (см. табл. 8.2 и 8.3).

Сопоставление описанной в настоящем параграфе системы команд с командами ЕС ЭВМ позволяет заключить, что в рассматриваемых малых и микро-ЭВМ реализуются команды типов *RR*, *RX*, *RS*, *SS*, *SI*, что оказывается возможным благодаря наличию в командах полей *УА*, определяющих функции, выполняемые указанными в команде общими регистрами.

8.11. ОСОБЕННОСТИ АДРЕСАЦИИ И СИСТЕМЫ КОМАНД 8- И 16-РАЗРЯДНЫХ МИКРОПРОЦЕССОРОВ

Микропроцессор К580 выполняет операции над 8-разрядными, а с уменьшенным быстродействием — и над 16-разрядными словами, которые могут быть целыми двоичными числами со знаком и без знака, десятичными двоично-кодированными числами и логическими полями.

Система команд микропроцессоров К580 выглядит довольно примитивной по сравнению с системой команд более крупных ЭВМ, что объясняется коротким 8-разрядным машинным словом и ограниченными аппаратными ресурсами. Операции умножения и деления над целыми числами, операции с плавающей запятой выполняются по подпрограммам или с помощью дополнительного специализированного арифметического микропроцессора («арифметического расширителя»).

Особенности системы команд помимо короткого слова определяются своеобразием архитектуры микропроцессора, которая отражает компромисс между стремлением удовлетворить требованиям многих областей применения и необходимостью уложиться по схемным затратам в рамки, диктуемые существующей в данный момент микроэлектронной технологией.

Отметим некоторые черты архитектуры, существенные для темы настоящего параграфа. При длине слова и ширине выборки из памяти 8 бит адрес в памяти занимает 16 бит, чтобы можно было обеспечить работу с ОП, имеющей емкость 64 Кбайт. На аппаратном уровне это соответствует тому, что в микропроцессоре общие регистры, АЛУ и шина данных, через которую происходит обмен данными между регистрами и между регистрами и памятью, являются 8-разрядными, а регистр адреса ОП и шина адреса — 16-разрядными. Для широкого диапазона применения необходимо, чтобы микропроцессор наряду с 8-разрядными словами мог обрабатывать с умеренными потерями в быстродействии и 16-разрядные слова.

Из сказанного вытекают специфические требования к составу команд (конечно, и к их аппаратной поддержке). Целесообразно иметь команды 2-байтовых («тандемных») передач, арифметических и других операций, в которых некоторые пары регистров адресуются как один регистр, а их содержимое рассматривается как одно 16-разрядное слово, передаваемое по шине данных тандемом — последовательно байт за байтом. В 8-разрядном АЛУ арифметические операции над 16-разрядными числами выполняются двумя командами: первая за-

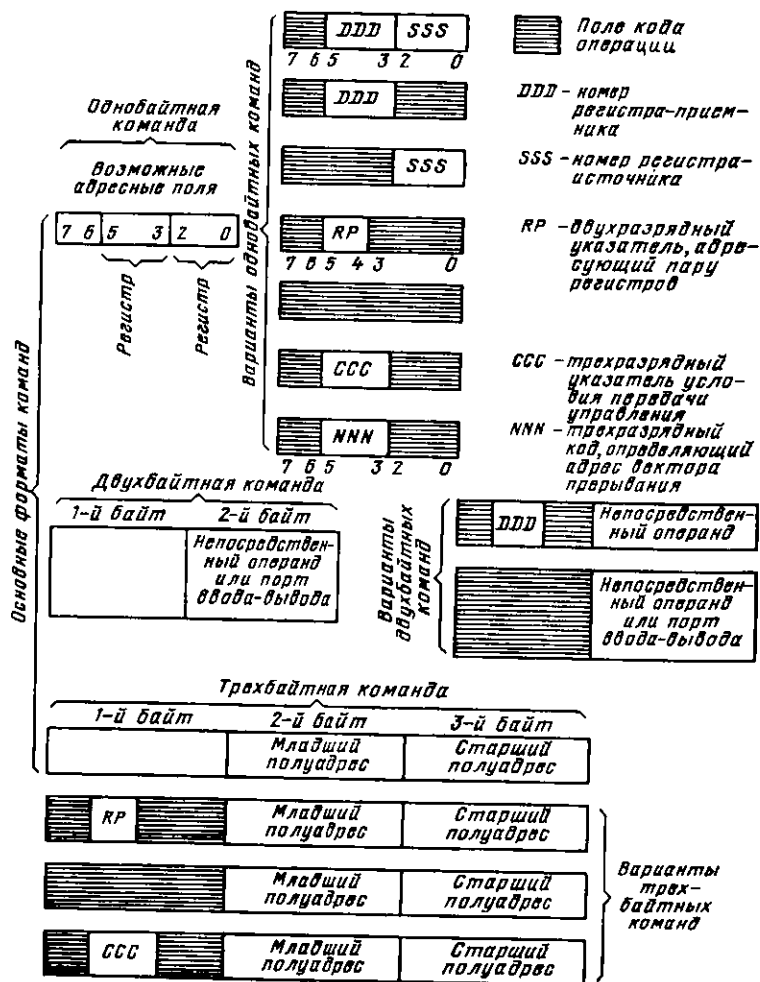


Рис. 8.16. Форматы команд микропроцессоров K580, микро-ЭВМ CM-1800

дает операцию над младшими, а вторая — над старшими байтами. Эти вторые команды — специфические, они задают операцию сложения (вычитания) байт, в которых участвует перенос (заем), полученный в операции с младшими байтами и запомненный в триггере переноса. Микропроцессор K580 имеет три формата команд: однобайтный (однословный), двухбайтный и трехбайтный (рис. 8.16). Формат команды и тип

адресации задаются неявно кодом операции. Адрес команды задается адресом ее первого байта.

Проблема построения системы команд при коротком машинном слове решается благодаря использованию регистра-аккумулятора с подразумеваемой адресацией для реализации одноадресных и безадресных команд. В последних адрес операнда неявно задается кодом операции. Широко применяются укороченная регистровая адресация для обращения к общим регистрам и регистровая косвенная и индексная адресации для задания операнда в ОП. Наличие в регистровой структуре специального 16-разрядного регистра косвенного адреса позволяет иметь команды с подразумеваемой косвенной адресацией, т. е. без указания в команде регистра, хранящего исполнительный адрес.

В двухбайтных командах реализуется непосредственная адресация, а в трехбайтных — прямая адресация ячейки памяти.

Микропроцессор содержит схему «инкрементатор» «декрементатор», которая позволяет реализовать процедуры автоматического задания приращений не только в указателе стека, но и в счетчике команд, в индексном регистре, в регистре косвенного адреса и т. д. Операции «Инкремент» и «Декремент» выполняются в процессе межрегистровых передач.

Стековая адресация используется при работе с подпрограммами и в процедурах прерывания.

Команды с прямой адресацией занимают много места в памяти и затрачивают много времени на выполнение из-за многократных обращений к памяти. Поэтому по возможности следует стремиться к использованию команд с регистровой, непосредственной и регистровой косвенной адресациями. При этом надо помнить, что регистровая косвенная адресация требует предварительной настройки, т. е. загрузки адреса в регистр косвенного адреса. Поэтому регистровая косвенная адресация оказывается эффективной при обработке списочных структур данных, когда после загрузки косвенного адреса он многократно модифицируется командами «Инкремент» или «Декремент».

В командах условного перехода 3-разрядный код ССС задает в трехбайтовой команде условие передачи управления по адресу, указанному в команде. Возможны задания восьми вариантов условия перехода: по переполнению, отсутствию переполнения, нулевому, ненулевому, положительному, отрицательному, четному и нечетному результатам.

Сравнительная простота операций, реализуемых командами, заставляет для выполнения сколько-нибудь сложных опера-

ций (умножения, деления и т. п.) прибегать к подпрограммам.

Для упрощения перехода и возврата из подпрограмм используются стековая память и ряд специальных команд. Имеются команды вызова подпрограммы и возврата из подпрограммы. По первой из них (трехбайтной) содержимое *СчК* побайтно загружается в стек (указатель стека перед загрузкой каждого байта уменьшается на 1), а указанный в команде адрес начала подпрограммы передается в *СчК*. По второй (однобайтной) команде, которая ставится в конце подпрограммы, из стека в *СчК* побайтно передается двухбайтный адрес возврата в основную программу (после передачи каждого байта *УС* увеличивается на 1). Дополнительную гибкость в организации работы с подпрограммами обеспечивают оригинальные команды *вызова по условию подпрограммы и возврата по условию из нее*. Условия те же, что и у команды условного перехода.

Для управления стеком имеются команды передачи из заданной командой пары регистров в стек и обратно, из регистра признаков в стек и обратно, команды обмена содержимого стека и регистровой пары, загрузки указателя стека.

Всего имеется две команды ввода-вывода: *ВВЕСТИ* и *ВЫВЕСТИ* (двухбайтный формат). В команде во втором байте указывается номер порта ввода-вывода, из которого байт информации соответственно вводится или куда байт выводится.

Шестнадцатиразрядный микропроцессор K1810 есть результат дальнейшего развития микропроцессов на основе МОП-технологии. Архитектура K1810 обеспечивает выполнение арифметических и логических операций над 8- и 16-битными операндами, в том числе аппаратную реализацию умножения и деления.

Система команд микропроцессора K1810 включает в себя все команды микропроцессора K580 (чем достигается про-

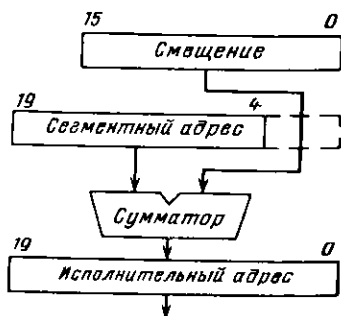


Рис. 8.17. Сегментная адресация в микропроцессоре K1810

граммная совместимость К1810 и К580) и ряд новых эффективных 16-битных команд, обеспечивающих работу с блоком общих регистров с удвоенным числом регистров, адресацию памяти (с байтовой организацией) емкостью до 1 Мбайт.

Шестнадцатибитный адресный код команды позволяет адресовать только 64 Кбайт памяти. Возможность работы с памятью большей емкости обеспечивается *механизмом сегментации*. Память разбивается на произвольное число сегментов (допускается их перекрытие), содержащих каждый не более 64 Кбайт. Начальный адрес сегмента кратен 16, т. е. содержит нули в четырех младших разрядах. Исполнительный (физический) адрес, имеющий 20 разрядов, формируется, как показано на рис. 8.17, путем сложения условно сдвинутого на четыре разряда влево сегментного адреса из сегментного регистра и смещения, задаваемого адресным словом команды.

Имеются четыре сегментных регистра, определяющих четыре непосредственно адресуемых текущих сегмента: программный, сегмент данных, сегмент стека и экстракодový. Сегмент памяти становится текущим и доступным для обращения после загрузки его сегментного адреса в сегментный регистр.

Команды основных операций микропроцессора К1810 задают два операнда, одним из которых является содержимое регистра или ячейки памяти, а другим — содержимое регистра или соответствующего поля в теле команды (непосредственная адресация). Результат операции может записываться на место операнда в регистре или памяти.

В микропроцессоре используются прямая, относительная (с помощью базовых регистров), косвенная и индексная адресации.

Команды выполняют арифметические операции над 8- и 16-разрядными числами со знаком и без знака, над упакованными и неупакованными десятичными двоично-кодированными числами. Для простых операций обработки последовательностей данных имеются однобайтные команды. Используются команды передачи управления двух типов: осуществляющие переходы соответственно внутри текущего программного сегмента и к произвольному сегменту, который при этом становится текущим.

ПРОЦЕССОРЫ И МИКРОПРОЦЕССОРЫ: ПРИНЦИПЫ ОРГАНИЗАЦИИ

9.1. НАЗНАЧЕНИЕ И СТРУКТУРА ПРОЦЕССОРА

Процессором называется устройство, непосредственно осуществляющее процесс обработки данных и программное управление этим процессом. Процессор дешифрирует и выполняет команды программы, организует обращения к оперативной памяти, в нужных случаях инициирует работу периферийных устройств, воспринимает и обрабатывает запросы, поступающие из устройств машины и из внешней среды («запросы прерывания»).

Процессор занимает центральное место в структуре ЭВМ, так как он осуществляет управление взаимодействием всех устройств, входящих в состав ЭВМ.

Выполнение команды (машинной операции) может быть разделено на более мелкие этапы (микрооперации), во время которых исполняются определенные элементарные действия. Конкретный состав микроопераций и последовательность их выполнения определяются системой команд, логической структурой и особенностями данной ЭВМ. Для разных ЭВМ понятия *операция* и *микрооперация* могут иметь различное содержание. Последовательность микроопераций (микрокоманд), реализующих данную операцию (команду), образует микропрограмму операции.

Для определения временных соотношений между различными этапами операции используется понятие *машинный такт*. Машинный такт определяет интервал времени, в течение которого выполняется одна или одновременно несколько микроопераций процессора. Границы тактов задаются синхросигналами, вырабатываемыми специальной схемой — генератором синхросигналов.

Таким образом, может быть установлена следующая иерархия этапов выполнения программ в процессоре: программа, команда (микропрограмма), микрооперация (микрокоманда).

Упрощенная структурная схема процессора представлена на рис. 9.1. На схеме изображены только его основные части: арифметическо-логическое устройство *АЛУ*, управляющее устройство (управляющий автомат) *УУ*, блок управляющих регистров *БУР*, блок регистровой памяти (*местная память*) и блок связи с ОП и некоторым другим, в том числе внешним по отношению к ЭВМ, оборудованием.

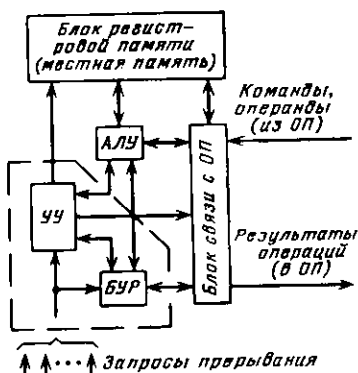


Рис. 9.1. Упрощенная структурная схема процессора

В состав процессора могут также входить и некоторые другие блоки, участвующие в организации вычислительного процесса (блок прерывания, блок защиты памяти, блок контроля правильности работы и диагностики процессора и др.).

Оперативная (основная) память выполняется в виде отдель-

ного устройства, хотя в небольших ЭВМ может конструктивно объединяться с процессором и использовать частично его оборудование.

Арифметическо-логическое устройство процессора производит логические и арифметические операции над данными. В общем случае в АЛУ выполняются логические преобразования над логическими кодами фиксированной и переменной длины (над отдельными битами, группами бит, байтами и их последовательностями), арифметические операции над числами с фиксированной и плавающей запятой, над десятичными числами, обработка алфавитно-цифровых слов переменной длины, а также служебные операции по преобразованию (модификации) кодов команд. Характер выполняемой АЛУ операции задается командой программы.

В процессоре может быть одно универсальное АЛУ для выполнения всех основных арифметических и логических преобразований или несколько специализированных для отдельных видов операций, например для операций над кодами команд, над десятичными числами и т. д. В последнем случае увеличивается количество оборудования процессора, но повышается его быстродействие за счет специализации и упрощения схем выполнения отдельных операций.

Управляющее устройство (управляющий автомат) вырабатывает необходимые управляющие сигналы для выборки очередной команды из памяти, дешифрирования кода команды, формирования адресов операндов, выборки операндов из памяти, передачи их в АЛУ, выполнения в АЛУ операции, предусмотренной кодом команды, передачи полученного в АЛУ результата операции в память, инициирования операций ввода-вывода, организации реакции процессора на запросы прерывания, поступающие от узлов самого процессора, других устройств

машины, в том числе от периферийных устройств ЭВМ и из внешней среды.

В процессорах ЭВМ и микропроцессорах применяют управляющие автоматы с хранимой в памяти логикой (микропрограммные управляющие устройства) и с «жесткой» логикой (см. гл. 7).

Блок управляющих регистров предназначен для временного хранения управляющей информации. Он содержит регистры и счетчики, участвующие в управлении вычислительным процессом: регистры, хранящие информацию о состоянии процессора, регистр-счетчик адреса команды — *счетчик команд*, счетчики тактов, регистр запросов прерывания и др.

К блоку управляющих регистров следует также отнести управляющие триггеры, фиксирующие режимы работы процессора.

Для повышения быстродействия и логических возможностей процессора и микропроцессора в их состав включают *блок регистровой памяти (местную память)* небольшой емкости, но более высокого, чем ОП, быстродействия. Регистры этого блока (или ячейки местной памяти) указываются в командах программы путем укороченной регистровой адресации и служат для хранения операндов, в качестве аккумуляторов (регистров результата операций), базовых и индексных регистров, указателя стека.

В некоторых процессорах базовые и индексные регистры входят в состав блока управляющих регистров.

Местная память выполняется главным образом в виде быстродействующих полупроводниковых интегральных ЗУ (в более ранних моделях ЭВМ — в виде ЗУ на тонких магнитных пленках).

Блок связи (интерфейс процессора) организует обмен информацией процессора с оперативной памятью и защиту участков ОП от недозволённых данной программе обращений, а также связь процессора с периферийными устройствами и внешним по отношению к ЭВМ оборудованием (другими ЭВМ и т. д.).

Блок контроля и диагностики (на рис. 9.1 не показан) служит для обнаружения сбоев и отказов в аппаратуре процессора, восстановления работы программы после сбоев и поиска места неисправности при отказах.

Процессор снабжается пультом со средствами индикации и ручного управления. В предыдущих главах были подробно рассмотрены принципы построения и функционирования основных устройств процессора — АЛУ и управляющего устройства, а также такие вопросы, как методы адресации, кодирова-

ние команд и др. Поэтому в этой главе сосредоточим внимание на методах и процедурах управления вычислительным процессом. Ряд вопросов, относящихся к этой проблеме (защита памяти, автоматический контроль правильности работы ЭВМ и др.), рассматриваются в последующих главах.

9.2. МИКРОПРОГРАММНАЯ ИНТЕРПРЕТАЦИЯ ЯЗЫКА КОМАНД ЭВМ

В гл. 6 и 8 уже использовались микропрограммы для описания операций в АЛУ, способов адресации и передачи управления, с тем чтобы дать возможность читателю детально ознакомиться с соответствующими процедурами.

В настоящем параграфе в целях подробного описания функционирования процессора воспользуемся микропрограммным уровнем рассмотрения рабочего процесса ЭВМ и применим в качестве языка описания язык микроопераций (гл. 5). Такой подход даст возможность представить микропрограммирование как способ интерпретации языка команд ЭВМ.

Чтобы максимально упростить изложение материала, остановимся на предельно простой структуре 8-разрядного процессора (рис. 9.2, а), напоминающего ранние микропроцессоры.

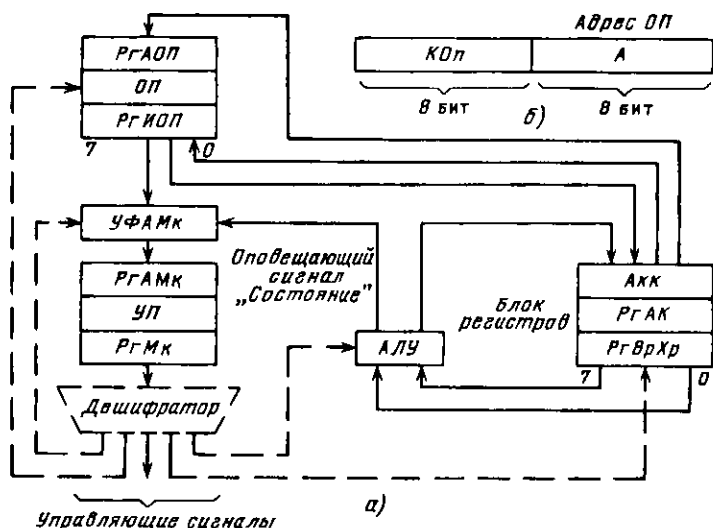


Рис. 9.2. Структура учебного процессора (а) и формат его команды (б)

Процессор содержит 8-разрядные АЛУ, блок регистров (аккумулятор *Акк*, регистр адреса команды *РзАК**, регистр временного хранения *РзВрХр*), основную память *ОП* (ширина выборки 8 бит) с адресным и информационным регистрами *РзАОП* и *РзИОП*, управляющую память (память микрокоманд) *УП* с регистром адреса микрокоманды *РзАМк* и регистром микрокоманд *РзМк*, узел формирования адреса микрокоманды *УФАМк*. На схеме показаны связи между упомянутыми функциональными блоками. Таким образом, введен в рассмотрение некоторый учебный процессор, или микропроцессор [124].

Примем для данного процессора двухбайтный одноадресный формат команды, показанный на рис. 9.2, б. Команда размещается в двух последовательных ячейках ОП.

Реализуем в процессоре следующий набор машинных команд [*А* — адрес 8-битной ячейки ОП, (*А*) — содержимое ячейки]:

Наименование команды	Описание команды
Сложение	$Акк := Акк + (А)$
Вычитание	$Акк := Акк - (А)$
Загрузка	$Акк := (А)$
Запоминание	$(А) := Акк$
Безусловный переход	Передача управления по адресу <i>А</i>
Условный переход	Переход по адресу <i>А</i> , если $Акк = 0$.
	Сохранение естественного порядка выборки команд, если $Акк \neq 0$

Приведенные команды реализуются в принятой структуре процессора с помощью, например, следующего набора микроопераций:

1. $РзИОП := РзZ;$	} Микрооперации межрегистровых передач ($РзZ$ — любой регистр из блока регистров)
2. $РзАОП := РзZ;$	
3. $РзZ := РзИОП;$	
4. $РзZ := Акк;$	} Микрооперации управления коммутатором выбора регистра в блоке регистров
5. $РзZ := РзАК;$	
6. $РзZ := РзВрХр;$	
7. $РзИОП := ОП[РзАОП];$	
8. $ОП[РзАОП] := РзИОП;$	} Микрооперации чтения и записи в ОП
9. $Акк := Акк + РзZ;$	} Микрооперации в АЛУ ($РзZ$ — любой регистр из блока регистров)
10. $Акк := Акк - РзZ;$	
11. $Акк := РзZ;$	
12. $РзZ := Акк;$	
13. $Акк := РзZ + 1;$	

* *РзАК* не является счетчиком.

14. $P_2AM_k := P_2AM_k + 1$;
15. $P_2AM_k := P_2ИОП < \text{в } P_2ИОП \text{ первый байт команды — код операции } >$;
или
 $P_2AM_k := \text{результат дешифрации } КОп \text{ в } P_2ИОП$;
16. $P_2AM_k := \text{Константа } < \text{в } P_2AM_k \text{ передается содержимое из следующей по порядку ячейки УП } >$;
17. **если** $Акк = 0$, **то** $P_2AM_k := P_2AM_k + 2$
иначе $P_2AM_k := P_2AM_k + 1$;

Микрооперации
формирования ад-
реса следующей
микрокоманды

В микрокомандах будут совмещаться некоторые микрооперации, например в микрокоманде передачи байта из аккумулятора в $P_2ИОП$ совмещаются микрооперация, иницилирующая эту передачу, и микрооперация выбора аккумулятора из блока регистров.

Перейдем к составлению микропрограмм отдельных процедур и операций. Выполнение любой команды начинается с процедуры выборки команды из $ОП$ по адресу, находящемуся в $P_2АК$. Представим эту общую для всех команд процедуру в виде отдельной микропрограммы.

Микропрограмма процедуры выборки команды

Адрес УП

- 000 $P_2АОП := P_2АК$ $\langle \text{загрузка адреса команды в } P_2АОП \rangle$;
- 001 $P_2ИОП := ОП[P_2АОП]$ $\langle \text{чтение } КОп \text{ (первого байта) команды в } P_2ИОП \rangle$;
- 002 $P_2ВрХр := Акк$ $\langle \text{сохранение содержимого } Акк \rangle$;
- 003 $Акк := P_2АК + 1$ $\langle \text{формирование адреса второго байта команды} \rangle$;
- 004 $P_2АК := Акк$ $\langle \text{передача адреса второго байта команды в } P_2АК \rangle$;
- 005 $Акк := P_2ВрХр$ $\langle \text{восстановление содержимого } Акк \rangle$;
- 006 $P_2АМ_k := \text{результат дешифрации } КОп \text{ в } P_2ИОП$ $\langle \text{формирование в } P_2АМ_k \text{ начального адреса в УП микропрограммы операции, соответствующей } КОп \text{ команды} \rangle$;

Микропрограмма команды «Сложение»

- $P_2АОП := P_2АК$ $\langle \text{загрузка адреса операнда} \rangle$;
- $P_2ИОП := ОП[P_2АОП]$ $\langle \text{чтение адреса операнда} \rangle$;
- $P_2ВрХр := Акк$ $\langle \text{сохранение содержимого } Акк \rangle$;
- $Акк := P_2АК + 1$ $\langle \text{формирование продвинутого адреса команды} \rangle$;
- $P_2АК := Акк$;

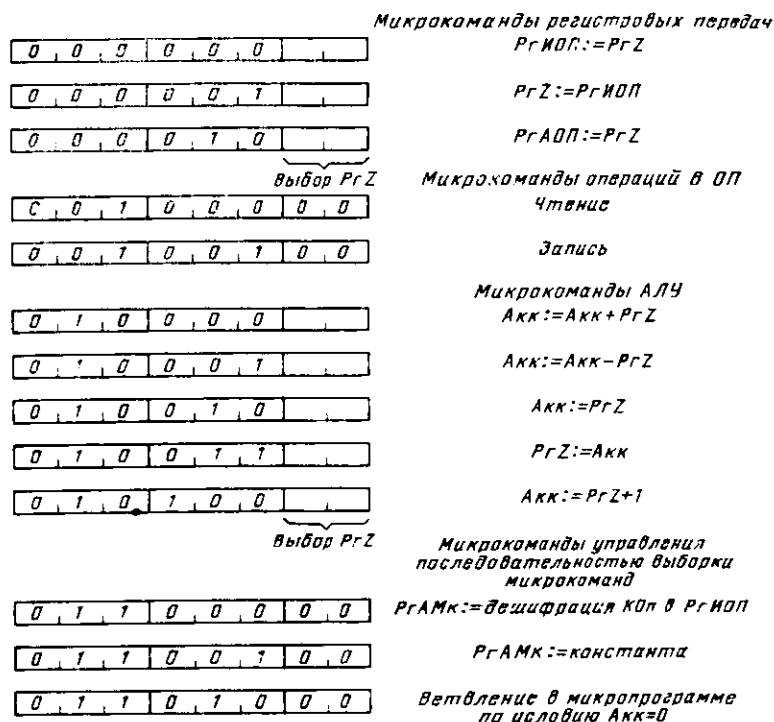


Рис. 9.3. Коды микрокоманд учебного процессора (вертикальное микропрограммирование)



Рис. 9.4. Формат микрокоманды учебного процессора при горизонтально-вертикальном микропрограммировании

$Акк := R_{гВрХр}$ <восстановление содержимого Акк>;

$R_{гВрХр} := R_{гИОП}$ <загрузка адреса операнда для временного хранения>;

$R_{гАОП} := R_{гВрХр}$ <передача адреса операнда>;

$R_{гИОП} := ОП[R_{гАОП}]$ <чтение операнда из ОП>;

$P_2BpXp := P_2ИОП$ <загрузка операнда>;
 $Акк := Акк + P_2BpXp$;
 $P_2AM_k := 0$ <занесение константы 0 — переход к микропрограмме выборки следующей команды>;
 Микропрограмма команды условного перехода, если $Акк = 0$
 $P_2АОП := P_2AK$ <загрузка адреса адреса перехода — второго байта команды условного перехода>;
 $P_2ИОП := ОП[P_2АОП]$ <чтение адреса перехода>;
 $P_2BpXp := Акк$;
 $Акк := P_2AK + 1$;
 $P_2AK := Акк$ <образование в P_2AK адреса следующей команды при естественном порядке выборки>;
 $Акк := P_2BpXp$;
 если $Акк = 0$ то $P_2AM_k := P_2AM_k + 2$ иначе $P_2AM_k := P_2AM_k + 1$;
 $P_2AM_k := 0$ <возврат при $Акк \neq 0$ к микропрограмме выборки команды — сохранение естественного порядка выборки команд>;
 $P_2AK := P_2ИОП$ <загрузка адреса перехода в $P_2AK
 $P_2AM_k := 0$ <возврат к микропрограмме выборки команды>;
 Микропрограммы остальных команд читатель может составить сам.$

Закончим рассмотрение учебного процессора проектированием формата и кодов микрокоманд. На рис. 9.3 и 9.4 представлены коды микрокоманд соответственно для вертикального и горизонтально-вертикального микропрограммирования.

9.3. ПОНЯТИЕ О СОСТОЯНИИ ПРОЦЕССОРА (ПРОГРАММЫ). ВЕКТОР (СЛОВО) СОСТОЯНИЯ

При выполнении процессором программы, как, впрочем, при работе любого другого устройства ЭВМ, после каждого рабочего такта, а тем более в результате завершения выполнения очередной команды, изменяется содержимое регистров, счетчиков, состояния отдельных управляющих триггеров. Можно говорить, что изменяется состояние процессора или, употребляя другую терминологию, *состояние программы*.

Понятие *состояние процессора (состояние программы)* занимает важное место в организации вычислительного процесса в ЭВМ.

Информация о состоянии процессора (программы) лежит в основе многих процедур управления вычислительным процессом, например при анализе ситуаций при отказах и сбоях, при возобновлении выполнения программы после перерывов, выз-

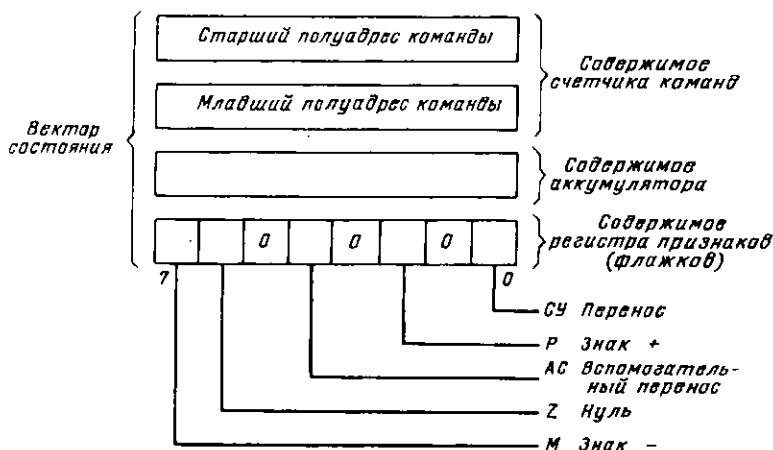


Рис. 9.5. Вектор состояния 8-разрядного микропроцессора К580 (четыре 8-разрядных слова)

ванных отказами, сбоями, прерываниями (см. § 9.4), для фиксации состояния процессора (программы) в момент перехода в мультипрограммном режиме от обработки данной программы к другой и т. п.

Состоянием процессора (программы) после данного такта или после выполнения данной команды, строго говоря, следует считать совокупность состояний в соответствующий момент времени всех запоминающих элементов устройства — триггеров, регистров, ячеек памяти.

Однако не вся эта информация исчезает или искажается при переходе к очередной команде или другой программе. Поэтому из всего многообразия информации о состоянии процессора (программы) отбираются наиболее существенные ее элементы, как правило, подверженные изменениям при переходе к другой команде или программе.

Совокупность значений этих информационных элементов получила название *вектора состояния* или *слова состояния процессора (программы)*.

Вектор состояния в каждый момент времени должен содержать информацию, достаточную для продолжения выполнения программы или повторного пуска программы с точки, соответствующей моменту формирования данного вектора состояния. При этом предполагается, что остальная информация, характеризующая состояние процессора, например содержимое регистров, или сохраняется, или может быть восстановлена программным путем по копии, сохраненной в памяти.

Вектор состояния формируется в соответствующем регистре (регистрах) процессора, претерпевая изменения после выполнения каждой команды.

Наборы информационных элементов, образующих векторы состояния, отличаются у ЭВМ разных типов. Наиболее просто он выглядит у микропроцессоров. Например, вектор состояния микропроцессора K580, как это показано на рис. 9.5, включает в себя содержимое 16-разрядного счетчика команд (адрес очередной команды), содержимое 8-разрядного регистра признаков, называемое в документации на этот микропроцессор *словом состояния процессора*, и содержимое 8-разрядного аккумулятора АЛУ.

Более крупные ЭВМ, например ЭВМ ЕС, имеют более сложные структуры вектора состояния или, иначе говоря, слова состояния программы.

Использование слова (вектора) состояния — распространенный прием построения управления устройствами вычислительной техники. Во многих устройствах ЭВМ для организации их функционирования формируются свои, специфические слова состояния (или *байты состояния*), фиксирующие в виде некоторого кода состояние устройства, например готовность его к выполнению задаваемой операции, успешное или неуспешное завершение операции и т. д.

9.4. ПРИНЦИПЫ ОРГАНИЗАЦИИ СИСТЕМ ПРЕРЫВАНИЯ ПРОГРАММ

Во время выполнения ЭВМ текущей программы внутри машины и в связанной с ней внешней среде (например, в технологическом процессе, управляемом ЭВМ) могут возникать события, требующие немедленной реакции на них со стороны машины.

Реакция состоит в том, что машина прерывает обработку текущей программы и переходит к выполнению некоторой другой программы, специально предназначенной для данного события. По завершении этой программы ЭВМ возвращается к выполнению прерванной программы.

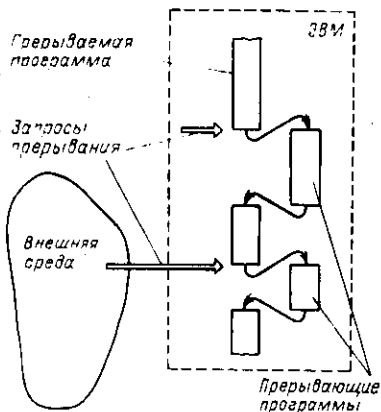
Рассматриваемый процесс, называемый *прерыванием программ*, поясняется на рис. 9.6. Принципиально важным является то, что моменты возникновения событий, требующих прерывания программ, заранее неизвестны и поэтому не могут быть учтены при программировании.

Каждое событие, требующее прерывания, сопровождается сигналом, оповещающим ЭВМ. Назовем эти сигналы *запросами прерывания*. Программу, затребованную запросом прерыва-

Рис. 9.6. Прерывание программ

ния, назовем *прерывающей программой*, противопоставляя ее *прерываемой программе*, выполнявшейся машиной до появления запроса.

Как показано на рис. 9.6, запросы на прерывание могут возникать внутри самой ЭВМ и в ее внешней среде. К первым относятся, например, запросы при возникновении в ЭВМ таких событий, как появление сбоев в ее аппаратуре, переполнение разрядной сетки, попытка деления на 0, выход из установленной для данной программы области памяти, затребование периферийным устройством операции ввода-вывода, завершение операции ввода-вывода периферийным устройством или возникновение при этой операции особой ситуации и др. Хотя некоторые из указанных событий порождаются самой программой, моменты их появления, как правило, невозможно предусмотреть. Запросы во внешней среде могут возникать от других ЭВМ, от аварийных и некоторых других датчиков технологического процесса и т. п.



В сущности запросы прерывания генерируются несколькими развивающимися параллельно во времени процессами, которые в некоторые моменты требуют вмешательства процессора. К этим процессам, в частности, относятся процесс выполнения самой программы, процесс контроля правильности работы ЭВМ, операции ввода-вывода, технологический процесс в управляемом машиной объекте и др.

Возможность прерывания программ — важное архитектурное свойство ЭВМ, позволяющее эффективно использовать производительность процессора при наличии нескольких протекающих параллельно во времени процессов, требующих в произвольные моменты времени управления и обслуживания со стороны процессора. В первую очередь это относится к организации параллельной во времени работы процессора и периферийных устройств машины, а также к использованию ЭВМ для управления в реальном времени технологическими процессами (см. гл. 13).

В некоторых машинах наряду или вместо прерывания с переключением управления на другую программу использует-

ся примитивное прерывание — так называемая *приостановка*, когда по соответствующему запросу приостанавливается выполнение программы и выполняется аппаратными средствами некоторая процедура без изменения содержания счетчика команд, а по ее окончании продолжается выполнение приостановленной программы.

Чтобы ЭВМ могла, не требуя больших усилий от программиста, реализовывать с высоким быстродействием прерывания программ, машине необходимо придать соответствующие аппаратные и программные средства, совокупность которых получила название *системы прерывания программ* или *контролера прерывания*.

Основными функциями системы прерывания являются:

запоминание состояния прерываемой программы и осуществление перехода к прерывающей программе;

восстановление состояния прерванной программы и возврат к ней.

При наличии нескольких источников запросов прерывания должен быть установлен определенный порядок (дисциплина) в обслуживании поступающих запросов. Другими словами, между запросами (и соответствующими прерывающими программами) должны быть установлены *приоритетные соотношения*, определяющие, какой из нескольких поступивших запросов подлежит обработке в первую очередь, и устанавливающих, имеет право или не имеет данный запрос (прерывающая программа) прерывать ту или иную программу. Приоритетный выбор запроса для исполнения входит в процедуру перехода к прерывающей программе.

Характеристики системы прерывания. Для оценки эффективности систем прерывания могут быть использованы следующие характеристики.

Общее количество запросов прерывания (входов в систему прерывания).

Время реакции — время между появлением запроса прерывания и началом выполнения прерывающей программы.

На рис. 9.7 приведена упрощенная временная диаграмма процесса прерывания в предположении, что управление запоминанием состояния и возвратом возложено на саму прерывающую программу, которая в этом случае состоит из трех частей: подготовительной и заключительной, обеспечивающих переключение программ, и собственно прерывающей программы, выполняющей затребованную запросом работу.

На рис. 9.7 время реакции обозначено t_p . Для одного и того же запроса задержки в исполнении прерывающей программы зависят от того, сколько программ со старшим приоритетом

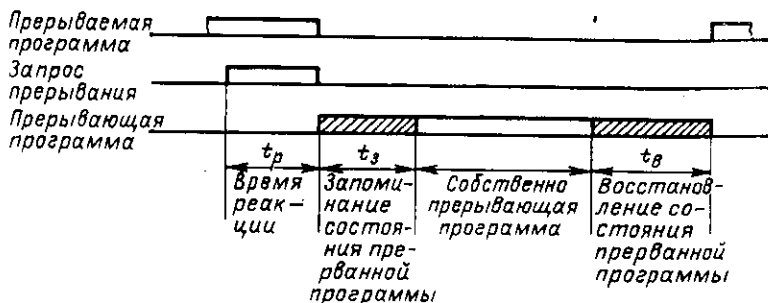


Рис. 9.7. Упрощенная временная диаграмма процесса прерывания

ждет обслуживания. Поэтому время реакции определяют для запроса с наивысшим приоритетом.

Затраты времени на переключение программ (издержки прерывания) равны суммарному расходу времени на запоминание и восстановление состояния программы:

$$t_{изд} = t_з + t_в.$$

Глубина прерывания — максимальное число программ, которые могут прерывать друг друга. Если после перехода к прерывающей программе и вплоть до ее окончания прием других запросов запрещается, то говорят, что система имеет глубину прерывания, равную 1. Глубина равна n , если допускается последовательное прерывание до n программ. Глубина прерывания обычно совпадает с числом уровней приоритета в системе прерываний. На рис. 9.8 показано прерывание в системах с различной глубиной прерывания (предполагается, что приоритет каждого следующего запроса выше предыдущего). Системы с большим значением глубины прерывания обеспечивают более быструю реакцию на срочные запросы.

Насыщение системы прерывания. Если запрос окажется необслуженным к моменту прихода нового запроса от того же источника, то возникает так называемое насыщение системы прерывания. В этом случае предыдущий запрос прерывания от данного источника будет машиной утрачен, что недопустимо. Быстродействие ЭВМ, характеристики системы прерывания, число источников прерывания и частоты возникновения запросов должны быть согласованы таким образом, чтобы насыщение было невозможным.

Допустимые моменты прерывания программ. Большей частью прерывание допускается после окончания любой теку-

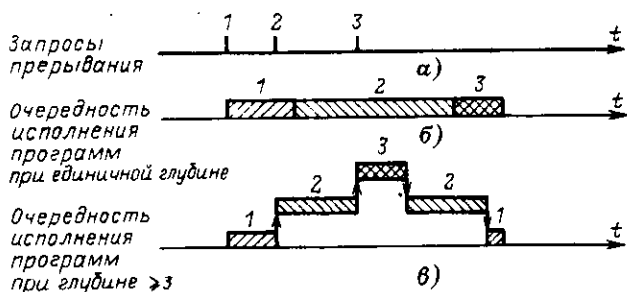


Рис. 9.8. Прерывание в системах с различной глубиной прерывания

щей команды. В этом случае время реакции определяется в основном длительностью выполнения одной команды.

Это время реакции может оказаться недопустимо большим для ЭВМ, предназначенных для работы в реальном масштабе времени. В таких машинах часто допускается прерывание после любого такта выполнения команды. Однако при этом возрастает количество информации, подлежащей запоминанию и восстановлению при переключении программ, так как в этом случае необходимо сохранять также и состояния в момент прерывания счетчика тактов, регистра кода операции и некоторых других. Поэтому такая организация прерывания возможна только в машинах с быстродействующей сверхоперативной памятью.

Имеются ситуации, в которых желательно немедленное прерывание. Если, например, аппаратура контроля обнаружила ошибку, то целесообразно сразу же прервать операцию, пока ошибка не оказала влияние на следующие такты работы машины.

Число классов (уровней) прерывания. В ЭВМ число различных запросов (причин) прерывания может достигать нескольких десятков или сотен. В таких случаях часто запросы разделяют на отдельные классы или уровни.

Совокупность запросов, инициирующих одну и ту же прерывающую программу, образует класс или уровень прерывания (рис. 9.9).

Запросы всех источников прерывания поступают на регистр запросов прерывания $P_2ЗП$, устанавливая соответствующие его разряды (флажки) в состояние 1, указывающее на наличие запроса прерывания определенного источника. Запросы классов прерывания $ЗПК_1 - ЗПК_k$ формируются элементами ИЛИ, объединяющими разряды $P_2ЗП$, относящиеся к соответствующим классам (уровням). Еще одна схема ИЛИ формирует об-

ший сигнал прерывания *ОСП*, поступающий в устройство управления процессора. Значение сигнала *ОСП* определяется выражением

$$ОСП = \bigvee_{i=1}^k ЗПК_i = \bigvee_{i=1}^k \left(\bigvee_{j=0}^{r_i} I_{ji} \right).$$

Информация о действительной причине прерывания, породившей запрос данного класса, содержится в коде прерывания, который отражает состояние разрядов *РзЗП*, относящихся к данному классу прерывания. После принятия запроса прерывания на исполнение и передачи управления прерывающей программе соответствующий триггер *РзЗП* сбрасывается. Объединение запросов в классы прерывания позволяет уменьшить объем аппаратуры, но связано с замедлением работы системы прерывания.

Организация перехода к прерывающей программе. Приоритетное обслуживание запросов прерывания. Назовем вектором прерывания вектор начального состояния прерывающей программы. Вектор прерывания содержит всю необходимую информацию для перехода к прерывающей программе, в том числе ее начальный адрес. Каждому запросу (уровню) прерывания, а в ряде случаев, например в малых и микро-ЭВМ и микропроцессорах, каждому периферийному устройству соответствует свой вектор прерывания, способный инициировать выполнение соответствующей прерывающей программы. Векторы прерывания обычно находятся в специально выделенных фиксированных ячейках памяти.

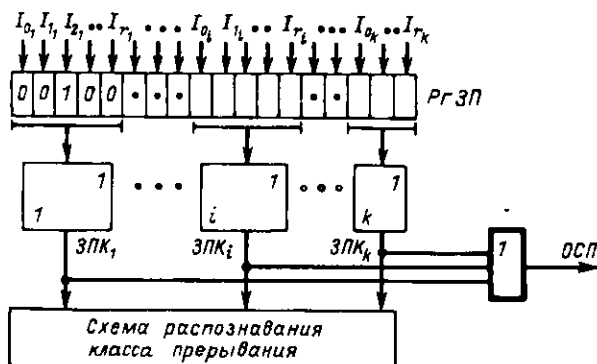


Рис. 9.9. Разделение запросов на классы прерывания

Главное место в процедуре перехода к прерывающей программе занимают передача из соответствующего регистра (регистров) процессора в память (в частности, в стек) на сохранение текущего вектора состояния прерываемой программы (чтобы можно было вернуться к ее исполнению) и загрузка в регистр (регистры) процессора вектора прерывания прерывающей программы, к которой при этом переходит управление процессором.

В простейшем случае вектор прерывания состоит лишь из начального адреса прерывающей программы.

Процедура организации перехода к прерывающей программе включает в себя выделение из выставленных запросов такого, который имеет наибольший приоритет.

Различают *абсолютный* и *относительный приоритеты*. Запрос, имеющий абсолютный приоритет, прерывает выполняемую программу и инициирует выполнение соответствующей прерывающей программы. Запрос с относительным приоритетом является первым кандидатом на обслуживание после завершения выполнения текущей программы.

Если наиболее приоритетный из выставленных запросов прерывания не превосходит по уровню приоритета выполняемую процессором программу, то запрос прерывания игнорируется или его обслуживание откладывается до завершения выполнения текущей программы.

Простейший способ установления приоритетных соотношений между запросами (уровнями) прерывания состоит в том, что приоритет определяется порядком присоединения линий сигналов запросов ко входам системы прерывания. При появлении нескольких запросов прерывания первым воспринимается запрос, поступивший на вход с меньшим номером. В этом случае приоритет является жестко фиксированным. Изменить приоритетные соотношения можно лишь пересоединением линий сигналов запросов на входах системы прерывания.

Процедура прерывания с опросом источников (флажков) прерывания. При указанном способе задания приоритета между запросами каждому источнику запросов соответствует разряд (*флажок*) в регистре запросов прерывания (*регистре флажков*).

При наличии запроса или нескольких запросов прерывания формируется общий сигнал прерывания ОСП (как это, например, показано на рис. 9.9), инициирующий выполняемую специальной программой или аппаратурой процедуру опроса регистра прерывания (флажков) или просто линий сигналов прерывания для установления источника, выставившего запрос прерывания наибольшего приоритета. По существу эта проце-

дура состоит в определении местоположения крайней слева единицы (крайнего флага) в регистре запросов прерывания.

На рис. 9.10 показаны различные способы реализации процедуры опроса источников сигналов прерывания. На рис. 9.10, а показан процесс прерывания с программным опросом флажков прерывания (или, другими словами, опросом периферийных устройств, затребовавших передачу данных). Программный опрос источников прерываний занимает сравнительно много времени. Для уменьшения этого времени процедуру опроса реализуют аппаратным путем.

Схема циклического опроса запросов (источников) прерываний (рис. 9.10, б). Опрос k линии запросов прерывания (или разрядов регистра запросов прерывания) производится последовательно (циклически) с помощью n -разрядного счетчика ($2^n \geq k$), на который с некоторой частотой поступают импульсы от генератора. Поиск приоритетного запроса прерывания начинается со сброса счетчика и одновременно триггера T в нулевое состояние, при этом импульсы генератора начинают поступать на вход счетчика. При помощи дешифратора и элементов $И$ в каждом такте поиска проверяется наличие запроса прерывания, номер которого совпадает с кодом счетчика. Если на данном входе нет запроса прерывания, то после прибавления 1 к счетчику проверяется следующий по порядку вход. Если имеется запрос, триггер T перебрасывается в 1, при этом в процессор посылается общий сигнал прерывания $ОСИ$ и прекращается поступление импульсов на вход счетчика, т. е. завершается цикл просмотра входов системы прерывания. Содержимое счетчика — код номера старшего по приоритету выставленного запроса — используется для формирования начального адреса прерывающей программы. После передачи управления прерывающей программе счетчик (и триггер T) сбрасывается в 0, и процедура опроса запросов возобновляется, начиная с первого входа.

Циклический (последовательный) опрос входов системы прерывания в аппаратном отношении сравнительно прост, однако время реакции и при этом методе все-таки велико, особенно при большом числе источников запросов. Поэтому во многих случаях, например в ряде микропроцессоров, предназначенных для использования при работе в реальном времени, применяют схемы, позволяющие определять номер выставленного запроса или уровня прерывания старшего приоритета за один такт.

Цепочечная одноканальная схема определения приоритетного запроса («дейзи-цепочка») представлена на рис. 9.10, в. Как и

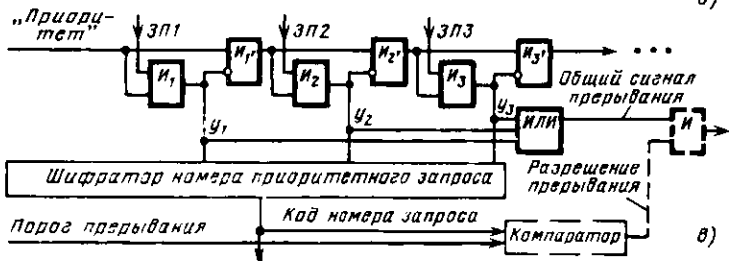
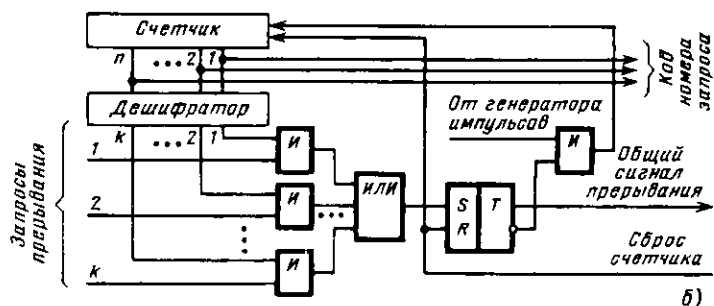
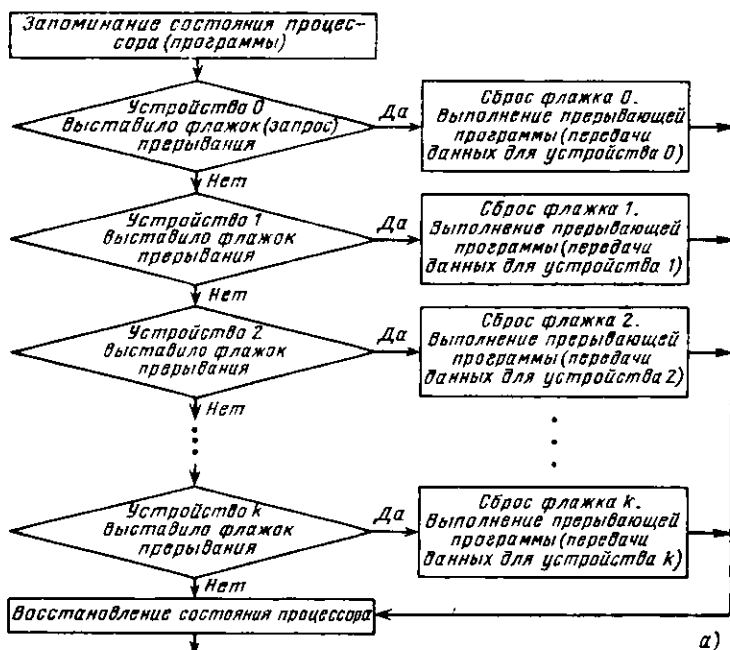


Рис. 9.10. Способы опроса источников сигналов прерывания:

a — программный опрос; *б* — циклический (многотактный) опрос; *в* — цепочечный одноктактный опрос («деджи-цепочка»)

в предыдущих случаях, приоритет запросов прерывания возрастает с уменьшением их номера.

Процедура определения приоритетного запроса инициируется сигналом *Приоритет*, поступающим на цепочку последовательно включенных схем *И*. При отсутствии запросов этот сигнал пройдет через цепочку и сигнал общего запроса прерывания не сформируется. Если среди выставленных запросов прерывания наибольший приоритет имеет *i*-й запрос ($3P_i = 1$, $3P_{i-1} = 0$), то распространение сигнала *Приоритет* правее

схемы *И* с номером *i* блокируется. На *i*-м выходе цепочечной схемы будет сигнал $y_i = 1$, на всех других 0. В процессор поступит общий сигнал прерывания, при этом шифратор по сигналу $y_i = 1$ сформирует код номера *i*-го запроса, принятого к обслуживанию. По сигналу процессора *Подтверждение прерывания* (на рис. 9.10 не показан) этот код передается в процессор и используется для формирования начального адреса прерывающей программы.

Схемы, представленные на рис. 9.10, *б* и *в*, производят поиск крайней левой единицы в наборе сигналов прерывания и формируют код номера *i* запроса, удовлетворяющего условию

$$3P_i \left(\bigwedge_{j=1}^{i-1} 3P_j \right) = 1.$$

Векторное прерывание

Представленные на рис. 9.10 способы определения запроса с наибольшим приоритетом включают в себя так или иначе выполняемую процедуру опроса источников прерывания (входов системы прерывания). Эта процедура, даже если она выполняется аппаратурными средствами, требует сравнительно больших временных затрат.

Более гибким и динамичным является *векторное прерывание*, при котором исключается опрос источников прерывания (флажков регистра прерывания).

Прерывание называется векторным, если источник прерывания, выставив запрос прерывания, посылает в процессор (выставляет на шины интерфейса) код адреса в памяти своего вектора прерывания.

Отметим, что если прерывание на основе опроса источников прерываний всегда сопровождается переходом по одному и тому же адресу и инициирует одну и ту же прерывающую подпрограмму, которая после идентификации источника запроса и формирования адреса начала соответствующей запросу прерывающей программы передает ей управление, то при векторном прерывании каждому запросу или уровню (классу) прерывания или, другими словами, устройству — источнику прерывания соответствует переход к начальному адресу соответствующей прерывающей программы, задаваемому вектором прерывания.

Программно-управляемый приоритет прерывающих программ

Относительная степень важности программ, их частота повторения, относительная степень срочности в ходе вычислительного процесса могут меняться, требуя установления новых приоритетных отношений. Поэтому во многих случаях приоритет между прерывающими программами не может быть зафиксирован раз и навсегда. Необходимо иметь возможность изменять по мере надобности приоритетные соотношения программным путем, другими словами, приоритет между прерывающими программами должен быть динамичным, т. е. программно-управляемым.

В ЭВМ широко применяются два способа реализации программно-управляемого приоритета прерывающих программ, в которых используются соответственно *порог прерывания* (в малых и микро-ЭВМ) и *маски прерывания* (в ЭВМ общего назначения).

Порог прерывания. Этот способ позволяет в ходе вычислительного процесса программным путем изменять уровень приоритета процессора (а следовательно, и обрабатываемой в данный момент на процессоре программы) относительно приоритетов запросов источников прерывания (в основном периферийных устройств), другими словами, задавать порог прерывания, т. е. минимальный уровень приоритета запросов, которым разрешается прерывать программу, идущую на процессоре.

Порог прерывания задается командой программы, устанавливающей в регистре порога прерывания *код порога прерывания*. Специальная схема выделяет наиболее приоритетный запрос прерывания, сравнивает его приоритет с порогом прерывания и, если он оказывается выше порога, вырабатывает

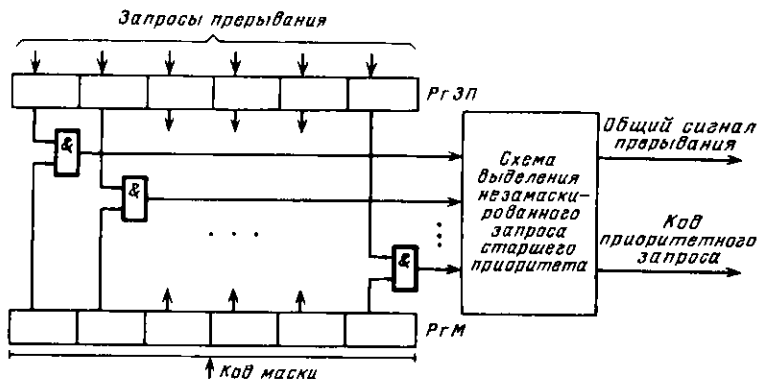


Рис. 9.11. Программно-управляемый приоритет на основе маски прерывания

общий сигнал прерывания, и начинается процедура прерывания (рис. 9.10, в).

В современных ЭВМ общего назначения наибольшее распространение получило программное управление приоритетом на основе маски прерывания (рис. 9.11).

Маска прерывания представляет собой двоичный код, разряды которого поставлены в соответствие запросам или классам прерывания. Маска загружается командой программы в регистр маски. Состояние 1 в данном разряде регистра маски разрешает, а состояние 0 запрещает (*маскирует*) прерывание текущей программы от соответствующего запроса. Таким образом, программа, изменяя маску в регистре маски, может устанавливать произвольные приоритетные соотношения между программами без перекоммутации линий, по которым поступают запросы прерывания. Каждая прерывающая программа может установить свою маску. При формировании маски 1 устанавливаются в разряды, соответствующие запросам (прерывающим программам) с более высоким, чем у данной программы, приоритетом.

Схемы И выделяют незамаскированные запросы прерывания, из которых специальная схема, аналогичная цепочечной схеме на рис. 9.10, в выделяет наиболее приоритетный и формирует код его номера i , удовлетворяющего условию

$$\left(\bigwedge_{j=1}^{i-1} \overline{3П_j P_2 M_j} \right) 3П_i P_2 M_i = 1.$$

С замаскированным запросом в зависимости от причины прерывания поступают двояким образом: или он игнорируется, или запоминается, с тем чтобы осуществить затребованные действия, когда запрет будет снят. Например, если прерывание вызвано окончанием операции в периферийном устройстве, то его следует, как правило, запомнить, так как иначе ЭВМ останется неосведомленной о том, что периферийное устройство освободилось. Прерывание, вызванное переполнением разрядной сетки при арифметической операции, следует при его маскировании игнорировать, так как запоминание этого запроса может привести к тому, что он окажет действие на часть программы или другую программу, к которым это переполнение не относится.

9.5. ОСОБЕННОСТИ СИСТЕМ ПРЕРЫВАНИЯ МАЛЫХ ЭВМ, МИКРОПРОЦЕССОРОВ И МИКРО-ЭВМ

- Во многих малых ЭВМ (например, СМ-4), микропроцессорах и построенных на микропроцессорах микро-ЭВМ реализованы многоуровневые векторные системы прерывания с использованием порога прерывания и стековой памяти в процедурах перехода к прерывающей программе и возврата к прерванной программе.

Векторная система прерывания в малых ЭВМ СМ-4 и микро-ЭВМ СМ-1300

Структура вектора состояния для рассматриваемых машин показана на рис. 9.12. Такова же структура вектора прерывания. В этих машинах возможны внутренние и внешние прерывания. Внутренние прерывания порождаются событиями внутри про-



Рис. 9.12. Вектор состояния процессора в малых ЭВМ (СМ-4)

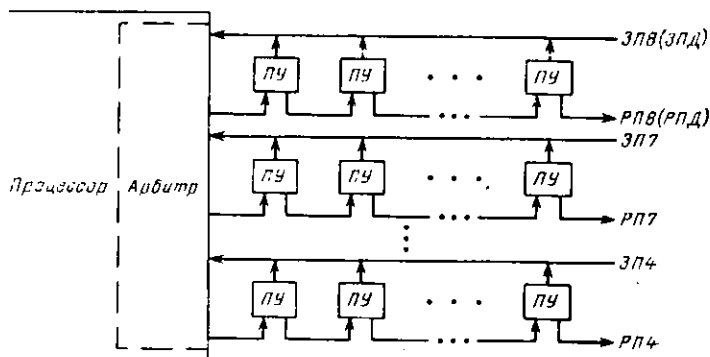


Рис. 9.13. Схема цепей запросов и разрешений прерываний в малых ЭВМ (СМ-4):

ЗП i , РП i – соответственно запрос и разрешение прерывания i -го класса; ЗПД, РПД – запрос и разрешение прямого доступа к памяти.

цессора, в основном самой программой (программные прерывания), точнее, ошибками в программе. К программным прерываниям относятся: а) прерывания при ошибочной адресации, когда в очередной подлежащей исполнению команде программы оказывается нечетный адрес слова или адрес, выходящий за границу адресации имеющейся ОП (вектор прерывания находится по адресу α); б) прерывания при обнаружении в очередной команде кода операции, отсутствующего в системе команд (адрес вектора прерывания β). К внутренним отнесем также прерывания при снижении напряжения питающей сети ниже допустимого значения и восстановлении его нормального уровня (в обоих случаях адрес вектора прерывания – γ). К этой же группе следует отнести прерывание по *разряду слежения* (адрес вектора прерывания ω), возникающее, если перед выполнением некоторой команды установлен в 1 разряд слежения в слове состояния процессора. При внутренних прерываниях векторы прерываний имеют фиксированные адреса¹.

Запросы внешних прерываний генерируются периферийными устройствами, подсоединенными к интерфейсу «общая шина» (см. гл. 11). На рис. 9.13 показана схема присоединения к процессору линий сигналов прерывания. Имеется пять уровней приоритета прерывания: самый старший, абсолютный приоритет принадлежит уровню 8 (запросы прямого доступа к па-

¹ Например, в малой ЭВМ СМ-4 векторам внутренних прерываний присвоены адреса: $\alpha = 4$, $\beta = 10$, $\gamma = 24$, $\omega = 14$.

мяти ЗПД), далее идут уровни 7—4. Каждый уровень обслуживают две линии: линия запросов прерывания ЗП_{*i*} (*i* — номер уровня), к которой параллельно подсоединяются ПУ соответствующего уровня приоритета, и линия разрешения прерывания РП_{*i*} для уровня *i*, проходящая последовательно через все устройства, относящиеся к данному уровню.

Схема Арбитр из выставленных запросов выделяет запрос старшего уровня приоритета и сравнивает его уровень с приоритетом процессора, т. е. с программно-устанавливаемым в регистре слова состояния процессора *Порогом прерывания* (может принимать значения 4—7). Если уровень наиболее приоритетного из выставленных запросов внешнего прерывания превышает порог прерывания, арбитр (процессор) после завершения выполнения текущей команды выдает сигнал разрешения прерывания на линию РП соответствующего уровня. Этот сигнал поступает в первое по пути его прохождения устройство, выставившее запрос и блокирующее дальнейшее распространение сигнала РП.

В случае поступления запросов ЗП14 — ЗП17 устройство, получив разрешение на прерывание, передает в процессор адрес соответствующего вектора прерывания. Процессор, получив адрес вектора прерывания, помещает в стек, т. е. в ячейки памяти, адресуемые указателем стека, два слова вектора состояния: сначала текущее слово состояния процессора (второе слово вектора состояния), затем первое слово — содержимое счетчика команд (продвинутый адрес прерванной программы). Перед каждой передачей в стек значение указателя стека уменьшается на два.

Далее в счетчик команд из ячейки, хранящей первое слово вектора прерывания, передается начальный адрес прерывающей программы, а из следующей ячейки второе слово вектора прерывания заносится в регистр слова состояния процессора. В новом слове состояния процессора порог прерывания должен быть не меньше уровня приоритета принятого к обслуживанию запроса, чтобы повторный запрос от этого источника прерывания не мог прервать выполняемую прерывающую программу. Управление переходит к программе обработки прерывания, заданной вектором прерывания. Если эта программа использует общие регистры, то она начинается с передачи их содержимого в стековую память с помощью команд передачи с автодекрементной прямой адресации по регистру указателя стека Р2 6.

Возврат к прерванной программе осуществляет заключительная часть прерывающей программы, в которой команды

передачи данных с автоинкрементной прямой адресацией по указателю стека производят передачу из стека сохраненных в нем состояний общих регистров в соответствующие регистры. Последней командой прерывающей программы — командой «Возврат из прерываний» первое слово вектора состояния прерванной программы загружается из стека в счетчик команд, а затем второе слово — в регистр слова состояния процессора. Передача каждого слова сопровождается увеличением УС на два. После этого восстанавливается выполнение прерванной программы.

Имеются особенности в процедуре выполнения запросов прерываний ЗП8 (запросов прямого доступа к памяти). Их приоритет всегда выше приоритета процессора. Поэтому в ответ на запрос ЗП8 (ЗПД) сигнал разрешения РП8 (РПД) посылается немедленно, даже если не завершено выполнение текущей команды¹, и производится обмен данными между периферийным устройством и ОП без участия процессора.

Система прерывания 8-разрядного микропроцессора К580

Система прерывания имеет много общего с только что рассмотренной, что определяется в первую очередь тем, что в этом случае используются модификация интерфейса типа «общая шина» («мультишина»), порог прерывания и стековая память. Есть и отличия, например в формировании адреса и структуре вектора прерывания.

Структура вектора состояния (а следовательно, и вектора прерывания) была представлена на рис. 9.5. Он занимает четыре 8-разрядных слова. Векторы прерываний для восьми уровней прерывания хранятся в фиксированных ячейках начальной области памяти с адресами 0, 8, 16, ..., 56.

На рис. 9.14 представлена функциональная схема блока приоритетного прерывания (БПП), входящая в состав схем (корпусов) микропроцессорного семейства. Этот блок приоритетного прерывания реализует восьмиуровневую систему прерывания. Запросы поступают на входы $R_7 - R_0$ (приоритет растет с увеличением номера входа). Число уровней (входов) может быть увеличено присоединением дополнительного БПП. При одновременном появлении нескольких запросов (запрос соответствует низкому уровню потенциала) в регистр запросов преры-

¹ В ЭВМ СМ-4 выдача сигнала РП8(РПД) может быть задержана до завершения процессором начатого сеанса передачи слова через общую шину, а также до завершения команды с плавающей запятой, если на это потребуется не более 8 мкс.

Блок приоритетного прерывания находится в режиме ожидания сигнала разрешения стробирования (на схеме не показан), который формирует микропроцессор по окончании цикла команды. Если в момент стробирования БПП, т. е. в момент поступления от микропроцессора сигнала разрешения прерывания (РП) ($INTE$), на выходе компаратора единичный сигнал, то триггер $T_2П$ переходит в состояние 1 и на выходе буферной схемы формируется общий сигнал прерывания $ОСП$ ($\overline{INT}$), который устанавливает триггер запрета прерывания $Tr_3т$, одновременно блокирующий регистр $Р_3П$ на случай появления запросов с более высоким приоритетом. Сигнал $ОСП$ ($\overline{INT}$) инициирует формирование кода команды передачи управления

$RST\ N$,

которая в данном случае имеет вид 11 $A_0A_1A_2$ 111. При этом 3-разрядный номер $N = A_0A_1A_2$ принятого к обслуживанию запроса передается в микропроцессор по сигналу PBK (ELR). Протекание процедуры прерывания поясняет рис. 9.15.

По команде $RST\ N$ аппаратура МП записывает в стек (в ОП по адресу из УС) содержимое $СчК$ (адрес возврата) и загружает в $СчК$ адрес (соответствует числу $8 \times N$)

$00...0 A_0A_1A_2 000$,

16 разрядов

по которому произойдет передача управления.

Код $N = A_0A_1A_2$ определяет адрес вектора прерывания соответствующей прерывающей программы, а точнее, начальный адрес группы ячеек ОП, хранящей информацию о начальном адресе прерывающей программы. В рассматриваемом МП сохранение в памяти вектора состояния прерываемой программы и замена его в регистрах вектором прерывания прерывающей программы в отличие от малых ЭВМ СМ-4 выполняются в основном не аппаратурными, а программными средствами. Поэтому в данном случае элементы вектора прерывания хранятся в ОП в виде операндов фрагмента программы, выполняющей указанные процедуры (рис. 9.15). С учетом принятого в настоящей главе определения вектора прерывания и описанной выше процедуры прерывания в ЭВМ СМ-4 код $N = A_0A_1A_2$ может рассматриваться как адрес вектора прерывания¹. Код N может

¹ В ряде руководств по микропроцессорам код $N = A_0A_1A_2$ называют вектором прерывания.

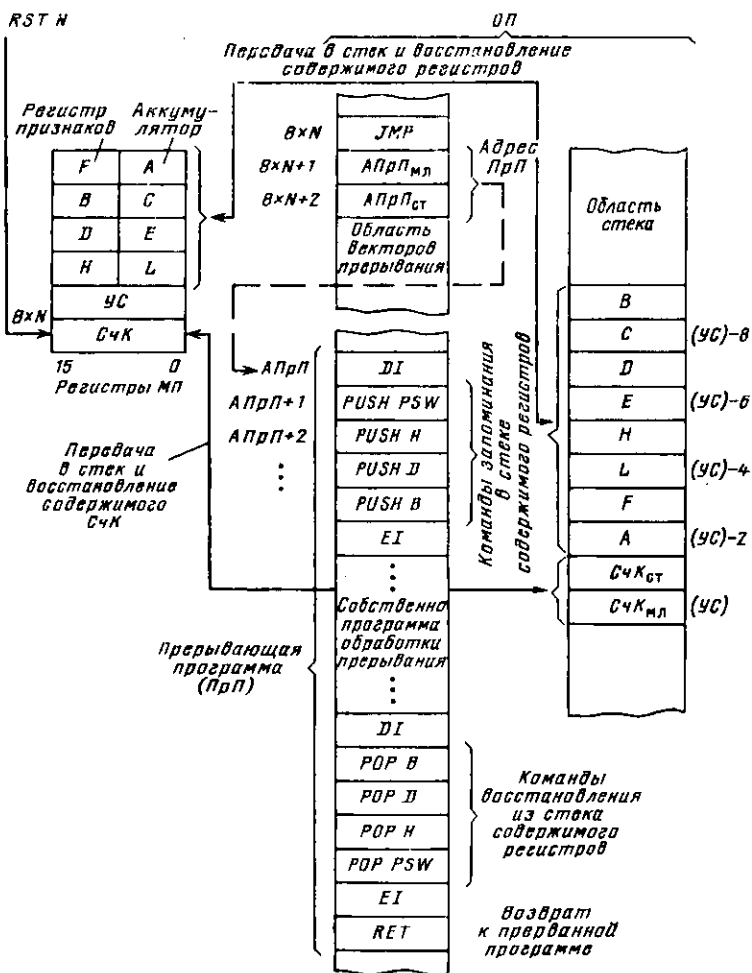


Рис. 9.15. Процедура прерывания программ в микропроцессоре К580: *JMP* – передача управления; *DI* и *EI* – запрещение и разрешение прерывания; *PUSH* и *POP* – запись в стек и считывание из стека; *RET* – загрузка из стека СЧК (возврат к прерванной программе)

не только формироваться аппаратурой, как это было описано выше, но и передаваться источником прерывания, например периферийным устройством, по шине данных интерфейса вместе с запросом прерывания (см. гл. 11).

Управление переходит к команде, загружающей в *CчК* адрес начала прерывающей программы. Начальная часть этой программы помещает в стек содержимое регистра-аккумулятора *A* и регистра признаков *F* (команда *PUSH PSW*), а также и других программно доступных регистров МП, если прерывающая программа будет их использовать. При этом каждой командой передачи в стек передается содержимое соответствующей пары регистров. Далее выполняется собственно программа обработки прерывания.

Перед выходом из прерывающей программы последняя восстанавливает состояния регистров, в том числе аккумулятора и регистра признаков, которые они имели перед прерыванием, и управление переходит к прерванной программе (команда *RET*). Перед выполнением процедуры запоминания и восстановления содержимого регистров прерывание запрещается (команда *DI*), а по их завершении разрешается (команда *EI*).

9.6. СИСТЕМА ПРЕРЫВАНИЯ И НЕКОТОРЫЕ ОСОБЕННОСТИ ОРГАНИЗАЦИИ РЕЖИМОВ УПРАВЛЕНИЯ В ЕС ЭВМ

В машинах общего назначения, входящих в систему ЕС ЭВМ II очереди, представляющей собой существенное развитие ЕС ЭВМ I очереди, возможны два режима: *режим основного управления* (режим *BC*), соответствующий режиму работы машин ЕС ЭВМ I очереди и полностью программно-совместимый с ними; *режим расширенного управления* (режим *EC*), позволяющий использовать новые аппаратные и программные средства и соответственно новые функциональные возможности ЕС ЭВМ II очереди¹.

Для реализации режима расширенного управления в состав процессора введены 16 32-разрядных управляющих регистров и используются дополнительные фиксированные ячейки из постоянно распределенной области памяти. В режиме расширенного управления возможно расширение системы прерывания, реализуются регистрация программных событий, развитая система отсчета времени, средства обеспечения мониторинговых программ, режим виртуальной памяти, блоковое мультиплексирование каналов ввода-вывода и другие дополнительные функции.

¹ Следует отметить, что переход к ЕС ЭВМ II очереди с сохранением совместимости с архитектурой ЕС ЭВМ I очереди достигнут ценой некоторого уменьшения стройности построения управления работой процессора.

В ЕС ЭВМ II очереди имеются следующие классы (уровни) прерывания: 1) от ввода-вывода; 2) программные; 3) при обращении к супервизору; 4) внешние; 5) от схем контроля и 6) повторного пуска (только в режиме ЕС).

В каждом классе имеется несколько источников (причин) прерывания и, следовательно, запросов прерывания, как это было показано на рис. 9.9.

Прерывания от ввода-вывода, поступающие из каналов, указывают на завершение (может быть и неудачное) операций в канале или периферийном устройстве.

Программные прерывания являются результатом ошибок в программе (использование несуществующего кода операции или адреса, нарушение защиты памяти, различные виды переполнений разрядной сетки, исчезновение порядка при арифметических операциях и др.).

Прерывание при обращении к супервизору происходит, если в программе встречается команда «Обращение к супервизору». Этот вид прерывания является средством, позволяющим пользователю инициировать работу супервизора для выполнения определенных действий (например, получить для программы пользователя дополнительную область памяти, пустить операцию ввода-вывода и др.).

Внешние прерывания вызываются сигналами, поступающими от кнопки прерывания на пульте оператора, от датчика времени (таймера), от внешних по отношению к машине объектов.

Прерывания от схем контроля возникают, если обнаруживаются ошибки в работе оборудования машины.

Прерывания повторного пуска поступают от кнопки повторного пуска на пульте оператора.

Система прерывания в ЕС, в сущности, является комбинированной — с аппаратным выделением наиболее приоритетного из запросов от классов прерывания и программно-опросным выявлением конкретной причины прерывания внутри класса по «коду прерывания», который запоминается в ОП при каждом прерывании, а также по дополнительной информации, запоминаемой в ОП при некоторых видах прерывания (например, по слову состояния канала при прерываниях от ввода-вывода). Особую гибкость этой системе прерывания придают маски прерывания.

Текущее состояние программы характеризуется вектором состояния, который в ЕС ЭВМ называется *словом состояния программы* (ССП).

В ЭВМ ЕС, архитектура которых сложилась в основном раньше, чем получили развитие малые и микро-ЭВМ, отсут-

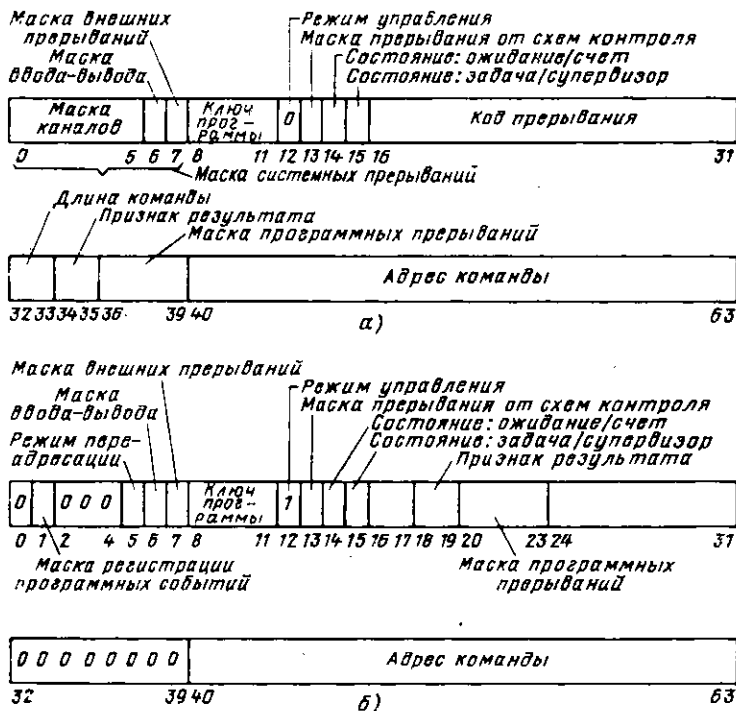


Рис. 9.16. Структура ССП в машинах ЕС ЭВМ:

а — режим основного управления; б — режим расширенного управления

ствуется механизм стековой памяти и ячейки памяти с фиксированными адресами используются для хранения векторов состояний прерванных и векторов прерываний прерывающих программ (которые в данном случае называются старыми и новыми словами состояния программы).

На рис. 9.16 представлена структура ССП для обоих режимов управления. Слово состояния программы занимает два 32-разрядных слова.

Режим управления устанавливается значением 12-го разряда ССП. При 0 в этом разряде имеет место режим основного управления, при 1 — расширенного.

Рассмотрим значения полей ССП основного режима управления.

Непосредственно к системе прерывания относятся поля ССП, содержащие маски прерывания и код прерывания. Предусмотрены следующие маски:

1) *маска системных прерываний*. Отдельные разряды этой маски имеют следующий смысл: 0 — маска мультиплексного канала; 1–6 — маска селекторных каналов соответственно № 1–5; 6 — определяет, допустимо ли прерывание для каналов с номерами выше 5 (прерывание возможно, если равны 1 этот разряд и разряд маски соответствующего канала в управляющем регистре); 7 — маска внешних прерываний. Замаскированные прерывания ввода-вывода и внешние сохраняются в ожидании последующей обработки;

2) *маска прерывания от схем контроля* (разряд 13). Замаскированное прерывание теряется;

3) *маска программных прерываний*. Отдельные разряды этой маски имеют следующий смысл: 36 — маска переполнения с фиксированной запятой; 37 — маска десятичного переполнения; 38 — маска исчезновения порядка; 39 — маска потери значимости (нулевая мантисса). Замаскированные прерывания теряются.

Маски устанавливаются и изменяются программным путем. Прерывание данного класса разрешается при наличии 1 в соответствующем разряде маски и запрещается (маскируется), если этот разряд содержит 0. Установка масок производится либо в момент прерывания в соответствии с содержанием нового ССП, либо при выполнении специальной команды загрузки ССП.

Код прерывания (разряды 16–31) указывает для данного класса прерывания фактическую причину (источник) запроса прерывания. Например, при прерывании от ввода-вывода этот код указывает источник запроса прерывания (номер канала, номер периферийного устройства), при программном прерывании — причину прерывания (некорректность кода операции, адресации, представления данных, переполнение разрядной сетки, исчезновение порядка, потеря значимости и др.).

В остальных полях ССП содержится информация, характеризующая другие параметры состояния программы в момент прерывания. Сюда относятся присвоенный программе код ключа защиты памяти и код признака результата последней выполнявшейся команды. Слово состояния программы содержит адрес команды, с которой программа должна начать работать при новом пуске, а также код длины предшествующей команды. Это позволяет в ряде случаев определить ее адрес, что используется при некоторых прерываниях (например, от программных ошибок).

Слово состояния программы хранит также *программные состояния* процессора для данной программы, определяемые состоянием соответствующих управляющих триггеров.

В рассматриваемых ЭВМ существуют четыре альтернативных программных состояния процессора:

1) *Стоп* или *Работа* (переключение производится вручную). В состоянии *Стоп* команды программы и прерывания не выполняются, электронные часы ЭВМ стоят. В состоянии *Работа* команды и прерывания могут выполняться;

2) *Ожидание* или *Счет* (определяется значением разряда 14 ССП). В состоянии *Ожидание* программа ожидает прерывания (например, от ввода-вывода), команды не выполняются, машинные часы работают. В состоянии *Счет* команды выполняются обычным путем;

3) *Супервизор* или *Задача* (определяется значением разряда 15 ССП). В состоянии *Супервизор* могут выполняться все команды, а в состоянии *Задача* не выполняются команды ввода-вывода и некоторые команды управления (*привилегированные команды*);

4) *Прерывание разрешено* или *Прерывание замаскировано*. Прерывание может быть замаскировано установкой нулей в разряды масок прерываний. Если какие-либо разряды этих масок содержат единицу, то соответствующие прерывания выполняются (состояние *Прерывания разрешено*).

В режиме расширенного управления введены новые прерывания. К ним относятся программные прерывания, связанные: а) с работой монитора (программа сбора статистических показателей о работе программ и процессора), б) с регистрацией программных событий, в) с динамической переадресацией (неправильное использование сегмента, страницы, нарушения спецификации при переадресации). Введены новые внешние прерывания для режима многопроцессорной системы (оповещение о сбое, экстренный сигнал и сигнал внешнего вызова) и от системы отсчета времени (от таймера процессора, компаратора и при нарушении синхронизации часов). Потребовалось ввести новые маски и коды прерывания. Поэтому в режиме ЕС расширен состав и изменено расположение полей в ССП — часть полей ССП вынесена в управляющие регистры и ячейки постоянно распределенной области памяти. Поля с кодом прерывания и кодом длины команды располагаются в фиксированных ячейках этой области памяти.

В управляющем регистре 0 разряд 0 задает (запрещает) блок-мультиплексный режим каналов ввода-вывода; разряд 1 — управление запретом установки маски системы; разряд 2 — управление синхронизацией часов; разряды 8, 9 и 11 — управление размером страниц и сегментов при динамическом распределении памяти; разряды 16–19 — маски оповещения о сбое, экстренном сигнале, сигнале внешнего вызова.

В управляющем регистре 1 разряды 0—7 и 8—25 выделены для указания соответственно длины и адреса таблицы сегментов. Расширенные маски каналов находятся в разрядах 0—31 управляющего регистра 2. Маска монитора находится в разрядах 16—31 управляющего регистра 8. Расширенная маска программных событий размещена в разрядах 0—3 и 16—31 управляющего регистра 9. В управляющих регистрах 10 и 11 в разрядах 8—31 находятся граничные адреса области памяти, на которую распространяется регистрация программных событий.

Процедура перехода к прерывающей программе и возврата из нее, реализуемая в ЕС ЭВМ, пояснена на рис. 9.17. Каждому классу прерывания отводятся в постоянно распределенной области памяти фиксированные ячейки ОП для хранения «старого» ССП и «нового» ССП.

При выполнении текущей программы ее ССП находится в регистре текущего ССП $P_2\text{ССП}$ и участвует в управлении вычислительным процессом, при этом отдельные поля ССП (например, счетчик команд) нужным образом изменяются. В ячейках «новых» ССП для всех классов прерывания хранятся ССП, содержащие информацию, достаточную для начала функционирования соответствующих прерывающих программ.

При поступлении запроса от данного класса прерывания, если этот запрос не маскирован в маске прерывания, аппаратные средства выполняют следующую процедуру. В соответ-

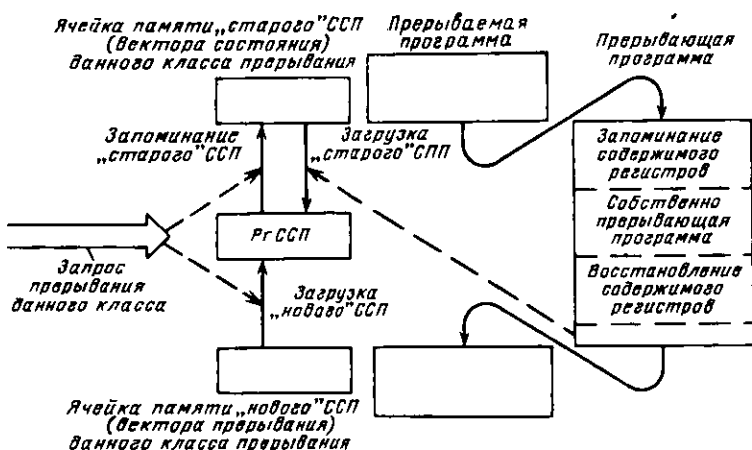


Рис. 9.17. Процедура перехода к прерывающей программе и возврата из нее в ЕС ЭВМ

ствующую группу разрядов *РССП* (режим *ВС*) или соответствующую ячейку постоянно распределенной области памяти (режим *ЕС*) записывается код прерывания, содержащий информацию о конкретной причине прерывания. Затем *ССП* из *РССП* передается в предусмотренную для данного класса прерывания ячейку ОП для «старого» *ССП*, а из ячейки «нового» *ССП* для этого класса «новое» *ССП* (*ССП* прерывающей программы, содержащее в том числе ее начальный адрес) загружается в *РССП*. С этого момента управление переходит к прерывающей программе. На время замены *ССП* прерывания от любых классов запрещены.

Выполнение прерывающей программы начинается с запоминания в памяти содержимого тех общих регистров и регистров плавающей запятой, которые ею будут использованы. Далее выполняется собственно прерывающая программа, которая начинается с анализа «кода прерывания» и определения конкретной причины прерывания и инициирования подпрограммы отработки прерывания, соответствующей этой причине.

Заключительная часть прерывающей программы восстанавливает сохраненное в ОП содержимое регистров и завершает свою работу командой загрузки слова состояния прерванной программы из ячейки «старого» *ССП* в *РССП*. Управление переходит к прерванной программе.

Следует различать приоритет между запросами прерывания разных классов и приоритет между прерывающими программами. Первый устанавливает лишь очередность восприятия запросов, поступивших одновременно, а второй, более важный — старшинство (степень срочности) в выполнении прерывающих программ разных классов, другими словами, определяет, имеет ли право данная прерывающая программа прервать выполняемую в данный момент программу.

Установлен следующий порядок приоритета между запросами прерывания: 1) прерывания от схем контроля; 2) прерывания повторного пуска; 3) программные прерывания или прерывания при обращении к супервизору (запросы этих прерываний не могут возникать одновременно); 4) внешние прерывания; 5) прерывания от ввода-вывода.

При обработке прерывания от схемы контроля прием других запросов прерывания не производится. Следовательно, этот класс прерываний имеет наивысший приоритет не только между запросами прерывания, но и между прерывающими программами. Для других классов прерывания приоритет между прерывающими программами имеет порядок, противоположный порядку приоритета между запросами.

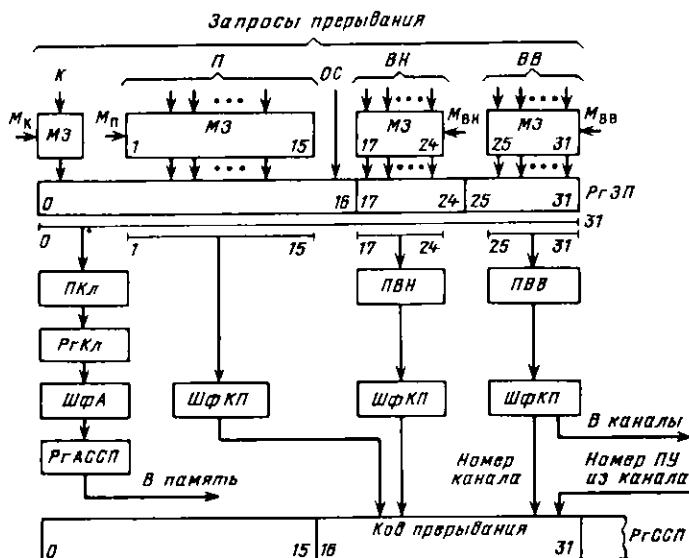


Рис. 9.18. Структура системы прерывания (режим основного управления):

Запросы прерывания: *К* – от схем контроля; *П* – программные; *ОС* – обращение к супервизору; *Вн* – внешние; *ВВ* – ввода-вывода. Маски: контроля $M_K = P_2CCП$ [13]; программы $M_P = P_2CCП$ [36–39]; внешних прерываний $M_{Вн} = P_2CCП$ [7]; ввода-вывода $M_{ВВ} = P_2CCП$ [0–6]

Фактический порядок исполнения прерывающих программ определяется приоритетом не между запросами прерывания, а между прерывающими программами [32].

Функционирование системы прерывания в машинах ЕС ЭВМ поясняется схемой на рис. 9.18 [27]. Работа системы прерывания управляется общими синхросигналами процессора. Запросы прерывания поступают в регистр запросов прерывания *РзП* через схемы маскирования запросов *МЗ*, состоящие из элементов И, управляемых сигналами триггеров соответствующих разрядов масок в *РгССП*. Схема приоритетного выбора запроса класса прерывания *ПКЛ*, анализируя незамаскированные запросы, устанавливает 1 в соответствующий разряд регистра класса прерывания *РгКЛ*, в котором разряды 0–5 присвоены отдельным классам в порядке убывания приоритетов запросов прерывания.

Для принятого к обслуживанию класса прерывания в соответствующее поле *РгССП* передается код прерывания, формируемый шифратором кода прерывания *ШфКП* непосредствен-

но из кода, стоящего в соответствующей классу группе разрядов $P_23П$ при программных прерываниях, и по выходе схемы приоритетного выбора источника запроса прерывания при прерываниях от ввода-вывода и внешних (схемы $ПВВ$ и $ПВн$). При прерываниях от ввода-вывода в поле кода прерывания $P_2ССП$ поступает от соответствующего $ШфКП$ номер канала, а из самого канала — номер периферийного устройства. Заметим, что при прерываниях от схем контроля код прерывания состоит из одних 0.

По коду в $P_2К1$ шифратор адреса $ШфА$ формирует в $P_2АССП$ фиксированный адрес ячейки «старого» ССП данного класса прерывания, куда передается ССП текущей программы из $P_2ССП$. Затем установкой 1 в старший разряд $P_2АССП$ образуется фиксированный адрес ячейки, из которой «новое» ССП загружается в $P_2ССП$, после чего управление переходит к прерывающей программе.

9.7. ВЛИЯНИЕ СИСТЕМЫ ПРЕРЫВАНИЯ НА ОЦЕНКУ ЭФФЕКТИВНОСТИ ВАРИАНТОВ СТРУКТУРЫ ПРОЦЕССОРА

С точки зрения пользователя в структуре процессора целесообразно увеличить число регистров для хранения операндов и промежуточных результатов (регистров-аккумуляторов), базовых и индексных регистров, регистров-указателей стека или общих регистров, способных выполнять все эти функции. Увеличение числа указанных регистров повышает гибкость программирования, облегчает организацию выполнения программ, содержащих большое число модулей, и при определенных условиях приводит к повышению производительности ЭВМ за счет уменьшения числа обращений к ОП в процессе выполнения программы.

Приведет ли действительно увеличение числа общих (базовых, индексных и др.) регистров к возрастанию производительности ЭВМ, зависит от функционирования системы прерывания. Дело в том, что прерывания связаны с потерями рабочего времени процессора (издержками прерывания) на запоминание в ОП содержимого программно-доступных регистров и последующее их восстановление при возврате к прерванной программе. Временные издержки увеличиваются с возрастанием числа регистров.

Поэтому, если частота прерываний достаточно высока, увеличение в структуре процессора числа доступных программе регистров может привести не к увеличению, а к уменьшению производительности ЭВМ [43].

Пусть $P_l^{\text{ном}}$ — номинальная (т. е. при отсутствии прерываний) производительность (число операций в секунду) процессора при некоторой его структуре с l регистрами. Если F — частота прерываний, то, считая, что все прерывания порождают одни и те же издержки времени $t_{\text{изд}l}$ (см. § 9.4), для реальной производительности имеем выражение

$$P_l^p = (1 - Ft_{\text{изд}l}) P_l^{\text{ном}}. \quad (9.1)$$

Пусть в другом варианте структуры процессора увеличено до m ($m > l$) число регистров и за счет этого возросли как номинальная производительность, так и временные издержки при прерывании, которые соответственно составляют $P_m^{\text{ном}}$ и $t_{\text{изд}m}$. При этом

$$P_m^{\text{ном}} > P_l^{\text{ном}} \text{ и } t_{\text{изд}m} > t_{\text{изд}l}.$$

Тогда для варианта с m регистрами, если считать, что частота прерываний не изменилась, имеем

$$P_m^p = (1 - Ft_{\text{изд}m}) P_m^{\text{ном}}. \quad (9.2)$$

Увеличение числа регистров приводит к увеличению реальной производительности

$$P_m^p > P_l^p,$$

лишь если частота прерываний

$$F < (\alpha - 1) / [t_{\text{изд}l}(m\alpha/l - 1)], \quad (9.3)$$

где $\alpha = P_m^{\text{ном}}/P_l^{\text{ном}}$ и приближенно принято $t_{\text{изд}m}/t_{\text{изд}l} = m/l$.

9.8. ПРОЦЕДУРА ВЫПОЛНЕНИЯ КОМАНД. РАБОЧИЙ ЦИКЛ ПРОЦЕССОРА

Функционирование процессора в основном состоит из повторяющихся *рабочих циклов*, каждый из которых соответствует выполнению одной команды программы. Завершив рабочий цикл для текущей команды, процессор переходит к выполнению рабочего цикла для следующей команды программы.

На рис. 9.19 представлена схема рабочего цикла процессора [25]. Хотя эта схема отражает некоторые особенности функционирования процессоров ЕС ЭВМ, тем не менее она имеет достаточно общий характер.

На схеме показаны варианты рабочего цикла для четырех групп команд: 1) основных (осуществляющих арифметические,

логические и пересылочные операции); 2) передачи управления; 3) ввода-вывода и 4) системных (устанавливающих состояние процессора, маску прерывания, слово состояния программы и др.).

Рабочий цикл начинается с распознавания состояния процессора. Устанавливается, какое из альтернативных состояний — *Счет* или *Ожидание* — имеет место. Далее проверяется наличие незамаскированных прерываний.

В состоянии *Ожидание* никакие программы не выполняются. Процессор ждет прихода запроса прерывания, после чего управление переходит к соответствующей прерывающей программе, переводящей процессор в состояние *Счет*.

В состоянии *Счет* при наличии незамаскированных прерываний происходят выход из нормального рабочего цикла и переход к процедуре обработки запросов прерывания.

При отсутствии в состоянии *Счет* запросов прерывания последовательно выполняются этапы рабочего цикла: выборка очередной команды и определение по коду операции команды ее группы, подготовка операндов (формирование исполнительных адресов и выборка операндов из памяти), обработка операндов в АЛУ и запоминание результата.

На этапе выборки очередной команды образуется согласно естественному порядку адрес следующей за ней команды (продвинутый адрес), при этом содержимое счетчика команд (соответствующего поля ССП) увеличивается на число, равное числу байт в очередной команде. В некоторых ЭВМ формирование адреса следующей команды составляет отдельный этап, завершающий рабочий цикл.

В процессе выполнения заданной командой операции формируется *признак результата операции*, используемый командами условного перехода при организации ветвлений в программах.

Указанная выше последовательность этапов составляет основной вариант рабочего цикла, реализуемый при выполнении основных команд.

При выполнении команд передачи управления проверяется заданное командой (например, ее полем маски) условие. Если условие не выполняется, то следующую команду указывает продвинутый адрес, ранее установленный в *СчК* (регистре ССП). Если условие выполняется или имеется один из вариантов команды безусловного перехода, то адрес, задаваемый командой передачи управления, передается в *СчК*.

Команды ввода-вывода инициируют в канале операцию обмена информацией между ядром ЭВМ (основной памятью) и периферийным устройством. Сама эта операция выполняется

каналом под управлением его собственной программы (см. гл. 11). Поэтому на долю процессора остается только процедура опроса состояний канала и периферийного устройства — свободны ли они для операции ввода-вывода. Если свободны, процессор выдает в канал информацию, необходимую для начала операции ввода-вывода. В противном случае процессор переключается в состояние *Ожидание* и ждет сигнала прерывания от этого канала.

Системные команды осуществляют переключения состояния процессора (программы) путем загрузки нового ССП или его части. В частности, эти команды изменяют маски прерывания, устанавливают ключи памяти и ключи защиты в ССП, реализуют операции прямого управления.

9.9. ПРИНЦИП СОВМЕЩЕНИЯ ОПЕРАЦИЙ

АКАДЕМИКА С. А. ЛЕБЕДЕВА. КОНВЕЙЕР ОПЕРАЦИЙ

Вернемся к схеме рабочего цикла (рис. 9.19) и рассмотрим совокупность этапов цикла для основных команд (основной вариант цикла). Если эти этапы выполняются последовательно во времени, то, суммируя обозначенные на рисунке продолжительности отдельных этапов, получаем время цикла

$$t_{\text{пол}} = t_1 + t_2 + t_3 + t_4 + t_5$$

и производительность процессора, операций (команд)/с,

$$P_{\text{пол}} = 1/t_{\text{пол}} = 1/(t_1 + t_2 + t_3 + t_4 + t_5).$$

Во многих случаях последовательная процедура выполнения этапов цикла не обеспечивает требуемую производительность процессора.

Академик С. А. Лебедев в 1956 г. предложил повышать производительность, используя принцип совмещения во времени отдельных операций (этапов) рабочего цикла, и реализовал этот принцип в ЭВМ М-20 в форме параллельного выполнения во времени операции в АЛУ и выборки из памяти следующей команды [50].

Пусть рабочий цикл процессора состоит из k этапов, причем i -й этап имеет продолжительность t_i . Тогда при последовательном выполнении этапов продолжительность процедуры

$$t_{\text{пол}} = \sum_{i=1}^k t_i \quad (9.4)$$

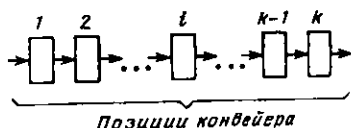


Рис. 9.20. Конвейер операций

и общая производительность процессора, операции/с,

$$P_{\text{посл}} = 1 / \sum_{i=1}^k t_i. \quad (9.5)$$

Скорость работы машины может быть увеличена, если для выполнения каждого этапа иметь отдельный аппаратный блок и соединить эти блоки в обрабатывающую линию (конвейер операций) так, чтобы результат выполнения в данном блоке некоторого этапа передавался для реализации очередного этапа на следующий блок и т. д. (рис. 9.20).

Синхронный конвейер операций [43]. Если конвейер работает в принудительном темпе и для выполнения любого этапа выделено одно и то же время t_T (такт конвейера), то такой конвейер называется синхронным.

Разбиение процедуры на этапы и выбор длительности такта производятся согласно условиям

$$t_T = \max \{t_i\}, \quad i = 1, \dots, k; \quad (9.6)$$

$$t_i + t_{i+1} > t_T, \quad i = 1, \dots, k, \quad (9.7)$$

причем в силу цикличности рабочего процесса в последнем неравенстве принимаем $t_{k+1} = t_1$.

Если для каких-либо смежных этапов второе условие не выполняется, то их следует объединить в один этап либо наиболее длинный этап разбить на несколько этапов. В последнем случае заново выбирается t_T и вновь проверяется условие (9.6).

На рис. 9.21 показана временная диаграмма выполнения команд на 5-позиционном синхронном конвейере. Одинаковыми символами помечены разные этапы рабочего цикла одной и той же команды.

После того как все позиции конвейера окажутся заполненными, параллельно во времени обрабатывается столько команд, сколько в конвейере обрабатывающих блоков (позиций).

Конвейер характеризуется коэффициентом совмещения операций, равным числу одновременно выполняемых этапов обработки информации.

Номинальная производительность синхронного конвейера при его полной загрузке

$$P_{\text{конв}}^{\text{ном}} = 1/t_T. \quad (9.8)$$

Найдем соотношение производительностей процессора при конвейерной обработке и при последовательном выполнении этапов рабочего цикла.

Из (9.4) и (9.6) имеем

$$kt_T \geq \sum_{i=1}^k t_i = t_{\text{посл}}, \quad (9.9)$$

а из (9.4) и (9.7) получаем

$$kt_T < \sum_{i=1}^k (t_i + t_{i+1}) = 2t_{\text{посл}}. \quad (9.10)$$

Из (9.9) и (9.10) получаем

$$k/2 < P_{\text{конв}}^{\text{ном}} / P_{\text{посл}} \leq k. \quad (9.11)$$

В действительности рост реальной производительности процессора окажется ниже из-за простоев (задержек) конвейера. В процедурах выполнения некоторых команд (например, команд пересылки данных) отдельные этапы общего рабочего цикла отсутствуют, и, следовательно, простаивают отдельные блоки конвейера. Для команды условного перехода по результату предыдущей операции выборка следующей команды должна быть задержана (конвейер простаивает несколько тактов), пока не будет сформирован признак результата (формируется на более позднем этапе) предыдущей операции.

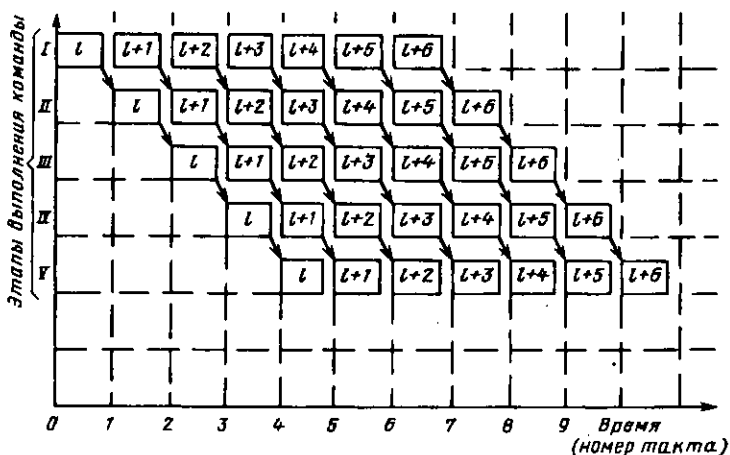


Рис. 9.21. Синхронный конвейер операций

Если p_m — вероятность выборки команды, вызывающей задержку конвейера на m тактов ($m = 1, 2, \dots, k$), то действительная производительность конвейера

$$P_{\text{конв}}^{-1} = P_{\text{конв}}^{\text{ном}} \left(1 + \sum_{m=1}^k p_m m \right). \quad (9.12)$$

Асинхронный конвейер операций [27]. При большой зависимости действительных продолжительностей выполнения процедур отдельных этапов от типа команды и вида операндов целесообразно применение асинхронного конвейера, в котором отсутствует единый такт работы блоков конвейера, а информация с одного блока конвейера передается на следующий, когда данный блок закончит свою процедуру, а следующий полностью освободится от обработки предыдущей команды и полученного при этом результата.

Управление передачей информации между соседними блоками в асинхронном конвейере осуществляется с помощью двух триггеров — готовности блока (сигнализирует о завершении операции в блоке) и освобождения последующего блока.

В качестве примера применения асинхронного конвейера может служить процессор ЭВМ ЕС-1050, в котором реализован 3-позиционный конвейер, обеспечивающий выполнение одновременно трех команд. Рабочий цикл выполнения команды разбит на три этапа: I — выборка очередной команды; II — формирование исполнительных адресов и выборка операндов; III — операция в АЛУ, формирование признака результата и запись результата в память.

Для каждого из указанных этапов выполнения команды имеется соответствующая аппаратура. Например, кроме сумматора АЛУ есть отдельный сумматор для формирования исполнительного адреса на этапе II. На рис. 9.22 представлена структура управляющего устройства с «жесткой» логикой процессора ЭВМ ЕС-1050, на которой показаны блоки, управляющие процедурами отдельных этапов выполнения команды.

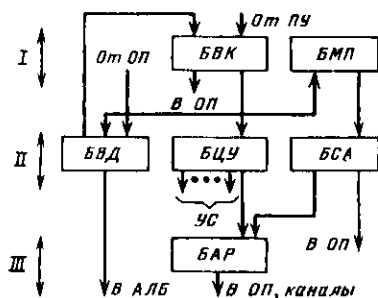


Рис. 9.22. Структура управляющего устройства процессора:

БВК — блок выборки команд; БМП — блок местной памяти; БВД — блок выборки данных; БЦУ — блок центрального управления; БСА — блок сумматора адреса; БАР — блок адреса результата; АЛБ — арифметическо-логический блок; ПУ — пульт управления; УС — управляющие сигналы

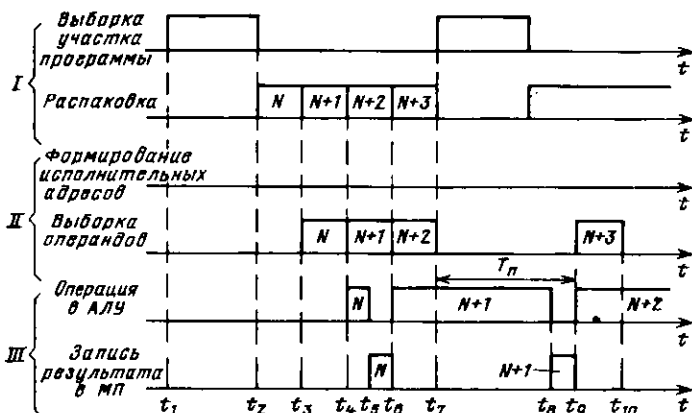


Рис. 9.23. Временная диаграмма совмещения выполнения трех команд (формата RR) в асинхронном конвейере (ЭВМ EC-1050)

На рис. 9.23 показана временная диаграмма совмещения выполнения трех команд в ЭВМ EC-1050 [27]. Временная диаграмма построена для случая, когда выбираемый за одно обращение к памяти «участок программы» содержит четыре команды формата RR.

Этап I содержит две процедуры: выборку из ОП участка программы (8 байт), распаковку участка — выделение из него очередной команды и размещение ее в регистре команды.

Этап II в общем случае включает в себя формирование исполнительных адресов (при выполнении команд формата RR отсутствует) и выборку операндов.

Этап III состоит также из двух процедур: выполнения операций в АЛУ и записи результата в память.

Из диаграммы видно, что, начиная с момента времени t_4 , выполняются одновременно три этапа цикла соответственно для трех команд. В приведенном примере с момента t_7 из-за большой длительности в команде $N+1$ операции в АЛУ приостанавливается работа блоков аппаратуры, соответствующих этапам I и II.

9.10. ОРГАНИЗАЦИЯ 8-РАЗРЯДНЫХ МИКРОПРОЦЕССОРОВ

Развитие интегральной технологии и схемотехники цифровых электронных схем привело к появлению больших и сверхбольших интегральных схем (БИС и СБИС), содержащих на одном кристалле (в одном корпусе) несколько десятков

тысяч, а в последних разработках свыше сотни тысяч элементарных транзисторов. На основе таких схем в последние годы удалось создать микропроцессоры — функционально законченные, управляемые хранимой в памяти программой (большой частью малоразрядные) устройства обработки цифровой информации, выполненные в виде одной или нескольких БИС или СБИС.

Микропроцессоры отличаются крайне малыми габаритами, малым потреблением энергии, довольно большим быстродействием, высокой надежностью и дешевизной. Очень важной особенностью микропроцессоров является их универсальность, т. е. возможность самого разнообразного применения благодаря их программируемости на выполнение конкретных функций. Малые габариты, универсальность, способность реализовать сложные функции обработки данных и управления служат основой для построения микропроцессорных устройств и систем управления, встраиваемых в различные аппараты, машины, орудия, приборы и системы. Применение микропроцессоров является в настоящее время важным направлением технического прогресса.

Развитие микропроцессоров ознаменовалось соревнованием биполярной и МОП-технологий микроэлектроники.

МОП-структуры электронных схем позволяют размещать на одном кристалле большое число элементарных схем благодаря их небольшим размерам и небольшой мощности рассеяния. Однако первые МОП-микропроцессоры имели сравнительно низкое быстродействие (сложение двух слов в регистрах занимало 20 мкс).

Биполярные БИС, например ТТЛ-схемы, обладали намного большим быстродействием, но значительно меньшей плотностью компонентов на кристалле. Поэтому довольно трудно было построить биполярный микропроцессор на одном кристалле.

Для преодоления ограничений, связанных со сравнительно небольшой плотностью компонентов у биполярных схем, предложен метод конструирования микропроцессора, называемый *разрядным слоем*. По этому методу микропроцессор состоит из нескольких одинаковых 2- или 4-разрядных слоев, размещенных на отдельных кристаллах и объединенных общим микропрограммным управлением. Использование метода разрядного слоя в сочетании с прогрессивной технологией маломощных ТТЛ-схем с диодами Шотки позволило к середине 70-х годов создать «микропроцессорное семейство» биполярных БИС, содержащее кристаллы 2- или 4-разрядного слоя и кристаллы, на которых реализованы отдельные законченные

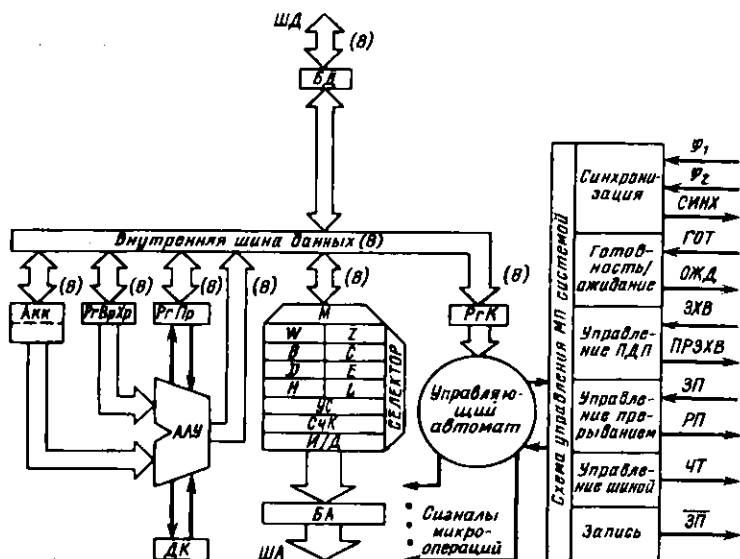


Рис. 9.24. Структура 8-разрядного микропроцессора:

Узлы: АКК — аккумулятор; РгВрХр — регистр временного хранения; РгПр — регистр признаков; М — мультиплексор; И/Д — инкрементор/декрементор; ДК — десятичный корректор; РгК — регистр команд; СЧК — счетчик команд; УС — указатель стека; БД — буфер данных; БА — буфер адреса; шины: ША — шина адреса; ШД — двунаправленная шина данных; сигналы: ЗП — запись; ЧТ — чтение; ЗП — запрос прерывания; ЗХВ — «захват» шин; ПРЗХВ — подтверждение «захвата» шин; ОЖД — ожидание; ГОТ — готовность; СИНХ — синхронизация; Φ_1 и Φ_2 — тактовые сигналы

функциональные блоки памяти, микропрограммного управления и т. д. На основе этого микропроцессорного семейства БИС можно строить микропроцессоры с изменяемой разрядностью слова и микропрограммным управлением (хранимой в памяти логикой). Эти микропроцессоры не имеют фиксированного набора команд, а имеют только набор микрокоманд, что позволяет (хотя это связано с определенными трудностями для широкого пользователя) реализовать микропрограммным путем оптимальный для данной задачи набор команд и отдельных процедур.

Параллельно с этим направлением быстро совершенствовалась МОП-технология и были достигнуты крупные успехи в повышении степени интеграции и быстродействия. На этой основе по мере совершенствования МОП-технологии были созданы 8-, а затем 16-разрядные с жесткой логикой и фиксированной системой команд микропроцессоры, а в последнее время — 32-разрядные микропроцессоры.

В настоящее время в микропроцессорной технике основное место занимают однокристалльные микропроцессоры, выполненные на различных вариантах МОП-технологии.

Структура 8-разрядного микропроцессора К580 представлена на рис. 9.24.

Микропроцессор реализован на кремниевом кристалле размером 10 мм² в корпусе с 40 выводами. Этот кристалл представляет собой выполненную по и-МОП-технологии БИС, содержащую 5 тыс. транзисторов. Длина слова 8 разрядов. Время выполнения операций примерно 15 мкс.

Создание высокопроизводительного микропроцессора с эффективными системой команд и процедурами обмена информацией с внешним по отношению к микропроцессору оборудованием затруднялось ограничениями, которые вызывались коротким словом микропроцессора (МП) и малым числом внешних выводов корпуса МП. Последнее, в частности, обусловило узкую (малоразрядную) шину данных в МП и узкий интерфейс обмена данными с внешним оборудованием.

Для преодоления этих ограничений потребовались разработки целого ряда новых архитектурных решений. С частью из них, относящихся к методам адресации, построению команд, системам прерывания, читатель познакомился в гл. 8 и § 9.5. Здесь остановимся главным образом на структурных аспектах организации МП. Организация ввода-вывода в МП рассмотрена в гл. 11.

Характерными особенностями организации МП К580 являются:

- магистральная структура связей — наличие внутренней шины данных, которая связывает все узлы внутри МП. Ширина шины данных (8 разрядов) равна разрядности слов, с которыми оперирует МП;

- регистровая память в виде блока программно доступных общих и некоторых специализированных регистров (СчК, указатель стека, регистр косвенного адреса, регистр признаков); наличие 16-разрядной шины адреса, позволяющей прямо адресовывать внешнюю память емкостью 64 Кбайт;

- разнообразие применяемых методов адресации — прямая, регистровая прямая, регистровая косвенная (в том числе подразумеваемая), непосредственная, индексная, стековая — в совокупности позволяющих при коротком 8-разрядном слове реализовать достаточно гибкую систему команд, использующую три формата команд: 1, 2, 3 байта (см. § 8.11);

- наличие эффективных средств работы с подпрограммами и быстродействующей системы прерывания, использующих стековую память, специальные команды вызова подпрограмм

и возврата (в том числе условного) из подпрограмм и процедуры перехода к прерывающим и возврата к прерванным программам;

реализация двухбайтных (тандемных) передач и некоторых других двухбайтных операций, с тем чтобы при 8-разрядных шине данных, общих регистрах и ширине выборки из ОП упростить процедуры обработки 16-разрядных слов и работу с 16-разрядными адресами и с тремя форматами команд. Последнее достигается тем, что первый байт команды, содержащий указание о ее формате, загружается всегда в регистр команды, а второй и третий, если они имеются, — в определенные регистры.

Помимо упоминавшихся шин данных и адреса имеется шина управления, содержащая линии, предназначенные для передачи управляющих сигналов, признаков состояния процессора и периферийного оборудования. Шина содержит следующие линии:

синхронизирующих сигналов для сопровождения информации при передачах ее в обоих направлениях по мультиплексированной шине данных;

сигналов, информирующих МП о состоянии (готовности) периферийных устройств;

сигналов запроса прерывания от периферийных устройств и разрешения прерывания и др.

Блок регистров содержит программно доступные (с регистровой прямой или подразумеваемой адресацией) регистры: 8-разрядные аккумулятор *A*, общие регистры *B*, *C*, *D*, *E*, 16-разрядные счетчик команд *CчК*, указатель стека *УС*, парный регистр косвенного адреса *Н-L*, а также 8-разрядный регистр признаков, отдельные разряды (флажки) которого, принимая значение 1, указывают: *СУ* — перенос, *P* и *M* соответственно — знак + и — результата, *АС* — вспомогательный перенос, *Z* — нулевой результат.

Кроме того, имеется регистр адреса ОП. Регистры используются для хранения операндов, промежуточных результатов и адресов.

Непосредственно в МП из оборудования стековой памяти содержатся только указатель стека и соответствующие цепи управления. Сам стек реализуется в виде зоны ячеек в общей ОП. Хотя такая организация стека по сравнению с внутренним (встроенным в кристалл МП) стеком связана с некоторым замедлением стековых процедур, однако операции со стеком выполняются намного реже, чем с другими регистрами. При таком выполнении стека его емкость практически неограниченна, оказывается возможным большее число вложений при работе

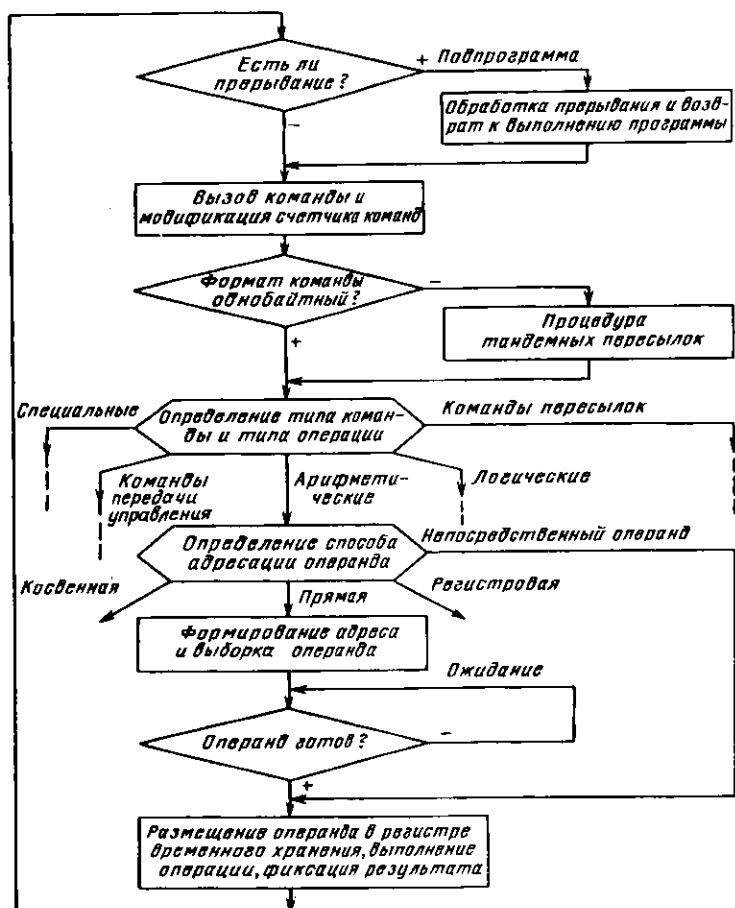


Рис. 9.25. Схема рабочего цикла 8-разрядного микропроцессора

с подпрограммами и прерываниями, что в общем обеспечивает высокую производительность МП.

Блок регистров содержит специальную схему «инкрементор-декрементор», позволяющую без привлечения АЛУ в процессе межрегистровых передач выполнять модификацию текущего адреса добавлением (вычитанием) 1.

Арифметическо-логическое устройство, выполненное в виде комбинационной схемы с собственным регистром временного хранения, производит арифметические и логические операции над 8-разрядными операндами. Один вход АЛУ связан с аккумулятором

мультиплексом, а другой может быть соединен с любым общим регистром. Имеется включаемый специальной командой десятичный корректор, превращающий результат непосредственно предшествующей операции двоичного сложения (вычитания) над двоично-десятичными операндами в результат соответствующей операции десятичной арифметики (см. гл. 6).

По завершению арифметических и логических операций 1 в соответствующих разрядах (флажках) регистра признаков указывают на наличие переполнения, положительного, отрицательного или нулевого результата.

Передача управления в командах условного перехода происходит в зависимости от состояния определенных разрядов (флажков) регистра признаков.

В МП применены буферные схемы шины данных и адреса. Буферная схема состоит из регистра и выходных схем с тремя состояниями: 0, 1 и полное электрическое отключение от нагрузки («высокоимпедансное» или «плавающее» состояние). Наличие третьего состояния позволяет реализовывать магистральный принцип связи между отдельными узлами внутри МП и между модулями (ОП, периферийные устройства), присоединяемыми наряду с МП к общему интерфейсу «мультишина» (см. гл. 11) микропроцессорной системы (или микро-ЭВМ).

Управляющий автомат с «жесткой» логикой, реализованный с использованием программируемой логической матрицы, выдает последовательность управляющих сигналов, инициирующих процедуру выполнения текущей команды. Последовательность сигналов и сама процедура зависят от задаваемых кодом команды формата команды и операндов, способа адресации.

На рис. 9.25 показана схема цикла выполнения команды в микропроцессоре, которая приводится, главным образом, с целью показать читателю, что, кроме некоторых деталей, она не отличается от схем рабочего цикла крупных ЭВМ.

9.11. ОРГАНИЗАЦИЯ 16-РАЗРЯДНЫХ МИКРОПРОЦЕССОРОВ

В последние годы «соревнование» биполярной и МОП-технологий микроэлектроники ознаменовалось новыми крупными достижениями последней, поставившей под сомнение существовавшее мнение, что быстродействие МОП-структур недостаточно для построения высокопроизводительных вычислительных устройств.

На основе *n*-МОП-технологии с кремниевыми затворами удалось реализовать принцип пропорционального уменьшения

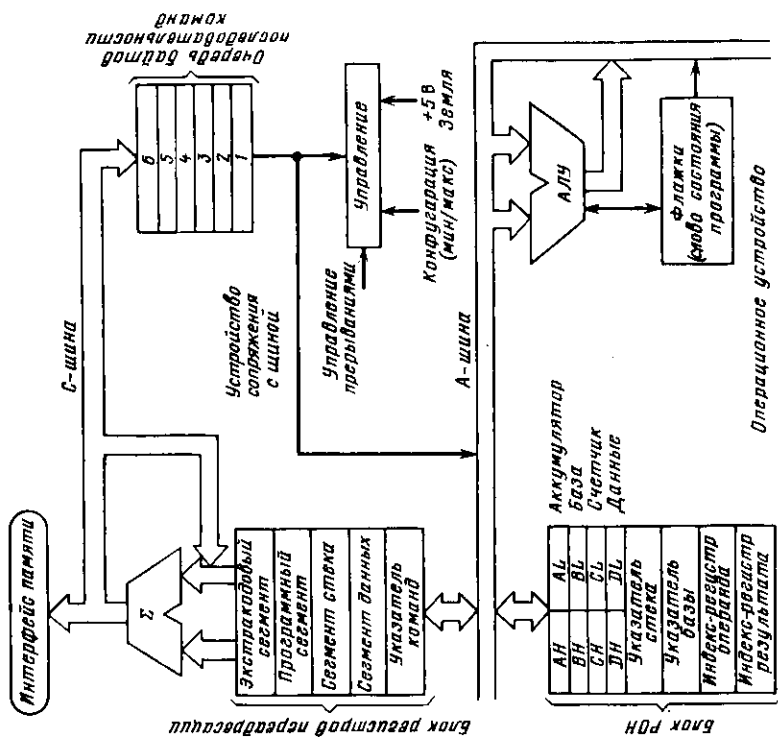


Рис. 9.26. Структура 16-разрядного микропроцессора

размеров МОП-схем, достигнуть большей степени интеграции (около 30 тыс. транзисторов на кристалле $5,5 \times 5,5$ мм) и высокого быстродействия. Задержка на элемент уменьшилась, она имеет тот же порядок, что и в ТТЛ-схемах с диодами Шоттки, имеющими значительно большие размеры и потребляемую мощность.

На МОП БИС указанного типа создан однокристалльный 16-разрядный МП К1810, который по уровню производительности и логической организации не уступает средним моделям малых ЭВМ. Приборы выпускаются в 40-контактном корпусе.

Производительность МП К1810 почти на порядок выше производительности МП К580. Это достигнуто благодаря повышению быстродействия схем (тактовая частота составляет 5–8 МГц) и архитектурным усовершенствованиям. Архитектура МП К1810 имеет следующие особенности:

- выполнение аппаратурными средствами арифметических операций над 8- и 16-разрядными числами со знаком, логических операций с цепочками данных, расширенные возможности работы с отдельными разрядами слова;

- наличие 16-разрядного АЛУ с аппаратурной реализацией умножения и деления;

- регистровая структура с удвоенным по сравнению с К580 числом общих регистров, практически неограниченное число уровней прерывания;

- полная совместимость по системе команд с МП К580 (в том числе работа с 8-разрядными командами последнего) и одновременно наличие новых эффективных 16-разрядных команд;

- сегментная адресация (см. § 8.11), позволяющая прямо адресовать одномогабайтную память (оперативную, дисковую и т. п.), производить динамическое перемещение программ;

- использование языка программирования ПЛ/М-86 с большими возможностями, чем язык ПЛ/М, принятый в МП К580.

Связь между МП, ОП и периферийным устройствами осуществляется с помощью интерфейса *мультишина* (см. гл. 11).

На рис. 9.26 представлена структурная схема МП К1810, в которой имеется независимо управляемое *устройство сопряжения с шиной*, обеспечивающее опережающую выборку команд и формирующее очередь выбранных байт последовательности команд в специальной регистровой стековой памяти (емкость 6 байт), и *операционное устройство*, извлекающее команды из очереди и реализующее предписанные командами операции в 16-разрядном АЛУ.

Устройство сопряжения с шиной имеет блок 16-разрядных регистров переадресации, а операционное устройство — блок 16-разрядных общих регистров.

Такая структура при надлежащем соотношении тактовой частоты МП и длительности цикла памяти позволяет получить эффективное совмещение процессов выборки и исполнения команд. Одному циклу основной памяти (800 нс) соответствуют четыре такта работы МП К1810. При совмещении за время одного цикла основной памяти выполняются две однобайтные команды.

Блок 16-разрядных общих регистров содержит четыре регистра данных: аккумулятор, базовый регистр, «счетчик» и «данные», регистры — указатели стека и базы и индексные регистры операнда и результата. В командах операций над данными могут использоваться все регистры блока общих регистров. К блоку общих регистров относятся также два 8-разрядных регистра признаков (флажков).

9.12. ОСОБЕННОСТИ ПРОЦЕССОРОВ ЭВМ ОБЩЕГО НАЗНАЧЕНИЯ (ЕС ЭВМ)

В машинах общего назначения, к которым относятся ЭВМ ЕС, процессор представляет собой объединенную в один конструктивный модуль (шкаф) совокупность устройств (блоков), выполняющих разнообразные функции, обеспечивающие обработку данных, программное управление вычислительным процессом, взаимодействие устройств машины, обмен информацией с ОП. Основная память не входит в состав процессора, конструктивно выполняется в виде отдельного устройства. Рассмотрим состав блоков (устройств) процессора и выполняемые ими функции на примере ЭВМ ЕС-1045 [102].

Процессор содержит следующие основные блоки (устройства).

Арифметическо-логическое устройство выполняет:

арифметические операции с двоичными числами с фиксированной запятой, шестнадцатиричными (двоично-кодированными) числами с плавающей запятой, десятичными (двоично-кодированными) числами переменной длины; логические операции над логическими кодами фиксированной и переменной длины; операции формирования исполнительных адресов операндов.

Местная память процессора, ячейки которой используются в качестве 16 общих регистров, четырех регистров с плавающей запятой и управляющих регистров (см. гл. 8).

Управляющее устройство (управляющий автомат) с управляющей (частично загружаемой) памятью, хранящей микропрограммы (см. гл. 7). В ЭВМ ЕС-1045 микрокоманда имеет длину 72 разряда, используется горизонтально-вертикальное микропрограммирование. В формате микрокоманды имеется поле, задающее адрес следующей микрокоманды. Два младших разряда этого адреса в общем случае устанавливаются в зависимости от значений оповещающих сигналов, вырабатываемых по результатам проверки определенных условий, задаваемых специальными полями микрокоманды. Управляющая память состоит из постоянной и загружаемой частей, имеющих емкость соответственно 7К и 1К микрокоманд. В постоянной части УП хранятся микропрограммы, управляющие процессором и каналами ввода-вывода. В загружаемую часть УП заносятся с пультавого ленточного накопителя диагностические микропрограммы (см. гл. 12). В каждом машинном такте (120 нс) выполняется очередная микрокоманда и одновременной считывается следующая.

Блок выборки команд производит предварительную (опережающую) выборку команд, их хранение, а также выборку операндов.

Акселератор служит для сокращения времени выполнения некоторых (около 30) команд (умножения, сдвига, пересылки, преобразования форматов и др). Акселератор содержит быстродействующую УП с циклом 60 нс, которая хранит микропрограммы («пикопрограммы») акселератора. Для уменьшения продолжительности выполнения ряда операций в акселераторе используются постоянные программируемые памяти (ППЗУ). При использовании акселератора выборка команд и подготовка операндов производятся микропрограммами процессора. Акселератор повышает производительность процессора на 15–20%.

Блок (система) прерываний (см. § 9.6).

Блок регистрации программных событий облегчает отладку программ, оповещая при помощи прерывания о некоторых событиях, возникающих при выполнении программы. Возможна регистрация следующих программных событий: успешное выполнение команды перехода; выборка команды из заданной области ОП; изменение заданной области ОП; изменение содержимого заданных общих регистров. Программа, засылая соответствующие коды в управляющие регистры 9–11 и единицу в разряд маски регистрации программных событий в ССП, задает подлежащие регистрации события и контролируемые при этом области памяти и общие регистры.

Блок измерения времени формирует коды даты и суточного времени, может по заданию программы измерять промежутки времени и вырабатывать внешнее прерывание по истечении заданного интервала времени.

Блок восстановления и диагностики (см. гл. 12).

Блок прямого управления позволяет осуществлять прямой обмен данными между процессорами в многопроцессорной или многомашинной системе или между процессором и внешним по отношению к ЭВМ оборудованием (см. гл. 15).

Блок памяти ключей защиты служит для предотвращения доступа программ к областям памяти, используемым другими программами (см. гл. 14).

Блок буферной памяти. При обращении процессора к ОП вместе с адресуемым словом из памяти выбирается содержащий это слово блок данных (16 байт) и помещается в буферную память, время обращения к которой примерно в 5 раз меньше времени обращения к ОП. Поскольку велика вероятность, что ближайшие обращения программы произойдут к данным, находящимся в буферной памяти, и эти данные могут быть извлечены из нее значительно быстрее, чем из ОП, применение буферной памяти позволяет значительно повысить производительность процессора (в ЭВМ ЕС-1045 примерно в 2 раза).

Блок динамического преобразования адресов преобразует виртуальные адреса в выполняемой программе в физические (реальные). Благодаря этому программист получает возможность при разработке программ иметь дело не с физической памятью, реально имеющейся в ЭВМ, а с виртуальной, емкость которой (в ЕС ЭВМ 16 Мбайт) ограничена только разрядной сеткой адреса ЭВМ (см. гл. 14).

Глава десятая

ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА ЭВМ: ВНЕШНИЕ ЗУ И УСТРОЙСТВА ВВОДА-ВЫВОДА

10.1. ОБЩИЕ ПОНЯТИЯ О ПЕРИФЕРИЙНЫХ УСТРОЙСТВАХ

В состав современных ЭВМ и систем входят многочисленные и разнообразные по выполняемым функциям, принципам действия и характеристикам *периферийные устройства* (ПУ), которые по их назначению можно разделить на две

группы: 1) *внешние запоминающие устройства*, предназначенные для хранения больших объемов информации, и 2) *устройства ввода-вывода*, обеспечивающие связь машины с внешней средой, в том числе с пользователями, путем ввода в ЭВМ и вывода из машины информации, ее регистрации и отображения.

Операции ввода и вывода определяются относительно ядра ЭВМ (процессора и основной памяти). *Операцией ввода* называется передача информации в ядро ЭВМ из внешней среды (в том числе от пользователей) или из внешних ЗУ. *Операцией вывода* называется передача информации из ядра ЭВМ во внешнюю среду или во внешние запоминающие устройства. Соответственно *устройства ввода* позволяют вводить в машину данные и программы, а также вносить исправления в программы и данные, хранящиеся в памяти ЭВМ. *Устройства вывода* служат для вывода из ЭВМ результатов обработки данных, их регистрации и отображения.

В настоящее время производительность и эффективность использования ЭВМ определяются не только возможностями ее процессора и основной (оперативной) памяти, но и в большей степени составом ее ПУ, их техническими данными и способами организации их совместной работы с центральной частью (ядром) ЭВМ.

Общей характеристикой для всех периферийных устройств является скорость, с которой устройство может принимать или выдавать данные. Большинство периферийных устройств имеют электромеханические узлы, скорость работы которых значительно ниже скорости работы электронных устройств ЭВМ (процессоров и др.). Скорости передачи данных, с которыми работают различные ПУ, отличаются весьма значительно: от нескольких единиц до 10^6 байт/с.

Периферийные устройства различают по реализуемому в них синхронному или асинхронному режиму передачи (приема) данных. При *синхронном режиме* передача данных производится в определенном темпе, который задается рабочей скоростью движения носителя информации, например магнитной ленты. При *асинхронном режиме* передача данных может происходить в свободном темпе с остановом после передачи любого байта.

10.2. ПРИНЦИПЫ ДЕЙСТВИЯ ВНЕШНИХ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Иерархически организуемая память ЭВМ (см. § 4.1) наряду с основной (оперативной) памятью высокого быстродействия (цикл обращения около 1 мкс), но сравнительно небольшой емкости (в больших

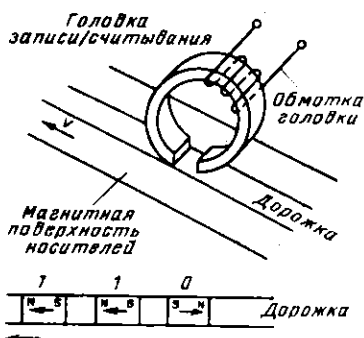


Рис. 10.1. Принцип магнитной записи цифровой информации

ЭВМ — до нескольких мегабайт) содержит внешнюю память, намного медленнее работающую, но способную хранить практически сколь угодно большой объем необходимой для функционирования вычислительной установки информации (данные, программы и др.).

Внешняя память состоит из нескольких внешних запоминающих устройств (ВЗУ), в качестве которых в современных ЭВМ используются *электромеханические* ЗУ с носителем информации в виде движущейся поверхности, покрытой тонким слоем магнитного материала. Внешние ЗУ являются устройствами с произвольным обращением, допускающими многократное считывание информации и запись новой информации на место ранее записанной. Электромеханические ВЗУ компактны, сравнительно дешевы (в пересчете стоимости на 1 бит хранимой информации), могут хранить в одном устройстве (модуле) объемы информации, достигающие сотен миллионов байт. Необходимый объем внешней памяти достигается подсоединением к ЭВМ соответствующего количества ВЗУ.

В вычислительной технике используются электромеханические ВЗУ с носителем информации в виде магнитных лент, дисков и, реже, барабанов.

Принцип магнитной записи цифровой информации поясняет рис. 10.1.

Основу процесса магнитной записи составляет взаимодействие магнитного носителя информации и магнитных головок.

Магнитные головки представляют собой миниатюрные электромагниты, располагаемые у поверхности движущегося магнитного носителя с небольшим зазором при *бесконтактной записи* или без зазора при *контактной записи*. Информация записывается на носителе вдоль дорожки, проходящей под головкой.

Сердечники головок изготавливаются из материала с малой коэрцитивной силой и большим значением индукции насыщения. Наоборот, магнитный материал носителя должен обладать достаточной коэрцитивной силой, чтобы записанная на носителе информация длительно сохранялась и не стиралась под действием внешнего поля. Значение коэрцитивной силы для материала магнитного слоя лент лежит в пределах 12 000—24 000 А/м, а для дисков и барабанов — 24 000—80 000 А/м. Остаточная индукция для лент обычно составляет 0,08—0,15 Тл, а для дисков и барабанов — от 0,15 до 0,6 Тл.

Для запоминания информации используют два противоположных состояния насыщения магнитного материала носителя. При записи

в обмотку магнитной головки посылается соответствующая записываемой информации и используемому методу записи последовательность положительных и отрицательных (или только однополярных) потенциальных сигналов. Каждая смена потенциального сигнала в обмотке головки приводит к изменению полярности намагничивания участка дорожки на магнитном носителе, проходящего под головкой.

Считывание информации производится при прохождении под головкой дорожки носителя с записанной информацией. Образующийся проходящим под головкой намагниченным участком дорожки магнитный поток частично замыкается через сердечник головки, пронизывая ее обмотку. При прохождении под головкой границы между участками с разной полярностью намагничивания потокоцепление обмотки головки меняется и в ней возникают импульсы той или иной полярности, которые в соответствии с используемым методом записи информации воспринимаются как сигналы считанных 1 или 0.

Информация записывается на магнитный носитель на нескольких дорожках, при этом для каждой дорожки носителя имеется своя магнитная головка или перемещающаяся головка обслуживает несколько дорожек.

Основные характеристики ВЗУ

Одной из основных характеристик ВЗУ является *общий объем хранимой информации*, или *емкость ВЗУ*, обычно измеряемая в байтах.

Из-за большого различия быстродействия ОП и ВЗУ обращения к внешней памяти порождают потери производительности ЭВМ. Поэтому быстродействие ВЗУ является показателем не менее, а в ряде случаев даже более важным, чем его емкость.

Обращение к ВЗУ в общем случае предполагает последовательное выполнение двух процессов:

а) доступа к ВЗУ — подведения к головке (головкам) участка носителя, где находится нужная информация или куда информация должна быть записана;

б) считывания и передачи информации из ВЗУ в ОП или передачи информации из ОП в ВЗУ и записи ее на носитель.

Соответственно быстродействие ВЗУ определяется двумя показателями — *средним или максимальным временем доступа* и *скоростью передачи информации (скоростью записи-считывания)*.

На носителе, как правило, информация располагается упорядоченно, поэтому оказывается целесообразным, чтобы при обращении к ВЗУ происходили запись или считывание не отдельного слова или байта, а последовательно расположенного на носителе блока или массива данных. Этим достигается уменьшение влияния времени доступа на временные характеристики процедур обмена информацией между ОП и ВЗУ.

Внешние ЗУ делятся на устройства с прямым и последовательным доступом. В *устройствах с прямым доступом*, к которым относятся устройства с магнитными дисками и барабанами, время доступа (обычно несколько десятков миллисекунд) практически мало зависит от положения носителя относительно головки (головок) в момент инициирования обращения к ВЗУ, что достигается циклическим движением

носителя с большой скоростью относительно головки (головок). В устройствах с *последовательным доступом* (ВЗУ на магнитных лентах) для поиска нужного участка носителя требуется последовательный просмотр записанной на носителе информации, для чего в неблагоприятных случаях расположения головок и актуального участка магнитной ленты может потребоваться несколько минут.

К важным характеристикам ВЗУ следует также отнести достоверность функционирования и относительную стоимость устройства.

Обычно *достоверность работы ВЗУ* оценивают числом правильно воспроизводимых в режиме записи-считывания двоичных знаков на один ошибочный знак. Например, современные ВЗУ на магнитной ленте обеспечивают более 10^9 правильно записанных и считанных двоичных знаков на один ошибочный знак.

Относительная стоимость ВЗУ определяется как отношение стоимости устройства к его емкости.

Плотность записи информации. Основные характеристики ВЗУ (емкость, скорость передачи данных, относительная стоимость, размеры устройства и др.) прямо зависят от плотности записи информации на носитель, поэтому одним из важнейших направлений улучшения характеристик ВЗУ является повышение плотности записи, что представляет собой сложную инженерную проблему, решение которой связано с улучшением конструкции и технологии изготовления основных узлов ВЗУ, в первую очередь носителя и магнитных головок, создания новых методов магнитной записи и способов кодирования записываемой информации, обеспечивающих корректирование ошибок при считывании.

Поверхностная плотность записи информации может быть выражена формулой

$$\delta_s = \delta_l \delta_q,$$

где δ_l — *продольная плотность записи*, равная числу бит, записываемых на единицу длины дорожки; δ_q — *поперечная плотность записи*, равная числу дорожек, приходящихся на единицу длины в направлении, перпендикулярном движению носителя.

Допустимая продольная плотность записи зависит от характеристик магнитного носителя, зазора между носителем и головкой, конструкции головки, способа записи информации и других факторов. Увеличения поперечной плотности записи можно достигнуть уменьшением ширины дорожки и уменьшением расстояния между центрами дорожек. Минимальная ширина дорожки ограничивается технологическими трудностями обработки тонкого сердечника для головок. Для повышения поперечной плотности применяют головки, изготовленные на основе тонкопленочной технологии. При уменьшении расстояния между центрами дорожек увеличиваются перекрестные электромагнитные наводки в головках.

Наибольшую плотность удается получить при контактной записи, когда магнитный носитель непосредственно соприкасается с головкой. Такой способ работы применяется главным образом в устройствах с магнитными лентами и гибкими дисками. В выпускаемых промышленностью устройствах с магнитной лентой достигнута продольная

плотность до 256 бит/мм. Обычно эти устройства выполняются для работы при продольной плотности 8,32 и 63 бит/мм. В современных устройствах с магнитной лентой ширина дорожки составляет 1,09 мм, а поперечная плотность — примерно 0,8 бит/мм.

При контактной записи трение между магнитным носителем и головкой, вызывая их износ, ограничивает допустимую скорость движения носителя относительно головки. Вместе с тем скорость носителя влияет на такие важные характеристики ВЗУ, как время доступа и скорость передачи информации (скорость записи-считывания). С увеличением этой скорости время доступа к ВЗУ уменьшается, а скорость передачи информации увеличивается.

Применение бесконтактной записи с небольшим зазором между головкой и носителем позволяет создавать ВЗУ с более высокой скоростью передачи информации путем значительного увеличения линейной скорости движения носителя (до нескольких десятков метров в секунду).

Бесконтактную запись широко применяют в устройствах с магнитными дисками и барабанами. Для поддержания плотности на высоком уровне при бесконтактной записи стремятся работать с возможным меньшим зазором. Однако этому препятствуют механические неточности изготовления дисков и барабанов (биение поверхности носителя, эксцентриситет), а также температурные деформации.

Зазор удастся значительно уменьшить при использовании *плавающих головок*. В этом случае головку укрепляют в подвижном башмаке, снабженном пружиной, стремящейся прижать его к носителю. Однако в клинообразном воздушном зазоре, образованном башмаком и движущейся поверхностью носителя, возникают аэродинамические силы, препятствующие соприкосновению башмака (головки) с носителем.

Устанавливается подвижное равновесное состояние, при котором зазор между головкой и поверхностью носителя составляет 3–5 мкм, а в устройствах, разработанных в последние годы, — 0,2–1 мкм. Башмак «плывет» около поверхности носителя, следуя за ее биением. При использовании плавающих головок удается работать с продольной плотностью записи 40–160 бит/мм, а в новейших конструкциях дисковых ЗУ, где зазор составляет 0,2–1 мкм, плотность достигает 400–600 бит/мм.

Внешние ЗУ на магнитной ленте. Принцип действия ВЗУ на магнитной ленте поясняет рис. 10.2. В этих устройствах применяют контактный способ записи. Магнитная лента 1 движется только во время записи и считывания информации, в остальное время лента неподвижна.

Блок магнитных головок 2 содержит несколько головок, расположенных по одной линии, перпендикулярной направлению движения ленты.

Каждой головке соответствует расположенная под ней на ленте дорожка записи. При помощи блока головок одновременно записывается поперек ленты двоичный код (обычно код байта).

В ВЗУ на лентах применяют универсальные, но большей частью вдвоенные головки. В первом случае одни и те же головки используют для считывания и для записи, во втором имеются отдельные блоки го-

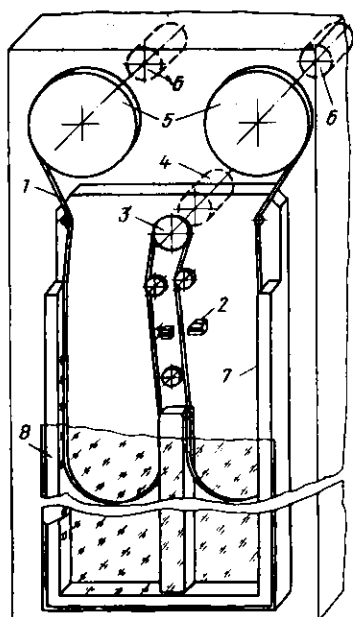


Рис. 10.2. ВЗУ с магнитной лентой (одновальный лентопротяжный механизм)

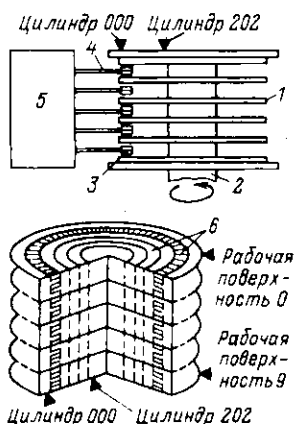


Рис. 10.3. ВЗУ на магнитных дисках:

1 — диск; 2 — вал; 3 — плавающая головка записи-считывания; 4 — пружинный рычаг; 5 — каретка; 6 — дорожка

ловков записи и считывания, расположенные на небольшом расстоянии друг от друга по направлению движения ленты. В этом случае головки считывания используют также для контроля правильности записи информации путем ее считывания сразу после записи.

Лента приводится в движение быстродействующим *старт-стопным лентопротяжным механизмом*. Скорость движения ленты относительно головок должна поддерживаться постоянной, чтобы сохранялись постоянными плотность записи и амплитуда сигналов при считывании информации. Время пуска и останова ленты должно быть минимальным, чтобы можно было работать с короткими промежутками между зонами, в которых записана информация. В противном случае площадь носителя будет использоваться неэффективно. В современных устройствах скорость движения ленты в режимах записи и считывания обычно составляет 2–5 м/с, а время пуска и останова не превышает 5 мс.

Лентопротяжный реверсивный механизм состоит из одного ведущего ролика (вала) 3, приводимого во вращение малоинерционным реверсивным двигателем 4, катушек с лентой 5 с их реверсивными двигателями 6 и вакуумных карманов 7, которые снабжены фотодиодным датчиком 8 положения ленты.

Быстрые пуски, остановки, реверсы, а также протягивание ленты по головке обеспечиваются малоинерционным ведущим роликом. Магнитная лента сцепляется с ведущим роликом только за счет трения между его поверхностью и поверхностью ленты. Для увеличения

трения угол охвата ролика лентой делается достаточно большим и, кроме того, поверхность ролика покрывают материалом с высоким коэффициентом трения. Двигатель ролика управляется системой, поддерживающей при помощи тахометрических датчиков (на рисунке не показаны) постоянную скорость движения ленты.

Во время движения ленты двигателя 6 вращают катушки так, чтобы с одной катушки лента сматывалась, а на другую наматывалась. При пуске и останове лента движется с большими ускорениями. Катушки с лентой имеют сравнительно большой момент инерции, и привод катушек не может в этих условиях поддерживать натяжение ленты в требуемых пределах. Чтобы избежать недопустимых натяжений и разрывов ленты, а также образования больших петель, нарушающих работу лентопротяжного механизма, между быстродействующим стартопным механизмом привода ленты и катушками с их инерционным приводом устраивают буферные хранилища в виде *вакуумных карманов*, в которых создается разрежение, удерживающее в карманах слабо натянутую петлю ленты.

Внешние ЗУ на магнитных дисках. Принцип действия этого устройства показан на рис. 10.3. Устройство содержит пакет тонких (2—2,5 мм) алюминиевых дисков 1, покрытых тонким слоем магнитного материала. Пакет дисков устанавливается на вал 2 и приводится во вращение электродвигателем. Обе поверхности дисков являются рабочими. Обычно только наружные поверхности крайних дисков не используют для хранения информации.

Информация записывается на концентрических дорожках 6. Дорожки одного и того же диаметра на разных дисках образуют как бы концентричные цилиндры. Понятие *цилиндра* является важным для организации данных на диске.

Чтобы избежать износа носителя и головок, применяют бесконтактную запись и используют плавающие головки. Головки записи-считывания 3 (по одной головке на каждую рабочую поверхность диска) перемещаются в зазорах между дисками. Головки установлены на пружинных рычагах 4, укрепленных в каретке 5. Электрический привод с большой точностью преобразует цифровой код номера цилиндра в соответствующее перемещение каретки с рычагами вдоль радиуса дисков. В результате перемещения головки попадают на нужный цилиндр. В плавающем башмаке укреплена одна головка считывания-записи.

Дисковые ЗУ различного конструктивного исполнения являются наиболее распространенными и совершенными ВЗУ с прямым доступом. По сравнению с магнитным барабаном устройство с дисками в том же физическом объеме имеет во много раз большую поверхность носителя информации.

В настоящее время в вычислительной технике применяются следующие типы дисковых ЗУ: а) со сменными пакетами жестких дисков; б) со сменными дисковыми модулями; в) с несменными пакетами жестких дисков большей емкости; г) с гибкими дисками (дискетами) [20].

Запоминающие устройства со сменными пакетами жестких дисков — наиболее распространенный тип ЗУ прямого доступа в ЭВМ

средней и большей производительности. Пакеты содержат стандартные диски диаметром 356 мм, изготавливаемые из алюминиевого сплава и покрытые ферролаком. Характерными устройствами этого типа дисковых ЗУ являются устройства емкостью 7,29 и 100 Мбайт, содержащие соответственно 6, 11 и 12 дисков и работающие с продольной (бит/мм)/поперечной (дорожек/мм) плотностью соответственно 40/4, 80/4 и 160/8. В первых двух устройствах установка головок на заданный цилиндр осуществляется разомкнутой системой позиционирования, точность работы которой зависит от точности изготовления узлов, фиксирующих положения блока магнитных головок, и не позволяет работать с поперечной плотностью свыше 4—5 дорожек/мм. В дисковом устройстве емкостью 100 Мбайт поперечная плотность увеличена до 8 дорожек/мм благодаря применению замкнутой системы позиционирования блока головок с использованием одной поверхности диска в пакете (сервоповерхности) с нанесенными на нее серводорожками и соответственно сервоголовки для выработки сигналов отклонения блока головок от заданного положения (заданного цилиндра).

Запоминающие устройства со сменными дисковыми модулями («винчестерские диски»). С увеличением поперечной и продольной плотностей записи усложняется проблема обеспечения совместимости сменных пакетов дисков [на разных устройствах размеры блока головок (и самих головок) могут отличаться в пределах допусков]. Это послужило причиной разработки ЗУ со сменными дисковыми модулями, каждый из которых представляет собой съемный герметизированный контейнер, содержащий пакет дисков вместе с кареткой с головками (в некоторых моделях применены тонкопленочные головки толщиной несколько десятков микрон). При такой конструкции продольную плотность записи удается повысить до 250—425 бит/мм, а поперечную — до 12—25 дорожек/мм и получить сменные модули емкостью 35 Мбайт с двумя дисками (три информационных поверхности), 75 Мбайт с четырьмя дисками (шесть информационных поверхностей) и 570 Мбайт (12 информационных поверхностей).

Запоминающие устройства с несменными пакетами жестких дисков. В них реализуется наибольшая емкость дисковых устройств (в ряде конструкций превышает 1 Гбайт). При использовании в вычислительной установке дискового ЗУ со стационарным пакетом возникает проблема дублирования информации записываемой на диски. Имеется тенденция снабжать устройства со стационарными дисками встроенным в него ленточным ЗУ специально для дублирования информации.

Запоминающие устройства на дисках с фиксированными головками. Благодаря исключению потерь времени на перемещение головок достигается высокое быстродействие (среднее время доступа 2,5—5 мс) при сравнительно небольшой емкости устройства. Однодисковые устройства этого типа емкостью 0,5—2 Мбайт используются в составе малых и микро-ЭВМ, работающих в системах реального времени, а многодисковые (емкостью 5—10 Мбайт) — для организации виртуальной памяти в высокопроизводительных ЭВМ.

Запоминающие устройства на гибких магнитных дисках — дешевые и малогабаритные ЗУ, широко используются в качестве внешней памяти

ти малых и микро-ЭВМ, специализированной памяти для хранения микропрограмм и диагностических тестов пультов управления (сервисных адаптеров и сервисных процессоров) ЭВМ общего назначения, запоминающих устройств в составе оборудования различных сложных по выполняемым функциям, в том числе интеллектуальных, терминалов (графические дисплеи и построители, бесперфокарточные устройства подготовки и ввода данных и т. п.).

Размеры гибкого диска (*дискеты*) в настоящее время стандартизированы. Пластмассовый (лавсановый) диск толщиной 0,12 мм и диаметром 198 мм, покрытый ферролаком, заключается в квадратную пластмассовую кассету со стороной квадрата 204 мм и толщиной 1,6 мм.

На гибком диске информация записывается на 77 концентрических дорожках. Дорожка разбивается на 26 секторов, в каждый из которых можно записать 128 байт. В начале дорожки записывается индекс начала дорожки (начала оборота диска). Емкость поверхности размеченного на секторы диска составляет 256 Кбайт. Дорожка 00 содержит описание информации на диске, дорожки 74 и 75 являются запасными, а 76 не используется. Дискеты поставляются пользователю размеченными на секторы. Частота вращения диска составляет 90 об/с, среднее время доступа 1,5 с.

Помимо рассмотренного типа существует много других разновидностей ЗУ на гибких дисках. В различных конструкциях ЗУ этого типа емкость лежит в пределах 0,256–3 Мбайт, частота вращения 90–300 об/с, среднее время доступа 1,5–0,4 с, продольная плотность записи 10–128 бит/мм.

В ЗУ на гибких дисках используется контактная запись. Хотя для уменьшения трения поверхность диска покрывается специальным пластиком, тем не менее из-за износа носителя регламентируется общее число проходов носителя под головкой, которое для современных конструкций составляет обычно до $5 \cdot 10^6$ проходов.

Совместимость внешних ЗУ является одной из важнейших эксплуатационных характеристик. Это свойство позволяет использовать катушки лент, сменные пакеты дисков, дискеты с информацией, записанной на одном ВЗУ и на других аналогичных ВЗУ, работающих в составе различных ЭВМ и систем. Благодаря совместимости ВЗУ имеется возможность обмена информацией, записанной на магнитных носителях, между различными ВЦ.

Для совместимости ВЗУ необходимо выполнение ряда конструктивных и технологических требований. Эти требования сводятся к стандартизации (в ряде случаев в международном масштабе) размеров катушек, пакетов дисков, дискет, материала и технологии изготовления носителей информации, размеров и характеристик магнитных головок, лентопротяжных механизмов, способов и плотности записи, размещения информации на носителе, способов контроля информации.

В табл. 10.1 приведены основные характеристики некоторых выпускаемых серийно отечественных электромеханических ЗУ Единой системы ЭВМ.

Разработка электромеханических ЗУ с высокими техническими характеристиками является исключительно трудной задачей, требующей

Таблица 10.1. Характеристики внешних ЗУ ЕС ЭВМ

Параметр	ЗУ на магнитных лентах		ЗУ на сменных магнитных дисках		
	ЕС-5017	ЕС-5025	ЕС-5050	ЕС-5061	ЕС-5066
Емкость, Мбайт	20	20 и 40	7,25	29	100
Продольная плотность записи, бит/мм	8 и 32	32 и 63	29—44	60—90	159
Способ записи информации	БВН-1	БВН-1 и фазокодированный	ЧМ	ЧМ	ЧМ, МФМ
Среднее время доступа, мс	—	—	90	75	32,5
Скорость передачи данных, Кбайт/с	64	64 и 126	156	312	806

решения комплекса сложных вопросов в области техники магнитной записи, новых магнитных и конструктивных материалов, электромеханики, кодирования информации, электроники, прецизионных технологий изготовления ответственных узлов. Несмотря на это, за последние 15 лет благодаря новым идеям и технологиям удалось достигнуть значительного улучшения характеристик электромеханических ЗУ. Например, плотность записи в дисковых ЗУ увеличилась на два порядка.

Можно ожидать дальнейшего улучшения характеристик электромеханических ЗУ.

В настоящее время интенсивно ведутся работы и по созданию ВЗУ большей емкости на новых принципах: на полупроводниковых приборах с зарядовыми связями, цилиндрических доменах, голографических, оптических и магнитооптических устройствах. Следует ожидать, что в итоге эти работы позволят создать ВЗУ, существенно превосходящие существующие электромеханические ЗУ по емкости, быстродействию и надежности. Однако прогнозы показывают, что, по крайней мере, на ближайшее десятилетие электромеханические ЗУ с движущимся магнитным носителем останутся основным типом устройств внешней памяти ЭВМ.

10.3. МЕТОДЫ ЗАПИСИ ИНФОРМАЦИИ НА МАГНИТНЫЙ НОСИТЕЛЬ

В электромеханических ЗУ используется ряд методов записи информации на носитель. Во всех применяемых методах запись производится потенциальными сигналами, поступающими в головку записи. При выборе метода записи учитываются следующие факторы: возможность получения высокой плотности записи, помехоустойчивость метода, обладает метод свойством самосинхронизации при считывании или требует внешней синхронизации, необходимо предварительное стирание ранее записанной информации или допустима запись новой информации без предварительного придания участку носителя исход-

ного магнитного состояния, сложность электронных схем обработки сигналов, считываемых с носителя.

Удешевление электронных схем делает возможным применение более эффективных, хотя и более сложных методов кодирования записываемой информации и процедур обработки считываемых сигналов, в том числе процедур контроля и коррекции ошибок при считывании информации.

Метод записи без возврата к нулю с переключением потока по единицам (БВН-1). При этом методе запись производится без возврата носителя к исходному состоянию (размагниченному состоянию или состоянию насыщения определенного знака).

Диаграмма для метода записи БВН-1 приведена на рис. 10.4, а. При записи 1 ток в обмотке записи изменяет направление, и носитель соответственно переходит из состояния насыщения одного знака в состояние насыщения другого знака. При записи 0 направление тока в обмотке и состояние носителя не меняются. Достоинством метода является возможность записи на носитель новой информации без стирания предыдущей.

При считывании 1 в обмотке головки индуцируются импульсы (разнополярные), а при считывании 0 сигнал с дорожки не поступает. Поэтому для распознавания информации при считывании необходимы синхросигналы, задающие границы каждого такта записи. Признаком 1 является совпадение информационного импульса, считанного с дорожки на ленте, с синхросигналом, а признаком 0 — наличие только синхросигнала (отсутствие информационного импульса). Синхросигналы могут записываться на отдельную дорожку при записи информации на носитель и восприниматься специальной головкой в процессе считывания информации. Если такой возможности нет, то в такте считывания применяют генератор синхросигналов. Однако поскольку скорость движения носителя непостоянна, то в этом случае должна быть заранее установлена максимально допустимая при записи длина последовательности нулей.

Стандартом предусмотрено применение метода БВН-1 при записи информации на магнитных лентах с плотностью 8 и 32 бит/мм. На лентах производится многодорожечная запись, и в каждой позиции (строке) записывается поперек ленты байт с дополнительным разрядом контроля по четности (см. гл. 12). Поэтому при считывании любой строки хотя бы с одной дорожки поступает сигнал 1, чем достигается самосинхронизация считываемой информации.

Метод записи с фазовым кодированием (ФК) или фазовой модуляцией. Диаграмма для этого метода приведена на рис. 10.4, б. На границе каждого такта записи происходит смена направления тока в записывающей головке и, следовательно, смена магнитных состояний носителя. Полярность тока изменяется в одном направлении при записи 0 (например, от отрицательной полярности к положительной) и в противоположном направлении при записи 1. Происходит как бы изменение фазы тока записи. Логическая схема тракта записи анализирует значение следующего записываемого двоичного знака. Если должен быть записан тот же знак, что и в предыдущем такте, то в середине такта изменяется направление тока записи в головке. Если

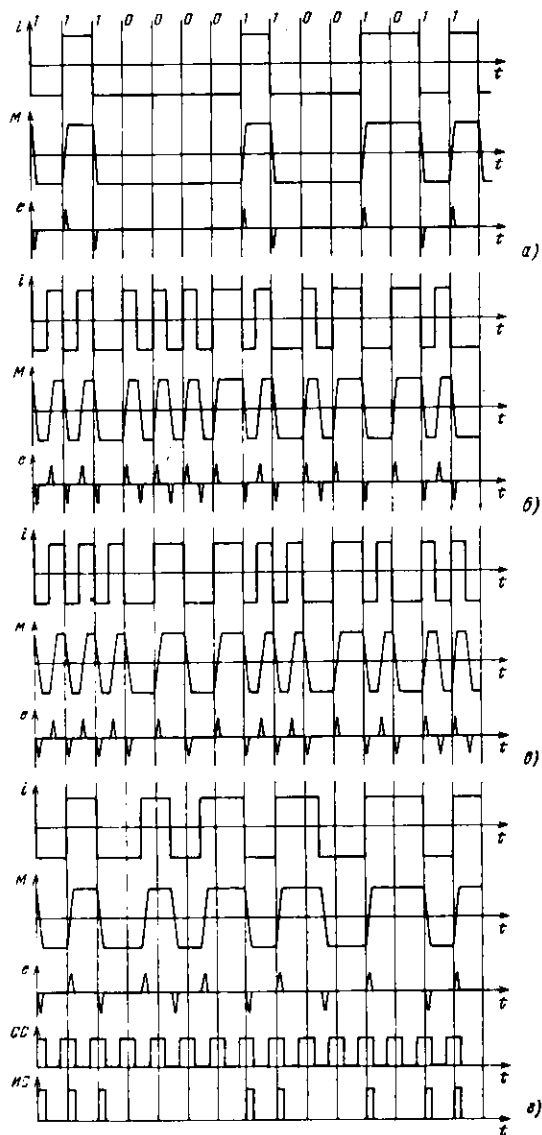


Рис. 10.4. Методы магнитной записи цифровой информации:

a — запись без возврата к нулю с переключением потока по единицам (БВН-1); *б* — запись с фазовым кодированием; *в* — запись с частотной модуляцией; *г* — запись с модифицированной фазовой модуляцией; *i* — ток в головке записи; *M* — намагниченность магнитного носителя; *e* — ЭДС считывания; *CC* — синхросигнал; *ИС* — сформированный информационный сигнал

должен быть записан другой знак, изменение направления тока посредине такта не производится. Так как при записи последовательности одинаковых знаков в серединах тактов производится дополнительное переключение направления тока, то частота изменения тока записи в этом случае увеличивается в 2 раза по сравнению с частотой изменения тока при записи последовательности неодинаковых знаков.

При считывании 1 и 0 распознаются по полярности импульса ЭДС в головке считывания в первом полутакте.

Несмотря на увеличение частоты переключения тока записи, метод ФК по сравнению с методом БВН-1 позволяет работать с большей плотностью записи, так как на фазовые соотношения сигналов оказывают малое влияние помехи, возникающие из-за наложения полей, создаваемых соседними участками носителя. Метод ФК является самосинхронизирующимся (в первом полутакте всегда имеется импульс той или иной полярности), не требующим предварительного стирания ранее записанной информации.

В соответствии со стандартом метод ФК применяется при записи информации на магнитные ленты с плотностью 63 бит/мм. В некоторых ЭВМ предусматривается возможность работы устройств с магнитной лентой при плотностях записи информации 32 и 63 бит/мм и соответственно с использованием методов БВН-1 и ФК, причем переключение с одного режима при записи (или считывании) на другой производится автоматически программным путем.

Метод записи с частотной модуляцией (ЧМ). Метод поясняется на рис. 10.4, в. Ток записи изменяет направление на границе каждого такта записи и, кроме того, посредине такта при записи 1. Запись производится под управлением вырабатываемой в ЗУ серии синхроимпульсов, определяющих границы тактов записи. При записи 1 в середине такта поступает дополнительный импульс, вызывающий дополнительное переключение тока в головке.

Таким образом, при записи 1 частота переключения тока записи вдвое больше, чем при записи 0. Поэтому этот метод называют также двухчастотным. При считывании 1 соответствует наличие импульса произвольной полярности во втором полутакте, а при считывании 0 — его отсутствие.

Метод ЧМ не требует перед записью предварительного стирания информации и является самосинхронизирующимся (при считывании в первом полутакте всегда присутствует импульс той или иной полярности), что особенно важно для ЗУ на дисках, где применяется односторонняя запись.

Применение метода записи ЧМ предусмотрено международным стандартом для ЗУ со сменными пакетами дисков.

Метод записи с модифицированной фазовой модуляцией МФМ (трехчастотный) (рис. 10.4, г) получил в последнее время распространение главным образом в ЗУ со сменными дисками большей емкости (100—300 Мбайт). Метод обеспечивает самосинхронизацию сигналов считывания и позволяет получить большую плотность записи, чем другие методы. Записью управляет серия синхросигналов, задающих ее такты. Переключение тока в головке происходит при записи 1 в начале такта записи, а при записи 0 — посредине такта, если только сле-

дующий записываемый знак не 1. В противном случае при записи 0 переключение тока блокируется.

При записи последовательности одинаковых двоичных знаков 0 или 1 частота переключений тока записи совпадает с частотой синхросигналов. При записи комбинации 10 или 01 интервал между переключениями тока увеличивается до полутора периодов синхросигналов, а при записи 101 — до двух. Таким образом, в последовательности переключений тока записи и соответственно в последовательности смены полярности намагничивания участков дорожки носителя имеются три частоты. При считывании каждое изменение полярности намагничивания участков дорожки индуцирует в обмотке головки импульс той или иной полярности. Этот импульс соответствует считанной 1, если он совпадает по времени с синхросигналом, и 0, если с отрицательным.

10.4. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ НА МАГНИТНЫХ ЛЕНТАХ И ДИСКАХ

Представление информации на магнитных лентах. В ВЗУ на магнитных лентах применяется главным образом последовательно-параллельное размещение информации. Строка информации, записанная поперек ленты, соответствует одному машинному слогу (байту). Машинное слово занимает на ленте несколько строк (рис. 10.5).

Блоки информации. т. е. данные, располагаемые на носителе компактно (без промежутков) и записываемые и считываемые одной командой ЭВМ, помещаются в отдельные зоны. Зоны разделяются между собой промежутками, называемыми *межзонными расстояниями*. Эти расстояния выбираются таким образом, чтобы пуск и останов ленты не вызывали потери информации.

При записи на магнитную ленту в ряде случаев используют зоны (блоки) фиксированной длины. Если в зоны фиксированной длины приходится записывать разное количество информации, то использование поверхности носителя будет неполным.

Более эффективными являются зоны переменной длины, размеры которых устанавливаются программой в процессе решения задачи. Следует отметить, что запись очень короткими зонами приводит к плохому использованию поверхности магнитного носителя, так как значительную часть площади носителя будут занимать промежутки между зонами.

На ленту помимо данных записывается также служебная информация: признаки начала и конца зоны, коды для контроля правильности записи и считывания информации.

Применяется контроль правильности считанной информации по строке (*поперечный контроль*) и правильности в зоне (*продольный контроль*). Поперечный контроль обычно осуществляется по нечетности суммы всех 1 разрядов строки, что дает возможность обнаружить в строке при считывании единичную ошибку. Для контрольного разряда нечетности выделяется специальная дорожка. При продольном контроле по четности в конце каждой зоны записывается отдельная

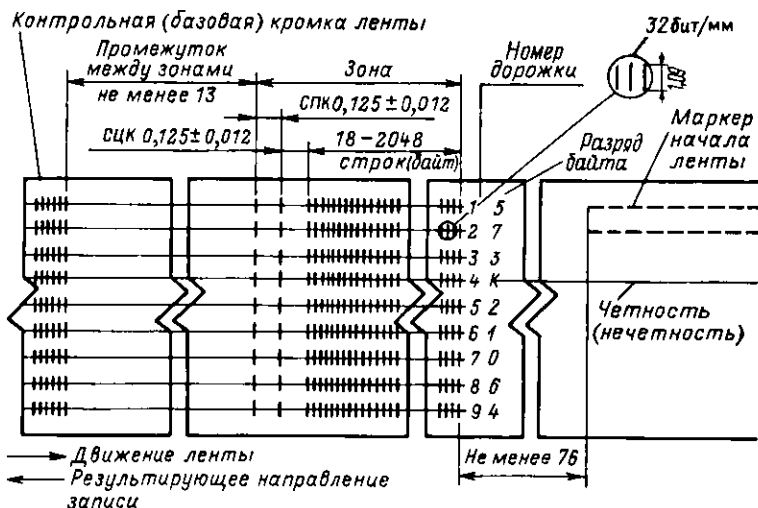


Рис. 10.5. Размещение информации на магнитной ленте при плотности записи 32 бит/мм (размеры указаны в миллиметрах):

СПК — строка циклического контроля; СПК — строка продольного контроля

контрольная строка (СПК) — разряды которой дополняют сумму единиц по соответствующим дорожкам по четности.

В ряде случаев, например при записи информации на магнитные ленты, предназначенные для обмена между различными потребителями, применяется дополнительный, так называемый *циклический контроль*. Строка *циклического контроля* (СПК) размещается в конце зоны перед СПК. Циклический контроль дает возможность обнаруживать и исправлять при считывании одиночные и групповые ошибки по одной дорожке (см. [2, 34]). Строка циклического контроля и СПК формируются в процессе записи устройством управления ленточного ЗУ.

Чтобы различать начало или конец зоны, применяют различные способы. Одним из способов является использование специальных кодовых комбинаций, записанных на ленты. При другом способе используются синхрои́мпульсы, которые записываются на отдельной дорожке одновременно с информацией. В промежутках между зонами синхрои́мпульсы отсутствуют, и поэтому факт их поступления при считывании информации служит признаком начала зоны. В устройствах, использующих поперечный контроль по нечетности, в каждой строке записана по крайней мере одна 1. В таких устройствах нет обычно специальной дорожки для синхрои́мпульсов, вместо этого используется схема ИЛИ для информационных сигналов (включая и сигнал с контрольной дорожки). Граница между зонами здесь устанавливается по отсутствию каких-либо сигналов с носителя в течение определенного времени.

При подходе магнитной головки к физическому краю ленты (началу или концу) необходимо вырабатывать сигналы для автоматического останова ленты. С этой целью на ленту наносят тем или иным способом маркеры начала и конца ленты¹. Эти маркеры устанавливаются на некотором расстоянии от края, которое выбирается так, чтобы осуществлялось надежное торможение ленты.

В связи с важным значением совместимости ВЗУ на магнитных лентах разработан международный стандарт формата размещения информации на них (рис. 10.5). Стандарт предписывает использование ленты шириной 12,7 мм с девятью дорожками для записи информации и контрольного разряда поперечной нечетности. Толщина ленты 48 мкм, длина ее на катушке около 750 м. Стандарт предусматривает запись с плотностью 8 и 32 бит/мм по методу БВН-1 или 63 бит/мм по методу ФК. На рис. 10.5 показан формат размещения информации на ленте при плотности записи 32 бит/мм. Строка циклического контроля записывается в конце зоны после последнего байта данных с промежутком четыре строки. Затем с тем же промежутком записывается СПК. При плотностях 8 и 63 бит/мм формат размещения данных на ленте аналогичен показанному на рис. 10.5, но при этом СЦК отсутствует.

В современных ЭВМ информация на носителях, в том числе и на магнитных лентах, хранится, как правило, в виде файлов. *Файлом* называется совокупность записей, объединенных по некоторому общему смысловому признаку (или нескольким признакам). Например, в файле сведений о заработной плате сотрудников предприятия за текущий месяц каждая запись содержит информацию об одном из работников предприятия.

Запись, входящая в файл, может занимать целиком блок информации на носителе, либо несколько записей могут быть объединены в одном блоке. В последнем случае говорят о так называемой *сблокированной записи*.

Файлы могут иметь различную длину. В связи с этим вводится понятие тома. *Том* — это стандартный для данного устройства типоразмер носителя информации. В ВЗУ на лентах томом является катушка магнитной ленты; другими видами томов являются пакет магнитных дисков, колода перфокарт и др.

Один том может содержать несколько файлов (*многофайловый том*); один файл может располагаться в нескольких томах (*многотомный файл*); наконец, один файл может занимать целиком один том (*однотомный файл*).

Для того чтобы правильно обрабатывать файлы различной длины и структуры, а также для автоматического исправления ошибок оператора, если он попытается установить на механизм не ту катушку, которую следует обрабатывать, на носитель помимо основной информации записывают служебную — так называемые метки. Метки представляют собой специального вида записи, описывающие характеристики

¹ Обычно маркерами служат наклеенные в начале и конце ленты полоски фольги, отражающие свет на маркерный фотодиод.

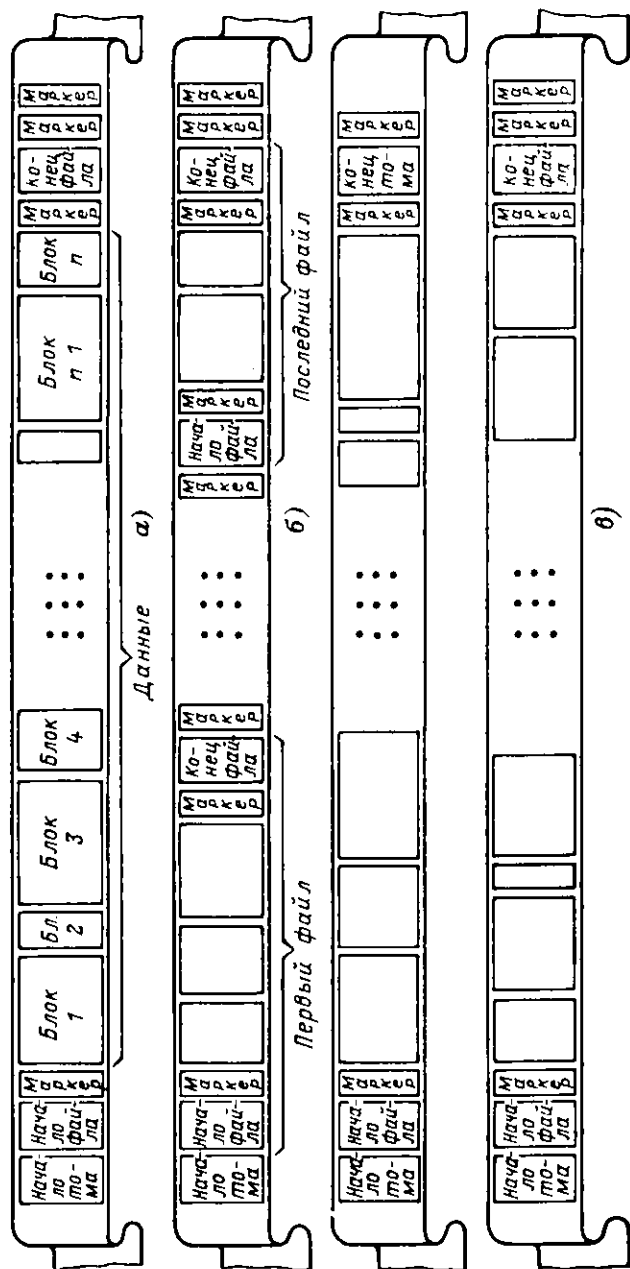


Рис. 10.6. Структура файлов на магнитных лентах:

а — файл, занимающий одну катушку магнитной ленты (однотомный файл.); б — расположение нескольких файлов на одной катушке ленты (многофайловый том); в — файл, занимающий несколько катушек ленты (многоотомный файл)

томов и файлов. Кроме того, на ленту заносятся специальные записи для отделения друг от друга различных видов информации — разделительные маркеры.

На рис. 10.6 показано в упрощенной форме представление на ленте файлов различного вида (однотомный файл, многофайловый том, многотомный файл).

В начале каждого тома располагается метка *Начало тома*, описывающая данную катушку ленты (номер тома в данном вычислительном центре, имя владельца и т. п.).

Далее следует метка *Начало файла*, которая указывает наименование файла, номер тома, в котором размещается файл, порядковый номер тома в данном многотомном файле, номер модификации файла в процессе его обработки или подновления, дата создания файла и срок его хранения, секретность информации и др.

Окончание файла отмечается меткой *Конец файла*, которая подобна начальной метке, но, кроме того, обычно указывает количество записей с основной информацией в данном файле.

Служебные записи (метки) отделяются от основных записей разделительными маркерами. После окончания файла на катушке обычно записываются подряд два маркера. В многотомном файле метка *Конец файла* записывается в конце последнего тома, а каждый из предшествующих томов заканчивается меткой *Конец тома* и одним (а не двумя) маркером.

В ВЗУ на магнитных лентах реализуются операции считывания-записи, стирания блоков информации, передвижения (вперед и назад) на один блок (одну зону) или до следующего разделительного маркера, перемотки ленты в исходное состояние. Используя эти операции, а также данные, хранимые в служебных метках, программа может осуществлять поиск и обработку записей внутри файла и поиск и идентификацию томов и файлов.

Представление информации на магнитных дисках. Запись информации на диск производится обычно последовательным кодом на концентрические дорожки на поверхности диска. Группа машинных слов (или байтов), записанная последовательно вдоль дорожки, образует блок информации, отделяемый от следующего блока на этой дорожке промежутком. В современных ВЗУ на магнитных дисках используются как блоки фиксированной длины — секторы (для относительно простых и дешевых устройств), так и блоки переменной длины.

Расположение файла на одном цилиндре позволяет производить поиск и обработку записей, входящих в этот файл, без радиального перемещения головок, существенно сокращая время доступа к данным.

В качестве примера организации данных на диске рассмотрим типичное расположение информации на дорожке для устройства с блоками переменной длины (рис. 10.7). Начало дорожки распознается с помощью метки начала оборота, которая физически выполняется обычно в виде прорези на специальном секторном диске.

В начале каждой дорожки помещается служебный блок, называемый *Собственным адресом*. Собственный адрес определяет номер цилиндра и номер магнитной головки внутри цилиндра для данной дорожки.

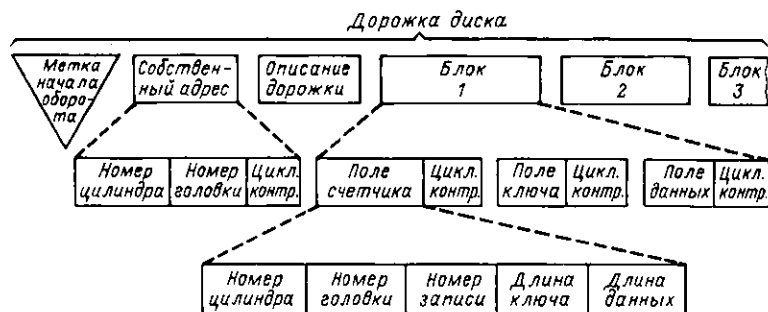


Рис. 10.7. Представление информации на магнитных дисках

Вслед за собственным адресом размещается служебный блок *Описание дорожки*, описывающий степень заполнения дорожки данными, адрес запасной дорожки, если данная дорожка повреждена, и др.

Далее располагаются блоки, содержащие рабочие записи. Эти блоки состоят из трех основных полей: *поля счетчика*, *поля ключа* и *поля данных*.

Поле счетчика содержит служебную информацию, необходимую для обработки записей, хранимых в блоке. В этом поле указываются номера цилиндра и головки, которые в случае исправности дорожки совпадают с соответствующими номерами в поле собственного адреса. Если же дорожка неисправна, то указывается номер соответствующей альтернативной (запасной) дорожки. На альтернативной дорожке записываются номера цилиндра и головки основной дорожки.

Поле счетчика указывает также номер записи и длину остальных двух полей — поля ключа и поля данных.

Поле ключа предназначается для информации, идентифицирующей данную запись (например, имя файла или какие-либо другие смысловые признаки). Поле данных хранит информацию, непосредственно образующую саму запись или группу записей. Поле ключа или поле данных может отсутствовать, при этом в поле счетчика указывается, что длина соответствующего поля равна нулю.

В конце служебных полей и в конце полей данных записываются контрольные циклические коды, позволяющие при считывании проверять правильность записанной информации.

Рассмотренная выше структура информации на диске позволяет строить файлы различных видов, при этом часть рабочих записей используется для представления меток тома, файла, маркеров и других идентификаторов, а в остальные записи заносятся запасаемые на диске данные.

В современных ЭВМ предусматривают широкий набор операций (приказов) в ВЗУ на магнитных дисках: установку цилиндра, головки; поиск информации по собственному адресу, по номеру цилиндра и головки, номеру записи, ключу, данным [эта группа операций позволяет найти запись, в которой указанный приказом признак (со-

ответствующий адрес, ключ, данные и т. п.) удовлетворяет определенному условию (равен, больше или меньше по сравнению с некоторым кодом из ячеек ОП)]; чтение, запись различных полей на диске (собственного адреса, ключа, поля данных и др.).

10.5. ОСНОВНЫЕ ТИПЫ УСТРОЙСТВ ВВОДА-ВЫВОДА ИНФОРМАЦИИ

Устройства ввода позволяют вводить в машину данные и программы, а также вносить исправления в программы и данные, хранящиеся в памяти ЭВМ. Устройства вывода служат для вывода из ЭВМ результатов обработки данных, в том числе для их регистрации и отображения.

Известные в настоящее время устройства ввода информации можно разделить на две группы: ручные и автоматические (рис. 10.8). В группу устройств ручного ввода входят пульты управления ЭВМ, электрифицированные пишущие машинки, дисплей и др. Отличительной особенностью этих устройств является наличие клавиатуры, с помощью которой осуществляется ввод. Устройства ручного ввода по-



Рис. 10.8. Классификация устройств ввода информации

звolyют вводить информацию со сравнительно низкой скоростью [не более 10 символов (знаков)/с].

В группу автоматических устройств входят *устройства для считывания информации с промежуточного носителя и устройства непосредственного ввода*.

К устройствам ввода с промежуточного носителя информации относятся устройства считывания информации с перфокарт или перфолент, а также с магнитных лент.

Информация наносится на перфокарты и перфоленты с помощью пробивок отверстий (перфораций). Устройства ввода с перфолент и перфокарт сравнительно просты и имеют относительно высокие скорости ввода информации. Однако они требуют предварительной ручной подготовки данных — перфорирования перфокарт и перфолент.

Перфокарты и перфоленты как носители информации имеют свои преимущества и недостатки. Перфокарты более долговечны, позволяют легко сменить часть информации (путем простой замены некоторых карт), удобны для хранения, но они дороже перфолент. Перфокартное оборудование, используемое при подготовке перфокарт, вводе и выводе информации, более громоздко и более сложно в эксплуатации и имеет стоимость, большую стоимости оборудования для перфолент.

Перфоленты быстрее изнашиваются, на них трудно произвести замену части информации (как правило, для этого требуется перебивка перфоленты). Следует также учитывать, что перфоленты широко используются в аппаратуре передачи данных по линиям связи, работающей в комплексе с ЭВМ.

Для подготовки данных и ввода в ЭВМ используются также устройства с магнитной лентой. Подобное устройство имеет буферное ЗУ, в которое с помощью клавиатуры заносится информация. Затем информация из буферного ЗУ записывается на непрерывно движущуюся магнитную ленту. За один цикл подготовки на ленту записывается одна зона (блок). Для ввода данных в ЭВМ кассета с лентой устанавливается на лентопротяжный механизм обычного ВЗУ.

К автоматическим устройствам непосредственного ввода информации относятся устройства, считывающие информацию со специальных бланков, с печатного текста и с графиков. Хотя подобные устройства достаточно сложны и дороги, их применение весьма целесообразно, так как позволяет исключить ручную подготовку данных (перфорация и т. п.), требующую больших затрат труда и нередко сопровождающуюся ошибками. В настоящее время ведутся интенсивные разработки устройств ввода с печатного текста и устройств ввода информации с голоса. К автоматическим устройствам непосредственного ввода информации относятся также преобразователи аналоговых сигналов в цифровые, устройства приема информации с линией связи и др.

Устройства вывода информации можно разделить на следующие группы (рис. 10.9):

1) устройства вывода цифровой информации на промежуточный носитель;

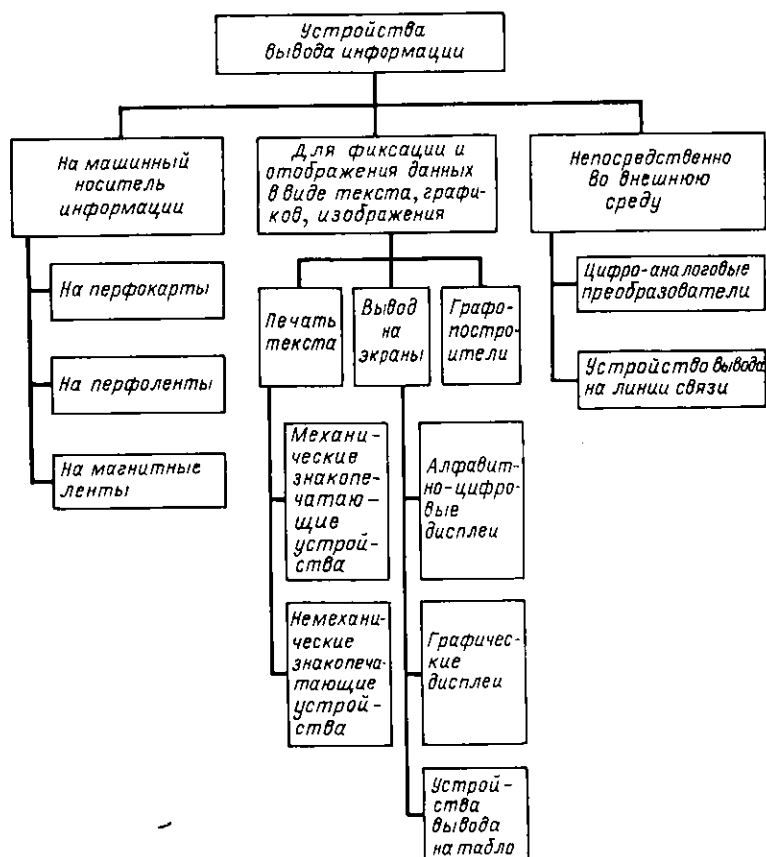


Рис. 10.9. Классификация устройств вывода информации

2) устройства, фиксирующие результаты обработки в виде текста, графиков и изображений, — различного рода печатающие устройства, устройства вывода информации на разного рода экраны, графопостроители;

3) устройства вывода информации во внешнюю среду (цифро-аналоговые преобразователи и устройства выдачи данных на линии связи).

В первую группу входят устройства вывода информации на перфокарты и перфоленты путем пробивки соответствующих комбинаций отверстий. Такой способ вывода информации удобен для последующего ввода ее в ЭВМ и неудобен для человека, когда ему необходимо прочитать полученные на ЭВМ данные.

Выводные устройства второй группы либо печатают поступающую из ЭВМ алфавитно-цифровую информацию (текст) на бумаге — *печатающие устройства*, либо отображают ее на экранах в виде текста, изображений и графиков — *дисплеи*, либо рисуют графики и чертежи на бумаге — *графопостроители и чертежные машины*.

Сами алфавитно-цифровые печатающие устройства можно разделить на механические и немеханические. В состав механических знакопечатающих устройств входят электрифицированные пишущие машинки, телетайпы, печатающие устройства с шаровыми головками, печатающие устройства барабанного, цепного и циклоидного типов.

В немеханических устройствах используются различные физические методы регистрации знаковой информации — ксерографический, электрохимический, термический, искровой и др. На основе этих методов удается значительно увеличить скорость вывода данных по сравнению с электромеханическими печатающими устройствами. Несмотря на высокое быстродействие, устройства подобного типа употребляются сравнительно редко из-за сложности их эксплуатации, высокой стоимости и низкой надежности.

В данной главе далее описаны принципы действия только наиболее распространенных устройств ввода-вывода (перфолентные и перфокартные устройства, механические печатающие устройства и дисплеи). С аналого-цифровыми и цифро-аналоговыми преобразователями читатель может ознакомиться в [37].

10.6. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ НА ПЕРФОЛЕНТАХ И ПЕРФОКАРТАХ

Информация, представляемая в двоичном алфавите, наносится на перфоленты и перфокарты путем пробивки отверстий. Наличие отверстия в определенной позиции означает 1, отсутствие — 0.

Перфолента. В зависимости от ширины ленты на ней имеется от пяти до восьми кодовых дорожек (позиций) для записи информации. Информация записывается строками, расположенными поперек ленты. В строке путем пробивки круглых отверстий записывается один *слог*, соответствующий коду символа (цифры, буквы, знаки препинания и т. д.).

В вычислительной технике используют обычно 5-, 6-, но большей частью 7- или 8-позиционную перфоленту, позволяющую представлять большое число различных символов. Стандартная процедура предусматривает ввод с перфоленты данных, представленных в 7-разрядном коде ЕС ЭВМ (КОИ-7) (рис. 10.10) с последующим преобразованием во внутримашинный код ДКОИ при обработке данных и без преобразования при обмене.

Помимо кодовых отверстий на перфоленте в каждой строке пробивается также круглое отверстие меньшего диаметра, используемое для создания синхроимпульсов и иногда для перемещения перфоленты. Эти отверстия называются *ведущими* или *синхронизирующими*.

Перфокарты. Применяются главным образом 80-колонные перфокарты. Колонки нумеруются слева направо от 1 до 80. Карта имеет 12

Латинский регистр							Русский регистр						
б ₇	0	0	0	0	1	1	1	1	0	0	0	1	1
б ₆	0	0	1	1	0	0	1	1	0	0	1	0	1
б ₅	0	1	0	1	0	1	0	1	0	1	0	1	0

б ₇	б ₆	б ₅	б ₄	б ₃	б ₂	б ₁	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	р	ПУС	AP1	пер. стр.	0	ю	п	ю
1	SOH	DC1	!	1	A	Q	q	НЗ	(CY1)	!	1	а	я	А
2	STX	DC2	"	2	B	R	r	НТ	(CY2)	"	2	б	р	Б
3	ETX	DC3	#	3	C	S	s	КТ	(CY3)	#	3	ц	с	Ц
4	EOT	STOP	␣	4	D	T	t	КТ	СТП	␣	4	д	т	Д
5	ENQ	NAK	%	5	E	U	e	КТМ	НЕТ	%	5	е	у	Е
6	ACK	SYN	&	6	F	V	f	У	ДА	&	6	ф	ж	Ф
7	BEL	ETB	'	7	G	W	w	ЗВ	КБ	'	7	г	в	Г
8	BS	CAN	(	8	H	X	x	ВШ	АН	(	8	х	ь	Х
9	HT	EM	)	9	I	Y	y	ГТ	КН	)	9	ц	ы	И
10	LF	SUB	*	10	J	Z	z	ПС	ЗМ	*	10	з	й	З
11	VT	ESC	+	11	K	[	k	БТ	АР2	+	11	к	ш	К
12	FF	IS4(FS)	<	12	L	\	l	ПФ	РИ4(РФ)	<	12	л	э	Л
13	CR	IS3(GS)	=	13	M	]	m	ВК	РИ3(РГ)	=	13	м	щ	М
14	SO	IS2(RS)	>	14	N	^	n	РУС	РИ2(РЗ)	>	14	н	ч	Н
15	SI	IS1(US)	/	15	O	_	o	ЛАТ	РИ1(РЗ)	/	15	о	ъ	О

1 Допускается (логическое НЕ) 2 Допускается прописная буква в 7-разрядный код КОИ-7

Рис. 10.10. 7-разрядный код КОИ-7

Зональная группа														
12	11	0	12	11	0	12	11	0	12	11	0	12	11	0
лифтовая группа														
8	-	0	12	11 <td>0<td>12</td><td>11<td>0<td>12</td><td>11<td>0<td>12</td><td>11<td>0</td></td></td></td></td></td></td>	0 <td>12</td> <td>11<td>0<td>12</td><td>11<td>0<td>12</td><td>11<td>0</td></td></td></td></td></td>	12	11 <td>0<td>12</td><td>11<td>0<td>12</td><td>11<td>0</td></td></td></td></td>	0 <td>12</td> <td>11<td>0<td>12</td><td>11<td>0</td></td></td></td>	12	11 <td>0<td>12</td><td>11<td>0</td></td></td>	0 <td>12</td> <td>11<td>0</td></td>	12	11 <td>0</td>	0
1	A	/	1	a	j	-	б	и	н	з	с	у	1	д
2	B	S	2	b	k	s	3	н	т	с	у	2	с	и
3	C	T	3	c	l	t	щ	к	т	с	у	3	д	1
4	D	U	4	d	m	u	э	д	2	8	д	2	9	д
5	E	V	5	e	n	v	щ	г	т	д	0	5	п	с
6	F	W	6	f	o	w	ч	д	0	6	в	ш	к	б
7	G	X	7	g	p	x	ь	3	5	д	0	7	а	п
8	H	Y	8	h	q	y	ш	д	2	3	а	н	д	0
9	I	Z	9	i	r	z		д	1	3	к	н	д	0
8-2	[	]	:	:	к		б	д	1	4	д	1	8	д
8-3	.	×	#		л		ц	б	т	д	1	5	п	с
8-4	<	*	%	@	ф	м	т	д	п	ф	р	ф	д	1
8-5	(	)	'	'	з	н			в	к	р	г	к	т
8-6	+	;	=	=			ф	р	у	с	р	з	д	а
	!	^	?	?	ц	п	в	г	л	а	т	р	з	з

¹ Допускается (логическое НЕ)

² Допускается прописная буква б

Рис. 10.11. Перфокартный код КПК-12

строка (позиций), которым присвоены номера сверху вниз: 12, 11, 0, 1, ..., 9. Прямоугольные отверстия перфорируются на пересечениях колонок и строк.

Информация наносится на перфокарту в перфокартном коде КПК-12 ЕС ЭВМ (рис. 10.11), при этом код каждого символа занимает одну колонку. Например, буква П находится в зональной группе 12-11 и цифровой группе 9-8-4. Следовательно, при занесении в колонку буквы П отверстия перфорируются на 12, 11, 9, 8, 4-й строках.

При использовании кода КПК-12 длина физической записи на перфокарте не превышает 80 байт. При вводе информация, нанесенная на перфокарту, преобразуется в код ДКОИ и в таком виде помещается в память ЭВМ.

Предусмотрен также другой способ представления данных на перфокарте, при котором данные перфорируются в произвольном коде. В каждой колонке располагаются два 6-разрядных байта произвольного кода (первый на строках 12, 11, 0, 1, 2, 3, а второй на строках 4, 5, 6, 7, 8, 9). При передаче данных в память принимается, что разряды 0 и 1 байт имеют нулевое значение. При таком представлении данных на перфокарте длина физической записи на перфокарте может достигать 160 байт.

10.7. УСТРОЙСТВА ВВОДА ИНФОРМАЦИИ С ПЕРФОЛЕНТ И ПЕРФОКАРТ

В устройствах ввода информации с перфоленты (они называются также считывающими устройствами) считывание информации осуществляется с помощью фотоэлектрических или емкостных датчиков, реагирующих на наличие отверстий. Эти устройства производят считывание информации последовательно строка за строкой, т. е. символ за символом (на каждой строке располагается один символ или байт).

В фотоэлектрическом считывающем устройстве перфолента проходит над считывающей головкой с несколькими фоточувствительными элементами, каждый из которых расположен против соответствующей дорожки на перфоленте. Над перфолентой располагается источник света. При совпадении отверстия на перфоленте с отверстием в фотоголовке свет через световод попадает на фотодиод, в результате чего возникает электрический импульс. В емкостном считывающем устройстве перфолента проходит между пластинами конденсаторов, которые расположены против каждой дорожки на перфоленте. Каждый конденсатор включен в мостовую схему, которая вырабатывает напряжение разбаланса при изменении емкости конденсатора, вызванном перемещением между его пластинами отверстия на данной дорожке перфоленты. Сигналы с фотодиодов или емкостных мостов после необходимого усиления и формирования поступают на вход блока управления считывающим устройством.

Считывающие устройства с перфоленты выполняются в различных конструктивных вариантах: без кассет (для перфоленты), с одной подающей кассетой, с приемной и подающей кассетами.

В состав ЕС ЭВМ входит устройство ввода с перфоленты ЕС-6022 с фотосчитывающим аппаратом FS-1501 (ЧССР). Устройство позволяет производить ввод данных с 5, 6, 7 или 8-дорожечных перфолент со скоростью 1500 строк (знаков/с) в режиме непрерывного ввода. Возможна работа в стартстопном режиме.

Устройство ввода с перфоленты ЕС-6022 может выполнять несколько операций в соответствии с поступающими кодами, называемыми приказами. Модификации приказа «*Читать*» позволяют производить считывание с перфоленты информации с преобразованием и без преобразования ее в машинный код ДКОИ, с определением момента окончания считывания машиной, в которую вводится информация, или самим устройством ввода при появлении на перфоленте символа «Конец записи». Выполняется также приказ запроса состояния (см. гл. 11).

В устройствах ввода информации с перфокарт считывание производится при помощи фотоэлектронных или емкостных чувствительных элементов. Обычно перфокарты подаются узкой стороной. В этом случае информация считывается с перфокарт поочередно при помощи 12 считывающих элементов, расположенных над каждой позицией перфокарты.

В ЕС ЭВМ входят устройства ввода с перфокарт ЕС-6012, ЕС-6013 и ЕС-6019. Считывание информации осуществляется фотоэлектрическим способом.

Информация, представленная на перфокартах в коде КПК-12, преобразуется в блоке управления устройством ввода и выдается в машинном коде ДКОИ. Устройства отличаются скоростями ввода и емкостями подающих и приемных карманов. Скорость и емкость карманов для ЕС-6012 соответственно составляют 500 карт/мин и 1000 карт, а для ЕС-6013 – 1200 карт/мин и 2500 карт. В устройстве ЕС-6019 (скорость 1200 карт/мин) использование вакуумной системы подачи перфокарт способствует уменьшению износа последних.

Устройство ввода информации с перфокарт выполняет приказы: «*Прочитать с преобразованием в ДКОИ с укладкой перфокарты в заданный приемный карман*», «*Прочитать без преобразования с укладкой в заданный карман*», а также приказы запроса состояния.

10.8. УСТРОЙСТВА ВЫВОДА ИНФОРМАЦИИ НА ПЕРФОЛЕНТЫ И ПЕРФОКАРТЫ

Устройства вывода информации производят пробивку отверстий на перфокартах или перфолентах. Скорость перфорации из-за инерционности движущихся частей ограничена. На перфоленте одновременно пробиваются синхроотверстие и все необходимые кодовые отверстия на данной строке. Для пробивки кода символа блок управления должен включить электромагнит пробивки синхроотверстия и электромагниты, соответствующие тем дорожкам перфоленты, где должны быть пробиты кодовые отверстия.

В ЕС входит устройство вывода на перфоленты ЕС-7022, укомплектованное перфорирующим механизмом ПЛ-150. На устройстве

можно перфорируют 5, 6, 7 и 8-дорожечные перфоленты длиной до 285 м. Устройство выполняет приказы записи и запроса состояния. По приказу «Запись» производится вывод (перфорация) информации. По приказу «Запись М» выполняется перфорация информации с пробивкой в конце последовательности символа «Конец записи», установленного на переключателях пульта управления устройством. Перфоленточные станции для ввода-вывода ЕС-7902 и ЕС-7903 обеспечивают скорость перфорации ленты соответственно 100 и 150 знаков/с.

Аппараты вывода информации на перфокарту состоят из механизмов пробивки, подачи, транспортировки и укладки перфокарт.

В ЕС ЭВМ имеются устройства вывода на перфокарты ЕС-7010 и ЕС-7012. Данные выводятся на карты построчно (попозиционно), для чего используются 80 пуансонов, управляемых электромагнитами. Скорость перфорации в устройстве ЕС-7010 составляет до 100, а в ЕС-7012 до 250 карт/мин.

В блоках управления механизмом вывода на перфокарты имеется свое буферное ЗУ с емкостью, соответствующей максимальному количеству данных, которые можно разместить на перфокарте. Информация, поступающая из ЭВМ в коде ДКОИ, преобразуется в код КПК-12, а затем записывается в ячейки буферного ЗУ в порядке поступления. Запись в буферное ЗУ может производиться и без преобразования. Затем информация из буферного ЗУ выводится на перфокарту.

Устройства вывода на перфокарты выполняют приказы «Записать с преобразованием и подать», «Записать без преобразования и подать», а также приказы запросов состояния. При выполнении приказа «Записать с преобразованием и подать» информация, поступающая для вывода в коде ДКОИ, преобразуется и наносится на карту в коде КПК-12. По приказу «Записать без преобразования и подать» выводимая информация без преобразования наносится на перфокарту по два 6-разрядных байта в колонке.

10.9. ПЕЧАТАЮЩИЕ УСТРОЙСТВА

Печатающие устройства позволяют получать в виде документов результаты обработки данных в удобной для восприятия человеком форме. В настоящее время существует много различных типов печатающих устройств, предназначенных для этой цели. Они различаются принципами работы, быстродействием и стоимостью.

Наиболее распространенными являются электромеханические печатающие устройства, которые по принципу своей работы могут быть разделены на устройства последовательной (посимвольной) и построчной печати. Последние являются более быстродействующими, хотя и более сложными.

Из устройств последовательной печати наиболее распространены электрифицированные пишущие машинки и матричные (знакосинтезирующие) устройства.

Электрифицированная пишущая машинка (например, устройство ЕС-7077) одновременно является устройством ввода (с клавиатуры) и вывода. Вводимые в ЭВМ и выводимые из нее данные обычно печат-

таются на бумаге соответственно красным и черным цветом. Скорость печати до 10 знаков/с.

В *знакосинтезирующих устройствах* используется матрица печатающих элементов (точек, отрезков прямых), из которых при печати образуется контур печатаемого символа (рис. 10.12). Матричные устройства последовательной печати работают со скоростью 100–180 знаков/с.

Из ударных электромеханических устройств построочной печати наибольшее распространение получили устройства барабанного и цепного типа, в которых используется динамический принцип печати.

Барабанное устройство построочной печати (рис. 10.13). Устройство содержит набор одинаковых печатающих колес с нанесенными на них литерами, собранных на одной оси в виде барабана, вращающегося с постоянной скоростью. Число колес равно числу знаков в строке. Регистрация знаков производится без остановки барабана (печать «на лету»).

Для каждой печатной позиции имеется своя отдельная печатающая система, состоящая из электромагнита, управляемого им молоточка 1 и колеса с полным набором знаков 2. Между бумагой 3 и печатающими колесами проходит красящая лента 4. В момент прохождения нужного знака под молоточком последний ударяет по бумаге и ленте, прижимая их к знаку на печатающем колесе. За каждый оборот барабана печатается строка. Включение и выключение тока в обмотках электромагнитов производится электронным блоком управления.

На одной оси с печатающими колесами закреплен ротор генератора кода 5 с постоянным магнитом 6. Магнит, проходя вблизи

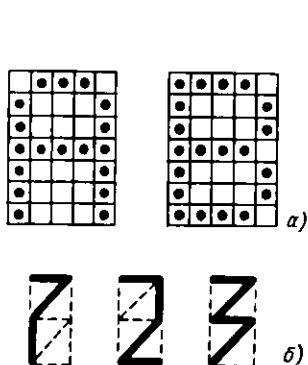


Рис. 10.12. Изображение символов в знакосинтезирующих (матричных) печатающих устройствах с элементами в виде точек (а) и отрезков прямых (б)

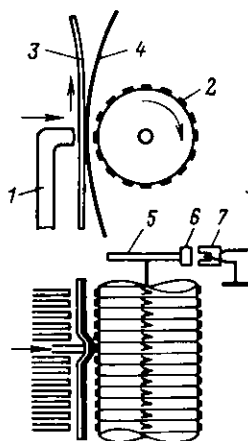


Рис. 10.13. Принцип действия построочного печатающего барабанного устройства

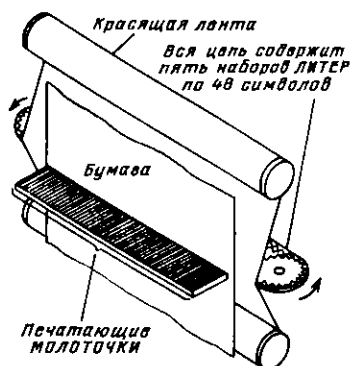


Рис. 10.14. Принцип действия цепного устройства построчной печати

П-образных сердечников катушек неподвижного статора датчика 7, наводит в их обмотках напряжение, из которого формируются стандартные импульсы, поступающие в блок управления. Блок управления подсчитывает число импульсов и определяет код знака, который в данный момент находится под печатающими молоточками.

Механизм для перемещения бумаги управляется специальным электромагнитом, служащим для включения муфты сцепления, которая передает движение транспортному валу. Вал, вращаясь, перемещает бумагу за счет трения или с помощью краевой перфорации, в которую входят зубцы валика.

Для управления построчно печатающими аппаратами требуются значительно более сложные электронные устройства, чем для управления печатающими аппаратами последовательного действия. Как правило, они имеют свое буферное ЗУ для хранения информации о печатаемой строке.

Устройства типов ЕС-7030 (с механизмом АЦПУ-128-5) и ЕС-7032 (с механизмом АЦАУ-128-6) имеют скорость печати 600–900 строк/мин. В строке размещается 128 знаков. Алфавит устройств (набор знаков на каждом колесе) состоит из 84 знаков. Устройства печати выполняют приказы «Записать с пропуском после печати», «Записать с прогоном после печати», «Записать без пропуска», «Пропуск», «Прогон» и приказы запроса состояния.

Цепное устройство построчной печати (рис. 10.14). Цепь, содержащая литеры символов, непрерывно движется с постоянной скоростью вдоль строки, на которой производится печать данных.

Для каждой печатной позиции строки имеется печатающий молоточек, управляемый электромагнитом.

В буферной памяти блока управления находятся переданные из ЭВМ коды знаков, подлежащих печати для всех печатных позиций строки. Совокупность этих кодов поступает на регистр строки. Для каждой печатной позиции электронная схема сравнивает код знака, подлежащего печати, с вырабатываемым в печатающем механизме кодом литеры, находящейся в данный момент против соответствующей печатной позиции. При совпадении кодов срабатывает электромагнит, приводящий в движение печатающий молоточек этой позиции, и нужный знак печатается. Проверка кодов знаков на совпадение с кодами литер производится каждый раз, когда к первой позиции строки подходит новая литера. После завершения печати строки производится перемещение бумаги и начинается печать следующей строки и т. д.

Устройство допускает изменение алфавита путем установки в цепь других литер. Если объем используемого алфавита знаков невелик, то в цепь можно установить несколько одинаковых комплектов литер и таким путем увеличить скорость печати.

Устройства типов ЕС-7037 и ЕС-7039 имеют скорость печати соответственно 1000 и 1200 строк/мин при числе знаков в строке 132 и 160.

10.10. ДИСПЛЕИ

Использование ЭВМ в автоматизированных системах управления, различных информационно-вычислительных системах, системах коллективного пользования (см. гл. 13) требует их укомплектования удобными средствами связи человека с машиной. Одним из таких средств является устройство ввода-вывода с электронно-лучевой трубкой (ЭЛТ), называемое *дисплеем*. В зависимости от типа дисплея на экран может выводиться как алфавитно-цифровая, так и графическая информация.

Оператор с помощью клавиатуры и *светового пера (фотоселектора)* может ввести в ЭВМ нужную информацию, а также вызвать любую информацию, хранящуюся в памяти машины, при необходимости откорректировать ее и снова отправить в машину для дальнейшей обработки.

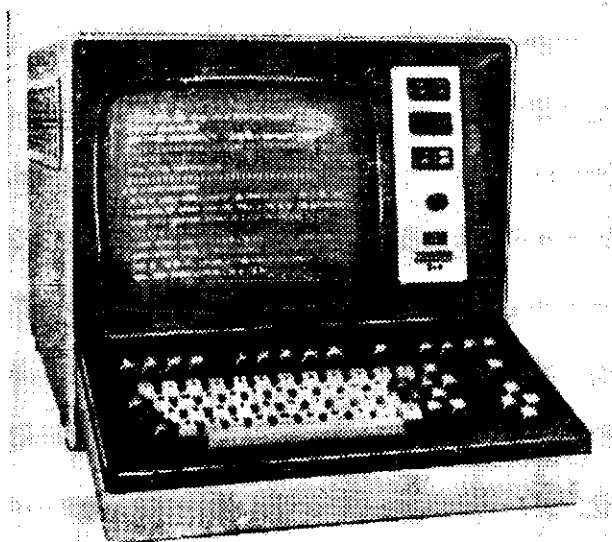


Рис. 10.15. Алфавитно-цифровой дисплей ЕС-7168 («Видеотон-340», ВНР)

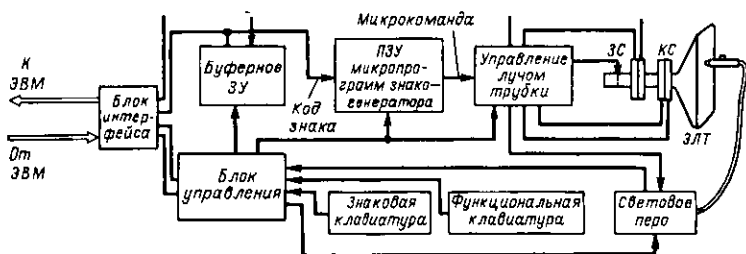


Рис. 10.16. Структурная схема дисплея

В настоящее время дисплей является наиболее эффективным средством ввода-вывода при осуществлении диалогового взаимодействия ЭВМ и пользователя.

При наличии дисплеев общение человека с машиной упрощается, отпадает необходимость набивать перфоленты или перфокарты с запросами, вводить их в вычислительную машину, печатать полученные от машины ответы, набивать исправления, вводить эти исправления в машину и т. д. На экран алфавитно-цифрового дисплея в зависимости от его размера и методов знакогенерирования обычно выводится от 128 до 4000 знаков.

На рис. 10.15 в качестве примера показан внешний вид алфавитно-цифрового дисплея ЕС-7168 («Видеотон-340», ВНР).

Структурная схема дисплея приведена на рис. 10.16. Буферное ЗУ дисплея предназначено для хранения 8-разрядных кодов знаков. Емкость ЗУ равна максимальному числу выводимых на экран знаков. Постоянное ЗУ знакогенератора хранит микропрограммы управления изображением на экране для всех знаков используемого алфавита. Рабочая часть экрана ЭЛТ разбивается на строки, строка — на знаковые места. Каждому знаковому месту на экране соответствует ячейка в буферном ЗУ.

Блок управления лучом ЭЛТ под воздействием микрокоманд, поступающих из ПЗУ, вырабатывает аналоговые сигналы, обеспечивающие перемещение и модуляцию луча, при которых на экране высвечивается рисунок заданного знака.

При выводе информации на экран по инициативе ЭВМ текст, сформированный в виде последовательности кодов знаков, передается из ЭВМ в буферное ЗУ. Коды знаков в ЗУ располагаются в той последовательности, в какой знаки должны размещаться на экране.

После записи кодов в буферное ЗУ в блок управления поступает приказ «Начать отображение». Дисплей переходит в автономный режим работы и начинает отображение текста, при этом в порядке размещения кодов в ЗУ производится обработка кодов знаков. Под воздействием сигналов блока управления код очередного знака поступает в ПЗУ микропрограмм знакогенератора. Знакогенератор воспринимает код как команду, в ответ на которую выдает последовательность микрокоманд управления изображением заданного знака.

Рис. 10.17. Функциональный способ формирования знаков

В настоящее время наиболее распространенными являются функциональный и растровый способы формирования знаков на экранах дисплеев.

Знак изображается в пределах предусмотренного знакового места. Знаковое место имеет свою невидимую сетку, по узлам и линиям которой формируется изображение знака.

При функциональном способе формирования знаков перемещение луча осуществляется двумя электромагнитными отклоняющими системами. Одна из них — знаковая система (ЗС) — перемещает пятно в пределах одного знакового места, другая — координатная система (КС) — после завершения изображения некоторого знака перемещает погашенное пятно в исходную точку соседнего знакового места.

При этом способе управление дисплеем должно обеспечивать перемещение пятна луча с постоянной скоростью по контуру изображаемого знака (рис. 10.17) и его модуляцию в нужные моменты времени. Здесь столбцы знакового места I—IV используются для изображения знака, а V — для формирования промежутка между соседними знаками. На рис. 10.17 цифрами 1—9 отмечена траектория луча при изображении буквы Б.

При растровом способе формирования знаков перемещение луча по экрану не зависит от изображаемых знаков и осуществляется всегда по одной и той же сетке. Растровый дисплей выполняется по той же структурной схеме, что и функциональный, но только с одной отклоняющей КС.

При растровом способе на изображение строки текста затрачивается, например, 16 телевизионных линий, из них семь отводится для собственно знаков (рис. 10.18), а следующие девять — для промежутка между строками текста. Пятно луча движется слева направо по всей рабочей части экрана сначала по линии Л0, затем по Л1 и т. д. Схема синхронизации блока управления дисплеем вырабатывает сигналы линий Л0, Л1, ..., Л6 и сигналы тактов Т0, Т1, ..., Т7 формирования эле-

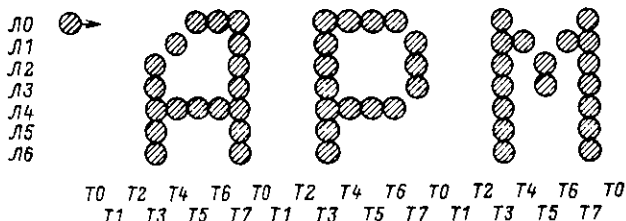
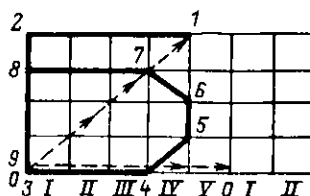


Рис. 10.18. Растровый способ формирования знаков

ментов знаков. Такты T_0 , T_1 , T_2 формируют промежуток между знаками, T_3 — T_7 участвуют в изображении знака. При движении пятна луча по первой линии (L_0) модуляцией луча формируются только верхние элементы всех знаков данной текстовой строки, при движении по второй линии (L_1) — вторые элементы и т. д.

Таким образом, для формирования на линии элементов знаков нужно иметь сведения, на какой линии находится пятно луча (сигналы L_0 , L_1 , ..., L_7), какой знак изображается (код знака) и на каком столбце знакового места находится пятно (сигналы T_3 , T_4 , ..., T_7).

Микропрограммы модуляции луча для всех знаков хранятся в ПЗУ. В процессе изображения текста ПЗУ выдает только сигналы модуляции, соответствующие знаку и положению луча на сетке раstra.

Растровый способ применен, в частности, в дисплее ЕС-7168 («Видеотон-340») производства ВНР (см. рис. 10.15). Этот дисплей имеет экран размером 28 см по диагонали. На экран выводится 16 текстовых строк, каждая на 80 знаков. Дисплей имеет алфавитно-цифровую и функциональную клавиатуры с алфавитом из 96 знаков.

Функциональный способ формирования знаков применен, например, в дисплее «Экран-М» и в дисплейных системах ЕС-7906.

Дисплей «Экран-М» позволяет выводить на экран 20 строк текста по 80 знаков каждая. Устройство работает с алфавитом из 93 знаков. Система ЕС-7906 имеет групповое устройство управления ЕС-7566, к которому может быть подключена группа дисплеев ЕС-7066 (до 16 шт.). Буферная память емкостью 3840 байт размещена в групповом устройстве управления и является общей для всех дисплеев. Поэтому при подключении четырех дисплеев на каждый из них отводится 12 строк по 80 знаков, а при подключении 16 дисплеев — 6 строк по 40 знаков. Для получения немерцающего изображения во всех дисплеях на телевизионных ЭЛТ блок управления обеспечивает воспроизведение (регенерацию) изображения на экране 50 раз/с.

В дисплеях для корректировки информации, вызванной из памяти, обычно используется маркер, который перемещается оператором по экрану с помощью специальной клавиши. Для стирания какого-либо знака на экране оператор совмещает маркер с этим знаком. В устройстве управления дисплея координаты положения маркера декодируются и определяется адрес знака, подлежащего стиранию в буферном ЗУ. Затем оператор нажимает на клавишу «Стирание». Информация в этой ячейке ЗУ стирается и исчезает с экрана ЭЛТ. Нажимая соответствующую клавишу, оператор может записать по выбранному адресу любой другой знак. Скорректированная и записанная с клавиатуры информация из буферного ЗУ может быть передана в ОП машины. Эта передача производится при нажатии оператором специальной клавиши «Возврат».

Для редактирования информации помимо клавиатуры в дисплеях используется также *световое перо*. На кончике его устанавливается элемент, чувствительный к свету (например, фотодиод или миниатюрный фотоумножитель). При совмещении кончика пера со знаком, отображенным на экране, в чувствительном элементе возникает импульс в момент чтения этого знака из буферного ЗУ и его подсвета. Тем

самым определяется адрес ячейки, хранящей этот знак в ЗУ, что позволяет изменить информацию в ячейке по данному адресу.

Наряду с алфавитно-цифровыми дисплеями существуют графические, отображающие графическую информацию и имеющие средства (реализованные с помощью микропроцессоров), позволяющие оператору задавать и вводить в память машины и корректировать изображение на экране. В этих устройствах реализуется режим, в котором луч ЭЛТ перемещается за кончиком светового пера. Это перемещение осуществляется с помощью специальных следящих схем так, чтобы чувствительный элемент карандаша был все время засвечен лучом ЭЛТ. Применение в дисплеях светового пера значительно облегчает связь человека с машиной при автоматизации проектно-конструкторских работ с получением графических изображений на экране.

Глава одиннадцатая

ПРИНЦИПЫ ОРГАНИЗАЦИИ СИСТЕМ ВВОДА-ВЫВОДА. ИНТЕРФЕЙСЫ ЭВМ И МИКРОПРОЦЕССОРОВ

11.1. ПРОБЛЕМЫ ОРГАНИЗАЦИИ СИСТЕМ ВВОДА-ВЫВОДА

Вычислительная машина содержит помимо процессора (процессоров) и основной памяти, образующих ее ядро, многочисленные и разнообразные по выполняемым функциям и принципам действия *периферийные устройства* (ПУ), предназначенные для хранения больших объемов информации (*внешние запоминающие устройства*) и для ввода в ЭВМ и вывода из нее информации, в том числе для ее регистрации и отображения (*устройства ввода-вывода*).

Передача информации с периферийного устройства в ядро ЭВМ (память или процессор) называется *операцией ввода*, а передача из ядра ЭВМ в периферийное устройство — *операцией вывода*.

Производительность и эффективность использования ЭВМ определяется не только возможностями ее процессора и характеристиками основной памяти, но в очень большой степени составом ее ПУ, их техническими данными и способом организации их совместной работы с ядром (процессором и основной памятью) ЭВМ.

Связь устройств ЭВМ друг с другом осуществляется с помощью сопряжений, которые в вычислительной технике называются интерфейсами.

Интерфейс представляет собой совокупность линий и шин, сигналов, электронных схем и алгоритмов, предназначенную для осуществления обмена информацией между устройствами. От характеристик интерфейсов во многом зависят производительность и надежность вычислительной машины.

При разработке систем ввода-вывода должны быть решены следующие проблемы. Во-первых, должна быть обеспечена возможность реализации машин с переменным составом оборудования (*машин с переменной конфигурацией*), в первую очередь с различным набором периферийных устройств, с тем чтобы пользователь мог выбирать состав оборудования (конфигурацию) машины в соответствии с ее назначением, легко дополнять машину новыми устройствами.

Во-вторых, для эффективного и высокопроизводительного использования оборудования ЭВМ должны реализовываться параллельная во времени работа процессора над программой и выполнение периферийными устройствами процедур ввода-вывода.

В-третьих, необходимо упростить для пользователя и стандартизировать программирование операций ввода-вывода, обеспечить независимость программирования ввода-вывода от особенностей того или иного периферийного устройства.

В-четвертых, необходимо обеспечить автоматическое распознавание и реакцию ядра ЭВМ на многообразие ситуаций, возникающих в ПУ (готовность устройства, отсутствие носителя, различные нарушения нормальной работы и др.).

Особенно актуально решение этих проблем для машин средней и большой производительности, содержащих большее число разнообразных периферийных устройств.

Отметим основные пути решения указанных проблем.

Модульность. Средства современной вычислительной техники проектируются на основе модульного (или агрегатного) принципа, который заключается в том, что отдельные устройства выполняются в виде конструктивно законченных модулей (агрегатов), которые могут сравнительно просто в нужных количествах и номенклатуре объединяться, образуя вычислительную машину.

Присоединение нового устройства не должно вызывать в существующей части машины никаких других изменений, кроме изменения кабельных соединений и некоторых корректировок программ.

Унифицированные (не зависящие от типа ПУ) *форматы данных*, которыми ПУ обмениваются с ядром ЭВМ, в том числе унифицированный формат сообщения, которое ПУ посылает в ядро о своем состоянии. Преобразование унифицированных форматов данных в индивидуальные, приспособленные для отдельных ПУ, производится в самих ПУ.

Унифицированный интерфейс, т. е. унифицированный по составу и назначению набор линий и шин, унифицированные схемы подключения, сигналы и алгоритмы управления обменом информацией между ПУ и ядром ЭВМ.

Унифицированные (не зависящие от типа ПУ) *формат и набор команд* процессора для операций ввода-вывода. Операция ввода-вывода с любым ПУ представляет для процессора просто операцию передачи данных независимо от особенностей принципа действия данного ПУ, типа его носителя и т. п.

Унификация распространяется на семейство (ряд, систему) моделей ЭВМ.

Для обеспечения параллельной во времени работы ПУ с выполнением программы обработки данных процессором схемы управления вводом-выводом отделяют от процессора и придают им достаточную степень автономности.

Многие функции управления операциями ввода-вывода, как, например, управление прямым доступом к памяти (см. § 11.2), являются общими, они не зависят от типа ПУ. Другие являются специфичными для данного типа устройств.

Выполнение общих функций возлагают на общие для групп периферийного оборудования унифицированные устройства — контроллеры прямого доступа к памяти, процессоры (каналы) ввода-вывода, а специфических — на специализированные для данного типа ПУ электронные блоки управления.

11.2. ПРЯМОЙ ДОСТУП К ПАМЯТИ

В системах ввода-вывода ЭВМ используются два основных способа организации передачи данных между памятью и периферийными устройствами: программно-управляемая передача и прямой доступ к памяти (ПДП).

Программно-управляемая передача данных (рис. 11.1, а) осуществляется при непосредственном участии и под управлением процессора, который при этом выполняет специальную подпрограмму процедуры ввода-вывода. Данные между памятью и периферийным устройством пересылаются через процессор. Операция ввода-вывода инициируется текущей командой про-

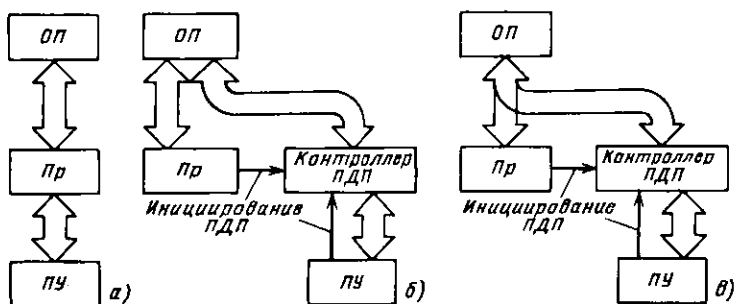


Рис. 11.1. Способы организации передачи данных между памятью и периферийными устройствами:

а — программно-управляемая передача; б — прямой доступ к памяти при наличии отдельной шины в памяти для ПДП; в — прямой доступ к памяти при использовании процессором и ПДП одной шины для связи с памятью

граммы или запросом прерывания от периферийного устройства.

При программно-управляемой передаче данных процессор на все время этой операции отвлекается от выполнения основной программы решения задачи. Операция пересылки данных логически слишком проста, чтобы эффективно загружать логически сложную быстродействующую аппаратуру процессора. В результате при использовании программно-управляемых передач данных снижается производительность вычислительной машины.

Вместе с тем при пересылке блока данных процессору приходится для каждой единицы передаваемых данных (байт, слово) выполнять довольно много команд, чтобы обеспечить буферизацию данных, преобразование форматов, подсчет количества переданных данных, формирование адресов в памяти и т. п. В результате скорость передачи данных при пересылке блока данных даже через высокопроизводительный процессор не превышает 0,02—0,05 Мбайт/с, что недостаточно для работы с высокоскоростными периферийными устройствами (например с ЗУ на дисках и барабанах, с аналого-цифровыми преобразователями и т. п.) и может оказаться вообще неприемлемой для систем управления, работающих в реальном времени. Между тем потенциально возможная максимальная скорость обмена данными при вводе-выводе определяется пропускной способностью памяти, которая, например, при цикле памяти 1 мкс составляет b Мбайт/с, где b — ширина выборки, измеряе-

мая числом байт, одновременно записываемых (считываемых) в память.

Для быстрого ввода-вывода блоков данных и разгрузки процессора от управления операциями ввода-вывода используют прямой доступ к памяти.

Прямым доступом к памяти называется способ обмена данными, обеспечивающий автономно от процессора установление связи и передачу данных между ОП и ПУ (рис. 11.1, б и в).

Прямой доступ к памяти освобождает процессор или микропроцессор от управления операциями ввода-вывода, позволяет осуществлять параллельно во времени выполнение процессором (микропроцессором) программы с обменом данными между периферийным устройством и ОП, производить этот обмен со скоростью, ограничиваемой только пропускной способностью ОП или ПУ. Таким образом, ПДП, разгружая процессор (микропроцессор) от обслуживания операций ввода-вывода, способствует возрастанию общей производительности ЭВМ или микро-ЭВМ. Повышение предельной скорости ввода-вывода информации делает машину более приспособленной для работы в системах реального времени. Прямым доступом к памяти управляет *контроллер ПДП*, который выполняет следующие функции:

- управление иницируемой процессором или ПУ передачей данных между ОП и ПУ;

- задание размера блока данных, который подлежит передаче, и области памяти, используемой при передаче;

- формирование адресов ячеек ОП, участвующих в передаче;

- подсчет числа единиц данных (байт, слов), передаваемых от ПУ в ОП или обратно, и определение момента завершения заданной операции ввода-вывода.

Указанные функции реализуются контроллером ПДП обычно с помощью одного или нескольких буферных регистров *РгБ*, регистра — счетчика текущего адреса данных *РгТАД* и текущего счетчика данных *ТСчД* (рис. 11.2).

При иницировании операции ввода-вывода в *ТСчД* заносится размер подлежащего передаче блока (число байт или слов), а в *РгТАД* — начальный адрес области памяти, используемой при передаче. При передаче каждого байта содержимое *РгТАД* увеличивается на 1, при этом формируется адрес очередной ячейки ОП, участвующей в передаче. Одновременно уменьшается на 1 содержимое *ТСчД*. Обнуление *ТСчД* указывает на завершение передачи. Контроллер ПДП обычно имеет более высокий приоритет в занятии цикла памяти по сравнению с процессором. Управление памятью переходит

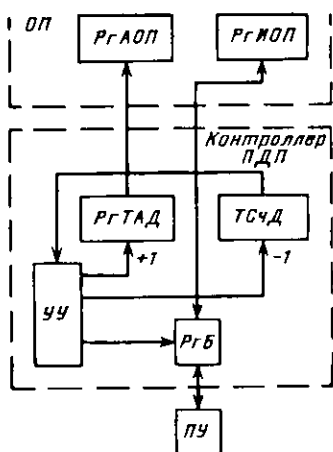


Рис. 11.2. Структурная схема прямого доступа к памяти

к контроллеру ПДП, как только завершится цикл ее работы, выполняемый для текущей команды процессора.

Прямой доступ к памяти обеспечивает высокую скорость обмена данными за счет того, что управление обменом производится не программным путем, а аппаратными средствами.

В крупных и средних ЭВМ ПДП является основным способом осуществления операций

ввода-вывода. Большинство микро-ЭВМ имеет программно-управляемый обмен данными при вводе-выводе. При необходимости имеется возможность добавления в состав микро-ЭВМ корпуса микросхемы контроллера ПДП. В таком случае программно-управляемый обмен сохраняют для операций ввода-вывода отдельных байт (слов), которые выполняются быстрее, чем при ПДП, так как исключаются потери времени на программно-управляемую установку начальных состояний регистров и счетчиков контроллера ПДП.

11.3. ОСНОВНЫЕ ПРИНЦИПЫ ПОСТРОЕНИЯ И СТРУКТУРЫ СИСТЕМЫ ВВОДА-ВЫВОДА

Принципы построения и структуры системы ввода-вывода сильно различаются в зависимости от типа ЭВМ и ее назначения. Определяющими факторами являются разнообразие и число периферийных устройств в составе ЭВМ, интенсивность операций ввода-вывода.

Можно выделить два характерных принципа построения и соответствующие структуры систем ввода-вывода: ЭВМ с *одним общим интерфейсом* и ЭВМ с *множеством интерфейсов и процессорами (каналами) ввода-вывода*.

Структура с одним общим интерфейсом (рис. 11.3) предполагает наличие общей шины (магистральной), к которой подсоединяются все модули, в совокупности образующие ЭВМ: процессор, оперативная и постоянная памяти и периферийные устройства. В каждый данный момент через общую шину мо-

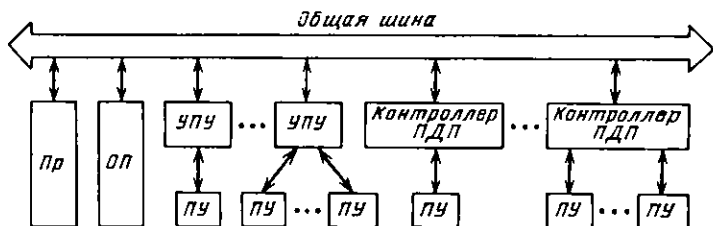


Рис. 11.3. Структура системы ввода-вывода на основе общего интерфейса (интерфейса «общая шина»)

жет происходить обмен данными только между одной парой присоединенных к ней модулей. Таким образом, модули ЭВМ разделяют во времени один общий интерфейс, причем процессор выступает как один из модулей системы.

Периферийные устройства подсоединяются к общей шине с помощью блоков управления (контроллеров) периферийными устройствами (УПУ), осуществляющих согласование форматов данных, используемых в ПУ, с форматом, принятым для передачи по общей шине («информационная ширина интерфейса»). Последний обычно соответствует машинному слову процессора и(или) ширине выборки в ОП.

Если в ПУ операции ввода-вывода производятся для отдельных байт (слов), то используется программно-управляемая передача данных через процессор и под его управлением, при этом значительно упрощается УПУ.

Однако такой способ обмена, как было показано ранее, мало подходит для ПУ с побочной передачей данных (ЗУ на дисках, лентах и некоторые другие ПУ). В этом случае применяют прямой доступ к памяти и контроллеры ПДП. Блоки буферизации и согласования форматов передаваемых данных сравнительно несложны, если совпадают или невелика кратность ширины выборки оперативной памяти (информационной ширины общей шины) и формата данных (обычно однобайтового), используемого для передачи в большинстве ПУ. Это имеет место в малых ЭВМ, микропроцессорах и микро-ЭВМ, где длина слова 1–2 байта.

Наоборот, в ЭВМ общего назначения с шириной выборки и форматом обмена между процессором и ОП 4–8 байт использование общего интерфейса потребовало бы усложнения блоков буферизации и согласования форматов в УПУ.

При общем интерфейсе аппаратура управления вводом-выводом рассредоточена по отдельным модулям ЭВМ и ее общий объем существенно зависит от числа ПУ в составе ЭВМ,

в особенности от числа ПУ с поблочным обменом, требующим контроллеров прямого доступа к памяти.

Следует отметить, что в структуре, представленной на рис. 11.3, процессор не полностью освобождается от управления операциями ввода-вывода. Более того, на все время операции передачи данных интерфейс оказывается занятым, а связь процессора с памятью блокированной. Все это приводит к снижению производительности ЭВМ.

Сказанное объясняет, почему общий интерфейс (интерфейс с общей шиной) нашел широкое применение (приобрел даже характер стандартного архитектурного решения) применительно к малым и микро-ЭВМ, которые имеют короткое слово, небольшой объем периферийного оборудования и к общей производительности которых предъявляются умеренные требования. Для машин этого класса существенным является обеспечение общим интерфейсом простота реализации системы ввода-вывода и гибкость при построении различных конфигураций вычислительных комплексов.

В то же время интерфейс с общей шиной оказывается малоприменимым для высокопроизводительных ЭВМ общего назначения, работающих с многобайтными словами, большим набором периферийных устройств, в том числе внешних ЗУ при высокой интенсивности процессов ввода-вывода данных. Использование общего интерфейса в этих машинах привело бы к снижению общей производительности машины, неоправданно большим затратам на аппаратуру управления вводом-выводом — УПУ и контроллеры ПДП, которые многократно дублировали бы выполнение одних и тех же функций, а главное, не позволило бы достигнуть высокой степени параллелизма в работе процессора и системы ввода-вывода, а следовательно, и требуемой высокой производительности ЭВМ.

Широко используется несколько модификаций общего интерфейса: «общая шина» (см. § 11.10), «мультишина» (см. § 11.11) и др.

Структура системы ввода-вывода с процессорами (каналами) ввода-вывода. В ЭВМ общего назначения система ввода-вывода строится путем централизации аппаратуры управления вводом-выводом на основе применения программно-управляемых процессоров ввода-вывода, иначе называемых *каналами ввода-вывода* (рис. 11.4). Обмен информацией между ОП и ПУ совершается через канал ввода-вывода.

Благодаря наличию в составе ЭВМ каналов ввода-вывода, способных реализовывать достаточно сложные процедуры, появляется возможность полностью разгрузить процессор от управления операциями ввода-вывода.

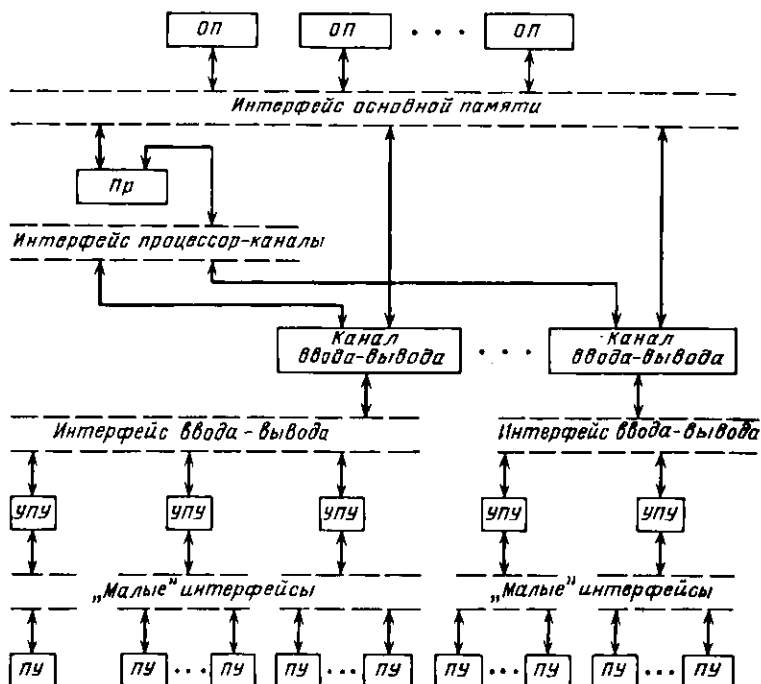


Рис. 11.4. Интерфейсы ЭВМ общего назначения (ЕС ЭВМ)

В вычислительной машине с каналами ввода-вывода отсутствует однородность в структуре потоков и форматах передаваемых данных, что приводит к необходимости иметь в ЭВМ, как это показано на рис. 11.4, несколько специализированных интерфейсов.

Можно выделить четыре типа интерфейсов: интерфейс основной (оперативной) памяти, интерфейс «процессор – каналы», интерфейс ввода-вывода (интерфейс периферийных устройств), интерфейс периферийных аппаратов («малый» интерфейс). Последний имеет место и в ЭВМ с общим интерфейсом.

Через *интерфейс основной памяти* производится обмен информацией между ОП, с одной стороны, и процессором (процессорами) и каналами ввода-вывода – с другой.

Интерфейс «процессор – каналы» предназначен для передачи информации между процессорами и каналами ввода-вывода. Через *интерфейс ввода-вывода* происходит обмен инфор-

мацией между каналами и УПУ. Через «малый» интерфейс осуществляется передача информации между УПУ и ПУ.

Наиболее быстродействующими являются интерфейс основной памяти и интерфейс «процессор — каналы». В (из) ОП информация передается словами или словами двойной длины (4—8 байт). Через интерфейс ввода-вывода информация передается байтами или парами байт. При проектировании ЭВМ стремятся унифицировать интерфейсы, и в первую очередь интерфейсы, обеспечивающие сопряжение с периферийными устройствами (интерфейсы ввода-вывода).

Интерфейсы периферийных аппаратов не могут быть унифицированы, так как сами периферийные аппараты весьма разнообразны по принципу действия, по выполняемым операциям и по используемым форматам данных и сигналам.

Интерфейсы характеризуются следующими параметрами:
пропускная способность интерфейса. Определяется количеством информации, которая может быть передана через интерфейс в единицу времени;

максимальная частота передачи информационных сигналов через интерфейс;

максимально допустимое расстояние между соединяемыми устройствами;

динамические параметры интерфейса. Время передачи отдельного слова и блока данных с учетом продолжительности процедур подготовки и завершения передачи. Эти параметры имеют существенное значение для систем управления, работающих в реальном масштабе времени;

общее число проводов (линий) в интерфейсе;

информационная ширина интерфейса — число бит или байт данных, передаваемых параллельно через интерфейс;

связность интерфейса. Интерфейс может быть *односвязным*, когда существует лишь единственный путь передачи информации между парой устройств машины, и *многосвязным*, позволяющим устройствам обмениваться информацией по нескольким независимым путям. Многосвязность интерфейсов требует дополнительной аппаратуры, но повышает надежность и живучесть вычислительной машины, обеспечивает возможность автоматической реконфигурации вычислительного комплекса при выходе из строя отдельных устройств (см. гл. 15).

11.4. ОСНОВНЫЕ ФУНКЦИИ КАНАЛОВ ВВОДА-ВЫВОДА. УПРАВЛЯЮЩАЯ ИНФОРМАЦИЯ ДЛЯ ОПЕРАЦИЙ ВВОДА-ВЫВОДА

При определении функций, которые следует возложить на каналы ввода-вывода, следует исходить из необходимости обеспечения условия для реализации параллельной во времени работы процессора над программой с выполнением ПУ операций ввода-вывода. Для этого следует стремиться в возможно большей степени освободить процессор от управления операциями обмена информацией между периферийными устройствами и ОП, возложить это на каналы, управляемые *канальными программами*.

За процессором следует оставить лишь инициирование операции ввода-вывода, задание номеров канала и ПУ, участвующих в операции, и указание адреса начала программы канала для задаваемой операции ввода-вывода.

Канал призван обеспечивать прямой доступ к памяти, а поэтому подобно рассмотренному в § 11.3 контроллеру ПДП должен выполнять следующие функции: задание размера массива данных и области памяти, участвующих в обмене информацией, формирование адресов последовательных ячеек ОП, используемых в передаче, подсчет числа единиц данных (слов, байт и т. д.), прошедших через канал, и определение момента завершения передачи массива данных. При этом канал должен осуществлять буферизацию и преобразование форматов передаваемых данных для согласования работы ОП и ПУ.

Помимо указанных функций на канал возлагается ряд дополнительных для минимизации участия процессора в операциях ввода-вывода.

Организация цепочки данных. Возможны случаи, когда массив информации, предназначенный для операции ввода или вывода с некоторым ПУ, не располагается в памяти подряд, а состоит из нескольких подмассивов, размещенных в произвольно расположенных участках ОП. Чтобы в этом случае ввод (вывод) каждого подмассива не требовал включения в программу процессора отдельной команды ввода-вывода, а передача всех подмассивов инициировалась всего одной командой процессора, канал должен допускать задание в канальной программе *цепочки данных* для передачи такого составного массива.

Организация пропуска информации. При операциях ввода может возникнуть необходимость переносить в память с носителя информации отдельные части массива, пропуская ненужные данные. Должна иметься возможность задания в канальной программе пропуска информации в цепочке данных

и реализации пропуска без привлечения процессора для выполнения этой процедуры.

Организация цепочки операций. Для выполнения программы может оказаться необходимым такой обмен информацией между ОП и некоторым ПУ, для выполнения которого с этим ПУ должна выполняться определенная последовательность операций ввода-вывода. Например, при работе с ЗУ на магнитных дисках может потребоваться следующая последовательность операций: а) установить головки на i -й цилиндр, б) прочитать информацию с j -й поверхности дисков, в) прочитать информацию с $(j+1)$ -й поверхности, г) установить головки на k -й цилиндр и т. д. Целесообразно, чтобы при подобных последовательностях операций ввода-вывода с одним и тем же ПУ не требовалось для каждой новой операции участия процессора, т. е. не требовалась новая команда ввода-вывода в программе процессора. Для этого канал ввода-вывода должен допускать задание цепочек операций в программе канала.

Блокировка контроля неправильной длины считанного массива. Операции ввода-вывода сопровождаются автоматическим контролем, в том числе контролем соответствия длины массива, считанного или записанного в результате операции ввода-вывода, длине физической записи. При нарушении соответствия возникает прерывание от ввода-вывода по *неправильной длине*. Поскольку имеется много случаев, когда следует блокировать это прерывание (соответствующие примеры будут приведены далее), целесообразно, чтобы канал допускал задание такой блокировки в канальной программе.

Формирование запросов прерывания от ввода-вывода. Канал должен прерыванием извещать процессор об окончании каждой операции ввода-вывода, а также об обнаружении ошибки или каких-либо других необычных условий, вследствие чего произошло принудительное окончание выполняемой операции. Это прерывание формируется автоматически аппаратурой канала. Наряду с этим должна иметься возможность задания в программе канала прерывания на любом этапе операции ввода-вывода. Такое прерывание называется *программно-управляемым*. Оно не нарушает нормальное выполнение текущей операции ввода-вывода. Появление запроса программно-управляемого прерывания означает, что выполнены все операции ввода-вывода, предшествующие в канальной программе этому запросу. Это позволяет процессору следить за выполнением канальной программы ввода-вывода и начинать обработку данных сразу, как только очередной этап операции ввода-вывода выполнен.

Управляющая информация для операций ввода-вывода. В ЭВМ с каналами ввода-вывода управление вводом-выводом строится иерархическим образом (рис. 11.5). В операциях ввода-вывода участвуют три типа устройств: процессор (первый уровень управления), канал ввода-вывода (второй уровень), периферийное устройство (третий уровень). Каждому типу устройств соответствует определенный вид управляющей информации: процессору — команды ввода-вывода, каналу — *управляющие слова канала (УСК)*, периферийному устройству — *приказы*. Кроме того, в управлении вводом-выводом используются коды состояния канала [*слово состояния канала (ССК)*] и ПУ (*байт состояния* и *байты уточненного состояния*). О кодах состояния см. в § 11.10.

Форматы основных видов управляющей информации для операций ввода-вывода в ЕС ЭВМ представлены на рис. 11.6.

Каждая операция или совокупность операций ввода-вывода производится под управлением соответствующей программы канала, реализуемой аппаратурными средствами канала, ПУ (и его УПУ) и интерфейса ввода-вывода.

Программа канала представляет собой некоторую последовательность УСК, обеспечивающую выполнение определенной операции ввода-вывода. Обычно каналные программы хранятся в ОП.

Таким образом, в ЭВМ организуется параллельное выполнение во времени взаимодействующих между собой программно-управляемых процессов: выполнение процессором программы обработки информации и выполнение каналами и ПУ канальных программ операций ввода-вывода.

Система команд ЭВМ содержит небольшое число универсальных по отношению к разным типам ПУ команд ввода-вывода. Так, в ЕС ЭВМ I очереди для управления вводом-выводом используется всего четыре команды: «Начать ввод-вывод», «Остановить ввод-вывод», «Проверить ввод-вывод», «Проверить канал».

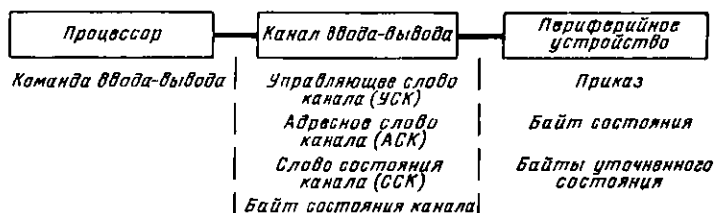


Рис. 11.5. Иерархия управляющей информации для операции ввода-вывода

Команды ввода-вывода являются привилегированными и выполняются только в состоянии «супервизор». Все команды ввода-вывода завершаются формированием признака результата в ССП.

Команда «Начать ввод-вывод» инициирует одну или несколько (цепочку) операций ввода-вывода с указанным в команде каналом и ПУ. Этой командой процессора начинаются операции ввода-вывода с любым ПУ. О завершении отдельного ее этапа, или цепочки операций, канал сообщает процессору путем прерываний.

Во всех командах ввода-вывода (кроме первой) достаточно указывать код операции и номера канала и ПУ. В команде «Начать ввод-вывод», кроме того, нужно указывать адрес первого УСК в программе канала. Для единообразного представления всех команд ввода-вывода в ЕС ЭВМ для них принят один общий формат, показанный на рис. 11.6, а. Адрес первого УСК программы канала содержится в так называемом *адресном слове канала (АСК)* (рис. 11.6, б), хранимом в определенной ячейке ОП, куда оно должно быть помещено до начала выполнения команды «Начать ввод-вывод».

При выполнении команды ввода-вывода содержимое указанного в команде регистра B_1 складывается с числом в поле D_1 и полученная сумма располагается в разрядах 21–31 слова, при этом разряды 21–23 и 24–31 указывают соответственно номер канала и номер ПУ.

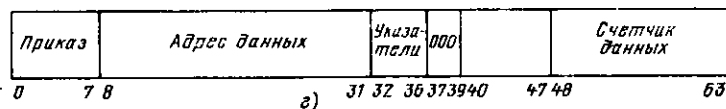
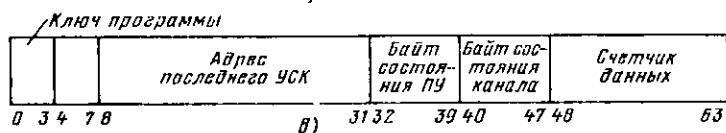
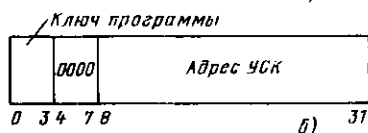
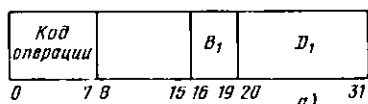


Рис. 11.6. Форматы команды ввода-вывода (а), адресного слова канала (б), слова состояния канала (в) и управляющего слова канала (г) в ЕС ЭВМ

Адресное слово канала содержит код «ключа», используемый для защиты памяти (см. гл. 14) при данной операции ввода-вывода. Признак результата, формируемый при выполнении команды «Начать ввод-вывод», указывает, нормально ли прошел нукс операции ввода-вывода.

По команде «Остановить ввод-вывод» операция ввода-вывода может быть принудительно прекращена процессором до ее завершения в адресуемых командой канале и ПУ.

Команды «Проверить канал» и «Проверить ввод-вывод» позволяют процессору определять состояние канала и ПУ. По команде «Проверить канал» канал в фиксированной ячейке ОП формирует ССК (рис. 11.6, в) и устанавливает в ССП признак результата, определяющий состояние адресуемого канала: канал доступен, канал хранит условия прерывания, канал работает в монопольном режиме, канал выключен. При выполнении команды «Проверить ввод-вывод» из ПУ в канал выдается байт его состояния, а из канала в процессор поступает признак результата, указывающий следующие возможные ситуации: адресуемое ПУ доступно, ССК записано, канал (подканал) занят, адресуемое ПУ выключено.

Управляющее слово канала (УСК). В ЕС ЭВМ формат УСК соответствует двойному слову, т. е. 64 разрядам (рис. 11.6, г). При выполнении канальной программы ее управляющие слова выбираются из последовательно расположенных в ОП 8-байтных ячеек, если только не имеется предусмотренное программой канала нарушение естественного порядка выборки УСК. Для удобства дальнейшего изложения материала будем пользоваться следующей упрощенной структурой УСК:

Код операции (приказ)	Указатели					Адрес данных (АД)	Счетчик данных (СчД)
	ЦД	ЦО	УБ	ПИ	ПР		

Код операции (приказ) определяет как для канала, так и для ПУ тип операции, задаваемой этим управляющим словом.

Приказ представляет собой часть УСК, которая задает операцию, выполняемую каналом, каналом и ПУ совместно или одним ПУ. В последних двух случаях приказ передается в ПУ и инициирует в нем определенные действия (например, установку головок на нужный цилиндр в ЗУ на дисках, запись информации и др.). Приказ содержит информацию, специфичную для данного типа ПУ.

Приведем основные виды приказов, допускающих ряд модификаций:

«Записать» (вывод информации из ОП в ПУ);

«Прочитать» (ввод информации из ПУ в ОП);

«Прочитать в обратном направлении» (только для ЗУ на магнитных лентах);

«Управление» (перемещение магнитной ленты, перемещение головок в ЗУ на дисках и другие служебные операции);

«Проверить ввод-вывод» (запрос байта состояния);

«Уточнить состояние» (запрос кода уточненного состояния ПУ).

Кроме операций, управляющих передачами информации, предусматриваются также операции, которые нужны для построения самих программ канала. К ним относится так называемый «Переход в канале». Эта операция является внутренней процедурой канала и непосредственно на работу ПУ не влияет. Управляющее слово, осуществляющее переход в канале, указывает адрес следующего УСК в цепи и, таким образом, позволяет выполнять программы канала, в которых УСК располагаются не в последовательных ячейках памяти. Этот вид перехода в программе канала является безусловным переходом.

Для реализации ветвлений в программах канала применяются условные переходы. В зависимости от признаков, характеризующих состояние ПУ, после выполнения очередного управляющего слова следующее УСК в цепи операции либо выполняется, либо пропускается.

В ЕС ЭВМ признаком условного перехода в канальной программе служит содержимое разряда «Модификатор состояния» в байте состояния, который ПУ посылает в канал по завершении операции, предписанной текущим УСК. Если модификатор состояния содержит 0, то выбирается и исполняется следующее по порядку УСК в цепочке операций. Если модификатор равен 1, следующее УСК пропускается.

Далее будет рассмотрен пример программы канала, в которой используется условный и безусловный переходы.

Адрес данных (АД) и счетчик данных (СчД) определяют область памяти, используемую в операции ввода-вывода. Адрес данных указывает адрес первого (или последнего при обратном вводе) байта из массива информации; счетчик данных указывает число байт.

Указатель цепочки данных (ЦД). При ЦД = 0 операция после использования данного УСК оканчивается, при ЦД = 1 операция продолжается с новым массивом данных, указанным в следующем УСК.

Указатель цепочки операций (ЦО). При ЦО = 0 программа канала для данного ПУ заканчивается на рассматриваемом УСК, если нет указаний о продолжении цепочки данных. Если ЦД = 0, то при ЦО = 1 после выполнения действий, предусмо-

тренных данным УСК, следующее по порядку УСК выбирается из памяти и начинается выполнение новой операции ввода-вывода с тем же ПУ.

Указатель блокировки (УБ) сигнала неправильной длины, формирующегося (в отсутствие признака цепочки данных ЦД = 1) при несоответствии числа фактически переданных байт длине физической записи. Этот сигнал прекращает выполнение канальной программы и вызывает прерывание программы процессора. При УБ = 1 сигнал неправильной длины блокируется.

Указатель пропуска информации (ПИ). При ПИ = 1 передача информации между каналом и оперативной памятью подвешивается, канал осуществляет только подсчет слов, проходящих между каналом и ПУ.

Указатель прерывания (ПР). При ПР = 1 канал посылает в процессор запрос программно-управляемого прерывания.

Примеры программ канала

Пример 1. Выборочная передача информации между ПУ и ОП. Предположим, что в ОП машины должны быть приняты из некоторой зоны магнитной ленты, содержащей 500 байт, первые 10 и последние 20 байт. Если считать, что требуемая зона уже подведена под считывающие головки, то для выполнения данной операции необходимо выполнить следующую последовательность УСК, образующих цепь данных:

Адрес УСК	Приказ	Указатели					Адрес данных (АД)	Счетчик данных (СЧД)
		ЦД	ЦО	УБ	ПИ	ПР		
1	Прочитать	1	0	0	0	0	α	10
1+8	Прочитать	1	0	0	1	1	0	470
1+16	Прочитать	0	0	1	0	0	$\alpha + 10$	20

Примечание. В УСК, продолжающих цепочку данных, в качестве приказа может быть записан любой код, кроме кода «переход в канал». Обычно используют нулевой код или повторяют приказ из первого УСК цепочки данных.

Первое УСК выполняет ввод первых 10 байт в ячейки с адресом от α до $\alpha + 9$. Второе УСК обеспечивает пропуск следующих 470 байт; установленный в этом УСК указатель ПР = 1, вызывая программно-управляемое прерывание, позволяет программе немедленно начать обработку начальных 10 байт. Третье УСК выполняет ввод в ОП последних 20 байт из зоны магнитной ленты и заканчивает программу канала, т. к. ЦД = 0 и ЦО = 0. Автоматически формируемое каналом по

окончании цепочки УСК прерывание сигнализирует процессору, что затребованная операция ввода-вывода выполнена полностью.

Пример 2. Ввод данных с перфокарт.

Пусть необходимо произвести ввод данных с двух перфокарт: с первой 25 байт, расположенных на карте в колонках с 31 по 55, а со второй 30 байт из колонок 41—70 и расположить их в ОП, начиная с адресов соответственно α и β . Для рассматриваемого примера приведем канальную программу:

Адрес УСК	Приказ	ЦД	ЦО	УБ	ПИ	ПР	АД	СчД
1	Прочитать	1	0	0	1	0	0	30
1+8	Прочитать	0	1	1	0	0	α	25
1+16	Прочитать	1	0	0	1	1	0	40
1+24	Прочитать	0	0	1	0	0	β	30

В программе канала организуется цепочка операций, так как считывание каждой перфокарты составляет самостоятельную операцию ввода-вывода. Использование указателя блокировки неправильной длины позволяет сократить число УСК в программе и освободить канал раньше, чем закончится чтение физической записи (в данном случае перфокарты).

Пример 3. Поиск информации в ЗУ на магнитных дисках (ЗУД).

Пусть необходимо найти в ЗУД по ключу (см. гл. 10) блок данных, пользуясь ключом-эталоном, имеющим длину n байт и расположенным в ОП, начиная с адреса α , а затем первые m байт передать в ОП, начиная с адреса β .

Блок управления ЗУД, просматривая последовательно все блоки информации на носителе (или заданной дорожке), должен найти блок, ключ которого совпадает с эталоном. Поскольку поле ключа может иметь большую длину (до 256 байт), невыгодно устанавливать в блоке управления ЗУД триггерные регистры для хранения ключа-эталона целиком на все время поиска. Ключ-эталон многократно считывается из ОП (при прохождении каждого блока информации под головкой) и побайтно сравнивается с данными, считываемыми с диска.

Для организации такого поиска построим программу канала в виде цепи операций с использованием безусловного перехода, выполняемого операций «Переход в канале», и условного перехода по признаку «Модификатор состояния» в байте состояния, который посылает в канал ЗУД.

Предполагая, что требуемые цилиндр и дорожки в ЗУД уже выбраны, можно написать следующую программу канала:

Адрес УСК	Приказ	ЦД	ЦО	УБ	ПИ	ПР	АД	СчД
<i>l</i>	<i>Поиск по ключу</i>	0	1	1	0	0	α	<i>n</i>
<i>l+8</i>	<i>Переход в канале</i>	0	1	0	0	0	<i>l</i>	0
<i>l+16</i>	<i>Прочитать</i>	0	0	1	0	0	β	<i>m</i>

Первое УСК программы канала инициирует в ЗУД операцию поиска. Запоминающее устройство на диске выбирает из первого подошедшего под головку блока информации байты ключа и последовательно сравнивает их с *n* байтами эталона, которые выбираются каналом из ОП, начиная с адреса α . Если совпадения ключей нет, то следующее УСК (*переход в канале*) вызывает повторное выполнение УСК из ячейки *l*, т. е. повторение поиска для следующего на дорожке блока информации. Если блок информации с требуемым полем ключа найден, то ЗУД формирует байт состояния с 1 в разряде «Модификатора состояния». Канал пропускает в своей программе УСК из ячейки *l+8* и переходит к выполнению УСК из ячейки *l+16*. В этом случае в ОП вводятся *m* байт и помещаются в группу последовательных ячеек, начиная с ячейки β .

11.5. ОСНОВНЫЕ ТИПЫ И СТРУКТУРЫ КАНАЛОВ ВВОДА-ВЫВОДА

Способ организации взаимодействия ПУ с каналом определяется соотношением быстродействия ОП и ПУ. По этому признаку ПУ можно классифицировать на две группы: быстродействующие (ЗУ на барабанах, дисках, лентах и другие устройства) со скоростью приема и выдачи информации примерно $(0,1 - 10) \cdot 10^6$ байт/с и медленнодействующие (перфоленочные и перфокарточные устройства, печатающие устройства и др.) со скоростью около 1–2 тыс. байт/с и менее. Оперативная память может выдавать или принимать данные со скоростью примерно $(1 - 10) \cdot 10^6$ байт/с.

В зависимости от соотношения быстродействия ОП и ПУ в каналах ввода-вывода реализуются два режима работы — монополюсный и разделения времени (мультиплексирования).

Монополюсный режим. После установления связи между каналом и ПУ последнее монополизирует канал на все время, пока полностью не завершится инициированная процессором канальная программа (цепочка операций) работы с данным ПУ и не будут произведены все предусмотренные этой программой

передачи данных между ПУ и ОП. На все время выполнения данной канальной программы канал оказывается занятым для других ПУ.

Режим разделения времени (режим мультиплексирования).

Несколько ПУ разделяет во времени канал ввод-вывод, при этом каждое из параллельно работающих с данным каналом ПУ связывается с каналом на короткие промежутки времени только после того, как ПУ подготовлено к приему или выдаче очередной порции информации (байта, группы байт и т. п.). Промежуток времени, в течение которого происходит передача информации между каналом и подготовленным к этому ПУ, может быть назван *сеансом связи*. Сеансы связи различных ПУ чередуются между собой. Во время сеанса связи одного из устройств с каналом другие устройства могут выполнять работу, не требующую использования средств канала (например, печатать очередной символ или считывать символ с носителя информации).

В соответствии с преимущественно реализуемым режимом работы различают каналы ввода-вывода *мультиплексный*, осуществляющий мультиплексирование ПУ, и *селекторный*, взаимодействующий с ПУ в монопольном режиме.

Мультиплексный (байт-мультиплексный) канал одновременно обслуживает несколько параллельно работающих ПУ, попеременно организуя с ними сеансы связи для передачи между ОП и ПУ небольших порций информации (1 байта или нескольких).

Если несколько ПУ подготовилось к очередному сеансу связи и запрашивает обслуживание со стороны мультиплексного канала, то канал выбирает одно из них в соответствии с принятыми для данной системы приоритетными правилами, например в соответствии с порядком подключения устройств к каналу. Остальные устройства, готовые к сеансу связи, должны ожидать, когда подойдет их очередь на обслуживание. Следовательно, мультиплексный канал предназначен главным образом для работы со сравнительно медленными устройствами, способными ожидать обслуживания без потери информации. Аппаратурные средства мультиплексного канала можно условно разделить на две части (рис. 11.7, а): средства, предназначенные для обслуживания отдельных ПУ, присоединенных к каналу, и оборудование, являющееся общим для всех устройств и разделяемое всеми устройствами во времени.

Средства канала, выделенные для обслуживания одного устройства, принято именовать *подканалом*.

Число подканалов определяет максимальное число одновременно работающих с данным каналом ПУ. Физически подканал реализуется в виде участка памяти, в котором хранятся па-

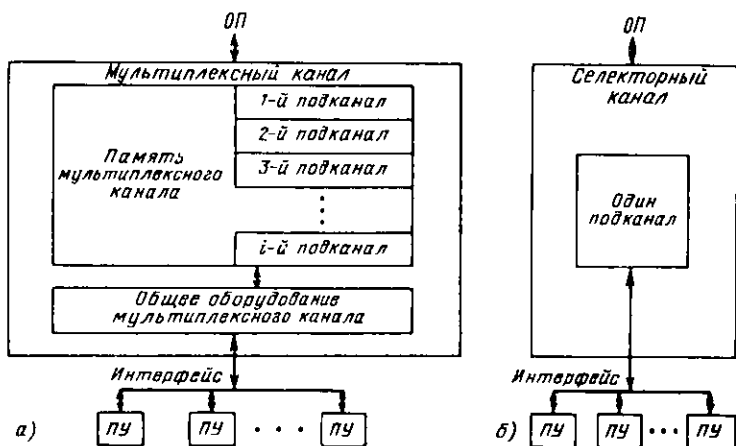


Рис. 11.7. Типы каналов:

а — мультимплексный; б — селекторный

параметры операции ввода-вывода, выполняемой данным устройством: текущие значения адреса и счетчика данных, код и указатели операции ввода-вывода, адрес следующего УСК и др. В качестве памяти для хранения этих параметров может использоваться либо специальная память, встроенная в мультимплексный канал, либо участок ОП машины.

Общее оборудование мультимплексного канала представляет собой набор триггерных регистров и комбинационных схем, позволяющих осуществлять обмен информацией с ОП и ПУ, модификацию текущих параметров операции ввода-вывода.

Селекторный канал предназначен для монопольного обслуживания одного ПУ. При работе с селекторным каналом ПУ после пуска операции остается связанным с каналом до окончания цепи операций. Запросы на обслуживание от других ПУ, так же как и новые команды пуска операций ввода-вывода от процессора, в это время не воспринимаются каналом: до завершения цепи операций селекторный канал по отношению к процессору представляется занятым устройством.

Управляющее слово выбирается селекторным каналом 1 раз в начале операции и содержится до окончания всех предписанных действий в триггерных регистрах канала. Необходимые изменения текущих параметров операции производятся быстро с помощью соответствующих действий над содержимыми триггерных регистров. Таким образом, все средства селекторного канала монополизируются на время операции од-

ним ПУ. Можно считать, что селекторный канал содержит только один подканал (рис. 11.7, б).

Вследствие отсутствия потерь времени на перезапоминание текущих параметров операции ввода-вывода селекторный канал обладает высокой степенью готовности к обслуживанию пущенного им устройства и предназначается для работы с быстродействующими устройствами, которые могут терять информацию вследствие задержек в обслуживании (ЗУ на магнитных лентах, дисках и др.).

Возможность монопольного обслуживания ПУ вводится иногда и в мультиплексный канал. В этом режиме один из подканалов полностью занимает канал; обслуживание других пущенных в работу подканалов задерживается до окончания монопольной операции и восстановления мультиплексного режима работы. Наличие монопольного режима позволяет подсоединять быстродействующие ПУ к мультиплексному каналу.

В новых моделях ЭВМ ЕС для организации параллельной работы нескольких ВЗУ с прямым доступом используют *блок-мультиплексные каналы*, в которых операции, не связанные с передачей данных (установка головок на цилиндр, поиск записи и др.), выполняются для нескольких устройств в мультиплексном режиме, а передача блока информации происходит в монопольном (селекторном) режиме. Важным свойством блок-мультиплексного канала является возможность мультиплексирования передач блоков данных, относящихся к различным ВЗУ прямого доступа (см. § 11.9).

Блок-мультиплексный канал содержит несколько подканалов.

Структура мультиплексного канала (рис. 11.8). В состав мультиплексного канала входят следующие основные узлы:

- 1) набор триггерных регистров, в которых во время сеанса связи с ПУ содержатся текущие параметры активного подканала (активным называется тот подканал, для которого производится данный сеанс связи). Он состоит из регистра кода операции и указателей $P_2КОУ$, регистра текущего адреса данных $P_2ТАД$, содержащего адрес байта в ОП, участвующего в данный момент в операции ввода-вывода, счетчика текущих данных $СЧТД$, указывающего число байт, которое осталось ввести в ОП или вывести в данной операции, регистра адреса управляющего слова канала $P_2АУСК$, определяющего адрес очередного УСК в цепи управляющих слов;

- 2) память подканалов, представляющая собой внутреннее ЗУ канала и предназначенная для хранения текущих параметров операций ввода-вывода, относящихся к пассивным подканалам. Каждому подканалу отведен в памяти участок, в кото-

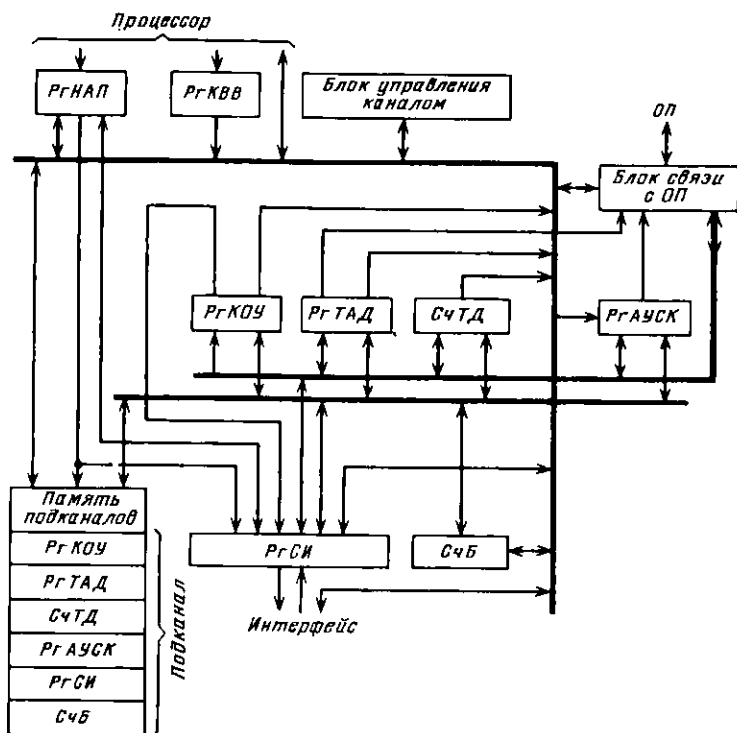


Рис. 11.8. Структурная схема мультиплексного канала

ром хранится содержимое регистров $R_2КОУ$, $R_2ТАД$, $СчТД$, $R_2АУСК$, соответствующее последнему сеансу связи для данного подканала. Кроме того, в памяти подканалов предусматривается место для хранения некоторых других параметров операции, о которых будет сказано далее. Подканалы в памяти располагаются упорядоченно по возрастанию номера подканала, который таким образом, может использоваться для адресации ячеек памяти подканалов. При формировании адресов ячеек памяти подканалов используется содержимое регистра номера активного подканала $R_2НАП$;

3) регистр связи с интерфейсом $R_2СИ$, в который поступает информация, получаемая из ПУ при вводе, и из которого в ПУ выдается выводимая информация.

Обмен информацией канала с ОП производится словами или двойными словами. Обмен с ПУ производится обычно более мелкими единицами информации, например байтами. По-

этому при вводе информации канал производит в $P_2СИ$ компоновку слова из поступающих в канал байт, а при выводе — развертку слова в $P_2СИ$ в последовательность выдаваемых в ПУ байт. Для определения конца компоновки или развертки слова используется счетчик байт $СчБ$, который указывает номер последнего обработанного байта в текущем слове данных. Содержимое $СчБ$ и $P_2СИ$ запоминается в соответствующем участке памяти подканалов наряду с регистрами $P_2КОУ$, $P_2ТАД$ и т. д.;

4) регистр команды ввода-вывода $P_2КВВ$, предназначенный для хранения кода (двухбитового) операции команды ввода-вывода, поступающего в канал из процессора, когда процессор в соответствии со своей программой выполняет новую команду ввода-вывода.

Выполнение операции ввода-вывода в канале можно рассматривать как совокупность нескольких видов процедур, из которых наиболее важными являются две: *начальная выборка и обслуживание* ПУ. Начальная выборка производится по инициативе процессора при пуске новой операции ввода-вывода. Процессор передает в канал код операции команды ввода-вывода, номер ПУ и адрес первого УСК. Эти параметры заносятся каналом в регистры $P_2КВВ$, $P_2НАП$ и $P_2АУСК$. В тех машинах, где адрес первого УСК не указывается в команде процессора, а содержится в специализированной ячейке ОП (в адресном слове канала), канал, заполнив $P_2КВВ$ и $P_2НАП$, сам считывает из ОП адрес первого УСК и заносит его в $P_2АУСК$. После этого, используя содержимое $P_2АУСК$ как адрес, канал выбирает из ОП первое УСК, размещая его поля в регистрах $P_2КОУ$, $P_2ТАД$, $СчТД$. К содержимому $P_2АУСК$ прибавляется число, равное длине управляющего слова, после чего $P_2АУСК$ указывает адрес следующего УСК в цепочке. Канал запускает в работу требуемое ПУ. Если устройство свободно и во время начальной выборки не обнаружены программные или аппаратные ошибки, считается, что пуск произошел нормально. Канал сообщает об этом процессору, который переходит к выполнению следующей команды своей программы. Канал записывает в участок памяти подканалов, соответствующий номеру ПУ в $P_2НАП$, содержимое регистров $P_2КОУ$, $P_2ТАД$, $СчТД$, $P_2АУСК$, $СчБ$, $P_2СИ$. На этом начальная выборка заканчивается, канал освобождается и готов к обслуживанию ранее пущенных ПУ или приему новых команд из процессора.

Процедура обслуживания производится по инициативе ПУ, которое посылает в канал требование обслуживания, после того как устройство готово к передаче очередной порции информации (например, байта). Канал, восприняв это требование, получает из интерфейса номер ПУ, пересылает его через $P_2СИ$

в $P_2НАП$ и в соответствии с содержимым $P_2НАП$ считывает из памяти подканалов участок, соответствующий активному подканалу. Текущие параметры операции из памяти подканалов размещаются в регистрах $P_2КОУ$, $P_2ТАД$, $С_4ТД$, вводимое (или выводимое) слово — в регистре $P_2СИ$, номер текущего байта — в $С_4Б$, адрес следующего УСК в $P_2АУСК$.

При выполнении операции ввода из ПУ принимается байт данных, который размещается в $P_2СИ$ на месте, определяемом счетчиком байт. При выводе байт данных из $P_2СИ$ посылается в ПУ. Содержимое $С_4Б$ увеличивается, а содержимое $С_4ТД$ уменьшается на 1.

Если $С_4Б$ указывает, что компоновка (или развертка) слова данных окончена, то канал осуществляет связь с ОП и в соответствии с адресом, размещенным в $P_2ТАД$, записывает в ОП введенное в $P_2СИ$ слово или считывает из ОП в $P_2СИ$ новое слово для вывода. Содержимое $P_2ТАД$ увеличивается на 1 (или уменьшается при обратном вводе), $С_4Б$ устанавливается в 0.

Если содержимое счетчика данных в $С_4ТД$ не равно 0, то канал запоминает в памяти подканалов новые значения параметров из $P_2ТАД$, $С_4ТД$, $С_4Б$, $P_2СИ$ и заканчивает сеанс связи с ПУ.

Если содержимое счетчика данных равно 0, то канал информирует ПУ об окончании операции (если только в УСК не указана цепь данных), затем, используя содержащийся в $P_2АУСК$ адрес следующего УСК для данного подканала, по адресу из $P_2АУСК$ считывает в свои регистры новое УСК и выполняет процедуру начальной выборки для пуска следующей операции в цепи управляющих слов. При обработке последнего УСК в цепочке канал посылает в процессор прерывание, сигнализирующее об окончании цепочки УСК.

Структура селекторного канала (рис. 11.9) содержит набор триггерных регистров, большинство из которых по своим функциям аналогично соответствующим регистрам в описанном ранее примере мультиплексного канала.

Регистр номера периферийного устройства $P_2НПУ$, заполняемый процессором при начальной выборке, указывает, с каким из периферийных устройств проводится текущая операция. Регистр данных $P_2Д$ и регистр предварительного управляющего слова канала $P_2ПУСК$ являются буферными и служат для уменьшения задержек в работе селекторного канала при обращениях в ОП.

Процедуры работы селекторного канала во многом похожи на процедуры мультиплексного канала. Основное отличие состоит в том, что текущие параметры операции в селекторном

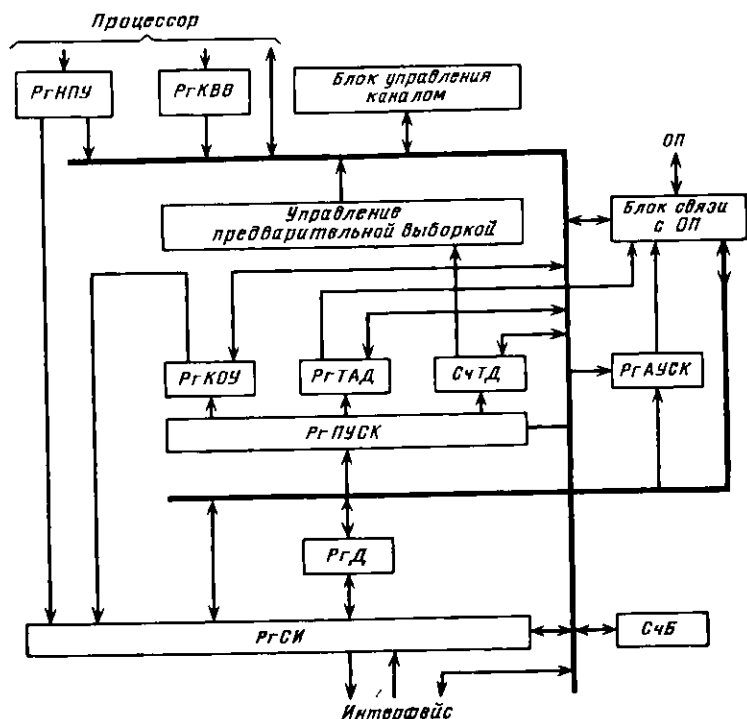


Рис. 11.9. Структурная схема селекторного канала

канале в течение всей операции содержатся и модифицируются в триггерных регистрах, что обеспечивает высокое быстродействие этого канала.

Кроме того, за счет введения дополнительных буферных регистров — регистра данных $R_2Д$ и регистра предварительной выборки управляющего слова $R_2ПУСК$ — в селекторном канале обеспечивается возможность совмещения во времени обмена информацией с ПУ и ОП.

При вводе информации поступающие из ПУ данные компонуются в регистре $R_2СИ$ в слово. Как только слово данных сформировано, оно передается в $R_2Д$ и канал начинает связываться с ОП для записи слова в памяти. Параллельно с обращением к ОП канал может производить накопление байт следующего слова в $R_2СИ$.

Аналогично совмещается связь с ОП и ПУ при выводе. Канал засылает в $R_2СИ$ через $R_2Д$ очередное слово, выбранное из

ОП, и пока это слово развертывается в последовательность байт, посылаемых в ПУ, канал связывается с ОП для выборки следующего слова.

При работе канала с быстродействующими устройствами с движущимся носителем информации (магнитные ленты, диски, барабаны) имеется опасность потери информации в моменты выборки очередных управляющих слов. Примером может служить процедура исполнения цепи данных, когда канал, не замедляя темпа поступления информации из интерфейса, должен успеть не только принимать порции данных, компоновать их в слово и записывать в память, но и выбирать следующее управляющее слово в цепочке.

В связи с этим в селекторных каналах применяют предварительную выборку управляющего слова. Схема управления предварительной выборкой следит за состоянием счетчика текущих данных, и, когда содержимое СчТД станет меньше принятого для данной машины значения, канал производит обращение к памяти по адресу из *Р₂АУСК*. Новое управляющее слово посылается в *Р₂ПУСК*, в то время как канал продолжает обработку предыдущего УСК. После того как содержимое СчТД станет равным 0, содержимое *Р₂ПУСК* замещает старое содержимое *Р₂ТАД*, СчТД и указателей в регистре *Р₂КОУ*.

В блоках управления каналами используют оба известных принципа построения управляющих автоматов: на основе хранимой в памяти логики (микропрограммное управление) — преимущественно в ЭВМ малой и средней производительности и на основе схемной (жесткой) логики, а также на основе комбинации обоих принципов — в высокопроизводительных ЭВМ.

В заключение данного параграфа в качестве примера приведены характеристики каналов ввода-вывода ЭВМ ЕС-1045. В состав этой машины входят до двух байт-мультиплексных и до пяти блок-мультиплексных (или селекторных) каналов. Общее число подсоединяемых каналов не более шести. Пропускная способность байт-мультиплексного канала 40 Кбайт/с, при работе в монопольном режиме 120 Кбайт/с. Пропускная способность блок-мультиплексного канала 0,5–1,5 Мбайт/с (зависит от номера канала). Общая пропускная способность при шести каналах достигает 5 Мбайт/с.

Управление каналами аппаратурно-микропрограммное. Передачей данных через интерфейс ввода-вывода управляет аппаратура канала, а передачей между каналами и ОП — микропрограммные средства процессора, в том числе его управляющая память (процессор при этом приостанавливается).

Памяти подканалов всех каналов реализованы в виде области ОП. Управляющие слова канала активных подканалов всех каналов размещаются в местной (сверхоперативной) памяти.

11.6. ДЕТЕРМИНИРОВАННАЯ МОДЕЛЬ КАНАЛА. КРИТЕРИИ ЭФФЕКТИВНОСТИ КАНАЛОВ

Большое значение, особенно для систем, работающих в реальном масштабе времени, имеет предельная оценка эффективности канала для наихудших сочетаний запросов обслуживания во времени.

Для оценки каналов ввода-вывода введем следующие критерии: а) нагрузочная способность канала, т. е. максимальное число $n_{\max}$ ПУ, способных работать одновременно с номинальной скоростью;

б) коэффициент работоспособности программы R , показывающий, какая доля времени может быть использована процессором для выполнения программы при одновременной работе данного числа ПУ.

Естественно стремиться к увеличению значений этих критериев. Однако при этом должно соблюдаться следующее ограничение, обусловленное опасностью потери информации из-за чрезмерно долгого ожидания обслуживания. Время ожидания, т. е. максимальный интервал времени t между моментом возникновения запроса на обслуживание от ПУ и моментом начала требуемой передачи информации, должно быть меньше интервала времени, в течении которого ПУ заменяет очередную порцию информации новой порцией.

Точный анализ указанных критериев в общем случае представляет собой сложную задачу, так как на характеристики канала ввода-вывода влияет множество факторов, например различный характер выполняемых процессором программ, различные сочетания одновременно приходящих от ПУ запросов и т. п. Часть параметров является случайными величинами, и учет их влияния требует применения вероятностных методов [26, 32]. Однако с помощью некоторых допущений, упрощающих задачу, можно построить детерминированную модель канала и на ее основе дать предельную оценку критериев эффективности [35].

Примем, что задержка в обслуживании запроса ПУ, другими словами, задержка доступа канала к ОП, имеет постоянную продолжительность t_0 , причем в каналах с прямым доступом к памяти она равна длительности цикла памяти, а в каналах с косвенным доступом — длительности выполнения команды.

Обозначим T_i — время подготовки в i -м ПУ передачи (прима) единицы информации (например, байта), а t_i — продолжительность сеанса связи канала с памятью при ее передаче. Если при каждой передаче единицы информации обслуживание ПУ задерживается на t_0 , то относительное время занятости канала при обслуживании i -го ПУ составляет $(t_0 + t_i)/T_i$, причем оно не должно превышать 1. Аналогично не должно превышать 1 суммарное относительное время занятости канала при обслуживании n устройств:

$$\sum_{i=1}^n \frac{t_0 + t_i}{T_i} \leq 1. \quad (11.1)$$

Нагрузочная способность канала $n_{\max}$ должна удовлетворять условию

$$\sum_{i=1}^{n_{\max}} \frac{t_0 + t_i}{T_i} \leq 1. \quad (11.2)$$

Это условие необходимое, но не достаточное. На $n_{\max}$ накладываются дополнительные ограничения, определяемые максимально допустимым временем ожидания τ_i .

Время ожидания зависит от приоритета, присвоенного ПУ. Будем считать, что приоритет определяется номером ПУ, причем младшим номерам соответствует более высокий приоритет. Рассматривая наихудший случай, когда ПУ должно ожидать выполнения передач единиц информации от всех устройств с большим приоритетом, получаем, что для i -го ПУ

$$\tau_i = t_0 + \sum_{j=1}^{i-1} t_j, \quad (11.3)$$

где t_j — продолжительность сеанса связи для j -го ПУ.

Для предотвращения потери информации из-за недопустимо долгого ожидания τ_i должно быть меньше времени T_i подготовки единицы информации в ПУ по крайней мере на время передачи информации t_i :

$$\tau_i \leq T_i - t_i$$

или

$$t_0 + t_i + \sum_{j=1}^{i-1} t_j \leq T_i.$$

Вводя t_i под знак Σ , получаем:

$$t_0 + \sum_{j=1}^i t_j \leq T_i \quad (i = 1, 2, \dots, n_{\max}). \quad (11.4)$$

Число устройств $n_{\max}$ должно удовлетворять как неравенству (11.2), так и (11.4). В общем случае при произвольном присвоении приоритета оба эти условия неэквивалентны. Для $n_{\max}$, удовлетворяющего условию (11.2), неравенства (11.4) могут и не соблюдаться, если, например, устройствам с малым T_i присвоен низкий приоритет. Однако если более быстродействующим устройствам присвоен более высокий приоритет и, следовательно,

$$T_i = \max \{T_i\} \quad (j = 1, 2, \dots, i), \quad (11.5)$$

то можно показать, что из (11.2) следует (11.4) [35].

Коэффициент работоспособности программы можно представить в виде

$$R = 1 - \sum_{i=1}^n \frac{t_i}{T_i}. \quad (11.6)$$

Рассмотрим случай, когда параметры всех устройств одинаковы, т. е.

$$\left. \begin{aligned} t_i &= \text{const} = t; \\ T_i &= \text{const} = T. \end{aligned} \right\} \quad (11.7)$$

Различия в скорости устройств можно приблизительно учесть, считая более быстродействующие устройства эквивалентными нескольким медленнодействующим. При сделанном предположении нагрузочная способность канала

$$n_{\max} = T/(t_0 + t). \quad (11.8)$$

Коэффициент работоспособности программы

$$R = 1 - nt/T \quad (\text{при } n < n_{\max}). \quad (11.9)$$

При максимальном числе устройств $n = n_{\max}$

$$R = R_{\min} = t_0/(t_0 + t). \quad (11.10)$$

Канал можно характеризовать произведением нагрузочной способности на коэффициент работоспособности программы

при $n = n_{max}$; назовем его коэффициентом добротности канала:

$$k = R_{min} n_{max}.$$

Приравнивая нулю производную k по t_0 , т. е.

$$\frac{dk}{dt_0} = \frac{d}{dt_0} \left[\frac{t_0 T}{(t_0 + t)^2} \right] = -\frac{T}{(t_0 + t)^2} - \frac{2t_0 T}{(t_0 + t)^3} = 0,$$

получаем условие максимальной добротности канала

$$t_0 = t.$$

Для такого канала

$$n_{max} = T/2t; \quad R_{min} = 1/2$$

независимо от t и t_0 .

В мультиплексном канале с прямым доступом и хранением текущих параметров подканалов в ОП задержка в обслуживании ПУ составляет $t_0 = t_{OP}$, где t_{OP} — длительность цикла ОП.

Длительность сеанса связи ПУ определяется длительностью цикла памяти, затрачиваемого на ввод-вывод слова, и временем считывания, изменения и записи управляющего слова. Если считать, что нужные изменения в регистре текущего адреса и счетчике данных производятся за время одного цикла обращения к памяти (между фазами считывания и регенерации), то для данного типа канала

$$t = 2t_{OP}.$$

В соответствии с (11.8) и (11.9) получаем

$$n_{max} = T/(t_0 + t) = T/3t_{OP}; \quad R_{min} = t_0/(t_0 + t) = 0,33.$$

Если, например, в качестве ПУ использовать устройства связи с управляемым объектом с $T = 50$ мкс, а t_{OP} считать равным 2 мкс, то канал обеспечивает одновременную работу восьми таких устройств, однако при этом около 70% машинного времени будет тратиться на обслуживание ПУ.

Для улучшения характеристик канала параметры подканалов вместо основной памяти можно хранить в специальной памяти каналов.

11.7. ЭЛЕМЕНТЫ ОРГАНИЗАЦИИ ИНТЕРФЕЙСОВ

Методы передачи информации между устройствами ЭВМ. Используются два метода передачи дискретных сигналов: синхронный и асинхронный.

При синхронном методе (рис. 11.10) передающее устройство Y_1 устанавливает одно из двух возможных состояний сигнала (0 или 1) и поддерживает его в течение определенного заранее выбранного времени, после истечения которого состояние сигнала на передающей стороне может быть изменено.

Время передачи сигнала, которое складывается из времени распространения сигнала по линии L_0 и времени распознавания и фиксации сигнала в регистре приемного устройства Y_2 , зависит от параметров линии связи и характеристик приемного и передающего устройств. Если обозначить через T максимальное время передачи сигнала (с учетом возможных наихудших условий), то для периода синхронной передачи информации должно выполняться условие

$$\tau \geq T. \quad (11.11)$$

При асинхронной передаче (рис. 11.11) устройство Y_1 устанавливает соответствующее передаваемому коду состояние сигнала на линии L_0 , а устройство Y_2 после приема нового состояния сигнала информирует об этом устройство Y_1 изменением состояния сигнала на линии L_1 . Передающее устройство, получив сигнал о приеме, снимает передаваемый сигнал. Таким образом, период τ , в течение которого передающее устройство должно поддерживать состояние сигнала, является переменным и зависит от характеристик конкретной линии связи и устройств, участвующих в передаче.

Если обозначить через t время передачи нового состояния сигнала в один конец линии связи, то при асинхронной передаче должно удовлетворяться условие

$$\tau \geq 2t. \quad (11.12)$$

Обычно время $2t$ значительно меньше времени T , которое приходится выбирать, исходя из максимально возможных расстояний между устройствами.

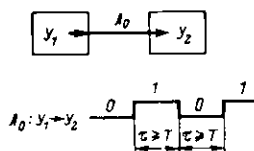


Рис. 11.10. Синхронный метод передачи информации

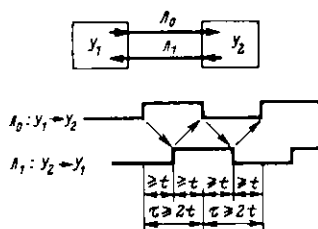


Рис. 11.11. Асинхронный метод передачи информации

Рис. 11.12. Передача информации со стробированием

При передаче параллельного кода по параллельным линиям сигналы поступят в приемное устройство в разное время из-за разброса параметров цепей, формирующих сигналы, и линий интерфейса. Максимальный разброс времени передачи

$$\Delta T = \max \{ |t_i - t_j| \}. \quad (11.13)$$

Рассмотрим два метода передачи параллельного кода по нескольким линиям: со стробированием, использующим синхронную передачу и, с квити́рованием, в котором используется асинхронная передача.

На рис. 11.12 приводится временная диаграмма для передачи со стробированием. Информация передается по линиям $L_1, \dots, L_n$ в интервале времени, когда сигнал на линии L_0 соответствует 1. При нулевом сигнале на линии L_0 сигналы на шинах $L_1, \dots, L_n$ не имеют смысла.

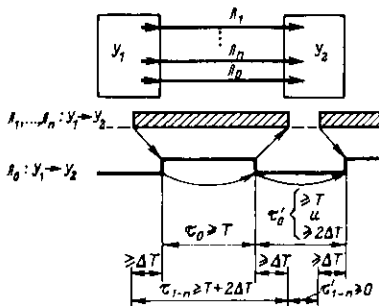
Для гарантированной передачи данных по линиям $L_1, \dots, L_n$ передаваемый код устанавливается как минимум на время ΔT раньше, чем появляется единственный сигнал на линии L_0 . Снятие передаваемого кода с линий $L_1, \dots, L_n$ может закончиться на время ΔT позже времени установления нулевого состояния на линии L_0 . На рисунке приведены необходимые временные соотношения между сигналами.

Передача параллельного кода с квити́рованием представлена на рис. 11.13.

При передаче с квити́рованием фронт сигнала на линии L_0 сигнализирует устройству U_2 , что устройство U_1 подготовило передаваемую информацию на линиях $L_1, \dots, L_n$. Фронт сигнала на линии L_{n+1} сигнализирует устройству U_1 , что устройство U_2 приняло передаваемую информацию. Восприняв фронт сигнала по линии L_{n+1} , устройство U_1 снимает информацию с линий $L_1, \dots, L_n$ и гасит сигнал на линии L_0 , срез которого сигнализирует устройству U_2 об окончании передачи данных. После снятия сигнала на линии L_0 устройство U_2 гасит сигнал на линии L_{n+1} , срез которого сигнализирует устройству U_1 о готовности устройства U_2 к приему следующей порции данных.

Передача с квити́рованием обычно используется, когда приемное устройство U_2 не всегда готово к приему информации (занято выполнением других операций). Этот способ передачи применяется в интерфейсе ввода-вывода в ЕС ЭВМ и др.

Передача со стробированием используется главным образом для пересылок информации внутри устройства, например между регистрами. В этом случае еще в процессе проектирования устройства для передачи информации можно выбрать такие моменты времени, когда участвующие в передаче элементы свободны и готовы к этой операции.



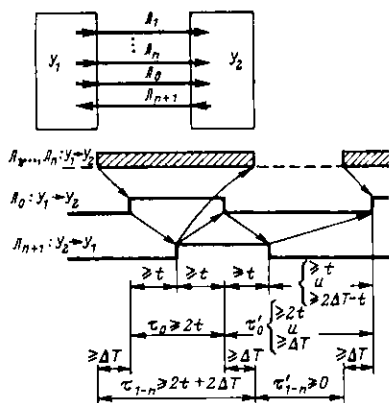


Рис. 11.13. Передача информации с кэтированием

Структуры шин и линий интерфейса. Выше были рассмотрены методы передачи информации между двумя устройствами. Однако при проектировании ЭВМ приходится решать более сложную задачу — организацию передачи информации в группе взаимосвязанных устройств. Характерным является случай централизованной связи, когда передача информации производится только между устройством Y_0 и одним из

устройств $Y_1, \dots, Y_n$. Примером является передача информации между каналом и ПУ.

При организации связи группы устройств возникает необходимость в адресации и идентификации устройств $Y_1, \dots, Y_n$. Адресация в данном случае состоит в выборе центральным устройством Y_0 одного из устройств $Y_1, \dots, Y_n$ для связи. Идентификация состоит в определении центральным устройством, какое из устройств $Y_1, \dots, Y_n$ запрашивает связь.

Адресация и идентификация устройства осуществляются путем передачи соответствующей информации по линиям интерфейса.

Различные структуры линий и шин интерфейса можно классифицировать следующим образом: индивидуальные, коллективные, комбинированные.

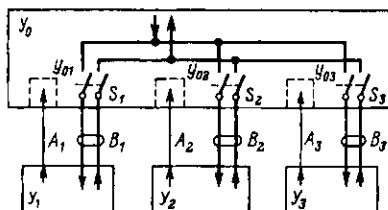
Наиболее надежной является структура с индивидуальными линиями и шинами, поскольку выход из строя одной группы линий и шин не влияет на работу других устройств. При использовании индивидуальных линий и шин упрощаются адресация и идентификация, но увеличивается количество оборудования. Индивидуальные линии и шины используются в основном для связи вычислительной машины с устройствами технологической автоматики.

Структура с коллективными шинами и линиями имеет меньшую надежность, но при необходимости организации связи с большим числом устройств такое выполнение позволяет уменьшить объем оборудования.

Применение коллективных линий и шин интерфейса возможно в том случае, если передача между различными устройствами происходит по единым установленным правилам. В интерфейсе ввода-вывода современных ЭВМ главным образом используется система коллективных линий и шин. В некоторых случаях используется комбинированная система индивидуальных и коллективных линий и шин.

На рис. 11.14 представлена структура с индивидуальными линиями и шинами. Жирными линиями изображены шины, по которым передаются данные.

Рис. 11.14. Структура с индивидуальными линиями и шинами



Центральное устройство U_0 с любым устройством U_i связывается с помощью индивидуальных линий A_i и шин B_i .

Устройство U_0 имеет переключатели S_i для подключения шин B_i . На рисунке переключатели изображены в виде электромеханических контактов, однако такие переключатели реализуются в виде электронных устройств.

Для адресации U_i устройство U_0 должно включить соответствующий переключатель S_i .

Идентификация устройства U_i осуществляется следующим образом: сначала U_i на линии A_i возбуждает сигнал требования на установление связи, затем соответствующий узел y_{0i} устройства U_0 определяет, от какого устройства пришел сигнал требования. Как только устройство U_0 будет готово к обмену информацией, замыкается переключатель S_i и начинается передача данных. Передача информации производится одним из методов, рассмотренных выше.

На рис. 11.15 представлена структура с коллективными линиями и шинами. По коллективной шине B происходит обмен информацией между U_0 и U_i , по коллективной линии A из U_i в U_0 передается сигнал требования на установление связи. Кроме того, имеется коллективная линия D , которая выходит из U_0 , последовательно проходит через устройство U_i и возвращается в устройство U_0 . При адресации U_i устройство U_0 устанавливает на шинах B код номера устройства U_i и посылает сигнал «выборки» по линии D . Если код номера на шинах B не совпадает с номером устройства U_j , то переключатель T_j остается в исходном состоянии и сигнал по линии D распространяется на следующее устройство U_{j+1} . При совпадении кода с номером устройства переключатель T_j замыкается, дальнейшее распространение сигнала по линии D прекращается, а выбранное устройство U_j соединяется с U_0 .

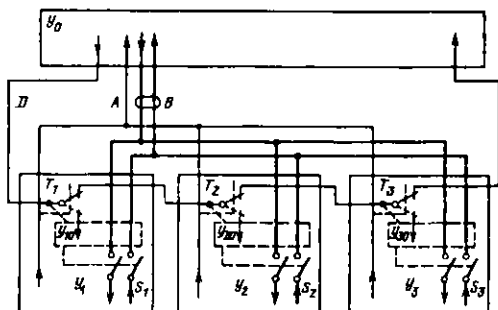


Рис. 11.15. Структура с коллективными линиями и шинами

путем замыкания переключателя S_i . Если сигнал, посылаемый по линии D , возвращается в U_0 , то это означает, что адресованное устройство U_i не найдено (обычно это свидетельствует о неисправности в работе интерфейса).

Необходимость в идентификации возникает, если устройство U_i по коллективной линии A передает сигнал запроса связи. Устройство U_0 , когда оно готово к обслуживанию U_i , посылает сигнал выборки по линии D . Устройства, не посылавшие запрос, пропускают сигнал по линии D . Первое на пути сигнала выборки устройство U_i , пославшее запрос, замыкает переключатели T_i и S_i и по шинам B посылает в U_0 код собственного номера.

Приведенная на рис. 11.15 структура коллективных линий и шин подобна используемой в интерфейсе ввода-вывода ЕС ЭВМ. Другие варианты структуры коллективных линий и шин будут приведены при рассмотрении интерфейса малых и микро-ЭВМ.

11.8. ИНТЕРФЕЙС ВВОДА-ВЫВОДА ЕС ЭВМ

В интерфейсе ввода-вывода ЕС ЭВМ применяется система коллективных линий и шин. Передача информации между устройствами производится с квитированием. Сигналы на линиях и шинах интерфейса электрически доступны всем ПУ, однако логика работы интерфейса строится таким образом, что

в каждый момент времени только одно из ПУ может быть логически связано с интерфейсом и реагировать на сигналы интерфейса. Если два или более ПУ требуют обслуживания, то из них выбирается одно в соответствии с установленным между ними приоритетом.

Подсоединение ПУ к интерфейсу показано на рис. 11.16. Периферийные устройства подсоединяются к интерфейсу через УПУ, которые могут быть групповыми (разделенными), обслуживающими

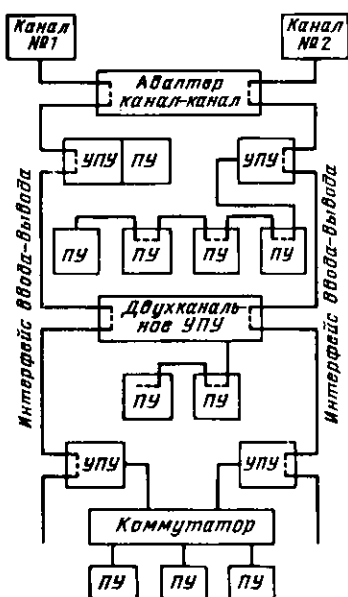


Рис. 11.16. Варианты подсоединения периферийных устройств к интерфейсу ввода-вывода (ЕС ЭВМ)

ми несколько ПУ, или индивидуальными, обслуживающими одно ПУ. Применяют также двухканальные и многоканальные групповые УПУ, способные производить ввод и вывод информации через два или более канала, принадлежащих одной или нескольким ЭВМ.

Несколько ПУ могут через коммутатор присоединяться к нескольким групповым УПУ, при этом каждое ПУ может работать с любым свободным в данный момент УПУ. Своеобразным ПУ для канала и интерфейса является *адаптер «канал — канал»*, через который осуществляется связь между каналами одной или разных ЭВМ.

Информация, передаваемая через интерфейс. Через интерфейс информация передается параллельным кодом побайтно, включая контрольный разряд. В расширенном интерфейсе (см. § 11.9) возможна двухбайтная передача. От канала через интерфейс в ПУ (точнее, в УПУ) передаются выводимые байты данных, приказы и номер (адрес) ПУ. От УПУ в канал поступают вводимые байты данных, номер (адрес) ПУ, коды состояния (*байт состояния и байты уточненного состояния*).

Коды состояния служат для опознавания процессором и каналом (и операционной системой) текущего, в том числе ненормального состояния ПУ.

Байт состояния содержит основную информацию о состоянии данного ПУ. Значения отдельных разрядов (указателей) байта состояния приведены в табл. 11.1. Указатели формируются ПУ или УПУ.

Указатель «ПУ кончилось» устанавливается при завершении операции ввода-вывода ПУ, а также при переходе ПУ из состояния «Не готов» в состояние «Готов». Периферийное устройство находится в состоянии «Не готов», если необходимо вмешательство оператора (нет перфокарт, нет бумаги, не заправлена лента в ПУ и т. п.). Указатель «УПУ кончилось» формируется (для групповых УПУ), если при обращении к нему оно было занято.

Указатель «Канал кончил» устанавливается при завершении передачи через интерфейс данных текущей операции ввода-вывода.

Указатель «Модификатор» формируется в том числе при задан-

Таблица 11.1

Номер разряда	Указатель	Номер разряда	Указатель
К	Контрольный разряд	4	Канал кончил
0	Внимание	5	ПУ кончилось
1	Модификатор	6	Сбой в устройстве
2	УПУ кончилось	7	Особый случай
3	Занято		

ном в УПУ условию окончания операции ввода-вывода (например, при поиске по ключу в ВЗУ на дисках).

Указатель «Сбой в устройстве» устанавливается при ошибках в ПУ и УПУ. Характер ошибки уточняется байтами уточненного состояния. Указатель «Особый случай» имеет определенное значение для каждого приказа и типа ПУ и служит для указания некоторого ненормального положения в ПУ.

Условия формирования указателя «Внимание» зависят от типа ПУ. При установке этого указателя прерывается выполнение цепочки УСК.

Байты уточненного состояния, содержащие информацию об ошибке во время выполнения последней операции ввода-вывода, передаются из ПУ в канал по приказу «Уточнить состояние». Число передаваемых при этом байт и значение информации зависят от типа ПУ. Обычно собственно информация уточненного состояния содержится в первых двух байтах, а в остальных находится диагностическая информация. Независимо от типа ПУ в первых шести разрядах начального (нулевого) байта расположены указатели, приведенные в табл. 11.2.

Нумерация (адресация) ПУ. Канал различает ПУ по присвоенным им номерам. Нумерация ПУ производится с помощью 8-разрядных номеров. В индивидуальном УПУ ему, а следовательно, и его ПУ может быть присвоен любой свободный номер от 0 до 255. Периферийным устройствам, подключенным к групповому УПУ (до 16 ПУ), присваивают смежные номера. Нумерация в группе всегда начинается с номера, в котором имеют нулевое значение младшие разряды, используемые для задания номеров ПУ в группе. Если к УПУ подклю-

Таблица 11.2

Номер разряда	Указатель	Условия установки указателя
0	Команда отвергнута	Периферийное устройство не может выполнять приказ (приказ запрещен для данного ПУ, неправильная последовательность приказов, оборудование, выполняющее приказ, не включено и т. п.)
1	Требуется вмешательство	Необходимы действия оператора (замятие перфокарты, нет бумаги в ПУ и т. п.)
2	Ошибка на ШИН-К	Периферийное устройство получило из канала байт с неправильным контрольным разрядом
3	Сбой в оборудовании	Сбой в логических схемах ПУ или УПУ
4	Ошибка в данных	Ошибка в данных на носителе ПУ
5	Переполнение	Канал не успел обслужить ПУ за отведенное время (пропускная способность памяти или канала меньше общей скорости передачи данных для выполняемых операций ввода-вывода)

чаются более 16 ПУ, то нумерация ПУ производится несколькими не обязательно смежными группами номеров (но смежными в группе). Периферийные устройства, которые связаны с несколькими каналами, могут иметь различные номера для различных каналов. Нумерация устройств производится с помощью встроенных в них тумблерных регистров или специальных коммутационных плат. Следует иметь в виду, что нумерация ПУ обычно не определяет приоритет между ними в использовании интерфейса. Приоритет задается порядком присоединения ПУ к интерфейсу.

Основные шины и линии интерфейса представлены на рис. 11.17. Для обозначения ПУ, связанного с каналом через интерфейс, используется термин «абонент»; соответственно сигналы, поступающие в канал из интерфейса, именуют сигналами абонента.

В интерфейсе выделяют две группы информационных шин:

1) шина прямой передачи (ШИН-К), по которой из канала в ПУ передаются выводимый слог (байт) информации, приказ и номер (адрес) ПУ;

2) шина обратной передачи (ШИН-А), по которой из ПУ в канал передаются вводимый слог (байт) информации, байты состояния и номер (адрес) ПУ.

Информация, передаваемая по информационным шинам, контролируется по нечетности. Для этого в каждой шине отводится дополнительная линия для передачи контрольного разряда (см. гл. 12).

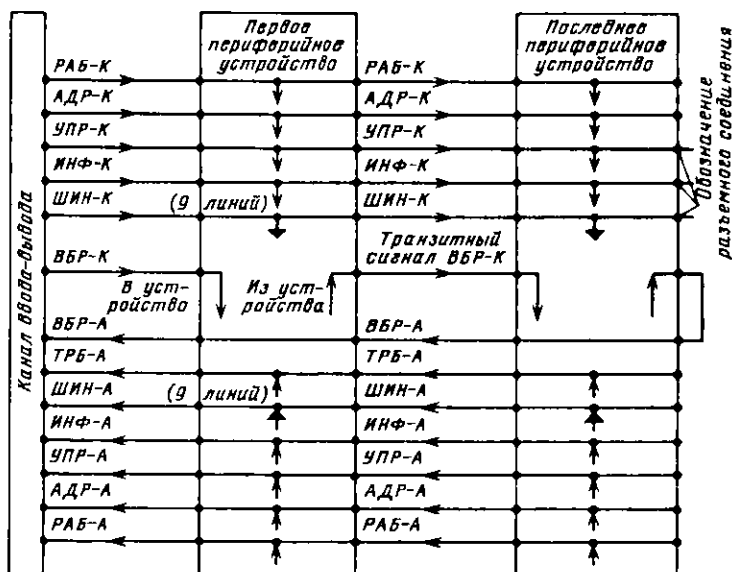


Рис. 11.17. Основные шины и линии интерфейса ввода-вывода

Смысл информации на шинах прямой и обратной передачи и промежуток времени, в течение которого информация на шинах имеет тот или иной смысл, определяются служебными сигналами, называемыми *сигналами идентификации*.

К каждой информационной шине относятся три сигнала идентификации — «Адрес», «Управление» и «Информация» — и соответствующие им линии. Сигналы идентификации от канала, относящиеся к шине прямой передачи, обозначим *АДР-К*, *УПР-К* и *ИНФ-К*. Сигналы идентификации от абонента, относящиеся к шине обратной передачи, обозначим *АДР-А*, *УПР-А* и *ИНФ-А*.

Сигналы *АДР-К*, *УПР-К* и *ИНФ-К* используются для идентификации появления на шине прямой передачи соответственно номера ПУ, приказа и байта данных; сигналы *АДР-А*, *УПР-А* и *ИНФ-А* служат для идентификации появления на шине обратной передачи соответственно номера ПУ, байта состояния и байта данных.

Кроме указанных в интерфейсе используются также следующие линии и сигналы управления:

линия и сигнал «Работа канала» — *РАБ-К*. Сигнал *РАБ-К* передается от канала к подсоединенным ПУ и используется для разрешения подключения ПУ к каналу. Сброс сигнала *РАБ-К* вызывает сброс всех сигналов абонентов и прекращение всех операций, выполняемых в это время через интерфейс;

линия и сигнал «Работа абонента» — *РАБ-А*. Линия соединяет все ПУ с каналом. Сигнал *РАБ-А* указывает каналу, что ПУ подключилось к каналу. Сигнал *РАБ-А* должен сохраняться, пока требуемая в сеансе связи передача информации между каналом и ПУ не будет завершена;

линия и сигнал «Требование абонента» — *ТРБ-А*. Линия *ТРБ-А* соединяет все ПУ с каналом. Сигнал *ТРБ-А* сигнализирует о том, что ПУ требуется установить связь с каналом. Сигналы требования могут возбудить несколько ПУ в одно и то же время. Обычно сигнал *ТРБ-А*, выданный данным ПУ, сбрасывается после выдачи этим ПУ сигнала *РАБ-А*;

линия и сигналы «Выборка канала» — *ВБР-К* и «Выборка абонента» — *ВБР-А*. Линии *ВБР-К* и *ВБР-А* образуют замкнутую цепь, которая выходит из канала (линия *ВБР-К*), последовательно проходит через все ПУ, начиная с устройства с высшим и кончая устройством с низшим приоритетом, после чего возвращается в канал в виде линии *ВБР-А*.

Все сигналы, поступающие на шины и линии интерфейса и принимаемые с интерфейса, усиливаются и формируются усилителями-передатчиками и усилителями-приемниками.

Каждое ПУ может выдать сигнал *РАБ-А* только при наличии на его входе сигнала *ВБР-К*. Если данное ПУ не участвует в выборке, оно пропускает сигнал *ВБР-К* на следующее устройство, после чего данное ПУ не может выдать сигнал *РАБ-А* до следующего поступления сигнала *ВБР-К*. В ответ на снятие сигнала *ВБР-К* снимается сигнал *РАБ-А*.

Сигнал *ВБР-К* является единственным сигналом, который недоступен сразу всем абонентам. Поступление в канал сигнала *ВБР-А* во

время выполнения процедуры начальной выборки ПУ свидетельствует об отсутствии или неисправности адресуемого ПУ.

Набор шин интерфейса, представленных на рис. 11.17, является упрощенной интерпретацией интерфейсов реальных ЭВМ. В действительности обычно имеются дополнительные линии и сигналы управления (разрешение выборки и блокировка), а также линии и сигналы для управления измерением времени и сменой состояния ПУ (измерение от канала, абонента, смена состояния).

Основные процедуры интерфейса. Основными процедурами являются *начальная выборка ПУ* и *обслуживание ПУ*. Рассмотрим в качестве примера процедуру начальной выборки ПУ для пуска операции ввода-вывода в селекторном канале (рис. 11.18).

1. Канал выдает на шину прямой передачи адрес (номер) ПУ и возбуждает сигнал *АДР-К*. Каждое ПУ, подсоединенное к каналу, дешифрирует номер, но только одно устройство опознает его как свой номер.

2. Канал выдает сигнал *ВБР-К*, который проходит последовательно по ПУ до тех пор, пока не достигает адресуемого устройства. В случае его отсутствия или неисправности в канал поступает сигнал *ВБР-А*.

3. Периферийное устройство, обнаружившее совпадение номеров, блокирует дальнейшее распространение сигнала *ВБР-К* и выдает в канал сигнал *РАБ-А*.

4. После прихода в канал сигнала *РАБ-А* канал сбрасывает сигнал *АДР-К*.

5. После снятия сигнала *АДР-К* ПУ выдает на шины обратной передачи свой номер и сигнал идентификации *АДР-А*.

6. Получив сигнал *АДР-А*, канал проверяет полученный им номер ПУ и при совпадении этого номера с заданным выдает на шину пря-

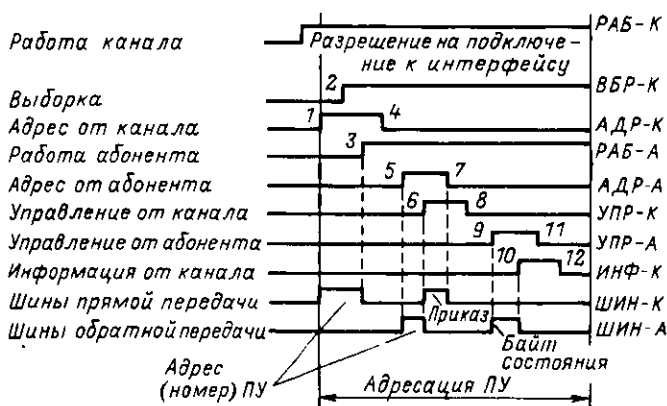


Рис. 11.18. Временная диаграмма процедуры начальной выборки периферийного устройства при пуске операции ввода-вывода в селекторном канале

мой передачи байт, представляющий собой приказ для ПУ, после чего возбуждает сигнал идентификации *УПР-К*.

7. Выбранное ПУ принимает приказ в свой регистр и сбрасывает сигнал *АДР-А*.

8. В ответ на сброс сигнала *АДР-А* канал сбрасывает сигнал *УПР-К*.

9. После снятия сигнала *УПР-К* ПУ выдает на шину обратной передачи байт состояния и формирует сигнал идентификации *УПР-А*.

10. Получив сигнал *УПР-А*, канал анализирует байт состояния ПУ. Если байт содержит нули во всех разрядах (кроме контрольного), что указывает на готовность ПУ, канал отвечает сигналом *ИНФ-К*.

11. В ответ на выдачу сигнала *ИНФ-К* ПУ сбрасывает сигнал *УПР-А*.

12. После снятия сигнала *УПР-А* канал сбрасывает сигнал *ИНФ-К*, завершая этим последовательность сигналов начальной выборки ПУ при пуске операции в селекторном режиме.

При работе канала в мультиплексном режиме после начальной выборки сеанс связи с ПУ заканчивается (канал снимает сигнал *ВБР-К*, а устройство — сигнал *РАБ-А*). Канал освобождается и может начинать процедуру начальной выборки или обслуживания других устройств. В селекторном канале после завершения начальной выборки начинается передача данных между ПУ и каналом.

Рассмотрим процедуру обслуживания ПУ при вводе (выводе) данных через мультиплексный канал (рис. 11.19).

1. Периферийное устройство, после того как оно готово к передаче информации, возбуждает сигнал *ТРБ-А*.

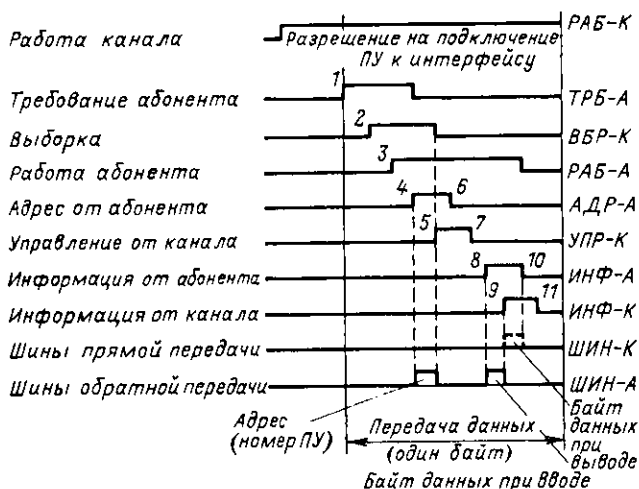


Рис. 11.19. Временная диаграмма процедуры обслуживания при вводе-выводе информации через мультиплексный канал

2. Канал, после того как сеанс по требованию ПУ окажется возможным, посылает сигнал *ВБР-К* (в этом режиме работы канал осуществляет выборку ПУ без выдачи номера ПУ на шину прямой передачи и без формирования сигнала идентификации *АДР-К*).

Появление сигнала *ВБР-К* без предварительной установки сигнала *АДР-К* является для ПУ признаком, отличающим процедуру начальной выборки (адресации) от процедуры обслуживания (идентификации).

3. Сигнал *ВБР-К* проходит последовательно по всем ПУ до тех пор, пока не достигнет первого устройства, пославшего сигнал *ТРБ-А* и ждущего сеанса связи. На этом распространение сигнала *ВБР-К* прекращается. Выбранное ПУ выдает сигнал *РАБ-А*.

4. Периферийное устройство выдает на шину обратной передачи свой номер и возбуждает сигнал идентификации *АДР-А*.

Этапы работы интерфейса 5—7 в данной процедуре такие же, как и этапы 6—8 на временной диаграмме на рис. 11.18. Отличие состоит в том, что формирование каналом сигнала *УПР-К* означает не выдачу байта приказа ПУ на шину прямой передачи, а указание ПУ продолжать работу, связанную с текущей операцией ввода-вывода. Кроме того, на этапе 5 канал после получения сигнала *АДР-А* снимает сигнал *ВБР-К*; на этапе 6 проверка номеров отсутствует, а вместо этого из памяти подканалов выбирается управляющая информация о текущих параметрах подканала. Этапы работы интерфейса 8—11 рассматриваемой временной диаграммы соответствуют этапам 9—12 на временной диаграмме рис. 11.18. Отличие состоит в том, что на шину обратной передачи выдается не байт состояния, а байт информации и вместо сигнала идентификации *УПР-А* формируется сигнал *ИНФ-А*. Кроме того, на этапе 10 наряду со снятием сигнала *ИНФ-А* гасится сигнал *РАБ-А*. На этом последовательность передачи данных через интерфейс в мультиплексном режиме завершается.

Временная диаграмма обслуживания ПУ при выводе данных отличается от рассмотренной только тем, что байт данных передается по шине прямой передачи, причем эта передача производится в интервале времени между фронтом сигнала *ИНФ-К* и срезом сигнала *ИНФ-А* (показано штриховыми линиями на рис. 11.19).

Отметим, что, как это следует из рассмотренных диаграмм, некоторые сигналы идентификации на ряде этапов работы интерфейса используются не по прямому назначению, а в качестве квитирующих сигналов.

11.9. РАЗВИТИЕ СИСТЕМЫ ВВОДА-ВЫВОДА В ЕС ЭВМ II ОЧЕРЕДИ

Повышение производительности моделей машин в ЕС ЭВМ II очереди и применение ЗУ прямого доступа на дисках со скоростью передачи информации около 1 Мбайт/с потребовали дальнейшего развития системы ввода-вывода в направлении повышения пропускной способности интерфейса ввода-вывода, повышения уровня параллелизма

в работе внешних ЗУ, придания системе ввода-вывода новых логических возможностей.

В систему ввода-вывода ЕС ЭВМ внесен ряд архитектурных новшеств, среди которых отметим следующие:

1) мультиплексирование блоков данных при работе с внешними ЗУ;

2) расширенный интерфейс ввода-вывода;

3) дополнительные команды ввода-вывода;

4) повторные выполнения управляющего слова канала;

5) селективный сброс, вводимый УПУ;

6) косвенная адресация данных в канале.

Мультиплексирование блоков данных при операциях ввода-вывода с внешними ЗУ реализуется каналами ввода-вывода особого типа — блок-мультиплексными каналами (см. § 11.5), в которых создаются несколько подканалов. Режим мультиплексирования возможен, если разряд «управление мультиплексированием» управляющего регистра 0 установлен в 1. На рис. 11.20 показан процесс мультиплексирования блоков данных при работе блок-мультиплексного канала с двумя периферийными устройствами — ПУ1 и ПУ2 [72], при этом для каждого ПУ выполняется своя канальная программа. Следует подчеркнуть, что сама передача блоков данных производится в монопольном режиме.

Расширенный интерфейс ввода-вывода. В целях повышения пропускной способности интерфейса применяется параллельная передача

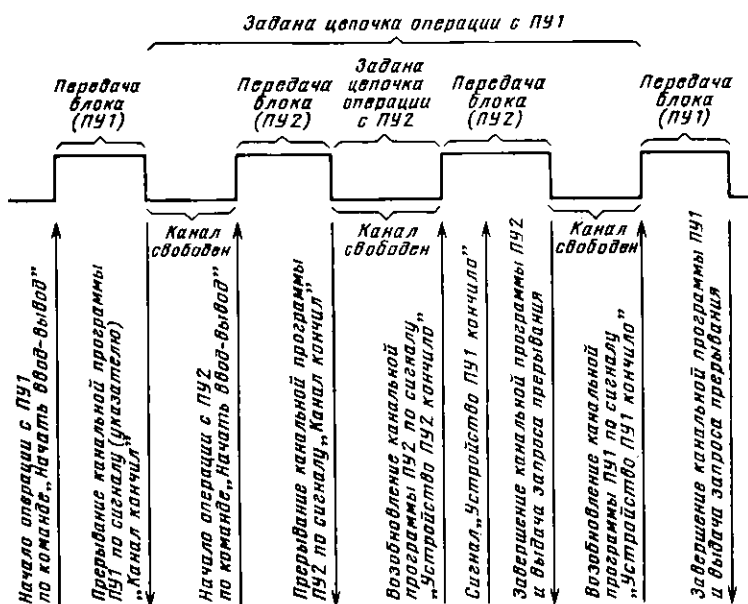


Рис. 11.20. Мультиплексирование блоков данных в блок-мультиплексном канале ввода-вывода

двухбайтных кодов и повышена частота передачи информации в интерфейсе.

Это достигнуто увеличением числа информационных шин интерфейса для передачи двухбайтных кодов и введением новых линий и сигналов управления и идентификации, позволяющих увеличить частоту передачи данных через интерфейс до 3 млн. байт/с. При этом сохраняется возможность работы с ПУ с однобайтной передачей данных через интерфейс.

Помимо имевшихся в ЕС ЭВМ I очереди шин из девяти линий для однобайтной передачи ШИН-К и ШИН-А (теперь именуемых ШИН-КО и ШИН-АО), добавлены шины из девяти линий для передачи параллельно вторых байт ШИН-К1 и ШИН-А1.

Введены линии и сигналы маркеров МРК-КО, МРК-АО, МРК-К1, МРК-А1, указывающих, на каких комплектах шин установлена информация.

Введены также дополнительные линии и сигналы идентификации ДАН-А и ДАН-К, аналогичные линиям и сигналам ИНФ-А и ИНФ-К. Сигналы ДАН-А и ДАН-К посылаются в моменты, когда сбрасываются сигналы ИНФ-А и ИНФ-К, а сигналы ИНФ-А и ИНФ-К устанавливаются, когда сбрасываются сигналы ДАН-А и ДАН-К и за этот счет увеличивается частота передачи данных по линиям интерфейса.

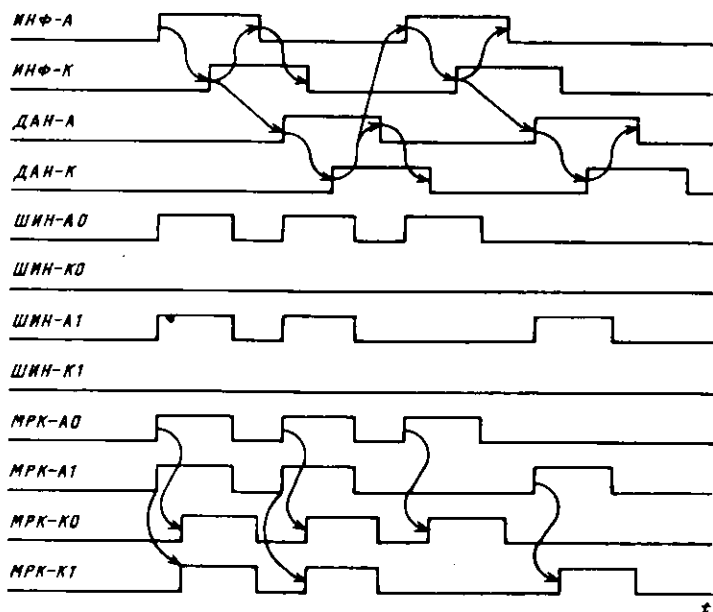


Рис. 11.21. Временная диаграмма процедуры считывания в расширенном интерфейсе ввода-вывода в ЕС ЭВМ II очереди

На рис. 11.21 приведена временная диаграмма расширенного интерфейса ввода-вывода.

Дополнительные команды ввода-вывода выполняются в привилегированном режиме, если разряд «управление мультиплексированием» управляющего регистра 0 установлен в 1. К дополнительным относятся следующие команды ввода-вывода.

„Начать ввод-вывод с быстрым отключением“. Процессор не ждет окончания процедуры начальной выборки, как это бывает при выполнении обычной команды «Начать ввод-вывод», а сразу после выборки каналом АСК освобождается и может продолжить выполнение программы.

„Освободить ввод-вывод“. Операция в указанном в команде ПУ прерывается, состояние операции в этот момент отображается в ССК, и подканал переходит в состояние «доступен». Команда быстрее переводит подканал в состояние «доступен», чем обычная команда «остановить ввод-вывод», после выполнения которой для освобождения подканала процессор должен обработать прерывание от соответствующего подканала.

„Остановить устройство“ — команда прекращает операцию только в адресованном командой ПУ, а сам канал продолжает работу с другими ПУ.

„Записать идентификатор канала“. Команда записывает в выделенные фиксированные ячейки ОП сведения о типе и состоянии адресуемого канала.

Повторение управляющего слова канала ускоряет повторение ошибочно выполняемой операции. При ошибке в операции ПУ может запросить повторение УСК, причем для повторения не требуется обработка запроса прерывания ввода-вывода. Периферийное устройство запрашивает повторение УСК выдачей соответствующего байта состояния и одновременно сигнала на линии МРК-АО.

Селективный сброс, вводимый блоком управления ПУ, позволяет прервать «зависание интерфейса», вызываемое нарушением из-за сбоев или отказов в ПУ или канале нормальной последовательности сигналов, которыми должны обмениваться канал и ПУ при операции ввода-вывода. Селективный сброс производится по запросу УПУ, выставляющего сигнал на специальной линии «отключение от абонента», в ответ на который канал выполняет «селективный сброс».

Косвенная адресация данных в канале дополняет процессорные средства реализации виртуальной памяти (динамического преобразования адресов) канальными средствами динамического преобразования адресов данных при операциях ввода-вывода. Благодаря этим средствам одно УСК может осуществлять обмен данными с несмежными страницами физической памяти.

Режим косвенной адресации (КАД) имеет место, если $УСК[37] = 1$. В режиме КАД поле текущего адреса данных в УСК ($УСК[8-31]$) указывает не адрес данных в ОП, а адрес в ОП первого слова в списке слов косвенной адресации данных. В поле текущего счетчика данных указывается число принадлежащих данному УСК слов косвенной адресации, которые размещаются в последовательных ячейках ОП. Каждое слово КАД (кроме первого) указывает физический адрес

начала блока памяти (2048 байт), а первое слово — физический адрес внутри блока. Как только при операции ввода-вывода обнуляются 11 младших разрядов физического адреса ячейки, участвующей в обмене, выбирается следующее слово КАД, которое задает начальный адрес нового блока ОП, который следует использовать в операции обмена.

11.10. СТРУКТУРА И ВРЕМЕННЫЕ ДИАГРАММЫ ИНТЕРФЕЙСА «ОБЩАЯ ШИНА» МАЛЫХ ЭВМ

К интерфейсам малых и микро-ЭВМ, в основном предназначенных для работы в системах реального времени, предъявляются повышенные требования в отношении простоты, гибкости и высокой динамичности. Для этих машин характерным архитектурным решением является общий интерфейс «общая шина» (ОШ), при котором один и тот же набор линий обеспечивает связь между процессором, основной памятью и периферийными устройствами. Для всех передач информации между любыми устройствами используются одни и те же линии, процедуры, команды, управляющие сигналы. Передача информации между устройствами выполняется в режиме разделения во времени интерфейса.

Рассмотрим интерфейс «общая шина» на примере малых ЭВМ СМ-4.

Для обеспечения гибкости и высокой динамичности в интерфейсе реализованы следующие свойства:

- наличие передач как программно-управляемых, так и внепроцессорных (прямой доступ к памяти);

- адресуемость регистров ПУ (точнее, УПУ) подобно ячейкам ОП;

- возможность присвоения устройствам ЭВМ нескольких уровней приоритета в занятии интерфейса, которые могут изменяться пересоединением линий запросов, а в процессоре — программным путем;

- совмещение по времени процедуры выбора с учетом приоритетов следующего устройства для предоставления интерфейса (*процедуры арбитража*) с обслуживанием интерфейсом предыдущего.

Упрощенная структура интерфейса «общая шина» представлена на рис. 11.22. Тонкими линиями на рисунке изображены одиночные линии, жирными — группы линий (шины). Большинство линий интерфейса является двунаправленными, т. е. по одной и той же линии сигнал может поступать в устройство и выходить из него. На рисунке в качестве примера показано, что в процессоре к двунаправленной шине присоединены как входные усилители, принимающие сигналы из интерфейса, так и выходные формирователи, посылающие сигналы в другие устройства. Подобное имеет место и в ПУ.

В каждый момент времени передача информации через интерфейс может происходить только между двумя устройствами, причем одно из них является *здатчиком* (ведущим), а другое *исполнителем*. Исполнителем может быть любое устройство, подсоединенное к интерфейсу, а *здатчиком* — любое, кроме модулей памяти. *Здатчик* получает интерфейс в свое распоряжение на время выполнения процедуры передачи данных.

Роли устройств в интерфейсе в процессе работы ЭВМ непрерывно меняются. Например, при выборе операнда из памяти процессор является задатчиком, а память — исполнителем. При получении сигнала прерывания процессор является исполнителем, а задатчиком — ПУ (источник прерывания).

Принятая в интерфейсе «общая шина» единая нумерация ячеек памяти, адресуемых регистров процессора и регистров (приемников и источников информации) периферийных устройств позволяет в операциях ввода-вывода обходиться без специальных команд, и использовать обычные адресные команды и все способы адресации. Таким образом, данные в регистрах ПУ становятся непосредственно доступными процессору без предварительной передачи их в его регистры или в оперативную память. Имеется возможность прямой передачи данных из регистра одного ПУ в регистр другого.

Адреса ячеек устройств оперативной, а также постоянной и полупостоянной памяти нумеруются в порядке возрастания, начиная с 0, а адреса регистров процессоров и ПУ — в порядке убывания, начиная с максимально возможного адреса (адреса регистров лежат в области 777777 — 760000).

«Общая шина» содержит 56 линий, которые можно разбить на четыре группы: 1) шина данных, 2) шина адреса, 3) шина арбитража

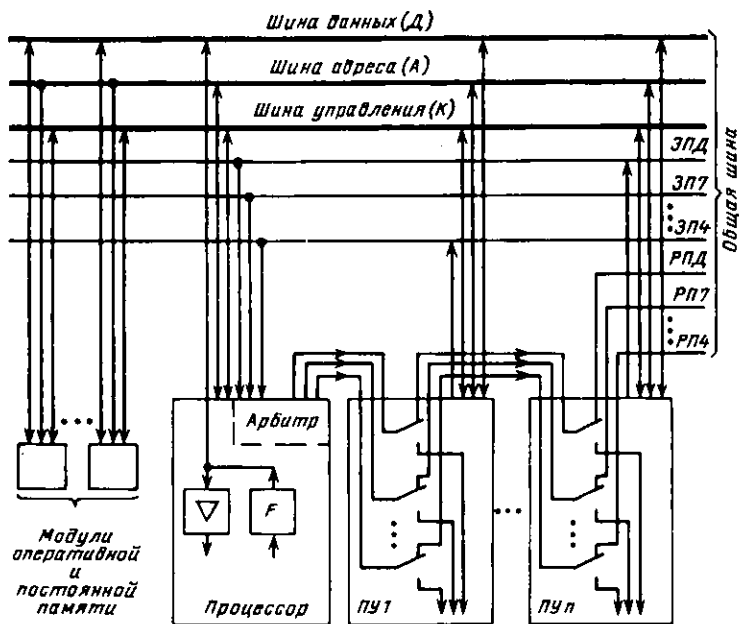


Рис. 11.22. Структура интерфейса «общая шина» (малая ЭВМ СМ-4)

приоритетов и 4) шина управления. Наименования линий и передаваемых по ним сигналов совпадают.

Шина данных (Д) включает в себя 16 двунаправленных линий для обмена 16-разрядными словами или байтами.

Шина адреса (А) содержит 18 линий.

Шина арбитража приоритетов содержит следующие линии: пять линий запроса передачи *ЗП4-ЗП7* и прямого доступа *ЗПД*, пять линий разрешения передачи *РП4-РП7* и прямого доступа *РПД*, линию подтверждения выборки *ПВБ*, линию указания занятости шин передачи данных *ЗАНЯТО*.

Шина управления (К) включает в себя линии *У0* и *У1* для задания типа приказа (в дальнейшем на временных диаграммах эти линии обозначаются как подшина *У*), линии контроля *К0*, *К1*, одиночные линии синхронизации задатчика *СХЗ*, синхронизации исполнителя *СХИ* и однонаправленную линию прерывания *ПРЕР*, передающую в процессор запросы прерывания.

Синхросигналы задатчика и исполнителя обеспечивают асинхронную передачу параллельных кодов по шинам интерфейса (метод передачи с квити́рованием).

Шина управления содержит также одинарную линию подготовки *ПОДГ*, по которой производится сброс системы в исходное состояние (сигнал поступает из процессора во все устройства) и двунаправленные линии состояния (аварии) сети питания переменного тока *АСП* и состояния (аварии) источника питания постоянного тока *АИП*. Сигналы *АСП* и *АИП* выдаются источником питания во все устройства, присоединенные к интерфейсу.

Приказы в интерфейсе. В интерфейсе выполняются операции ввода-вывода: считывание (передача от исполнителя к задатчику) и запись (передача от задатчика к исполнителю). Тип передачи в интерфейсе задается приказом, который задатчик посылает исполнителю. В интерфейсе «общая шина» возможны четыре приказа (операции): 1) «*Запись байта*» *ЗПБ* ($У1 = 1$, $У0 = 1$); 2) «*Запись слова*» *ЗПС* ($У1 = 1$, $У0 = 0$); 3) «*Чтение слова*» *ЧТС* ($У1 = 0$, $У0 = 0$); 4) «*Чтение слова с паузой*» — блокировкой регенерации прочитанного из ячейки слова *ЧТСП* ($У1 = 0$, $У0 = 1$). За операцией *ЧТСП* должен следовать приказ *ЗПС* или *ЗПБ* с записью в ту же ячейку.

Приоритеты устройств на занятие интерфейса. Поскольку одновременно у нескольких устройств может возникнуть необходимость в передаче данных, существует система приоритетного выбора устройства, которому разрешается занятие интерфейса. Система приоритетов должна быть достаточно гибкой, многоуровневой и перестраиваемой, с тем чтобы облегчить применение малых ЭВМ для управления в реальном времени (см. гл. 9).

В интерфейсе «общая шина» реализуются пять уровней при-

оритета, присваиваемых ПУ путем подсоединения его к одной из пяти линий запроса $ЗП4-ЗП7$ или $ЗПД$ (перечислены в порядке возрастания приоритета). Процессор может иметь любой уровень приоритета (от четырех до семи), устанавливаемый программой путем засылки соответствующего кода в регистр порога прерывания.

Приоритет присваивания ПУ в зависимости от типа и характеристик устройства или от важности выполняемой операции ввода-вывода. Наибольший приоритет обычно присваивается во избежание потерь информации ЗУ на дисках, лентах, барабанах, осуществляющих обмен блоками с прямым доступом к памяти.

К одной линии запроса могут быть присоединены (по схеме ИЛИ) несколько ПУ. В этом случае приоритет между ними определяется порядком присоединения к соответствующей линии разрешения передачи, последовательно проходящей через все ПУ данного уровня приоритета.

Выбор задатчика — кандидата на получение интерфейса производится специальным блоком процессора — арбитром.

Арбитр игнорирует запросы с линий с приоритетами, меньшими текущего приоритета процессора. При поступлении запросов более высокого приоритета, чем у процессора, арбитр определяет наиболее приоритетный из них и возбуждает соответствующую линию разрешения передачи. Первое из устройств на пути распространения сигнала разрешения, пославшее запрос, прекращает распространение сигнала и становится кандидатом в задатчики. Это устройство фактически станет задатчиком после снятия текущим задатчиком сигнала занятости интерфейса.

Наличие отдельной от шины передачи данных шины арбитража делает возможным совмещение процедур передачи данных через интерфейс и выбора следующего задатчика.

Перейдем к рассмотрению процедур и временных диаграмм арбитража и передачи данных в интерфейсе «общая шина».

Процедура и временная диаграмма арбитража [54, 85]. В интерфейсе «общая шина» возможны два вида передач — путем прямого доступа к памяти (внепроцессорная передача) и программно-управляемая.

На рис. 11.23 представлена объединенная временная диаграмма процедуры арбитража для запросов передачи при программном прерывании $ЗП_i$ и прямого доступа к памяти (внепроцессорного обмена) $ЗПД$. На временной диаграмме слева указаны наименования линий (шин) и сигналов, справа — устройства, пославшие сигналы. Заштрихованы сигналы, выдаваемые ПУ — претендентом на интерфейс.

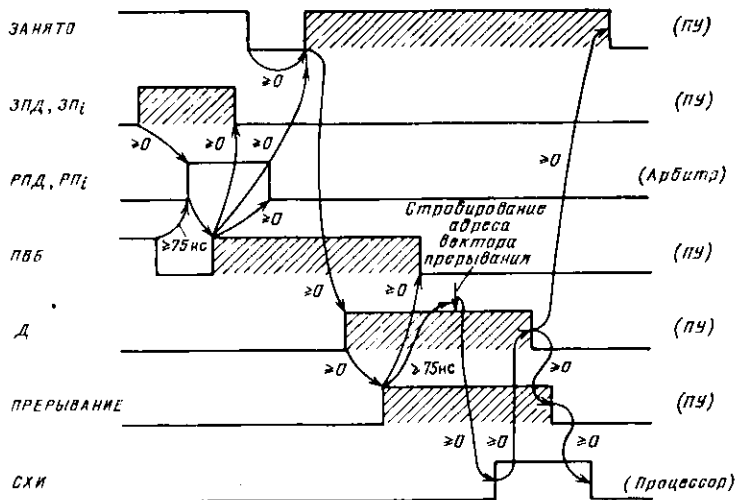


Рис. 11.23. Временная диаграмма процедуры арбитража при запросах программно-управляемого обмена (при программном прерывании) и прямого доступа

Более сложной является процедура арбитража при обработке запросов ПУ на программно-управляемую передачу. В этой процедуре участвуют ПУ, арбитр и процессор. Начнем рассмотрение с этой процедуры.

1. Периферийное устройство, претендующее на занятие интерфейса для программно-управляемого обмена, выставляет сигнал прерывания $ЗП_i$ на закрепленную за ним линию i -го уровня приоритета (на диаграмме принято, что в этот момент шина передачи данных занята другим периферийным устройством, установившим сигнал на шине $ЗАН$).

2. Арбитр, получив сигнал $ЗП_i$, сравнивает его уровень i с текущим уровнем приоритета процессора. Если уровень i оказывается равным или ниже, запрос игнорируется. В противном случае арбитр при условии, что прошло более 75 нс с момента снятия сигнала $ПВБ$ задатчиком, занимавшим интерфейс, выдает разрешение на прерывание посылкой сигнала $РП_i$. Этот сигнал блокирует выдачу арбитром других разрешений на прерывание.

3. Сигнал $РП_i$ воспринимается первым на пути его распространения ПУ из числа выставивших запросы на линию $ЗП_i$. В ответ это ПУ выдает сигнал $ПВБ$ и затем сбрасывает сигнал $ЗП_i$, если производится передача одного слова.

4. Арбитр в ответ на сигнал *ПВБ* сбрасывает сигнал *РП_i*.

5. Периферийное устройство воспринимает сброс сигнала *РП_i* как разрешение на занятие шины передачи данных интерфейса после ее освобождения.

После сброса предыдущим задатчиком сигнала *ЗАН* ПУ выставляет свой сигнал на линию *ЗАН* и становится задатчиком.

6. Задатчик выставляет на шину *Д* адрес вектора прерывания.

7. Задатчик при сброшенном сигнале *СХИ* посылает сигнал *ПРЕРЫВАНИЕ*.

8. Процессор и арбитр воспринимают сигнал *ПРЕРЫВАНИЕ*. Процессор через время, равное или большее 75 нс, принимает адрес вектора прерывания с шины *Д* и выдает сигнал *СХИ*.

9. Задатчик по получении сигнала *СХИ* снимает с шины *Д* адрес вектора прерывания, сбрасывает сигнал *ПРЕРЫВАНИЕ* и освобождает интерфейс, сбрасывая сигнал *ЗАНЯТО*.

После этого запрошенная передача данных реализуется под управлением процессора, выполняющего прерывающую программу ввода-вывода, инициируемую вектором прерывания. Перед переходом к этой программе процессор запоминает в стеке состояние прерванной программы и свой текущий уровень приоритета.

В процедуре арбитража при обработке запроса прямого доступа к памяти взаимодействуют только арбитр и ПУ, выставившее сигнал *ЗПД*, имеющий самый высокий уровень приоритета (всегда выше уровня приоритета процессора). В этом случае процедура арбитража ограничивается пп. 1–5 только что рассмотренной процедуры. Став задатчиком (п. 5), устройство, пославшее запрос *ЗПД*, берет на себя управление обменом.

Процедура и временная диаграмма для операций считывания. На рис. 11.24 представлена временная диаграмма для операций считывания слова (приказы ЧТС и ЧТСП) (заштрихованы сигналы задатчика). Выполняется следующая процедура:

1. Задатчик, выставив сигнал *ЗАН*, получает в свое распоряжение шину передачи данных интерфейса.

2. Задатчик выставляет на шинах *А* и *У* соответственно адрес ячейки ОП (или регистра устройства), участвующей в обмене, и приказ, определяющий тип передачи.

3. Задатчик с задержкой не менее 150 нс при условии снятия сигнала *СХИ* от процедуры предыдущей передачи устанавливает синхронизирующий сигнал *СХЗ*, указывающий всем

11.11. ОСОБЕННОСТИ ИНТЕРФЕЙСА «МУЛЬТИШИНА» МИКРОПРОЦЕССОРОВ И МИКРО-ЭВМ

В современных микропроцессорах и микро-ЭВМ широко используется интерфейс «мультишина» (интерфейс И41), в котором обмен данными осуществляется асинхронно по схеме «задатчик-исполнитель». Интерфейс «мультишина» применяется, например, в микро-ЭВМ СМ-1800, в устройствах и системах, в которых используются микропроцессоры К580.

Интерфейс «мультишина», являясь модификацией интерфейса «общая шина», по сравнению с последним обладает большими логическими возможностями.

Интерфейс «мультишина» может работать как с 8-, так и с 16-разрядными микропроцессорами, имеются средства расширения адресного пространства интерфейса. Анализ приоритетов запросов прямого доступа к памяти и запросов прерывания для программно-управляемого (процессорно-управляемого) обмена данными производится отдельными устройствами (соответственно «арбитром» и «блоком приоритетного прерывания»), благодаря чему достигаются упрощение и ускорение процедур арбитража запросов прямого доступа.

Имеются варианты выполнения схем арбитража и процедур прерывания, которые могут выбираться в зависимости от назначения проектируемого микропроцессорного оборудования.

На рис. 11.25 представлена упрощенная структура интерфейса «мультишина». Всего в интерфейсе 72 линии, которые подразделяются на три основные шины: адреса (16 линий для адресации памяти емкостью 64 Кбайт, плюс 4 резервные), данных (8 двунаправленных линий плюс 8 резервных, используемых при работе с 16-разрядным микропроцессором), управления (36 линий).

Адрес, выставленный задатчиком на шине адреса, определяет исполнителя, при этом шина адреса может мультиплексироваться на обслуживании той или иной группы устройств и общее рабочее адресное пространство интерфейса расширяется до 128 Кбайт.

В интерфейсе выполняются операции (приказы) «Чтение» и «Запись» (с устройствами памяти), «Ввод» и «Вывод» (с периферийными устройствами).

В шине управления можно выделить несколько подшин и отдельных функциональных линий.

Линия и сигнал инициализации используются для установки начального состояния.

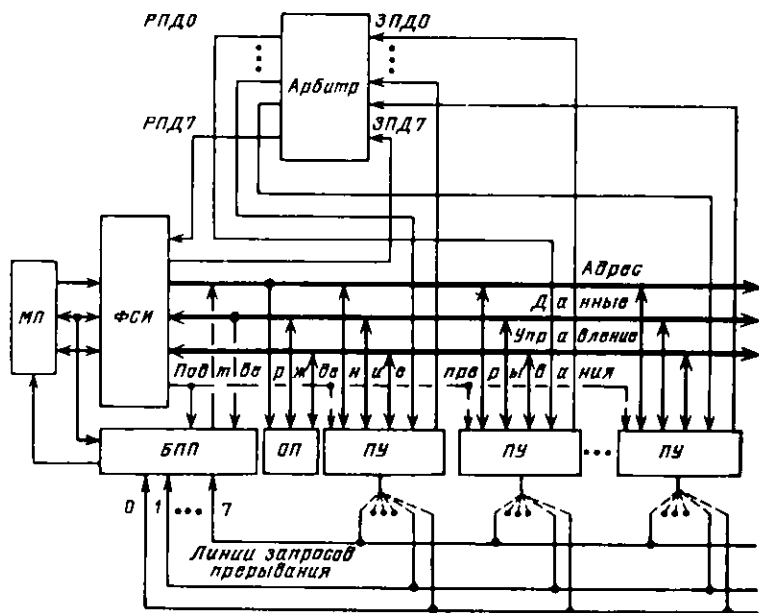


Рис. 11.25. Упрощенная структура интерфейса «мультишина»:
ФСИ – схема формирования сигналов интерфейса; БПП – блок приоритетного прерывания

Линия и сигнал управления разрядностью шины данных (8 разрядов при работе с МП К580 и 16 или 8 при работе с МП К1810).

Подшина управления передачей данных содержит линию «Подтверждение», по которой исполнитель посылает одноименный сигнал, и линии «Чтение», «Запись», «Ввод» и «Выход». Задатчик, выставя сигнал на одну из этих линий, задает тот или иной приказ. Этот сигнал задатчика, определяющий вид приказа и тем самым указывающий, адресуется устройство памяти или регистры периферийного устройства, используется для мультиплексирования шины данных на работу с устройствами памяти или с периферийными устройствами. Сигнал «Подтверждение» исполнителя и сигнал задатчика, назначающий приказ, участвуют в управлении передачей данных с квитированием аналогично сигналам СХИ и СХЗ в «общей шине».

Линии и сигналы «Запрет ОП» и «Запрет ПЗУ» служат для мультиплексирования шины адреса для работы с ОП и ПЗУ.

Подшина защиты от сбоев и отказов в источнике питания (пять линий). При сбоях и отказах в системе питания сигналы,

передаваемые по этой подшине, инициируют процедуры сохранения информации в памяти и переключения на резервное питание.

Подшины управления прерыванием и арбитражи на рис. 11.25 выделены из шины управления и показаны соответственно в нижней и верхней частях рисунка.

Подшина управления прерыванием содержит восемь линий (уровней запросов прерывания ЗП и линию подтверждения прерывания (число уровней прерывания может увеличиваться путем присоединения дополнительных блоков приоритетного прерывания).

В интерфейсе реализуются две процедуры прерывания для запросов программно-управляемого обмена: с внеинтерфейсным формированием адреса вектора прерывания (формируется блоком приоритетного прерывания БПП) и с векторным прерыванием, при котором источник запроса прерывания составляет на шине данных адрес своего вектора прерывания.

Первая процедура описана в § 9.5. Пока к уровням (линиям запросов) подключено по одному источнику запросов прерывания, процедура прерывания производится достаточно быстро. Однако она значительно замедляется при возрастании числа источников запросов и подключении нескольких источников к одному уровню прерывания, так как необходимо программным путем определять источник, выставивший запрос прерывания.

В подобных случаях более быстродействующим оказывается векторное прерывание, при котором используются специальный блок прерываний (контроллер прерывания — КП) и показанные на рис. 11.25 прерывистыми линиями дополнительные связи.

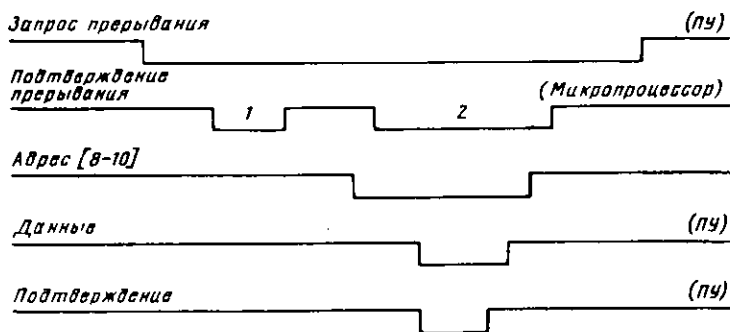


Рис. 11.26. Временная диаграмма векторного прерывания в интерфейсе «мультишина»

На рис. 11.26 представлена временная диаграмма векторного прерывания¹. Устройство, которому понадобилось прервать работу процессора, выставляет сигнал ЗП на одну из линий запроса прерывания. Получив запрос, контроллер прерываний выдает сигнал прерывания процессору, на что процессор отвечает двумя сигналами «Подтверждение прерывания». Первым из этих сигналов микропроцессор подтверждает прием сигнала прерывания, фиксирует состояние контроллера прерываний и оставляет за собой управление интерфейсом, чтобы гарантировать корректность последующей процедуры, выполняющейся по второму сигналу «Подтверждение прерывания».

Контроллер прерываний (КП) выставляет на шину адреса (разряды 8–10) код номера обрабатываемого запроса. Устройство, опознавшее номер «своего» запроса, выставляет на шину данных адрес вектора прерывания, поступающий в процессор, который приступает к обработке прерывающей программы (см. § 9.5).

Процедура арбитража производится по отношению к запросам прямого доступа памяти и выполняется схемой, называемой арбитром. В рассматриваемом интерфейсе реализуются два варианта арбитража: параллельный и последовательный.

В процедурах параллельного арбитража, схема которого представлена в верхней части рис. 11.25, используются восемь линий запросов ЗПД и соответственно восемь линий разрешения (РПД) прямого доступа.

При арбитраже каждое устройство выставляет запрос ЗПД на собственную линию запроса. Арбитр сравнивает приоритеты поступающих запросов и выдает сигнал РПД самому приоритетному устройству по индивидуальной линии. В этом случае время выполнения процедуры арбитража может быть существенно уменьшено, так как оно определяется только скоростью работы арбитра и не зависит от числа ПУ. Вместе с тем при этом способе организации арбитража требуются дополнительные аппаратные затраты и ограничивается число устройств в системе числом входов и выходов схемы арбитра.

Более просто реализуется «последовательный» арбитраж (рис. 11.27). В этой процедуре сигналы запросов прямого доступа отсутствуют и становятся ненужными не только линии запросов, но и сам арбитр.

На входе самого приоритетного ПУ всегда присутствует «Входной сигнал разрешения», который последовательно транс-

¹ На рис. 11.26 и 11.28 действующему значению сигнала соответствует низкий уровень потенциала.

лируется ко входам всех устройств, подключенных к интерфейсу. Порядок подключения определяет приоритет устройства. Обычно микропроцессор имеет самый низкий приоритет.

Любое ПУ может занять интерфейс, если на его входе присутствует «Входной сигнал разрешения». При этом оно сбрасывает «Выходной сигнал разрешения», запрещая тем самым устройствам с более низким приоритетом запрашивать прямой доступ. Число подключаемых ПУ ограничивается временем распространения сигнала по линии разрешения прямого доступа.

«Мультишина» позволяет воспользоваться и компромиссным решением, так называемым последовательно-параллельным арбитражем, при котором к линии запросов i -го уровня схемы параллельного арбитража подключаются k устройств, соединенных между собой по схеме последовательного арбитража.

Иногда требуется смена уровня приоритета устройства в процессе работы. Одним из простых способов динамического изменения приоритета является циклическая смена уровней приоритетов устройств после каждого обращения к интерфейсу.

В интерфейсе ОШ процедура арбитража выполняется асинхронно. Это позволяет удалять ПУ на значительные расстояния, но существенно снижает быстродействие системы прерывания. В интерфейсе «мультишина» процедура арбитража синхронизируется синхросигналами.

Интерфейс «мультишина» рассчитан на микромашины с короткими межмодульными связями. Это позволяет выполнять смену задатчика на шине синхронно со специальными синхросигналами.

На рис. 11.28 приведена временная диаграмма обмена данными в интерфейсе «мультишина» в режиме прямого доступа при параллельном арбитраже.

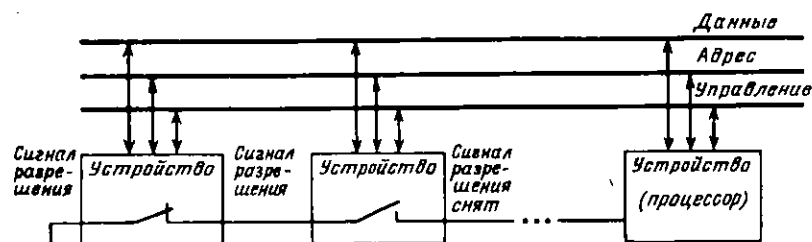


Рис. 11.27. «Последовательный» арбитраж

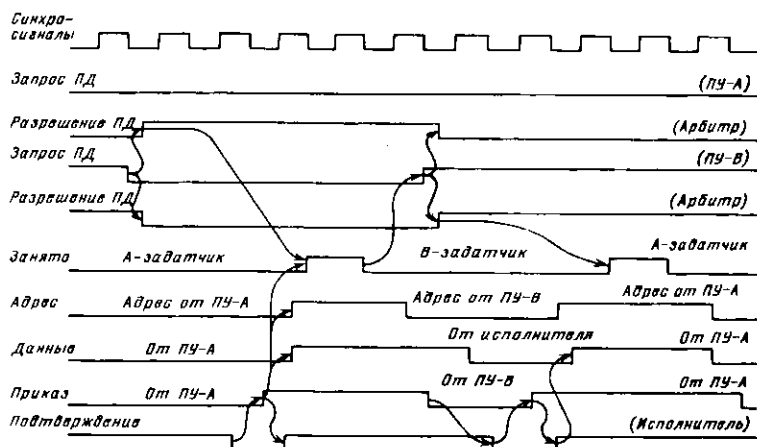


Рис. 11.28. Временная диаграмма смены задатчика и обмена данными в интерфейсе «мультишина» (параллельный арбитраж)

В исходном состоянии устройство *A* занимает интерфейс. Затем устройство *B*, имеющее более высокий приоритет, запрашивает прямой доступ, выставляя по срезу синхросигнала сигнал запроса прямого доступа на индивидуальную линию запроса. Поскольку устройство *B* имеет более высокий приоритет, чем устройство *A*, арбитр до следующего фронта синхросигнала выдает устройству *B* сигнал «Разрешение», сбрасывая одновременно сигнал «Разрешение» на входе устройства *A*. По сбросу сигнала «Разрешение» устройство *A* должно завершить выполнение приказа (на рис. 11.28 устройство *A* выполняет приказ «Запись» для текущего байта), при этом сбрасываются сигнал, задающий приказ, адрес исполнителя на шине адреса, данные на шине данных. По ближайшему срезу синхросигнала устройство *A* освобождает интерфейс, сбрасывая сигнал «Занято», а по следующему срезу синхросигнала устройство *B*, выставляя сигнал «Занято», занимает интерфейс, становится задатчиком и выдает приказ «Чтение».

СИСТЕМЫ АВТОМАТИЧЕСКОГО КОНТРОЛЯ И ДИАГНОСТИРОВАНИЯ ЭВМ

12.1. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ НАДЕЖНОСТИ ЭВМ. ФУНКЦИИ СИСТЕМ КОНТРОЛЯ И ДИАГНОСТИРОВАНИЯ

Согласно современным представлениям под надежностью понимается свойство изделия (элемента, узла, устройства, машины, системы) выполнять заданные функции, сохранять свои характеристики в установленных пределах при определенных условиях эксплуатации.

Надежность вычислительной машины определяется безотказностью, достоверностью функционирования и ремонтнопригодностью.

Безотказность есть свойство машины или системы, характеризующееся закономерностями возникновения отказов. Под *отказом* понимается событие, заключающееся в полной или частичной утрате машиной (системой) работоспособности. Отказ ЭВМ — это такое нарушение ее работоспособности, для восстановления которой требуются определенные действия обслуживающего персонала по ремонту, замене и регулировке неисправного элемента, узла или устройства. Безотказность может измеряться *средним временем наработки машины на один отказ*.

Ремонтпригодность есть степень приспособленности машины или системы к предупреждению, обнаружению и устранению отказов. Ремонтпригодность определяет потерю работоспособности машины вследствие необходимости производить устранение неисправностей и может измеряться *средним временем устранения неисправности*.

Работа ЭВМ заключается в выполнении преобразований информации, основными из которых являются передача информации в пространстве (между отдельными блоками и устройствами машины), хранение информации (передача информации во времени), арифметические и логические преобразования.

В силу указанной специфики рабочего процесса надежность ЭВМ наряду с безотказностью определяется также достоверностью функционирования. *Достоверность функционирования* есть свойство машины (системы), определяемое безошибочностью производимых машиной (системой) преобразований ин-

формации и характеризуемое закономерностями появления ошибок из-за сбоев.

Сбоем называют кратковременное самоустраниющееся нарушение нормального функционирования машины вследствие кратковременного воздействия на некоторый элемент (или элементы) внешних помех или изменений некоторых других входных воздействий, а также из-за кратковременного изменения параметров элементов (кратковременные нарушения контактов и т. п.). После сбоя машина длительное время может работать нормально. Сбой сопровождается искажением информации при операциях передачи, хранения или обработки ее. Следовательно, если не устранить последствия боя, то задача может оказаться неправильно решенной из-за искажений в данных, промежуточных результатах или в самой программе.

Однако если при отказе для восстановления работоспособности машины или системы необходимо устранить неисправность в аппаратуре, то при сбое требуется восстановить лишь достоверность информации, что хотя и связано с потерями рабочего времени ЭВМ (например, на повторный пуск программы или ее части), но не требует ремонта или регулировки аппаратуры. В силу этого восстановление достоверности функционирования сравнительно легко может быть автоматизировано.

Достоверность функционирования ЭВМ можно оценить *средним временем наработки машины на один сбой*. Для более полной оценки достоверности функционирования введем в состав характеристик надежности ЭВМ *среднее время восстановления достоверности информации после сбоя*.

В процессе развития вычислительной техники повышение безотказности ЭВМ достигалось использованием более надежных элементов (переход от ламповых схем к полупроводниковым и затем к интегральным микросхемам), применением облегченных нагрузочных режимов для схемных компонентов, совершенствованием конструкции и технологии (печатный, в том числе многослойный монтаж, улучшение вентиляции блоков машин, автоматизация проектирования, изготовления и контроля узлов и др.) и соответствующими логическими решениями (см. § 15.6).

Для уменьшения вероятности сбоев принимаются меры для уменьшения помех в цепях электронных схем: согласование нагрузок электронных схем, специальные методы монтажа и выполнения заземлений схем.

Пользователь должен быть уверен в правильности производимых машиной расчетов, особенно при работе ЭВМ в реаль-

ном времени с выдачей управляющих воздействий на объект управления.

Если вычислительная машина не обеспечивает пользователя средствами контроля достоверности ее функционирования, он вынужден непроизводительно затрачивать машинное время на двойной просчет, решение контрольных вариантов и т. д.

Выбор методов повышения надежности ЭВМ и их эффективность в значительной мере зависят от того, является вычислительная машина восстанавливаемой или невосстанавливаемой, обслуживаемой или необслуживаемой системой.

Система называется *восстанавливаемой*, если во время эксплуатации может производиться ремонт для устранения возникающих отказов. Система считается *обслуживаемой*, если допускается периодическое проведение профилактических испытаний для выявления элементов и узлов, параметры которых близки к предельно допустимым. Целью профилактических испытаний является увеличение среднего времени наработки на отказ в период между профилактическими работами.

В большинстве случаев ЭВМ представляют собой в терминах теории надежности восстанавливаемые обслуживаемые системы. К невосстанавливаемым системам относятся, например, бортовые вычислительные машины ракет.

Профилактическое обслуживание связано с потерями рабочего времени машины и затратами труда обслуживающего персонала. Эти потери по своему характеру близки к потерям, связанным с устранением отказов, и они должны учитываться наряду с показателями надежности устройств машины при назначении периодов и объема профилактического обслуживания.

Очевидно, что чем выше надежность ЭВМ, тем больше может быть период между профилактическими работами. Чем меньше тратится машинного времени и квалифицированного труда на профилактические работы и устранение неисправностей, тем выше степень *обслуживаемости* ЭВМ, т. е. степень приспособленности машины к процессам обслуживания.

С целью оценки совокупного влияния на работу ЭВМ рассмотренных выше отдельных показателей надежности введем *комплексный коэффициент эксплуатационной надежности (комплексный коэффициент использования)*

$$K_{\text{И}} = \frac{\sum_{i=1}^n t_i - \sum_{r=1}^m \tau_{\text{в.с.р.}} - \tau_k - \sum_{s=1}^e \tau_{\text{пф.с.}}}{\sum_{i=1}^n t_i + \sum_{j=1}^m \tau_{\text{в.о.}}}, \quad (12.1)$$

где n , m , e — соответственно число отказов, сбоев и профилакти-

тических обслуживаний за достаточно большой период эксплуатации; t_i — интервал времени исправной работы машины (системы) между $(i-1)$ -м и i -м нарушениями функционирования машины из-за отказов; $\tau_{в.о./j}$ — время восстановления после j -го отказа; $\tau_{в.с./r}$ — время восстановления достоверности информации после r -го сбоя (время, потраченное на повторный пуск программы, части программы, команды и т. д.); τ_k — суммарное машинное время, затраченное в рассматриваемый период на контроль достоверности (на двойной просчет, контрольные варианты, работу схем контроля и т. д.); $\tau_{пф. s}$ — время, затраченное на s -е профилактическое обслуживание.

Приведенное выражение позволяет сделать важные выводы. Чтобы уменьшить потери от сбоев и отказов, порождающих ошибки, надо предотвратить распространение ошибки в вычислительном процессе, так как в противном случае существенно усложнятся и удлинятся процедуры проверки правильности работы программы, определения и устранения искажений в программе, данных и промежуточных результатах.

Для этого необходимо обнаруживать появление ошибки в выполняемых машиной преобразованиях информации возможно ближе к моменту ее возникновения. С этой целью надо иметь *систему автоматического контроля* правильности работы ЭВМ, которая при появлении ошибки в работе машины немедленно приостанавливает выполнение программы. Наличие такой системы освобождает пользователя от забот по контролю достоверности и снижает связанные с этим потери $[\tau_k$ в (12.1)].

Для уменьшения значения второго члена в числителе (12.1) следует иметь *систему автоматического восстановления вычислительного процесса*, распознающую характер (сбой или отказ) ошибки и при сбое автоматически восстанавливающую достоверность информации и выполнение программы, а при отказе извещающую обслуживающий персонал о необходимости ремонта машины.

Для повышения комплексного коэффициента использования необходимо повышать обслуживаемость машины и добиваться уменьшения потерь времени на устранение отказов (повышение ремонтпригодности) и на проведение профилактических работ. Эти потери времени в таких сложных объектах, как ЭВМ, в первую очередь связаны с поиском места неисправности. Важнейшим средством уменьшения указанных потерь и повышения обслуживаемости ЭВМ является *система автоматического диагностирования*, позволяющая локализовать неисправность.

Обнаружение ошибок должно производиться в машине непрерывно и, следовательно, не должно вызывать заметного снижения быстродействия машины. Поэтому эта функция возлагается обычно на быстродействующие аппаратные средства контроля, которые позволяют почти полностью совместить во времени выполнение основных и контрольных операций.

Необходимость в коррекции ошибок, восстановлении вычислительного процесса и диагностировании неисправностей при современном уровне надежности ЭВМ возникает достаточно редко. Поэтому целесообразно использовать для выполнения этих функций главным образом микропрограммные, а также программные средства в виде корректирующих и диагностических микропрограмм и программ. Однако чтобы эти программы не были чрезмерно сложны, предусматриваются и определенные аппаратные средства, поддерживающие процедуры восстановления после сбоев и локализации неисправностей.

Основными характеристиками системы автоматического контроля правильности функционирования ЭВМ являются: а) отношение количества оборудования, охваченного системой контроля, к общему количеству оборудования ЭВМ; б) вероятность обнаружения системой контроля ошибок в функционировании ЭВМ; в) степень детализации, с которой система контроля указывает место возникновения ошибки¹; г) отношение количества оборудования системы контроля к общему количеству оборудования ЭВМ.

Основными характеристиками систем автоматического диагностирования являются: а) вероятность правильного обнаружения места отказа; б) разрешающая способность, равная среднему числу подозреваемых сменных блоков; в) среднее время диагностирования неисправности; г) доля аппаратных средств системы диагностирования в общем оборудовании ЭВМ.

Для снижения затрат машинного времени и труда на профилактические испытания следует снабжать ЭВМ *аппаратурно-программными средствами автоматизации испытаний*.

При создании новых ЭВМ разработка вопросов их технического обслуживания, создания аппаратурно-программных средств повышения надежности функционирования и поддержки эксплуатационного обслуживания ЭВМ, в первую очередь

¹ Детальное распознавание места неисправности является задачей системы диагностирования (см. § 12.5).

систем автоматического контроля, восстановления и диагностирования, должна вестись параллельно и взаимосвязанно с проектированием машины

12.2. КОНТРОЛЬ ПЕРЕДАЧИ ИНФОРМАЦИИ

При контроле передачи информации наибольшее распространение получили методы информационной избыточности, использующие коды с обнаружением и коррекцией ошибок.

Если длина кода n разрядов, то таким двоичным кодом можно представить максимум 2^n различных слов. Если все разряды слова служат для представления информации, код называется простым (неизбыточным). Коды, в которых лишь часть кодовых слов используется для представления информации, называются избыточными. Часть слов в избыточных кодах является запрещенной, и появление таких слов при передаче информации свидетельствует о наличии ошибки.

Принадлежность слова к разрешенным или запрещенным словам определяется правилами кодирования, и для различных кодов эти правила различны.

Коды разделяются на равномерные и неравномерные. В равномерных кодах все слова содержат одинаковое число разрядов. В неравномерных кодах число разрядов в словах может быть различным. В вычислительных машинах применяются преимущественно равномерные коды.

Равномерные избыточные коды делятся на делимые и неделимые. Делимые коды всегда содержат постоянное число информационных (т. е. представляющих передаваемую информацию) и избыточных разрядов, причем избыточные занимают одни и те же позиции в кодовом слове. В неделимых кодах разряды кодового слова невозможно разделить на информационные и избыточные.

Способность кода обнаруживать или исправлять ошибки определяется так называемым минимальным кодовым расстоянием. Кодовым расстоянием между двумя словами называется число разрядов, в которых символы слов не совпадают. Если длина слова n , то кодовое расстояние может принимать значения от 1 до n . Минимальным кодовым расстоянием данного кода называется минимальное расстояние между двумя любыми словами в этом коде. Если имеется хотя бы одна пара слов, отличающихся друг от друга только в одном разряде, то минимальное расстояние данного кода равно 1.

Простой (неизбыточный) код имеет минимальное расстояние $d_{min} = 1$. Для избыточных кодов $d_{min} > 1$. Если $d_{min} \geq 2$, то

любые два слова в данном коде отличаются не менее чем в двух разрядах, следовательно, любая одиночная ошибка приведет к появлению запрещенного слова и может быть обнаружена. Если $d_{min} = 3$, то любая одиночная ошибка создает запрещенное слово, отличающееся от правильного в одном разряде, а от любого другого разрешенного слова — в двух разрядах. Заменяя запрещенное слово ближайшим к нему (в смысле кодового расстояния) разрешенным словом, можно исправить одиночную ошибку.

В общем случае, чтобы избыточный код позволял обнаруживать ошибки кратностью r , должно выполняться условие

$$d_{min} \geq r + 1. \quad (12.2)$$

Действительно, одновременная ошибка в r разрядах слова создает новое слово, отстоящее от первого на расстоянии r . Чтобы оно не совпало с каким-либо другим разрешенным словом, минимальное расстояние между двумя разрешенными словами должно быть хотя бы на единицу больше, чем r .

Для исправления r -кратной ошибки необходимо, чтобы новое слово, полученное в результате такой ошибки, не только не совпадало с каким-либо разрешенным словом, но и оставалось ближе к правильному слову, чем к любому другому разрешенному слову. От правильного слова новое отстоит на расстоянии r . Следовательно, от любого другого разрешенного слова оно должно отстоять не менее чем на $r + 1$, а минимальное кодовое расстояние должно быть не менее суммы этих величин:

$$d_{min} \geq 2r + 1. \quad (12.3)$$

Код с проверкой четности. Код с проверкой четности образуется добавлением к группе информационных разрядов, представляющих простой (неизбыточный) код, одного избыточного (контрольного) разряда.

При формировании кода слова в контрольный разряд записывается 0 или 1 таким образом, чтобы сумма 1 в слове, включая избыточный разряд, была четной (при контроле по четности) или нечетной (при контроле по нечетности). В дальнейшем при всех передачах, включая запись в память и считывание, слово передается вместе со своим контрольным разрядом. Если при передаче информации приемное устройство обнаруживает, что в принятом слове значение контрольного разряда не соответствует четности суммы 1 слова, то это воспринимается как признак ошибки.

Минимальное расстояние кода $d_{min} = 2$, поэтому код с проверкой четности обнаруживает все одиночные ошибки, а кроме

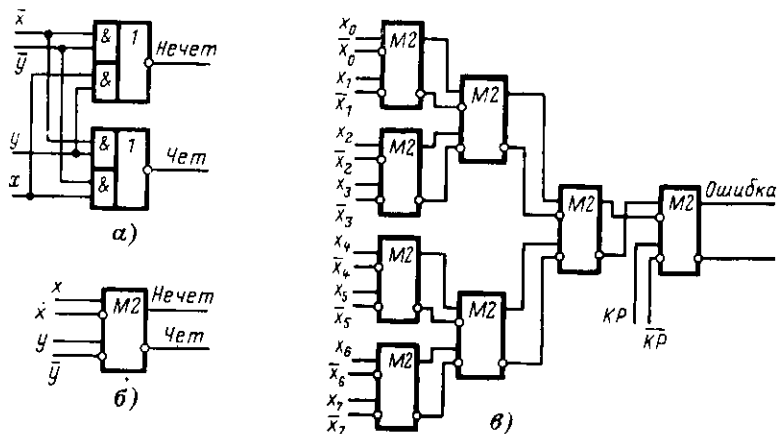


Рис. 12.1. Схемы определения четности

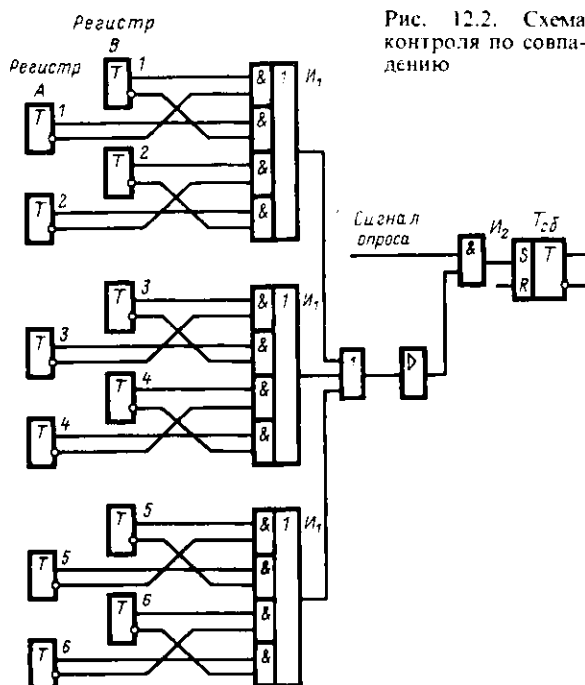
того, все случаи нечетного числа ошибок (3, 5 и т. д.). При одновременном возникновении двух или любого другого четного числа ошибок код с проверкой четности не обнаруживает ошибок.

При контроле по нечетности контролируется полное пропадание информации, поскольку кодовое слово, состоящее из 0, относится к запрещенным.

Код с проверкой четности имеет небольшую избыточность и не требует больших затрат оборудования на реализацию контроля. Этот код широко применяется в вычислительных машинах для контроля передач информации между регистрами и для контроля считываемой информации в оперативной памяти.

При построении схем определения четности суммы 1 слова используют логические элементы с парафазным выходом, подобные изображенному на рис. 12.1, а и б. Показанные схемы выполняют операцию сложения по модулю 2 (условное обозначение M2) для двоичных переменных x и y . На рис. 12.1, в показана схема определения признака четности байта.

Каждый информационный символ должен быть задан прямым и инверсным кодом. Структура схемы проверки четности является многоступенчатой, т. е. слово делится на несколько групп разрядов, в каждой из которых проверка четности производится прямым способом (первая ступень), далее производится проверка четности для групп второй ступени, образованных из групп первой ступени, четности которых в этом случае рассматриваются как обычные двоичные разряды, и т. д.



до окончательной проверки четности суммы 1 всего слова. В последней ступени четность байта сравнивается со значением контрольного разряда *KP*.

Легко установить связь кодирования при контроле по четности с выполнением сложения по модулю 2. Если количество 1 в слове должно быть четным, то в контрольный разряд записывается прямой код суммы по модулю 2 всех информационных разрядов слова. При контроле на нечетность в контрольный разряд заносится обратный код указанной суммы.

Контроль по совпадению. При передачах можно использовать и другой вид контроля – контроль по совпадению. После передачи информации из одного регистра в другой правильность передачи можно проверить поразрядным сравнением содержимого всех разрядов регистров.

При контроле по совпадению не требуется формирования каких-либо дополнительных контрольных разрядов, следовательно, этот метод основывается не на информационной, а на схемной избыточности.

Один из вариантов схемы контроля передач по совпадению показан на рис. 12.2. После передачи информации из регистра

А в регистр В (или из В в А) через время, несколько большее времени установления переходных процессов в триггерах регистров, на выходе схемы появляется сигнал 1 при несовпадении содержимого А и В и 0 в противном случае. При контроле передачи по совпадению обнаруживаются ошибки любой кратности. Затраты оборудования при контроле по совпадению меньше, чем при контроле по четности.

Контроль по совпадению является быстродействующим, так как используется одна ступень формирования, в то время как в схеме проверки четности многоразрядного числа, как правило, больше одной ступени.

Однако контроль по совпадению обладает существенным недостатком. Этот метод позволяет проверить правильность передачи числа в регистр и отсутствие сбоев при его хранении только до тех пор, пока не изменит своего состояния регистр, из которого передавалась информация. При контроле по четности проверяется не только правильность передачи, но и отсутствие сбоев при хранении числа в регистре (памяти) в течение сколь угодно большого времени.

Корректирующий код Хэмминга. В оперативной памяти применяют код Хэмминга, позволяющий исправлять ошибки.

Код Хэмминга строится таким образом, что к имеющимся информационным разрядам слова добавляется определенное число контрольных разрядов, которые формируются перед записью слова в ОП и вместе с информационными разрядами слова записываются в память.

При считывании слова контрольная аппаратура образует из прочитанных информационных и контрольных разрядов корректирующее число, которое равно 0 при отсутствии ошибки либо указывает место ошибки, например двоичный порядковый номер ошибочного разряда в слове. Ошибочный разряд автоматически корректируется изменением его состояния на противоположное.

Рассмотрим процесс кодирования для кода Хэмминга с коррекцией одиночной ошибки (минимальное кодовое расстояние $d_{\min} = 3$). Если кодовое слово не содержит ошибок, то корректирующее число должно быть равно 0. При наличии ошибки корректирующее число должно содержать номер ошибочного разряда. Если в младшем разряде корректирующего числа появится 1, то это означает ошибку в одном из тех разрядов слова, порядковые номера которых имеют 1 в младшем разряде (т. е. разрядов с нечетными номерами). Введем первый контрольный разряд, которому присвоим нечетный порядковый номер и который установим при кодировании таким образом, чтобы сумма 1 всех разрядов с нечетными порядковыми номе-

рами была равна 0. Эта операция может быть записана в виде

$$E_1 = x_1 \oplus x_3 \oplus x_5 \oplus \dots = 0,$$

где x_1, x_3 и т. д. — двоичные символы, размещенные в разрядах с порядковыми номерами 1, 3 и т. д.

Появление 1 во втором разряде (справа) корректирующего числа означает ошибку в тех разрядах слова, порядковые номера которых (2, 3, 6, 7, 10, 11, 14, 15 и т. д.) имеют 1 во втором справа разряде. Поэтому вторая операция кодирования, позволяющая найти второй контрольный разряд, которому должен быть присвоен какой-либо порядковый номер из группы 2, 3, 6, 7, 10, 11 и т. д., имеет вид

$$E_2 = x_2 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_{10} \oplus x_{11} \oplus \dots = 0.$$

Рассуждая аналогичным образом, можно определить все другие контрольные разряды путем выполнения операций

$$E_3 = x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15} \oplus \dots = 0,$$

$$E_4 = x_8 \oplus x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15} \oplus x_{24} \oplus \dots = 0 \quad \text{и т. д.}$$

После приема кодового слова (совместно со сформированными контрольными разрядами) выполняются те же операции подсчета, которые были описаны выше, а образующееся число

$$E_k E_{k-1} \dots E_3 E_2 E_1$$

считается корректирующим.

При отсутствии ошибок $E_k E_{k-1} \dots E_2 E_1 = 0$ при наличии ошибки не равными нулю будут те суммы E_i , в образовании которых участвовал ошибочный разряд; корректирующее число при этом будет равно порядковому номеру ошибочного разряда.

Выбор места для контрольных разрядов производится таким образом, чтобы контрольные разряды участвовали только в одной операции подсчета четности. Это упрощает процесс кодирования. Рассмотрение выражений для E_1, E_2, E_3 и т. д. показывает, что такими позициями являются разряды с номерами, являющимися целыми степенями двойки: 1, 2, 4, 8, 16 и т. д.

Требуемое число контрольных разрядов (или, что то же самое, разрядность корректирующего числа) определяется из

следующих соображений. Пусть кодовое слово длиной n разрядов имеет m информационных и $k = n - m$ контрольных разрядов. Корректирующее число длиной k разрядов описывает 2^k состояний, соответствующих отсутствию ошибки и появлению ошибки в i -м разряде. Таким образом, должно соблюдаться соотношение

$$2^k \geq n + 1 \quad (12.4)$$

или

$$2^k - k - 1 \geq m. \quad (12.5)$$

Из этого неравенства следует, например, что пять контрольных разрядов позволяют передавать в коде Хэмминга до 26 информационных разрядов и т. д.

Если в ОП одновременно записываются или считываются восемь информационных байт (64 разряда), то при использовании кода Хэмминга потребуется семь дополнительных контрольных разрядов.

Контроль по коду Хэмминга реализуется с помощью набора схем подсчета четности (см. рис. 12.1), которые при кодировании определяют контрольные разряды, а при декодировании формируют корректирующее число.

Модифицированный код Хэмминга. К контрольным разрядам Хэмминга добавляется еще один (в последнем примере восьмой) разряд КР контроля четности всех одновременно считываемых (записываемых) информационных и контрольных разрядов. При считывании формируются корректирующее число $E_k E_{k-1} \dots E_1$, и разряд общей четности КР' для всех считанных разрядов, включая КР. Модифицированный код Хэмминга позволяет устранять одиночные и обнаруживать двойные ошибки, как это следует из табл. 12.1.

Коррекция двойных ошибок в ОП. При использовании в ОП модифицированного кода Хэмминга может производиться коррекция двойных ошибок.

Таблица 12.1

Корректирующее число Хэмминга $E_k E_{k-1} \dots E_1$	КР'	Наличие ошибок
Нули	0	Нет ошибок
$\neq 0$	1	Одиночная ошибка
$\neq 0$	0	Двойная ошибка
Нули	1	Ошибка в контрольном разряде общей четности

Пусть X — слово, записанное в ОП, а X' — считанное из ОП слово, в котором обнаружены две ошибки. Тогда по сигналу схемы контроля инициируется следующая процедура.

В неисправную ячейку ОП записывается обратный код считанного слова $\bar{X}'$ и затем производится его считывание. Над получаемым при этом кодом $(\bar{X}')'$ и кодом $\bar{X}'$ производится операция

$$Z = \bar{X}' \oplus (\bar{X}')'.$$

Код Z содержит 1 в разрядах, в которых имеются ошибки.

Схемы управления ОП по коду Z корректируют одну ошибку. После этого схема коррекции одной ошибки исправляет вторую ошибку.

12.3. КОНТРОЛЬ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ

Арифметические операции, как правило, можно представить в виде последовательностей следующих элементарных операций: передача слова и операции преобразования содержимого триггерных регистров — сдвиг, взятие обратного кода и сложение.

Операция сдвига информации в регистре представляет собой по существу передачу информации из i -х разрядов регистра в $(i + m)$ -е или $(i - m)$ -е разряды в зависимости от направления сдвига (m — число разрядов, на которое производится сдвиг). Поэтому для контроля операции сдвига можно использовать те же методы, что и для контроля передачи информации, например контроль четности суммы 1.

Регистр, в котором производится сдвиг, должен иметь дополнительный контрольный разряд, устанавливаемый перед сдвигом в такое состояние, чтобы сумма 1 регистра вместе с контрольным разрядом была, например, четной. Кроме схемы определения общей четности содержимого регистра необходимы схемы, устанавливающие четность разности между числом 1, выдвигаемых из регистра, и числом 1, вдвигаемых в регистр. Если в освобождающиеся при сдвиге разряды вдвигаются 0, то достаточно иметь схему определения четности суммы 1 выдвигаемых разрядов. Одновременно со сдвигом выполняется контрольная операция, состоящая в том, что состояние контрольного разряда меняется на противоположное при нечетности суммы 1 выдвигаемых разрядов. Тогда при правильном выполнении сдвига общая четность суммы 1 в регистре после сдвига не меняется.

Операция взятия обратного кода может быть также проконтролирована путем использования кодов с проверкой четности.

Если число информационных разрядов в слове четно, то число 1 в слове четно при четном числе 0 и число 1 нечетно при нечетном числе 0. В этом случае после образования обратного кода четность числа 1 в слове сохранится, и правильность выполнения операции можно определить, проверив сохранение четности или нечетности суммы 1 в слове, включая контрольный разряд.

Если число информационных разрядов в слове нечетно, то четному числу 1 в слове соответствует нечетное число 0, а нечетному числу 1 — четное число 0. В этом случае после образования обратного кода четность числа 1 изменится на обратную, и для проверки правильности выполнения операции необходимо взять обратный код от содержимого контрольного разряда и проверить сохранение четности или нечетности суммы 1 в слове, включая контрольный разряд.

Рассмотрим метод контроля операции сложения, использующий проверки четности.

При сложении двух чисел a и b разряды суммы S образуются в соответствии с выражениями

$$\left. \begin{aligned} S_1 &= a_1 \oplus b_1 \oplus P_1; \\ S_2 &= a_2 \oplus b_2 \oplus P_2; \\ &\dots \\ S_n &= a_n \oplus b_n \oplus P_n. \end{aligned} \right\} \quad (12.6)$$

где S_i , a_i , b_i , P_i ($i = 1, 2, \dots, n$) — соответственно значения разрядов суммы, слагаемых и переноса, поступающего в i -й разряд. Знак $\oplus$ означает сложение по модулю 2; n — число разрядов слагаемых и суммы.

Сложив все n приведенных выше равенств по модулю 2, получим

$$S_1 \oplus S_2 \oplus \dots \oplus S_n = (a_1 \oplus a_2 \oplus \dots \oplus a_n) \oplus (b_1 \oplus b_2 \oplus \dots \oplus b_n) \oplus (P_1 \oplus P_2 \oplus \dots \oplus P_n). \quad (12.7)$$

Поскольку сумма по модулю 2 всех разрядов слова выражает четность суммы 1 слова, последнее уравнение можно переписать в виде

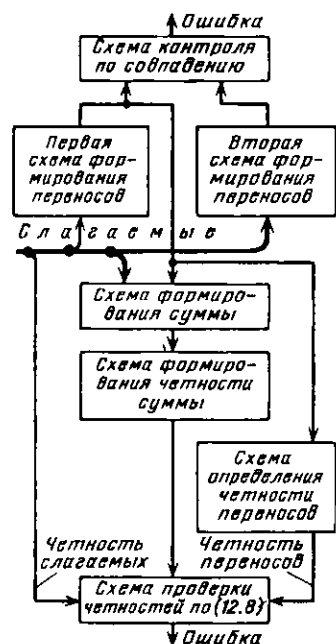
$$\text{Четность } S = \text{Четность } a \oplus \text{Четность } b \oplus \text{Четность } P. \quad (12.8)$$

Таким образом, при правильном образовании суммы четность суммы ее единиц должна совпадать с четностью, определяемой выражением (12.8). Однако, как указывалось ранее, при контроле по четности не обнаруживается четное число ошибок.

Сбой в схеме образования цифры разряда суммы дает одиночную ошибку, и она будет обнаружена. Если сбой произойдет в схеме формирования переноса, то он может привести к распространению ошибки по многим разрядам суммы. В связи с этим для полноты контроля необходимо проверять правильность образования переноса P_i . Такой контроль может быть организован с помощью схемы, которая проверяет, что в каждом разряде существует либо перенос в прямом коде, либо инверсия переноса и не существует одновременно и то, и другое.

Другой способ заключается в дублировании схем формирования переносов и сравнении переносов основной и дублирующей схем.

Упрощенная схема контроля сумматора приведена на рис. 12.3. Оба слагаемых поступают на схемы формирования переносов. Сформированные переносы проверяются на совпадение с помощью схемы, аналогичной изображенной на рис. 12.2. Для переносов и отдельно для суммы формируются контрольные разряды четности. Затем схема проверки четности проверяет выполнение условия (12.8).



Контроль выполнения арифметических операций (сложения, вычитания, умножения) можно осуществлять с помощью контрольных кодов, представляющих собой остатки от деления чисел на некоторый модуль R (контроль по модулю R).

Если в качестве контрольного кода используется остаток по модулю R , то в качестве контрольной операции над остатками может быть выбрана та же арифметическая операция, которая производится над числами. Это вытекает из того, что для сложения, вычитания и умножения действительно соотношение

$$R(A * B) = R[R(A) * R(B)], \quad (12.9)$$

Рис. 12.3. Контроль сложения с использованием проверок по четности

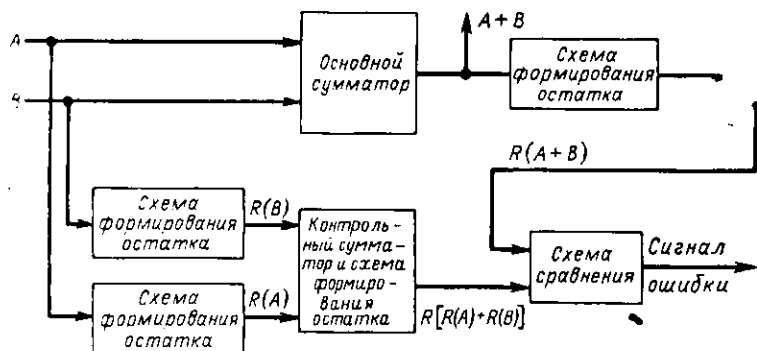


Рис. 12.4. Структурная схема сумматора с контролем по модулю R

где $R(X)$ обозначает остаток числа X по модулю R ; $*$ — знак арифметической операции сложения, вычитания или умножения.

Контроль арифметических операций по модулю организуется следующим образом. Каждому числу, участвующему в арифметической операции, ставится в соответствие контрольный код — остаток по модулю R . Одновременно с выполнением основной операции над числами та же операция производится над их контрольными кодами, и контрольный код результата основной операции сравнивается с результатом операции над контрольными кодами исходных чисел. При несовпадении фиксируется ошибка.

Чем меньше R , тем меньше разрядность контрольного кода и проще дополнительная аппаратура. Для двоичных чисел контроль по модулю возможен при $R \geq 3$, поэтому в ЭВМ часто используют контроль по модулю 3.

Покажем, что с помощью контроля по модулю 3 обнаруживаются любые одиночные ошибки. Одиночная ошибка в каком-либо разряде двоичного числа соответствует изменению числа на значение $\pm 2^i$. Для обнаружения ошибки необходимо, чтобы контрольные коды чисел a и $a \pm 2^i$ не совпадали, т. е.

$$R(a) \neq R(a \pm 2^i) = R(a) \pm R(2^i)$$

или

$$R(2^i) \neq 0.$$

Но 2^i не делится на 3 без остатка, следовательно, требуемое условие выполняется. Кроме одиночных ошибок при кон-

троле по модулю 3 обнаруживается часть двойных ошибок, а именно те, при которых правильные и ошибочные результаты имеют несовпадающие остатки от деления на 3.

На рис. 12.4 изображена структурная блок-схема сумматора с контролем по модулю R . Отметим, что контрольный блок значительно проще основного, при $R = 3$ содержит только два двоичных разряда. При реализации контроля важное значение имеет построение схем формирования остатков при минимальных затратах оборудования. Очевидно, что нахождение остатка путем прямого деления двоичного числа на 3 — путь неприемлемый. Однако кодирование по модулю 3 обладает свойствами, позволяющими строить достаточно простые комбинационные схемы формирования остатка по модулю 3.

Может быть применен контроль по модулю с большим основанием, чем 3, например по модулю 7. При этом увеличивается число кратных ошибок, которые могут обнаруживаться системой контроля, однако возрастает сложность кодирующей и декодирующей аппаратуры.

12.4. ВЗАИМОДЕЙСТВИЕ СИСТЕМ АВТОМАТИЧЕСКОГО КОНТРОЛЯ, ВОССТАНОВЛЕНИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА И ДИАГНОСТИРОВАНИЯ

В ЭВМ, снабженной системой автоматического контроля, возникновение ошибок в каком-либо устройстве порождает сигнал ошибки, с появлением которого приостанавливается выполнение программы целиком или только рабочей процедуры в неисправном устройстве. При этом в соответствующем разряде регистра ошибок, высвечиваемом на сигнальном табло на пульте оператора, указывает укрупненно место, где обнаружена ошибка (устройство, узел, регистр, группа разрядов регистра и т. д.). Сигнал ошибки инициирует работу системы восстановления.

Система автоматического восстановления во взаимодействии с системой автоматического контроля обычно выполняет следующие функции:

1) распознавание характера обнаруженной ошибки, т. е. выяснение, вызвана ошибка случайным сбоем, перемежающимся или устойчивым отказом;

2) организация «рестарта», т. е. продолжения выполнения программы путем устранения возникшей ошибки в информации повторением ошибочно выполненной микрооперации, команды или сегмента программы (при обнаружении, что ошибка вызвана сбоем);

3) запись в память информации о сбое;

4) инициирование (при обнаружении отказа) работы системы автоматического диагностирования (САД).

На САД в данном случае возлагается:

1) локализация места отказа с заданной степенью подробности, например до уровня сменной платы и, если возможно, реконфигурация, т. е. отключение неисправного узла или устройства с передачей его функции другому соответствующему узлу или устройству;

2) запись в память информации об отказе для последующей обработки.

Общая логика взаимодействия систем автоматического контроля, восстановления и диагностирования показана на рис. 12.5. Первой выполняется процедура обработки сигнала прерывания от системы контроля, начинающаяся записью состояний регистров процессора и старого слова состояния программы (ССП) в соответствующие ячейки памяти и выборкой нового ССП системы восстановления. Далее производится запись в специальные регистры или ячейки памяти ситуации в ЭВМ в момент обнаружения ошибки (записывается состояние запоминающих элементов и шин передачи данных неисправного устройства)¹.

Для распознавания характера ошибки (сбой или отказ) необходимо повторить, может быть даже несколько раз, операцию, в которой обнаружилась ошибка. Однако желательно это сделать с минимальными потерями времени. С этой точки зрения лучше всего, если процессор и система контроля выполнены таким образом, что проверяется правильность каждой микрооперации и вычислительный процесс останавливается на ошибочно выполненной микрооперации, которая может быть повторена. Микрооперация (команда) может быть повторена, если не исказилась используемая в операции информация (операнды, адреса и т. д.), т. е. если не пройден «порог повторения». Это проверяет входящая в систему программа обработки сбоев, которая по записанной ситуации, соответствующей появлению ошибки, определяет, пройден или нет порог повторения микрооперации или команды². Если соответствующий порог пройден, то вместо микрооперации повторяется команда,

¹ В случае отказа периферийного устройства используется информация о его состоянии, доставляемая командой «Уточнить состояние».

² Операции ввода-вывода допускают повторения на уровне команды.

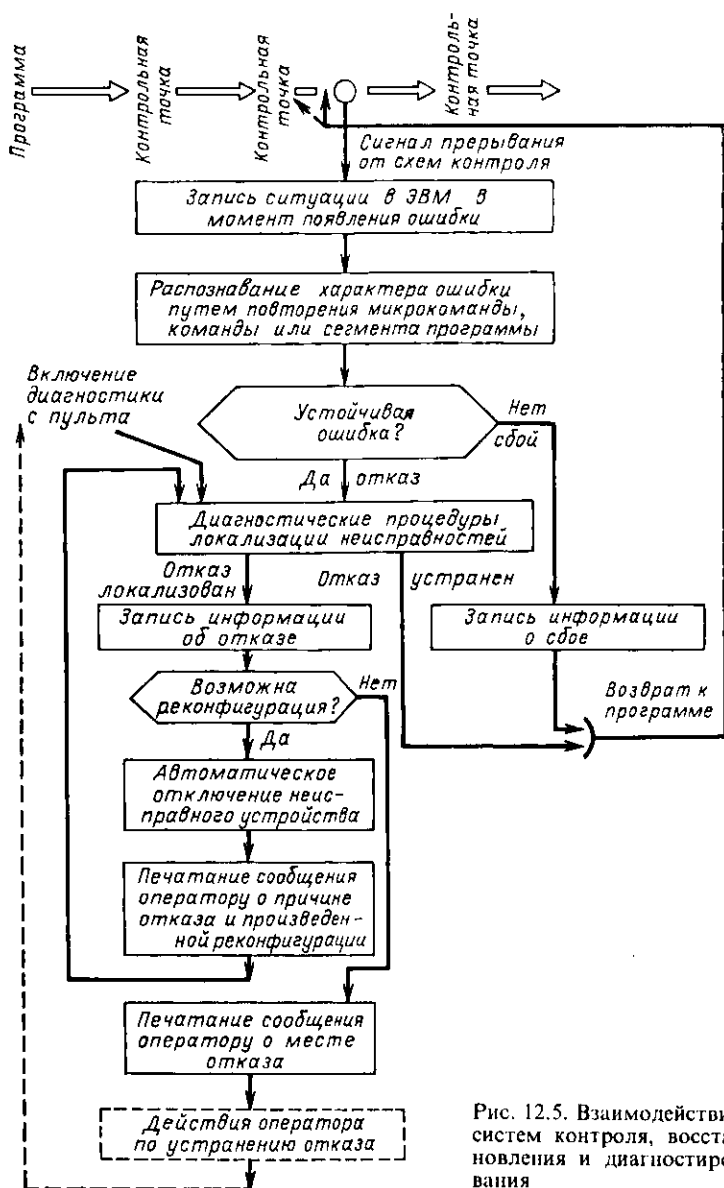


Рис. 12.5. Взаимодействие систем контроля, восстановления и диагностирования

а вместо команды — сегмент программы. Последнее возможно, если программист предусмотрел в программе «контрольные точки», сохраняющие промежуточные данные, позволяющие повторить программу с данного места.

Если при повторении микрокоманды (команды, сегмента программы) ошибка не повторяется, событие распознается как сбой и происходит возврат к точке прерывания программы, но перед этим в память записывается дополнительная информация о состоянии в момент сбоя вычислительной системы (какие в это время выполнялись операции в периферийных и других устройствах, название программы, адрес команды, операнды, время). Информация о ситуациях при сбоях и об отказах накапливается во внешнем ЗУ и в последующем обрабатывается специальной программой, вырабатывающей определенные рекомендации обслуживающему персоналу, выполняющему профилактические работы.

Если при определенном числе повторений (например, 8) ошибка сохраняется, событие распознается как отказ и автоматически приводится в действие САД для определения места неисправности. Диагностические процедуры также могут включаться вручную с пульта, например при пуске машины.

После выявления места отказа проверяется возможность реконфигурации путем автоматического отключения неисправного устройства и передачи его функций другому устройству. Если это возможно, то производится реконфигурация системы. Затем после сообщения оператору об отказе и произведенной реконфигурации происходит возврат к контрольной точке программы. Если реконфигурация невозможна, САД индицирует на пульте код неисправностей. Оператор, пользуясь справочником неисправностей, находит неисправный блок, ремонтирует или заменяет его. Затем с пульта включает диагностические процедуры. Если отказ устранен, производится возврат к контрольной точке программы.

12.5. ПРИНЦИПЫ ПОСТРОЕНИЯ СИСТЕМ АВТОМАТИЧЕСКОГО ДИАГНОСТИРОВАНИЯ ЭВМ

В техническом обслуживании ЭВМ наиболее сложной проблемой является поиск места (локализация) неисправности, вызвавшей отказ в работе машины. Поиск места неисправности вручную в таком сложном объекте, как современные ЭВМ, требует очень высокой квалификации обслуживающего персонала и больших затрат времени.

Для облегчения и ускорения поиска причины отказа ЭВМ снабжаются системами автоматического диагностирования неисправностей.

Диагностирование некоторого объекта может производиться функциональным или тестовым способом. Функциональное диагностирование предполагает контроль при помощи тех или иных средств правильности функционирования объекта диагностирования при реализации его рабочего процесса. Тестовое диагностирование производится путем подачи на диагностируемый объект специальных тестовых воздействий и сравнения реакций объекта на эти воздействия с эталонными ответами.

Применяемые в ЭВМ средства автоматического контроля правильности функционирования помимо главной своей задачи — обнаружения ошибки в работе машины — позволяют диагностировать (локализовать) место неисправности, но с точностью лишь до устройства или крупного блока (например, модуля ОП, АЛУ и т. п.), т. е. с разрешающей способностью, которая для большинства ЭВМ оказывается недостаточной.

В ЭВМ используется главным образом метод тестового диагностирования, позволяющий локализовать место неисправности с достаточной разрешающей способностью.

Диагностические процедуры могут осуществляться путем пропуска на ЭВМ специальных диагностических программ. Однако несравненно более гибкое диагностирование, с большей детализацией места неисправности (большей разрешающей способностью) обеспечивают диагностические процедуры, выполняемые под управлением специальных диагностических микротестов и микропрограмм («микропрограммное диагностирование», или, короче, «микродиагностика»).

В связи со сложностью аппаратуры диагностирование ЭВМ производится в форме многоэтапного процесса, причем на разных этапах используются различные средства САД — микропрограммные, программные, а также некоторые аппаратурные средства системы диагностирования.

Система автоматического диагностирования представляет собой комплекс аппаратурных, микропрограммных и программных средств и справочной документации (справочников неисправностей, инструкций, схем ЭВМ, тестов).

В зависимости от размещения аппаратурных средств системы диагностирования различают встроенные САД, когда диагностирующие средства размещаются внутри ЭВМ, и внешние САД, когда эти средства находятся вне машины. На практике часто САД строятся комбинированными: одна часть их средств встраивается в машину, а другая оформляется в виде

специального дополнительного оборудования, подсоединяемого к ЭВМ при диагностировании неисправностей.

Обычно объем диагностических микропрограмм (тестов «микродиагностики») и программ столь велик, что их не удается хранить внутри ЭВМ (в УП, ОП или специальных встроенных памяти), но это в ряде случаев и не так важно, так как диагностические тесты не так часто выполняются. Поэтому обычно они хранятся во внешних ЗУ, в том числе в специальных внешних ЗУ САД.

Для САД современных средних и больших ЭВМ общего назначения характерным является наличие в их составе построенных на основе микропроцессоров или микро-ЭВМ специализированных диагностических процессоров (называемых *сервисными адаптерами* или *сервисными процессорами*), управляющих загрузкой в ЭВМ из внешних ЗУ («сервисных ЗУ») диагностической информации, инициированием диагностических процедур, опросом состояния ЭВМ после подачи тестовых воздействий, сравнением полученных ответов с эталонными, индикацией и регистрацией результатов диагностирования.

При построении САД ЭВМ широкого назначения (например, в ЕС ЭВМ) используется диагностирование по методу «раскрутки». Метод «раскрутки» предполагает поэтапное последовательное расширение работоспособной части ЭВМ путем включения в эту часть оборудования, проверенного на предыдущем этапе.

На каждом i -м этапе диагностического процесса подмашина M_i , представляющая собой часть оборудования проверяемой ЭВМ, участвует в диагностировании аппаратуры ΔM_i , которая, если в ней нет отказов, присоединяется к подмашине M_i , образуя новую подмашину $M_{i+1} = M_i \cup \Delta M_i$, выполняющую диагностическую процедуру на следующем этапе.

Подмашина M_0 , с которой начинается «раскрутка», называется «*диагностическим ядром*» ЭВМ. Диагностическое ядро должно иметь повышенную надежность и допускать проверку работоспособности вручную или в полуавтоматическом режиме.

Системы автоматического диагностирования в машинах ЕС ЭВМ II очереди. В рассматриваемых САД реализовано микропрограммное диагностирование (микродиагностика).

Микропрограммное диагностирование в отличие от диагностирования программными средствами позволяет контролировать изменение состояния аппаратуры ЭВМ на каждом машинном такте, благодаря чему достигается высокая разрешающая способность САД — 1–2 сменных элемента (ТЭЭ). Однако при этом требуется дополнительная аппаратура (около 3–5%

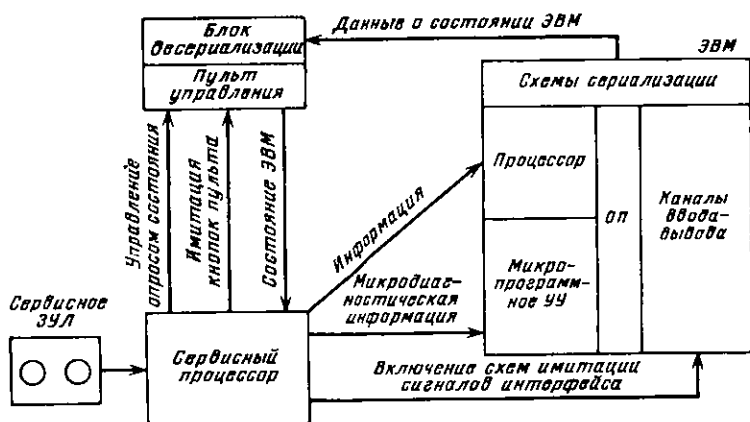


Рис. 12.6. Система автоматического диагностирования ЭВМ ЕС-1045

объема оборудования процессора и каналов) и необходима разработка большого объема (около 1 Мбайт) микродиагностической информации (микротестов).

Средства САД и, в частности, микродиагностирование в некоторых моделях ЭВМ используются системой автоматизации профилактических испытаний, выполняемых при программно-управляемом изменении уровней питающих напряжений на $\pm 5\%$ номинальных значений.

Системы диагностирования ЭВМ ЕС используют дополнительное внешнее оборудование – сервисный адаптер (сервисный процессор) с собственной небольшой оперативной памятью и сервисное кассетное ЗУЛ (пультный накопитель). На рис. 12.6 показаны связи этих сервисных устройств с оборудованием ЭВМ в САД ЭВМ ЕС-1045. Микродиагностическая информация, а в ряде моделей ЕС ЭВМ II очереди и рабочие микропрограммы машины хранятся на кассетах сервисного ЗУЛ.

Управляющая память (УП) микропрограммного управляющего устройства процессора машины вся или частично является загружаемой (например, в ЭВМ ЕС-1045 загружаемая секция имеет емкость 1 Кбайт при общей емкости УП 8 Кбайт).

Из сервисного ЗУЛ передаются в регистр микрокоманд УУ процессора микротесты, в загружаемую УП (или ее загружаемую секцию) диагностические микропрограммы. Кроме того, из сервисного ЗУЛ производится также загрузка диагностических программ в ОП ЭВМ.

Сервисный процессор имеет собственную ориентированную на диагностические операции систему команд. Эти команды поступают в сервисный процессор с сервисного ЗУЛ и дешифрируются дешифратором диагностических операций сервисного процессора.

Данные о текущем состоянии машины, т. е. данные о состоянии ее регистров, счетчиков, управляющих триггеров, уровней сигналов на шинах (общий объем этих данных в ЭВМ ЕС составляет 3–4 тыс. бит), должны передаваться на пульт управления машины, а в режиме диагностирования – выборочно и в сервисный процессор для сравнения реакций аппаратуры на микротесты и диагностические микропрограммы с эталонными, содержащимися в микродиагностической информации.

Передача из машины на пульт управления и в сервисный процессор указанного объема данных параллельным кодом практически невозможна, так как требует нескольких тысяч проводов. Поэтому эти данные подвергаются *сериализации*, другими словами, преобразованию объединенного параллельного кода состояния ЭВМ в последовательный. Этот последовательный код передается на пульт управления машины, там преобразуется в поток байт (*десериализация*) и отображается на индикаторах пульта, запоминается в памяти пульта, а нужные фрагменты данных о состоянии ЭВМ, адресуемые сервисным процессором, поступают в последний для сравнения с эталонными кодами. Сериализация данных о состоянии ЭВМ производится многоступенчатой (в ЭВМ ЕС-1045 четырехступенчатой) схемой, построенной на мультиплексорах (см. гл. 3) [102].

В ЕС ЭВМ диагностирование машин производится по методу «раскрутки». Диагностическое ядро образуют сервисный процессор, сервисное ЗУЛ и часть аппаратуры пульта управления.

Диагностирование производится в несколько этапов. Например, в САД ЭВМ ЕС-1045 можно выделить следующие укрупненные этапы: 1) статическое микродиагностирование, 2) динамическое микродиагностирование, 3) функциональное (программное) диагностирование.

Статическое микродиагностирование. Управление процессором машины берет на себя сервисный процессор. Производится загрузка из сервисного ЗУЛ регистра микрокоманд микротестом и иницируется выполнением микрооперации с заданным тестовым набором.

Одновременно из сервисного ЗУЛ поступают эталонный код и адрес (номер) байта в сериализованном потоке данных о состоянии ЭВМ, который следует сравнить с эталоном.

В данном случае микродиагностирование называется «статическим», так как проверяемая аппаратура работает не с рабочей тактовой частотой, а в темпе, определяемом загрузкой оборудования микротестами. На данном этапе проверяются аппаратура процессора, каналов, адаптера канал — канал, управление памятью, загружаемая УП, частично другие памяти.

Динамическое микродиагностирование производится при работе аппаратуры ЭВМ с рабочей тактовой частотой, что позволяет выявить неисправности, связанные с искажениями в реализации временных диаграмм. При этом используется проверенная на предыдущем этапе загружаемая УП, в которую с сервисного ЗУ загружаются диагностические микропрограммы, принимающие на себя управление ЭВМ. Сервисный процессор сравнивает результаты выполнения микропрограмм с эталонами. На этом этапе в динамическом режиме проверяется аппаратура ЭВМ, в том числе все ее памяти.

Функциональное (программное) диагностирование. Средствами функционального диагностирования служат проверенная на предыдущих этапах аппаратура машины и диагностические программы: 1) программа «базовый тест»; 2) программы тест-секций, выполняемые под управлением мониторной программы (*диагностического монитора*); 3) программы автономных диагностических тестов отдельных устройств. Базовый тест, загружаемый с сервисного ЗУЛ, проверяет выполнение группы команд процессора («ядро система команд»), в том числе команд ввода-вывода, цепи связи с некоторыми периферийными устройствами.

Далее при помощи программ тест-секций производится диагностирование всех операций процессора, системы прерывания, защиты памяти, таймера, средств прямого управления. ОП, памяти ключей защиты, каналов ввода-вывода, периферийных устройств и их УПУ.

Если результат прохождения какого-либо теста оказывается неудачным, САД высвечивает на индикаторном табло пульты управления номер этого теста. Оператор, пользуясь справочником неисправностей, по номеру теста находит место неисправности. В такой форме система сообщает оператору о месте неисправности на всех этапах — от начала диагностического процесса до этапа проверки мультиплексного канала включительно. В последующих за этим этапах диагностики система выдает на печатающее устройство сообщение оператору о месте неисправности и указание, какая сменная плата (ТЭЗ) подлежит замене.

ПРИНЦИПЫ ОРГАНИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ: МУЛЬТИПРОГРАММНЫЕ ВС, СИСТЕМЫ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ И РЕАЛЬНОГО ВРЕМЕНИ

13.1. КЛАССИФИКАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

В предыдущих главах были рассмотрены принципы построения и особенности функционирования основных элементов, узлов и устройств ЭВМ. В последующих главах излагаются принципы организации более сложных объектов, таких, как мультипрограммные ЭВМ, вычислительные системы, комплексы и сети.

Стремление удовлетворить требования разнообразных областей и форм применения электронной вычислительной техники, повысить производительность и расширить логические возможности ЭВМ, повысить надежность их функционирования, облегчить контакты человека с ЭВМ при подготовке программ и в процессе решения задач, повысить обслуживаемость машин привело в ряде случаев к созданию таких объектов вычислительной техники, которые из-за сложности входящего в них оборудования, тесной логической взаимосвязи аппаратных и программных средств при реализации сложных процессов функционирования, множества возможных конфигураций, территориальной рассредоточенности оборудования не укладываются в наше представление о понятии *машина*. В таких случаях вместо термина *вычислительная машина* пользуются термином *вычислительная система*.

Одним из важнейших путей повышения производительности вычислительных машин и систем, их эффективности и надежности является использование различных форм реализации принципа параллелизма в функционировании вычислительного оборудования. Поэтому в основу классификации ВС следует положить в первую очередь реализуемую системой форму параллелизма.

По режиму работы ВС делятся на *однопрограммные* и *мультипрограммные* (рис. 13.1, а).

В однопрограммной системе в памяти машины присутствует только одна рабочая программа (или часть ее), которая, начав выполняться, завершается до конца. При этом, даже если допускается совмещение во времени операций ввода-вывода

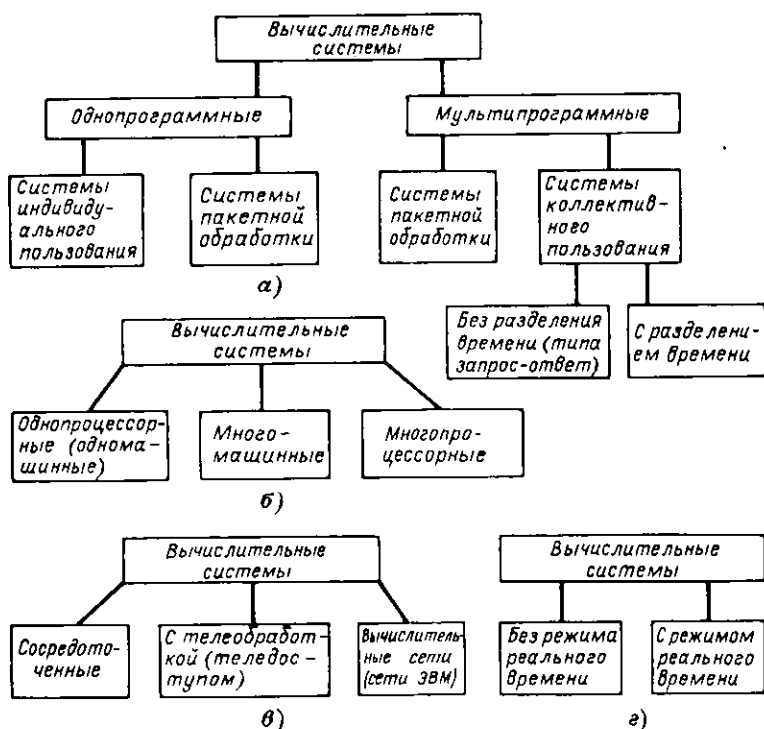


Рис. 13.1. Классификация вычислительных систем

с обработкой данных, возможен простой оборудования (например, машина не может продолжать работу, пока не будут введены новые данные).

Стремление повысить эффективность использования вычислительного оборудования привело к разработке мультипрограммных машин и систем, которые могут одновременно выполнять несколько программ или несколько частей одной и той же программы.

Когда говорится об одновременности выполнения программ, то подразумевается, что процессор после выполнения части одной программы может перейти к выполнению другой программы, не закончив ее, перейти к третьей и т. д., сохраняя возможность вернуться позднее к неоконченным программам и продолжить их выполнение. При этом моменты и очередность переключений программ должны быть выбраны так,

чтобы повысить общую эффективность вычислительной системы, хотя время, в течение которого решается каждая отдельная задача, по сравнению с однопрограммным режимом может даже увеличиться.

Мультипрограммный режим реализуется и по отношению к частям одной программы, что приводит к сокращению времени ее выполнения по сравнению со случаем, когда режим мультипрограммирования не используется.

Классификация систем по режиму обслуживания (рис. 13.1, а). Процесс решения задачи может рассматриваться как обслуживание пользователя ВС. Рассмотрим основные режимы обслуживания.

Режим индивидуального пользования. Машина предоставляется полностью в распоряжение пользователя, по крайней мере на время решения его задачи. Пользователь имеет непосредственный доступ к машине и может вводить информацию в оперативную память машины (или выводить из нее), используя пульт управления или устройства ввода-вывода.

Режим индивидуального пользования удобен пользователю, но плохо используется вычислительное оборудование из-за простоев, когда пользователь, получив некоторый промежуточный результат (например, при отладке программы), обдумывает, что он будет делать дальше. Этот режим применялся в ЭВМ первого поколения, а в настоящее время возродился как форма использования персональных микро-ЭВМ (*персональных компьютеров*).

Режим пакетной обработки. В этом режиме пользователи не имеют непосредственного доступа к ВС. Подготовленные ими программы передаются персоналу, обслуживающему систему, и затем накапливаются во внешней памяти (на магнитных дисках, лентах и т. п.). Система последовательно либо по заранее составленному расписанию выполняет накопленный пакет программ. Пакетная обработка может производиться в однопрограммном или мультипрограммном режимах.

Мультипрограммная пакетная обработка обеспечивает высокую степень загрузки вычислительного оборудования, но при этом из-за отсутствия непосредственной связи между системой и пользователем производительность и эффективность труда самих пользователей снижается по сравнению с индивидуальным обслуживанием. Это противоречие преодолевается в настоящее время путем создания систем коллективного пользования, содержащих высокопроизводительные ЭВМ, или, наоборот, путем применения персональных ЭВМ умеренной производительности в режиме индивидуального пользования.

Режим коллективного пользования или мультидоступа — форма обслуживания, при которой возможен одновременный доступ нескольких независимых пользователей к вычислительным ресурсам мощной ВС. Каждому пользователю предоставляется терминал, с помощью которого он устанавливает связь с системой коллективного пользования (ВСКП).

В наиболее простых ВСКП обработка всех запросов занимает примерно одно и то же время (системы типа «запрос — ответ»).

Обеспечение более тесного взаимодействия пользователей с вычислительными средствами в системе коллективного пользования, в которой запросы сильно разнятся по времени их обработки, требует в первую очередь сокращения времени ожидания пользователем результата выполнения коротких программ (коротких запросов), для чего применяют различные методы квантования времени, уделяемого процессором выполнению отдельных программ. Системы коллективного пользования с *квантованным обслуживанием* называются *системами с разделением времени*.

По количеству процессоров (машин) в ВС, определяющему возможность параллельной обработки программ, ВС делятся на *однопроцессорные (одномашинные)*, *многомашинные* и *многопроцессорные* (рис. 13.1, б).

По особенностям территориального размещения и организации взаимодействия частей системы различают следующие типы ВС (рис. 13.1, в).

Сосредоточенные ВС. В них весь комплекс оборудования, включая терминалы пользователей, сосредоточен практически в одном месте и связь между отдельными машинами и устройствами комплекса обеспечивается, причем без существенных запаздываний, стандартными для системы внутренними интерфейсами (без использования передачи данных по каналам связи).

Вычислительные системы с телеобработкой (или теледоступом). В этих ВС отдельные источники и приемники информации, включая терминалы пользователей, расположены на таком значительном расстоянии от вычислительных средств, что связь их с центральными средствами ВС осуществляется путем передачи данных по каналам связи.

Вычислительные сети (сети ЭВМ). Они представляют собой территориально рассредоточенную многомашинную систему, состоящую из взаимодействующих ЭВМ, связанных между собой каналами передачи данных.

По особенностям функционирования ВС во времени различают ВС, работающие не в масштабе реального времени, и ВС

реального времени (рис. 13.1, г). Последние должны работать в темпе с процессом, информация о котором автоматически поступает в ВС и обрабатывается. Результаты обработки информации должны получаться столь быстро, чтобы можно было ими воспользоваться для воздействия на сам процесс.

Вычислительные системы различаются по назначению следующим образом:

- информационно-справочные;

- сбора и обработки данных в АСУ, в частности информационно-планирующие;

- управления технологическими процессами в реальном времени;

- автоматизации обработки данных при сложных экспериментах;

- автоматизации проектирования и др.

Многие применения ВС, например автоматизация проектирования, требуют, чтобы поиск решений задач осуществлялся в режиме взаимодействия человека с машиной.

Проблемная ориентация ВС достигается соответствующей конфигурацией аппаратных средств и укомплектованием их соответствующим проблемно-ориентированным программным обеспечением.

13.2. ОРГАНИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА В МУЛЬТИПРОГРАММНЫХ ВС

Общая характеристика мультипрограммного режима. Рассмотрим организацию вычислительного процесса современных мультипрограммных ВС на примере машин ЕС ЭВМ, снабженных операционной системой (ОС), структура которой представлена на рис. 13.2.

В состав ОС входят программные средства планирования и организации вычислительного процесса, распределения ресурсов, управления вводом-выводом и данными (*управляющая программа*) и средства подготовки и отладки программ (*системные обрабатывающие программы*).

Операционная система позволяет осуществлять пакетную обработку в однопрограммном и мультипрограммном режимах, работу в режимах разделения времени, реального времени, диалоговом и др.

Важным вопросом в организации вычислительного процесса в ЭВМ общего назначения является использование мультипрограммирования, которое предполагает, грубо говоря, одновременное выполнение ЭВМ нескольких программ. Естественно, в каждый момент времени ЭВМ выполняет команду

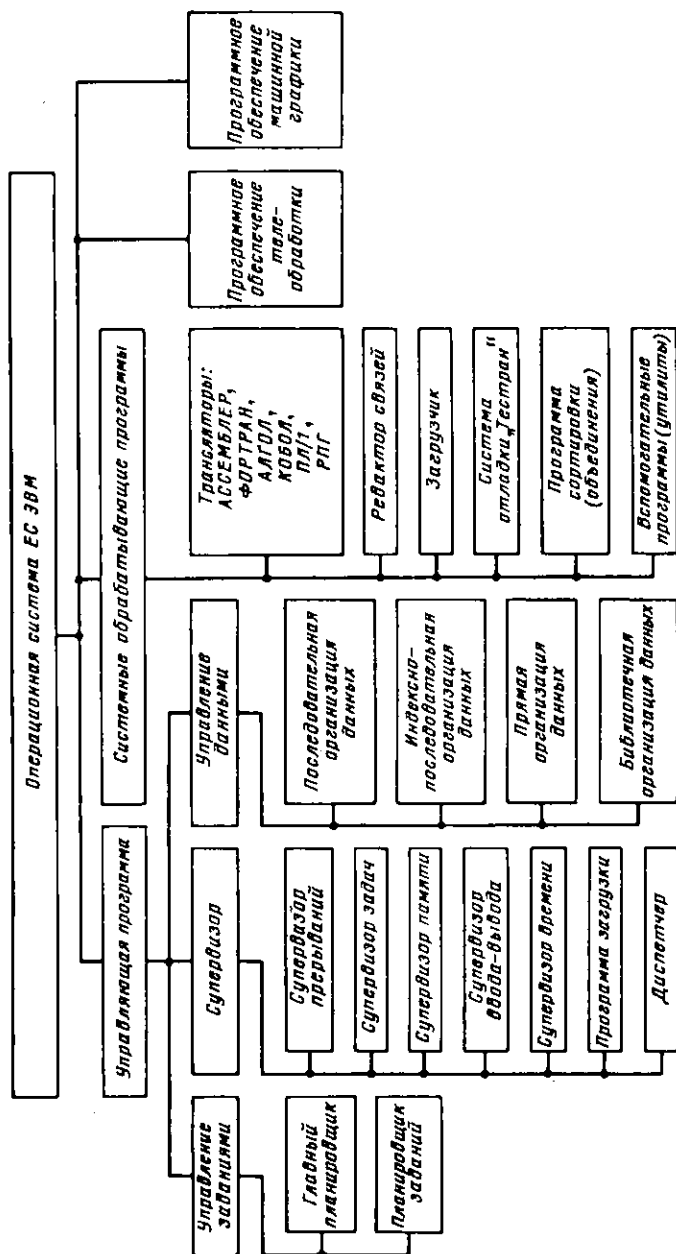


Рис. 13.2. Структура ОС ЕС ЭВМ

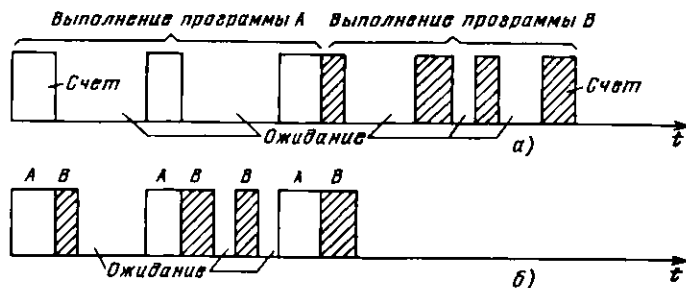


Рис. 13.3. Выполнение программ в однопрограммном (а) и мультипрограммном (б) режимах

какой-то одной программы. Однако всякий раз, когда выполнение процессором некоторой программы приостанавливается из-за необходимости произвести, например, операцию ввода-вывода, процессор переходит к обработке другой готовой для выполнения программы. При этом предполагается, что одновременно в оперативной памяти присутствует несколько программ (или их фрагментов), которые могут находиться в состоянии *активном* (программа обрабатывается на процессоре), *готовности* к обработке или *ожидания* некоторого события, например завершения операции ввода-вывода или освобождения нужного ресурса.

Программа находится в состоянии готовности, если ей предоставлены все необходимые ресурсы, кроме времени процессора.

Мультипрограммирование предназначено для повышения пропускной способности вычислительной установки путем более равномерной и более плотной загрузки всего ее оборудования, в первую очередь процессора. При этом скорость работы самого процессора и номинальная производительность ЭВМ, как она определена в гл. 1, не зависят от мультипрограммирования. Существенной характеристикой для пользователя является пропускная способность, которая оценивается средним объемом вычислений, выполняемых ВС в единицу времени при решении наборов практических задач.

На рис. 13.3 показаны диаграммы выполнения двух программ при однопрограммном и мультипрограммном режимах работы. Как видно из рисунка, общее время выполнения программ А и В при мультипрограммном режиме значительно меньше, чем при однопрограммном. В рассмотренном примере в мультипрограммном режиме сохранились (хотя и существенно уменьшились) паузы в работе процессора. Дальнейшее уве-

личение пропускной способности в рассматриваемом примере можно получить, увеличивая число одновременно обрабатываемых программ (задач) (*коэффициент мультипрограммности*).

Поскольку участвующие в мультипрограммной обработке программы конкурируют за получение времени процессора, доступа к памяти и другим устройствам, между ними должны быть установлены приоритетные соотношения.

Заметим, что переключение программ из состояния готовности в активное состояние связано с некоторыми издержками времени на работу управляющей программы.

Практически за счет мультипрограммирования удается повысить пропускную способность, а следовательно, загрузку процессора в 4—5 раз.

Из сказанного следует, что мультипрограммирование является также средством уменьшения времени выполнения заданного набора программ. Более того, если обрабатывается всего одна программа, то за счет распараллеливания программы и мультипрограммной обработки удастся сократить время ее выполнения.

Распаралеливание программы состоит в выделении участков программы, которые можно выполнять параллельно во времени, и задании требований на синхронизацию обработки этих участков. Далее будет показано, что распараллеливание программ является широко используемым приемом повышения коэффициента мультипрограммности.

Управляющая программа операционной системы, реализующая мультипрограммный режим, должна распределять (в том числе и динамически) между параллельно выполняемыми программами и участками программы ресурсы системы (время процессора, основную и внешнюю память, каналы, устройства ввода-вывода, ключи защиты памяти и др.), с тем чтобы достигалось значительное увеличение пропускной способности с учетом ограничений на ресурсы и требований по срочности выполнения отдельных программ.

Иерархия единиц работы вычислительной системы [82]

На рис. 13.4 представлена иерархия единиц работы ВС, с которыми имеет дело ОС при организации мультипрограммного режима.

Для пользователя внешней единицей работы ВС служит *задание*, соответствующее некоторой совокупности вычислений и представляющее для него законченную работу (расчет сменного задания цехам предприятия, расчет потребности в материалах и т. д.).

Рис. 13.4. Иерархия единиц работы в мультипрограммных ВС

Задания независимы друг от друга и поэтому могут группироваться в пакеты и выполняться в мультипрограммном режиме без синхронизации завершения обработки одних заданий с началом обработки других.

Задание разбивается на выполняемые последовательно части, называемые *пунктами* или *шагами задания*.

Пункт задания предполагает исполнение определенной программы, которая в процессе обработки может инициировать выполнение других программ.

Пункту задания соответствует внутренняя единица работы — *задача пункта задания* или просто *задача*. Задачу можно рассматривать как совокупность ее программы, используемых и формируемых ею наборов данных, сведений о необходимых и выделенных ей ресурсах системы.

В мультипрограммном режиме задачи могут создавать *подзадачи*, а те — в свою очередь, новые подзадачи. В результате программы динамическим образом приобретают параллельную структуру, при которой задачи и подзадачи могут выполняться параллельно во времени, если только это допускают ресурсы системы. В противном случае управляющая программа не выделит ресурсы задачам и подзадам с низким приоритетом, поэтому их создание будет отложено до освобождения ресурсов. Если несколько задач требует одного и того же ресурса, супервизор формирует очередь на занятие этого ресурса.

Особенности мультипрограммных режимов в ЕС ЭВМ

Операционная система ЕС наряду с работой в однопро-

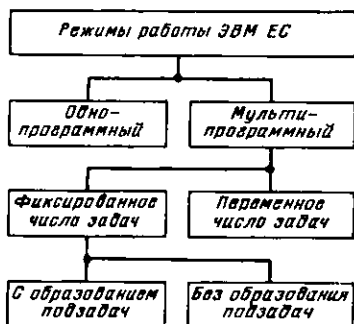
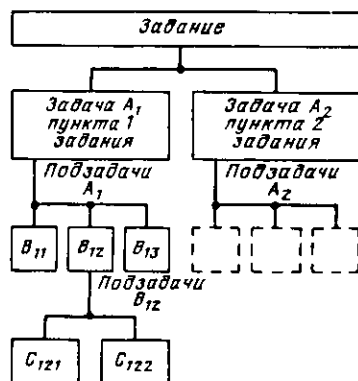


Рис. 13.5. Режимы работы ЭВМ ЕС

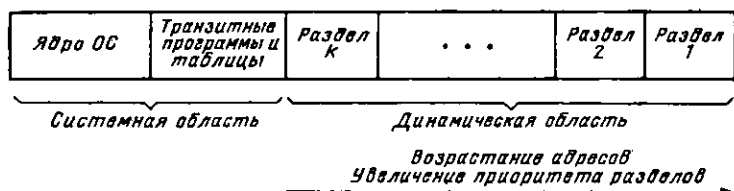


Рис. 13.6. Распределение основной памяти при мультипрограммировании

граммном режиме позволяет реализовать два основных мультипрограммных режима (рис. 13.5): с фиксированным числом задач и с переменным числом задач.

Режим с фиксированным числом задач имеет два варианта: без подзадач и с подзадачами. В режиме без подзадач все программы пункта задания (задачи) выполняются последовательно. В режиме с подзадачами вычисления в пределах пункта задания могут распараллеливаться путем формирования подзадач.

В режимах мультипрограммирования могут совместно выполняться управляющая программа и до 15 пользовательских заданий, так как в системе используется всего 16 ключей защиты памяти. Основная память делится на две области — *системную* и *динамическую* (рис. 13.6). В системной области постоянно находится ядро (*резидентная часть*) операционной системы, по мере необходимости размещаются другие ее программы (*транзиты*), находятся таблицы и системные очереди.

Динамическая область памяти делится на *разделы*. В каждый момент времени раздел обслуживает только одно задание. В режиме с постоянным числом задач число разделов (от 3 до 15) фиксируется при генерации ОС. Тогда же устанавливаются размеры разделов. В режиме с переменным числом задач динамическая область памяти распределяется между разделами не заранее, а в самом процессе обработки заданий, что позволяет лучше использовать ОП и таким образом увеличить коэффициент мультипрограммности. Каждому разделу выделяются определенные периферийные устройства, и за ним закрепляется некоторый класс (подпакет) заданий.

Раздел памяти вместе со своим подпакетом заданий и выделенными периферийными устройствами, а в некоторых случаях и с приписанным ему вариантом режима работы операционной системы, можно рассматривать как отдельную виртуальную машину.

В режиме без подзадач приоритетные соотношения опреде-

ляются положением раздела в памяти — приоритет тем выше, чем больше начальный адрес раздела. При работе с подзадачами каждой задаче и подзадаче присваивается уровень приоритета в пределах 0—255.

Организация обработки пакета заданий

Организация обработки пакета заданий показана на рис. 13.7. Предварительно на языке управления заданиями составляется описание задания и его пунктов, в котором для задания указываются необходимый объем ОП, приоритет и класс, а для каждого пункта — имена используемых программ, входных и выходных наборов данных, приоритет пункта, необходимые объем ОП и устройства. Описания заданий объединяются в пакет.

Обработка пакета заданий организуется и управляется программами: *управление заданиями (главный планировщик, планировщик заданий, системный ввод, системный вывод), супервизор и управление данными* (см. рис. 13.2).

Тексты заданий и входные наборы данных вводятся программой системного ввода с устройств системного ввода (перфокартный ввод, ЗУ на лентах или дисках). При этом программа системного ввода преобразует текст задания в *управляющие таблицы* и помещает последние вместе с наборами входных данных во входную очередь (их всего 15), соответствующую указанному в описании классу задания. Очереди заданий, упорядоченные для каждого класса в соответствии с приоритетом

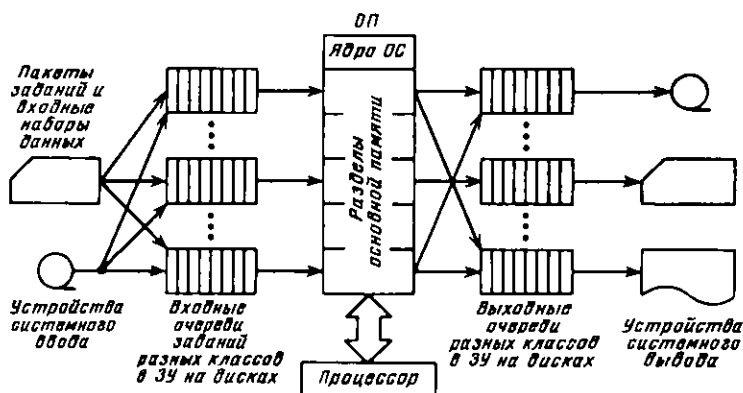


Рис. 13.7. Обработка заданий в мультипрограммном режиме

заданий (принимают значения 0—13), располагаются в ЗУ прямого доступа (ЗУ на дисках).

Планировщик выбирает задания из очереди старшего класса, назначает ему периферийные устройства, выдает сообщение оператору об установке нужных томов (пакетов дисков, катушек с лентой) и загружает программу очередного пункта задания. *Главный планировщик* обеспечивает связь между оператором и операционной системой.

Средства супервизора — супервизор задач, супервизор памяти, супервизор прерываний, супервизор времени и другие — управляют обработкой задач в мультипрограммном режиме, распараллеливают процесс обработки программы, распределяют ресурсы (память, время процессора и др.) между выполняемыми параллельно задачами и подзадачами, следят за освобождением задачами ресурсов, производят динамическую загрузку программ, обрабатывают возникающие прерывания.

Задача, находящаяся в состоянии готовности, переводится супервизором в активное состояние и начинает обрабатываться процессором, если все другие совместно выполняемые задачи с более высоким приоритетом оказываются в состоянии ожидания. Обработка этой задачи будет прервана супервизором, если перейдет в состояние готовности какая-либо задача более высокого приоритета.

Группам заданий (задач), находящимся в ОП, может быть назначена обработка с *квантованием времени*, при которой супервизор поочередно выделяет заданиям один и тот же интервал (*квант*) времени работы процессора. Квантованная обработка позволяет избежать монополизации процессора одной задачей, что особенно важно при диалоговом режиме работы ВС.

Результаты выполнения задания (задачи) помещаются в ЗУ прямого доступа (ЗУ на дисках) в выходную очередь, соответствующую выходному классу задания.

Выходные данные выводятся *программой системного вывода* на одно из устройств системного вывода (перфокартный вывод, печатающее устройство, ЗУ на ленте).

Для организации вычислительного процесса ВС требуются определенные аппаратные и программные средства.

Супервизор — программа, осуществляющая планирование вычислительного процесса, распределение ресурсов системы, обработку прерываний.

Прерывание программ. Часть функций, которые выполняет система прерывания мультипрограммной ВС, свойственна и однопрограммным системам: обеспечение параллельной работы периферийных устройств и процессора, синхронизация

работы ВС с внешними по отношению к ней устройствами. Некоторые другие функции являются специфическими для мультипрограммного режима. Если, например, в однопрограммной машине произошло обращение к памяти по несуществующему в данной конфигурации адресу, то машина остановится. Аналогичное явление произойдет при попытке использования программой несуществующего в данной системе кода операции, лентопротяжного механизма с невставленной катушкой с лентой и т. п. При мультипрограммной работе в подобных ситуациях останавливается выполнение только данной программы, но не система. Следовательно, каждая ситуация, подобная указанному выше, должна вызывать запрос прерывания, позволяющий супервизору указать пользователю на его ошибку, не прекращая исполнения других программ.

Имеются и другие особенности прерывания в мультипрограммных системах. Часть функций системы может выполняться только под управлением супервизора. Целевая программа должна для этого передать управление супервизору. В систему команд мультипрограммной системы должна быть включена команда *«Вызов супервизора»*, которая производит прерывание целевой программы и снабжает супервизор информацией о характере запрошенного действия.

В однопрограммных машинах маскирование какого-либо запроса прерывания обычно приводит к тому, что он запоминается до тех пор, пока запрещение прерывания не будет снято. В мультипрограммных системах действие маски должно иметь избирательный характер. Прерывания, которые должны воздействовать только на какую-либо одну целевую программу, при маскировании должны игнорироваться. В противном случае они могут нарушить работу другой целевой программы, если в последней маска для данного прерывания снята.

Динамическое распределение памяти между программами в ходе самого вычислительного процесса.

Защита памяти, предотвращающая непредусмотренное воздействие одной программы на другую путем запрета вторжения одних программ в области памяти других программ. При отсутствии защиты памяти ошибки в одних программах могут приводить к порче информации, принадлежащей другим программам.

Привилегированные команды. В мультипрограммной системе часть команд может выполняться только супервизором. Выполнение этих команд целевыми программами запрещается, так как это может повлечь за собой нарушение нормальной работы системы. Такие команды называют привилегированными.

К привилегированным операциям относятся установка и изменение кодов защиты памяти, установка и изменение маски прерывания, останов процессора, команды управления устройствами ввода-вывода и др. В мультипрограммных системах в состав процессора вводят триггер, состояние которого задает режим либо *супервизора*, либо целевой программы (*задачи*). Появление привилегированной команды в режиме *задачи* вызывает прерывание.

Датчик времени (таймер). Переключение программ из активного режима в пассивный в мультипрограммной системе с квантованной обработкой производится не только при достижении программой такого состояния, когда она не может дальше работать (например, ожидание ввода-вывода), но и по истечении определенного промежутка времени, отведенного данной программе под цикл активности. Кроме того, должен вестись учет времени, фактически использованного каждой целевой программой для последующей оплаты машинного времени пользователем. Датчик времени представляет собой схему, позволяющую измерять интервал времени между различными программными событиями.

13.3. СТРУКТУРЫ И ПРИНЦИП ДЕЙСТВИЯ ВС КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ (РАЗДЕЛЕНИЯ ВРЕМЕНИ)

По мере развития ВС, расширения состава входящего в них оборудования и средств программного обеспечения все большую актуальность приобретает повышение эффективности их использования. Широкое внедрение вычислительной техники в различные области народного хозяйства, науки и культуры сопровождается увеличением общего количества персонала, связанного с подготовкой и решением задач на вычислительных машинах. Эффективность вычислительной техники в очень большой степени определяется производительностью труда лиц, решающих задачи на машинах. Эта производительность во многом зависит от предоставляемых средствами вычислительной системы возможностей взаимодействия и общения пользователей с машиной в процессе подготовки и отладки программ, а также в процессе решения задач.

При переходе к ЭВМ второго поколения для повышения эффективности использования собственно средств вычислительных систем стал применяться операторный (пакетный) режим решения задач на ЭВМ, при котором пользователи сдают свои задачи оператору ЭВМ, формирующему пакет задач, решаемых в однопрограммном или мультипрограммном режиме

в зависимости от типа системы. В результате пользователь утратил непосредственную связь с ЭВМ, характерную для эксплуатации машин первого поколения.

Ожидание результатов решения по переданной оператору программе затрудняет отладку программ, делает невозможным оперативное внесение в программы изменений по результатам решения, что ведет к снижению эффективности труда программистов по сравнению с эффективностью в режиме непосредственного общения с ЭВМ.

Выход из создавшейся ситуации был найден в идее *вычислительной системы коллективного пользования* (ВСКП), в которой реализуется практически одновременный доступ многих независимых пользователей к ресурсам ВС (мультидоступ).

Мультидоступ предполагает наличие у пользователя своего терминала, например дисплея или пишущей машинки, с помощью которых он может независимо от других пользователей обратиться к системе с запросами на обработку его программ и получить результаты обработки. При одновременном обращении нескольких пользователей ВС должна реагировать на их запросы с задержкой, которая пользователю представляется почти такой же, как и при индивидуальном пользовании. Независимость пользователей означает, что они могут ставить на отладку и обработку не связанные одна с другой программы и при подготовке программ считать, что каждый из них располагает всеми ресурсами системы, в том числе всей ее памятью.

Если все запросы пользователей имеют примерно одинаковую длину и не требуют много машинного времени, как это имеет место в информационно-справочных системах или автоматизированных обучающих системах, то запросы формируются в пакет и обрабатываются согласно дисциплине обслуживания «первый пришел — первый ушел» или какой-либо другой, в том числе и с учетом различных приоритетов запросов. Такие ВСКП (системы типа «запрос — ответ») работают без квантованного обслуживания запросов.

Более характерным является случай, когда в ВСКП поступают запросы, сильно отличающиеся по необходимому машинному времени, причем заранее нельзя определить продолжительность обработки того или иного запроса. Такая ситуация характерна для любого вычислительного центра без постоянной номенклатуры расчетов.

В этом случае поддержание в ВСКП у пользователей иллюзии прямого общения с ВС и обладания ее ресурсами требует уменьшения времени отклика системы в первую очередь на короткие запросы. Действительно, пользователи, затребовавшие

выполнения программ, расходующих много машинного времени, склонны спокойно ждать обслуживания сравнительно долго. В то же время пользователи, запрашивающие решение коротких задач, оказываются более нетерпеливыми, и поэтому такие запросы должны получить обслуживание в первую очередь.

Для создания эффекта прямого контакта пользователя с системой путем сокращения времени ожидания ответа на короткие запросы в ВСКП применяют *квантованное обслуживание*, или, другими словами, *разделение времени* при выполнении запросов.

Вычислительная система с разделением времени (СРВ), работающая в мультипрограммном режиме, в определенной последовательности обслуживает пользователей, выделяя программе (запросу) каждого пользователя некоторый интервал времени (*квант обслуживания*). Если в течение выделенного программе кванта времени ее обработка не заканчивается, программа прерывается и становится в очередь программ, ожидающих обработки.

Порядок распределения между пользователями основного ресурса СРВ — времени процессора и соответственно других ее ресурсов — устанавливается дисциплиной квантованного обслуживания. Дисциплина обслуживания выбирается с учетом следующих требований: обеспечения режима прямого контакта пользователя с вычислительной системой, высокого уровня эффективности использования технических средств вычислительной системы, простоты аппаратных и программных средств, обеспечивающих реализацию принятой дисциплины обслуживания. Поскольку нельзя допустить фактической монополизации времени процессора и других средств вычислительной системы каким-либо пользователем, то кажется естественной и одновременно технически просто осуществимой дисциплина обслуживания, при которой всем программам выделяется один и тот же квант времени, по возможности меньший, чтобы промежутки между интервалами активности программ пользователя были минимальны.

Однако уменьшение выделяемого программам кванта времени ведет к снижению эффективности использования технических средств системы, так как переключение программ из пассивной фазы в активную сопровождается служебными операциями, доля которых при этом в общем времени работы системы возрастает. Значение кванта времени выбирается в результате разумного компромисса, с тем чтобы и время ожидания ответа на запрос, и потери производительности от переключения активности программ были приемлемыми.

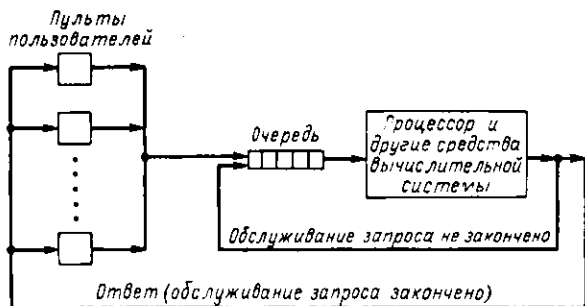


Рис. 13.8. Система разделения времени с одноочередной дисциплиной обслуживания

Дисциплины обслуживания в СРВ. Было предложено несколько дисциплин обслуживания пользователей в СРВ. Их можно разбить на две основные группы: одноочередные и многоочередные.

Одноочередная дисциплина обслуживания. Схема одноочередной СРВ представлена на рис. 13.8. Вновь поступающие от пользователей запросы (программы) ставятся в конец очереди. Для обслуживания выбирается программа из начала очереди, и ей отводится квант времени Δ процессора. Если за это время программа успеет завершиться, результат будет выдан пользователю, а процессор приступит к обработке следующей из очереди программы. Если программа за выделенный квант времени не закончится, обработка ее прерывается и она поступает в конец очереди.

Иногда с целью уменьшения времени отклика системы на короткие (по времени выполнения) запросы рассматриваемая дисциплина модифицируется, и вновь поступающий в систему запрос ставится в начало очереди. В остальном процесс функционирования остается без изменения.

Многоочередная дисциплина обслуживания. Для сокращения потерь времени, связанных с переключением программ, и обеспечения приоритета для коротких запросов (программ) используют дисциплину обслуживания с несколькими очередями (рис. 13.9). Приоритет очереди убывает с возрастанием ее номера m ($0 \leq m \leq k$). Вновь поступающий в СРВ запрос ставится в конец очереди, имеющей старший приоритет ($m=0$).

В такой системе следующим подлежит обслуживанию запрос из начала очереди с номером m , если очереди с меньшими номерами (с более высоким приоритетом) пустые. Если за выделенный программе квант времени ее обработка не заканчи-

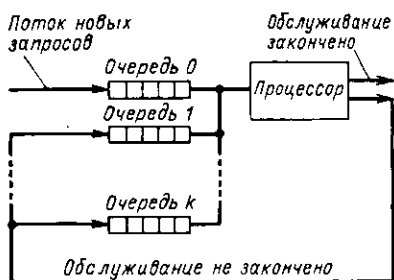


Рис. 13.9. Система разделения времени с многоочередной дисциплиной обслуживания

вается, программа прерывается и переходит в конец очереди с номером $m+1$. Программы из последней очереди обслуживаются до конца без прерывания. Выделяемый программе квант времени возрастает с увеличением номера очереди обычно по правилу

$$\Delta_m = 2^m \Delta,$$

где Δ_m — квант, выделяемый программе из m -й очереди ($0 \leq m \leq k$); Δ — квант для программы из нулевой очереди. Обычно $\Delta = 200 \div 500$ мс.

Если во время обслуживания программы из очереди m в очереди с большим приоритетом появляется новая программа, то после окончания текущего кванта Δ обрабатываемая программа прерывается и возвращается в начало своей очереди, с тем чтобы впоследствии дополучить недоданное по отношению к значению $2^m \Delta$ время.

Чтобы избежать недопустимо долгого ожидания для больших программ, приоритет делается зависящим от времени ожидания. Если ожидание превысит некоторое установленное значение (например, 60 с), программа переходит в следующую очередь с меньшим номером. Таким образом, в рассматриваемой системе приоритетные соотношения между запросами устанавливаются в самом процессе выполнения программ (*динамические приоритеты*).

Описанный механизм распределения времени обладает определенными адаптивными свойствами: программы с большей длительностью обработки вытесняются в очереди с большими номерами. Этот процесс можно сделать более эффективным, если при вводе запросов сразу распределять запрашиваемые программы по очередям, используя для этого некоторые оценки.

В первом приближении можно считать, что продолжительность выполнения программы пропорциональна ее длине (чис-

лу байт). По меньшей мере от длины программы прямо зависит время, затрачиваемое на передачу программы между оперативной и внешней памятью при переключении ее активности.

В алгоритме планирования Ф. Корбатто, реализованном в одной из первых СРВ [97], программа сразу поступает в очередь, имеющую номер

$$i = \lfloor \log_2 (l_p / l_\Delta + 1) \rfloor,$$

где l_p — число байт в программе; l_Δ — число байт, которое может быть передано из оперативной памяти во внешнюю или обратно за время Δ ; от выражений, стоящих в квадратных скобках, берется целая часть. В остальном дисциплина обслуживания соответствует описанной выше.

Процесс взаимодействия пользователя с СРВ показан на рис. 13.10. Пользователь со своего пульта вводит запрос (требование на выполнение некоторой программы и указание на используемые исходные данные).

Программа пользователя обслуживается в режиме разделения времени в соответствии с принятой в системе дисциплиной квантованного обслуживания до того момента, когда или обработка программы полностью завершится и произойдет вывод результата обработки на пульт пользователя, или программа дойдет до точки, когда требуются от пользователя новые указания или данные. В первом случае пользователь ставит системе новую задачу, во втором сообщает дополнительные

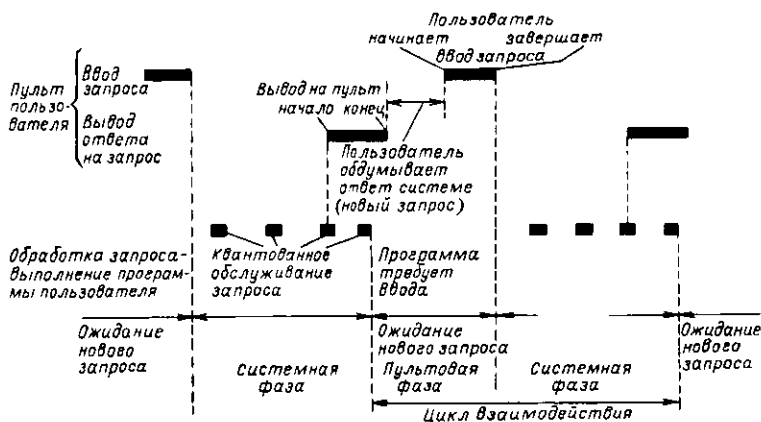


Рис. 13.10. Взаимодействие пользователя с СРВ

данные, позволяющие системе продолжить обработку его программы. В дальнейшем не будем делать различия между этими случаями, считая, что всякий раз имеет место посылка пользователем в систему нового запроса на обработку некоторой программы.

Важнейшим элементом ВСКП является сам пользователь, и поэтому без внимательного изучения его нужд, особенностей его взаимодействия с системой, соответствующих физиологических ограничений невозможно создавать эффективно действующие СРВ.

13.4. ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ. ПРИНЦИП ДЕЙСТВИЯ УПРАВЛЯЮЩЕГО ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА АСУ ТП

В ряде применений ВС, таких, как автоматизация научных экспериментов, испытание сложных изделий, контроль и управление различными технологическими процессами, на обработку вводимых данных и выдачу результата накладываются жесткие временные ограничения, диктуемые темпом развития процессов в контролируемом или управляемом вычислительной системой объекте. Говорят, что такие системы работают *в реальном времени*.

Вычислительной системой реального времени (системой, работающей в реальном времени) называется система, соединенная с некоторым внешним объектом и обрабатывающая поступающую в нее информацию о его состоянии достаточно быстро для того, чтобы результат обработки мог использоваться для воздействия на протекание процесса в объекте. Обычно к системам реального времени относят системы, в которых реакция (ответ) на поступающую от управляемого объекта информацию (определение соответствующего управляющего воздействия) составляет несколько долей секунды или несколько секунд.

Характерным примером системы реального времени является автоматизированная или автоматическая система управления технологическим процессом (АСУ ТП).

Автоматизированная система управления технологическим процессом предназначена оптимизировать технологический процесс при переменных значениях параметров, влияющих на него, с тем чтобы обеспечивались максимальная производительность оборудования, экономия материалов и энергии и т. п. при условии выполнения требований к качеству конечного

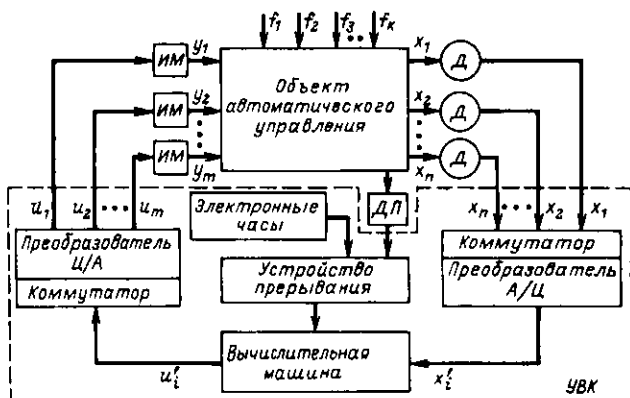


Рис. 13.11. Упрощенная структурная схема АСУ ТП с управляющим вычислительным комплексом

продукта. В АСУ ТП для обработки информации о протекании технологического процесса и его оптимизации используют управляющие вычислительные комплексы (УВК), создаваемые на основе малых и микро-ЭВМ.

Упрощенная структурная схема АСУ ТП с УВК в замкнутом контуре управления представлена на рис. 13.11.

Технологический процесс характеризуется следующими группами параметров:

1. Измеряемые параметры $x_1, x_2, \dots, x_n$, к которым относятся:

а) измеряемые, но нерегулируемые параметры, зависящие от внешних факторов (характеристики исходного продукта и др.);

б) выходные параметры, характеризующие качество продукции;

в) выходные параметры, по которым непосредственно или путем расчета определяются эффективность производственного процесса (производительность, экономичность и др.) или ограничения, наложенные на условия его протекания.

2. Регулируемые параметры $y_1, y_2, \dots, y_m$, которые могут изменяться соответствующими исполнительными механизмами ИМ, уставками регуляторов и т. п.

3. Нерегулируемые и неизмеряемые параметры $f_1, f_2, \dots, f_k$ (например, химический состав сырья, изменяющиеся со временем характеристики оборудования и др.).

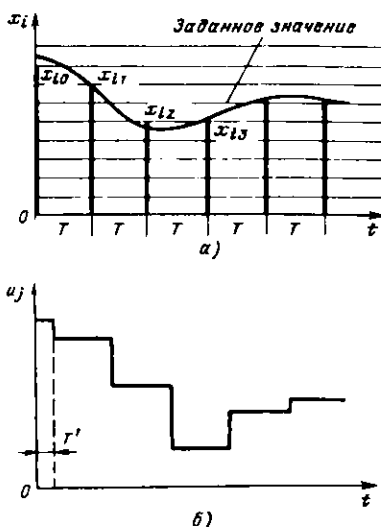
На вход УВК от соответствующих датчиков D (термопар, расходомеров, измерителей толщины и др.) поступает измерительная информация о текущих значениях параметров $x_1, x_2, \dots, x_n$. Управляющий вычислительный комплекс обрабатывает эту информацию в соответствии с принятым законом управления (алгоритмом управления), определяет управляющие воздействия $u_1, u_2, \dots, u_m$, которые необходимо приложить к исполнительным механизмам для изменения регулируемых параметров $y_1, y_2, \dots, y_m$ с тем чтобы управляемый процесс протекал оптимальным образом. Измерительные датчики, как правило, вырабатывают свои сигналы в виде напряжения, тока или угла поворота (в форме аналогового непрерывного сигнала). Некоторые входные сигналы (например, сигналы конечных выключателей, фотореле и др.) имеют форму аналоговых дискретных сигналов (сигналы типа ДА — НЕТ). Подводимые к исполнительным механизмам управляющие воздействия $u_1, u_2, \dots, u_k$ должны вырабатываться в форме напряжений, т. е. в аналоговой форме.

ЭВМ оперируют с цифровыми величинами. Поэтому поступающие на ее вход аналоговые величины $x_1, x_2, \dots, x_n$ предварительно преобразуются в цифровую форму x'_i ($i = 1, \dots, n$), а вырабатываемые ЭВМ управляющие воздействия u'_j преобразуются из цифровой формы в аналоговую — в соответствующие напряжения.

Таким образом, УВК должен содержать на входе преобразователь аналоговых непрерывных величин в цифровые (преобразователь А/Ц) для преобразования измерительной информации, а на выходе — преобразователь цифровых величин в аналоговые (преобразователь Ц/А) для преобразования управляющей информации (величин u'_j).

Для уменьшения объема оборудования преобразователи аналоговых величин в цифровые и обратно выполняются многоканальными. С помощью коммутатора преобразователь поочередно подключается к каждому датчику и осуществляет преобразование соответствующей аналоговой величины в цифровую, после чего полученный в результате преобразования код вводится в память УВК. Аналогичным образом коммутатор в выходных цепях обеспечивает передачу в нужную цепь управляющего воздействия, преобразованного из цифровой формы в соответствующее напряжение. Это управляющее напряжение фиксируется в данной цепи, т. е. сохраняется при помощи специальной схемы неизменным, пока в следующем цикле управления УВК не выработает новое значение управляющего воздействия.

Рис. 13.12. Принцип работы управляющего вычислительного комплекса в АСУ ТП



Наибольшее распространение получил синхронный принцип связи УВК с объектом, при котором процесс управления разбивается на циклы равной продолжительности тактирующими сигналами электронных часов. Упрощенная временная диаграмма работы УВК при синхронном принципе связи представлена на рис. 13.12.

Цикл начинается с приходом тактирующего сигнала на устройство прерывания. В начале каждого цикла производится последовательный опрос и преобразование в цифровую форму сигналов датчиков. Преобразование и ввод преобразованных величин в память УВК, как правило, занимают ничтожно малое время по сравнению с интервалом времени, в течение которого измеряемые величины успевают заметно измениться.

Обычно можно считать, что вся введенная в данном цикле измерительная информация соответствует одному и тому же моменту времени. При необходимости алгоритмы управления могут учитывать неодновременность снятия измерительной информации из разных каналов. Совокупность последовательных преобразований одной и той же входной величины превращает непрерывную функцию входной величины в решетчатую функцию, состоящую из ординат, соответствующих значениям непрерывной функции в моменты преобразования. Таким образом, в процессе преобразования непрерывных аналоговых величин в цифровую форму осуществляется *квантование по времени* вводимых в УВК величин (рис. 13.12, а).

Так как преобразование аналоговой величины в цифровой эквивалент происходит с точностью, соответствующей числу разрядов цифрового кода, то процесс преобразования сопровождается также *квантованием по уровню* входных величин.

После того как вся измерительная информация преобразована и передана в память, УВК в течение времени T' произво-

дит расчет необходимых управляющих воздействий и осуществляет преобразование их в аналоговую форму (рис. 13.12, б). Затем УВК или останавливается до поступления следующего тактирующего сигнала, или переходит к каким-либо вспомогательным расчетам, которые могут прерываться (без порчи программы и промежуточных результатов) тактирующими сигналами.

В ряде установок необходима более тесная связь объекта с УВК. В таких случаях используется асинхронный принцип связи УВК с объектом. Вместо тактирующих сигналов в УВК поступают сигналы от датчиков прерывания ДП (см. рис. 13.11), непосредственно связанных с объектом (например, конечных выключателей, датчиков аварийного состояния и др.). Каждый сигнал прерывания эквивалентен требованию о прекращении производимых вычислений и переходе к выполнению подпрограммы, соответствующей данному каналу прерывания. Управляющий вычислительный комплекс реагирует на сигналы прерывания с учетом приоритета одних сигналов прерывания перед другими.

Применяют также комбинированный способ связи УВК с объектом: наряду с электронными часами, дающими тактирующие сигналы, используются связанные с объектом датчики прерывания, например датчики аварийных сигналов, переводящие УВК на работу по программе для аварийного режима.

В системе, схема которой изображена на рис. 13.11, УВК, воздействуя на исполнительные механизмы, непосредственно осуществляет управление производственным процессом. Такой режим работы УВК называется *прямым цифровым управлением*. Однако для сложных технологических объектов, содержащих несколько агрегатов, связанных между собой через технологический процесс, система управления обычно строится таким образом, что отдельные параметры процесса регулируются соответствующими автоматическими регуляторами, а УВК, обрабатывая измерительную информацию, рассчитывает и устанавливает оптимальные настройки этих регуляторов.

При подобном построении схемы управления повышается надежность системы, так как ее работоспособность сохраняется и при отказах в работе УВК. При такой схеме УВК может быть более простым, так как снижаются требования к его быстродействию и другим характеристикам. В то же время появляется практическая возможность реализации более эффективных алгоритмов оптимизации, требующих большого объема вычислений.

Многие важные задачи управления производственными процессами решаются при работе УВК в разомкнутом контуре

управления. По такому принципу построены системы числового программного управления станками и системы, в которых УВК выполняет функции советчика.

В режиме советчика УВК обрабатывает измерительную информацию, поступающую с контролируемого объекта, и в результате соответствующих расчетов определяет, какие регулирующие воздействия следует произвести для того, чтобы производственный процесс протекал наилучшим образом. Выработанная УВК информация служит рекомендацией оператору, управляющему процессом. Работа УВК в режиме советчика, как правило, предшествует включению УВК в замкнутый контур управления.

Глава четырнадцатая

ПРИНЦИПЫ ОРГАНИЗАЦИИ МНОГОУРОВНЕВОЙ ПАМЯТИ В МУЛЬТИПРОГРАММНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

14.1. ПРОБЛЕМЫ ОРГАНИЗАЦИИ ПАМЯТИ МУЛЬТИПРОГРАММНЫХ СИСТЕМ

Память современных вычислительных систем имеет иерархическую многоуровневую структуру. Чем выше уровень, тем выше быстродействие соответствующей памяти, но меньше ее емкость. К верхнему уровню относятся запоминающие устройства, с которыми процессор непосредственно взаимодействует в процессе выполнения программы — основная или оперативная память (ОП) (см. гл. 4).

Сравнительно небольшая емкость ОП (64—8000 Кбайт) компенсируется практически неограниченной емкостью внешних запоминающих устройств на магнитных дисках и лентах (сотни миллионов или миллиарды байт). Однако эти устройства сравнительно медленные, и время обращения за данными для дисков составляет десятки миллисекунд, а для лент может достигать сотен секунд. Поэтому вычислительный процесс должен протекать с возможно меньшим числом обращений к внешним запоминающим устройствам и максимально возможным использованием ОП.

Оперативная память является наиболее дефицитным ресурсом в вычислительных системах, которым надо пользоваться

экономно и эффективно. Проблема усложняется при переходе к мультипрограммным системам, так как в них ОП одновременно используют несколько программ (заданий). В таких системах необходимо исключить несанкционированное воздействие одних программ на другие. Это достигается с помощью механизма *защиты памяти*.

Эффективное распределение ресурса памяти между программами не может быть статическим, т. е. производиться предварительно до пуска программы. В процессе обработки программ потребности в ресурсе памяти отдельных программ изменяются, что заранее не может быть учтено. Необходимо распределять память между программами динамически непосредственно в ходе вычислительного процесса, т. е. осуществлять *динамическое распределение памяти*, при этом должна обеспечиваться возможность независимой работы программистов над своими программами, подлежащими мультипрограммной обработке. Динамическое распределение памяти не должно приводить к дроблению ее свободного пространства — *фрагментации памяти*, затрудняющему его использование. Это достигается использованием *одноуровневой виртуальной памяти*, допускающей адресацию на все адресное пространство, размер которого определяется полем адресного кода команды или базового регистра.

Быстродействие ОП часто оказывается недостаточным для обеспечения требований, предъявляемых к скорости работы больших ЭВМ. Поэтому в состав процессора включают буферную сверхоперативную память небольшого объема и повышенного быстродействия, хранящую команды и данные, относящиеся к обрабатываемому участку программы.

14.2. ЗАЩИТА ПАМЯТИ

Если в памяти одновременно могут находиться несколько независимых программ, необходимы специальные меры для предотвращения или ограничения обращений одной программы к областям памяти, используемым другими программами. Программы могут содержать такие ошибки, которые, если этому не воспрепятствовать, приводят к искажению информации, принадлежащей другим программам. Последствия таких ошибок особенно опасны, если разрушению подвергнутся программы операционной системы. Другими словами, надо исключить воздействие программы пользователя на работу программ других пользователей и программ операционной системы.

Чтобы воспрепятствовать разрушению одних программ другими, достаточно защищать область памяти данной программы от попыток записи в нее со стороны других программ, а в некоторых случаях и своей программы (*защита от записи*), при этом допускается обращение других программ к этой области памяти для считывания данных.

В других случаях, например при ограничениях на доступ к информации, хранящейся в системе, необходимо иметь возможность запрещать другим программам производить как запись, так и считывание в данной области памяти. Такая *защита от записи и считывания* помогает отладке программы, контролируя каждый случай выхода за область памяти своей программы.

Для облегчения отладки программ желательно выявлять и такие характерные ошибки в программах, как попытки использования данных вместо команд или команд вместо данных в собственной программе, хотя эти ошибки могут и не разрушать информацию.

Отметим следующие варианты дифференцированной защиты при различных операциях с памятью:

1) задается отношение к области памяти чужой программы, определяющее, относится защита памяти только к операции записи или к любому обращению в память;

2) задается одно из следующих отношений к области памяти собственной программы:

а) разрешается доступ к данному блоку как для записи, так и для считывания;

б) разрешается только считывание;

в) разрешается обращение любого вида, но по адресу, взятому только из счетчика команд;

г) разрешается обращение по адресу из любого регистра, кроме счетчика команд.

Если нарушается защита памяти, исполнение программы приостанавливается и вырабатывается запрос прерывания по нарушению защиты памяти.

Защита от вторжения программ в чужие области памяти может быть организована различным образом, при этом реализация защиты не должна заметно снижать производительность машины и требовать слишком больших аппаратных затрат.

Защита отдельных ячеек памяти. В управляющих вычислительных комплексах, предназначенных для работы в АСУ ТП, необходимо обеспечить возможность отладки новых программ параллельно с функционированием находящихся в памяти рабочих программ, управляющих технологическим процессом.

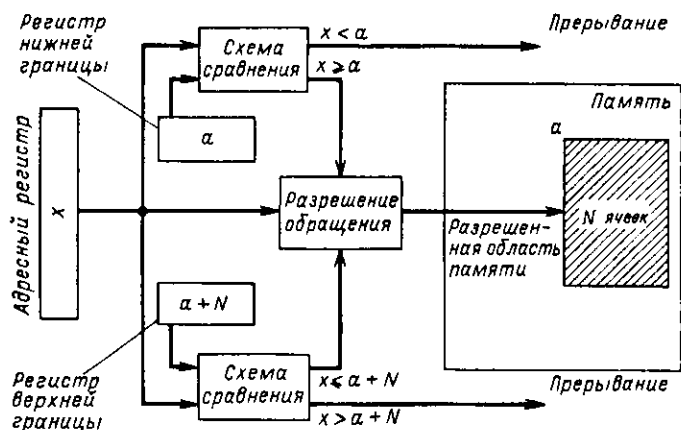


Рис. 14.1. Защита памяти с помощью граничных регистров

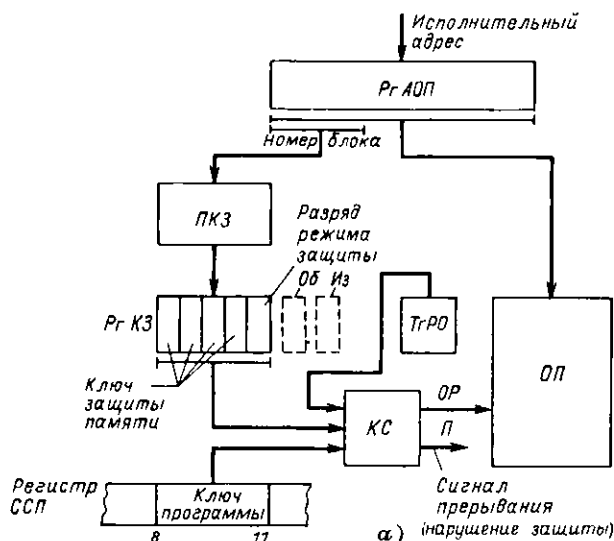
Это может быть достигнуто выделением в каждой ячейке памяти специального «разряда защиты». Установка 1 в этот разряд запрещает производить запись в данную ячейку.

В системах с мультипрограммной обработкой большого числа программ защищаются не отдельные ячейки, а области памяти или блоки, на которые делится память, при этом часто предусматривается возможность указывать для разных программ различные допустимые режимы обращения к отдельным областям или блокам памяти.

Метод граничных регистров (рис. 14.1) состоит во введении двух граничных регистров, указывающих верхнюю и нижнюю границы области памяти, куда программа имеет право доступа. При каждом обращении к памяти проверяется, находится ли используемый адрес в установленных границах; при выходе за границы обращение к памяти подавляется и формируется запрос прерывания, передающий управление операционной системе. Содержание граничных регистров устанавливается операционной системой перед тем, как для очередной целевой программы начнется активный цикл. Если для динамического распределения памяти используется базовый регистр, то он одновременно определяет и нижнюю границу. Верхняя граница подсчитывается операционной системой в соответствии с длиной программы в оперативной памяти.

Метод ключей защиты. По сравнению с предыдущим данный метод является более гибким: он позволяет организовать доступ программы к областям памяти, расположенным не подряд.

Память в логическом отношении делится на одинаковые блоки. Каждому блоку памяти ставится в соответствие код, называемый ключом защиты памяти, а каждой программе, принимающей участие в мультипрограммной обработке, присваивается код ключа программы. Доступ программы к данному



Ключ защиты блока А	Разряд режима защиты	Ключ программы в ССП	Описание	Результат
1101	0	1101	Запись в А, считывание из А	Разрешено (OP=1)
1101	1	1101	Запись в А, считывание из А	
0011	0	1101	Запись в А, считывание из А	Запрещено (П=1) разрешено (OP=1)
0011	1	1101	Запись в А, считывание из А	
0011	0	0000	Запись в А, считывание из А	Разрешено (OP=1)
0011	1	0000	Запись в А, считывание из А	

Рис. 14.2. Защита памяти при помощи ключей защиты. Структура защиты памяти (а) и различные ситуации в работе защиты (б) [дополнительные разряды ключа Об (обращение) и Из (изменение) используются в режиме виртуальной памяти (см. § 14.4)]

блоку памяти для чтения и записи разрешен, если ключи совпадают или один из них имеет код 0.

Метод ключей защиты используется в ЕС ЭВМ. Блоки содержат по 2048 байт. В мультипрограммном режиме могут одновременно обрабатываться до 16 (включая операционную систему) программ. Поэтому для исключения влияния программ друг на друга следует иметь возможность задавать 16 различных вариантов кодов ключей защиты памяти и ключей программы, для чего необходимы 4-разрядные коды. Четырехразрядный ключ программы указывается в специальном поле слова состояния программы (ССП), а при операциях ввода-вывода — в слове состояния канала (ССК). В ключе защиты памяти предусматривается дополнительный, пятый (младший), *разряд режима защиты*. Защита действует только при попытке записи в блок, если в этом разряде стоит 0, и при записи и считывании, если стоит 1. Коды ключей защиты памяти и ключей программы устанавливаются операционной системой с помощью специальных команд, относящихся к привилегированным операциям.

Коды ключей защиты памяти хранятся в специальной *памяти ключей защиты*, более быстродействующей, чем ОП. В машине ЕС-1030 эта память выполнена на цилиндрических магнитных пленках, а в ЭВМ ЕС-1045 — на интегральных микросхемах.

Функционирование защиты памяти поясняется схемой на рис. 14.2. При обращении к памяти исполнительный адрес поступает в регистр адреса оперативной памяти *РгАОП*. Запускается цикл ОП, и одновременно группа старших разрядов кода исполнительного адреса, соответствующая номеру блока, к которому производится обращение, используется как адрес для выборки из памяти ключей защиты *ПКЗ* и передачи на регистр ключа защиты памяти *РгКЗ* кода ключа защиты памяти, присвоенного операционной системой данному блоку. Комбинационная схема *КС* сравнивает ключ защиты памяти блока и ключ программы в регистре ССП или ССК и вырабатывает с учетом режима обращения (запись или считывание), указываемого триггером режима обращения *ТгРО*, и режима защиты (устанавливается разрядом режима защиты в ключе защиты памяти) сигнал «Обращение разрешено» ($OP = 1$) либо сигнал «Прерывание» ($P = 1$) по нарушению защиты.

Если ключи защиты блока памяти и программы совпадают или при любом ключе защиты памяти ключ программы имеет код 0000 (такой ключ программы присвоен программе операционной системы), то разрешены обращения к данному блоку для записи и считывания независимо от назначения разряда ре-

жима защиты. Обращения также разрешены при любом ключе программы, если ключ памяти имеет код 0000 (такой ключ обычно имеют блоки, хранящие стандартные подпрограммы).

14.3. ОРГАНИЗАЦИЯ РАБОТЫ ПАМЯТИ В РЕЖИМЕ МНОГОАБОНЕНТНОГО ОБСЛУЖИВАНИЯ

В современных мультипрограммных ВС для повышения производительности реализуется независимая параллельная работа во времени процессора, мультиплексного и селекторных каналов и некоторых других устройств (например, таймера), которые асинхронно и независимо друг от друга формируют запросы на обращение к ОП. Более того, в крупных моделях ЭВМ в процессоре имеется ряд параллельно работающих блоков, независимо друг от друга обращающихся к ОП. Например, в ЭВМ ЕС-1050 такими блоками процессора является блок контроля и диагностирования (БКД), блок адреса результата (БАР), блок центрального управления (БЦУ) и блок выборки команд (БВК).

Перечисленные выше устройства и блоки можно рассматривать как абоненты памяти. Возникает задача обеспечения работы ОП в режиме многоабонентного обслуживания. Этот режим должен осуществляться таким образом, чтобы эффективно использовалась для повышения общей пропускной способности памяти ее модульная организация, допускающая независимые обращения к отдельным модулям, обеспечивалась защита памяти, контроль считываемых и записываемых данных, контроль нарушения адресации.

Указанные выше функции реализуются блоком обращения к оперативной памяти (БОП). Рассмотрим принцип построения БОП, подобного блоку ЭВМ ЕС-1050 [27].

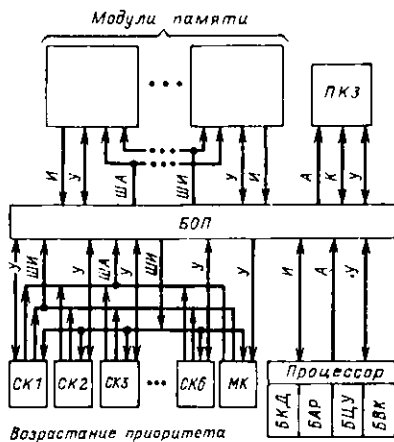


Рис. 14.3. Многоабонентный режим работы ОП:

ПКЗ — память ключей защиты; ША и ШИ — адресные и информационные коллективные шины; И, А, У, К — индивидуальные шины соответственно передачи информации, адреса, управляющих сигналов и ключей защиты

Возрастание приоритета

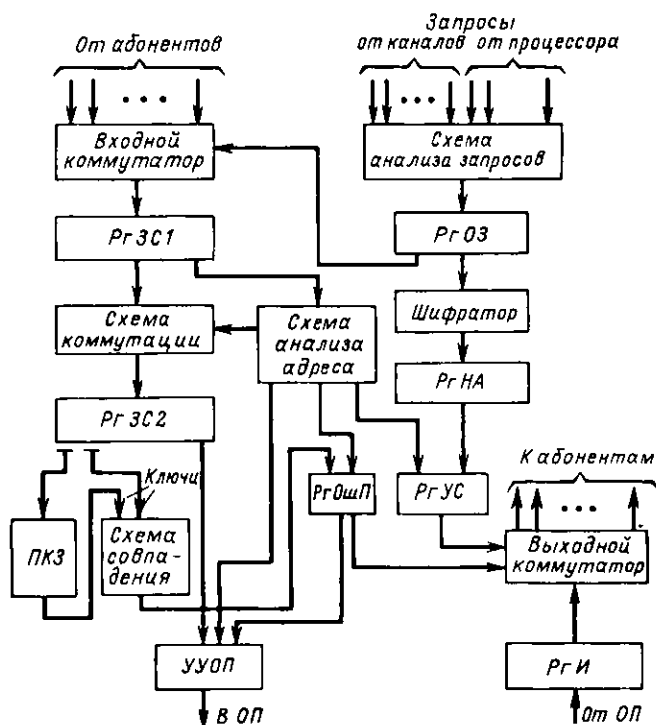


Рис. 14.4. Структурная схема блока обращений к оперативной памяти

Приоритеты запросов отдельных устройств и блоков на обслуживание со стороны ОП упорядочиваются согласно рис. 14.3. Каналы имеют более высокий приоритет, чем процессор. Это связано с опасностью потери информации при работе с движущимся носителем.

Как видно из рис. 14.3, связи БОП с абонентами, модулями ОП и памятью ключей защиты (ПКЗ) осуществляются коллективными и индивидуальными (радиальными) шинами.

На рис. 14.4 приведена упрощенная структурная схема БОП. Обработка запроса в БОП состоит из трех этапов. Поскольку на каждом этапе используются отдельные средства, возникает своеобразный конвейер, и при наличии запросов одновременно обрабатываются три запроса по одному для каждого этапа.

1. *Обработка сигналов запросов.* Схема анализа приоритетов запросов циклически с периодом, равным времени обработки одного запроса в ОП (в ЭВМ ЕС-1050 это время равно 320 нс), анализирует поступившие от абонентов запросы, выделяет наиболее приоритетный из

них и фиксирует его наличие засылкой 1 в соответствующий данному абоненту разряд регистра обобщенных запросов $P_2OЗ$.

Далее по содержимому $P_2OЗ$ шифратор формирует в регистре номера адреса $P_2НА$ 4-разрядный номер абонента, принятого на обслуживание. При считывании информации из ОП номер абонента определяет, куда выдается информация. Код в $P_2OЗ$, управляя входным коммутатором, обеспечивает прием «запросного слова» (ЗС) соответствующего абонента в $P_2ЗС1$. Запросное слово содержит адрес ОП, признак обращения (запись или считывание), ключ программы, а при записи и информационное слово. На этом завершается первый этап.

2. *Обработка запросного слова.* Схема анализа адреса проверяет код адреса на нечетность, допустимость адреса для данной конфигурации ОП (нарушение адресации). Она также определяет по старшим разрядам $P_2ЗС1$ номер модуля ОП, свободен ли этот модуль и формирует семиразрядное управляющее слово. Это слово содержит 4-разрядный номер абонента, пославшего данный адрес, 2-разрядный номер модуля ОП и 1-разрядный признак операции обращения (запись или считывание).

Если адресуемый ЗС модуль памяти оказывается занятым операцией обращения для другого ЗС (в том числе от данного абонента), то ЗС остается в $P_2ЗС1$ до освобождения модуля памяти, если ЗС принадлежит каналу, и освобождает $P_2ЗС1$, если принадлежит процессору. В последнем случае запросы процессора не обрабатываются до освобождения модуля памяти, оказавшегося занятым.

Если запрашиваемый модуль ОП свободен, ЗС передается в $P_2ЗС2$ из $P_2ЗС1$, который таким образом освобождается для приема нового ЗС. На этом заканчивается второй этап.

3. *Обращение к модулю ОП.* В соответствии с содержимым управляющего слова запускается цикл обращения в определенном модуле ОП, записывается по указанному в ЗС адресу информационное слово из ЗС или производится считывание слова в $P_2И$. Далее считанная информация через выходной коммутатор, управляемый кодом номера абонента в $P_2УС$, передается соответствующему абоненту.

Одновременно с запуском цикла в ОП старшие разряды адреса (номер блока) передаются в качестве адреса в память ключей защиты. Происходит сопоставление ключа защиты из $ПКЗ$ и ключа программы в ЗС. При их несоответствии в определенном разряде регистра ошибок памяти $P_2ОП$ фиксируется нарушение защиты. В случае выявления нарушений при обращении к памяти (нарушение адресации или защиты), на что указывает наличие 1 в соответствующем разряде 3-разрядного регистра ошибки памяти $P_2ОшП$, блокируется запись слова из ЗС, а при считывании блокируется выдача считанного слова из $P_2И$ абоненту (выдается нулевой код). Содержимое $P_2ОшП$ выдается в регистр ошибок абонента.

Отметим в заключение, что с помощью БОП производятся запись ключей защиты в $ПКЗ$ и считывание ключа при его передаче в регистр ССП.

Многоабонентное обслуживание порождает конфликты, возникающие при одновременном обращении абонентов к памяти. Эти конфликты, с которыми связаны простои отдельных аба-

нентов в ожидании освобождения памяти, снижают производительность ВС. Для уменьшения частоты конфликтов используются быстродействующие буферные памяти в процессорах и каналах. Этой же цели служит расслоение памяти, заключающееся в том, что ячейки (соответствующие ширине выборки) со смежными адресами находятся в разных модулях памяти. Это позволяет повысить эффективное быстродействие памяти за счет высокой степени совмещения циклов обращения к разным модулям памяти (циклы обращения к памяти запускаются в ее модулях со сдвигом на один машинный такт).

14.4. ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ. ОРГАНИЗАЦИЯ ВИРТУАЛЬНОЙ ПАМЯТИ

В мультипрограммных системах размещение всех исполняемых целевых программ полностью в ОП во многих случаях невыполнимо: программы часто имеют большую длину, а объемы существующих ОП ограничены. Однако нет принципиальной необходимости в том, чтобы вся целевая программа находилась в ОП, так как в любой момент времени работа программы концентрируется на определенных сравнительно небольших участках. Таким образом, в ОП следует хранить только используемые в данный период части программ, а неиспользуемые части могут располагаться в ВЗУ. Программируя свою программу, пользователь не знает, в комбинации с какими программами будет выполняться его программа, какое место в памяти отведет ей операционная система.

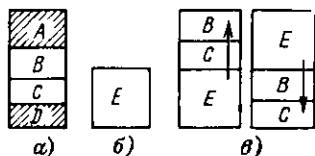
При подготовке целевых программ используются условные адреса. Позднее в процессе выполнения программы операционная система выделяет активным частям программы место в памяти, и условные адреса переводятся в исполнительные. Эта процедура получила название динамического распределения памяти.

Осуществление динамического распределения чисто программным путем привело бы к значительным потерям машинного времени. Целесообразнее пользоваться для этой цели аппаратными средствами.

Один из способов динамического распределения памяти основан на использовании базовых регистров. Такой способ применялся в ЭВМ «Урал-14». Операционная система каждой целевой программе ставит в соответствие свой базовый адрес. Базовые адреса обрабатываемых программ находятся в общих регистрах. При выполнении программы реальный или физический адрес образуется суммированием базового и относительного адресов. При динамическом распределении памяти с по-

Рис. 14.5. Пример распределения памяти:

a — распределение памяти до ввода новой программы; *б* — вводимая программа; *в* — варианты распределения памяти после ввода новой программы



мощью базовых регистров программа (или по крайней мере та часть ее, адреса которой преобразуются с помощью одного и того же базового адреса) должна располагаться в последовательных ячейках и вводиться в ОП целиком, хотя в ближайшем цикле активности может потребоваться лишь небольшой фрагмент программы.

При рассматриваемом способе динамического распределения памяти свободная память может состоять из несвязанных областей (*фрагментация памяти*) и для ввода нужной программы может понадобиться сдвиг содержимого памяти. На рис. 14.5, *a* показано распределение памяти между программами *A*, *B*, *C*, *D*, из которых две (*A* и *D*) являются в данный момент наименее активными и, следовательно, могут рассматриваться как кандидаты на удаление во внешнюю память. Если вновь вводимая программа *E* (рис. 14.5, *б*) больше, чем любая из программ *A* и *D*, то для ее размещения в памяти необходимо, как показано на рис. 14.5, *в*, сдвигать программы *B* и *C*. Это перемещение связано с потерей времени. Более того, в ряде систем подобное перемещение требует выполнения заново операции редактирования связей в программе и новой загрузки программы.

Отмеченные недостатки в распределении памяти отсутствуют в виртуальной памяти со страничной организацией.

Виртуальная память есть способ организации памяти мультипрограммной вычислительной системы, при котором достигается гибкое динамическое распределение памяти, устраняется ее фрагментация и создаются значительные удобства для работы программистов. Это удастся достигнуть без заметного снижения производительности машины ценой усложнения аппаратуры и операционной системы и процессов их функционирования.

Принцип виртуальной памяти предполагает, что пользователь при подготовке своей программы имеет дело не с физической ОП, действительно работающей в составе вычислительной установки и имеющей некоторую фиксированную емкость, а с виртуальной (т. е. кажущейся) одноуровневой памятью, объем которой равен всему адресному пространству, определяемому размером адресных полей в форматах команд и ба-

зовых регистров. В ЕС ЭВМ это пространство составляет 16 777 216 байт.

Пользователь имеет в своем распоряжении все адресное пространство системы независимо от объема ее физической памяти и объемов памяти, необходимых для других программ, участвующих в мультипрограммной обработке.

На всех этапах подготовки программ, включая загрузку в оперативную память, программа представляется в *виртуальных адресах*, и лишь при самом исполнении машинной команды производится преобразование виртуальных адресов в реальные адреса действующей памяти (в так называемые *физические адреса*).

Преобразование виртуальных адресов в физические упрощается и устраняется фрагментация памяти, если физическую и виртуальную память разбить на блоки, называемые в этом случае *страницами*, содержащие одно и то же число байт. Страницам виртуальной и физической памяти присваивают номера, называемые номерами соответственно виртуальных и физических страниц. Каждая физическая страница способна хранить одну из виртуальных страниц. Порядок расположения (нумерация) байт в виртуальной и физической страницах сохраняется одним и тем же.

В мультипрограммной системе страничная организация памяти дает определенные преимущества. Когда новая программа загружается в ОП, она может быть направлена в любые свободные в данный момент физические страницы независимо от того, расположены они подряд или нет. Не требуется перемещения информации в остальной части памяти. Страничная организация позволяет сократить объем передачи информации между внешней памятью и ОП, так как страница программы не должна загружаться до тех пор, пока она действительно не понадобится. Сначала в ОП загружается начальная страница программы, и ей передается управление. Если по ходу работы делается попытка выборки слов из другой страницы, то производится автоматическое обращение к операционной системе, которая осуществляет загрузку требуемой страницы.

На рис. 14.6 показано соответствие между виртуальной и физической памятью, устанавливаемое страничной таблицей, причем видно, что физические страницы могут содержаться в текущий момент времени как в оперативной, так и во внешней памяти. Страничная таблица формируется операционной системой в процессе распределения памяти и перерабатывается ею каждый раз, когда в распределении памяти производятся изменения. Процедура обращения к памяти состоит в том, что номер виртуальной страницы извлекается из адреса и исполь-

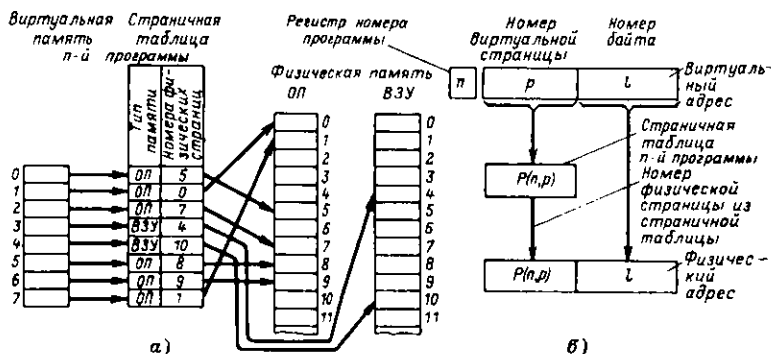


Рис. 14.6. Страничная организация памяти:

а – соответствие между виртуальными и физическими страницами; б – перевод виртуального адреса в физический

зуется для входа в страничную таблицу, которая указывает номер соответствующей физической страницы. Этот номер вместе с номером байта, взятым непосредственно из виртуального адреса, представляет собой физический адрес, по которому происходит обращение к ОП.

Если страничная таблица указывает на размещение требуемой информации во внешней памяти, то обращение к ОП не может состояться немедленно: операционная система должна организовать передачу из внешней памяти в ОП нужной страницы.

Для каждой из программ, обрабатываемых в мультипрограммном режиме, организуется своя виртуальная память и создается своя страничная таблица, при этом все программы делят между собой одну общую физическую память.

Страничные таблицы программ хранятся в ОП, и обращение к нужной строке активной страничной таблицы в ОП происходит по адресу, который определяется номером активной программы и номером виртуальной страницы (рис. 14.6).

Для ускорения преобразования адресов используется небольшая сверхоперативная память, куда передается из ОП страничная таблица активной программы. В другом варианте в сверхоперативной памяти могут находиться сведения о номерах виртуальных и соответствующих физических страниц для нескольких недавно использовавшихся страниц, в том числе принадлежащих разным программам. В этом варианте сверхоперативная память, используемая при преобразовании адресов, строится как ассоциативная (см. гл. 4) с обращением не

адресным, а по содержанию хранимой в ячейке информации — в данном случае по хранимому в ячейке номеру виртуальной страницы.

Сегментно-страничная организация памяти. До сих пор предполагалось, что виртуальная память, которой располагает программист, представляет собой непрерывный массив с единой нумерацией байт. Однако программа обычно состоит из нескольких массивов — подпрограмм, одной или нескольких секций данных. Так как заранее длины этих массивов неизвестны, то удобно, чтобы при программировании каждый массив имел свою собственную нумерацию байт, начинающуюся с нуля и продолжающуюся в возрастающем порядке. Желательно также, чтобы составленная таким образом программа могла работать при динамическом распределении памяти, не требуя от программиста усилий по объединению различных ее частей в единый массив. Эта задача решается в некоторых вычислительных системах путем использования особого метода преобразования виртуальных адресов в физические, называемого сегментно-страничной организацией памяти.

Виртуальная память каждой программы делится на части, именуемые *сегментами*, с независимой адресацией байт внутри каждой части. К виртуальному адресу следует добавить дополнительные разряды левее номера страницы; эти разряды определяют номер сегмента.

Возникает определенная иерархия в организации программ, состоящая из четырех ступеней: 1) программа, 2) сегмент, 3) страница, 4) байт. Этой иерархии программ соответствует иерархия таблиц, служащих для перевода виртуальных адресов в физические. Программная таблица для каждой программы, загруженной в систему, указывает начальный адрес соответствующей сегментной таблицы. Сегментная таблица перечисляет сегменты данной программы с указанием начального адреса страничной таблицы, относящейся к данному сегменту. Страничная таблица определяет расположение каждой из страниц сегмента в памяти. Страницы сегмента могут располагаться не подряд, часть страниц данного сегмента может находиться в оперативной памяти, остальные — во внешней.

Рассмотрим с некоторыми упрощениями организацию сегментно-страничной виртуальной памяти, реализованной в машинах ЕС ЭВМ II очереди [72].

В этой системе работа с виртуальной памятью возможна только в режиме расширенного управления (ССП [12] = 1).

Сегмент представляет собой блок последовательных адресов размером 64 Кбайт или 1 Мбайт, размер страницы 2 или 4 Кбайт. Начальные адреса сегментов и страниц кратны их

размерам. Размеры сегментов и страниц виртуальной памяти активной в данный момент программы задаются значениями соответствующих разрядов управляющего регистра 0.

Виртуальный адрес (как и физический) имеет длину 24 разряда, причем поле номера сегмента занимает 8 или 4 старших разряда соответственно для сегментов размером 64 Кбайт и 1 Мбайт, поле номера байта занимает 11 или 12 младших разрядов для страниц размером 2048 и 4096 байт. Промежуточные разряды адреса занимает поле номера страниц, которое может иметь 4, 5, 8 или 9 разрядов в зависимости от размеров сегмента и страниц.

Сегментные и страничные таблицы находятся в ОП, а в программной таблице нет необходимости, так как для каждой активной в данный момент программы управляющий регистр 1 хранит начальный адрес и длину соответствующей сегментной таблицы. Хранит он также номер программы.

Процесс преобразования адресов представлен на рис. 14.7. В общем случае преобразование адреса происходит в два этапа и требует двух дополнительных обращений к ОП (рис. 14.7, а).

Первый этап. Начальный адрес сегментной таблицы, установленный в управляющем регистре 1, суммируется с номером сегмента из виртуального адреса. В результате образуется адрес, по которому из ОП считывается строка сегментной таблицы, содержащая адрес начала и длину страничной таблицы для данного сегмента.

Второй этап. Полученный адрес начала страничной таблицы суммируется с номером страницы из виртуального адреса, при этом образуется адрес, по которому из ОП считывается строка страничной таблицы, содержащая номер искомой физической страницы. Если эта страница оказывается в ОП, то в старшие разряды регистра физического адреса передается ее номер, а в младшие заносится номер байта из регистра виртуального адреса. Формирование физического адреса на этом завершается.

Если нужная физическая страница оказывается во внешней памяти, то происходит прерывание по страничному сбою (см. § 14.5). Операционная система инициирует передачу этой страницы из внешней памяти в ОП (при этом меняется номер физической страницы) и корректирует соответствующим образом страничную таблицу, находящуюся в оперативной памяти. В старшие разряды регистра физического адреса передается новый номер физической страницы, а в младшие — номер байта. Физический адрес сформирован.

Выполняется запрошенное программой обращение к ОП. Одновременно информация о текущей странице (номера про-

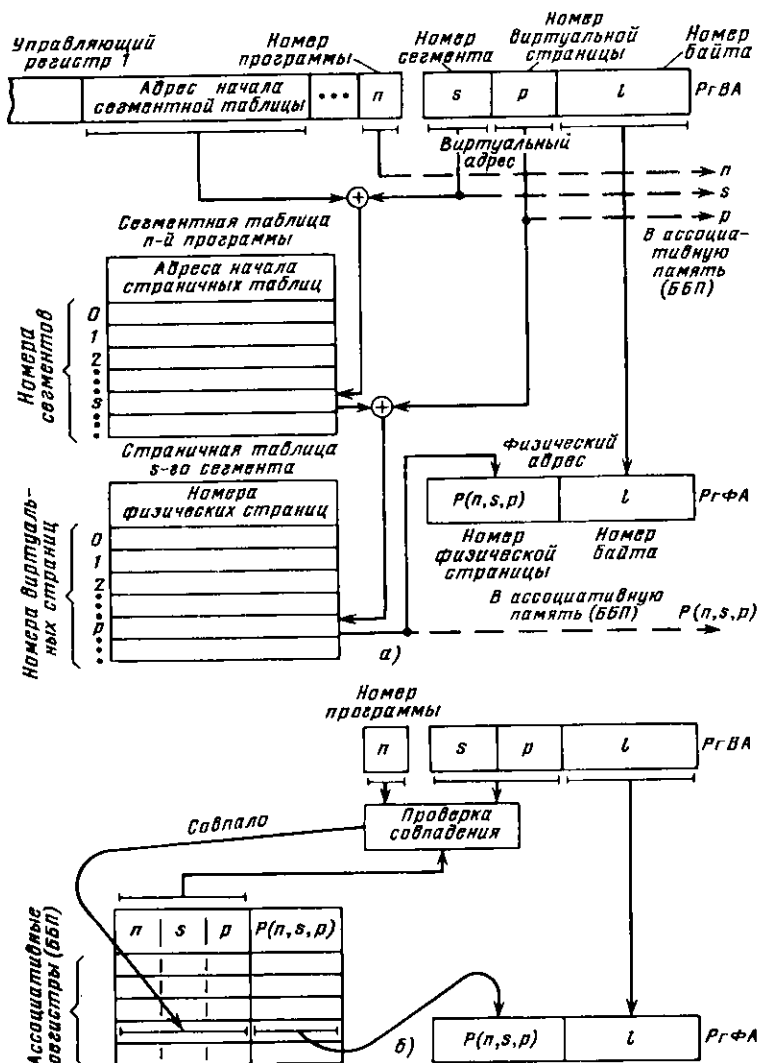


Рис. 14.7. Преобразование адресов при сегментно-страничной организации памяти:

а — с использованием таблиц в ОП; б — с использованием ассоциативной памяти (блока быстрой переадресации)

граммы, сегмента, виртуальной и соответствующей физической страницы) помещается в сверхоперативную ассоциативную память [или в блок быстрой переадресации (ББП)] небольшой емкости. Ассоциативная память (ББП) хранит указанные данные для небольшого числа недавно использовавшихся страниц. Так, в ЭВМ ЕС-1045 ББП представляет собой память на интегральных микросхемах емкостью 128 слов с временем выборки 25–30 нс [102]. В менее крупной ЭВМ ЕС-1035 объем ББП всего 8 слов. Блок быстрой переадресации может обслуживать одновременно до трех программ, другими словами, до трех виртуальных памяти.

При наличии ассоциативной памяти (ББП) значительно ускоряется процесс преобразования адресов, так как на каждом участке вычислительного процесса обращения к ОП сосредотачиваются на небольшом числе страниц, и поэтому имеется большая вероятность, что текущее обращение произойдет к странице, информация о которой уже имеется в ассоциативной памяти (ББП), а следовательно, возможно быстрое преобразование адресов без дополнительных обращений к ОП.

Преобразование адресов всегда начинается с просмотра ассоциативной памяти (ББП), и если оказывается, что в одной из ее строк (ассоциативном регистре) хранится информация о странице, к которой должно произойти обращение, то из этой строки непосредственно выбирается номер физической страницы и дополнительные обращения к ОП (к сегментной и страничной таблицам) не производятся (рис. 14.7, б).

Если нужной информации нет в ассоциативной памяти (ББП), то делается попытка сократить время преобразования путем исключения одного дополнительного обращения к ОП (первый этап на рис. 14.7, а). Может оказаться, что страница, к которой происходит обращение, принадлежит сегменту предыдущего обращения к ОП. В аппаратуре преобразования адресов сохраняются номер сегмента и адрес начала его страничной таблицы для предыдущего обращения. Если совпадают номера сегментов текущего и предыдущего обращений, первый этап преобразования исключается, используется сохраненный адрес начала сегментной таблицы и выполняется только второй этап преобразования, т. е. производится только одно дополнительное обращение к ОП. Если номера сегментов не совпадут, реализуется полная процедура преобразования адресов, показанная на рис. 14.7, а.

Дополнительные обращения к ОП сопровождаются занесением информации о текущей странице в ассоциативную память (ББП). Если в ассоциативной памяти (ББП) не оказывается свободного регистра (строки), данные о новой странице записы-

ваются на место данных, которые дольше других не использовались в процессе преобразования адресов.

14.5. АЛГОРИТМЫ УПРАВЛЕНИЯ МНОГОУРОВНЕВОЙ ПАМЯТЬЮ

Будем рассматривать двухуровневую память со страничной организацией, состоящую из оперативной (верхний уровень) и внешней (нижний уровень) памяти.

Если при выполнении программы обнаруживается, что страница с нужными данными (операндами, куском программы) отсутствует в памяти верхнего уровня, она передается туда из памяти нижнего уровня. Если при этом в памяти верхнего уровня нет для нее свободного места, то она замещает одну из страниц, находящихся в ОП. При этом, если в замещаемую страницу во время ее пребывания в ОП производилась запись, она должна быть передана в память нижнего уровня (заменит там свою устаревшую копию).

Эти операции передачи информации между уровнями памяти вызывают простой процессора и, следовательно, потери производительности вычислительной системы. Поэтому следует стремиться уменьшить число таких операций в процессе выполнения программы. Очевидно, что число этих операций обмена информацией зависит от того, какая информация отсылается из памяти верхнего уровня, так как к этой информации возможно обращение в процессе дальнейшего выполнения программы.

Приведем формализованную модель процесса обмена информацией между верхним и нижним уровнями памяти [16].

Пусть программа вместе с исходными данными состоит из k страниц, которым присвоены номера $1, 2, \dots, k$. Тогда программу можно рассматривать как множество страниц

$$Q = \{1, 2, \dots, k\}.$$

Все страницы программы постоянно хранятся в памяти нижнего уровня, а кроме того, r из них могут находиться в памяти верхнего уровня (оперативной памяти), при этом

$$1 < r < k.$$

Выполнение программы порождает последовательность обращений к страницам памяти. Рассматриваем эту последовательность как реализацию некоторого случайного процесса

$$q_0, q_1, q_2, \dots, q_n$$

где q_i — случайная дискретная величина, принимающая в мо-

мент времени t значение одного из номеров страниц программы ($q_i \in Q$).

Если S_t — совокупность страниц в памяти верхнего уровня в момент t , причем в любой момент в этой памяти присутствует r страниц программы, то изменение состояния памяти верхнего уровня после обращения q_i описывается следующими соотношениями:

$$S_{t+1} = \begin{cases} S_t, & \text{если } q_i \in S_t; \\ S_t - v_i + q_i, & \text{если } q_i \notin S_t. \end{cases}$$

В первом случае обращение производится к странице, которая находится в памяти верхнего уровня, и поэтому состояние этой памяти не меняется.

Во втором случае происходит обращение к странице, отсутствующей в памяти верхнего уровня. Эта ситуация называется *страничным сбоем*, так как программа не может дальше выполняться, пока нужная страница q_i не будет переписана из памяти нижнего уровня в память верхнего уровня, что сопряжено с потерями времени. Поскольку в памяти верхнего уровня нет свободного места, из нее приходится удалять некоторую страницу v_i , с тем чтобы на ее место можно было поместить страницу q_i . Такая процедура называется *процессом замещения страниц*, а правило, по которому при возникновении страничного сбоя выбирается страница $v_i \in S_t$ для удаления из памяти верхнего уровня, — *алгоритмом замещения*.

Для данной программы, порождающей некоторый поток обращений к памяти, существует по крайней мере одна такая последовательность замещений страниц, которая дает минимальное для этой программы число страничных сбоев — *минимально возможную последовательность замещений*. При конструировании алгоритма замещений стремятся приблизить реализуемую этим алгоритмом последовательность замещений к минимальной.

Оптимизация процесса замещений страниц упрощается, если известно, в каком порядке в будущем будут происходить обращения к памяти или, по крайней мере, вероятности обращений в будущем к отдельным страницам программы. Ясно, например, что в первую очередь из памяти верхнего уровня

* Если во время пребывания страницы v_i в памяти верхнего уровня в нее производилась запись, эта страница при замещении должна переписываться в память нижнего уровня.

следует удалить страницу, к которой обращений больше не будет (вероятность обращений в будущем равна 0).

Трудность состоит в том, что, как правило, при выполнении программы отсутствуют информация о потоке обращений или сколько-нибудь достоверные сведения о вероятности обращений к отдельным страницам в будущие моменты времени.

Алгоритмы замещения можно разделить на две группы:

1) *физически нереализуемые*, использующие (реально отсутствующую) информацию о потоке обращений в будущие моменты времени;

2) *физически реализуемые* или *эвристические*, использующие только информацию об обращениях к памяти в прошедшие моменты времени, т. е. только историю процесса.

Хотя алгоритмы первой группы на практике применить нельзя, они играют важную роль в теории алгоритмов замещения, позволяя производить оценки (в том числе экспериментальные), в какой степени характеристики эвристических алгоритмов приближаются к предельно возможным оптимальным.

Физически нереализуемые алгоритмы. Алгоритм Михновского — Шора¹: при каждом замещении страницы из памяти верхнего уровня отсылается в память нижнего уровня страница, очередное обращение к которой произойдет позже, чем к любой другой странице в памяти верхнего уровня.

Справедливо следующее предложение. Число замещений страниц в памяти верхнего уровня (число страничных сбоев) при выполнении замещений по алгоритму Михновского — Шора является минимальным для заданных потока обращений и исходного распределения памяти.

Доказательство справедливости предложения можно найти в [56, 21].

Таким образом, алгоритм Михновского — Шора реализует минимально возможную для данной программы последовательность замещений. Поэтому этот алгоритм называют *МИН-алгоритмом*.

Если условиться, что известна вероятность обращений к отдельным страницам программы, то оптимальным в смысле минимума среднего числа страничных сбоев является *ОПТ-алгоритм*:

¹ Алгоритм и доказательство его оптимальности впервые опубликованы С. Д. Михновским и Н. З. Шором в 1965 г. [56]. В литературе часто этот алгоритм приписывают американскому автору Беледи на основании его статьи, опубликованной в 1966 г.

при каждом замещении страницы из памяти верхнего уровня отсылается страница, вероятность обращения к которой не больше, чем к любой другой странице в этой памяти.

Физически реализуемые (эвристические) алгоритмы замещения. Был предложен ряд алгоритмов этого класса.

Алгоритм случайного замещения (СЗ-алгоритм). При возникновении страничного сбоя из памяти верхнего уровня с равной вероятностью отсылается любая из находящихся там страниц.

НДИ-алгоритм. Из памяти верхнего уровня отсылается страница, наиболее давно использовавшаяся.

Алгоритм «первый пришел — первый ушел» (ПППУ-алгоритм). Отсылается страница, дольше других находившаяся в памяти верхнего уровня.

Алгоритм «последний пришел — первый ушел». Отсылается страница, позже других поступившая в память верхнего уровня.

Следующие два алгоритма обладают определенными свойствами адаптации к потоку обращений к памяти.

Алгоритм «карабкающаяся страница» (КС-алгоритм). Страницы в памяти верхнего уровня образуют последовательность (рис. 14.8)

$$S_t = j_1, j_2, \dots, j_{m-1}, j_m, j_{m+1}, \dots, j_r \quad (j_m \in Q),$$

которая при очередном обращении q_t к памяти изменяется по правилу

$$S_{t+1} = \begin{cases} S_t & \text{при } q_t = j_1; \\ j_1, j_2, \dots, j_m, j_{m-1}, \dots, j_r & \text{при } q_t = j_m, m \neq 1; \\ j_1, j_2, \dots, j_{r-1}, q_t & \text{при } q_t \notin S_t. \end{cases}$$

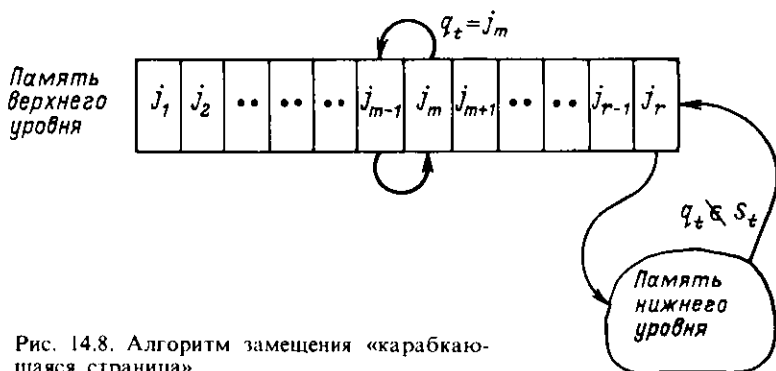


Рис. 14.8. Алгоритм замещения «карабкающаяся страница»

При обращении к странице, присутствующей в памяти верхнего уровня, последняя меняется местами с соседней слева страницей, другими словами, «карабкается» к началу последовательности, подальше от ее конца, куда происходит замещение при страничном сбое.

Алгоритм «рабочий комплект» (РК-алгоритм). Страницы в памяти верхнего уровня, использовавшиеся в течение заданного интервала времени, образуют «рабочий комплект». Страницы из этой памяти, не вошедшие в «рабочий комплект», формируют две очереди кандидатов на замещение: 1) очередь страниц, в которые не вносились изменения, пока они присутствовали в памяти верхнего уровня; 2) очередь страниц, в которые вносились изменения. Замещение производится по правилу: первый пришел из рабочего комплекта — первый ушел из памяти верхнего уровня, при этом сначала подлежат замещению страницы из первой очереди.

Формирование указанных очередей производится с использованием ключа защиты, расширенного дополнительными разрядами *Об* и *Из* (см. рис. 14.2), устанавливаемыми в 1, если соответственно происходит любое обращение к данному блоку (странице) и если в него произведена запись. Описанный алгоритм применен в ЕС ЭВМ.

Модель процесса замещения страниц. Предположим, что последовательность обращений $q_1, q_2, \dots, q_t$ соответствует последовательности независимых случайных дискретных величин, таких, что

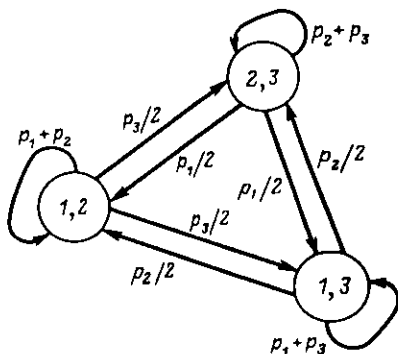
$$P\{q_t = j\} = p_j, \quad 1 \leq j \leq k, \quad \sum_{j=1}^k p_j = 1.$$

Примем за состояние процесса замещения набор (а в некоторых случаях упорядоченную последовательность) страниц, находящихся в памяти верхнего уровня. Тогда для ряда алгоритмов замещения (СЗ, НДИ, ПППУ и некоторых других) процесс изменения состояния верхнего уровня описывается однородной конечной эргодической цепью Маркова, что указывает на существование стационарных вероятностей пребывания процесса в определенных состояниях и, как следствие этого, стационарных вероятностей страничных сбоев.

В качестве критерия эффективности $W_{r,k}$ алгоритма замещения A примем стационарную вероятность страничных сбоев

$$W_{r,k}(A) = \lim_{t \rightarrow \infty} P\{q_t \notin S_t\}.$$

Рис. 14.9. Граф состояний для алгоритма случайного замещения



Можно для ряда алгоритмов замещения найти зависимость $W_{r,k}$ от $p_1, p_2, \dots, p_k$ и сравнить алгоритмы между собой и также с физически нереализуемым ОПТ-алгоритмом. Определить $W_{r,k}$ для ряда алгоритмов можно, используя метод, основанный на однородных эргодических цепях Маркова [1, 16].

Покажем применение этого метода на примере алгоритма случайного замещения. Для простоты рассмотрим случай

$$k = 3, r = 2, Q = \{1, 2, 3\}, p_1 + p_2 + p_3 = 1.$$

Состояниями марковской цепи, описывающей рассматриваемый процесс, будут все возможные состояния памяти верхнего уровня, т. е. в данном случае состояния (1, 2), (1, 3), (2, 3).

При СЗ-алгоритме порядок страниц в памяти не играет роли (в отличие, например, от алгоритмов НДИ и ПППУ).

На рис. 14.9 приведен граф состояний. Вероятности переходов из состояния в состояние указаны на дугах графа.

Напишем уравнения для стационарных вероятностей $\Pi_{12}, \Pi_{23}, \Pi_{13}$ пребывания марковского процесса соответственно в состояниях (1, 2), (2, 3), (1, 3):

$$\Pi_{12} = (p_1 + p_2) \Pi_{12} + p_1 \Pi_{23}/2 + p_2 \Pi_{13}/2;$$

$$\Pi_{23} = (p_2 + p_3) \Pi_{23} + p_3 \Pi_{12}/2 + p_2 \Pi_{13}/2;$$

$$\Pi_{13} = (p_1 + p_3) \Pi_{13} + p_3 \Pi_{12}/2 + p_1 \Pi_{23}/2.$$

С учетом условия нормировки $p_1 + p_2 + p_3 = 1$ получаем следующее решение системы уравнений:

$$\Pi_{12} = p_1 p_2 (p_1 p_2 + p_1 p_3 + p_2 p_3); \quad \Pi_{23} = p_2 p_3 / (p_1 p_2 + p_2 p_3 + p_1 p_3); \quad \Pi_{13} = p_1 p_3 / (p_1 p_2 + p_2 p_3 + p_1 p_3).$$

Стационарная вероятность страничного сбоя

$$W_{2,3}(\text{СЗ}) = \Pi_{12} p_3 + \Pi_{23} p_1 + \Pi_{13} p_2 = 3 \frac{p_1 p_2 p_3}{p_1 p_2 + p_2 p_3 + p_1 p_3}.$$

Аналогичным путем можно получить выражение для общего случая

$$W_{r,k}(C3) = (r+1) \frac{\sum_{[j_1, j_2, \dots, j_{r+1}]} p_{j_1} p_{j_2} \dots p_{j_{r+1}}}{\sum_{[j_1, j_2, \dots, j_r]} p_{j_1} p_{j_2} \dots p_{j_r}},$$

где $[j_1, j_2, \dots, j_r]$ обозначает все возможные сочетания из множества $\{1, 2, \dots, k\}$ по r элементов.

Можно получить выражения и для некоторых других алгоритмов. Примем такую нумерацию страниц, что

$$p_1 \geq p_2 \geq \dots \geq p_k.$$

Тогда для ОПТ-алгоритма имеем

$$W_{r,k}(\text{ОПТ}) = \sum_{i=1}^k p_i - \frac{\sum_{i=1}^k p_i^2}{\sum_{i=1}^k p_i}.$$

Имеет место следующее соотношение:

$$W_{r,k}(\text{ПППУ}) = W_{r,k}(C3).$$

Для НДИ- и КС-алгоритмов выражения $W_{r,k} = f(p_1, \dots, p_k)$ не приводим из-за их громоздкости. Анализ показывает, что

$$W_{r,k}(\text{НДИ}) \leq W_{r,k}(\text{ПППУ});$$

$$W_{r,k}(\text{КС}) \leq W_{r,k}(\text{НДИ}).$$

Глава пятнадцатая

ПРИНЦИПЫ ОРГАНИЗАЦИИ МНОГОПРОЦЕССОРНЫХ И МНОГОМАШИННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

15.1. ПОНЯТИЕ О МНОГОМАШИННЫХ И МНОГОПРОЦЕССОРНЫХ ВС

Вычислительная техника в своем развитии по пути повышения быстродействия ЭВМ приблизилась к физическим пределам. Время переключения электронных схем достигло долей наносекунды, а скорость распространения сигналов в линиях,

связывающих элементы и узлы машины, ограничена значением 30 см/нс (скоростью света). Поэтому дальнейшее уменьшение времени переключения электронных схем не позволит существенно повысить производительность ЭВМ. В этих условиях требования практики (сложные физико-технические расчеты, многомерные экономико-математические модели и другие задачи) по дальнейшему повышению быстродействия ЭВМ могут быть удовлетворены только путем распространения принципа параллелизма на сами устройства обработки информации и создания *многомашинных* и *многопроцессорных* (мультипроцессорных) *вычислительных систем*. Такие системы позволяют производить распараллеливание во времени выполнения программы или параллельное выполнение нескольких программ.

В настоящее время исключительно важное значение приобрела проблема обеспечения высокой надежности и готовности вычислительных систем, работающих в составе различных АСУ и АСУ ТП, особенно при работе в режиме реального времени. Эта проблема решается на основе использования принципа избыточности, который ориентирует также на построение *многомашинных* или *многопроцессорных* систем (комплексов). Появление дешевых и небольших по размерам микропроцессоров и микро-ЭВМ облегчило построение и расширило область применения многопроцессорных и многомашинных ВС разного назначения.

Различие понятий *многомашинной* и *многопроцессорной* ВС поясняет рис. 15.1. Многомашинная ВС (ММС) содержит несколько ЭВМ, каждая из которых имеет свою ОП и работает

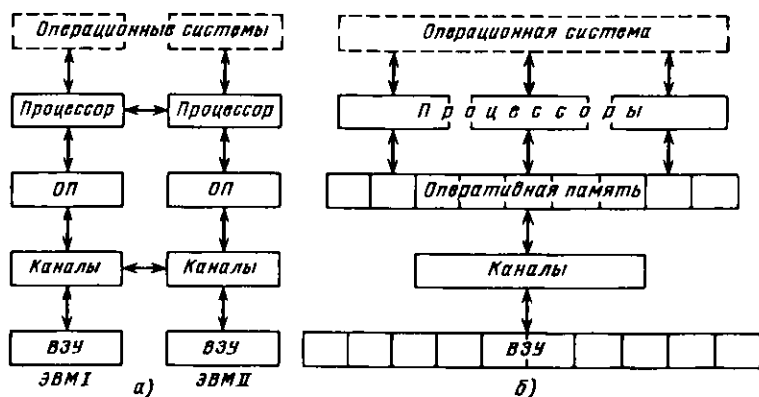


Рис. 15.1. К сопоставлению многомашинной (а) и многопроцессорной (б) систем

под управлением своей операционной системы, а также средства обмена информацией между машинами. Реализация обмена информацией происходит в конечном счете путем взаимодействия операционных систем машин между собой. Это ухудшает динамические характеристики процессов межмашинного обмена данными. Применение многомашинных систем позволяет повысить надежность вычислительных комплексов. При отказе в одной машине обработку данных может продолжать другая машина комплекса. Однако можно заметить, что при этом оборудование комплекса недостаточно эффективно используется для этой цели. Достаточно в системе, изображенной на рис. 15.1, а, в каждой ЭВМ выйти из строя по одному устройству (даже разных типов), как вся система становится неработоспособной.

Этих недостатков лишены многопроцессорные системы (МПС). В таких системах (рис. 15.1, б) процессоры обретают статус рядовых агрегатов вычислительной системы, которые подобно другим агрегатам, таким, как модули памяти, каналы, периферийные устройства, включаются в состав системы в нужном количестве.

Вычислительная система называется многопроцессорной, если она содержит несколько процессоров, работающих с общей ОП (*общее поле оперативной памяти*) и управляется одной общей операционной системой. Часто в МПС организуется общее поле внешней памяти.

Под общим полем понимается равнодоступность устройств. Так, общее поле памяти означает, что все модули ОП доступны всем процессорам и каналам ввода-вывода (или всем периферийным устройствам в случае наличия общего интерфейса); общее поле ВЗУ означает, что образующие его устройства доступны любому процессору и каналу.

В МПС по сравнению с ММС достигается более быстрый обмен информацией между процессорами и поэтому может быть получена более высокая производительность, более быстрая реакция на ситуации, возникающие внутри системы и в ее внешней среде, и более высокие надежность и живучесть, так как система сохраняет работоспособность, пока работоспособны хотя бы по одному модулю каждого типа устройств.

Многопроцессорные системы представляют собой основной путь построения ВС сверхвысокой производительности. При создании таких ВС возникает много сложных проблем, к которым в первую очередь следует отнести *распараллеливание вычислительного процесса (программ)* для эффективной загрузки процессоров системы, преодоление *конфликтов* при попытках нескольких процессоров использовать один и тот же ресурс

системы (например, некоторый модуль памяти) и уменьшение влияния конфликтов на производительность системы, осуществление быстродействующих экономичных по аппаратурным затратам межмодульных связей. Указанные вопросы необходимо учитывать при выборе структуры МПС.

На основе многопроцессорности и модульного принципа построения других устройств системы возможно создание *отказоустойчивых систем*, или, другими словами, *систем повышенной живучести*.

Однако построение многомашинных систем из серийно выпускаемых ЭВМ с их стандартными операционными системами значительно проще, чем построение МПС, требующих преодоления определенных трудностей, возникающих при реализации общего поля памяти, и, главное, трудоемкой разработки специальной операционной системы.

Многомашинные и многопроцессорные системы могут быть *однородными и неоднородными*. Однородные системы содержат однотипные ЭВМ или процессоры. Неоднородные ММС состоят из ЭВМ различного типа, а в неоднородных МПС используются различные специализированные процессоры, например процессоры для операций с плавающей запятой, для обработки десятичных чисел, процессор, реализующий функции операционной системы, процессор для матричных задач и др.

Многопроцессорные системы и ММС могут иметь *одноуровневую или иерархическую (многоуровневую) структуру*. В первом случае процессоры (машины) системы образуют один общий уровень обработки данных (рис. 15.1), а во втором (рис. 15.2) система содержит отдельные машины (процессоры) для выполнения различных уровней обработки информации. Обычно менее мощная машина (*машина-сателлит*) берет на себя ввод информации с различных терминалов и ее предварительную обработку, разгружая от этих сравнительно простых процедур основную, более мощную ЭВМ, чем достигается увеличение общей производительности (пропускной способности) комплекса. В качестве машин-сателлитов используют малые или микро-ЭВМ.

Важной структурной особенностью рассматриваемых

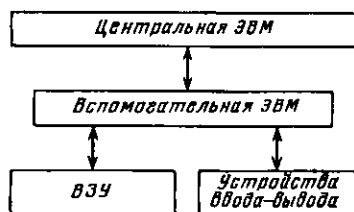


Рис. 15.2. Многомашинный вычислительный комплекс иерархической структуры

ВС является *способ организации связей между устройствами (модулями) системы*. Он непосредственно влияет на быстроту обмена информацией между модулями, а следовательно, на производительность системы, быстроту ее реакции на поступающие запросы, приспособленность к изменениям конфигурации и, наконец, размеры аппаратных затрат на осуществление межмодульных связей. В частности, от организации межмодульных связей зависят частота возникновения конфликтов при обращении процессоров к одним и тем же ресурсам (в первую очередь модулям памяти) и потери производительности из-за конфликтов.

Используются следующие способы организации межмодульных (межустройственных) связей:

регулярные связи между модулями;

многоуровневые связи, соответствующие иерархии интерфейсов ЭВМ;

многовходовые модули (в частности, модули памяти);

коммутатор межмодульных связей;

общая шина.

Принципы организации МПС и ММС существенно отличаются в зависимости от их назначения. Поэтому целесообразно различать:

1) ВС, ориентированные в первую очередь на достижение сверхвысокой производительности;

2) ВС, ориентированные в первую очередь на повышение надежности и живучести.

15.2. ТИПЫ СТРУКТУР МНОГОПРОЦЕССОРНЫХ ВС, ОРИЕНТИРОВАННЫХ НА ДОСТИЖЕНИЕ СВЕРХВЫСОКОЙ ПРОИЗВОДИТЕЛЬНОСТИ

Многопроцессорные системы, ориентированные на достижение сверхбольших скоростей работы (сотни миллионов операций в секунду), содержат несколько десятков или сотен сравнительно простых (элементарных) процессоров с упрощенными блоками управления. Отказ от универсальности применения таких ВС и специализация их на определенном, правда, достаточно широком круге задач, допускающих эффективное распараллеливание вычислений, позволяют строить их с регулярной структурой связей между процессорами.

Удачной, на наш взгляд, является классификация Флина [117], облегчающая понимание особенностей структур и функционирования МПС, в первую очередь систем с регулярными связями. Классификация строится по признаку одинарности или множественности потоков команд и данных.

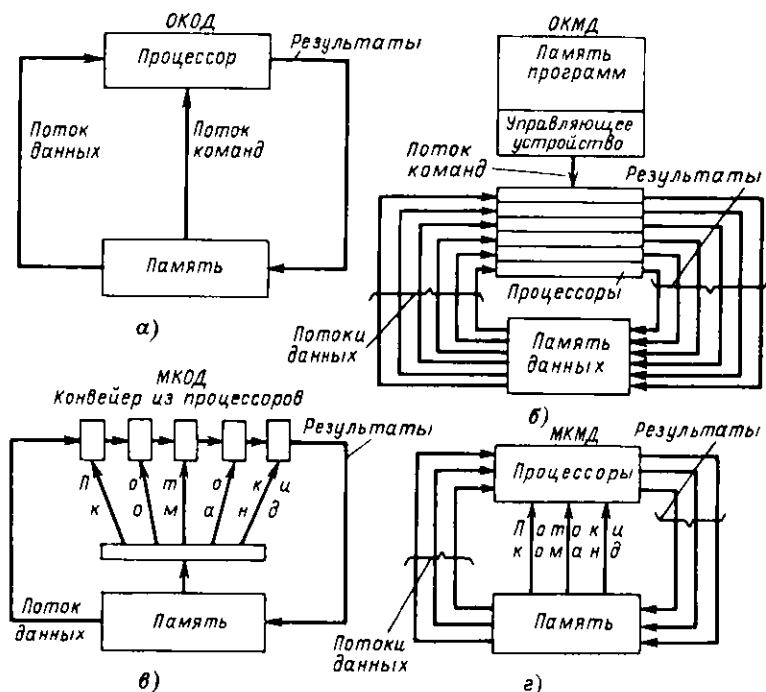


Рис. 15.3. К классификации многопроцессорных ВС по одинарности и множественности потоков команд и данных:

а — однопроцессорная ЭВМ; б — ВС с общим потоком команд; в — конвейерная ЭВМ; г — общий случай

Однопроцессорная ЭВМ. Структура обыкновенной однопроцессорной ЭВМ (рис. 15.3, а) содержит одинарный поток команд и одинарный поток данных (структура ОКОД).

Матричная МПЭС — структура типа ОКМД (рис. 15.3, б). Система содержит некоторое число одинаковых сравнительно простых быстродействующих процессоров, соединенных друг с другом и с памятью данных регулярным образом так, что образуется сетка (матрица), в узлах которой размещаются процессоры.

В системе имеется несколько потоков данных и один общий поток команд, другими словами, все процессоры выполняют одновременно одну и ту же команду (допускается пропуск выполнения команды в отдельных процессорах), но над разными операндами, доставляемыми процессорам из памяти несколь-

кими потоками данных. Подобные МПС часто называют системами с общим потоком команд.

В МПС с общим потоком команд возникает сложная задача распараллеливания алгоритмов решаемых задач для обеспечения загрузки процессоров. В ряде случаев эти вопросы лучше решаются в конвейерной системе.

Конвейерная МПС — структура типа МКОД (рис. 15.3, в). Система имеет регулярную структуру в виде цепочки последовательно соединенных процессоров, так что информация на выходе одного процессора является входной информацией для другого процессора. Процессоры образуют *процессорный конвейер (трубопровод)*. На вход конвейера *одинарный поток данных* доставляет операнды из памяти. Каждый процессор обрабатывает соответствующую часть задачи, передавая результаты соседнему процессору, который использует их в качестве исходных данных. Таким образом, решение задач для некоторых исходных данных развергивается последовательно в конвейерной цепочке. Это обеспечивается подведением к каждому процессору своего потока команд, т. е. имеется *множественный поток команд*.

Если «трубопровод наполнен», выходной процессор выдает результаты для последовательности входных данных через очень короткие интервалы времени, хотя действительное время прохождения задачи через конвейер может быть существенно большим.

Конвейерная обработка была описана в гл. 9, где говорилось о конвейере, образованном блоками процессора, реализующими отдельные этапы рабочего цикла выполнения команды под управлением множественного потока управляющих сигналов микроопераций. Внутрипроцессорная конвейерная обработка в настоящее время широко используется. Примером являются входящие в состав ЭВМ ЕС специализированные конвейерные процессоры для матричных и векторных операций (по терминологии ЕС ЭВМ называемые «матричными процессорами»). Здесь рассматривается конвейер из более сложных модулей — процессоров, управляемых каждый своим потоком команд.

Общий случай МПС — структура типа МКМД. На рис. 15.3, г представлен общий случай структуры МПС, в которой *несколько потоков данных и несколько потоков команд*.

В настоящее время конкурируют между собой две структуры построения крупных многопроцессорных систем высокой производительности: а) с общим потоком команд, б) конвейерная.

К МПС с общим потоком команд относится разработанная под руководством академика АН Грузинской ССР И. В. Прангишвили высокопроизводительная система ПС-2000, структура которой представлена на рис. 15.4 [70].

Параллельный процессор (ППр) системы содержит до 64 процессорных элементов (ПЭ) и имеет суммарную производительность до 200 млн. коротких операций. В состав системы входят также подсистемы внешней памяти ПСВП и мониторная подсистема МмС на основе малой ЭВМ СМ-2М. Общее управление всей системой производит МмС, которая загружает в УУ параллельного процессора микропрограммы, программы и константы, управляет обменом информацией между ППр и ПСВП, осуществляет связь с устройствами ввода-вывода.

Процессорные элементы выполняют операции над 24-разрядными словами. В состав каждого ПЭ входят АЛУ, память емкостью 4 или 16 Кслов с циклом обращения соответственно 640 и 960 нс, сверхоперативная память на 16 слов, локальное УУ.

Соседние ПЭ соединены друг с другом регулярными связями, допускающими параллельную передачу слов. Кроме того, имеется шина («магистральный канал») для последовательной передачи слов, связывающая все ПЭ и УУ. По этой шине слово может передавать одновременно только один ПЭ или УУ, а принимать сразу все адресуемые ПЭ.

Как видно из структурной схемы, система ПС-2000 построена на основе иерархической децентрализации и распределения обработки, управления, хранения и обмена данными между ее отдельными модулями.

Эффективность использования матричных (параллельных) ВС зависит от возможностей реализации распараллеливания программ («параллельное программирование»), что во многих случаях связано с большими трудностями. В связи с этим возникла идея ЭВМ с управлением не потоками команд, как в обычных ЭВМ, а потоками данных, что позволяет достигнуть высокого уровня децентрализации и распараллеливания управления вычислительным процессом.

В системах с управлением потоками данных предполагается наличие большого числа специализированных операционных блоков для определенных видов операций (сложения, умножения и т. п., отдельных для разных типов данных). Данные снабжаются указателями типа данных (*тегами*), на основании которых по мере готовности данных (операндов) к обработке (а не в порядке последовательности команд в программе) они загружаются в соответствующие освободившиеся операционные блоки. Высокий уровень распараллеливания вычисли-

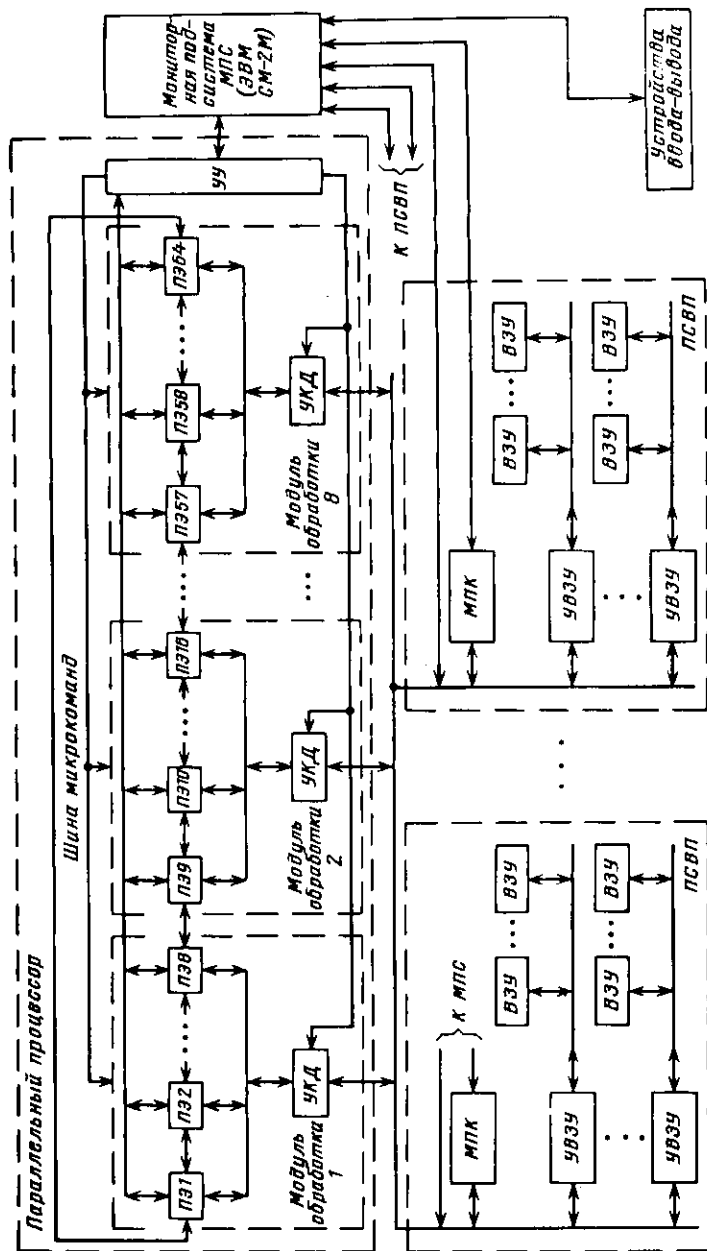


Рис. 15.4. Структура МПС с общим потоком команд ПС-2000:

УКД — управление каналами данных; МПК — микропрограммируемый контроллер внешней памяти

тельного процесса позволяет достигнуть высокой производительности системы.

15.3. МЕТОДЫ И СРЕДСТВА ОРГАНИЗАЦИИ МНОГОМАШИННЫХ И МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ НА ОСНОВЕ ЭВМ ОБЩЕГО НАЗНАЧЕНИЯ (ЕС ЭВМ)

ЭВМ Единой системы создавались в основном как однопроцессорные универсальные по назначению машины, содержащие большой набор периферийных устройств. В этих ЭВМ используются разнообразные интерфейсы для связи между различными группами устройств (см. § 11.3), что определяет специфику построения многомашинных и многопроцессорных комплексов в ЕС ЭВМ.

Многомашинные комплексы с ЭВМ ЕС создают в первую очередь для повышения надежности и достоверности функционирования при обработке данных в реальном масштабе времени в ответственных АСУ, не допускающих перерывов в обработке информации. Для этой цели часто используют комплексы, образованные из двух одинаковых ЭВМ, снабженных средствами автоматического контроля правильности функционирования (см. гл. 12) и средствами обмена информацией между машинами.

Основными режимами работы двухмашинных комплексов являются: дуплексный, параллельная работа (распределение задач между машинами); автономная работа машин (например, с выводом одной машины на профилактические испытания или ремонт), со второй машиной в горячем или холодном резерве. Необходимо иметь средства задания (в том числе осуществляемого программно) режима работы комплекса.

В дуплексном режиме обе машины выполняют одни и те же операции над одной и той же информацией. Работа обеих машин синхронизируется электронными часами одной из машин (ведущей). В памяти обеих машин в каждый момент времени находится одна и та же информация. В определенные моменты, например после выполнения каждой команды или некоторых частей программы, результаты вычислений в обеих машинах сравниваются, при их совпадении работа машин продолжается, и ведущая машина выдает результаты во внешнюю среду, например управляющие воздействия на объект управления. Выходы другой машины (ведомой) при этом блокированы.

При обнаружении несовпадения в результатах обработки автоматически определяется неисправная ЭВМ (по сигналу си-

системы автоматического контроля или при помощи тестов), которая выводится на ремонт, а исправная ЭВМ продолжает работу под контролем встроенной в ЭВМ системы автоматического контроля, при этом, если она ранее была ведомой, ее выводы разблокируются.

После устранения отказа отремонтированная машина вводится в дуплексный режим в качестве ведомой. Предварительно в ее память переписывается содержимое памяти ведущей машины.

Очевидно, что если средняя продолжительность ремонта существенно меньше среднего интервала времени между возникновениями неисправностей (см. гл. 12), то дуплексный режим дает значительное повышение общей надежности комплекса. Если этого оказывается недостаточно, то в комплекс добавляется третья, резервная ЭВМ.

Двухмашинные комплексы используют также, когда необходимо получить производительность и пропускную способность, большие, чем может обеспечить одна ЭВМ, например чтобы снять пиковые нагрузки по обработке или вводу информации. В таких случаях машины комплекса, обмениваясь друг с другом информацией, выполняют разные программы (задачи) или по одним и тем же программам обрабатывают разные данные (режим параллельной работы машин).

Многомашинные комплексы в ЕС ЭВМ содержат две или более ЭВМ одинаковых или разных моделей, каждая со своей операционной системой, оперативной памятью, каналами и периферийными устройствами, связанных между собой через специальные устройства обмена информацией, которые называют *средствами комплексирования или системными средствами*.

В ЕС ЭВМ связь между машинами может осуществляться несколькими способами (на нескольких уровнях) (рис. 15.5), которые ниже перечисляются в порядке возрастания скорости обмена информационными массивами:

- 1) через интерфейс прямого управления;
- 2) через разделенные блоки управления (РУПУ) периферийными устройствами;
- 3) через двухканальные переключатели;
- 4) через адаптер «канал — канал»;
- 5) через общую оперативную память.

В каждом конкретном многомашинном комплексе не обязательно реализуются все виды межмашинных связей.

Понятие *интерфейс прямого управления* включает в себя систему шин, сигналов и команд прямого управления, предназна-

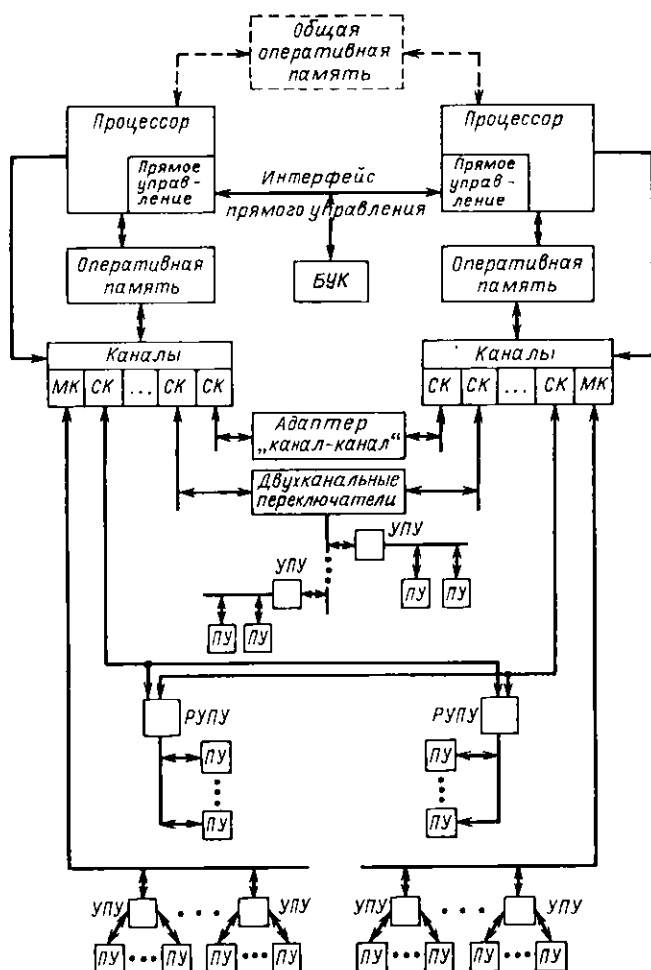


Рис. 15.5. Способы и средства системной организации в ЕС ЭВМ

ченных для выполнения процедуры обмена информацией между процессорами.

В ЕС ЭВМ имеются две команды прямого управления (формата *SI*): «Прямая запись» и «Прямое чтение». При выполнении команды «Прямая запись» процессор данной ЭВМ выставляет на выходной информационной шине прямого управления сигналы, соответствующие коду адресуемого командой байта из своей ОП, и одновременно посылает в другую ЭВМ

сигнал запроса внешнего прерывания. Процессор другой ЭВМ по сигналу прерывания переходит к выполнению команды «Прямое чтение», принимает байт данных со своей входной информационной шины и помещает его в свою ОП по адресу, указанному в этой команде.

Связь через интерфейс прямого управления используют главным образом для передачи информации, управляющей и синхронизирующей работу комплекса.

Связи типа 2—4 используют селекторные и мультиплексные каналы машин.

Разделенные блоки управления обеспечивают доступ обеих машин к некоторым общим периферийным устройствам.

Двухканальные переключатели позволяют подключать блоки управления ВЗУ к двум каналам различных машин и создавать общие поля внешней памяти на дисках и лентах. Обычно при этом часть ВЗУ остается в индивидуальном пользовании отдельных машин.

Адаптеры «канал — канал» (АКК) позволяют производить обмен большими информационными массивами со скоростью, соответствующей пропускной способности используемых каналов; например, АКК ЕС-4061, работая в монопольном режиме, обеспечивает обмен информацией между машинами со скоростью около 1 Мбайт/с. Адаптер «канал — канал» подключается к каналу и управляется им как обычное УПУ, при этом каждая из связанных адаптером ЭВМ по отношению друг к другу является ПУ.

Наибольшую скорость обмена данными обеспечивает *связь между машинами через общую оперативную память (общее поле оперативной памяти)*. В ЕС ЭВМ II очереди реализованы все пять указанных выше способов системной связи.

Повышение надежности и возможность быстрой подмены отказавшей машины достигаются дублированием информации в ОП и общих ВЗУ, формированием пакета запросов и файлов входных данных в общих ВЗУ, использованием АКК для быстрого «выравнивания» содержания ОП машины при вводе отремонтированной ЭВМ в дуплекс.

В индивидуальных ВЗУ хранятся операционная система машины и программы и данные фоновых (не основных) задач, выполняемых в режиме параллельной работы, когда другая ЭВМ обрабатывает основную программу.

Комплекс снабжают блоком управления комплексом (БУК), подключаемым к интерфейсу прямого управления. Блок управления комплексом, беря на себя функции пультов управления отдельных ЭВМ, обеспечивает задание конфигурации комплекса и режимов его работы (ввод ЭВМ в дуплексный режим

и вывод из него, режим параллельной обработки с межмашинным обменом информацией, автономная работа машин, одна ЭВМ в холодном резерве, начальная загрузка и др.), а также индикацию состояний ЭВМ.

Проблемы построения многопроцессорных ВС в ЕС ЭВМ

При построении многопроцессорных ВС в ЕС возникает ряд довольно сложных проблем, связанных с организацией многопроцессорности на основе штатных процессоров и мультипрограммных однопроцессорных ЭВМ универсального назначения, недостаточно приспособленных для этой цели. Основные затруднения связаны с осуществлением общего поля ОП.

Разнесение зон фиксированных ячеек. В машинах ЕС ЭВМ выделена зона фиксированных ячеек — постоянно распределенная область памяти (первые 4 Кбайт), ячейки которой имеют определенное функциональное назначение, а их адреса зафиксированы для определенных процедур, например адреса «старых» и «новых» ССП, адресного слова канала и др. В общем случае при работе процессоров в МПС их зоны фиксированных ячеек содержат неодинаковую информацию. Поэтому при построении общего поля ОП возникает необходимость выделения каждому процессору своей зоны фиксированных ячеек, не пересекающейся с зонами других процессоров, и создания механизма, который обращения данного процессора по адресу из начальной (4 Кбит) области трансформирует в соответствующий адрес выделенной процессору зоны. Этот механизм называется префиксацией.

Механизм префиксации. Каждому процессору МПС присваивается префикс — 12-разрядный код, задающий 12 старших разрядов начального адреса (младшие 12 разрядов начального адреса — нули), выделенных процессору 4 Кбайт в общем поле ОП в качестве его постоянно распределенной области памяти. Префикс определяет значение смещения в общем поле ОП адресов 0—4095 для данного процессора. Процессоры снабжаются программно-загружаемыми регистрами префикса. Префиксация производится блоком управления памятью непосредственно перед обращением и состоит в следующем преобразовании адреса: старшие 12 разрядов адреса заменяются префиксом, если эти разряды равны нулю, заменяются нулями, если эти разряды равны префиксу (обратная префиксация), не изменяются в других случаях. Младшие 12 разрядов адреса не подвергаются изменениям.

«Выравнивание» содержимого собственных специальных памяти процессоров. Процессоры ЕС ЭВМ имеют собственные быстродействующие память ключей защиты (ПКЗ), ассоциативную память страничной таблицы (буфер быстрой переадресации), буферную память. Если один процессор с санкции операционной системы произвел изменение ключа защиты блока (страницы) в своей ПКЗ или строки в страничной таблице в своих соответствующих памяти, то аналогичные изменения должны производиться и в памяти других процессоров. В противном случае могут возникнуть серьезные нарушения в работе МПС. Отметим, что применение в устройстве управления памятью

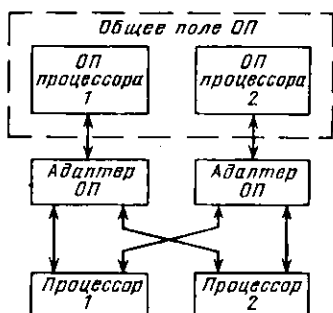


Рис. 15.6. Организация общего поля памяти с помощью многовходовых адаптеров

одной для всего поля ОП памяти ключей защиты оказывается нерациональным из-за недопустимого возрастания частоты конфликтов при обращении к памяти.

Нарушения в работе системы могут возникнуть, если один процессор изменил данные в ОП, а другой, выполняя программу, воспользовался устаревшей копией этих данных в своей буферной памяти.

Поэтому при изменении процессором некоторых данных в ОП копии этих данных, ставшие неактуальными в буферных памяти других процессоров, должны объявляться недействительными. Аналогично если процессор изменил строку страничной таблицы в ОП, то для соответствующей физической строки должна быть аннулирована строка в буферах быстрой переадресации всех процессоров. «Выравнивание» информации в собственных памяти процессоров достигается с помощью соответствующих управляющих сигналов.

Межпроцессорная связь в МПС помимо упомянутых выше команд прямого управления достигается с помощью специальной команды «Сигнал процессору» (формат RS), задающей номер вызываемого процессора (находится в общем регистре, указанном в поле R_3 команды) и приказ (находится в полях B_2 , D_2 команды). С помощью соответствующих приказов вызываемый процессор может быть установлен в состояние *стоп* или *работа*, в него может быть произведена начальная загрузка микропрограмм, могут быть вызваны экстренное внешнее прерывание и некоторые другие процедуры.

Синхронизация часов. В МПС должна быть единая служба астрономического времени, что достигается соответствующей синхронизацией часов процессоров путем одновременного изменения значений младших разрядов всех часов.

Общее поле памяти многопроцессорных систем в ЕС ЭВМ реализуется либо с помощью многовходовых адаптеров модулей памяти, либо с помощью коммутаторов межмодульных связей (коммутаторов перекрестных связей). Так, например, в двухпроцессорной системе на основе ЭВМ ЕС-1045 общее поле ОП (до 8 Мбайт) строится с использованием двухвходовых адаптеров памяти (АП) (рис. 15.6).

В ЕС ЭВМ на основе рассмотренных средств комплексирования можно создавать многомашинные комплексы, содержащие до четырех ЭВМ, причем каждая из них может быть двухпроцессорной.

Важным направлением повышения производительности ЭВМ и систем общего назначения является включение в их состав специализированных процессоров, ориентированных на определенный круг задач. Примером является «матричный процессор», подключаемый как

канал ввода-вывода и воспринимающий и по-своему интерпретирующий команды ввода-вывода и управляющие слова канала, при этом обеспечивается возможность параллельных вычислений в процессоре ЭВМ и матричном процессоре. Матричный процессор предназначен для выполнения над потоком входных данных операций свертки, корреляции, преобразования Фурье, операций над матрицами и векторами. В матричном процессоре при выполнении арифметических операций реализуется конвейерная обработка (совмещаются операции умножения, сложения с нормализацией, выдача результатов). ЭВМ ЕС-1045 в штатном составе имеет производительность 800 тыс. операций/с, а при подсоединении матричного процессора и решении задач с многократно повторяющимися действиями над группой данных (например, матрицами) производительность возрастает до 28 млн. эквивалентных операций/с.

15.4. МНОГОПРОЦЕССОРНАЯ СИСТЕМА С КОММУТАТОРОМ МЕЖМОДУЛЬНЫХ СВЯЗЕЙ («ЭЛЬБРУС»)

Одним из способов построения МВС универсального назначения является использование быстродействующего коммутатора межмодульных связей (КМС). Коммутатор, приобретающий в таких системах характер центрального устройства, обеспечивает возможность установления связи любого процессора с любым модулем памяти и любым модулем управления вводом-выводом, а также связи любого модуля управления вводом-выводом с любым модулем памяти (рис. 15.7).

Коммутатор образует межмодульные связи путем установления межсоединений в соответствующих точках пересечения прямоугольной сетки шин. Соединения между модулями сохраняются на все время передачи данных. Возможны одновременная связь и передача данных через коммутатор между различными парами модулей системы.

Таким образом, КМС реализует пространственное разделение при коммутации соединений в отличие от временного раз-

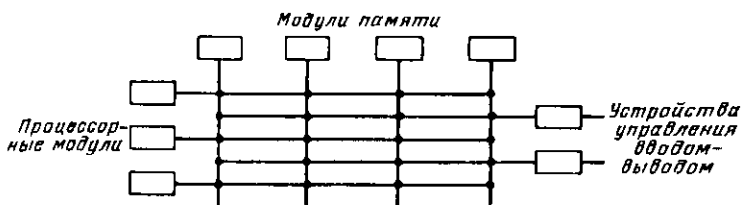


Рис. 15.7. Коммутатор межмодульных связей

деления при использовании для межмодульных связей общей шины.

При наличии коммутатора существенно уменьшается число конфликтов в системе, так как сам коммутатор не порождает конфликтов, и последние могут возникнуть только при запросах от нескольких устройств на связь с одним и тем же модулем. Однако следует иметь в виду, что коммутатор представляет собой достаточно сложное устройство, обусловленное высокими требованиями к его быстродействию, сложностью реализуемых им логических функций и необходимостью обеспечить возможность изменения за счет модульного построения самого коммутатора числа подключаемых к нему устройств (модулей) системы. Чтобы достигнуть высокой общей производительности системы, время переключения КМС должно быть существенно меньше времени выполнения операции в процессоре.

К МПС, использующим коммутатор межмодульной связи, относится разработанная под руководством чл.-корр. АН СССР В. С. Бурцева вычислительная система «Эльбрус», структура которой представлена на рис. 15.8 [17]. Эта универ-

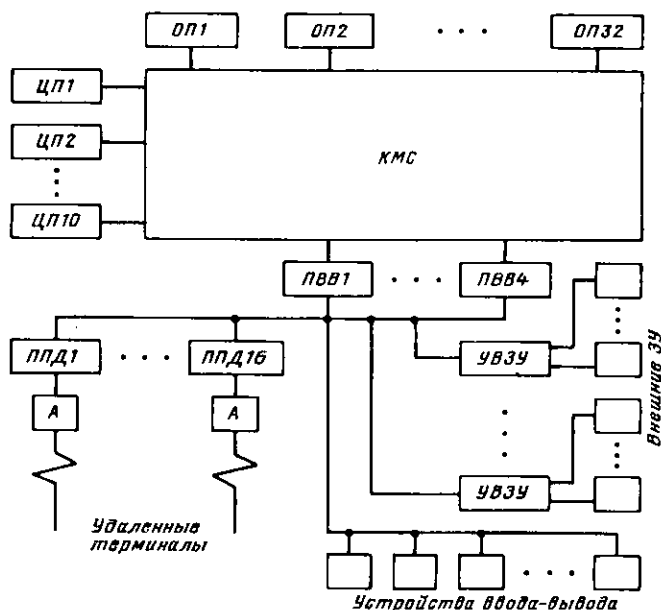


Рис. 15.8. Структура многопроцессорной ВС «Эльбрус»

сальная по назначению система с самого начала проектировалась как многопроцессорная, причем многопроцессорность рассматривалась как средство достижения высокой производительности (до 12 млн. операций/с в системе «Эльбрус-1» и до 100 млн. операций/с в «Эльбрус-2») при решении широкого круга задач, а также как средство обеспечения повышенной надежности и живучести за счет автоматической реконфигурации.

Система «Эльбрус» в зависимости от конфигурации содержит от 1 до 10 центральных процессоров и от 4 до 32 модулей ОП (соответственно емкость памяти от 576 до 4608 Кбайт). К коммутатору межмодульных связей помимо процессоров и модулей памяти может быть подключено до четырех процессоров ввода-вывода (ПВВ).

С помощью коммутатора реализуется общее поле оперативной памяти и общее поле процессоров ввода-вывода. Процессоры ввода-вывода через дополнительный коммутатор (на схеме не показан) и соответствующие блоки управления имеют доступ к любому периферийному устройству, т. е. обеспечивается общее поле периферийных устройств.

Работа системы через линии связи с удаленными терминалами осуществляется с помощью специализированных процессоров передачи данных (ППД1 – ППД16), подключенных к ПВВ1 – ПВВ4, и адаптеров А.

Процессоры ввода-вывода полностью освобождают центральные процессоры от управления операциями ввода-вывода.

Общая операционная система комплекса динамически распределяет ресурсы между решаемыми задачами и обеспечивает параллельную и независимую работу модулей системы.

Модули системы снабжены средствами аппаратного контроля правильности функционирования. При возникновении ошибки при передаче или преобразовании информации схемы контроля соответствующего модуля посылают сигнал, по которому операционная система производит реконфигурацию МПС, отключает неисправный модуль от МПС и выводит его в ремонт. По завершении последнего операционная система включает его в работу в составе МПС.

Многопроцессорная система «Эльбрус» в целом и входящие в ее состав ЦП отличаются кроме отмеченных выше и рядом других архитектурных решений.

Одной из важнейших особенностей МПС «Эльбрус» является приближение уровня машинного языка (системы команд) к уровню алгоритмического языка высокого уровня, в результате чего значительно повышается скорость трансляции программ, а следовательно, и производительность системы, облег-

чается труд программиста. К особенностям системы также относится широкое использование аппаратурных стеков, в том числе для реализации языка безадресных команд, соответствующего польской инверсной записи (см. гл. 8), повышения производительности системы, динамического распределения ресурсов (в первую очередь памяти и быстрых регистров), организации обращений к процедурам, обработки прерываний.

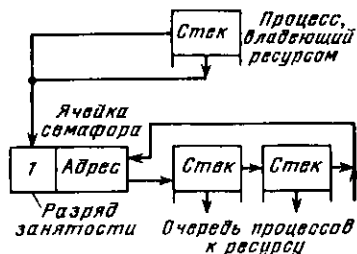
В МПС «Эльбрус» каждое слово памяти снабжается ярлыком (*тегом*), представляющим собой группу дополнительных разрядов, указывающих тип данных (операнд — число с плавающей запятой, целое число и т. п. или имя переменной — косвенная ссылка), а также режим защиты слова. Используется аппарат дескрипторов, являющихся гибким средством описаний хранящихся в памяти массивов информации. Дескриптор задает адресные границы массива, указывает, содержит ли массив данные или команды и некоторые другие характеристики массива. Механизм дескрипторов можно рассматривать как развитие косвенной адресации.

В данном параграфе ограничимся рассмотрением вопросов, относящихся к организации в этой системе многопроцессорной обработки данных.

В МПС «Эльбрус» реализуется *многопроцессная обработка данных*, при этом под *вычислительным процессом* понимается совокупность программ задачи (или комплекса программ), используемых ею данных и выделенных ей ресурсов системы. Каждому процессу в системе соответствует свой *стек* в ОП. Любой процессор работает на любом стеке. Самостоятельные стеки могут представляться не только каждой задаче, но и отдельным процедурам. Таким образом достигается широкое распараллеливание решения задач.

Операционная система динамически распределяет ресурсы системы между процессами с учетом их приоритетных соотношений. Процесс может быть *активным*, если он обрабатывается системой, или *пассивным*, если он ожидает некоего события: освобождения центрального процессора (заняты все процессоры), окончания необходимой для дальнейшей обработки процесса операции ввода-вывода, освобождения другим процессом общих данных. Пассивный процесс может находиться в неготовом и готовом для выполнения состоянии. Из очереди готовых процессов выбирается наиболее приоритетный и назначается на первый освободившийся процессор. При этом процесс получает в свое распоряжение аппаратуру процессора, в том числе указатель стека, сверхоперативную память, базовые регистры. В следующий раз при переходе из пассивного в активное состояние процесс может быть назначен на другой

Рис. 15.9. Синхронизация процессов при помощи семафоров



процессор. Допускается одновременное функционирование нескольких активных процессов, использующих общие данные и другие ресурсы, что достигается механизмом синхронизации процессов.

Синхронизация вычислительных процессов осуществляется с помощью *семафоров* (рис. 15.9).

Отдельным ресурсам системы — наборам данных, процессорам ввода-вывода и другим — ставятся в соответствие специальные ячейки в ОП — семафоры. Ресурс доступен процессу, если семафор открыт (в соответствующем разряде ячейки семафора 0). Занимая ресурс, процесс специальной командой «*Закрыть семафор*» устанавливает 1 в этот разряд ячейки.

Если семафор нужного процессу ресурса оказывается закрытым (в ячейке семафора стоит 1), процесс записывает в адресное поле семафора адрес своего стека (возможно образование у семафора очереди стеков), переходит в пассивное состояние, освобождая ЦП, который предоставляется наиболее приоритетному из очереди готовых процессов.

Занимая ресурс, процесс специальной командой «*Закрыть семафор*» устанавливает 1 в разряд занятости ячейки семафора. При освобождении ресурса, например после выполнения ПВВ запрошенного процессором ввода данных или завершения обращения другого процесса к общим данным, освобождающий ресурс процесс командой «*Открыть семафор*» заносит 0 в ячейку семафора соответствующего ресурса и ставит в очередь готовых процесс, адрес стека которого находится в ячейке семафора.

15.5. МНОГОПРОЦЕССОРНЫЕ И МНОГОМАШИННЫЕ КОМПЛЕКСЫ С ОБЩЕЙ ШИННОЙ (СМ ЭВМ)

Организация межмодульных (и межмашинных) связей на основе общей шины является одним из распространенных способов построения многопроцессорных и многомашинных вычислительных комплексов.

Общая структура многопроцессорной системы с общей шиной изображена на рис. 15.10. Входящие в состав системы процессоры, модули ОП, блоки управления периферийными

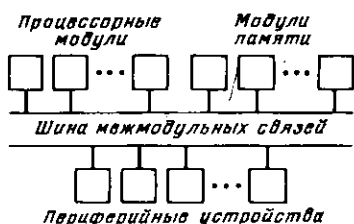


Рис. 15.10. Многопроцессорные ВС с общей шиной межмодульных связей

устройствами (ВЗУ и устройствами ввода-вывода) подключены к одной общей шине, состоящей из линий, по которым передаются ин-

формационные и управляющие сигналы.

Одновременно через общую шину может передаваться информация только между двумя устройствами, т. е. шина используется подсоединенными к ней устройствами *в режиме разделения времени*. Это является причиной возникновения конфликтов, при которых несколько устройств претендуют на занятие шины. Наличие конфликтов на общей шине вызывает простои оборудования и уменьшает производительность системы. Частоту возникновения конфликтов и потери из-за них производительности удастся снизить, если наряду с общей доступной всем процессорам памяти иметь для каждого процессора небольшую местную (локальную) оперативную память, к которой процессор может обращаться непосредственно, минуя общую шину (см. § 17.4).

Из-за конфликтов в общей шине, снижающих производительность системы, организация межмодульных связей на основе общей шины не находит применения в МПС, ориентированных на достижение высокой производительности. Однако общая шина широко применяется при построении многопроцессорных и многомашинных вычислительных комплексов, предназначенных для обеспечения повышенной надежности и живучести, необходимой производительности и пропускной способности при работе в режиме реального времени в различных системах управления технологическими процессами, автоматизации экспериментов и испытаний и др. Применению шинной структуры способствует то, что в оборудовании этих комплексов широко используются малые и микро-ЭВМ, микропроцессорные средства, в которых реализованы различные модификации интерфейса «общая шина».

В СМ ЭВМ при построении на основе аппаратуры малых ЭВМ СМ-4 многопроцессорных и многомашинных комплексов с шинной структурой в качестве средств комплексирования (системных средств) служат переключатель шины (ПШ) и адаптер межшинной связи (АМС) (рис. 15.11) [54, 59, 85].

Переключатель шины. С помощью ПШ дополнительная общая шина ДОШ с подсоединенным к ней оборудованием мо-

жет быть подключена к общей шине одного из двух процессоров комплекса или устанавливаться в отключенное состояние. Дополнительная общая шина допускает подключение любых устройств (модулей памяти, ПУ и др.), кроме процессоров.

После подключения *ДОШ* становится участком соответствующей общей шины комплекса и обмен с подсоединенным к ней оборудованием как программно-управляемый, так и с прямым доступом производится по обычным процедурам интерфейса общей шины.

Переключатель шины (рис. 15.12, а) содержит блок управления и синхронизации *БУС*, таймер, электронные ключи для подключения шины (для каждой из основных шин), формирователи, обеспечивающие передачу сигналов между основной и дополнительной шинами, регистр управления и состояния *РЗУС* и схему формирования сигналов прерывания.

Процессор *Пр1* при необходимости обмена информацией с устройствами, присоединенными к *ДОШ*, устанавливает 1 в разряд запроса *РЗУС1*. Если *ДОШ* свободна (на это указывает 0 в разряде занятости *РЗУС2*), она не более чем на 1 мкс подключается к *ОШ1*. Если *ДОШ* занята (подключена к *ОШ2*), устанавливается 1 в разряд «запроса от другого процессора» *РЗУС2*, посылается прерывание в процессор *Пр2* и включается таймер. В ответ на прерывание *Пр2* не позже чем через 10 мс должен освободить *ДОШ* либо сбросить «запрос другого процесса» в *РЗУС2*. В противном случае (процессор *Пр2* неисправен)

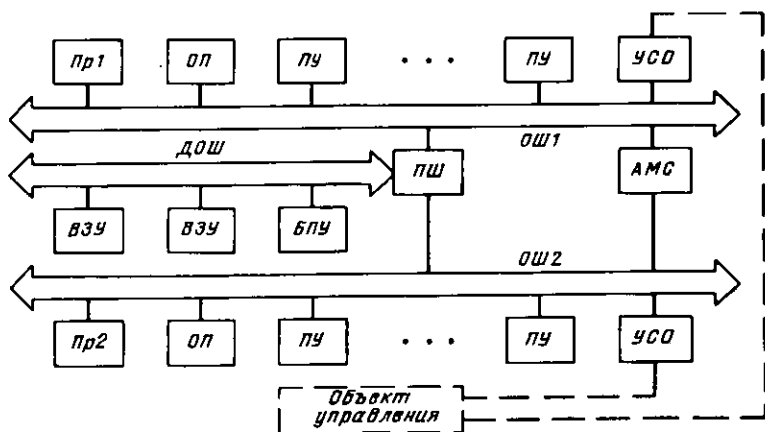


Рис. 15.11. Пример использования системных средств ЭВМ СМ-4 — переключателя шины и адаптера межшинной связи — для построения многомашинных и многопроцессорных комплексов:

Пр1, *Пр2* — процессоры; *УСО* — устройство связи с объектом управления; *БПУ* — быстродействующее печатающее устройство

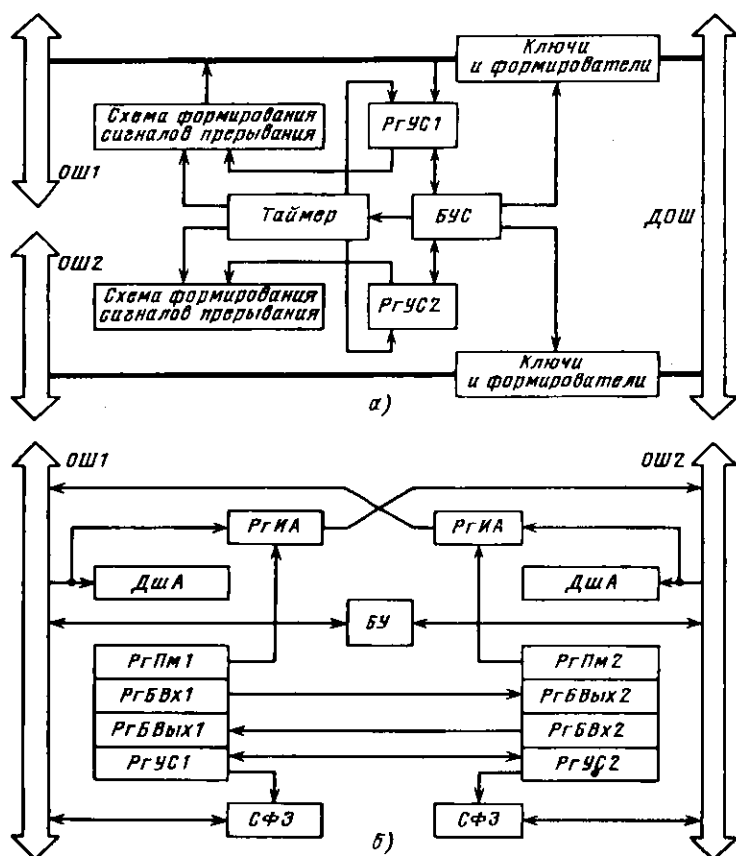


Рис. 15.12. Структурные схемы переключателя шин (а) и адаптера межшинной связи (б)

ДОШ подключается к *ОШ1*. Если *Пр2* сбросит «запрос другого процессора», то *Пр1* периодически возобновляет свой запрос на *ДОШ*.

Адаптер межшинной (межмикропроцессорной) связи предназначен для осуществления быстрого доступа процессора и ПУ одного комплекса к памяти и ПУ другого комплекса. Здесь под комплексом понимается совокупность устройств, подключенных к одной общей шине.

Наличие АМС позволяет реализовать совместное решение комплексами задачи, иерархическую организацию обработки данных с обменом данными между главным и подчиненным комплексами, различные режимы резервирования оборудования.

Межшинная связь с помощью АМС основана на автоматическом преобразовании адресов в командах инициированного межкомплексный обмен информацией «комплекса-задатчика» в адреса «комплекса-исполнителя». Это преобразование осуществляется с помощью *окна интерфейса*, представляющего собой фиксируемую при изготовлении комплекса часть неиспользуемых им адресов, позволяющую адресовать от 512 до 32 Кслов (размер окна). Размер окна ограничивает размер блока данных, передаваемых из одного комплекса в другой.

Адаптер позволяет производить обращение устройства одного комплекса с помощью адресных команд процессора через окно к устройствам другого комплекса для выборки команды, записи и считывания данных, выполняемых в этом случае в другом комплексе как передача с прямым доступом, и выполнение запроса программного прерывания из другого комплекса.

Процессор задает область адресов, к которым через свое окно может обратиться другой комплекс, имеет возможность запретить обращение из другого комплекса к устройствам своего комплекса или только запретить запись в свои устройства, запретить прерывания от другого комплекса.

Адаптер (рис. 15.12, б) содержит две одинаковые секции, подключаемые к разным ОШ как периферийные устройства, позволяющие производить обращения к своим адресуемым регистрам. В состав секции входят адресуемые регистры: управления и состояния $P_2УС$, буферные входной $P_2БВх$ и выходной $P_2БВых$, исполнительного адреса $P_2ИА$, адреса перемещения $P_2Пм$, а также дешифратор адреса $ДшА$, блок управления $БУ$ и схемы формирования запросов прерывания $СФЗ$.

Регистр исполнительного адреса (младшие его разряды носят название регистра смещения) содержит сформированный окном исполнительный адрес обращения к другому комплексу. Младшие разряды этого адреса («смещение») берутся из адреса окна, выставленного на своей шине комплексом — инициатором обмена, а старшие — из регистра адреса перемещения, устанавливаемого другим комплексом при подготовке обмена через окно. Регистр адреса вектора прерывания реализуется в виде колодки контактов, на которые подаются уровни напряжений 0 и 1.

Организация связи между комплексами через окна интерфейса поясняется на рис. 15.13. Адресное пространство 0—64 К обоих комплексов используется модулями памяти. Неиспользуемые 8 К адресов (64—72 К) предоставлены для окна интерфейса (фиксируются при изготовлении комплекса). При подготовке передачи комплекс-исполнитель засылает перемещение в АМС, устанавливая таким образом связь окна другого комплекса с областью своих адресов. Так, на рис. 15.13 первый комплекс через свое окно связан с ПУ второго комплекса,

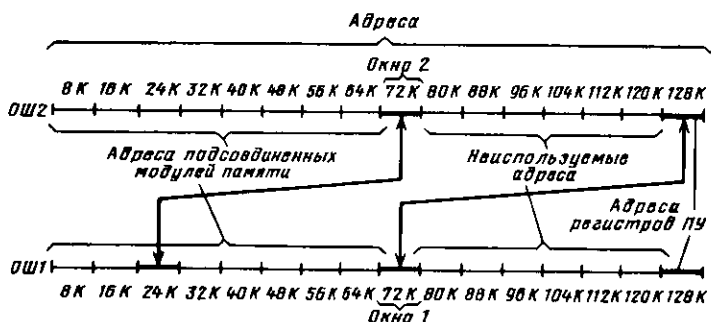


Рис. 15.13. Организация межмашинной связи через окно интерфейса

а второй комплекс — через свое окно с областью 24—32 К памяти первого комплекса. Время преобразования адреса в окне составляет примерно 300 нс.

Рассмотрим особенности процедуры выполнения процессором *Пр2* операции с использованием в качестве операнда слова с адресом *A* из памяти первого комплекса. Предварительно *Пр2* устанавливает в АМС *перемещение* — группу старших разрядов адреса, задающих область памяти с адресом *A*, и выдает команду чтения по адресу *A'*, принадлежащему окну 2. Выполняя эту команду, *Пр2* устанавливает на *ОШ2* запрос на чтение по адресу *A'*. Окно 2 с помощью перемещения, предварительно установленного *Пр2*, трансформирует этот адрес в адрес *A* области памяти первого комплекса и выставляет на *ОШ1* запрос прямого доступа. Запрашиваемая информация (слово) из адреса *A* памяти первого комплекса передается во входной буферный регистр *Р:Вх1*. Затем это слово из выходного буферного регистра *Р:ВВх2* пересылается по *ОШ2* в процессор *Пр2*.

Аналогично одной командой выполняется передача блока информации от одного комплекса другому. Например, для передачи в память первого комплекса блока данных с ЗУ на дисках второго комплекса процессор *Пр1* предварительно устанавливает перемещение для задания области своей памяти (24—32 К), куда будут записываться данные с ЗУД, и выдает на *ОШ1* запрос на чтение по соответствующему адресу из области адресов своего окна. Окно 1 преобразует этот адрес в адрес регистра ЗУД и выставляет на *ОШ2* запрос прямого доступа для передачи блока данных с ЗУД по адресам окна 2. При этом адреса окна 2 преобразуются в адреса области (24—32 К) памяти первого комплекса, установленной перемещением.

На одной ОШ может быть организовано несколько окон (выделено несколько областей неиспользуемых адресов) для связи с несколькими комплексами.

На рис. 15.11 показан пример использования *ПШ* и *АМС* для построения на основе оборудования ЭВМ СМ-4 многомашинных управляющих вычислительных комплексов. В представленной структуре реализуются общее поле памяти и общее

поле ВЗУ, двухпроцессорная параллельная обработка данных и резервирование наиболее важных для управления объектом устройств — процессора, памяти, УСО и некоторых ПУ.

15.6. ОСОБЕННОСТИ ОРГАНИЗАЦИИ ОТКАЗОУСТОЙЧИВЫХ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ

Перспективы широкого применения построенных на основе микро-ЭВМ и микропроцессоров управляющих вычислительных устройств (УВУ) и комплексов (УВК) для управления в реальном масштабе времени различными технологическими процессами и установками выдвигают на первый план проблему обеспечения надежности их работы. Управляющие комплексы и устройства, работающие в составе различных АСУ ТП, должны обеспечивать длительное достоверное (безошибочное) функционирование без остановки технологического процесса при сбоях и отказах в оборудовании УВУ и УВК. В ряде случаев предъявляются жесткие требования в отношении недопустимости перерывов или задержек в осуществлении УВУ и УВК управляющих функций.

Указанную проблему призваны решать *отказоустойчивые УВУ и УВК*, т. е. устройства и комплексы, которые сохраняют работоспособность при выходе из строя отдельных элементов, узлов или модулей.

Вопросам обеспечения отказоустойчивости уделялось довольно много внимания в рамках ЭВМ III поколения, и при этом получены определенные результаты, например в составе ЕС ЭВМ созданы многомашинные и многопроцессорные комплексы (см. § 15.3), хотя ЭВМ ЕС, как, впрочем, и машины СМ ЭВМ, сначала проектировались в основном для одномашинного и однопроцессорного режимов работы. Новые возможности, связанные с небольшими размерами и сравнительно низкой стоимостью микропроцессорных и других БИС, позволяют существенно продвинуться в этом направлении путем создания ЭВМ и комплексов со специальной архитектурой, ориентированной на достижение отказоустойчивости.

Основным принципом, на основе которого создаются отказоустойчивые ЭВМ, системы и комплексы, является аппаратурная избыточность. В настоящее время определилось несколько направлений в организации отказоустойчивых вычислительных комплексов. При сравнении этих направлений следует учитывать размеры аппаратурной избыточности (хотя в ряде случаев это может и не иметь определяющего значения), меру обеспечения достоверности результатов обработки комплексом

данных, степень исключения влияния сбоев и отказов в аппаратуре на изменение временной диаграммы выдачи результатов обработки данных на объект управления.

Надежность УВУ и УВК при необходимости длительного функционирования весьма сильно зависит от степени их приспособленности к техническому обслуживанию (в первую очередь от ремонтпригодности) в конкретных условиях эксплуатации. Этот показатель может в ряде случаев оказаться основным при выборе варианта построения отказоустойчивого УВУ и УВК.

Организация отказоустойчивого УВК (его структура, динамическое изменение конфигурации и т. п.) должна быть «прозрачной» для пользователя в том смысле, что пользователь при взаимодействии с УВК и при подготовке и отладке программ для него должен воспринимать комплекс как обычную однопроцессорную ЭВМ с неизменной конфигурацией.

Отказоустойчивые УВК с автоматической реконфигурацией. Обычно функции (программы), реализуемые УВК, неравнозначны, и временная утрата некоторых второстепенных функций при сохранении основных жизненно важных допустима. Поэтому для таких УВК основной характеристикой надежности следует считать не среднее время наработки на отказ, используемое для оценки надежности ЭВМ общего назначения, а вероятность утраты на заданном интервале любой из основных функций вследствие отказов отдельных модулей системы. Эта вероятность может быть принята за меру живучести комплекса. Особенности построения отказоустойчивых УВК с автоматической реконфигурацией описаны в [75].

Отказоустойчивость достигается введением избыточного оборудования и логической организацией, обеспечивающей при отказах в оборудовании *автоматическую реконфигурацию* системы для сохранения жизненно важных функций, возможно, ценой утраты второстепенных. Системы, обладающие указанной способностью, иногда называют «системами с элегантной деградацией».

Основными принципами построения отказоустойчивых комплексов на основе автоматической реконфигурации являются *многоустойчивость* (в том числе

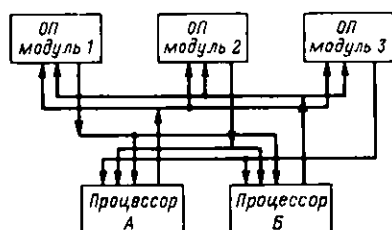


Рис. 15.14. Многократный интерфейс

многопроцессорность), общие поля процессоров, оперативной памяти, каналов и периферийных устройств; динамическое распределение функций между однотипными устройствами.

Многоустройство предполагает, что система должна содержать несколько экземпляров однотипных устройств (процессоров, модулей ОП, каналов и др.), при этом должна быть обеспечена избыточность устройств всех типов по отношению к минимальному набору, необходимому для выполнения жизненно важных функций.

Динамическое распределение функций означает, что программы не привязаны жестко к процессорам, каналам и ПУ. Заранее неизвестно, какое из однотипных устройств будет выполнять данную функцию. Более того, работа может быть начата на одном, продолжена на другом и закончена на третьем устройстве.

Наличие общих полей устройств и динамического распределения функций позволяет комплексу сохранять работоспособность, пока имеется хотя бы по одному исправному устройству каждого типа.

Практическое осуществление отказоустойчивых УВК с автоматической реконструкцией требует реализации рассматриваемых ниже логических свойств.

Многосвязные интерфейсы (рис. 15.14) позволяют многовходовым устройствам связываться друг с другом по независимым шинам. В отличие от характерных для ЭВМ общего назначения односвязных интерфейсов, в которых общие (коллективные) шины разделяются устройствами во времени, в многосвязном интерфейсе исключается выход из строя комплекса при неисправности в одной из шин. Это требует дополнительного оборудования.

Неявная адресация. Однотипные устройства, обслуживающие операции ввода-вывода [каналы, блоки управления (ПУ)], имеют одинаковые логические номера и поэтому фактически командой ввода-вывода не адресуются. Запрос на выполнение операции ввода-вывода воспринимается первым по пути прохождения запроса исправным и свободным устройством данного типа.

Динамическое распределение (диспетчирование) программ (рис. 15.15). В ОП организуются очереди для программ (работ), имеющих различные уровни приоритета. Программы заносятся в очереди своего приоритета прерываниями или идущими на процессорах программами. Аппаратура выделяет среди процессоров кандидата на прерывание, которым является процессор, обрабатывающий программу наименьшего приоритета.

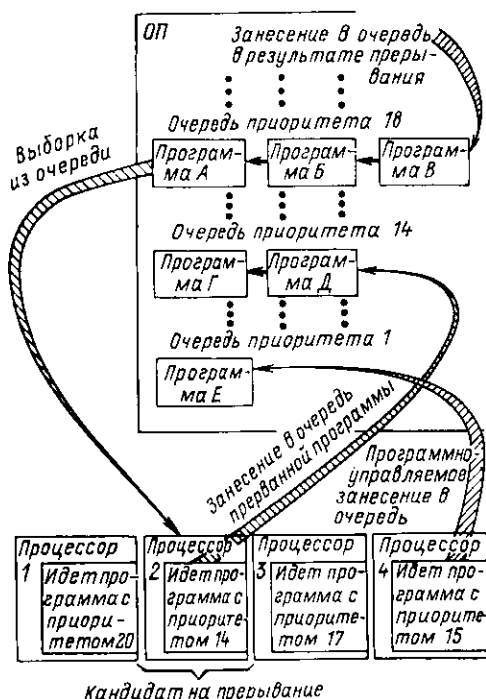


Рис. 15.15. Динамическое распределение программ (приоритет растет с увеличением номера очереди)

Аппаратура непрерывно сравнивает приоритет программы процессора — кандидата на прерывание с приоритетом программ, находящихся в очередях. Если в очереди появляется программа большего приоритета, процессор-кандидат прерывает свою программу, заносит ее в очередь соответствующего приоритета и начинает выполнять программу из непустой очереди наибольшего приоритета. Обработка прерванной программы будет продолжена, причем необязательно на том же процессоре, когда ее приоритет станет выше приоритета программы — кандидата на прерывание. Следует обратить внимание на то, что здесь образуется общая очередь к освобождающимся процессорам.

Описанный механизм позволяет достигнуть взаимной независимости программ и процессоров и обеспечивает в каждый момент времени выполнение наиболее приоритетных программ на наличных исправных процессорах.

Системы автоматического контроля правильности работы устройства и диагностирования — см. гл. 12.

Иерархия рестартов (восстановлений) программ при сбоях и отказах. Для уменьшения потерь времени на восстановление программ после сбоев и отказов следует иметь возможность восстанавливать работоспособность программ на разных уровнях: а) сбои и отказы в ОП; б) сбои в процессорах и каналах; в) отказы в процессорах и каналах; г) сбои и отказы в периферийных устройствах.

В ОП сбои и отказы, приводящие к ошибке в одном разряде (при записи или считывании слова), исправляются с помощью самокорректирующего кода Хэмминга (см. гл. 12).

Микрокомандный рестарт. Средства контроля проверяют правильность каждой микрооперации, и при неправильном выполнении микрокоманда повторяется. Для микрокомандного рестарта необходимо, чтобы в процессе исполнения микрооперации сохранялись ее операнды. Это требует дополнительного оборудования (регистров и др.) и приводит к некоторому уменьшению скорости работы процессора.

Если при повторении микрокоманда выполнится правильно, обработка программ на устройстве продолжается. Если повторение микрокоманды установленное число раз не дает правильного результата, фиксируется отказ.

Рестарт при отказе процессора (рис. 15.16). После фиксации системой контроля отказа состояния регистров неисправного процессора запоминаются в выделенной для этого области памяти. Неисправный процессор посылает сигнал отказа во все другие процессоры. Этот сигнал воспринимает процессор — кандидат на прерывание. Он заносит свою программу в соответствующую очередь программ, а из области аварийной регистрации считывает информацию о состоянии регистров отказавшего процессора и продолжает обработку программы последнего, начиная с неправильно выполненной им микрокоманды.

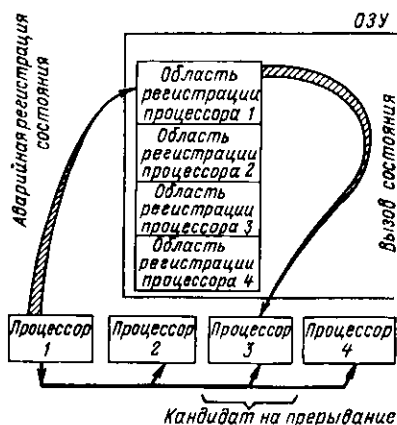


Рис. 15.16. Рестарт при отказе процессора

Рестарт при сбоях и отказах в периферийных устройствах состоит в повторении, начиная с ближайшей контрольной точки подпрограммы операции ввода-вывода (канальной программы).

Средства повышения ремонтоспособности. Обеспечение свойства живучести системы на протяжении длительного интервала времени возможно лишь при высокой ремонтоспособности оборудования, когда обеспечиваются быстрый ремонт и ввод исправленных устройств в эксплуатацию. Повышение ремонтоспособности достигается аппаратурно-программными средствами, обеспечивающими возможность без перерыва в работе исправной части оборудования УВК производить *программно-управляемое отсечение оборудования* для ремонта, *программно-управляемое присоединение* к комплексу вводимых в эксплуатацию устройств, *программно-управляемое изменение номиналов питающих напряжений и параметров основных сигналов* для автоматизации профилактических испытаний оборудования.

В вычислительных комплексах с автоматической реконфигурацией должна обеспечиваться независимость операционной системы и целевого программного обеспечения («прозрачность») относительно конфигурации аппаратурных средств.

Достоинством отказоустойчивых комплексов с автоматической реконфигурацией является сравнительно небольшой объем избыточного оборудования. Однако в подобных системах достоверность функционирования проверяется только схемами контроля, что может оказаться недостаточным, так как схемы контроля охватывают обычно не все оборудование комплекса, во многих случаях реагируют только на одиночную ошибку и, кроме того, в них самих могут возникать отказы. Комплексы с реконфигурацией не позволяют полностью исключить задержки в реализации управляемых функций, так как требуется определенное время на процедуры селекции сбоев и отказов и процедуры рестартов. Приспособленность к техническому обслуживанию (ремонтпригодность и др.) фактически остается аналогичной приспособленности обычных машин.

Вычислительные комплексы с мажоритарным управлением (тройное дублирование). Принцип мажоритарного управления состоит в выполнении одних и тех же вычислений несколькими (более двух) ЭВМ или устройствами, сопоставлении получаемых машинами результатов по методу «голосования» и выдаче на объект управления результатов обработки, оказавшихся одинаковыми у большинства машин (устройств) комплекса. Комплекс снабжается мажоритарным устройством, в определенные моменты сравнивающим выходные данные машин

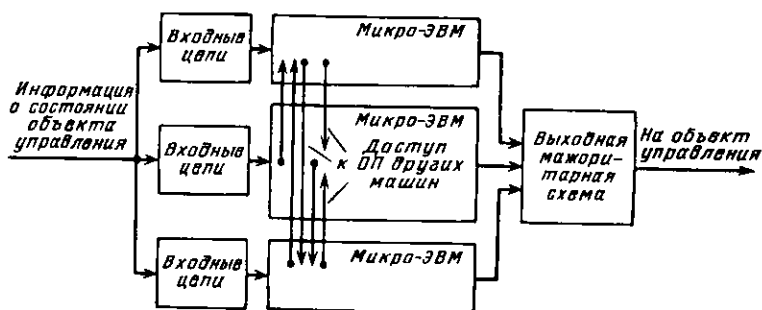


Рис. 15.17. Отказоустойчивый УВК с мажоритарным управлением

и формирующим путем «голосования» общий выходной управляющий сигнал. Выход из строя одного из элементов практически не сказывается на выходных сигналах УВК. Комплексы с мажоритарным управлением в состоянии практически исключить ошибки в управлении.

Примером УВК с тройным дублированием и мажоритарным управлением может служить система «Август» (рис. 15.17) [91].

Три одинаковые управляющие микро-ЭВМ выполняют независимо все операции. Имеется возможность обращения для считывания одной микро-ЭВМ к памяти двух других. Это используется для обнаружения и исправления ошибок.

В комплексе реализуются три степени мажоритарного выбора (рис. 15.18).

1. Каждая микро-ЭВМ принимает от датчиков входные сигналы через свои входные цепи. Затем микро-ЭВМ считывает значения входных величин, полученные другими машинами, и при необходимости корректирует свои входные данные по правилу «2 из 3». Таким образом исключаются влияния неисправностей входных схем.

2. Каждая микро-ЭВМ (процессор) по выравненным входным данным производит одни и те же вычисления, определяя свой вариант выходных управляющих сигналов или величин для объекта управления. Затем микро-ЭВМ сравнивают полученные выходные результаты и выравнивают их по правилу «2 из 3».

3. Одинаковые выходные результаты выдаются через собственные интерфейсы каждой микро-ЭВМ на мажоритарную схему (аппаратурный мажоритарный выбор).

Достоинствами рассматриваемого подхода являются последовательный контроль и коррекция ошибок от сбоев и отказов

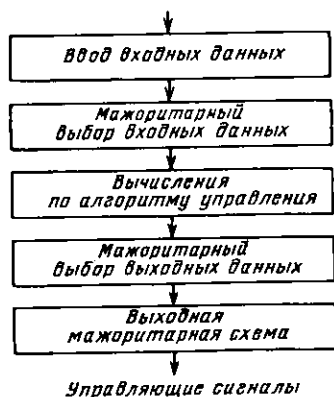
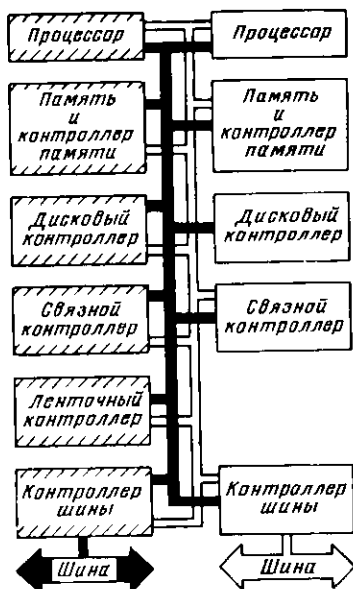


Рис. 15.18. Многоступенчатый мажоритарный выбор

Рис. 15.19. Отказоустойчивый УВК с двухуровневым дублированием



в подсистемах ввода, обработки и вывода данных. К недостаткам помимо большего, чем при автоматической реконфигурации, объема аппаратуры можно отнести следующее. Выравнивание информации и определение неисправного модуля производится программным путем с соответствующей потерей времени и задержками в управлении. Обнаруженный неисправный модуль является достаточно сложным устройством, и дальнейшее уточнение места неисправности требует достаточно квалифицированного обслуживания.

Отказоустойчивые комплексы с двухуровневым дублированием (четырёхкратное дублирование). Небольшие размеры и низкая стоимость микропроцессорных схем позволяют реализовать многопроцессорные комплексы с обеспечением отказоустойчивости на уровне аппаратуры и полностью исключить перерывы или задержки в осуществлении комплексом управляющих функций, которые могут возникнуть при рестартах после отказов или сбоях, выполняемых программными или микропроцессорными средствами. Этого можно достигнуть, используя двухуровневое дублирование аппаратуры, подобно тому, как это сделано в системе «Стратус» [93].

Комплекс содержит двойной комплект всех плат (модулей), попарно выполняющих одновременно одни и те же действия (первый уровень дублирования). На каждой плате размещены

полностью дублированные тракты преобразования информации и схемы, контролирующие совпадение выходных сигналов обоих логических трактов платы и блокирующие выдачу сигналов в случае несовпадения (второй уровень дублирования). Структура комплекса представлена на рис. 15.19.

При обнаружении в какой-либо плате несовпадения выходных сигналов ее трактов из-за сбоя или отказа данная плата логически отключается от комплекса, а в операционную систему и на панель сигнализации оператора поступает соответствующий сигнал, при этом нормальную работу продолжает дублирующая плата и не требуется никакой специальной процедуры рестарта.

Рассматриваемый подход требует большего объема аппаратуры, чем тройное дублирование с мажоритарным управлением. Однако значительно упрощаются сравнивающие процедуры и схемы, упрощаются схемы аппаратного контроля правильности функционирования, полностью исключаются оперативные действия программных и микропрограммных средств после сбоев и отказов (так как отказоустойчивость обеспечивается на аппаратном уровне, отказы и сбои обнаруживаются и их последствия блокируются непосредственно в момент их возникновения), исключаются возвраты к контрольным точкам, которые имеются при использовании программных и микропрограммных средств в процедурах рестартов. В комплексах с двухуровневым дублированием в большой степени упрощается техническое обслуживание аппаратуры, так как автоматически обнаруживаются неисправные платы и имеется возможность замены последних без остановки УВК.

При создании УВК с двухуровневым дублированием должны быть решены вопросы синхронизации работы дублированных плат и логических трактов, особенно при работе с периферийными устройствами.

15.7. МНОГОМИКРОПРОЦЕССОРНЫЕ КОМПЛЕКСЫ НА ОСНОВЕ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ МИКРОПРОЦЕССОРОВ

Важное место среди многопроцессорных систем должны занять *многомикропроцессорные системы*. Среди них к числу наиболее сложных следует отнести системы коллективного пользования и распределенные системы обработки данных. Такие системы требуют больших вычислительных мощностей, высокой надежности, развитых систем программного обеспечения. Одним из первых микропроцессоров, разработанных для такого применения, является «объектно-ориентированный» 32-раз-

рядный микропроцессор (микросистема) iAPX-432 фирмы Intel [79, 122].

Этот микропроцессор в основном состоит из трех сверх-больших интегральных схем с 64 выводами. На двух из них, содержащих вместе более 150 тыс. транзисторов, реализован «процессор общего назначения» (ПрОН), а на третьей (содержащей около 65 тыс. транзисторов) — «процессор интерфейса» (При).

Рассматриваемый микропроцессор построен на основе высококачественной технологии МОП-схем (Н-МОП-технологии). Он использует один источник питания 5 В и работает с тактовой частотой 8 МГц.

Процессоры ПрОН и При имеют микропрограммное управление с УП в виде встроенных в СБИС процессоров ПЗУ емкостью соответственно 4 и 2 К 16-разрядных слов.

Микропроцессор iAPX-432 работает с виртуальной памятью, область адресов которой составляет 2^{40} байт. Физическая ОП имеет объем до 2^{24} байт, т. е. 16 Мбайт.

Архитектура микропроцессора iAPX-432 ориентирована на использование языка высокого уровня АДА. Ряд функций операционной системы, в том числе интерпретаторы входных языков высокого уровня, реализованы аппаратурными, точнее, микропрограммными средствами.

В многомикропроцессорных системах возможно использование различных вариантов структур и функционирования. Представляет удобство такая логическая организация, которая дает возможность изменения в системе числа ПрОН и При без изменения программного обеспечения. Модули программного обеспечения при этом не требуют информирования о запуске в работу или останове какого-либо ПрН и При.

Построение многомикропроцессорных систем требует стандартизации способа связи процессоров и памяти. В рассматриваемом комплексе способ связи фиксируется разработчиком системы, при этом не используется какая-либо стандартная «шина», а реализуется стандартный протокол взаимодействия процессоров с памятью и друг с другом. Основной задачей, решаемой с помощью протокола, служит минимизация времени занятости шины процессорами. Протокол предусматривает формирование запросов и ответов в пакеты. Процессор занимает шину только в течение того времени, которое необходимо для передачи пакета. Передав пакет запросов, процессор освобождает шину, а через некоторое время получает пакет ответов. В промежутках между передачами этих пакетов шину могут использовать другие процессоры.

Формат пакета позволяет процессорам посылать оповеща-

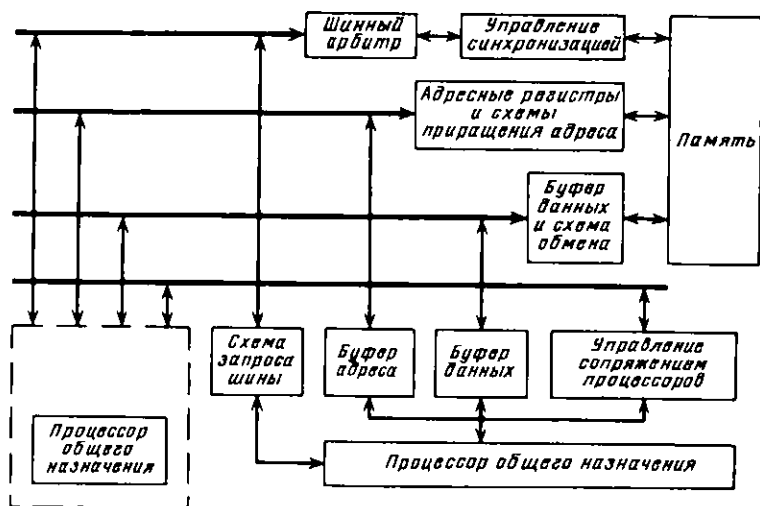


Рис. 15.20. Упрощенная шинная структура многопроцессорного комплекса (на основе микропроцессора iAPX-432)

тельный сигнал одному или сразу нескольким процессорам о засылке для них сообщения в область памяти для межпроцессорных сообщений. Принятое сообщение служит указанием о выполнении некоторых действий, в том числе об останове или запуске в работу.

Пример шинной структуры, в которой реализуется протокол взаимодействия процессоров и памяти, представлен на рис. 15.20. Эта структура обеспечивает взаимодействие до четырех ПрОН. При большем числе процессоров возможно возникновение конфликтных ситуаций и требуется более сложная схема.

Ввод-вывод в рассматриваемом комплексе реализуется с помощью вспомогательного 16-разрядного микропроцессора («процессора ввода-вывода»), подключаемого к системе через процессор интерфейса. В центральной части системы взаимодействия осуществляются при помощи сообщений. Процессор интерфейса, принимая сообщения, которые посылаются подсистеме ввода-вывода, запоминает их и вызывает прерывание в процессоре ввода-вывода, который выбирает сообщения из внутренней памяти процессора интерфейса. При пересылке сообщений в центральную систему процессор ввода-вывода записывает сообщение в оперативную память процессора интерфейса. Этот процессор имеет возможность выполнять ряд

функций, в частности пересылать сообщения. Поэтому При может передать сообщение от микропроцессора ввода-вывода так, что ПрОН воспримет его так же, как и сообщения, посылаемые другими ПрОН.

Архитектурной особенностью микропроцессора iAPX-432 является его объектная ориентация. В это понятие вкладывается использование единой формы для задания самых разных информационных блоков. В качестве объектов рассматриваются объекты данных, доменные объекты, контекстные объекты, объекты процесса, объекты ресурсов памяти, объекты портов и процессорные объекты. Все объекты являются содержимым памяти. Местонахождение объекта задается дескриптором.

Объекты данных являются простейшими объектами, с помощью которых обеспечиваются основные вычислительные операции. Это просто линейный участок памяти, хранящий двоичные данные любого типа.

Доменный объект содержит все дескрипторы для обращения к командам и объектам данных программного модуля. Домен задает также связи с другими доменами и, таким образом, представляет связанную модульную программу.

Контекстные объекты — это средства, обеспечивающие динамическое распределение памяти и активизацию процедур с объектами данных.

Независимая отдельно выполняемая задача называется объектом процесса. Объект, представляющий собой часть свободной памяти системы, называется объектом ресурсов памяти. Часть памяти, используемая для буферизированной передачи между процессами, называется объектом порта.

Сам физический процессор не является объектом. Процессорным объектом является соответствующая ему структура данных в памяти, содержащая сведения о состоянии процессора (работа или останов и др.), результатах диагностических процедур, а также ссылки на другие объекты, необходимые и доступные процессору.

Процессорный объект может находиться в любом месте памяти. Процессор обращается к объекту, используя соответствующую ссылку.

Процессор запускает процесс. Ссылка на объект процесса находится в объекте процессора. Объект процесса — это структура данных, описывающих процесс, — состояние процесса (работа, ожидание), как он планируется, приоритетность процесса и т. д.

Следующим элементом является контекст (процедура), определяющий совокупность команд, соответствующую некоторой завершенной части в процессе. Контекстный объект со-

держит ссылки на все объекты, к которым может обращаться контекст. Последним элементом в иерархии является объект данных.

Иерархическое построение объектов с соответствующими ссылками, динамически изменяемыми при работе процессора, является гибким механизмом, упорядочивающим и ограничивающим доступ к информации в памяти, что повышает надежность функционирования системы.

Объекты хранятся в сегментах памяти. Информация об объекте, определяющая его тип и размещение в физической памяти, хранится в дескрипторе объекта. Адресация объекта осуществляется по дескриптору объекта. Местоположение этого дескриптора, в свою очередь, указывается в дескрипторе обращения к объекту. Дескрипторы обращения располагаются в специальном сегменте доступа. Обработка этого сегмента осуществляется только специальными командами.

Входящая в дескриптор объекта информация о длине объекта используется для защиты от обращений по адресам, выходящим за границы объекта. По коду типа дескриптора аппаратные средства определяют, что содержат объекты, и, зная это, реализуют ряд функций, традиционно выполнявшихся программными средствами.

В некоторых случаях аппаратные средства выполняют действия, независимые от конкретно выполняемых команд, как, например, при автоматической переписи из внешней памяти в оперативную отсутствующего сегмента, необходимого для работы системы. Таким образом, процессор работает в основном под управлением аппаратно распознаваемых объектов, а не программных средств пользователя. Программные средства отвечают за формирование и уничтожение объектов.

В системе команд микропроцессора, ориентированного на язык АДА, наиболее употребимые операторы этого языка при компиляции отображаются отдельными командами, что позволяет упростить как компилятор, так и объем компилируемой программы. Команды имеют переменную длину и адресность и допускают произвольное с точностью до 1 бита обращение к памяти. Команды могут не использовать обращения к регистрам, так как они грудно учитываются при компиляции, при этом операнды хранятся либо в адресуемой памяти, либо в стеке. Примеры команд микропроцессора iAPX-432, ориентированного на язык АДА, даны на рис. 15.21.

Предусматривается несколько уровней обеспечения отказоустойчивости.

Средства первого уровня производят контроль передач информации (в пространстве и времени) и повторение попыток

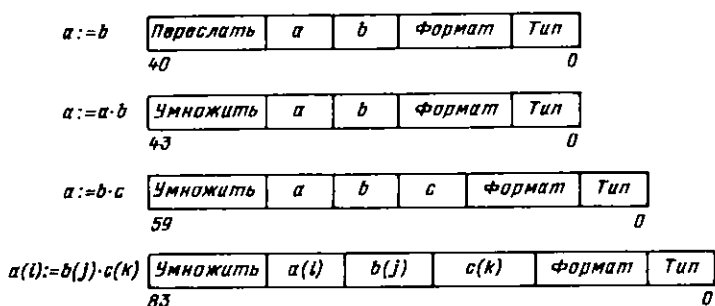


Рис. 15.21. Примеры команд (микропроцессор iAPX-432)

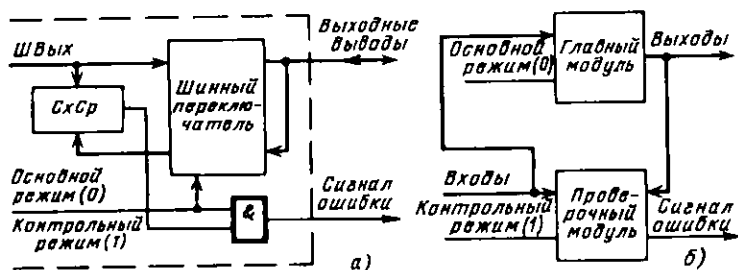


Рис. 15.22. Контроль работы модулей системы:

а — подключение выходных выводов; б — контроль по сравнению

выполнения операции при обнаружении искажения информации.

Средства второго уровня обеспечивают обнаружение ошибок и отказов при помощи дублирования модулей системы. Для этой цели используются микросхемы, работающие в двух режимах — основном и контрольном. В основном режиме микросхема функционирует как обычное устройство. В контрольном режиме выходные выводы используются как входы, сигналы с которых поступают в микросхему и через шинный переключатель подаются на схему сравнения СхСр, где сравниваются с сигналами, которые вырабатывала бы микросхема при работе в основном режиме. При обнаружении несоответствия микросхема выдает сигнал ошибки. Задание режима работы осуществляется отдельным управляющим сигналом (рис. 15.22, а). Включая параллельно две одинаковые микросхемы, одна из которых работает в основном режиме, а другая — в контрольном (рис. 15.22, б), можно с точностью до такта за-

фиксировать момент несовпадения значений сигналов, вырабатывавшихся логическими схемами разных устройств, и принять меры по устранению распространения искаженной информации.

Третий уровень обеспечения отказоустойчивости образуется за счет дополнительной аппаратурной избыточности — использования двухступенчатого дублирования (см. § 15.6), при котором каждый модуль системы строится из пары модулей «главный — проверочный» и дублирующей пары «главный — проверочный». Одна пара считается основной, а другая — дублирующей. Обе пары работают параллельно. Наличие в системе с двухступенчатым дублированием дублирующих шин позволяет производить реконфигурацию системы при обнаружении отказов в модулях или шинах.

Глава шестнадцатая

ПРИНЦИПЫ ОРГАНИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

16.1. ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ. ОБЩИЕ СВЕДЕНИЯ

Развитие архитектуры, аппаратных и программных средств ВС и успехи в развитии методов организации и аппаратуры систем передачи данных по каналам связи позволили приступить к созданию вычислительных систем качественно нового типа — вычислительных сетей.

Вычислительной сетью (ВСт) или *сетью ЭВМ* называется многомашинная система, состоящая из распределенных по большой территории ЭВМ, связанных между собой каналами передачи данных.

Целесообразность создания ВСт обуславливается возможностью использования территориально рассредоточенными пользователями программного обеспечения и информационных баз, находящихся в различных вычислительных центрах сети, возможностью организации «распределенной обработки» данных путем привлечения вычислительных ресурсов нескольких вычислительных центров сети для решения особо сложных задач.

Вычислительную сеть можно рассматривать как систему с распределенными по территории аппаратными, программными и информационными ресурсами. Возможна реализация на основе ВСт *распределенного* (децентрализованного)

банка данных, отдельные информационные базы которого создаются в местных вычислительных центрах, например в процессе функционирования АСУ отдельных предприятий и объединений, а при решении задач более высокого уровня управления используются как единая база данных.

Другая возможность — это создание *централизованного банка данных*, к которому имеют доступ многочисленные, в том числе находящиеся на значительном расстоянии абоненты через свои терминалы (*абонентские пункты*) и терминалы местных систем коллективного пользования.

Объединение в сеть ЭВМ нескольких вычислительных центров способствует повышению надежности функционирования вычислительных средств, так как создается возможность резервирования одних вычислительных центров за счет технических ресурсов других центров.

Вычислительная сеть позволяет оперативно перераспределять нагрузку между ЭВМ сети и снижать пиковую нагрузку на вычислительные средства.

С созданием ВСт возникли предпосылки для специализации отдельных ВЦ сети на решении задач определенного класса, что по оценкам специалистов [18] дает значительный эффект, так как позволяет существенно сократить общие затраты высококвалифицированного труда на разработку моделей, алгоритмов и пакетов прикладных программ. Специализация отдельных ВЦ сети становится возможной, так как пользователь имеет доступ к уникальным программам и данным, а также уникальным вычислительным средствам (например, специализированным процессорам) любого ВЦ сети.

В ВСт с программно-несовместимыми ЭВМ теряет остроту проблема переноса программных средств с одних машин на другие, так как пользователь может воспользоваться именно той машиной, для которой нужна программа имеется.

Вычислительные сети позволяют решать качественно новые задачи, такие, например, как создание распределенных по обширной территории систем информационного обслуживания научных и конструкторских работ, автоматизированных обучающих систем или систем резервирования билетов на различные виды транспорта.

Обобщенная структурная схема ВСт приведена на рис. 16.1. Основу ВСт составляют крупные ЭВМ (вычислительные центры коллективного пользования — *ВЦКП*), объединяемые сетью передачи данных. Эти ЭВМ, называемые *главными вычислительными машинами* (ГВМ), выполняют основные функции по выполнению программ пользователей, сбору, хранению, выдаче информации. Сеть передачи данных (СПД)

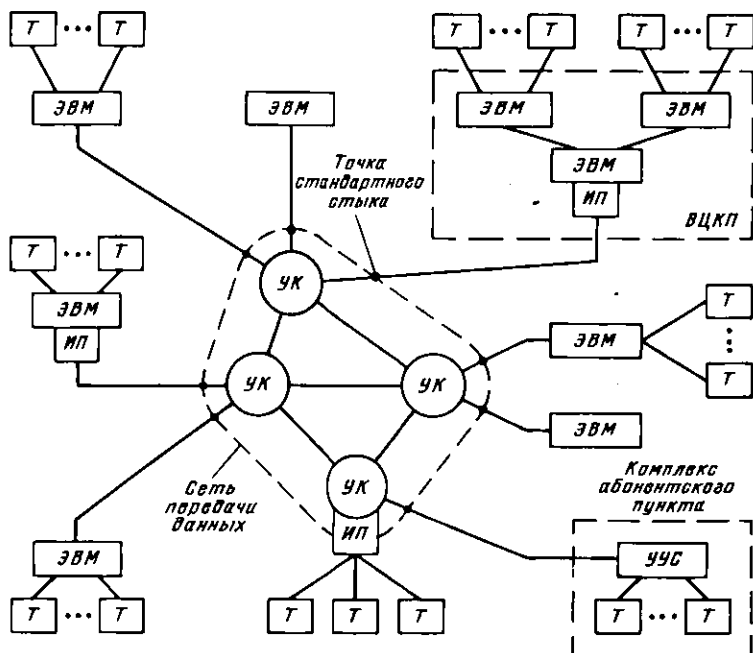


Рис. 16.1. Структурная схема вычислительной сети

образуют каналы связи и узлы (центры) коммутации *УК*, в которых связные процессоры (СП) управляют выбором маршрутов передачи данных в сети.

Главные вычислительные машины подсоединяются к сети (точнее, к СП) непосредственно через *точки стандартного стыка*, если обеспечена совместимость по физическим сигналам (управляющим и информационным) и форматам информации между ГВМ и СП, или с помощью *интерфейсных процессоров (ИП)*. В последнем случае интерфейсный процессор (обычно это микро-ЭВМ) может выполнять и некоторые функции по предварительной обработке данных (преобразование кодов передаваемых (принимаемых) данных, контроль правильности полученных сообщений и др.), разгружая от этих операций ГВМ.

Терминалы пользователей подключаются либо к ГВМ (ВЦКП), либо непосредственно к СП. Для подключения к сети группы терминалов, удаленных от ГВМ и СП, используются терминальные процессоры (терминальные концентраторы), называемые часто *абонентскими пунктами (АП)*. В этом случае

отпадает необходимость выделения каждому терминалу отдельного канала связи.

В качестве терминалов используются телетайпы, электрифицированные пишущие машинки, дисплеи, графопостроители и другие устройства ввода-вывода, а также их комбинации. Оборудование терминала может включать в себя микро-ЭВМ, при этом терминал будет выполнять некоторые функции по вспомогательной обработке информации, что служит основанием называть его «интеллектуальным терминалом».

Интегрируемые в вычислительную сеть ГВМ, обычно сохраняющие в основном функции по отношению к своим вычислительным центрам, берут на себя часть общесетевой нагрузки по сбору, обработке и передаче данных.

Административное управление в ВСт включает в себя планирование и учет работы отдельных машин сети, анализ и учет работы сети передачи данных, проведение измерений на сети («сетиметрия») и т. п. Эти функции возлагают на одну из ГВМ сети, которую называют административным комплексом.

За рубежом и в Советском Союзе создан ряд крупных ВСт, ведется интенсивная разработка новых сетей. Некоторые из существующих и разрабатываемых ВСт охватывают обширные территории, имеют общегосударственный и даже межгосударственный масштаб функционирования.

Одной из крупнейших является вычислительная сеть АРПА (США), объединяющая свыше 300 программно-несовместимых ЭВМ разной производительности. Вычислительная сеть АРПА охватывает значительную часть территории США и через спутниковую связь связана с несколькими вычислительными центрами в Европе.

В Советском Союзе функционирует ряд отраслевых вычислительных сетей, экспериментальная ВСт Академии наук СССР, создана межгосударственная, охватывающая страны СЭВ сеть научно-технической информации.

Протоколы вычислительных сетей. Управление таким сложным процессом, как передача данных в разветвленной ВСт, в котором участвует многочисленная и разнообразная аппаратура, требует формализации и стандартизации процедур установления и разъединения соединений, передачи информации, контроля правильности передачи, исправления ошибок, выделения и освобождения ресурсов ЭВМ и сети передачи данных. Эти задачи решаются с помощью протоколов, регламентирующих стандартизованные процедуры взаимодействия элементов сети при установлении связи и передаче данных. Реализацией протокольных процедур управляют специальные программы или аппаратные средства.

16.2. КЛАССИФИКАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

Классификацию ВСт можно производить по ряду признаков.

По функциональному назначению различают сети *информационные*, представляющие пользователю в основном информационное обслуживание (сети научно-технической информации, здравоохранения, резервирования билетов на транспорте и т. п.), *вычислительные*, выполняющие главным образом решение задач с обменом данными и программами между ЭВМ сети, и смешанные *информационно-вычислительные*.

По размещению информации в сети разделяют сети с централизованным банком данных, формируемым в одном из узлов сети, и с распределенным банком данных, состоящим из отдельных локальных банков, расположенных в узлах сети.

По степени территориальной рассредоточенности можно выделить крупномасштабные или *глобальные ВСт*, охватывающие территорию страны, нескольких стран с расстояниями между отдельными узлами сети, измеряемыми тысячами километров, *региональные сети*, охватывающие определенные территориальные регионы — город, район, область и т. п., *локальные вычислительные сети* (ЛВСт) с максимальным расстоянием между узлами сети не более нескольких километров.

По числу ГВМ, в которых производится сбор, хранение и обработка информации, следует различать ВСт с несколькими ГВМ и с одной. К последним относятся упоминавшиеся в § 13.2 ВСт с телеобработкой, которые представляют собой комплексы, состоящие из вычислительной машины и удаленных абонентских пунктов, связанных с помощью каналов и аппаратуры передачи данных.

По типу используемых ЭВМ выделяют *однородные сети*, содержащие программно-совместимые машины, и *неоднородные*, если машины сети программно-несовместимы. На практике сети часто являются неоднородными.

По методу передачи данных различают вычислительные *сети с коммутацией каналов*, *с коммутацией сообщений*, *с коммутацией пакетов* и *со смешанной коммутацией*. Для современных ВСт характерно использование коммутации пакетов (см. § 16.4).

Признаком различия сетей является тип используемых в ней протоколов обмена информацией.

Важным признаком классификации ВСт является их *топология*. Топологическая структура ВСт оказывает значительное влияние на ее пропускную способность, устойчивость сети к от-

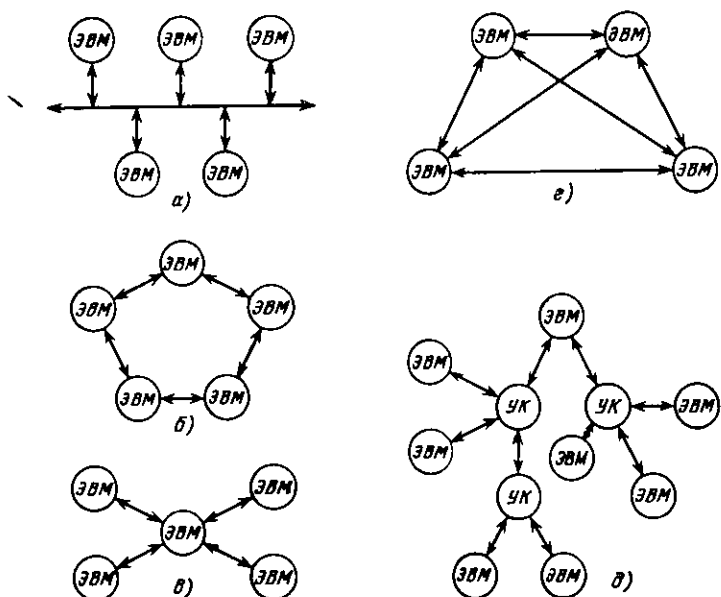


Рис. 16.2. Виды топологий вычислительных сетей:

а — одиная многоточечная линия; б — петлевая (кольцевая) сеть; в — звездообразная сеть; г — полностью связанная сеть; д — древовидная сеть

казам ее оборудования, на логические возможности и стоимость сети. В настоящее время наблюдается большое разнообразие в топологических структурах вычислительных сетей.

На рис. 16.2 представлены различные варианты топологий ВСт. Топология крупных вычислительных сетей может представлять собой комбинацию нескольких топологических решений.

16.3. ЭТАЛОННАЯ ЛОГИЧЕСКАЯ МОДЕЛЬ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ И ИЕРАРХИЯ ПРОТОКОЛОВ

Сложность и разнообразие реальных физических структур вычислительных сетей, разнотипность используемых в вычислительных сетях ЭВМ и другой аппаратуры, необходимость упорядочения разработки программных и аппаратурных средств сетевого комплексирования вычислительных установок делают целесообразным введение обобщенной логической

структуры вычислительной сети, с которой могут быть соотнесены физические структуры конкретных сетей.

Такая обобщенная логическая структура предложена Международной организацией стандартов в виде *эталонной семиуровневой логической модели открытых систем* [111]. Системы называются открытыми друг для друга в том смысле, что, несмотря на их технические и логические различия, они могут взаимодействовать с помощью определенных процедур.

В вычислительной сети связь в действительности устанавливается не между отдельными машинами, а между прикладными программами, или, более точно, между *прикладными вычислительными процессами*, протекающими в машинах. При этом под прикладным процессом следует понимать прикладную программу вместе с ее наборами данных и выделенными ей ресурсами машины. Связь эта реализуется с помощью целого ряда специальных процессов и обслуживающих их аппаратурных и программных средств, например процессов установления маршрута передачи сообщения, управления передачей, установления соединения, интерпретации принятого сообщения и др. Это обстоятельство закономерно приводит к рассмотрению иерархии функций, реализуемых в оборудовании сети.

В рассматриваемой модели под системой понимается иерархически упорядоченная совокупность функций, обеспечивающая выполнение некоторым аппаратурным комплексом, включающим в себя одну или несколько ЭВМ, приписанного ему в сети функционального назначения.

Связи между системами в логической модели называются физическими соединениями.

Для упорядочения функций системы вводятся функциональные уровни (рис. 16.3).

В общем случае система имеет следующие семь функциональных уровней, перечисленных ниже в порядке возрастания уровня иерархии:

1. Обеспечение электрических, механических и функциональных характеристик подключения к физическим каналам связи, преобразование сигналов — *физический уровень*.

2. Управление каналом передачи данных, установление, поддержание и разъединение каналов (соединений) — *канальный уровень* или *уровень управления каналом*.

3. Маршрутизация, коммутация и адресация информации, управление потоками данных — *сетевой уровень*.

4. Управление передачей данных (без обработки их в промежуточных узлах) от системы-источника к системе-адресату — *транспортный уровень*.

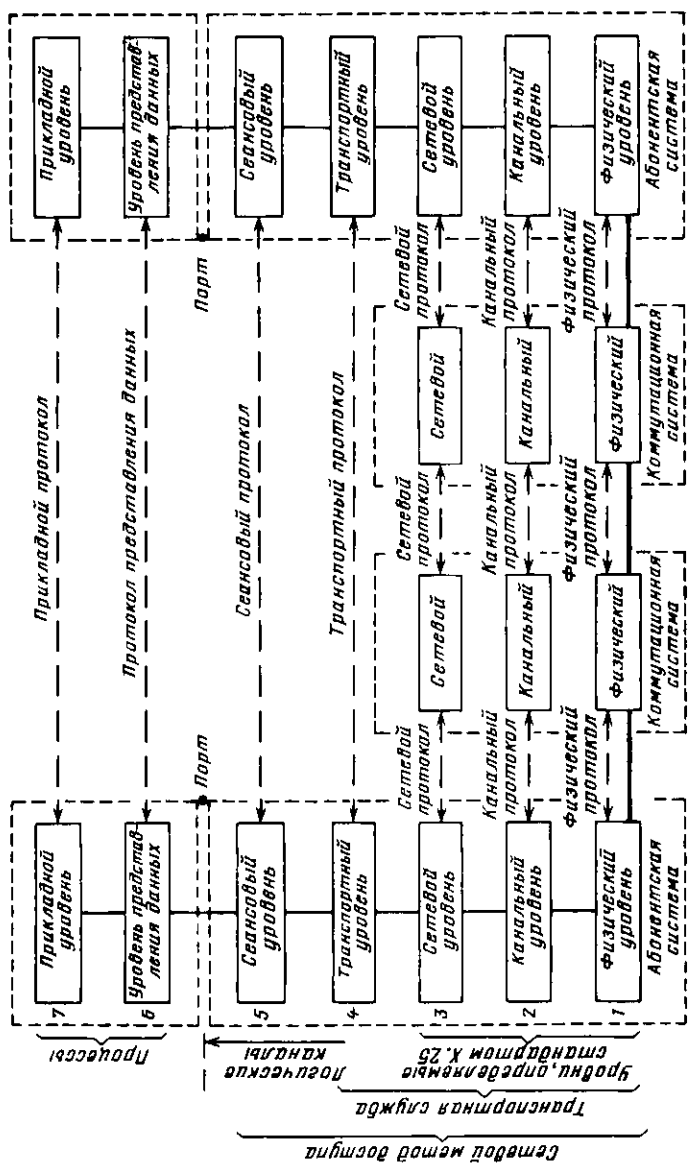


Рис. 16.3. Семьюровневая логическая модель вычислительной сети. Иерархия протоколов сети

5. Организация и проведение сеансов связи между прикладными процессами — *сеансовый уровень*.

6. Интерпретация и преобразование передаваемых между процессами данных к виду, удобному для прикладных процессов, — *уровень представления данных*.

7. Выполнение прикладных программ, управление терминалами, административное управление сетью — *прикладной уровень*.

Первые четыре уровня образуют *транспортную службу* вычислительной сети, освобождающую более высокие уровни от забот по организации передачи данных. Части вычислительной сети, реализующей транспортную службу, в логической структуре ВСт соответствует *транспортная сеть*.

Уровню могут быть поставлены в соответствие некоторые процессы, аппаратные и программные средства (*объекты уровня*).

Каждый уровень обслуживает соседний старший уровень. Связь между объектами соседних уровней одной системы регламентируется *межуровневым интерфейсом*. Организация взаимодействия между одинаковыми уровнями различных систем определяется соответствующим *протоколом*.

Введенная выше иерархия уровней функций логической системы определяет соответствующую иерархию протоколов и соответственно протокольных программ, действующих в вычислительной сети, как это показано на рис. 16.3.

Два старших уровня (6 и 7) соответствуют *процессам* (процессам представления и преобразования данных, выполнения прикладных программ, административного управления сетью). Остальные уровни в совокупности определяют *сетевой метод доступа* к указанным процессам. Точки в процессах, через которые осуществляется эта связь, получили название *портов*.

Управление взаимодействием различных объектов сети осуществляется с помощью сетевого программного обеспечения — *транспортных станций*, представляющих собой специальные программы, располагаемые в ГВМ, а также *сетевых программ* связанных процессоров. Транспортная станция обеспечивает мультиплексирование независимых потоков данных от многих абонентов. Она может быть реализована как часть операционной системы ЭВМ либо как независимая системная задача.

Нижние три уровня и соответственно физический, канальный и сетевой протоколы охватывает международный стандарт — «протоколы X.25», предусматривающий передачу данных путем коммутации пакетов, при этом разборка сообщения на пакеты при передаче и сборка пакетов в сообщение при приеме производится в ГВМ или в интерфейсных процессорах.

Протоколы уровней 4–7 («сквозные») регламентируют процедуры «сквозной» связи между абонентами сети.

16.4. МЕТОДЫ ПЕРЕДАЧИ ДАННЫХ ПО КАНАЛАМ СВЯЗИ. КОММУТАЦИЯ КАНАЛОВ, СООБЩЕНИЙ, ПАКЕТОВ

Каналом связи называют физическую среду и аппаратные средства, осуществляющие передачу информации от одного узла коммутации к другому либо к абоненту связи. Под физической средой понимается пространство или материал, обеспечивающие распространение сигналов: проводная воздушная или кабельная линия, скрученная пара проводов, коаксиальный кабель, световодная (стекловолоконная) линия, эфир [103].

В технике связи используются телеграфные и телефонные каналы связи. По телеграфным каналам информация передается в дискретной форме, что облегчает их сопряжение с ЭВМ. Однако скорость передачи информации по телеграфным каналам невелика и составляет всего 50–200 бит/с. По телефонным каналам информация (речь) передается в аналоговой форме, и поэтому значительно усложняется сопряжение этих каналов с ЭВМ. Однако эти каналы позволяют при использовании соответствующей дополнительной аппаратуры производить передачу двоично-кодированной информации с большей скоростью (до 96 000 бит/с).

В настоящее время в технике вычислительных сетей и телеобработки для передачи данных используют главным образом аналоговые каналы связи. При этом требуется специальная аппаратура передачи данных (АПД), осуществляющая необходимые для сопряжения аналоговых каналов с ЭВМ преобразования форм представления передаваемой информации, а также некоторые другие операции.

Канал связи, оснащенный аппаратурой для передачи дискретной информации, называется *каналом передачи данных*.

По возможным направлениям передачи информации различают каналы:

- 1) *симплексные*, позволяющие передавать данные только в одном направлении;
- 2) *полудуплексные*, передающие данные в обоих направлениях, но не одновременно;
- 3) *дуплексные*, позволяющие передавать одновременно данные в обоих направлениях.

Каналы передачи данных в зависимости от скорости передачи делятся на низкоскоростные со скоростью передачи 50, 75, 100, 200 бит/с (по телеграфным линиям связи), среднескоростные со скоростью передачи 600, 1200, 2400, 4800 и 9600 бит/с (по телефонным каналам), высокоскоростные со скоростью 24 000, 48 000, 96 000 бит/с (по ВЧ-каналам).

Передача дискретной (двоично-кодированной) информации по аналоговым каналам связи осуществляется путем *модуляции* в передающем пункте колебаний несущей частоты (значение частоты зависит

Рис. 16.4. Виды модуляции:

а — амплитудная; б — частотная; в — фазовая

от скорости передачи данных; наименьшая из употребляемых частот 1500 Гц) двоичными сигналами передаваемого кода с последующей демодуляцией — восстановлением первоначальной формы сообщения в приемном пункте. Операции модуляции и демодуляции выполняются в устройствах, называемых модемами.

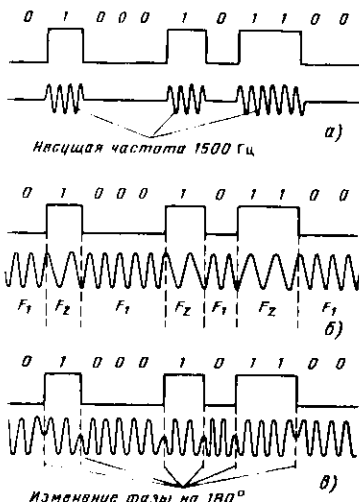
Применяются три основных вида модуляции: амплитудная, частотная и фазовая, которые поясняются на рис. 16.4.

Сама передача данных в канале связи может быть асинхронной и синхронной (рис. 16.5). При асинхронной передаче символы передаются в свободном темпе независимо друг от друга, причем каждый символ передается со своими сигналами «Старт» и «Стоп», указывающими на начало и конец передачи символа.

При синхронной передаче блок символов передается непрерывно в принудительном темпе, а синхронизация передающего и принимающего устройств достигается посылкой специальных кодовых комбинаций перед каждым блоком данных.

Асинхронная передача позволяет передавать информацию с устройств, которые выдают ее асинхронно во времени (например, ручные клавиатуры печатающих машинок, дисплеев и т. д.). Однако скорость передачи информации при асинхронном методе низка, так как велика ее избыточность из-за большого числа служебных сигналов. Поэтому асинхронная передача используется только при скорости передачи в телефонном канале до 200 бит/с. Синхронный метод обеспечивает большую скорость передачи данных из-за меньшей избыточности информации, но требует более сложной аппаратуры.

Помехи в каналах связи могут вызывать ошибки при передаче информации. Достоверность передачи данных оценивается отношением числа ошибочно принятых символов к общему числу переданных. Для телефонных коммутируемых каналов достоверность передачи составляет 10^{-3} . Такое низкое значение достоверности передачи заставляет в ряде случаев применять специальные методы (контроль по четности, контрольные суммы, циклические коды) и средства контроля правильности передачи, автоматического повторения передачи при появлении ошибки или автоматической коррекции.



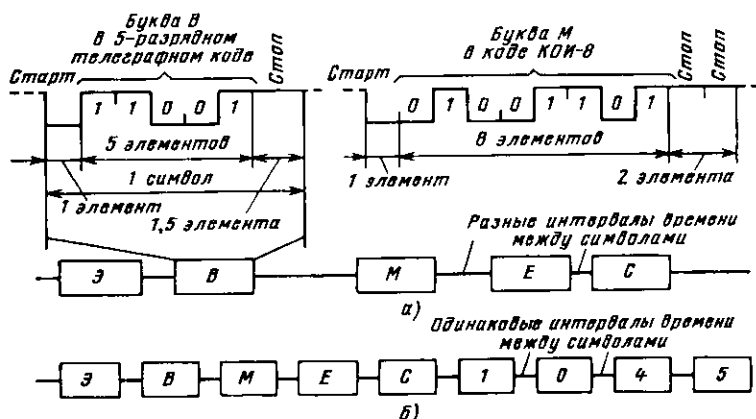


Рис. 16.5. Асинхронная (а) и синхронная (б) передачи данных

Большую часть стоимости вычислительных сетей и крупных систем телеобработки составляют каналы связи. Это делает необходимым применение уплотнения и концентрации каналов связи.

Уплотнение каналов связи представляет собой статическое разделение канала, при котором определенные полосы частот или периоды времени в уплотняемом канале выделяются в фиксированном, заранее заданном порядке для использования в качестве отдельных каналов.

Концентрация каналов связи в отличие от уплотнения является динамической процедурой распределения меньшего числа более скоростных выходных каналов концентратора в соответствии с имеющимися запросами между большим числом менее скоростных входных каналов.

Передача данных в ВСт производится следующими методами: коммутацией каналов, коммутацией сообщений и коммутацией пакетов (рис. 16.6) [47]. Выбор метода зависит от назначения сети и характера передаваемых данных.

Коммутация каналов. При этом методе в сети передачи данных устанавливается физическое соединение между пунктами отправления и назначения (источником и адресатом) путем образования составного канала из последовательно соединенных отдельных канальных участков. Установление связи между источником и адресатом производится путем послыки пунктом отправления сигнализирующего сообщения, которое, перемещаясь по сети передачи данных от одного узла коммутации каналов к другому и занимая пройденные каналы, прокладывает путь от источника к пункту назначения. Этот путь (составной канал) состоит из физических каналов, имеющих одну и ту же скорость передачи данных. Об установлении физического соединения из пункта назначения в источник посылается сигнал обратной связи. Затем из источника передается сообщение по установленному пути с одновременным использованием всех образующих его каналов, которые

оказываются недоступными для других передач, пока источник их не освободит.

При коммутации сообщений и пакетов информация передается с запоминанием в промежуточных узлах сети передачи данных без установления физического соединения между пунктами отправления и назначения. Между ними устанавливается *виртуальное или логическое соединение*.

Коммутация сообщений. При этом методе физическое соединение устанавливается только между соседними узлами сети (называемыми центрами или узлами коммутации сообщений) и только на время передачи сообщения. Каждое сообщение снабжается заголовком и транспортируется по сети как единое целое. Поступившее в узел сообщение запоминается в его буферном ЗУ и в подходящий момент, когда освободится соответствующий канал связи, передается в следующий соседний узел. Сообщение как бы прыгает от одного узла к другому, занимая в каждый момент передачи только канал между соседними узлами. При этом виртуальный канал между источником и адресом может состоять из физических каналов с разной скоростью передачи данных. Коммутация сообщений по сравнению с коммутацией каналов позволяет ценой усложнения аппаратуры узла коммутации уменьшить задержку при передаче данных и повысить общую пропускную способность сети передачи данных.

Коммутация пакетов является развитием метода коммутации сообщений. Она позволяет добиться дальнейшего увеличения пропускной способности сети, скорости и надежности передачи данных.

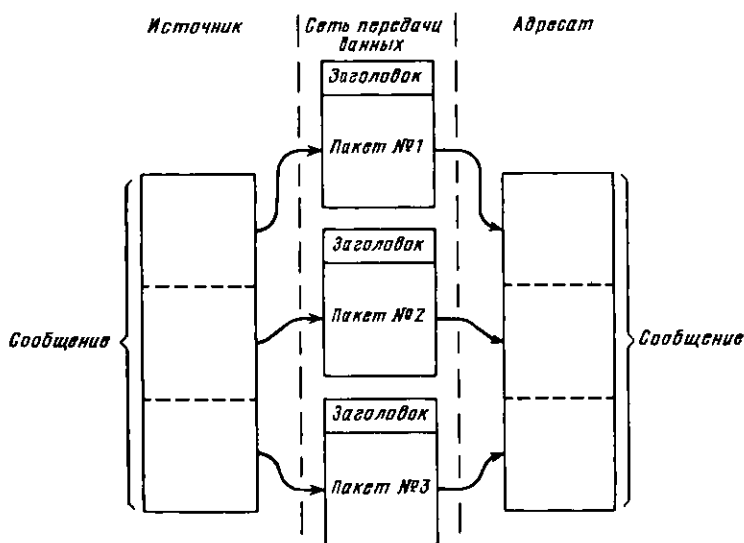


Рис. 16.7. Пакетирование сообщений

Поступающее от абонента сообщение подвергается в интерфейсных процессорах пакетированию — разбивается на пакеты, имеющие фиксированную длину, например 1 Кбайт (рис. 16.7). Пакеты метятся служебной информацией — заголовком, указывающим адрес пункта отправления, адрес пункта назначения и номер пакета в сообщении. Пакеты транспортируются в сети как независимые сообщения и поступают в узел коммутации пакетов, где накапливаются в буферах каналов связи. Затем они передаются в выходной буфер, где попеременно накапливаются пакеты различных сообщений, которые выдаются на скоростной канал связи для передачи в соседний УК. В пункте назначения интерфейсный процессор формирует из пакетов исходное сообщение.

Важным достоинством коммутации пакетов является возможность одновременной передачи пакетов одного и того же сообщения разными маршрутами, что уменьшает время и увеличивает надежность передачи сообщения.

При передаче короткими пакетами уменьшаются вероятность появления ошибок и время занятости каналов повторными передачами.

Как видно из рис. 16.6, при коммутации каналов имеется наибольшая задержка в передаче информации, что определяется сравнительно большим временем установления связи из-за ожидания освобождения промежуточных каналов. Монополизация источником на все время связи каналов, образующих путь от него к адресату, снижает общую пропускную способность сети передачи данных. Однако коммутация каналов обладает важной особенностью: если связь источник — адресат установлена, то при передаче сохраняются неизменными временные соотношения между элементами передаваемой информации (*временная прозрачность*), что имеет значение для систем, работающих в реальном масштабе времени. Если в таких системах недопустимы задержки с установлением связи, то используют сравнительно дорогие некоммутируемые (постоянно выделенные) каналы связи.

Метод коммутации пакетов по сравнению с другими методами обеспечивает наименьшую задержку при передаче данных и наибольшую пропускную способность сети передачи данных, особенно заметную при передаче коротких сообщений, характерных для диалогового режима. Использование коммутации пакетов способствует повышению надежности и живучести сети вследствие того, что облегчается адаптация управления передачей данных к отказам и перегрузкам отдельных участков. Поэтому в настоящее время коммутация пакетов является основным методом передачи данных в ВСт, но во многих случаях этот метод оказывается непригодным для систем, работающих в реальном времени.

Для выбора маршрута пакета в сети могут использоваться статические процедуры, основанные на наличии таблиц предпочтительных маршрутов в УК, и различные варианты адаптивной процедуры. В отказоустойчивых сетях часто используется лавинный метод, согласно которому пакет посылается из каждого исходного узла во все соседние. Если пакет попадает в узел, где он уже был, то пакет уничтожается.

16.5. УПРАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ СЕТЬЮ ПО ПРОТОКОЛУ X.25

Рассмотрим процедуры управления вычислительной сетью, построенной в соответствии с протоколом (а точнее, с протоколами) рекомендации X.25 Международного консультативного комитета по телеграфии и телефонии (МККТТ) [111]. Протокол X.25 реализован во многих ВСт, в том числе TELENET (США), TRANSPAC (Франция), DATAPAC (Канада), EURO-NET (Западная Европа). Он определяет стандартный интерфейс для соединения оконечного оборудования (ГВМ или терминала) с сетью передачи данных, т. е. интерфейс «пользователь — сеть».

В ВСт используют два характерных режима передачи пакетов: режим виртуальных каналов и режим датаграмм.

Режим виртуальных каналов предполагает, что передаче сообщения предшествует прокладка (организация) *виртуального канала (логического соединения)*, по которому затем строго в порядке номеров передаются пакеты сообщения. Такой режим облегчает сборку пакетов в сообщение на приемном конце.

Режим датаграмм допускает независимое перемещение по сети пакетов сообщения (называемых в этом случае *датаграммами*) и не требует предварительного установления логического соединения. Этот режим более сложен в реализации, так как усложняется сборка пакетов в сообщение у адресата, однако позволяет достигнуть большей надежности и живучести сети передачи данных.

Вычислительные сети с протоколом X.25 в принципе работают в режиме виртуальных каналов. Однако в заголовке пакета может быть установлен специальный индикатор, указывающий на то, что данный пакет является одиночным (датаграммой), содержит все подлежащие передаче данные, и поэтому для него нет необходимости устанавливать виртуальный канал между отправителем и адресатом. Для простоты изложения далее предполагается, что абоненты (процессы пользователей) находятся в ГВМ, хотя все, что будет сказано, в равной степени относится и к случаю, когда в качестве оконечного оборудования рассматривается терминал.

Рекомендация X.25 включает в себя протоколы трех нижних уровней рассмотренной в § 16.3 эталонной логической семиуровневой модели вычислительной сети: на физическом уровне — стандартный физический интерфейс X.21, на канальном уровне — процедуру управления каналом *LAPB* [103] и на сете-

вом уровне — протокол X.25/3 обмена пакетами между оконечным оборудованием и сетью передачи данных.

Пусть в ходе выполнения процесса пользователя в ГВМ понадобилось обеспечить его взаимодействие через сеть передачи данных с удаленным процессом в другой ГВМ, например передать массив данных. Для этого процесс регистрируется в транспортной станции своей ГВМ и становится абонентом сети. После этого между соответствующими процессами должно быть установлено логическое соединение (ЛС) — виртуальный канал для передачи данных.

Протокол X.25/3 в зависимости от условий взаимодействия абонентов регулирует создание временных ЛС на один сеанс связи и постоянных ЛС, которые нет необходимости устанавливать заново и разрывать при каждом сеансе связи. ГВМ могут одновременно поддерживать несколько временных и (или) постоянных ЛС.

На рис. 16.8 показаны три фазы взаимодействия ГВМ — установки ЛС, передачи данных, разъединения ЛС.

Фаза установки временного ЛС протекает следующим образом. Запрашивающая соединение ГВМ передает в сеть передачи данных управляющий пакет *запрос соединения*, содержащий адреса обеих ГВМ, процессы в которых требуется соединить. Кроме того, в пакете могут указываться условия обслуживания: необходимые дополнительные средства (программы, устройства) и до 16 байт данных абонента-отправителя. Пакет *запрос соединения* на выходе из сети передачи данных преобразуется в пакет *входящий запрос соединения* для того, чтобы различать в одном физическом канале ГВМ-СП управляющие пакеты, направленные в разные стороны.

Если ГВМ-адресат согласен на установку ЛС, он передает в сеть связи пакет *согласие на соединение*, который на выходе из сети передачи данных преобразуется в пакет *соединение установлено*. После этого наступает фаза передачи данных.

Фаза разъединения ЛС может иметь место как после фазы передачи данных, так и во время фазы установки ЛС, когда вызываемая ГВМ не соглашается на установку ЛС и вместо пакета *согласие на соединение* направляет в сеть передачи данных пакет *запрос разъединения*.

После окончания сеанса связи или при нарушении порядка передачи пакетов между ГВМ, или когда сеть передачи данных испытывает перегрузку любая из ГВМ может начать фазу разъединения передачей пакета *запрос разъединения*. На выходе из сети связи этот пакет преобразуется в *индикацию разъединения*. Запрашиваемая ГВМ должна ответить пакетом *подтверждение разъединения*, получение которого переводит

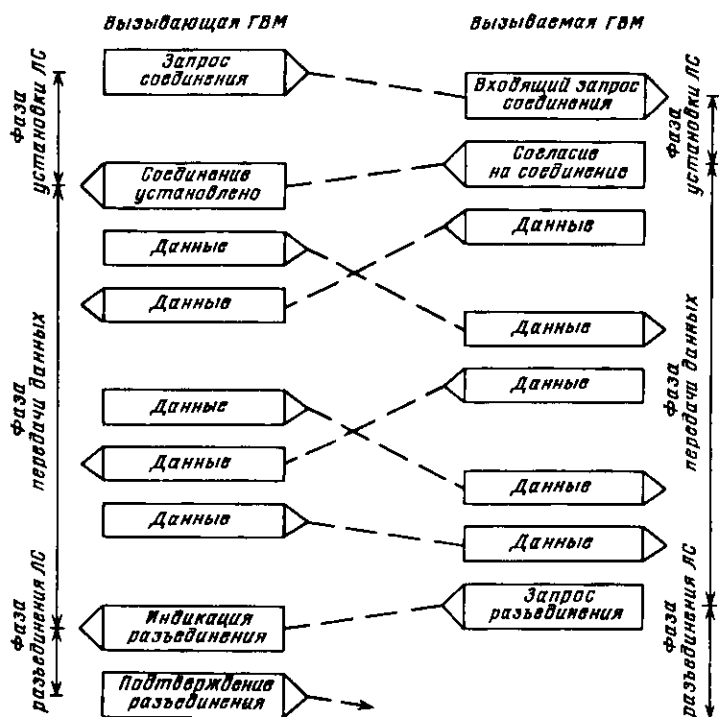


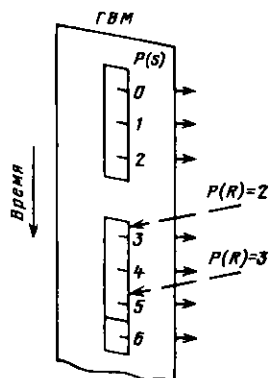
Рис. 16.8. Фазы взаимодействия ГВМ в соответствии с протоколом X.25/3

ГВМ в состояние *готовность*: сетевая служба готова создать новое временное ЛС.

Пакеты с данными могут передаваться в любое время через постоянное ЛС или после установки временного ЛС. Абонент в ГВМ-отправителе выдает сообщение с данными для передачи удаленному абоненту в виде отдельных блоков. Программа управления передачей, относящаяся к четвертому (транспортному) уровню логической модели сети, добавляет к блоку заголовков, преобразуя его в фрагмент, и передает программе, реализующий протокол X.25/3, которая в свою очередь формирует пакет, добавляя к фрагменту заголовок пакета (заголовок X.25/3).

Полученный в результате пакет может быть одиночным (пакет датаграмма) или частью многопакетного сообщения (пакет данные). Хотя протокол X.25/3 устанавливает рекомендуемый

Рис. 16.9. Передача с использованием «окна» ($w=3$)



максимальный размер поля данных в пакете, равный 128 байт, он разрешает администрации сети вводить и другие максимальные размеры в пределах 16–1024 байт, являющиеся целой степенью числа 2.

Пакеты данные и датаграмма, передаваемые или принимаемые в ГВМ по некоторому ЛС, циклически нумеруются от 0 до 7 (по модулю 8) или от 0 до 127 (по модулю 128). В первом случае заголовок пакета занимает 3 байта, во втором — 4 байта.

В ГВМ в заголовке пакета помещаются два номера: присвоенный данному пакету порядковый номер передачи $P(s)$ и порядковый номер $P(R)$ пакета, последовательно принятого данной ГВМ (номер приема пакета). Порядковый номер передачи необходим для обеспечения последовательной транспортировки пакетов данные по виртуальному каналу, обнаружения потерь пакетов и для управления интенсивностью поступления пакетов в сеть передачи данных.

Номер приема пакета используется как квитанция, подтверждающая прием пакетов адресатом и разрешающая передачу последующих пакетов в сеть передачи данных.

Если имеется поток пакетов только в одном направлении, то разрешения на передачу последующих пакетов передаются в виде специальных управляющих пакетов *готов к приему*. Если ГВМ (или СП) временно не может принимать пакеты с данными по некоторому ЛС, то они передают управляющий пакет *не готов к приему*.

В заголовке пакета данные имеется также специальный разряд, принимающий значение 1, когда пакет является промежуточным. Если значение этого разряда равно 0, то пакет является последним в данной последовательности.

Для управления потоком пакетов в протоколе X.25/3 реализован так называемый механизм *окна*. Согласно ему фиксируется максимальное число w последовательно пронумерованных пакетов, которые разрешается передать без получения подтверждения об их приеме от адресата. Величина w называется размером окна. При нумерации пакетов по модулю

8 размер окна может доходить до 7, а при нумерации по модулю 128 — до 127.

Граница окна изменяется (окно «продвигается») при получении номера приема $P(R)$, большего (по модулю 8 или 128), чем последний принятый номер $P(R)$ (рис. 16.9). Окно большего размера с нумерацией пакетов по модулю 128 используется в сетях, в которых задержки при транспортировании пакетов велики.

После того как программа, реализующая протокол X.25/3, сформировала пакет, он передается программе, реализующей протокол *LAPB* канального уровня. В соответствии с этим протоколом пакет упаковывается в кадр: к пакету добавляют заголовок и «концевик» кадра. Затем этот кадр согласно протоколу X.21 физического уровня передается по каналу связи в следующий узел сети, где пакет из кадра передается программе, реализующей протокол X.25/3, которая считывает адрес назначения пакета и определяет канал для его дальнейшей передачи.

Когда пакет достигает ГВМ-адресата, из него извлекается фрагмент, который передается программе управления передачей, и данные поступают к абоненту-адресату.

16.6. ЛОКАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ

Локальной вычислительной сетью (ЛВС) называют сеть, все элементы которой — вычислительные машины (как правило, малые или микро-ЭВМ), терминалы, связанная аппаратура — располагаются на сравнительно небольшой территории, например в радиусе 3–10 км.

Такая локальная сеть обычно предназначается для сбора, передачи, рассредоточенной и распределенной обработки информации в пределах одного предприятия или организации, часто специализируется на выполнении определенных функций в соответствии с профилем деятельности предприятия и отдельных его подразделений. Во многих случаях ЛВС, обслуживая АСУ организационного типа на предприятии, связана, с одной стороны, с автоматизированными системами управления отдельными технологическими процессами и установками, а с другой стороны, с крупной региональной вычислительной сетью.

Структура ЛВС отображает в определенных пределах структуру обслуживаемой организации, а поэтому часто имеет иерархическое построение. В этом случае отдельные фрагменты ЛВС (их можно рассматривать как ее подсети) специализируются на обслуживании отдельных подразделений и форм ра-

бот — автоматизация проектирования, автоматизация испытаний, автоматизация технологических процессов, информационное обслуживание, планово-экономическое и оперативное управление производством и т. п.

Наличие локальной сети позволяет специализировать и приближать обработку информации к местам, где она зарождается, а результаты обработки — к лабораториям, КБ, цехам, отделам и т. д., освобождая от ряда работ вычислительный центр организации и обеспечивая при этом быстрый обмен информацией между отдельными подразделениями и между ними и главным ВЦ.

Эффективность локальной сети основана на том, что, как показывает практика, подавляющая часть информации, с которой имеют дело на предприятии (в организации), циркулирует между отдельными его (ее) службами или даже внутри подразделений и таким образом охватывается локальной сетью.

Значение создания локальных сетей в последнее время сильно возросло благодаря принимающему широкий масштаб распространению персональных микро-ЭВМ, для эффективного применения которых на предприятии или организации во многих случаях необходимо иметь возможность обмена информацией между отдельными персональными ЭВМ (работниками предприятия) и между ними и главным вычислительным центром.

Наличие локальной сети позволяет упростить и удешевить персональные вычислительные машины вследствие коллективного использования ими через локальную сеть (в режиме разделения времени) наиболее дорогих ресурсов, таких, как дисковая память большей емкости и печатающие устройства.

При построении ЛВС важно найти сравнительно простые и дешевые решения для реализации связи между объединяемыми сетью устройствами.

В ЛВС применяется, главным образом, прямая передача дискретной информации, при которой цифровые сигналы (0, 1) прямо, без модуляции несущей частоты (используемой при широкополосной передаче по телефонным каналам) поступают в физический канал (соединительный кабель).

Тип физического канала зависит от требований к скорости передачи информации в сети. При скорости передачи от нескольких сотен килобит в секунду до 1 Мбит/с используют витые пары проводов и (при расстояниях до 500 м) ленточные кабели, при скорости свыше 1 Мбит/с — коаксиальный или световодный кабель.

На рис. 16.10 представлены типичные структуры ЛВС с шиной и кольцевой организациями связи. Абонентские

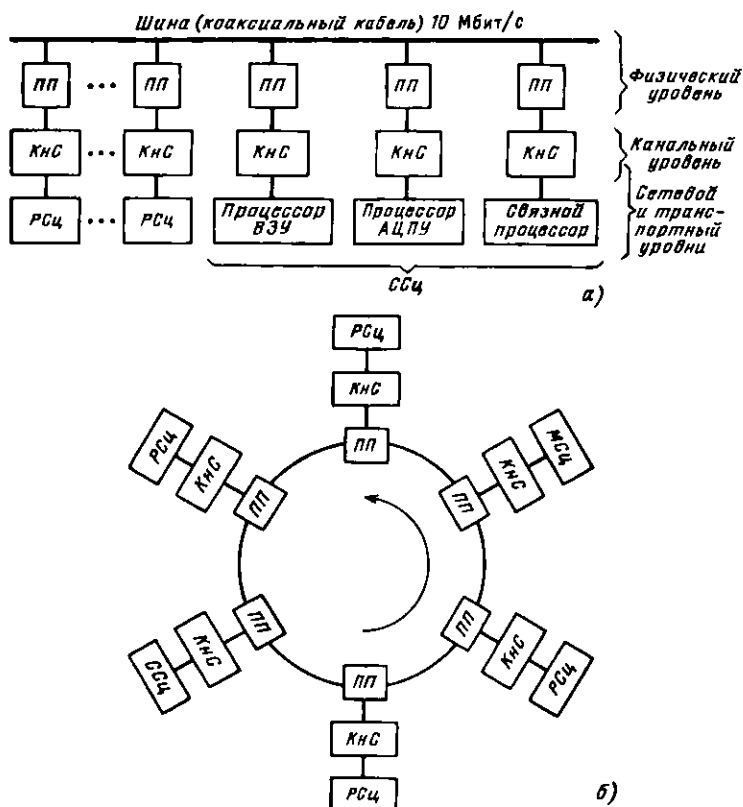


Рис. 16.10. Структура локальных шинной (а) и кольцевой (б) вычислительных сетей:

ПП – приемник-передатчик; *МСц* – мониторная станция; *КНС* – контроллер сети; *РСц* – рабочая станция; *ССц* – сервисная станция

устройства сети (станции), к которым относятся микро-ЭВМ, терминалы, внешние ЗУ, печатающие устройства и некоторые другие, подключаются к шине (магистрالي) или кольцу при помощи приемопередатчиков *ПП* и контроллеров сети *КНС*, иногда называемых блоками доступа [106]. Контроллеры производят прием сигналов из канала, выдачу их в канал, преобразования форматов данных для согласования реализованной в локальной сети последовательной передачи (последовательного интерфейса) с внутренним параллельным интерфейсом

устройств (станций), подключенных к сети, а также в зависимости от особенностей ЛВС_т выполняют некоторые другие функции.

Шинная ЛВС_т (рис. 16.10, а). Шина ЛВС_т существенно отличается от внутренней шины в интерфейсе ввода-вывода малых и микро-ЭВМ. В отличие от внутренней шины ввода-вывода, по которой данные и управляющие сигналы передаются по нескольким параллельным линиям и управление передачей производится сигналами синхронизации, идентификации, стробирования и квитирования, которыми обмениваются передающая и приемная стороны при передаче (приеме) каждого элемента информации (например, байта), в шине ЛВС_т передача производится последовательным кодом и синхронизируется самой передаваемой информацией, т. е. так, как производится передача по каналам связи. В шинной ЛВС_т с коаксиальным кабелем скорость передачи достигает 10 Мбит/с.

Станция, получив в свое распоряжение шину, передает по шине снабженное адресом сообщение или пакет (кадр), который практически одновременно поступает на все станции (или контроллеры) сети, распознается одной из них (а при циркулярной передаче — несколькими станциями) как ей адресованное и принимается ею. Параллельная во времени передача информации всем абонентам сети является важной особенностью ЛВС_т рассматриваемого типа.

Управление доступом станции к шине может быть централизованным и осуществляться специальной мониторной или супервизорной станцией либо децентрализованным — распределенным между всеми станциями сети.

В шинных ЛВС_т используются два метода доступа к шине: детерминированный и случайный. При детерминированном доступе и централизованном управлении станция производит передачу своего кадра в шину при получении полномочий на это со стороны управляющей сетью станции, которая выдает разрешение на передачу с учетом приоритетов станций, наличия у них подготовленных для передачи кадров, времени ожидания станцией и подготовленным кадром разрешения на передачу. Учет указанных обстоятельств требует сложного обмена служебной информацией по шине или специальным линиям между станциями сети и управляющей станцией.

В более простом варианте детерминированный доступ организуется следующим образом. Управляющая станция задает последовательность, в которой должны передаваться полномочия станциям на доступ к шине для передачи данных, и адресует кадр с меткой разрешения на передачу первой станции этой последовательности. Получившая полномочия станция

передает свой подготовленный кадр в шину и после этого, а если кадра для передачи нет, то сразу посылает кадр с разрешением на передачу (передает полномочия) другой станции согласно установленному порядку передачи полномочий и т. д., пока управляющая станция не инициирует новую серию передач.

При случайном доступе и децентрализованном управлении каждая станция в любой момент времени может пытаться занять шину для своей передачи. При этом станция или ее контроллер сети прослушивают шину и, если она свободна, начинают передачу. Если шина занята, станция повторяет попытку занять шину с некоторой задержкой по времени. Значение задержки должно выбираться таким, чтобы была малая вероятность, что новое обращение к шине не окажется в конфликте с попытками других станций занять шину. Например, в ЛВС *Ethernet* (США) формируется случайная задержка, среднее значение которой возрастает примерно вдвое с каждой новой неудачной попыткой произвести передачу.

Кольцевая ЛВС. Другой типичной структурой ЛВС является кольцевая (петлевая) сеть (рис. 16.10, б). Кольцо образуют несколько каналов, связываемых последовательно друг с другом при помощи приемопередающих блоков. К *ПП* через контроллеры сети присоединяются станции сети (ЭВМ, терминалы и др.). Если станция через свой *ПП* производит ввод сообщения, то кольцо в этом пункте на время ввода логически разрывается.

В подобных сетях реализуется эстафетная передача информации: снабженное адресом сообщение (кадр) передается по кольцу в одном направлении, при этом *ПП* каждой станции с небольшой задержкой ретранслирует сообщение соседней станции.

Станция, распознавшая в сообщении свой адрес, принимает сообщение на свои регистры и также производит его ретрансляцию. Станция-отправитель, получив свое сообщение, воспринимает его как квитанцию адресата о получении. Следует обратить внимание, что в рассматриваемой сети сообщение (кадр) *последовательно* во времени проходит через все станции сети. Некоторые варианты кольцевых сетей позволяют осуществлять одновременную передачу данных несколькими станциями.

Поскольку в каждом узле кольцевой ЛВС производится ретрансляция сообщений, территория, охватываемая кольцевой сетью, может быть больше территории, охватываемой шинной (магистральной) сетью (при отсутствии в последних ретрансляторов).

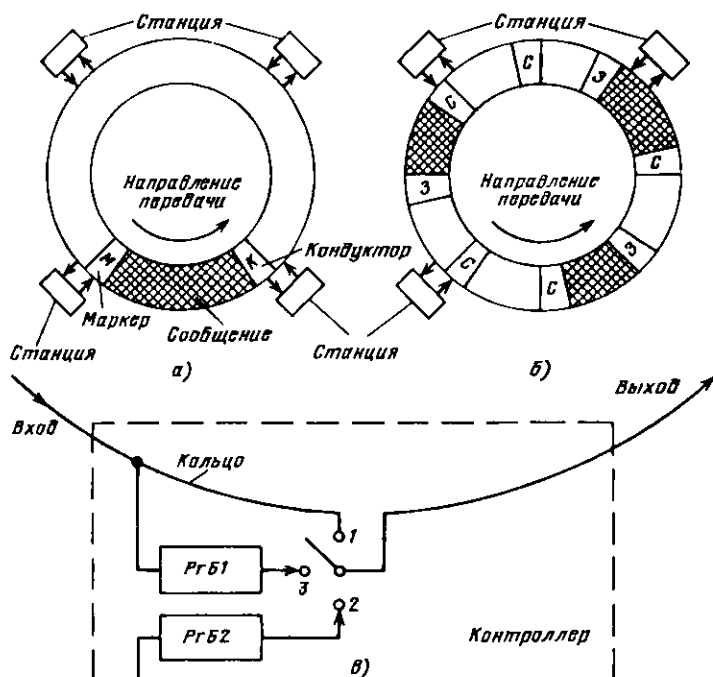


Рис. 16.11. Передача информации в кольцевой локальной вычислительной сети:

а — метод контрольного маркера; б — метод деления на временные сегменты; в — метод вставки регистров

Можно выделить три варианта построения кольцевых ЛВС: с циркулирующим маркером, временными сегментами и вставкой регистров [19].

Кольцевая ЛВС с циркулирующим маркером (рис. 16.11, а). По свободному от передачи сообщений кольцу от одного ПП к другому циркулирует маркер М, регенерируемый мониторной станцией.

Передачу сообщений может производить только та станция, на которой находится маркер. Остальные станции производят только прием. Как только на подготовившую сообщение станцию (точнее, ее контроллер сети) поступит маркер, его дальнейшее перемещение по кольцу временно блокируется. Маркер заменяется другим служебным символом — «кондуктором» К, и с него начинается передача подготовленного сообщения, которая завершается передачей маркера.

Соседняя станция регистрирует все принятые символы вплоть до маркера и далее может передать либо собственное сообщение (завершив передачу маркером), либо сразу передать маркер. В соответствии с адресом, указанным в начале сообщения, оно будет воспринято соответствующей станцией как ей адресованное. Станция-адресат в служебном поле сообщения фиксирует факт его приема и результат контроля правильности принятого сообщения.

Сообщение продолжает двигаться по кольцу до станции-отправителя. Эта станция воспринимает свое сообщение как квитанцию, проверяет его и в случае отсутствия ошибок удаляет его из кольца, а в противном случае организует повторную передачу.

Кольцевая ЛВСт с циркулирующим маркером позволяет передавать сообщения произвольной длины, но не допускает одновременную передачу сообщений несколькими станциями, требует централизованного управления.

Кольцевая ЛВСт с делением на временные сегменты (рис. 16.11, б). В кольцевом канале связи сигналами мониторинговой станции выделяются временные сегменты фиксированной длины.

Сегменты циркулируют по кольцу, и каждая станция может поместить свой пакет данных (кадр) в один из них, если он помечен как «свободный», при этом в сегменте метка «свободный» заменяется меткой «занятый». После доставки пакета адресату сегмент вновь освобождается.

Данный вариант построения кольцевой ЛВСт требует централизованного управления, допускает одновременную передачу данных несколькими станциями. Передача данных производится пакетами (кадрами) фиксированной длины.

Кольцевая ЛВСт с вставкой регистров (рис. 16.11, в). Оборудование каждого узла сети содержит буферные сдвигающие регистры P_2B1 , P_2B2 одинаковой емкости и электронный переключатель, который в положении 1 соединяет входную и выходную линии узла (отключает станцию от выходной линии), а в положениях 2 и 3 подключает к выходной линии выходы сдвигающих регистров соответственно P_2B2 и P_2B1 .

Вход сдвигающего регистра P_2B1 всегда соединен с входной линией, и в него поочередно поступают передаваемые по кольцу сообщения. Если поступившее в P_2B1 сообщение распознается как адресованное данному узлу, оно поступает на его станцию, при этом сообщение продолжает передаваться по кольцу, пока не достигнет станции-отправителя.

Когда станция готова передать сообщение, она загружает его в сдвигающий регистр P_2B2 . Передаче должно предшество-

вать состояние узла, соответствующее положению 1 переключателя. В этом состоянии станция (контроллер сети) проверяет, происходит ли в данный момент через ее узел ретрансляция сообщения, и ждет его окончания. Затем переключатель устанавливается на определенное время в положение 2 и подготовленное сообщение из *P2B2* поступает в кольцевую сеть. Если во время его передачи на вход узла поступит другое сообщение, оно принимается на буферный сдвигающий регистр *P2B1*.

После окончания передачи своего сообщения станция устанавливает переключатель в положение 3 и в кольцевую сеть передается сообщение, сохраненное в буферном регистре *P2B1*. Переключатель остается в положении 3 до момента, когда в *P2B1* поступит с кольца посланное станцией сообщение, что служит квитанцией о его передаче. Переключатель снова устанавливается в положение 1.

Рассматриваемый вариант предполагает централизованное управление, при котором мониторинговая станция обнаруживает и удаляет сообщения с отсутствующими в сети адресами, осуществляет синхронизацию работы кольцевой сети. Возможна одновременная передача сообщений несколькими станциями. Длина сообщений не фиксирована, но ограничена емкостью буферных регистров.

К общим недостаткам кольцевых (петлевых) сетей надо отнести их невысокую живучесть: достаточно выйти из строя одной станции или одной линии связи — и сеть становится неработоспособной. Поэтому были предложены конфигурации, использующие несколько однонаправленных петель. Такие системы менее критичны к отдельным неисправностям. При возникновении отказов может быть произведена реконфигурация с целью восстановить хотя бы одну петлю. Наличие дополнительных линий связи позволяет повысить пропускную способность системы.

На рис. 16.10,а аппаратура ЛВСт соотнесена с иерархией функциональных уровней эталонной модели, рассмотренной в § 16.3. Сетевой и транспортный уровни обеспечиваются либо контроллером сети, либо им совместно со станцией. Более высокие функциональные уровни осуществляются станциями программным образом. Отметим, что в шинной и кольцевой ЛВСт функции сетевого уровня значительно упрощаются, так как фиксированы маршруты передачи сообщений (кадров): в шинной ЛВСт — от передающей станции через шину параллельно ко всем другим станциям, а в кольцевой сети — от передающей станции последовательно ко всем станциям кольца.

16.7. ПРИНЦИПЫ ОРГАНИЗАЦИИ И СРЕДСТВА ТЕЛЕОБРАБОТКИ ДАННЫХ

В автоматизированных системах управления рассредоточенными на значительной территории объектами, например предприятиями в отраслевой АСУ, при построении крупных информационно-вычислительных систем ЭВМ должна взаимодействовать с удаленными пользователями (абонентами). В таких случаях используются ВС с телеобработкой данных.

Телеобработкой данных называется режим работы вычислительной системы, при котором ЭВМ и удаленные абоненты [абонентские пункты (АП)] взаимодействуют через каналы передачи данных.

Вычислительные системы с телеобработкой данных можно рассматривать как простейший случай вычислительных сетей, в которых применяются лишь некоторые элементы сетевой технологии (например, коммутация каналов, но не коммутация пакетов).

Характерными режимами систем телеобработки данных являются:

а) дистанционные вычисления, при выполнении которых с АП через канал передачи данных вводятся в ЭВМ исходные данные и наименование запрашиваемой программы, а обратно на АП передаются результаты обработки программ;

б) дистанционный информационно-справочный режим, при реализации которого с АП через канал передачи данных посылается запрос на информацию, хранимую в базе данных ЭВМ, и запрашиваемая справка отсылается ЭВМ на пославший запрос АП;

в) дистанционный режим сбора данных, часто выполняемый путем предварительного нанесения подлежащих передаче сведений на перфоленту (перфокарты) с последующей передачей их из АП в ЭВМ через канал передачи данных при помощи перфоленточного (перфокарточного) устройства ввода;

г) системы коллективного пользования с дистанционным доступом (теледоступом) к ЭВМ абонентов с удаленных терминалов.

Структура системы телеобработки данных в ЕС ЭВМ представлена на рис. 16.12 [27].

Одним из абонентов системы телеобработки данных может быть другая ЭВМ. В таком случае получаем сеть телеобработки.

Реализация систем телеобработки данных требует специальных аппаратурных и программных средств.

Аппаратурные средства телеобработки включают в себя аппаратуру передачи данных (АПД), мультиплексоры передачи данных (МПД) и абонентские пункты (АП).

Абонентский пункт представляет собой объединяемый общим устройством управления и связи комплекс устройств ввода-вывода, с помощью которых абонент производит ввод данных в систему и получает сообщения от ЭВМ. В зависимости от назначения используемых каналов связи и скорости передачи данных АП может включать в себя АПД, электрифицированные пишущие машинки, печатающие устройства, перфоленточные или перфокарточные устройства, дисплеи, датчики значений различных величин и т. п.

Функционирование комплекса оборудования АП организуется устройством управления и связи (УУС), которое устанавливает и прекращает связь АП с МПД, участвует в управлении передачей информации, воспринимая поступающие с линии и выдавая в линию соответствующие управляющие сигналы, выполняет необходимые преобразования кодов, формирует контрольные суммы для передаваемых сообщений.

Оборудование большинства АП реализует как дистанционную пакетную обработку, так и диалоговый режим работы. Некоторые АП позволяют организовать дистанционный удаленный режим коллективного пользования ЭВМ для нескольких одновременно работающих пользователей (в АП-4 до 8).

Аппаратура передачи данных. В состав АПД входят модемы, устройства защиты от ошибок, автоматические вызывные устройства (АВУ). В зависимости от скорости передачи информации и используемых каналов различают АПД низкоскоростную (до 200 бит/с по телеграфным каналам), среднескоростную (до 4800 бит/с по каналам тональной частоты) и высокоскоростную (более 4800 бит/с по широкополосным каналам).

Используются некоммутируемые (постоянно выделенные абоненту), а также коммутируемые каналы. В последнем случае связь между абонентскими пунктами и ЭВМ устанавливается автоматическими станциями телефонными (АТС) и абонентского телеграфа (АТА). При этом в случае использования телефонных каналов со стороны ЭВМ вызов АП производится автоматически с помощью АВУ, а со стороны АП связь с ЭВМ устанавливается при помощи телефона. В случае телеграфных каналов для вызова со стороны ЭВМ и АП применяются вызывные приборы (ВП).

В зависимости от вида каналов связи и скорости передачи данных используются модемы различных типов. В состав низкоскоростной АПД входит модем 200 с частотной модуляцией, среднескоростной — модемы 2400 и 4800 с фазовой модуля-

цией. Номера модемов соответствуют скорости передачи в битах в секунду.

Устройства защиты от ошибок (УЗО) производят помехоустойчивое кодирование блоков данных при передаче и соответствующее декодирование при приеме.

Используются циклические коды со следующими порождающими полиномами:

$$\begin{aligned}x^{16} + x^{12} + x^5 + 1; \\ x^{24} + x^{23} + x^6 + x^4 + 1; \\ x^{24} + x^{23} + x^7 + x^5 + x^2 + 1.\end{aligned}$$

Если при приеме обнаруживается ошибка, инициируется повторная передача ошибочного блока. При наличии УЗО достоверность передачи данных достигает 10^{-7} .

Мультиплексоры передачи данных управляют передачей данных через сеть связи. В функции МПД входят реализация сопряжения между ЭВМ и аппаратурой передачи данных нескольких каналов, управление установлением и прекращением связи с несколькими АП, осуществление дисциплины обслуживания АП. Мультиплексоры подключаются к мультиплексному (байтмультиплексному) каналу ЭВМ и управляются канальными программами.

Мультиплексоры передачи данных в процессе передачи данных для каждого канала связи осуществляют проверку правильности принятых блоков данных и их буферизацию, формируют контрольные символы при передаче блоков, производят преобразование кодов, формируют и распознают регистровые и управляющие символы, управляют символьной синхронизацией и контролируют временные паузы.

В ЕС ЭВМ при передаче данных по телеграфным каналам используется 5-разрядный код МТК-2, по телефонным и широкополосным каналам — 7-разрядный код КОИ-7.

Из поступающих из мультиплексора через мультиплексорный (байтмультиплексный) канал ввода-вывода кодов ЭВМ формирует сообщение, преобразует сообщение в код ДКОИ, используемый при обработке в ЭВМ алфавитно-цифровой информации. При передаче сообщения ЭВМ осуществляет преобразование из кода ДКОИ в код, используемый при передаче по каналам связи, формирует символы адреса АП.

МПД специального типа — «удаленный МПД» — выполняет помимо прочих также функции концентратора, связывая несколько низкоскоростных каналов связи с одним высокоскоростным.

Сложность выполняемых функций делает целесообразным использование в качестве МПД малых и микро-ЭВМ. В этом случае имеется возможность в большей степени разгрузить основную ЭВМ от выполнения процедур, связанных с передачей данных по каналам связи, преобразованием кодов и др. Выполняющую такие функции ЭВМ называют связным процессором.

Программные средства телеобработки в ЕС ЭВМ представлены пакетами программ: а) *базисного телекоммуникационного метода доступа* (БТМД) и б) *общего телекоммуникационного метода доступа* (ОТМД), главным назначением которых является управление передачей информацией между ЭВМ и АП.

Пакет программ БТМД позволяет пользоваться специальными макрокомандами: «Открыть» и «Закрыть», соответственно иницилирующей и прекращающей сеанс связи через канал передачи данных; «Читать», опрашивающей терминалы путем сканирования или по заданному адресу и в случае наличия подготовленного сообщения, принимающего его в ЭВМ; «Писать», передающей сообщение адресуемому терминалу.

Программы этого пакета производят также необходимые преобразования кодов, контроль правильности передачи с помощью контролирующих кодов, диагностирование терминалов, сбор и обработку статистических данных об ошибках при передаче данных, динамическое распределение памяти при приеме сообщений. Метод теледоступа БТМД не обеспечивает телеобработку с очередями сообщений.

Более сложный пакет теледоступа ОТМД помимо функций, обеспечиваемых пакетом БТМД, позволяет также создавать очереди сообщений, обрабатывать сообщения с учетом их приоритетов и реализовывать некоторые другие возможности, например использовать язык пакета для упрощения составления программ специфических процедур телеобработки.

Глава семнадцатая

ВЕРОЯТНОСТНЫЕ АНАЛИТИЧЕСКИЕ МОДЕЛИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

17.1. ОБЩИЕ ПРИНЦИПЫ ПОСТРОЕНИЯ МОДЕЛЕЙ ВС

При проектировании новых и модернизации действующих вычислительных систем и сетей необходимо предварительно оценивать эффективность их функционирования с учетом раз-

личных вариантов структурной организации. Эти варианты могут отличаться составом и характеристиками модулей (устройств), структурой и характеристиками межмодульных связей, режимами работы и алгоритмами управления. Для таких оценок используют модели вычислительных систем.

В вычислительных системах запросы пользователей на обслуживание поступают, как правило, в случайные моменты времени. Случайными величинами оказываются также требуемые для обслуживания запроса пользователя объем памяти и времени работы процессора, интерфейсов, аппаратуры ввода-вывода и других устройств. Случайными являются моменты возникновения отказов и сбоев в оборудовании ВС и продолжительности ремонтных работ.

Из-за случайности моментов появления запросов и случайной продолжительности времени их обслуживания при функционировании ВС возникают очереди к ресурсам системы, в том числе очереди к процессорам, к устройствам ввода-вывода, к внешним ЗУ, к каналам передачи данных (см., например, рис. 13.7).

Обычно разработчика интересуют оценки временных задержек и длин очередей при обращении к ресурсам ВС и передаче сообщений между удаленными ЭВМ, вероятности различных блокировок, сбоев, перегрузок, несвоевременной доставки сообщений в сети, коэффициенты использования (загрузки) отдельных устройств (процессоров, памяти, каналов) и программных средств, производительность или пропускная способность как системы в целом, так и ее отдельных подсистем и устройств, потери времени и пропускной способности из-за различных нежелательных событий в системе.

Во многих случаях модели ВС могут быть представлены в виде совокупности взаимосвязанных обслуживающих устройств с очередями [систем массового обслуживания (СМО)], в которой запросы с определенной вероятностью переходят из одной СМО в другую. Такая совокупность СМО называется *сетью массового обслуживания* (сетью МО).

Не следует путать абстрактную модель ВС, которой является сеть МО, с рассмотренными в предыдущей главе аппаратурно-программными комплексами, называемыми вычислительными сетями.

В настоящее время при анализе ВС и оценке их эффективности широко используются модели ВС в виде сетей МО и применяется аппарат теории сетей МО [1, 62].

Модели в виде сетей МО можно классифицировать по типу источников запросов. Источник называется конечным, если он порождает конечное число запросов, например имеется конеч-

ное число терминалов, за которыми работают пользователи, и пользователь может вводить новый запрос, лишь получив ответ на предыдущий. Сети МО с конечными источниками называются *замкнутыми* (обработанные запросы возвращаются в источники).

Если источник содержит бесконечно большое число запросов, то он называется бесконечным, а соответствующая сеть МО — *разомкнутой*. Поток запросов от бесконечного источника называется *внешним потоком сети*. Таким образом, замкнутые сети МО содержат фиксированное число запросов, которые не покидают сеть МО, а новые запросы извне не поступают. В разомкнутой сети МО допускается приход извне и уход запросов, поэтому число запросов в сети может меняться.

Обычно замкнутые сети МО служат моделями ВС, в которых обрабатывается фиксированное или ограниченное число запросов. Разомкнутые сети МО моделируют ВС с переменным, неограниченным числом запросов.

Каждая СМО в сети МО содержит одно или несколько одинаковых обслуживающих устройств, способных реализовать одно и то же специфическое обслуживание запросов, например выполнение программы, вывод данных и т. д. Устройства одной СМО обслуживают запросы параллельно и независимо. Если число обслуживающих устройств не меньше максимально возможного числа запросов в данной СМО, тогда в этой СМО отсутствует очередь запросов.

Традиционно для определения типа СМО используют обозначение вида $X/Y/k$, где первая буква обозначает закон распределения времени между поступлением запросов, вторая — закон распределения длительности обслуживания запросов, третья — число обслуживающих устройств. При $k = 1$ СМО называется *однолинейной*, при $k > 1$ — *многолинейной* или *k — линейной*. Используемые далее обозначения соответствуют следующим распределениям: M — экспоненциальному, G — произвольному. Например, в СМО типа $M/M/k$ имеется k обслуживающих устройств, входящий поток запросов является стационарным пуассоновским, т. е. интервалы времени между поступлениями запросов распределены по экспоненциальному закону с функцией распределения

$$F(t) = 1 - e^{-\lambda t}, \quad t > 0.$$

Параметр λ этого распределения определяет среднюю интенсивность поступления запросов и равен величине, обратной средней длине интервала между поступлениями запросов $\lambda =$

$= 1/t_{\text{вх. ср}}$ Длительность обслуживания запросов каждым устройством распределена по экспоненциальному закону с параметром μ , который определяет среднюю интенсивность обслуживания запросов одним устройством и равен величине, обратной средней длительности обслуживания: $\mu = 1/t_{\text{об. ср}}$

Важным параметром СМО является коэффициент загрузки обслуживающего устройства

$$\rho = \lambda/\mu k. \quad (17.1)$$

Стационарный режим в СМО возможен, если интенсивность входящего потока запросов меньше интенсивности обслуживания запросов, т. е. если

$$0 \leq \rho < 1. \quad (17.2)$$

В противном случае очередь будет неограниченно возрастать. В замкнутых сетях МО всегда возможен стационарный режим работы. В разомкнутых сетях МО стационарный режим возможен при условии, что в каждой СМО сети выполняется условие (17.2).

В стационарном режиме в СМО интенсивности входящего и выходящего потоков запросов равны между собой.

Реальные распределения для случайных величин — интервалов между моментами поступления запросов и длительности обслуживания запросов в отдельных СМО сети МО устанавливаются путем исследования поведения отдельных программных и аппаратурных компонентов ВС, которые моделируются этими СМО.

Чтобы сеть МО можно было рассчитывать аналитически, за редким исключением приходится предполагать, что интервалы времени между поступлениями запросов и длительности обслуживания распределены по экспоненциальным законам.

На практике в реальных системах входящий поток запросов часто оказывается весьма близким к пуассоновскому, поскольку он обычно является суперпозицией отдельных независимых потоков.

Предположение об экспоненциально распределенной длительности обслуживания запросов нередко оказывается не соответствующим реальной ситуации. Однако опыт моделирования показывает, что приближенные результаты, полученные при этих предположениях, оказываются вполне приемлемыми при инженерных расчетах. Именно такие сети МО, называемые экспоненциальными, используются далее при построении моделей ВС.

17.2. СЕТИ МАССОВОГО ОБСЛУЖИВАНИЯ

Рассмотрим некоторую i -ю СМО типа $M/M/k_i$ (рис. 17.1, а) с неограниченной длиной очереди. Пусть λ_i — интенсивность входящего потока запросов, μ_i — интенсивность обслуживания запросов каждым из устройств и справедливо условие (17.2).

Пусть $\Pi_i(x_i)$ — стационарная вероятность состояния, в котором в i -й СМО находится x_i запросов. Граф состояний этой СМО показан на рис. 17.1, б. Система линейных уравнений для стационарных вероятностей состояний имеет следующий вид:

$$\left. \begin{aligned} \lambda_i \Pi_i(0) &= \mu_i \Pi_i(1); \\ [x_i \mu_i + \lambda_i] \Pi_i(x_i) &= \lambda_i \Pi_i(x_i - 1) + [x_i + 1] \mu_i \Pi_i(x_i + 1) \\ &\text{при } 1 \leq x_i < k_i; \\ [k_i \mu_i + \lambda_i] \Pi_i(x_i) &= \lambda_i \Pi_i(x_i - 1) + k_i \mu_i \Pi_i(x_i + 1) \text{ при } x_i \geq k_i. \end{aligned} \right\} \quad (17.3)$$

Легко убедиться, что решение системы уравнений (17.3) имеет вид:

$$\Pi_i(x_i) = A_i(x_i) (\lambda_i / \mu_i)^{x_i} \Pi_i(0), \quad (17.4)$$

где

$$A_i(x_i) = \begin{cases} (x_i!)^{-1} & \text{при } x_i \leq k_i; \\ (k_i! k_i^{x_i - k_i})^{-1} & \text{при } x_i > k_i. \end{cases} \quad (17.5)$$

Учитывая условие нормировки $\sum_{x_i=0}^{\infty} \Pi_i(x_i) = 1$ и суммируя

члены геометрической прогрессии (17.4) со знаменателем $\lambda_i / x_i \mu_i$, получаем

$$\Pi_i(0) = \left[\sum_{x_i=0}^{k_i-1} \frac{(\lambda_i / \mu_i)^{x_i}}{x_i!} + \frac{(\lambda_i / \mu_i)^{k_i}}{k_i! (1 - \lambda_i / \mu_i)} \right]^{-1}. \quad (17.6)$$

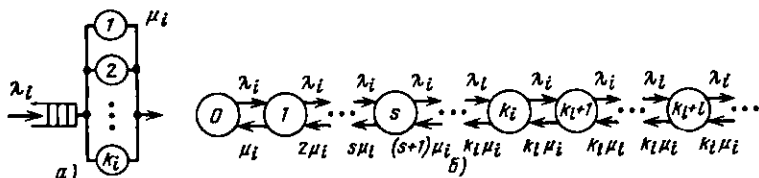


Рис. 17.1. Многолинейная система массового обслуживания (а) и ее граф состояний (б)

Модели ВС в виде экспоненциальных сетей МО часто содержат также СМО типа $M/M/1$ и $M/M/\infty$. В первом случае из приведенных формул получаем

$$P_i(x_i) = (1 - \lambda_i/\mu_i)(\lambda_i/\mu_i)^{x_i}, \quad (17.7)$$

во втором случае («общая» СМО)

$$P_i(0) = \left[\sum_{x_i=0}^{\infty} \frac{(\lambda_i/\mu_i)^{x_i}}{x_i!} \right]^{-1} = e^{-\lambda_i/\mu_i}; \quad (17.8)$$

$$P_i(x_i) = e^{-\lambda_i/\mu_i} \frac{(\lambda_i/\mu_i)^{x_i}}{x_i!}. \quad (17.9)$$

Если в замкнутой сети МО имеется k_i -линейная СМО, а общее число запросов N , циркулирующих в этой сети, не превышает k_i , то в k_i -линейной СМО очередь отсутствует и ее можно рассматривать как СМО типа $M/M/\infty$.

Простейший случай разомкнутой сети МО — две последовательно соединенные однолинейные СМО (рис. 17.2, а).

Предполагается, что входящий поток является пуассоновским с интенсивностью λ , длительности обслуживания в СМО1 и СМО2 распределены по экспоненциальному закону с интенсивностями обслуживания μ_1 и μ_2 соответственно, длины очередей неограниченны, сеть стационарна (т. е. $\lambda < \min(\mu_1, \mu_2)$). Состояние сети МО, изображенной на рис. 17.2, а, определяется вектором $\vec{x} = (x_1, x_2)$, где x_1 и x_2 — число запросов, находящихся соответственно в СМО1 и СМО2.

Пусть через $P(x_1, x_2)$ обозначается стационарная вероятность состояния $\vec{x} = (x_1, x_2)$. Граф состояний этой сети МО показан на рис. 17.2, б.

Система линейных уравнений для стационарных вероятностей состояний (уравнения баланса) записывается следующим образом:

$$\left. \begin{aligned} \lambda P(0,0) &= \mu_2 P(0,1); \\ (\lambda + \mu_2) P(0, x_2) &= \mu_1 P(1, x_2 - 1) + \mu_2 P(0, x_2 + 1); \\ (\lambda + \mu_1) P(x_1, 0) &= \mu_2 P(x_1, 1) + \lambda P(x_1 - 1, 0); \\ (\lambda + \mu_1 + \mu_2) P(x_1, x_2) &= \mu_1 P(x_1 + 1, x_2 - 1) + \\ &+ \mu_2 P(x_1, x_2 + 1) + \lambda P(x_1 - 1, x_2), \end{aligned} \right\} \quad (17.10)$$

где $x_1 \geq 1, x_2 \geq 1$.

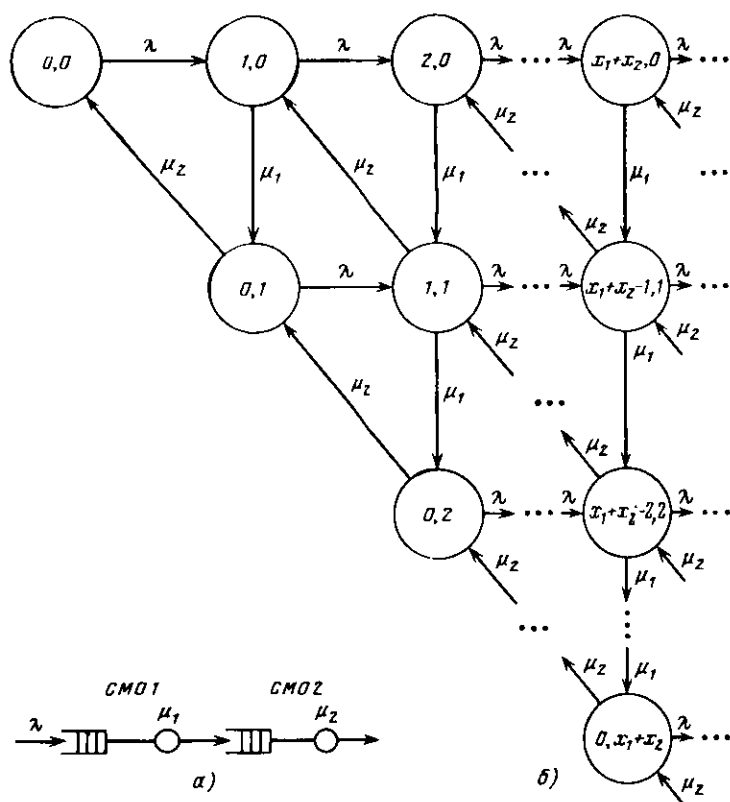


Рис. 17.2. Простейшая разомкнутая сеть массового обслуживания (а) и ее граф состояния (б)

Нетрудно убедиться в том, что с учетом условия нормировки

$$\sum_{i,j} P(x_1, x_2) = 1$$

решение системы уравнений (17.10) имеет вид

$$P(x_1, x_2) = \left(1 - \frac{\lambda}{\mu_1}\right) \left(\frac{\lambda}{\mu_1}\right)^{x_1} \left(1 - \frac{\lambda}{\mu_2}\right) \left(\frac{\lambda}{\mu_2}\right)^{x_2} = P_1(x_1) P_2(x_2). \quad (17.11)$$

Эта формула означает [ср. с (17.7)], что каждая СМО может рассматриваться независимо как система типа $M/M/1$.

Система массового обслуживания типа $M/M/1$ с неограниченной очередью и со стационарным режимом работы имеет пуассоновский выходящий поток [110]. Интервалы между моментами выхода запросов с обслуживания, так же как и интервалы между поступлением запросов, имеют экспоненциальное распределение с параметром λ .

Отсюда следует, что и для n последовательно соединенных СМО с неограниченными очередями стационарная вероятность состояния $\bar{x} = (x_1, \dots, x_n)$, в котором в СМО с номерами $1, \dots, n$ находятся соответственно $x_1, \dots, x_n$ запросов:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P_i(x_i), \quad (17.12)$$

где $P_i(x_i)$ – стационарная вероятность такого состояния i -й СМО, когда в ней находится x_i запросов. Каждая СМО может рассматриваться изолированно как система типа $M/M/1$ с интенсивностью поступления запросов λ и интенсивностью обслуживания μ_i ($\lambda < \mu_i$).

Этот результат обобщен на случай более сложных разомкнутых сетей МО с обратными связями (петлями) и наличием внешних входящих потоков запросов в любую из СМО (рис. 17.3) [117]. Запросы перемещаются в сети случайным образом в соответствии с фиксированными вероятностями переходов из одной СМО в другую.

Обычно в разомкнутых сетях МО рассматривается один внешний источник запросов с интенсивностью потока запросов, равной λ_0 . Интенсивности всех внешних потоков запросов в любую из СМО составляют некоторую долю от λ_0 . Например, на рис. 17.3 $\lambda_1 = p_{01}\lambda_0$; $\lambda_2 = p_{02}\lambda_0$; $\lambda_5 = p_{05}\lambda_0$; $p_{01} + p_{02} + p_{05} = 1$.

Так же как и при последовательно соединенных СМО, стационарная вероятность $P(x_1, \dots, x_n)$ состояния сети МО представима в виде произведения вероятностей состояний отдель-

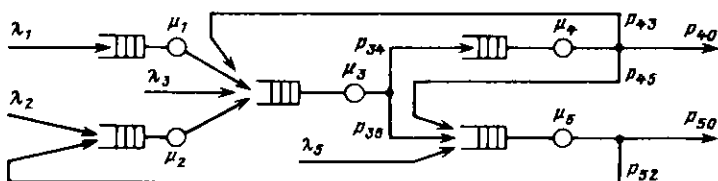


Рис. 17.3. Пример разомкнутой сети МО с обратными связями

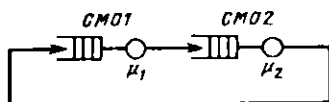
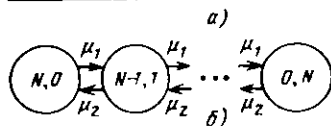


Рис. 17.4. Простейшая замкнутая сеть МО (а) и ее граф состояний (б)



ных СМО $\Pi_i(x_i)$, т. е. в форме (17.12). Интересно, что сеть МО ведет себя так, как будто входящие потоки отдельных СМО являются пуассоновскими, не-

смотря на то что на самом деле в сети МО с петлями потоки не пуассоновские.

Для использования этого результата в общем случае сети, содержащей n СМО, необходимо вычислить интенсивности λ'_i поступления запросов в i -ю СМО. Пусть λ_0 — общая интенсивность потока запросов, входящего в разомкнутую сеть от внешнего источника. Введем коэффициенты передач e_i ($i = 1, n$) такие, что $\lambda'_i = e_i \lambda_0$. Интенсивность λ'_i складывается из интенсивности λ_i поступления запросов от внешнего источника ($\lambda_i = p_{0i} \lambda_0$) и интенсивностей $\lambda'_{ji} p_{ji}$ поступления запросов в i -ю СМО из j -й СМО. Таким образом,

$$e_i = p_{0i} + \sum_{j=1}^n e_j p_{ji}, \quad i = 1, n. \quad (17.13)$$

Найдя из этой системы уравнений коэффициенты передач e_i , можно определить стационарные вероятности состояний отдельных СМО по выражениям (17.4), (17.7) и (17.9), в которых вместо интенсивности λ_i надо подставить интенсивность $e_i \lambda_0$. Очевидно, e_i определяет среднее число проходов запроса через i -ю СМО.

Рассмотрим простейший случай замкнутой сети МО, состоящей из двух однолинейных СМО (рис. 17.4, а). В качестве конечного источника можно рассматривать любую СМО. Предполагается, что в сети МО циркулируют N запросов, длительности обслуживания запросов в СМО1 и СМО2 распределены по экспоненциальному закону с параметрами μ_1 и μ_2 . Состояние сети определяется вектором $\bar{x} = (x_1, x_2)$, причем $x_1 + x_2 = N$. Граф состояний этой сети МО показан на рис. 17.4, б.

Система линейных уравнений для стационарных вероятностей состояний имеет следующий вид:

$$\left. \begin{aligned} \mu_1 \Pi(N, 0) &= \mu_2 \Pi(N-1, 1); \quad \mu_2 \Pi(0, N) = \mu_1 \Pi(1, N-1); \\ (\mu_1 + \mu_2) \Pi(x_1, x_2) &= \mu_1 \Pi(x_1+1, x_2-1) + \\ &\quad + \mu_2 \Pi(x_1-1, x_2+1), \end{aligned} \right\} \quad (17.14)$$

где $0 < x_1, x_2 < N$.

С учетом условия нормировки $\sum \Pi(x_1, x_2) = 1$ получаем

$$\begin{aligned}\Pi(x_1, x_2) &= \Pi(N, 0) \left(\frac{\mu_1}{\mu_2} \right)^{x_2} = \\ &= \Pi(N, 0) \mu_1^{x_1 + x_2} \left(\frac{1}{\mu_1} \right)^{x_1} \left(\frac{1}{\mu_2} \right)^{x_2} = \\ &= C(N) \left(\frac{1}{\mu_1} \right)^{x_1} \left(\frac{1}{\mu_2} \right)^{x_2} = C(N) \Pi(x_1) \Pi(x_2), \quad (17.15)\end{aligned}$$

где $C(N) = \Pi(N, 0) \mu_1^N =$

$$= \left[\sum_{i=0}^N \left(\frac{1}{\mu_1} \right)^i \left(\frac{1}{\mu_2} \right)^{N-i} \right]^{-1} - \text{нормировочная константа.}$$

Выражение для стационарных вероятностей состояний замкнутой сети не зависит от интенсивностей λ_i , а зависит от коэффициентов передачи e_i . В рассматриваемом примере $e_1 = e_2 = 1$.

Как и в разомкнутой сети МО, стационарные вероятности состояний замкнутой сети МО, где i -я СМО может быть k_i -линейной ($i = 1, \dots, n$), выражаются в виде произведения вероятностей состояний $\Pi_i(x_i)$ отдельных СМО. Отличие состоит в дополнительном нормирующем множителе $C(N)$, который учитывает, что $\sum_{i=1}^n x_i = N$ [114].

Поскольку в замкнутой сети МО внешние входящие потоки отсутствуют ($p_{0i} = 0$ и $\lambda_i = 0$, $i = \overline{1, n}$), уравнение (17.13) принимает вид

$$e_i = \sum_{j=1}^n e_j p_{ji}. \quad (17.16)$$

Чтобы решение этой системы было однозначно, полагают $e_1 = 1$. Тогда величины e_i можно интерпретировать как среднее число проходов запроса через i -ю СМО между двумя последовательными его посещениями первой СМО (т. е. за один полный цикл обслуживания запроса в сети МО). Например, в сети МО на рис. 17.8, б $e_1 = 1$, $e_2 = p$, $e_3 = q$.

Для такой замкнутой сети стационарная вероятность состояния, заданного вектором $\bar{x} = (x_1, x_2, \dots, x_n)$, при $\sum_{i=1}^n x_i = N$ определяется выражением

$$\Pi(\bar{x}) = C(N) \prod_{i=1}^n \Pi_i(x_i) = C(N) \prod_{i=1}^n A_i(x_i) (e_i/\mu_i)^{x_i}, \quad (17.17)$$

где $P_i(x_i)$ — стационарная вероятность того, что в i -й СМО находятся x_i запросов, $A_i(x_i)$ определено в (17.5).

Нормировочная константа

$$C(N) = \left[\sum_{x_i=1}^n \prod P_i(x_i) \right]^{-1}. \quad (17.18)$$

Для всех рассмотренных сетей МО выражения для стационарных вероятностей состояний имеет форму простого произведения (*мультипликативную форму*). Оказалось, что мультипликативная форма стационарных вероятностей состояний существует для значительно более широкого класса сетей МО с разнородными запросами, которые различным образом перемешаются по сети МО и в некоторых СМО могут обслуживаться с разными интенсивностями. Разнородные запросы разделяются на классы. Такого рода сеть МО может быть замкнутой, разомкнутой или смешанной: замкнутой для запросов одних классов и разомкнутой для других [108].

При моделировании ВС во многих случаях важно учитывать разнородность обслуживаемых запросов. Примерами таких запросов могут служить обращения к памяти за скалярными и векторными операндами, которые затем обрабатываются на разных процессорах (скалярных или векторных), системные и пользовательские программы, требующие обслуживания разными устройствами, различные типы сообщений, передаваемые в сети ЭВМ, и т. д.

Рассмотрим пример простой замкнутой сети МО из двух однолинейных СМО, в которой циркулируют запросы двух классов (рис. 17.5, а). Пусть каждый из классов содержит по одному запросу, а длительность обслуживания любого запроса в СМО1 и СМО2 распределена по экспоненциальному закону с интенсивностями μ_1 и μ_2 соответственно. Запросы обслуживаются в порядке их поступления в СМО.

Из СМО1 запрос r -го класса с вероятностью p_{12r} переходит в СМО2, а с вероятностью p_{11r} возвращается в конец очереди СМО1 ($p_{12r} + p_{11r} = 1$, $r = 1, 2$).

Состояние i -й СМО описывается вектором $\bar{z}_i = (\alpha_{i1}, \alpha_{i2})$, отражающим положение запросов в данной СМО. Значение символа α_{ij} указывает номер класса, запрос которого стоит в очереди на j -м месте. Например, вектор $\bar{z}_1 = (2, 1)$ определяет состояние СМО, в которой в данный момент обслуживается запрос второго класса, а в очереди стоит запрос первого класса. Тогда пара $(z_1; z_2)$ определяет состояние сети МО в целом. Граф состояний показан на рис. 17.4, б.

Соответствующая система линейных уравнений для стационарных вероятностей состояний имеет следующий вид:

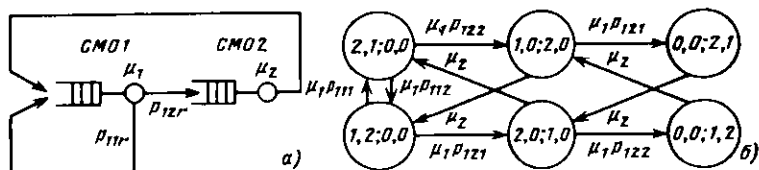


Рис. 17.5. Простейшая замкнутая сеть МО с запросами двух классов (а) и ее граф состояний (б)

$$\left. \begin{aligned} \mu_1 P(2,1; 0) &= \mu_2 P(2; 1) + \mu_1 p_{111} P(1,2; 0); \\ (\mu_2 + \mu_1 p_{121}) P(1; 2) &= \mu_1 p_{122} P(2,1; 0) + \mu_2 P(0; 1,2); \\ \mu_2 P(0; 2,1) &= \mu_1 p_{121} P(1,2). \end{aligned} \right\}$$

Уравнения для вероятностей $P(1,2; 0)$, $P(2;1)$, $P(0; 1,2)$ записываются аналогично.

Непосредственной подстановкой легко убедиться в том, что

$$\begin{aligned} P(2,1; 0) &= P(1,2; 0) = C(1/\mu_1)^2; \\ P(1; 2) &= C(1/\mu_1)(1/\mu_2)p_{122}; \\ P(2; 1) &= C(1/\mu_1)(1/\mu_2)p_{121}; \\ P(0; 2,1) &= P(0; 1,2) = C(1/\mu_2)^2 p_{121}p_{122}, \end{aligned}$$

где C — нормировочная константа.

Можно укрупнить состояния сети МО, определив их не с точностью до порядка запросов в очередях, а лишь числом запросов каждого класса в соответствующей СМО. В этом случае состояние рассматриваемой сети МО задается парой $(\bar{x}_1; x_2)$, где $\bar{x}_i = (x_{i1}, x_{i2})$, а x_{ri} — число запросов r -го класса в i -й СМО ($r = 1, 2$).

Таким образом,

$$\begin{aligned} P(1,1; 0,0) &= C2(1/\mu_1)^2; \quad P(1,0; 0,1) = C(1/\mu_1)(1/\mu_2)p_{122}; \\ P(0,1; 1,0) &= C(1/\mu_1)(1/\mu_2)p_{121}; \\ P(0,0; 1,1) &= C2(1/\mu_2)^2 p_{121}p_{122}. \end{aligned} \quad (17.19)$$

Из условия нормировки $P(1,1; 0,0) + P(0,1; 1,0) + P(0,0; 1,1) = 1$ легко получить выражение для константы C .

В теории сетей МО получены мультипликативные выражения для стационарных вероятностей состояний широкого класса замкнутых и смешанных сетей МО с разнородными запросами [108].

Пусть сеть МО содержит n СМО и обслуживает запросы R классов, причем запросы, покидая некоторую СМО, могут перейти из одного класса в другой. Каждому r -му классу соответствует фиксированная вероятность $p_{ijrr'}$ того, что запрос этого класса переходит из i -й СМО в j -ю и одновременно в r' -й класс. Внешний поток входящих в сеть запросов предполагается пуассоновским.

Возможны следующие четыре типа СМО:

1) СМО, обслуживающие запросы в порядке их поступления в очередь (прямая очередность). Длительность обслуживания имеет одинаковое для всех запросов экспоненциальное распределение с интенсивностью $\mu(x_i)$, зависящей от числа запросов в данной СМО (т.е. СМО может быть многолинейной);

2) однолинейные СМО с разделением времени, при котором, если в СМО находятся l запросов, каждый запрос за единицу времени обслуживается в течение $1/l$ единицы времени в циклическом порядке;

3) СМО, в которых отсутствует очередь запросов, так как всегда находится свободное обслуживающее устройство («общая» СМО);

4) однолинейные СМО с дисциплиной обслуживания «последним пришел — первым обслужен» (обратная очередность) и абсолютным приоритетом у запроса, поступившего последним.

Для СМО 2, 3 и 4-го типов длительности обслуживания запросов разных классов могут иметь различные распределения: экспоненциальное, гиперэкспоненциальное, эрланговское с разными параметрами.

Состояние сети МО определяется набором векторов $(\bar{x}_1; \dots; \bar{x}_n)$, где $\bar{x}_i = (x_{i1}, \dots, x_{iR})$ задает распределение числа запросов разных классов в i -й СМО [x_{ir} ($r = 1, R$) — число запросов r -го класса в i -й СМО].

В [108] показано, что стационарные вероятности состояний такой сети МО имеют мультипликативную форму (теорема о вероятностях состояний сетей МО с разнородными запросами):

$$P(x_1; \dots; \bar{x}_n) = C(N) \prod_{i=1}^n P_i(\bar{x}_i),$$

где $P_i(\bar{x}_i)$ зависит от типа i -й СМО.

В частности, для замкнутых сетей с многолинейными СМО первого типа, обобщая (17.4) и (17.19), находим

$$P(\bar{x}_1; \dots; \bar{x}_n) = C(\bar{N}) \prod_{i=1}^n \frac{x_i!}{x_{i1}! \dots x_{iR}!} A_i(x_i) \prod_{r=1}^R \left(\frac{e_{ri}}{\mu_i} \right)^{x_i}, \quad (17.20)$$

где $x_i = \sum_{r=1}^R x_{ri}$ — общее число запросов в i -й СМО.

Распределение запросов по разным классам задается вектором $N = (N_1, \dots, N_R)$, где N_r — число запросов r -го класса в сети МО. Значение $A_i(x_i)$ определяется формулой (17.5).

Величина e_{ri} определяет среднее число проходов запроса r -го класса через i -ю СМО за полный цикл обслуживания этого запроса в сети МО. Эти величины в общем случае находятся решением R систем линейных уравнений (для каждого класса r решается отдельная система уравнений):

$$e_{ri} = \sum_{j=1}^n e_{rj} p_{jir}, \quad i = \overline{1, n}, \quad (17.21)$$

где p_{jir} — вероятность перехода запроса r -го класса из j -й СМО в i -ю.

Нормировочная константа

$$C(N) = \left[\sum_{S(\bar{N})} \prod_{i=1}^n P_i(\bar{x}_i) \right]^{-1},$$

где множество состояний сети МО

$$S(\bar{N}) = \{(\bar{x}_1; \dots; \bar{x}_n) | \sum_{i=1}^n x_{ri} = N_r, \quad r = \overline{1, R}\}.$$

Заметим, что при $R = 1$ выражение (17.20) совпадает с (17.17), а (18.21) — с (17.16).

В разомкнутых сетях МО совокупная интенсивность $\lambda'_i(r)$ поступления запросов r -го класса в i -ю СМО равна $e_i(r) \lambda_o(r)$, где $\lambda_o(r)$ — общая интенсивность потока запросов r -го класса, поступающих в сеть от внешнего источника, а $e_i(r)$, $i = \overline{1, n}$, — коэффициенты передач, рассчитываемые из уравнений баланса потоков [см. (17.13)]. Если i -я СМО относится к первому типу, то мультипликативный член

$$P_i(\bar{x}_i) = \left[\frac{\sum_r \lambda'_i(r)}{\mu_i(x_i)} \right]^{x_i}, \quad (17.22)$$

где суммирование выполняется по тем классам, запросы которых поступают в i -ю СМО, а интенсивность обслуживания

$\mu_i(x_i)$ может зависеть от $x_i = \sum_{r=1}^R x_{ri}$ — общего числа запросов в i -й СМО.

Из теоремы о стационарных вероятностях состояний сетей МО с разнородными запросами следует, что даже сложные сети МО с СМО 1, 2 и 4-го типов функционируют как соответствующая совокупность автономных СМО типа $M/M/1$, а СМО 3-го типа в сети МО функционируют как СМО типа $M/M/\infty$. В частности, для разомкнутых сетей

$$P(x_1; \dots; x_n) = \prod_{i=1}^n P_i(x_i),$$

где $P_i(x_i)$ определяются выражением (17.7) или (17.9).

В общем случае для замкнутых и смешанных сетей МО отсутствует компактное выражение для нормировочной константы. Поэтому ее значение вычисляется на ЭВМ с помощью рекуррентных алгоритмов.

В качестве примера приведем алгоритм вычисления нормировочной константы для замкнутых сетей МО с многолинейными СМО и одним классом запросов [62].

Формулу (17.17) перепишем в виде

$$P(x_1, \dots, x_n) = \frac{\prod_{i=1}^n A_i(x_i) x_i^{x_i}}{G(N)}, \quad (17.23)$$

$$\text{где } x_i = e_i/\mu_i; \quad G(N) = C^{-1}(N) = \sum_{\substack{\mathcal{E}(N, n) \\ i=1^n}} \prod_{i=1}^n A_i(x_i) x_i^{x_i}.$$

Элементами множества $\mathcal{E}(N, n)$ являются все возможные состояния $(x_1, \dots, x_n)$ рассматриваемой сети МО.

Для рекуррентного вычисления значений $G(1)$, $G(2)$, ..., $G(N)$ введем вспомогательную функцию

$$g(s, j) = \sum_{\mathcal{E}(s, j)} \prod_{i=1}^j A_i(x_i) x_i^{x_i}, \quad (17.24)$$

причем

$$g(N, n) = G(N), \quad g(s, n) = G(s).$$

При $j=1$ имеем единственную СМО, поэтому

$$g(s, 1) = A_1(x_1) x_1^{x_1} = A_1(x_1) x_1^s \quad (s = \bar{0}, \bar{N}). \quad (17.25)$$

При $s = 0$ в сети МО нет запросов, поэтому $P(0, \dots, 0) = 1$ и

$$g(0, j) = 1 \quad (j = 1, \bar{n}). \quad (17.26)$$

Выражения (17.25) и (17.26) дают начальные условия для рекуррентного вычисления $G(N)$. При $s > 0$ и $j > 1$ из (17.24) имеем

$$g(s, j) = \sum_{l=0}^s \sum_{\substack{\mathcal{J}(s, j) \\ x_j = l}} \prod_{i=1}^j A_i(x_i) x_i^{x_i} = \sum_{l=0}^s A_j(l) x_j^l \times \\ \sum_{\mathcal{J}(s-l, j-1)} \prod_{i=1}^{j-1} A_i(x_i) x_i^{x_i} = \sum_{l=0}^s A_j(l) x_j^l g(s-l, j-1). \quad (17.27)$$

Рекуррентное уравнение (17.27) при начальных условиях (17.25) и (17.26) дает алгоритм последовательного вычисления величины $g(s, j)$ вплоть до $g(N, n) = G(N) = C^{-1}(N)$.

Зная стационарные вероятности состояний сети МО, можно определить основные характеристики отдельных СМО и сети МО в целом.

Рассмотрим разомкнутую экспоненциальную сеть МО, содержащую в общем случае многолинейные СМО, которые, как установлено, можно рассматривать изолированно как СМО типа $M/M/k_i$. Среднее число запросов L_i , ожидающих обслуживания в очереди i -й СМО (средняя длина i -й очереди), вычисляется непосредственно как математическое ожидание дискретной случайной величины — числа запросов в очереди. Из (17.4) нетрудно получить

$$L_i = \sum_{x_i=1}^{\infty} x_i P_i(x_i) = \left[\left(\frac{\lambda_i}{\mu_i} \right)^{k_i+1} / k_i! k_i \left(1 - \frac{\lambda_i}{k_i \mu_i} \right)^2 \right] P_i(0), \quad (17.28)$$

где $P_i(0)$ определяется из (17.6).

Среднее число занятых устройств в i -й СМО

$$K_i = \lambda_i / \mu_i. \quad (17.29)$$

Среднее число запросов в i -й СМО

$$M_i = L_i + K_i. \quad (17.30)$$

Среднее время пребывания запросов в i -й СМО

$$T_i = M_i / \lambda_i. \quad (17.31)$$

Характеристики разомкнутой сети МО в целом определяются на основании формул (17.28) — (17.31). Среднее число

запросов, ожидающих обслуживания в сети, и среднее число запросов в сети МО находятся суммированием соответствующих величин по всем СМО сети.

Поскольку каждый запрос проходит через i -ю СМО в среднем e_i раз, то среднее время пребывания в сети МО

$$T = \sum_{i=1}^n e_i T_i. \quad (17.32)$$

Для замкнутых сетей МО среднее число запросов i -го класса и общее среднее число запросов, находящихся в i -й СМО, определяются по формулам

$$M_i^{(r)} = \sum_{j=0}^{N_r} j P[x_{ri} = j] \text{ и } M_i = \sum_{j=0}^N j P[x_i = j], \quad (17.33)$$

где вероятность того, что в СМО находится j запросов i -го класса,

$$P[x_{ri} = j] = \sum_{x_{ri}=j} \prod (x_1; \dots; x_n), \text{ а } x_i = \sum_{r=1}^R x_{ri}.$$

Среднее число запросов, находящихся в очереди i -й СМО,

$$L_i = \sum_{j=k_i+1}^N (j - k_i) P[x_i = j]. \quad (17.34)$$

Среднее число устройств, занятых в i -й СМО, равно разности между числом имеющихся устройств и средним числом простаивающих устройств:

$$K_i = k_i - \sum_{j=0}^{k_i-1} (k_i - j) P[x_i = j]. \quad (17.35)$$

Интенсивность потока запросов, выходящих из i -й СМО, определяет фактическую производительность СМО и равна $K_i \mu_i$, где μ_i — интенсивность обслуживания запросов одним устройством. Отсюда среднее время пребывания запросов в i -й СМО

$$T_i = M_i / K_i \mu_i. \quad (17.36)$$

Зная характеристики отдельных СМО, по аналогии с разомкнутой сетью МО легко определить характеристики замкнутой сети МО в целом или ее отдельных участков. В последующих па-

раграфах будут рассмотрены примеры построения на основе сетей МО моделей функционирования ВС и вычислительных сетей.

17.3. МОДЕЛЬ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ

В гл. 13 рассматривалась однопроцессорная вычислительная система коллективного пользования с разделением времени (ВСКП) с одноочередной дисциплиной обслуживания запросов пользователей (см. рис. 13.8).

При построении модели системы будем предполагать, что случайное время работы пользователя за пультом t_n и случайная длительность обработки запроса процессором (без учета времени пребывания в очереди к процессору) $t_{об}$ распределены по экспоненциальному закону со средними значениями соответственно $t_{n, ср}$ и $t_{об, ср}$. Пользователь не посылает нового запроса в систему, не получив ответа на предыдущий.

Считаем, что переключение программ происходит мгновенно и потери времени («издержки») на переключение отсутствуют.

Модель ВСКП строится с целью определения следующих основных характеристик: среднего числа пользователей, ожидающих ответа, и средней продолжительности ожидания ответа на запрос пользователя. Исходными параметрами являются среднее время подготовки пользователем запроса $t_{n, ср} = 1/\mu_1$, средняя длительность обработки запроса процессором $t_{об, ср} = 1/\mu_2$, число пользователей N за пультами.

Модель ВСКП представляет собой замкнутую сеть МО с двумя СМО (рис. 17.6): N -линейной без очереди (пультовая фаза) и однолинейной (системная фаза). Состояние этой сети МО определяется парой $(N - j, j)$, где j — число запросов, находящихся в данный момент в системной фазе, а $N - j$ — число запросов в пультовой фазе.

Стационарные вероятности состояний данной сети МО в соответствии с (17.17) определяются следующим выражением:

$$P(N - j, j) = C(N) / (N - j)! \mu_1^{N-j} \mu_2^j. \quad (17.37)$$

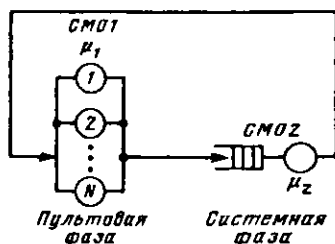


Рис. 17.6. Модель системы разделения времени

В данном случае легко получить компактное выражение для нормировочной константы $C(N)$. Из (17.37) после преобразования находим

$$\begin{aligned} \Pi(N-j, j) &= C(N) \left(\frac{1}{N! \mu_1^N} \right) \cdot \frac{N!}{(N-j)!} \cdot \left(\frac{\mu_1}{\mu_2} \right)^j = \\ &= \Pi(N, 0) \frac{N!}{(N-j)!} \cdot \left(\frac{\mu_1}{\mu_2} \right)^j. \end{aligned} \quad (17.38)$$

В соответствии с условием нормировки $\sum_{j=0}^N \Pi(N-j, j) = 1$.

Поэтому вероятность незанятости процессора

$$\Pi(N, 0) = \left[\sum_{i=0}^N \frac{N!}{(N-i)!} \cdot \left(\frac{\mu_1}{\mu_2} \right)^i \right]^{-1}. \quad (17.39)$$

Из (17.38) и (17.39) окончательно получаем

$$\Pi(N-j, j) = \frac{N!}{(N-j)!} \cdot \left(\frac{\mu_1}{\mu_2} \right)^j \cdot \sum_{i=0}^N \frac{N!}{(N-i)!} \cdot \left(\frac{\mu_1}{\mu_2} \right)^i. \quad (17.40)$$

Среднее число пользователей, ожидающих ответа, равно среднему числу запросов, находящихся в СМО2. В соответствии с (17.33)

$$M_2 = \sum_{j=0}^N j \Pi(N-j, j) = \sum_{j=0}^N \frac{jN!}{(N-j)!} \cdot \left(\frac{\mu_1}{\mu_2} \right)^j \cdot \sum_{i=0}^N \frac{N!}{(N-i)!} \cdot \left(\frac{\mu_1}{\mu_2} \right)^i. \quad (17.41)$$

Среднее время ожидания ответа пользователем равно среднему времени пребывания запроса в системной фазе, т. е. в СМО2, и находится из (17.36):

$$T_2 = M_2 / \mu_2. \quad (17.42)$$

17.4. МОДЕЛИ КОНФЛИКТОВ В ПАМЯТИ МНОГОПРОЦЕССОРНЫХ ВС

В гл. 15 при изложении вопросов построения многопроцессорных ВС отмечалась острота проблемы снижения производительности системы из-за конфликтов при обращении процессоров к общей памяти. Модели конфликтов позволяют оценивать влияние последних на производительность ВС, сравнивать с этой точки зрения различные структурные варианты построения ВС.

Если межмодульные связи в многопроцессорной системе реализованы на основе полносвязной коммутации (полносвязный

интерфейс «процессоры – ОП»), то конфликты возникают только тогда, когда два или более процессоров одновременно обращаются к одному модулю ОП. Если в системе реализован интерфейс в виде общей шины, к которой подключены все модули процессоров и памяти, то конфликты возникают в общей шине, когда два или более процессоров одновременно обращаются к ОП (даже если они обращаются к разным модулям памяти). Конфликты означают непроизводительные простои процессоров в ожидании, когда освободится модуль ОП или общая шина.

Аналитические модели конфликтов, основанные на теории сетей МО, позволяют оценить влияние конфликтов в памяти на производительность ВС с учетом таких параметров, как степень связности интерфейса «процессоры – ОП», число модулей ОП, число процессоров, длительность цикла ОП, среднее время подготовки процессором запроса к ОП, и оценить влияние дополнительных блоков местной памяти на эффективность работы ВС. При этом оцениваются следующие показатели: среднее число процессоров, простаивающих из-за конфликтов, фактическая производительность системы, процент потерь производительности системы из-за конфликтов.

Модель конфликтов в многопроцессорной ВС с полностью связным интерфейсом

Рассмотрим систему, состоящую из m процессоров и n модулей ОП, связанных полностью связным интерфейсом (рис. 17.7, а). Процессор не посылает новый запрос в ОП, пока не будет обслужен уже выставленный им ранее запрос. После того как ОП удовлетворяет запрос процессора, следующий запрос данного процессора к ОП происходит через интервал времени, имеющий экспоненциальное распределение с параметром ν .

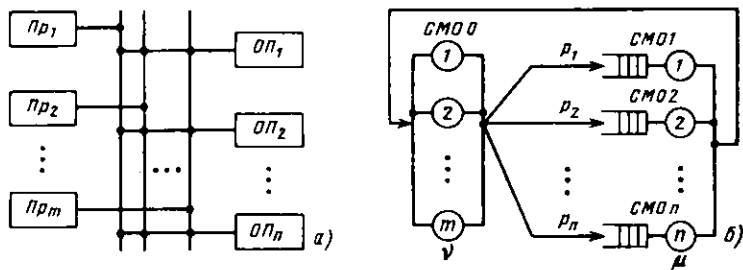


Рис. 17.7. Многопроцессорная система с полностью связным интерфейсом «процессоры – ОП» (а) и ее модель (б)

Этот параметр одинаков для всех процессоров и равен $1/t_{п.ср}$, где $t_{п.ср}$ — среднее время подготовки запроса процессором. Пусть p_i — вероятность того, что процессор обращается к i -му модулю ОП. Предполагаем, что длительность обслуживания запроса в любом модуле ОП имеет экспоненциальное распределение с параметром $\mu = 1/t_{об.ср}$, где $t_{об.ср}$ — среднее время обслуживания запроса.

В рассматриваемом случае каждый процессор может обратиться к некоторому модулю ОП по собственной шине. При наличии запросов к данному модулю от нескольких процессоров (конфликтная ситуация) возникает очередь запросов к модулю ОП. Обслуживание запросов осуществляется в порядке их поступления.

Рассматриваемая многопроцессорная система моделируется замкнутой сетью МО (рис. 17.7, б), в которой СМО с m обслуживающими устройствами соответствует m процессорам, а каждому из n модулей ОП соответствует однолинейная СМО с очередью. Число запросов в сети МО совпадает с числом процессоров и равно m . Поэтому в СМО очереди нет.

Пусть x_0 — число процессоров, подготавливающих запросы в данный момент времени, а x_i , $i = 1, n$, — число запросов в i -й СМО, соответствующей i -му модулю ОП, тогда

$$\sum_{j=0}^n x_j = m.$$

Из (17.17) следует, что стационарная вероятность состояния $(x_0, \dots, x_n)$ данной сети МО

$$P(x_0, \dots, x_n) = C(m) \left[\left(\frac{1}{x_0! \nu^{x_0}} \right) \prod_{i=1}^n \left(\frac{p_i}{\mu} \right)^{x_i} \right]. \quad (17.43)$$

Пусть $l = \sum_{i=1}^n x_i$ — число процессоров, обратившихся к ОП.

Тогда $x_0 = m - l$. Выражение (17.43) по аналогии с (17.38) можно переписать в виде

$$P(x_0, \dots, x_n) = P(m, 0, \dots, 0) \frac{m!}{(m-l)!} \prod_{i=1}^n \left(\frac{p_i \nu}{\mu} \right)^{x_i}. \quad (17.44)$$

Для случая когда $p_i = 1/n$, $i = 1, n$, из условия нормировки трудно получить по аналогии с (17.39), что

$$P(m, 0, \dots, 0) = \left[\sum_{j=0}^m \binom{m}{j} \frac{(n+j-1)!}{(n-1)!} \left(\frac{\nu}{\mu n} \right)^j \right]^{-1}. \quad (17.45)$$

Среднее число процессоров L_k , простаивающих из-за конфликтов, равно среднему числу запросов, ожидающих в очередях однолинейных СМО. Очевидно,

$$L_k = \sum_{i=1}^n L_i, \quad (17.46)$$

где L_i вычисляется по формуле (17.34) при $k_i = 1$, $i = 1, n$.

В силу однородности процессоров вероятность простоя из-за конфликтов для одного процессора

$$f_n = L_k/m. \quad (17.47)$$

Это означает, что на произвольном интервале времени T каждый из процессоров будет простаивать в среднем в течение времени $T_n = f_n T$. Пусть система «процессор – ОП» при отсутствии конфликтов выполняет одну операцию в среднем за время T_0 . Тогда фактическая производительность многопроцессорной системы в целом (среднее число операций в единицу времени) определяется выражением

$$V = m(1 - f_n)/T_0. \quad (17.48)$$

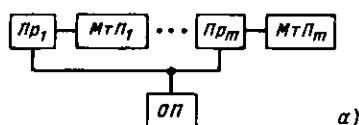
Потери производительности системы (в процентах) за счет конфликтов составляют $\delta = 100 f_n$.

Модель конфликтов в многопроцессорной ВС с односвязным интерфейсом «процессоры – ОП» и блоками местной памяти (МтП).

Пусть каждый блок МтП доступен только собственному процессору и при обращении к нему конфликтов не возникает (рис. 17.8, а).

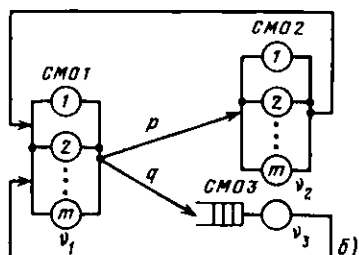
Предполагается, что число процессоров и блоков МтП одинаково и равно m . Пусть p – вероятность того, что запрос процессора адресован к собственному блоку МтП, а времена подготовки запроса в процессоре и обслуживания запроса в ОП и МтП являются экспоненциально распределенными величинами со средними значениями $t_{п_ср}$, $t_{оп_ср}$ и $t_{м_п_ср}$ соответственно.

Многопроцессорная система моделируется сетью МО из трех СМО (17.8, б): СМО1 соответствует подготовке запросов в процессорах, СМО2 – обслуживанию запросов в МтП, СМО3 – обслуживанию запросов в ОП. Поскольку каждый



а)

Рис. 17.8. Система с односвязным интерфейсом «процессоры – ОП» и блоками местной памяти (а) и ее модель (б).



б)

процессор может обращаться только к своему блоку МТП, то СМО1 и СМО2 являются m -линейными без очередей, а СМО3 – однолинейной с очередью. Из СМО1 запрос с вероятностью p попадает в СМО2, а с вероятностью $q = 1 - p$ – в СМО3. Из СМО2 и СМО3 запросы с вероятностью, равной

единице, возвращаются в СМО1. Интенсивности обслуживания запросов в СМО равны соответственно:

$$v_1 = t_{\text{П}_{\text{CP}}}^{-1}; \quad v_2 = t_{\text{МТП}_{\text{CP}}}^{-1}; \quad v_3 = t_{\text{ОП}_{\text{CP}}}^{-1}.$$

Из (17.17) следует, что стационарная вероятность состояния (x_1, x_2, x_3) данной сети МО

$$P(x_1, x_2, x_3) = C(m) \left[\frac{1}{x_1!} \left(\frac{1}{v_1} \right)^{x_1} \right] \left[\frac{1}{x_2!} \left(\frac{p}{v_2} \right)^{x_2} \right] \left[\frac{q}{v_3} \right]^{x_3}. \quad (17.49)$$

Для рассматриваемой сети МО выражение (17.49) можно привести к явному виду. Учитывая условие нормировки (17.18) и условие $x_1 + x_2 + x_3 = m$, перепишем (17.49) в следующем виде:

$$P(x_1, x_2, x_3) = \frac{\frac{1}{x_1!} \frac{1}{x_2!} \left(\frac{p v_1}{v_2} \right)^{x_2} \left(\frac{q v_1}{v_3} \right)^{x_3}}{\sum_{\{l_1, l_2, l_3\}} \frac{1}{l_1!} \frac{1}{l_2!} \left(\frac{p v_1}{v_2} \right)^{l_2} \left(\frac{q v_1}{v_3} \right)^{l_3}}, \quad (17.50)$$

где суммирование в знаменателе производится по всем возможным комбинациям (l_1, l_2, l_3) , для которых $l_1 + l_2 + l_3 = m$.

Дальнейшие преобразования позволяют получить следующую формулу:

$$P(x_1, x_2, x_3) = \frac{\left(\frac{v_3}{v_1 q}\right)^{x_1}}{x_1!} \frac{\left(\frac{v_3 p}{v_2 q}\right)^{x_2}}{x_2!} \frac{\alpha^j}{j!}, \quad (17.51)$$

где $\alpha = v_3(v_2 + v_1 p)/v_1 v_2 q$.

Среднее число процессоров, простаивающих из-за конфликтов, равно среднему числу запросов L_3 в очереди СМОЗ и вычисляется по формуле (17.34) с использованием (17.51).

Отсюда в силу однородности процессоров фактическая производительность многопроцессорной системы

$$V = m(1 - L_3/m)/T_0. \quad (17.52)$$

17.5. МОДЕЛЬ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ

При проектировании вычислительных сетей (ВСт) (см. гл. 16) приходится решать такие вопросы, как, например, выбор структуры связи между узлами сети, характеристик каналов связи, параметров управления потоками данных в ВСт и т. д. Важно исследовать, как изменяются основные показатели эффективности функционирования ВСт в зависимости от нагрузки на нее, т. е. в зависимости от числа пакетов, одновременно транспортируемых в ВСт, и определить «узкие» места в ВСт, т. е. узлы, в которых образуются относительно длинные очереди пакетов.

Основными показателями эффективности функционирования сети являются среднее время, затрачиваемое на транспортировку пакета от отправителя к адресату (средняя задержка пакета), и пропускная способность ВСт в целом или ее отдельных участков. Кроме того, важно оценить средние длины очередей и времена пребывания пакетов в отдельных узлах сети.

Исходными параметрами при анализе ВСт являются ее структура, характеристики узлов и каналов связи, ограничения, налагаемые на число пакетов, одновременно транспортируемых в ВСт.

Обычно моделью ВСт является сеть МО той же топологической структуры. Каждое направление передачи данных (выходной канал связи) в главных вычислительных машинах (ГВМ) и связанных процессорах (СП) моделируется системой массового обслуживания с очередью, образуемой в буферной

памяти ГВМ или СП. Если выходной канал связи содержит k независимых подканалов, то он моделируется k -линейной СМО.

В модели длительность обслуживания пакетов каждым обслуживающим устройством учитывает как время передачи пакета по каналу связи, так и время его обработки (коммутации) непосредственно в узле сети.

При моделировании ВСт с малым числом узлов могут использоваться более детальные модели, в которые включаются СМО, отражающие работу отдельных сетевых программных и аппаратных компонентов.

Для согласования скоростей работы отправителя пакетов и адресата и для ограничения нагрузки на ВСт в каждом логическом соединении отправитель — адресат устанавливается предельное число транспортируемых пакетов.

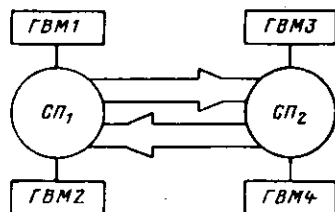
Пусть в ВСт работают R однонаправленных логических соединений отправитель — адресат данных. Пакеты передаются по фиксированным маршрутам. С каждым r -м соединением связано максимальное число одновременно транспортируемых пакетов N_r , $r = \overline{1, R}$. ВСт необходимо анализировать в условиях максимальных требований отправителя к ресурсам. Поэтому обычно считается, что у отправителя в любой момент найдется пакет для отправки адресату. Когда в ВСт достигнуто максимально возможное число одновременно транспортируемых пакетов, принадлежащих данному логическому соединению, то процесс генерации пакетов отправителем приостанавливается.

Предполагается, что на каждый поступивший в узел-адресат пакет отсылается квитанция в обратном направлении. Поскольку число пакетов в каждом логическом соединении ограничено, то число пакетов и квитанций, принадлежащих r -му соединению, в сумме постоянно и равно N_r .

Следовательно, каждая пара разомкнутых *цепочек* из СМО, моделирующих прямой путь пакетов и обратный путь квитанций некоторого r -го логического соединения, преобразуется в замкнутую сеть МО, в которой циркулирует N_r запросов.

Каждая такая сеть МО в общем случае содержит СМО, моделирующие отправителя пакетов (абонентов, являющихся источником пакетов в ГВМ), обслуживание пакетов сетевым обеспечением ГВМ, канал доступа в сеть передачи данных (канал ГВМ — СП), транзитные узлы и каналы связи, канал СП — ГВМ, адресата пакетов, обслуживание пакетов сетевым обеспечением в ГВМ адресата и аналогично обратное направление адресат — отправитель. На прямом пути в такой сети

Рис. 17.9. Вычислительная сеть СЕКОП



МО запросы представляют собой пакеты с данными, а на обратном — квитанции.

Учитывая, что квитанции имеют небольшой размер и обладают приоритетом в обслуживании, задержка квитанций в сети обычно моделируется одной СМО для каждого логического соединения.

Соответствующей моделью ВСт в целом будет замкнутая сеть МО, в которой циркулируют разнородные запросы (пакеты), принадлежащие R классам.

Запросы разных классов различаются маршрутами следования по сети МО. Предполагается, что длительности обслуживания запросов в отдельных СМО имеют экспоненциальное распределение.

В качестве примера рассмотрим вычислительную сеть СЕКОП (рис. 17.9) Института прикладной математики АН СССР и ее модель (рис. 17.10) [15]. Сеть состоит из двух СП, соединенных десятью симплексными каналами: по пять в каждую сторону. К каждому СП подключены по две ГВМ.

Для анализа работы СЕКОП выделим два логических соединения: $ГВМ_1 \rightarrow СП_1 \rightarrow СП_2 \rightarrow ГВМ_3$ и $ГВМ_2 \rightarrow СП_1 \rightarrow СП_2 \rightarrow ГВМ_4$. На рис. 17.10 представлена соответствующая замкнутая сеть МО с двумя классами запросов, в которой $СМО1$, $СМО5$, $СМО9$, $СМО10$ моделируют обслуживание пакетов сетевым обеспечением ГВМ, $СМО3$ и $СМО6$ моделируют обслуживание пакетов в СП₁ и СП₂; каналы межмашинного

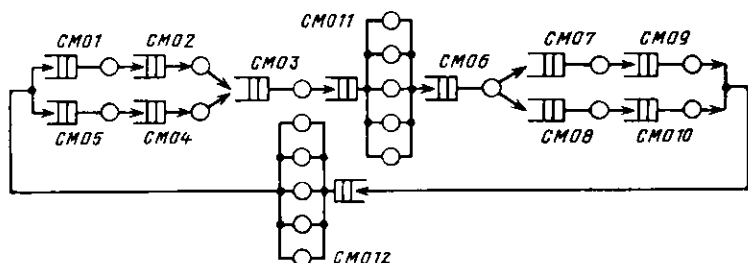


Рис. 17.10. Модель вычислительной сети СЕКОП

обмена ГВМ – СП моделируются *СМ02*, *СМ04*, *СМ07* и *СМ08*; два направления передачи СП – СП моделируются 5-линейными *СМ011* и *СМ012*.

Каналы межмашинного обмена ГВМ – СП и СП обслуживают как пакеты с данными в прямом направлении, так и квитанции в обратном направлении. Чтобы это учитывалось, средняя длительность обслуживания запросов в *СМ03* и *СМ06* удваивается, а в *СМ02*, *СМ04*, *СМ07* и *СМ08* увеличивается на время передачи квитанции по каналу межмашинного обмена.

В обратном направлении запросы проходят только *СМ012*.

Характеристики рассматриваемой сети МО вычисляются по формулам (17.20), (17.33) – (17.36). Приведенная модель позволяет оценить взаимное влияние логических соединений на ресурсы сети передачи данных в зависимости от значений нагрузок N_1 и N_2 .

СПИСОК ЛИТЕРАТУРЫ

1. Авен О. И., Гурин Н. Н., Коган Я. А. Оценка качества и оптимизация вычислительных систем. — М.: Наука, 1982. — 464 с.
2. Адасько В. И., Каган Б. М., Пац В. Б. Основы проектирования запоминающих устройств большей емкости. — М.: Энергоатомиздат, 1984. — 285 с.
3. Алексенко А. Г., Шагурин В. И. Микросхемотехника. — М.: Радио и связь, 1982. — 416 с.
4. Альянах И. А. Внешние запоминающие устройства ЕС ЭВМ. М.: Советское радио, 1979. — 208 с.
5. Амдаль Дж., Блоу Дж., Брук Ф. Архитектура системы IBM-360. — Кибернетический сборник, 1965, № 1 (новая серия), с. 101 — 136.
6. Анисимов Б. В., Петров В. Я. Организация вычислительных процессов в ЦВМ. — М.: Высшая школа, 1977. — 420 с.
7. Анисимов Б. В., Савельев А. Я. Проектирование электронных вычислительных машин. — М.: Машиностроение, 1971. — 420 с.
8. Анисимов Б. В., Четвериков В. Н. Основы теории и проектирования ЭЦВМ. — М.: Высшая школа, 1970. — 576 с.
9. Артамонов Г. Т. Анализ производительности ЦВМ методами теории массового обслуживания. — М.: Энергия, 1979. — 176 с.
10. Архитектура многопроцессорных вычислительных систем / О. С. Козлов, Е. А. Мелникий, А. В. Экало и др.: Под ред. В. И. Тимохина. — Л.: Изд-во ЛГУ, 1981. — 104 с.
11. Балашев Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы: Под ред. В. Б. Смолова. — М.: Радио и связь, 1981. — 328 с.
12. Баранов С. И. Синтез микропрограммных автоматов. — Л.: Энергия, Ленингр. отделение, 1974. — 216 с.
13. Бесков Б. А., Соловьев В. П. Микропрограммное управление. — Методические указания к практическим и лабораторным занятиям: Под ред. Б. М. Кагана. — М.: МИИТ, 1975. — 40 с.
14. Богуславский Л. Б. Управление потоками данных в сетях ЭВМ. — М.: Энергоатомиздат, 1984. — 150 с.
15. Богуславский Л. Б., Дрожжинов В. И., Семенова Т. А. Исследование сети СЕКОП с помощью моделирования и измерений. — Автоматика и вычислительная техника, 1983, № 2, с. 2 — 12.
16. Богуславский Л. Б., Коган Я. А. Аналитическое исследование алгоритмов замещения страниц в двухуровневой памяти ЦВМ. — Автоматика и телемеханика, 1974, № 11, с. 129 — 136.
17. Бурцев В. С. Принципы построения многопроцессорных вычислительных комплексов «Эльбрус». — М.: ИТМ и ВТ. Препринт № 1, 1977. — 39 с.

18. Бутрименко А. В. Разработка и эксплуатация сетей ЭВМ. — М.: Финансы и статистика, 1981. — 256 с.
19. Вейцман К. Распределенные системы мини- и микро-ЭВМ. — М.: Финансы и статистика, 1983. — 382 с.
20. Веремева В. С., Максимов Ю. А., Макурочкин В. Г. Внешние запоминающие устройства. — Радиоэлектроника в 1981 году. — 1982, с. 25—41.
21. Вычислительные машины с развитыми системами интерпретации/В. М. Глушков, А. А. Барабанов, Л. А. Калининченко и др. — Киев: Наукова думка, 1970. — 260 с.
22. Глушков В. М. Синтез цифровых автоматов. — М.: Физматгиз, 1962. — 467 с.
23. Грамолин В. В. Полупроводниковая динамическая память. — Методические указания к лабораторным работам. — М.: МИИТ, 1977. — 30 с.
24. Грамолин В. В., Перисев В. Г. Синтез комбинационных схем на мультиплексорах. — Методические указания к лабораторной работе. — М.: МИИТ, 1980. — 31 с.
25. Грицевский П. М., Мамченко А. Е., Стененский Б. М. Основы автоматики, импульсной и вычислительной техники. — М.: Советское радио, 1979. — 392 с.
26. Димитров А. А., Каган Б. М., Крейнин А. Я. Вероятностные аналитические модели каналов ввода-вывода вычислительных систем. — Автоматика и вычислительная техника, 1981, № 2, с. 78—83.
27. Дроздов Е. А., Комарницкий В. А., Пятибратов А. П. Электронные вычислительные машины единой системы. — М.: Машиностроение, 1981. — 648 с.
28. Дружинин Г. В. Надежность автоматизированных систем. — М.: Энергия, 1977. — 576 с.
29. Дэвис Д., Барбер Д. Сети связи для вычислительных машин. — М.: Мир, 1976. — 680 с.
30. Жимерин Д. Г., Мясников В. А. Автоматизированные и автоматические системы управления. — М.: Энергия, 1975. — 680 с.
31. Заренин Ю. Г. Корректирующие коды для передачи и переработки информации. — Киев: Техника, 1965. — 170 с.
32. Каган Б. М. Электронные вычислительные машины и системы. — М.: Энергия, 1979. — 528 с.
33. Каган Б. М., Адасько В. И., Пуэр Р. Р. Запоминающие устройства большой емкости. — М.: Энергия, 1968. — 320 с.
34. Каган Б. М., Мкртумян И. Б. Об одном подходе к проектированию систем автодиагностики ЦВМ. — Вопросы радиоэлектроники, сер. ЭВТ, 1971, вып. 8, с. 42—51.
35. Каган Б. М., Каневский М. М. Цифровые вычислительные машины и системы. — М.: Энергия, 1973. — 680 с.
36. Каган Б. М., Михайлюк А. В. Язык описания цифровых вычислительных устройств и машин. — Методические указания к практическим занятиям. — М.: МИИТ, 1975. — 44 с.
37. Каган Б. М., Воителев А. И., Лукьянов Л. М. Системы связи УВМ с объектом управления в АСУ ТП. — М.: Советское радио, 1977. — 320 с.
38. Каган Б. М., Сташин В. В. Микропроцессоры в цифровых системах. — М.: Энергия, 1979. — 193 с.
39. Каган Б. М., Крейнин А. Я. Модели конфликтов в памяти мультипроцессорных систем. — Автоматика и вычислительная техника, 1982, № 2, с. 59—65.

40. Каган Б. М., Мкртумян И. Б. Основы эксплуатации ЭВМ. — М.: Энергия, 1983. — 376 с.
41. Каган Б. М., Першеев В. Г. Систематизация структур адресных ЗУ. — В кн.: Микропроцессорные устройства и системы. — М.: МИИТ, 1984, вып. 710, с. 31—41.
42. Карцев М. А. Арифметика цифровых машин. — М.: Наука, 1969. — 575 с.
43. Карцев М. А. Архитектура цифровых вычислительных систем. — М.: Наука, 1978. — 296 с.
44. Королев Л. Н. Структура ЭВМ и их математического обеспечения. — М.: Наука, 1978. — 352 с.
45. Козн К. А. Многопроцессорная архитектура для обработки транзакций. — Электроника, 1983, № 2, с. 26—32.
46. Катцан Г. Вычислительные машины системы 370. — М.: Мир, 1974. — 508 с.
47. Клейнрок Л. Вычислительные системы с очередями. — М.: Мир, 1979. — 600 с.
48. Коффон Дж. Технические средства микропроцессорных систем: Практический курс. — М.: Мир, 1983. — 344 с.
49. Ларионов А. М. Основные концепции развития ЕС ЭВМ. — В кн.: Вычислительная техника социалистических стран/Под ред. М. Е. Раковского. — М.: Статистика, 1977, с. 41—51.
50. Лебедев С. А. Электронные вычислительные машины. — Доклад из сессии АН СССР по научным проблемам автоматизации производства. Пленарное заседание. — М.: Изд-во АН СССР, 1957. — 20 с.
51. Лебедев С. А., Мельников В. А. Общее описание БЭСМ и методики выполнения операций. — М.: Физматгиз, 1959. — 250 с.
52. Майоров С. А., Новиков Г. А. Структура цифровых вычислительных машин. — М.: Машиностроение, 1970. — 480 с.
53. Майоров С. А., Новиков Г. А. Принципы организации цифровых машин. — Л.: Машиностроение, 1974. — 432 с.
54. Малые ЭВМ и их применение Ю. А. Дедов, М. А. Островский, К. В. Песелев и др.; Под общей ред. Б. Н. Наумова. — М.: Статистика, 1980. — 231 с.
55. Мануэль Т. Планы Японии по захвату лидерства в области вычислительной техники. — Электроника, 1981, № 23, с. 72—74.
56. Михновский С. Д., Шор Н. З. Оценка минимального числа пересылок при динамическом распределении страничной памяти. — Кибернетика, 1965, № 5, с. 30—40.
57. Мультипроцессорные системы и параллельные вычисления/Под ред. Ф. Г. Энслоу. — М.: Мир, 1976. — 384 с.
58. Мультипроцессорные вычислительные системы/Под ред. А. Я. Хетагурова. — М.: Энергия, 1971. — 320 с.
59. Мячев А. А. Организация управляющих вычислительных систем. — М.: Энергия, 1980. — 272 с.
60. Мячев А. А. Организация ввода-вывода. — М.: Энергия, 1983. — 210 с.
61. Новик Г. Х., Животова Т. М., Зыкина Н. В. Интегральная полупроводниковая память. — Методические указания к лабораторной работе. — М.: МИИТ, 1977. — 36 с.
62. Основы теории вычислительных систем/Под ред. С. А. Майорова. — М.: Высшая школа, 1978. — 408 с.
63. Папернов А. А. Логические основы цифровой вычислительной техники. — М.: Энергия, 1972. — 592 с.
64. Пашкеев С. Д. Основы мультипрограммирования для специа-

лизованных вычислительных систем. — М.: Советское радио, 1972. — 184 с.

65. Пархоменко П. П., Согомонян Е. С. Основы технической диагностики. — М.: Энергоиздат, 1981. — 304 с.

66. Пospelов Д. А. Арифметические основы вычислительных машин дискретного действия. — М.: Высшая школа, 1970. — 308 с.

67. Пospelов Д. А. Логические методы анализа и синтеза схем. — М.: Энергия, 1968. — 328 с.

68. Пospelов Д. А. Введение в теорию вычислительных систем. — М.: Советское радио, 1972. — 280 с.

69. Прагншвили И. В. Микропроцессоры и микро-ЭВМ. — М.: Энергия, 1979. — 232 с.

70. Прагншвили И. В., Виленкин С. Я., Медведев И. Л. Параллельные вычислительные системы с общим управлением. — М.: Энергоатомиздат, 1983. — 312 с.

71. Преснухин Л. Н., Нестеров П. В. Цифровые вычислительные машины. — М.: Высшая школа, 1981. — 580 с.

72. Прихляковский В. В., Ломов Ю. С. Технические и программные средства Единой системы ЭВМ (ЕС ЭВМ-2). — М.: Статистика, 1980. — 232 с.

73. Применение вычислительных машин для проектирования цифровых устройств (сб. статей). Под ред. Н. Я. Матюхина. — М.: Советское радио, 1968. — 266 с.

74. Принципы работы системы IBM-370. — М.: Мир, 1975. — 576 с.

75. Проблемы создания управляющих комплексов повышенной живучести. — Доклад на Всемирном электротехническом конгрессе/ В. М. Долгарт, Б. М. Каган, М. М. Каневский и др. — М.: ВЭЛК, 1977. — 23 с.

76. Проектирование цифровых вычислительных машин/Под ред. С. А. Майорова. — М.: Высшая школа, 1972. — 344 с.

77. Пшеничников Л. Е., Сахин Ю. Х. Архитектура многопроцессорного вычислительного комплекса «Эльбрус». — М.: ИТМ и ВТ. Препринт № 7, 1977. — 39 с.

78. Рамеев Б. И. Универсальная автоматическая цифровая машина типа «Урал». — Конференция «Пути развития советского математического машиностроения и приборостроения», ч. 1, изд. ВИНТИ, 1956, с. 38—52.

79. Ратнер Д., Лэтгин У. 32-разрядный микропроцессор с архитектурой, ориентированной на язык АДА. — Электроника, 1981, № 4, с. 119—126.

80. Роландер Т. А. Микрокомпьютерные программные средства для локальных сетей. — Электроника, 1982, № 2, с. 30—36.

81. Самофалов К. Г., Корнейчук В. И., Тарасенко В. П. Цифровые электронные вычислительные машины. — Киев: Вища школа, 1983. — 455 с.

82. Система математического обеспечения ЕС ЭВМ В. Г. Лесюк, А. С. Марков, Г. В. Пеледов, Л. Д. Райков; Под ред. А. М. Ларионова. — М.: Статистика, 1974. — 216 с.

83. Соловьев Г. Н. Арифметические устройства ЭВМ. — М.: Энергия, 1978. — 176 с.

84. Соловьев Г. Н., Никитин В. Д. Операционные системы цифровых вычислительных машин. — М.: Машиностроение, 1977. — 112 с.

85. Смолов В. Б., Барашенко В. В., Байков В. Д. Специализированные ЭВМ/Под ред. В. Б. Смолова — М.: Высшая школа, 1981. — 279 с.

86. **Справочник по цифровой вычислительной технике**/Под ред. Б. Н. Малиновского. — Киев.: Техника, 1974. — 512 с.
87. **Сурков Л. В.** Сети электронных вычислительных машин. — М.: изд. МВТУ, 1977. — 56 с.
88. **Турбер К. Дж., Фримен Х. А.** Много изготовителей — много локальных сетей. — *Электроника*, 1982, № 2, с. 21—30.
89. **Универсальная быстродействующая вычислительная машина М-3/Б. М. Каган, Н. Я. Матюхин, В. В. Бельинский, В. М. Долкарт, Г. П. Лопато.** — Конференция «Пути развития советского математического машиностроения», ч. 1, изд. ВИНТИ, 1956, с. 70—78.
90. **Флорес А.** Внешние устройства ЭВМ. — М.: Мир, 1977. — 550 с.
91. **Уэсли Д. Х.** Высоконадежная система с тройным резервированием для управления технологическими процессами. — *Электроника*, 1983, № 2, с. 32—39.
92. **Хассон С.** Микропрограммное управление. Вып. 1 и 2. — М.: Мир, 1974. — 240 с. и 477 с.
93. **Хендри Г.** Полностью аппаратное резервирование без простоя программ. — *Электроника*, 1983, № 2, с. 39—43.
94. **Четвериков В. Н.** Кодирование и преобразование информации. — М.: Высшая школа, 1975. — 320 с.
95. **Чирков М. К., Шауман А. М.** Основы функциональной структуры вычислительных машин. — Л.: Изд-во ЛГУ, 1974. — 268 с.
96. **Чу Я.** Организация ЭВМ и микропрограммирование. — М.: Мир, 1975. — 592 с.
97. **Шерр А.** Анализ вычислительных систем с разделением времени. — М.: Мир, 1970. — 136 с.
98. **Шигин А. Г., Дерюгин А. А.** Цифровые вычислительные машины (память ЦВМ). — М.: Энергия, 1975. — 536 с.
99. **Электронная вычислительная машина ЕС-1020**/Под общ. ред. А. М. Ларионова. — М.: Статистика, 1975. — 128 с.
100. **Электронная вычислительная машина ЕС-1030**/Под общ. ред. А. М. Ларионова. — М.: Статистика, 1977. — 256 с.
101. **Электронная вычислительная машина ЕС-1050**/Под общ. ред. А. М. Ларионова. — М.: Статистика, 1976. — 302 с.
102. **Электронная вычислительная машина ЕС-1045**/А. Т. Кучукян, Т. Е. Саркисян, И. Б. Мкртумян и др. — М.: Финансы и статистика, 1981. — 183 с.
103. **Якубайтис Э. А.** Архитектура вычислительных сетей. — М.: Статистика, 1980. — 279 с.
104. **Якубайтис Э. А.** Концепция современной вычислительной сети. — *Автоматика и вычислительная техника*, 1981, № 2, с. 3—11.
105. **Якубайтис Э. А.** Классификация вычислительных сетей. — *Автоматика и вычислительная техника*, 1983, № 1, с. 3—6.
106. **Якубайтис Э. Я.** Архитектура локальных вычислительных сетей. — *Автоматика и вычислительная техника*, 1983, № 2, с. 3—20.
107. **Bannister B. R., Whitehead D. G.** Fundamentals of digital systems. — London: McGraw-Hill, 1973. — 325 p.
108. **Baskett F., Chandy K. M., Muntz R. R.** Open, closed and mixed networks of queuing with different classes of customers. — *J. Assoc. Comp. Mach.*, 1975, vol. 22, № 2, p. 248—260.
109. **Blaauw G. A.** Digital system implementation. — N. J.: Englewood Cliffs: Prentice-Hall, 1976. — 348 p.
110. **Burke P. J.** The output of a queuing system. — *Operations Res.*, 1956, vol. 4, p. 699—704.
111. **Data Processing-Open Systems Interconnection — Basic Reference Model.** — ISO/DP 7498, August 1981. — 105 p.

112. Dixon R. C., Strole N. C., Markov J. D. A token-ring network for local data communications. — IBM System Journal, 1983, vol. 22, № 1–2, p. 47–60.

113. Doran K. W. Computer Architecture: A structured Approach. — London: Academic Press, 1979. — 244 p.

114. Gordon W. J., Newell G. F. Closed queueing Systems with exponential servers. — Operations Res., 1967, vol. 15, p. 254–265.

115. Gschwind H. W., McCluskey E. I. Design of digital computers. — Heidelberg, Springer, 1975. — 548 p.

116. Falkoff A. D., Iverson K. E. A formal description of system/360. — IBM System Journal, 1963, vol. 3, № 3, p. 19–24.

117. Flynn M. I. Some Computer Organisations and their Effectiveness. — IEEE Transactions on Computers, 1972, vol. 21(9), p. 948–960.

118. Iverson K. E. A programming Language. — N. Y.: John Willey and Sons, 1962. — 480 p.

119. Jackson J. R. Job shop-like queueing systems. — Management Sci., 1963, vol. 10, p. 131–142.

120. Leipold K., Spreen H. Organisation des Nachrichtenverkehrs zwischen Zentraleinheiten und peripheren Einheiten in Datenverarbeitungssystem. — Elektronische Rechenanlagen, 1969, № 3, p. 23–32.

121. Mano M. M. Computer System Architecture. — Englewood Cliffs. — N. Y.: Prentice-Hall, 1976. — 478 p.

122. Peterson T. Two chips endow 32-bit processor with fault-tolerant architecture. — Electronics, 1983, № 7, p. 159–164.

123. Randell B. The origins of digital computers-selected papers. — Berlin: Springer-Verlag, 1973. — 380 p.

124. Rauscher T. G., Adams P. M. Microprogramming: A Tutorial and Survey of Recent Developments. — IEEE Transactions on Computers. Vol. C29, 1980, № 1, p. 2–19.

125. Wilkes M. V. The best way to design an automatic calculating machine. Report of Manchester University Computer. In Angular Conference, 1951, p. 16–18.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ

А

- Абонентский пункт 480
- Автомат Мили 63, 159
 - Мура 63, 159
 - управляющий 115, 157
 - цифровой 61
- Автоматизация профилактических испытаний 372
- Автоматизированная система управления (АСУ) 25
- Автоматический контроль 368, 371, 385
- Автоматическое восстановление вычислительного процесса 371, 385
 - диагностирование 368, 371, 385, 387, 390
- Адаптер канал-канал 450, 452
 - межшинной связи 462
- Адрес 9
 - виртуальный 428
 - физический 428
- Адресация 187, 209, 210, 211
 - абсолютная 211
 - автодискретная 190, 210
 - автоинкрементная 190, 210
 - косвенная 190, 210
 - непосредственная 188, 211
 - относительная (базирование) 188, 211
 - регистровая 190, 210
 - стековая 192
 - укороченная 190
- Адресная структура памяти 180
- Акселератор 273
- Алгоритм деления с восстановлением остатка 143
 - без восстановления остатка 144
 - замещения 434
 - сложения (вычитания) чисел с плавающей запятой 150
 - — двоичных чисел с фиксированной запятой 127
 - умножения чисел 134
- АЛУ 9, 125, 129, 130, 135, 141
 - микропроцессора 155
 - многофункциональное 152

- Алфавит 7, 52
- Арбитраж 356, 365
- Арбитража процедура 359, 367
- Архитектура вычислительной машины 15

Б

- Байт 42, 53
 - состояния 321
- Байты уточненного состояния 345
- Безотказность 368
- Блок адреса результата 262
 - выборки адреса 262
 - — команд 262
 - операционный 115
 - прямого управления 274
 - регистрации программных событий 273
 - управляющий 115
- Блокировка контроля неправильной длины 320
- Булева переменная 63
 - функция 63

В

- Вектор прерывания 233, 240
 - (слово) состояния 226, 249
- Выравнивание порядков 150
- Вычислительная машина 7
 - — аналоговая 7
 - — цифровая 7
- Вычислительная сеть 396, 479
 - — локальная 498
 - — — кольцевая (петлевая) 501
 - — — — с вставкой регистров 504
 - — — — делением на временные сегменты 504
 - — — — циркулирующим маркером 503
 - — — шинная 500
- Вычислительная система 14, 393
 - — коллективного пользования 396, 406
 - — многомашинная 396, 441

- — многопроцессорная 396, 441
- — — типа МКОД (конвейерная) 446
- — — — МКМД 446
- — — — ОКМД (матричная) 445
- — мультипрограммная 394
- — однопрограммная 393
- — пакетной обработки 395
- — реального времени 412
- — с разделением 396, 411
- — — телеобработкой 396, 506

Г

Главная ВМ (ГВМ) 480, 495

Д

- Двоичная арифметика 41
- Двухканальный переключатель 450
- Дешифратор 76
- Дисплей 305
- графический 309
- Дисциплина обслуживания 409
- Достоверность функционирования 368

Е

Единая система ЭВМ (ЕС ЭВМ)
18, 34, 180, 205, 206, 247, 272,
344, 351, 387, 430, 449

З

- Задание 401
- Задания пункт (шаг) 401
- Задача пункта задания 401
- Запоминающий массив 97
- элемент 61, 97
- ЗУ 9, 87
- биполярное 105
- внешнее 92
- динамическое МОП 107
- на магнитных дисках 280
- — — лентах 279
- постоянное (ПЗУ) 109
- — программируемое 112
- типа 2Д 98
- — 3Д 99
- — 2,5Д 100
- — 2Д-М 102
- электромеханическое 279

И

- Индексация 199, 210
- Интеллектуальный терминал 30
- Интерфейс ввода-вывода 26
- — — ЕС ЭВМ 316, 344, 353
- "мультишина" 362

- «общая шина» 314, 355
- прямого управления 450

К

- Канала адресное слово 321
- управляющее слово 321
- Канал блокмультимплексный 26, 330, 352
- мультиплексный (байтмультимплексный) 26, 328, 330
- селекторный 26, 329, 334
- Квантование по времени 415
- — — уровню 415
- Квитирование 341
- Ключ защиты памяти 249, 421
- Код ДКОИ 53, 303
- КОИ-7 298
- КОИ-8 53
- КПК-12 299
- 8421 56
- дополнительный 50
- импульсный 59
- обратный 50
- параллельный 59
- последовательный 59
- потенциальный 58
- признака результата (ПР) 128
- прямой 50
- Команда 10, 183
- безусловного перехода 11, 196
- — — с возвратом 198
- условного перехода 11, 196
- — — по косвенному адресу 196
- прямого управления 451
- Коммутатор межмодульных связей 455
- Коммугация каналов 490
- пакетов 491
- сообщений 491
- Комплект программ технического обслуживания 14
- Комплексный коэффициент эксплуатационной надежности (коэффициент использования) 22, 376
- Конвейер операций 259
- Контроль арифметических операций 380
- по четности 374
- передач информации 373
- Корректирующий код Хэмминга 377
- — — модифицированный 379
- Косвенная адресация в канале ввода-вывода 352
- Коэффициент добротности канала ввода-вывода 337
- загрузки 513
- эффективности ЭВМ 20

Л

- Логический элемент 64
- — МОП 67
- — ТТЛ 67
- — ЭСЛ 67

М

- Малая ЭВМ 23, 27
- Мантисса числа 45
- Метод записи без возврата к нулю 285
- — с модифицированной фазовой модуляцией 287
- — — фазовой модуляцией 285
- — — частотной модуляцией 287
- Метод выполнения умножения 131
- — деления 143
- — ускорения умножения 138
- Микроадаптирование динамическое 392
- статическое 391
- Микрокоманда 162, 225
- многофазная 166
- однофазная 166
- Микроконтроллер 31
- Микрооперация 116
- Микропрограмма 158, 170, 198, 223
- Микропрограммирование 160, 222
- вертикальное 164
- вертикально-горизонтальное 165
- горизонтальное 164
- горизонтально-вертикальное 165
- Микропроцессор 18, 29, 180, 205, 215, 219, 243, 263, 269, 362, 473
- Микро-ЭВМ 18, 29
- Модель аналитическая ВС 510
- вычислительной сети 533
- ВС коллективного пользования 527
- канала ввода-вывода 336
- конфликтов в памяти ВС 528
- Модуляция 489
- Многоабонентное обслуживание памяти 423
- Многопроцессорная обработка 458
- Мультиплексор 78

Н

- Нормализация 46, 150
- Нормализованное число 46

О

- Обработка заданий в мультипрограммном режиме 403
- Общее поле ОП 452
- Окно интерфейса 464
- Операнд 10

- Операции десятичной арифметики 147
- над числами с плавающей запятой 149, 151
- Операционная система 13
- — ОС ЕС ЭВМ 398
- Операция записи 10
- считывания 10
- Основание системы счисления 38
- Отказоустойчивые ВК с автоматической реконфигурацией 465
- — — мажоритарным управлением 470
- — — двухуровневым дублированием 472

П

- Пакет «данные» 496
- «датаграмма» 496
- прикладных программ 14
- Памяти быстродействие 87
- динамическое распределение 405, 426
- емкость 87
- защита 405, 418
- сегмент 430
- фрагментация 427
- цикл обращения 88
- ширина выборки 89
- Память 9, 86
- адресная 92
- ассоциативная 95
- виртуальная 426
- внешняя 9
- ключей защиты 421
- основная (оперативная) 9, 90
- сегментно-страничная 432
- с последовательным доступом 89
- — произвольным обращением 89
- — прямым доступом 89
- стековая 96
- страничная 429
- управляющая 161
- Передача информации асинхронная 340, 489
- — синхронная 340, 489
- Переключатель шины 460
- Персональная ЭВМ (персональный компьютер) 19, 395
- Перфокорта 11, 295, 297
- Перфолента 11, 295, 297
- Печатающее устройство барабанное 303
- — знакосинхронизирующее 303
- — ценное 304
- Плавающая головка 279
- Плотность записи информации 278
- Подзадача 401
- Подканал 328
- Подразумеваемый адрес 187
- операнд 187

Т

- Такт 58
- Транспортная сеть 487
 - станция 487, 494
- Триггер 68
 - *D* 73
 - *JK* 73
 - *RS* 69
 - *T* 72
- Тн 458

У

- Управляющая программа 397
- Устройство ввода 11
 - — с перфокарт 300
 - — — перфолент 300
 - вывода 11
 - — на перфокарты 301
 - — — перфоленты 301
 - периферийное 25
 - управляющее 9

Ф

- Файл 290
- Формат команды 182, 208, 212, 215
 - распакованный (зонный) 56
 - упакованный 56

- Функция выходов 62
- переходов 62

Х

- Характеристика числа 45

Ц

- Цепочка данных 319
- операций 320

Ш

- Шина 120
- Шины индивидуальные 343
- коллективные 343

Э

- ЭВМ общего назначения 24
- специализированная 30
- Эталонная логическая модель вычислительной сети 484

Я

- Язык микроопераций 115
- Ячейка памяти 9

ОГЛАВЛЕНИЕ

Предисловие	3
Глава первая. Основные понятия	6
1.1. Два класса электронных вычислительных машин	6
1.2. Принцип действия ЭВМ	8
1.3. Понятие о системе программного (математического) обеспечения ЭВМ	12
1.4. Поколения ЭВМ	15
1.5. Основные характеристики ЭВМ	20
1.6. Основные области применения вычислительной техники и основные типы ЭВМ	23
1.6.1. ЭВМ общего назначения	24
1.6.2. Малые ЭВМ	27
1.6.3. Микропроцессоры и микро-ЭВМ	29
1.7. Понятие о системах ЭВМ. Единая система ЭВМ (ЕС ЭВМ) и система малых ЭВМ (СМ ЭВМ) социалистических стран	31
Глава вторая. Представление информации в ЭВМ	37
2.1. Позиционные системы счисления	37
2.2. Двоичная арифметика	41
2.3. Формы представления чисел в ЭВМ	42
2.4. Прямой, обратный и дополнительный коды	49
2.5. Кодирование десятичных чисел и алфавитно-цифровой информации	52
Глава третья. Логические элементы и типовые узлы	58
3.1. Представление информации физическими сигналами	58
3.2. Понятие о комбинационной схеме и цифровом автомате	60
3.3. Системы логических элементов	66
3.4. Триггеры	68
3.5. Регистры	75
3.6. Дешифраторы	76
3.7. Мультиплексоры	78
3.8. Счетчики	81
3.9. Сумматоры	83
Глава четвертая. Принципы построения устройств памяти	86
4.1. Общие сведения и классификация устройств памяти	86
4.2. Адресная, ассоциативная и стековая организации памяти	92
4.3. Структуры адресных ЗУ	97

4.4. Запоминающие устройства с произвольным обращением	104
4.5. Постоянные ЗУ	109
Глава пятая. Язык микроопераций	115
5.1. Декомпозиция вычислительного устройства на операционный и управляющий блоки. Принцип акад. В. М. Глушкова	115
5.2. Иерархия языков описания вычислительных устройств	116
5.3. Язык микроопераций	119
Глава шестая. Принципы организации арифметико-логических устройств	125
6.1. Общие сведения	125
6.2. Структура и микропрограмма АЛУ для сложения и вычитания чисел с фиксированной запятой . . .	127
6.3. Структуры и микропрограмма АЛУ для умножения чисел с фиксированной запятой	130
6.4. Методы ускорения умножения	138
6.5. Структура АЛУ для деления чисел с фиксированной запятой	141
6.6. Устройства для выполнения логических операций	145
6.7. Особенности операций десятичной арифметики . .	147
6.8. Операции над числами с плавающей запятой . . .	149
6.9. Многофункциональное АЛУ	152
6.10. Особенности АЛУ микропроцессоров	155
Глава седьмая. Управляющие автоматы	157
7.1. Общие сведения	157
7.2. Принцип действия управляющего автомата с хранимой в памяти логикой. Микропрограммное управление	160
7.3. Управляющие автоматы с «жесткой» логикой . . .	168
7.4. Программируемые логические матрицы в управляющих автоматах	178
Глава восьмая. Процессоры и микропроцессоры: элементы архитектуры	180
8.1. Предварительные замечания	180
8.2. Адресные структуры основных памятей	180
8.3. Проблема выбора структуры и формата команд. Кодирование команд	182
8.4. Способы адресации	187
8.5. Стековая адресация	192
8.6. Команды, процедуры и микропрограммы передачи управления в программах	194
8.7. Индексация	199
8.8. Сопоставление программистских моделей (регистровых структур) машин общего назначения, малых и микро-ЭВМ и микропроцессоров	204
8.9. Структура команд ЕС ЭВМ II очереди	206
8.10. Особенности адресации и системы команд 16-разрядных малых и микро-ЭВМ (СМ-4, СМ-1300, «Электроника-60»)	209
8.11. Особенности адресации и системы команд 8- и 16-разрядных микропроцессоров	214

Глава девятая. Процессоры и микропроцессоры: принципы организации	219
9.1. Назначение и структура процессора	219
9.2. Микропрограммная интерпретация языка команд ЭВМ	222
9.3. Понятие о состоянии процессора (программы). Вектор (слово) состояния	226
9.4. Принципы организации систем прерывания программ	228
9.5. Особенности систем прерывания малых ЭВМ, микропроцессоров и микро-ЭВМ	240
9.6. Система прерывания и некоторые особенности организации режимов управления в ЕС ЭВМ	247
9.7. Влияние системы прерывания на оценку эффективности вариантов структуры процессора	255
9.8. Процедура выполнения команд. Рабочий цикл процессора	256
9.9. Принцип совмещения операций академика С. А. Лебедева. Конвейер операций	259
9.10. Организация 8-разрядных микропроцессоров	263
9.11. Организация 16-разрядных микропроцессоров	269
9.12. Особенности процессоров ЭВМ общего назначения (ЕС ЭВМ)	272
Глава десятая. Периферийные устройства ЭВМ: внешние ЗУ и устройства ввода-вывода	274
10.1. Общие понятия о периферийных устройствах	274
10.2. Принципы действия внешних запоминающих устройств	275
10.3. Методы записи информации на магнитный носитель	284
10.4. Представление информации на магнитных лентах и дисках	288
10.5. Основные типы устройств ввода-вывода информации	294
10.6. Представление информации на перфолентах и перфокартах	297
10.7. Устройства ввода информации с перфолент и перфокарт	300
10.8. Устройства вывода информации на перфоленты и перфокарты	301
10.9. Печатающие устройства	302
10.10. Дисплеи	305
Глава одиннадцатая. Принципы организации систем ввода-вывода. Интерфейсы ЭВМ и микропроцессоров	309
11.1. Проблемы организации систем ввода-вывода	309
11.2. Прямой доступ к памяти	311
11.3. Основные принципы построения и структуры системы ввода-вывода	314
11.4. Основные функции каналов ввода-вывода. Управляющая информация для операций ввода-вывода	319
11.5. Основные типы и структуры каналов ввода-вывода	327
11.6. Детерминированная модель канала. Критерии эффективности каналов	336
11.7. Элементы организации интерфейсов	340
11.8. Интерфейс ввода-вывода ЕС ЭВМ	344
11.9. Развитие системы ввода-вывода в ЕС ЭВМ	
II очереди	351

11.10. Структура и временные диаграммы интерфейса «общая шина» малых ЭВМ	355
11.11. Особенности интерфейса «мультишина» микропроцессоров и микро-ЭВМ	362
Глава двенадцатая. Системы автоматического контроля и диагностирования ЭВМ	368
12.1. Основные характеристики надежности ЭВМ. Функции систем контроля и диагностирования	368
12.2. Контроль передачи информации	373
12.3. Контроль арифметических операций	380
12.4. Взаимодействие систем автоматического контроля, восстановления вычислительного процесса и диагностирования	384
12.5. Принципы построения систем автоматического диагностирования ЭВМ	387
Глава тринадцатая. Принципы организации вычислительных систем: мультипрограммные ВС, системы коллективного пользования и реального времени	393
13.1. Классификация вычислительных систем	393
13.2. Организация вычислительного процесса в мультипрограммных ВС	397
13.3. Структуры и принцип действия ВС коллективного пользования (разделения времени)	406
13.4. Вычислительные системы реального времени. Принцип действия управляющего вычислительного комплекса АСУ ТП	412
Глава четырнадцатая. Принципы организации многоуровневой памяти в мультипрограммных вычислительных системах	417
14.1. Проблемы организации памяти мультипрограммных систем	417
14.2. Защита памяти	418
14.3. Организация работы памяти в режиме многоабонентного обслуживания	423
14.4. Динамическое распределение памяти. Организация виртуальной памяти	426
14.5. Алгоритмы управления многоуровневой памятью	434
Глава пятнадцатая. Принципы организации многопроцессорных и многомашинных вычислительных систем	440
15.1. Понятие о многомашинных и многопроцессорных ВС	440
15.2. Типы структур многопроцессорных ВС, ориентированных на достижение сверхвысокой производительности	444
15.3. Методы и средства организации многомашинных и многопроцессорных вычислительных комплексов на основе ЭВМ общего назначения (ЕС ЭВМ)	449
	551

15.4. Многопроцессорная система с коммутатором междо- дульных связей («Эльбрус»)	455
15.5. Многопроцессорные и многомашинные комплексы с общей шиной (СМ ЭВМ)	459
15.6. Особенности организации отказоустойчивых много- процессорных вычислительных комплексов	465
15.7. Многомикропроцессорные комплексы на основе объ- ектно-ориентированных микропроцессоров	473
Глава шестнадцатая. Принципы организации вычисли- тельных сетей	479
16.1. Вычислительные сети. Общие сведения	479
16.2. Классификация вычислительных сетей	483
16.3. Эталонная логическая модель вычислительной сети и иерархия протоколов	484
16.4. Методы передачи данных по каналам связи. Ком- мутация каналов, сообщений, пакетов	488
16.5. Управление вычислительной сетью по протоколу X. 25	494
16.6. Локальные вычислительные сети	498
16.7. Принципы организации и средства телеобработки данных.	506
Глава семнадцатая. Вероятностные аналитические модели вычислительных систем	510
17.1. Общие принципы построения моделей ВС	510
17.2. Сети массового обслуживания	514
17.3. Модель вычислительной системы коллективного поль- зования	527
17.4. Модели конфликтов в памяти многопроцессорных ВС	528
17.5. Модель вычислительной сети	533
Список литературы	537
Алфавитный указатель	543

БОРИС МОИСЕЕВИЧ КАГАН

Электронные вычислительные машины и системы

Редактор *В. Г. Першеев*

Редактор издательства *А. Н. Гуслянская*

Художественный редактор *Т. А. Дворецкова*

Технические редакторы *О. Д. Кузнецова, А. С. Давыдова*

Корректор *З. Б. Драновская*

ИБ № 3208

Сдано в набор 24.03.84. Подписано в печать 22.02.85. Т-06667. Формат 84 × 108¹/₃₂.
Бумага хн.-журн. Гарнитура таймс. Печать высокая. Усл. печ. л. 28,98. Усл. кр.-отт. 28,98.
Уч.-изд. л. 33,45. Тираж 60000 экз. Заказ 1443. Цена 1 р. 40 к.

Энергоатомиздат, 113114, Москва, М-114, Шлюзовая наб., 10

Ордена Октябрьской Революции, ордена Трудового Красного Знамени Ленинградское
производственно-техническое объединение «Печатный Двор» имени А. М. Горького
Союзполиграфпрома при Государственном комитете СССР по делам издательства,
полиграфии и книжной торговли. 197136, Ленинград, П-136, Чкаловский пр., 15.