

В. П. Корячко

Микропроцессоры и микроЭВМ в радиоэлектронных средствах

Допущено
Государственным комитетом СССР
по народному образованию
в качестве учебника
для студентов вузов,
обучающихся по специальности
«Конструирование и технология
радиоэлектронных средств»



МОСКВА «ВЫСШАЯ ШКОЛА», 1990

ББК 32.844

К70

УДК 681.322 : 621.396.644

Рецензенты: кафедра «Конструирование и производство ЭВА» МГТУ им. Н. Э. Баумана (зав. кафедрой д-р техн. наук, проф. Б. И. Белов); кафедра вычислительной техники ЛЭТИ им. В. И. Ульянова (Ленина) (зав. кафедрой д-р техн. наук, проф. В. Б. Смоллов)

Корячко В. П.

К70 Микропроцессоры и микроЭВМ в радиоэлектронных средствах: Учеб. для вузов по спец. «Конструирование и технология радиоэлектронных средств». — М.: Высш. шк., 1990. — 407 с.; ил.

ISBN 5-06-001535-1

В книге изложены основы применения микропроцессоров и микроЭВМ в современных радиоэлектронных средствах, радиотехнических устройствах и системах различного назначения, в автоматизированных системах контроля радиоэлектронных средств и в безлюдной технологии производства радиоэлектронных средств.

К 2304030000(4309000000)—509
001(01)—90 184—90

ББК 32.844

6Ф

ISBN 5-06-001535-1

© В. П. Корячко, 1990

ПРЕДИСЛОВИЕ

Одно из главных направлений ускорения научно-технического прогресса — автоматизация технологических процессов на основе повсеместного внедрения средств микропроцессорной техники, создание современных радиоэлектронных средств со встроенными микропроцессорами и микроЭВМ, контрольно-измерительной аппаратуры на базе микропроцессоров, робототехнических комплексов и т. д.

Микропроцессоры и микроЭВМ стали массовой продукцией электронной промышленности. Низкое энергопотребление, малая материалоемкость, высокая надежность, постоянно снижающаяся стоимость и широкие функциональные возможности стали причиной и следствием бурного внедрения микропроцессоров в самые разнообразные сферы производства. Массовость этого нового класса вычислительной техники и его высокие технико-экономические параметры сделали возможным проектирование и производство новых приборов, оборудования и агрегатов с расширенными функциональными и интеллектуальными возможностями на базе встроенных микропроцессорных средств.

Знание основ микропроцессорной техники необходимо инженерам любого профиля, особенно инженерам-конструкторам и инженерам-технологам радиоэлектронных (РЭС) и электронно-вычислительных средств (ЭВС).

Микропроцессоры (МП) и микроЭВМ нашли самое широкое применение в современных радиоэлектронных устройствах и системах различного назначения, в технологических системах контроля и в безлюдной технологии производства радиоэлектронных средств. Особенно важными факторами их использования являются повышение производительности труда при изготовлении РЭС и ЭВС, улучшение качества бытовой аппаратуры (радиоприемников, магнитофонов, телевизоров и др.) и аппаратуры различного назначения (контроля, диагностики, обработки информации и т. д.). Поэтому в учебные планы специальности «Конструирование и технология радиоэлектронных средств» (23.03) введен курс по изучению методов применения микропроцессоров и микроЭВМ в радиоэлектронных средствах. Целью этого курса является изучение студентами основ проектирования радиоэлектронных микропроцессорных устройств (МПУ) и их технической реализации; основ проектирования и методов технической реализации специализированных микропроцессорных систем (МПС); архитектуры МП; методики выбора микропроцессорных комплектов БИС или микроЭВМ; методов разработки и отладки программ для МПУ и специализированных МПС с командным и микрокомандным уровнями управления; специфических вопросов конструирования МПС различного назначения.

В указанном курсе содержатся основные сведения об архитектуре микропроцессоров и микроЭВМ, системах команд и способах реализа-

ции типовых функций в МП, программном обеспечении МПС, методиках проектирования и программирования МП и специализированных МПС, основах конструирования, производства радиоаппаратуры и проектирования электронного технологического оборудования с использованием микропроцессорной техники.

Для подготовки радиоинженеров-конструкторов-технологов по вопросам правильного применения микропроцессорной техники и микроЭВМ, а также их разработки предпринято издание данного учебника.

Содержание учебника полностью соответствует программе курса по микропроцессорной технике для студентов, обучающихся по специальности 23.03, и базируется на материалах различных источников, оригинальных результатах работ, а также на курсе лекций, прочитанных автором в Рязанском радиотехническом институте. Студентам, изучающим основы применения микропроцессорной техники в радиоэлектронике и вычислительной технике, можно порекомендовать систематическую проработку материалов, публикуемых в журналах «Микропроцессорные средства и системы», «Управляющие системы и машины», «Электронная промышленность» и др. Материал учебника может быть полезен студентам смежных специальностей: «Конструирование и технология электронных вычислительных средств» (22.05), «Радиотехника» (23.01).

Автор благодарен рецензентам — сотрудникам кафедры «Конструирование и производство ЭВА» МГТУ им. Н. Э. Баумана (зав. кафедрой д-р техн. наук, проф. Б. И. Белов) и д-ру техн. наук, проф. Д. В. Пузанкову — кафедре вычислительной техники ЛЭТИ им. В. И. Ульянова (Ленина) (зав. кафедрой д-р техн. наук, проф. В. Б. Смоллов) за ценные замечания, способствовавшие улучшению содержания книги.

Автор с благодарностью примет все замечания по книге, которые можно направлять по адресу: 101430, Москва, ГСП-4, Неглинная ул., д. 29/14, издательство «Высшая школа».

Автор

ВВЕДЕНИЕ

Применение микропроцессоров и микроЭВМ в РЭС различного назначения привело к расширению функциональных возможностей РЭС, улучшению качества обработки информации, повышению надежности и стабильности функционирования аппаратуры.

Языковые, программные и аппаратурные средства, используемые в микропроцессорных РЭС, постоянно совершенствуются. Производительность микропроцессоров непрерывно возрастает по мере совершенствования микроэлектронной технологии и архитектуры МП, разрядность однокристалльных МП достигла 32 бит. Для прямой обработки аналоговых сигналов созданы аналоговые МП, в структуре которых имеются несколько каналов аналого-цифровых и цифроаналоговых преобразователей, а также цифровой процессор. С целью ускорения процесса обработки информации разработаны сопроцессоры для ускоренного проведения операций умножения, деления, работы с плавающей точкой, обработки цифровых сигналов посредством преобразований Фурье и др. Все это свидетельствует о больших и еще не раскрытых перспективах и возможностях применения микропроцессоров и микроЭВМ в составе РЭС.

В случае применения в РЭС микроЭВМ разработчики должны уметь оценивать возможности архитектуры и технические характеристики микроЭВМ, а также владеть языками программирования различных уровней. Так, для создания системного программного обеспечения довольно распространенным является язык ассемблера, позволяющий наиболее полно использовать аппаратурные и программные ресурсы микроЭВМ. В то же время с точки зрения производительности труда программистов задачи обработки данных решаются с помощью языков высокого уровня.

Поэтому современным радиоинженерам необходимы знания как в области микропроцессорной техники, так и в программировании радиотехнических задач на языках различных уровней.

Глава 1

ВВЕДЕНИЕ В ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

§ 1.1. ТЕНДЕНЦИИ РАЗВИТИЯ РЭС И ПОДХОДЫ К ИХ ПРОЕКТИРОВАНИЮ

Радиоэлектронные средства как система. Радиоэлектронные средства (РЭС) являются важнейшей составной частью различных технических систем: радиотехнических, вычислительных, электросвязи, автоматики, гибких производственных и прочих технических систем разнообразного назначения.

Система — комплекс элементов, находящихся во взаимодействии. Под технической системой понимают совокупность взаимосвязанных разнородных устройств (подсистем), совместно выполняющих заданные функции в условиях взаимодействия с внешней средой. Каждая система состоит из взаимосвязанных частей или подсистем и в свою очередь входит в систему более высокого уровня. Радиоэлектронные средства входят в качестве подсистем в состав различных технических систем и предназначены для преобразования информации в технических системах и организации их энерго- и жизнеобеспечения. Свойства РЭС описываются множеством электрических, конструкторско-технологических и экономических параметров и характеристик. Для исследований, проектирования, производства и эксплуатации РЭС широко применяется системный метод, заключающийся в возможности выделения в системе наиболее существенных для ее функционирования подсистем и взаимосвязей и отбрасывания, особенно на начальных этапах исследований и проектирования тех подсистем и взаимосвязей, которые можно оценить как менее значимые. При системном подходе к исследованиям и проектированию радиоэлектронные средства как систему удобно подразделять на подсистемы в соответствии с выполняемыми ими функциями. Обычно к подсистемам РЭС применяют термин «устройство». Подсистемами РЭС являются: передающие и приемные устройства; устройства обработки и выдачи информации; вторичные источники питания; микропроцессорные устройства управления, устройства сопряжения РЭС с потребителями информации и т. д. Каждое из устройств имеет алгоритм функционирования и схему. Все устройства, входящие в состав РЭС определенной технической системы, характеризуются своими конструкциями, особенностями технологии и методами эксплуатации.

Системные методы в технике являются конкретизацией материалистической диалектики для решения сложных технических задач, в них находят отражение все категории и законы диалектики. Ярким

примером прикладной применимости законов материалистической диалектики является процесс развития радиоэлектронных средств. Так, согласно закону единства и борьбы противоположностей, вскрывающему источники развития природы, общества и мышления, источниками развития РЭС являются открытые новые физические явления в электронике и новые технологии их реализации. Согласно этому закону на всех уровнях развития систем существуют противоречия между многофункциональностью и специализацией, причем в начале определенного этапа развития преобладают многофункциональные системы, а в конце — специализированные. Что касается развития микропроцессорных РЭС, то многофункциональность и специализация сосуществуют на всех уровнях иерархии аппаратных и программных средств. Закон перехода количественных изменений в качественные вскрывает механизм развития РЭС — создание БИС и СБИС, увеличение степени интеграции, совершенствование микроэлектронных технологий и, как следствие, получение качественно новой элементной базы: микропроцессорных комплектов, заказных и полужаказных БИС и СБИС. Направление развития РЭС — от аппаратуры на электронных лампах до получения современных результатов в виде аппаратуры на интегральных микросхемах (ИМС) различной степени интеграции с использованием микропроцессорной техники — полностью соответствует закону отрицания отрицания. Системные методы имеют большое мировоззренческое и методологическое значение, их эффективное использование в науке и технике возможно только на базе материалистической диалектики.

Тенденции развития РЭС. Радиоэлектронные средства и основные области их применения (радиотехнические системы, электронные вычислительные машины и системы, системы автоматики и др.) претерпели значительное развитие, которое интенсивно продолжается. Сменилось четыре поколения РЭС: первое поколение — аппаратура на электронных лампах и дискретных электрорадиоэлементах (до середины 50-х годов), второе поколение — аппаратура на транзисторах и дискретных электрорадиоэлементах (до начала 60-х годов), третье поколение — аппаратура на интегральных микросхемах и дискретных микроэлектрорадиоэлементах (до середины 70-х годов), четвертое поколение — аппаратура на БИС, СБИС, микропроцессорах, устройствах функциональной электроники и микросборках. Переход от одного поколения аппаратуры к другому характеризуется резким улучшением тактико-технических характеристик и новым расширением функциональных возможностей. Так, в системах радиосвязи, радиолокации, радионавигации и радиоуправления разработанные новые принципы действия, реализованные на усовершенствованной элементной базе, позволили значительно увеличить дальность действия систем, точность, помехоустойчивость, а также число выполняемых функций. Соответственно с развитием РЭС совершенствуются конструкции, технология, производство и эксплуатация. Так, на начальных этапах развития РЭС были раздельная механическая обработка деталей (штамповка, прес-

сование из пластмасс и т. п.) и многоэтапная ручная сборка (механический и электрический монтаж). На современном этапе развития РЭС — это конструкции, использующие высокие уровни интеграции (микроэлектроника), и групповые технологические процессы, основанные на сложнейших физических процессах (осаждение пленок, воздействие на поверхностные слои твердого тела и т. п.).

Переход от одного поколения РЭС к другому качественно меняет и характер работы инженеров-разработчиков РЭС. В эпоху дискретных элементов основные усилия разработчиков были направлены на решение схмотехнических вопросов, поскольку тогда многое определялось тем, каким образом будут соединены между собой активные элементы (электронные лампы и транзисторы) и электрорадиоэлементы. При этом конструкторы и технологи практически не вносили существенных изменений в схмотехнические решения. В настоящее время в БИС используются стандартные схемные решения, реализованные в различных технологиях, при этом роль конструктора и технолога при создании новой аппаратуры резко возросла. Принципиальные изменения технологии проектирования РЭС имеются в микропроцессорных РЭС, поскольку в этой аппаратуре функции определяются не схемой устройства или системы, а программой, хранящейся в памяти и управляющей преобразованиями в микропроцессоре. При проектировании микропроцессорных РЭС разработчик должен перенести свои усилия от создания новых схем к разработке и созданию алгоритмов функционирования РЭС и их программной реализации. При этом должна решаться проблема рационального распределения функций между аппаратными и программными средствами проектируемых РЭС.

Детерминированный и статистический подходы к проектированию РЭС. Свойства и возможности технических систем (в том числе и РЭС) описываются множеством параметров и характеристик. Под параметром понимают величину, описывающую количественно то или иное свойство изделия или системы (например, масса, выходная мощность, емкость памяти, стоимость). Характеристикой называют описание свойства изделия или системы, выраженное в виде аналитической зависимости (например, амплитудно-частотная характеристика избирательного устройства).

Большинство параметров и характеристик РЭС можно исследовать и количественно выразить только на основе вероятностных методов, так как они связаны со случайными событиями, величинами и функциями. На основе методов теории вероятностей и математической статистики производится вероятностное описание сигналов и помех, внешних воздействий, значений и изменений параметров аппаратуры, времени поиска неисправностей и ремонта, времени безотказной работы и т. п. Даже такие исходные параметры элементов, как значения номиналов резисторов и конденсаторов, параметры транзисторов в дискретном или интегральном исполнении, вследствие большого числа факторов, влияющих на технологический процесс их изготовления, имеют случайные отклонения от паспортных данных. Только некоторые

параметры, такие, как масса, габариты, потребляемая мощность, диапазон частот и ряд других, можно описать детерминированно.

Определение детерминированных параметров и характеристик в процессе проектирования РЭС производится с помощью аналитических соотношений, позволяющих получать однозначные соответствия между значениями детерминированных параметров и характеристик системы и влияющими на них факторами. Эти соотношения обоснованы и выведены в различных фундаментальных дисциплинах, базирующихся на аппарате детерминированной математики: в теории электрических цепей, основах радиоэлектроники, теоретической и прикладной механике и др. Детерминированные методы имеют, как правило, небольшой объем вычислительных работ и математических преобразований.

Необходимость использования вероятностных методов при проектировании объясняется прежде всего случайным характером многих явлений и процессов и наличием большого числа степеней свободы у исследуемых процессов, что не позволяет описать зависимости и процессы количественно и однозначно. При увеличении числа степеней свободы количество переходит в качество и в соответствии с сущностью исследуемого явления или процесса необходимо переходить к использованию таких закономерностей, которые учитывают возможность появления вероятностных исходов. Вероятностные методы требуют более высокого уровня математической проработки в процессе проектирования аппаратуры и, как правило, широкого применения ЭВМ.

§ 1.2. РОЛЬ И МЕСТО МИКРОПРОЦЕССОРОВ В СОВРЕМЕННЫХ РЭС

Исторические предпосылки появления микропроцессорных РЭС. Развитие технических систем, например радиотехнических систем и систем связи, связано с постоянным количественным усложнением РЭС, причем это усложнение происходит с интенсивностью примерно в 2 раза за пять лет. История развития техники показывает, что происходит постоянное усложнение алгоритмов преобразования сигналов, фильтрации смеси сигнала и помехи, навигации, обнаружения целей и т. д. Постоянно ужесточаются требования к точности выполнения преобразований и выдачи информации, тактико-техническим характеристикам систем, а также к средствам автоматизации и управления системами. Особенно жесткими оказываются требования к точности и стабильности параметров и характеристик аналоговых РЭС. В частности, в современных радиотехнических системах при проектировании и изготовлении фильтров предъявляются требования обеспечения относительной точности и стабильности параметров элементов порядка $10^{-5} \dots 10^{-6}$, что практически трудно выполнимо. Для создания высококачественных РЭС, удовлетворяющих жестким требованиям к тактико-техническим характеристикам, потребовался переход от анало-

говых методов приема, обработки и регистрации информации к цифровым. При этом, к примеру, в радиотехнических системах обработка радиосигналов и их смеси с помехами (модуляция, фильтрация, усиление, органичение, детектирование и другие преобразования) осуществляется аналоговыми методами в обычных линейных и нелинейных элементах и цепях, а преобразования и отображения информации — цифровыми методами. В ряде технических систем, в частности в системах автоматики и телеметрии, обработка информации, включая и задачи фильтрации, производится только с помощью цифровых методов.

Для аналоговых РЭС характерны определенные преобразования токов и напряжений: интегрирование, дифференцирование, нелинейные преобразования и т. п. В цифровых РЭС преобразованию подвергаются не напряжения (токи), непосредственно отображающие информацию, а напряжения (токи), отображающие сигналы, в параметрах которых заложена информация, а также наложенные на сигналы помехи. В цифровых РЭС информация внутри аппаратуры циркулирует обычно в параллельном коде. Отсчеты, представляемые в кодовых словах, подвергаются обработке в специализированных вычислителях по алгоритму функционирования РЭС. Цифровые алгоритмы могут реализовать различные функции, в частности функции первичной обработки информации (тарировка, фильтрация, калибровка и др.), модуляции, детектирования и т. п.

Основной частью цифрового устройства, в котором производится обработка информации, является процессор, состоящий из устройства управления (УУ) и операционного устройства (ОУ). Операционное устройство состоит из сумматоров, регистров, счетчиков и т. п. В нем реализуются арифметические и логические операции в соответствии с командами (микрокомандами), которые поступают из устройства управления. Существуют два способа организации УУ: 1) на схемной (жесткой) логике; 2) на программируемой логике. Устройство управления на схемной логике строится на интегральных микросхемах (ИМС) различной степени интеграции, в том числе на специализированных БИС, посредством выбора определенного комплекта ИМС и способа их электрического соединения. Функционирование такой схемы УУ заключается в генерации последовательности управляющих сигналов, вырабатываемых УУ под воздействием генератора синхронизирующих импульсов. При такой организации ОУ и УУ узко специализированы, любое изменение алгоритма обработки информации или различных коэффициентов в расчетных соотношениях требует изменения состава ИМС и схемы их соединений.

Микропроцессорные РЭС — качественно новые радиоэлектронные средства с использованием сложной элементной базы. Развитие цифровых методов и микропроцессоров (МП) привело к формированию нового направления в создании РЭС, основанного на принципах программируемой логики или программной реализации РЭС.

В микропроцессорных РЭС обработка информации производится во встроенном микропроцессорном устройстве (МПУ). Схема МПУ

обычно заранее определена, а функционирование аппаратуры определяется программой, записанной в памяти МПУ. При проектировании микропроцессорных ЭЭС инженер разрабатывает алгоритм функционирования ЭЭС и программу работы МПУ, т. е. функции разработчика перемещаются из схемной области в программную. Переход к МП связан с цифровыми методами, при этом техника сделала шаг, который давно сделала природа, создав живые организмы, способные с высокой точностью накапливать и перерабатывать большие количества информации, сформировав мозг и наследственность.

Появление и развитие МП не есть случайность или результат удачного изобретения. Они являются логическим следствием диалектического развития технических систем, микроэлектроники, а также вычислительной техники и прикладной математики. Внедрение МП в различные технические системы привело к революции не в технологии микроэлектроники, так как по конструкции и технологии МП представляют собой БИС и СБИС, а в мышлении инженера, поскольку для проектирования систем с микропроцессорами требуется качественно новый подход.

Применение МП в ЭЭС позволило широко использовать принципы программируемой логики, заключающиеся в том, что все преобразования информации, производимые в ОУ, осуществляются по программе, записанной в памяти и реализующей алгоритмы функционирования ЭЭС. Применительно к ЭЭС — это преобразования над сигналами, значения которых в аналого-цифровых преобразователях (АЦП) превращаются в последовательность кодов (чисел). В принципе программное построение ЭЭС было возможно и прежде, но только благодаря появлению МП и транзисторных БИС программируемой памяти удастся создавать ЭЭС с приемлемыми габаритами, массой и энергопотреблением, обеспечивающим реализацию сложных алгоритмов с высокими точностью и стабильностью.

Применение МП в ЭЭС нельзя изолировать от общих задач их создания, изготовления и эксплуатации. Микропроцессоры являются современной элементной базой радиоэлектронных средств, они органически входят в состав аппаратуры. Поэтому при проектировании микропроцессорных ЭЭС должны быть тщательно распределены функции между аналоговой и цифровой частями на основе полного понимания задачи создания технической системы, возможностей применения аналоговых и цифровых методов и устройств. При этом следует учитывать принципиальную разницу между функционированием универсальных ЭВМ (в том числе и микроЭВМ) и МПУ, встроенных в ЭЭС. Универсальные ЭВМ предназначены для решения множества самых различных задач, причем ряд задач может встречаться только по одному разу в течение всего жизненного цикла функционирования ЭВМ. Поэтому главное для ЭВМ — обеспечение быстрой и удобной смены программ. Встроенные МПУ ориентированы на решение единственной задачи в реальном времени в течение всего периода их функционирования. Поэтому для МПУ важны быстроедействие, экономия емкости

памяти, надежность и точность решения задачи. Кроме того, для МПУ очень важно согласование аналоговых и цифровых частей аппаратуры. Конечно, во многих случаях применительно к сложным задачам и в РЭС, и в технологическом оборудовании могут использоваться микроЭВМ.

Основные понятия и определения. Рассмотрим ряд определений и понятий, относящихся к микропроцессорной технике.

Микропроцессор (МП) — программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управления им, построенное, как правило, на одной или нескольких БИС.

Микропроцессорное устройство (МПУ) — функционально и конструктивно законченное изделие, представляющее собой схемно-конструктивное соединение нескольких микросхем, включая один или несколько микропроцессоров, предназначенное для выполнения одной-двух из следующих функций: получение, обработка, передача, преобразование информации и управление. МПУ имеет унифицированные присоединительные характеристики (интерфейс, конструкцию, программное обеспечение и т.д.) и функционирует, как правило, в составе какой-либо технической системы.

Аналогичное изделие, реализующее весь набор указанных функций, называется *микропроцессорной системой управления* (МПС). МПС выдает управляющие воздействия на объект управления в соответствии с реализуемым законом управления.

МикроЭВМ — ЭВМ, состоящая из микропроцессора, полупроводниковой памяти, средств связи с периферийными устройствами и при необходимости пульта управления и источника питания, объединенных общей конструкцией.

Встраиваемая микроЭВМ — микроЭВМ, конструктивно приспособленная для работы в составе приборов и оборудования. Встраиваемые микроЭВМ не имеют индивидуального пульта управления, источника электропитания и декоративного оформления.

Микропроцессорный комплект БИС (МПК БИС) — совокупность конструкторско-технологических и электрически совместимых интегральных микросхем, предназначенных для построения МПУ, МПС или микроЭВМ с определенным составом и требуемыми техническими характеристиками.

Алгоритм — набор предписаний, однозначно определяющих содержание и последовательность выполнения операций для систематического решения определенной задачи.

Программа — набор инструкций, реализующих алгоритм. Программы могут быть написаны: а) в двоичном или шестнадцатеричном (машинном) коде, который непосредственно выполняется процессором; б) на языке типа ассемблер; в) на языке высокого уровня.

Микропрограммирование — представление машинной команды в виде последовательности микрокоманд, т. е. реализуемых аппаратно элементарных машинных операций.

Архитектура микропроцессора — его логическая организация, определяемая возможностями МП по аппаратной или программной реализации функций, возлагаемых на проектируемые МПУ или микроЭВМ. Архитектура отражает структуру МП, способы представления и форматы данных, набор операций, форматы управляющих слов, способы обращения ко всем доступным для пользователя элементам структуры, реакцию МП на внешние сигналы.

Интерфейс — система шин, магистралей, вспомогательной аппаратуры и алгоритмов, реализованных на этом оборудовании, предназначенная для организации обмена между МП, памятью, устройствами ввода — вывода и другими устройствами, входящими в состав МПС или микроЭВМ.

Магистраль — группа линий передачи информации, объединенных общим функциональным признаком (например, магистраль данных, адресов, управления).

§ 1.3. ФОРМАЛИЗАЦИЯ ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ РЭС

Уровни проектирования. Технические системы, в состав которых входят микропроцессорные РЭС, как правило, состоят из большого количества взаимодействующих элементов, т. е. принадлежат к классу сложных систем. Проектирование сложных систем занимает несколько лет и требует привлечения значительного числа специалистов. Такие сроки часто неприемлемы, так как оказываются соизмеримыми с временем морального износа проектируемой аппаратуры. Диалектические противоречия — усложнение систем и сокращение сроков проектирования — присущи процессу проектирования сложных систем, в том числе и микропроцессорных РЭС.

При проектировании технических систем используется блочно-иерархический подход, при котором представления о проектируемой системе расчленяются на *иерархические уровни*. На высшем уровне используется наименее детализированное представление, отражающее только самые общие черты и особенности проектируемой системы. На следующих уровнях степень подробности рассмотрения возрастает, при этом система рассматривается не в целом, а отдельными блоками. Такой подход позволяет на каждом уровне формулировать и решать задачи приемлемой сложности, поддающиеся уяснению и пониманию человеком и решению с помощью имеющихся средств проектирования. Разбиение на блоки должно быть таким, чтобы документация на блок любого уровня была обозрима и воспринимается одним человеком. Преимущества блочно-иерархического подхода состоят в том, что сложная задача большой размерности разбивается на последовательно решаемые группы задач малой размерности, причем внутри групп разные задачи могут решаться параллельно.

Принципы блочно-иерархического проектирования отражают существующее по ЕСКД деление схем на принципиальные, функциональные, структурные.

Таким образом, иерархические уровни представляют собой уровни описаний объектов, различающиеся степенью подробности отображения свойств объекта. Иерархические уровни называют также горизонтальными уровнями или уровнями абстрагирования. Совокупность описаний некоторого уровня вместе с постановками задач и методами получения этих описаний называют иерархическим уровнем проектирования.

На горизонтальных уровнях имеются группы задач, связанные с проектированием схем, конструкций, технологий. Эти группы задач совместно с используемыми для их решения моделями, методами, формами документации называются *аспектами* проектирования (иногда аспекты проектирования называют вертикальными уровнями проектирования). В большинстве областей техники различают такие аспекты, как функциональное, конструкторское и технологическое проектирование. При проектировании микропроцессорных РЭС важным аспектом является алгоритмическое (программное) проектирование, результатом которого является математическое и программное обеспечение разрабатываемой аппаратуры. Практика проектирования микропроцессорных РЭС привела к выделению горизонтальных и вертикальных уровней, перечисленных в табл. 1.1.

Функциональное проектирование микропроцессорных РЭС можно подразделить на следующие уровни.

На системном уровне проектируется структурная схема РЭС, в связи с чем его называют также структурным уровнем. На этом уровне ведется укрупненное рассмотрение всех РЭС, а элементами являются такие устройства, как усилители, модуляторы, аналого-цифровые и цифроаналоговые преобразователи, МПУ, антенно-фидерные устройства, запоминающие устройства и др.

На функционально-логическом уровне проектируются функциональные и принципиальные схемы всех устройств, входящих в состав РЭС.

На схемотехническом уровне проектируются принципиальные электрические схемы ИМС или фрагментов БИС. Элементами здесь являются компоненты электронных схем (резисторы, конденсаторы, транзисторы, диоды).

На компонентном уровне разрабатываются отдельные компоненты интегральных микросхем, рассматриваемые как системы, состоящие из элементов — участков полупроводникового кристалла, или дискретные компоненты.

Высшие иерархические уровни алгоритмического проектирования служат для разработки закона функционирования РЭС, определения задач, решаемых микропроцессорной частью РЭС, планирования программной системы и разработки схем алгоритмов. Затем разрабаты-

Уровни и аспекты проектирования микропроцессорных РЭС

Уровни	Аспекты			
	функциональный	алгоритмический	конструкторский	технологический
1	—	Разработка закона функционирования РЭС. Проектирование схем алгоритмов. Программирование модулей	—	—
2	Системный (структурный)	Архитектурный (машинный)	Стойка, панель	Разработка принципиальной схемы технологического процесса
3	Функционально-логический	Микропрограммный	ТЭЗ, модуль	Разработка маршрутов технологического процесса
4	Схемотехнический. Компонентный	—	Кристалл интегральной микросхемы	Проектирование технологических операций

ваются программные модули, которые программируются на каком-либо алгоритмическом языке.

Основная задача архитектурного уровня проектирования — выбор архитектуры микропроцессорной части РЭС, т. е. выбор МПК БИС и определение таких структурно-алгоритмических особенностей, как форматы данных и команд, система команд, принципы выполнения операций и т. п. Часто выбор архитектуры рассматривают как одну из задач системного уровня, т. е. объединяют архитектурный и системный уровни в один иерархический уровень функционального проектирования.

Микропрограммный уровень предназначен для проектирования микропрограмм операций и процедур, выполняемых в микропроцессорных РЭС аппаратным способом. Этот уровень тесно связан с функционально-логическим уровнем проектирования.

Конструкторское проектирование включает в себя иерархические уровни проектирования стоек, панелей, типовых элементов замены (ТЭЗов), модулей и кристаллов (чипов) интегральных микросхем.

Технологическое проектирование заключается в определении принципиальной схемы технологического процесса, т. е. состава и последо-

вательности этапов изготовления РЭС; разработке маршрутов технологических процессов, т. е. определении состава и последовательности операций, выборе баз и группы технологического оборудования; проектировании технологических операций.

Схема процесса проектирования. В настоящей книге рассматриваются системный и функционально-логический уровни функционального аспекта проектирования, а также все уровни алгоритмического проектирования микропроцессорных РЭС. Рассмотрим основные задачи системного и функционально-логического уровней проектирования микропроцессорных РЭС совместно с соответствующими уровнями алгоритмического проектирования.

Исходной информацией для системного и архитектурного уровней проектирования микропроцессорных РЭС являются: закон функционирования РЭС, заданный в виде аналитических соотношений преобразования входной информации; набор схем алгоритмов выполнения основных функций РЭС, реализуемых программным путем; набор основных программных модулей; критерии оптимальности и набор ограничений на различные параметры разрабатываемой аппаратуры. Основными задачами системного и архитектурного уровней являются: определение принципов организации микропроцессорных РЭС;

разработка структурной схемы, т. е. определение состава устройств и способа их взаимодействия в процессе функционирования аппаратуры;

выбор микропроцессорного комплекта БИС;

выбор архитектуры микропроцессорной части РЭС, уточнение функций, реализуемых МПУ, и их разделение на функции, реализуемые аппаратным и программным путями;

определение требований к параметрам устройств и формирование технических заданий (ТЗ) на разработку отдельных устройств микропроцессорного РЭС.

В ТЗ на разработку отдельных устройств микропроцессорных РЭС входят: перечисление всех функций, выполняемых каждым устройством; условия работоспособности устройства; требования к его входным и выходным параметрам; данные о содержании и форме информации, которой данное устройство обменивается с другими устройствами аппаратуры; элементная база для построения устройства.

Основные задачи функционально-логического и микропрограммного уровней следующие:

детализация функций, выполняемых каждым устройством;

разработка принципиальных схем всех аналоговых устройств (усилителей, модуляторов и т. д.);

выбор принципа действия и разработка принципиальных схем аналого-цифровых (АЦП) и цифроаналоговых (ЦАП) преобразователей;

алгоритмическая реализация функций, выполняемых программным путем, и представление алгоритмов на одном из принятых алгоритмических языков;

выбор принципов организации МПУ и разработка его принципиальной схемы;

разработка микропрограмм, т. е. определение для каждой команды совокупности микрокоманд и последовательности их выполнения;

синтез функциональных и принципиальных схем цифровых устройств, входящих в состав РЭС;

синтез контролирующих и диагностических тестов для микропроцессорной части РЭС;

формулировка ТЗ для схемотехнического уровня проектирования.

На каждом из рассматриваемых уровней разработчику приходится решать задачи синтеза и анализа. Целью задачи синтеза является получение конкретных вариантов проектируемой аппаратуры, а целью задачи анализа — изучение свойств вариантов аппаратуры и их оценка.

Различают структурный и параметрический синтез. Цель *структурного синтеза* — получение структуры устройства, т. е. состава элементов и способа связи их между собой. Задачу выбора оптимальной структуры называют структурной оптимизацией. Цель *параметрического синтеза* — определение числовых значений параметров элементов и устройств, причем задачу получения оптимальных значений параметров называют параметрической оптимизацией.

При решении задач анализа используют модели проектируемой системы. Различают физические и математические модели. Физическими моделями являются различного рода макеты, стенды. Математическая модель — это совокупность математических объектов (чисел, переменных, векторов, множеств и т. п.) и отношений между ними, которая адекватно отображает свойства проектируемого объекта. Математические модели могут быть функциональными, если они отражают физические или информационные процессы, протекающие в моделируемом объекте, и структурными, если они отображают только структурные (в том числе и геометрические) свойства объектов. Функциональные модели, как правило, задаются в виде аналитических соотношений, а структурные — в виде графов и матриц. При проектировании микропроцессорных РЭС, разработке и отладке программ функционирования МПУ широко используют функциональные модели работы МПУ, реализуемые на универсальных ЭВМ и микроЭВМ.

На рис. 1.1 представлена типичная для рассматриваемых уровней схема процесса проектирования. Разработка аппаратуры по предъявляемому ТЗ начинается с анализа возможностей реализации требований ТЗ. В случае положительного результата анализа разрабатывают модели для структурного и параметрического синтеза аппаратуры. В соответствии с разработанными моделями генерируется исходный вариант аппаратуры, параметры которого оцениваются с позиций удовлетворения условий работоспособности. Если для синтезируемого варианта структуры обеспечивается выполнение условий работоспособности с заранее оговоренным запасом (с учетом допустимых отклонений реально достигнутых значений параметров аппаратуры), то задача син-

теза считается решенной. При этом результаты проектирования оформляют в виде необходимой технической документации. В случае невыполнения условий работоспособности проводят анализ возможности улучшения характеристик синтезированного варианта объекта путем изменения структуры и управляемых параметров в рамках применя-

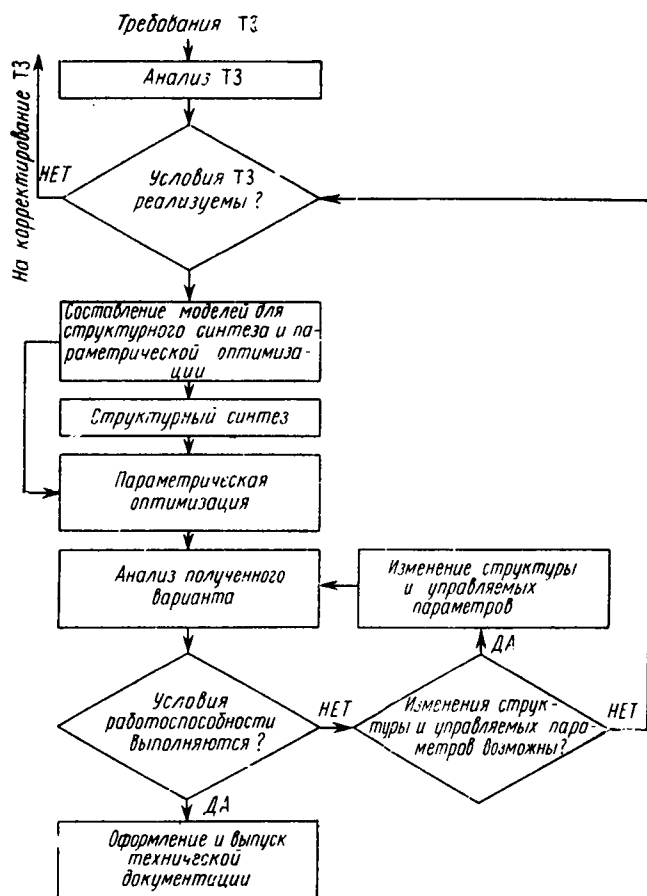


Рис. 1.1. Схема процесса проектирования

мых моделей для структурного синтеза и параметрической оптимизации. В случае невозможности выполнения этого производят корректирование ТЗ на проектируемый объект. Таким образом, процесс проектирования носит итерационный характер, итерации могут включать в себя и более одного уровня проектирования, необходимость возврата к предыдущему уровню может обнаружиться на любом последующем уровне проектирования.

Проектирование сложных технических систем производится на основе главных критериев: а) качества проектирования; б) стоимости проектирования; в) сроков разработки; г) количества занятых специалистов-разработчиков. Возможность формализации в процессе проектирования микропроцессорных РЭС и итерационный характер проектирования с учетом перечисленных критериев делает предпочтительным выбор автоматизированных методов проектирования РЭС. Разработка же микропроцессорной части аппаратуры без использования автоматизированных методов проектирования и отладки программ функционирования РЭС в настоящее время практически невозможна.

Методика решения задач проектирования. Реализация блочно-иерархического подхода к проектированию сложных систем может быть осуществлена либо в восходящей, либо в нисходящей последовательности решения задач проектирования разных уровней.

Нисходящее проектирование (проектирование «сверху вниз») характеризуется тем, что решение задач проектирования верхних уровней предшествует решению задач проектирования нижних уровней. При этом исходные данные для проектирования представляются в виде ТЗ для высшего иерархического уровня. Восходящее проектирование предусматривает обратную последовательность решения задач. Объекты, проектируемые на каждом уровне восходящего проектирования, являются базовыми для реализации верхних уровней.

При разработке микропроцессорных РЭС находит применение как нисходящее, так и восходящее проектирование. Если для создания аппаратуры известна элементная база и выбран микропроцессорный комплект БИС, то используется нисходящее проектирование. Если для построения РЭС требуется разработка специализированных БИС и нового МПК, то применяется восходящее проектирование.

Представление о проектируемых объектах на разных уровнях разработки микропроцессорных РЭС задается математическими моделями, описывающими возможные структуры объекта и протекающие в них процессы.

При функциональном проектировании аппаратуры используются следующие математические модели и методы.

На системном уровне модель проектирования должна отражать необходимую совокупность функциональных элементов системы, взаимосвязь между этими элементами и процессы преобразования информации. Поэтому для решения задач структурного синтеза используется аппарат теории систем, теории графов и сетей, теории преобразования сигналов, теории информации, дискретного математического программирования и др. Задача анализа на этом уровне заключается в определении параметров и характеристик, описывающих процесс обработки информации в системе. При этом нужно оценить производительность и относительную загрузку работы всех функциональных элементов системы, точность и достоверность преобразования информации, основные параметры и характеристики спроектированных РЭС. Математическим аппаратом анализа на системном уровне являются теория

массового обслуживания, теория радиотехнических и вычислительных систем.

На функционально-логическом уровне проектирования микропроцессорных РЭС разрабатываются функциональные и принципиальные схемы всех цифровых и аналоговых устройств аппаратуры. На этом уровне в математических моделях отображаются переменные и действия, которые выполняются моделируемым устройством в соответствии с алгоритмом функционирования. При этом информационные переменные отождествляются с сигналами и каждому сигналу ставится в соответствие некоторая физическая величина — напряжение или ток на выходе каждого элемента. Математический аппарат, используемый для решения задач анализа и синтеза на этом уровне: теория радиотехнических цепей и сигналов, теория проектирования радиотехнических устройств различного назначения, математическая логика и теория цифровых автоматов, нелинейное программирование, положения теории вероятностей и математической статистики.

При решении задач алгоритмического проектирования применяются алгоритмические языки различных уровней, теория алгоритмов, теория вычислительных систем и программирование.

Цифровые алгоритмы в РЭС. Цифровые алгоритмы и методы, реализованные на микропроцессорной элементной базе, являются основным направлением в обеспечении стабильной и точной реализации непрерывно усложняющихся принципов действия и сигналов в радиотехнических системах.

Первоначально в радиотехнических системах цифровые методы применялись для вторичной обработки информации, например для преобразования гиперболических координат в радионавигационных системах или вычисления режима движения ракеты в упрежденную точку встречи с целью в системах радиоуправления. В таких системах цифровые алгоритмы реализовывались в специализированных ЭВМ, которые как бы дополняли РЭС. С появлением интегральных микросхем появилась возможность встраивать цифровую аппаратуру в состав РЭС. При этом радиочастотная часть РЭС реализовывалась на аналоговых принципах, а цифровые методы использовались для обработки и отображения информации.

С внедрением микропроцессоров в РЭС цифровые методы и алгоритмы получают применение для фильтрации, модуляции и демодуляции; контроля, диагностики и прогнозирования; управления функционированием радиотехнических систем; выдачи информации оператору; в технике связи и телевидения; контрольно-измерительной и бытовой радиоаппаратуре; системах управления технологическими процессами и т. д. Следует иметь в виду, что в микропроцессорных РЭС до настоящего времени на аналоговых принципах основывается построение устройств, связанных с приемом и передачей радиосигналов (радиочастотные входы приемников и др.), и устройств, в которых в соответствии с принципом действия развиваются большие мощности

(каскады радиопередатчиков, устройства вторичного электропитания, усилительные устройства, устройства поворота антенн и т. п.).

Взаимосвязь цифровых и аналоговых устройств. Основным недостатком аналоговых радиотехнических систем является высокая чувствительность к отклонениям и нестабильности параметров элементов и устройств аналоговых РЭС, изменениям внешней среды, нестабильности источников питания, что существенно влияет на точность и стабильность обработки информации. Для повышения точности и стабильности функционирования аналоговых РЭС имеются три основных направления:

1) использование элементов и устройств, обладающих повышенной стабильностью и высокой точностью значений параметров. При этом увеличиваются стоимость, габариты, масса, а относительная погрешность получается не ниже 0,001;

2) использование большого числа регулировочных и настроечных элементов. При этом усложняются РЭС, усложняется и удорожается эксплуатация, требуется высококвалифицированный обслуживающий персонал. Однако получение погрешности ниже 0,001 очень проблематично;

3) использование цифровой аппаратуры, имеющей практически идеальную стабильность. Точность преобразования информации зависит в основном от разрядности и выбранных численных методов реализации алгоритмов преобразования информации. Однако использование цифровой аппаратуры на частотах выше 1 МГц в настоящее время вызывает серьезные трудности.

При разработке РЭС, содержащих цифровые и аналоговые устройства, возникают следующие специфические проблемы:

1) переход от математических моделей преобразования информации, заданных в терминах непрерывной математики, к моделям, заданным в терминах дискретной математики. Конечным результатом этого перехода являются алгоритмы цифровой обработки сигналов и их смеси с помехами;

2) синтез специализированных вычислительных устройств, входящих в состав РЭС и реализующих алгоритмы цифровой обработки сигналов в реальном времени. Вычислительные устройства могут быть построены либо на серийных БИС и ИМС на основе принципов жесткой логики или же на микропроцессорах по принципу программируемой логики;

3) определение рационального соотношения между аналоговыми и цифровыми методами в РЭС с учетом ограниченного быстродействия цифровых устройств;

4) учет ограниченной разрядности цифровой аппаратуры, которая однозначно ограничивает точность обработки информации в цифровых РЭС;

5) учет специфических для цифровой аппаратуры помех, вызванных ограниченной разрядностью аппаратуры и связанных с шумами квантования, округлением и усечением.

Разработчики микропроцессорных РЭС должны умело использовать все достоинства аналоговых и цифровых методов построения аппаратуры для получения оптимальных проектных решений. Это требует от коллектива инженеров — разработчиков микропроцессорных РЭС глубоких знаний в области проектирования радиотехнических систем, разработки аналоговых радиотехнических устройств, а также в области микропроцессорной техники и программирования МПУ.

§ 1.4. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ И КЛАССИФИКАЦИЯ МИКРОПРОЦЕССОРОВ

Микропроцессор является сложным программно-управляемым цифровым процессором в микроэлектронном исполнении, выполненным в виде интегральной микросхемы большой степени интеграции. Поэтому он описывается множеством параметров, присущих как электронным приборам (быстродействие, потребляемая мощность, габариты и масса, количество уровней питания, надежность, стоимость, тип корпуса, температурный диапазон и др.), так и вычислительным средствам (разрядность, цикл выполнения команд или микрокоманд, число внутренних регистров, наличие микропрограммного управления, тип стековой памяти, состав резидентного и программного обеспечения и т. д.). Микропроцессоры классифицируют по наиболее существенным из перечисленных характеристик с целью выбора эффективной сферы применения.

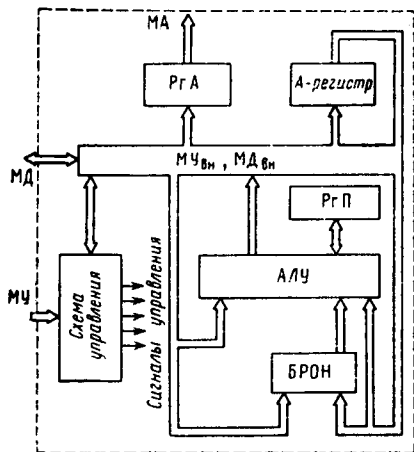


Рис. 1.2. Структурная схема однокристалльного микропроцессора

По числу БИС в микропроцессорном комплекте различают однокристалльные, многокристалльные и многокристалльные секционные МП.

Однокристалльные микропроцессоры получают при реализации всех аппаратурных средств процессора в виде одной БИС или СБИС. Однокристалльный МП имеет фиксированную разрядность, набор команд и конструктивно выполняется в виде одной интегральной микросхемы. Все выполняемые им операции определяются набором команд МП. В состав однокристалльного МП входят (рис. 1.2): арифметико-логическое устройство АЛУ, выходной регистр адреса РгА, регистр-аккумулятор (А-регистр), блок регистров общего назначения БРОН, регистр признаков РгП, схема управления, магистрали данных МД, адресов МА и управления МУ. Особенность однокристалльного МП —

наличие внутренней магистрали для передачи внутренних информационных данных $МД_{вн}$ и управляющих сигналов $МУ_{вн}$. Возможности однокристальных микропроцессоров ограничены аппаратными ресурсами кристалла и корпуса, но по мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса параметры однокристальных МП непрерывно улучшаются.

Рассмотрим логическую структуру развитого однокристального микропроцессора (рис. 1.3). Как отмечалось, микропроцессор имеет магистральную внутреннюю структуру. Общая информационная магистраль состоит из информационных магистралей вводимых данных $МД_1$, выводимых данных $МД_2$, внутренних информационных данных $МД_{вн}$ и управляющих сигналов $МУ_{вн}$. Внешние связи микропроцессора образуют информационные магистрали однонаправленных адресов $МА$, двунаправленных данных $МД$, сигналов системного управления $МУС$, сигналов запроса на прерывание процессора $МУПр$.

Основные блоки микропроцессора и их назначение: АЛУ — для выполнения арифметических и логических операций; блок управления вводом — выводом информации БУВВ с внутренними буферными регистрами — для временного хранения вводимых и выводимых кодов; блок прерываний микропроцессора БПр — для осуществления прерываний, поступающих от управляемых объектов (РЭС, технологического оборудования и др.); блок местного управления и синхронизации БМУС.

Арифметико-логическое устройство связано с регистром признаков $РгП$, число триггеров в котором может меняться в широких пределах. Регистр признаков обычно содержит разряды для хранения признаков результатов операции и служебных признаков. Обычно используют признаки нулевого результата, отрицательного результата, переноса, переполнения, четности (нечетности) результата, разрешения прерывания. Выделение этих признаков в одно слово позволяет легко организовать засылку его в память и восстановление состояния МП. Для системного управления кроме сигналов БМУС используют сигналы с выходов регистра кода состояния микропроцессора $РгС$.

В микропроцессоре может быть один или несколько регистров результата $РгР$ (накапливающих регистров или аккумуляторов). Включение аккумуляторов в состав МП позволяет не указывать в

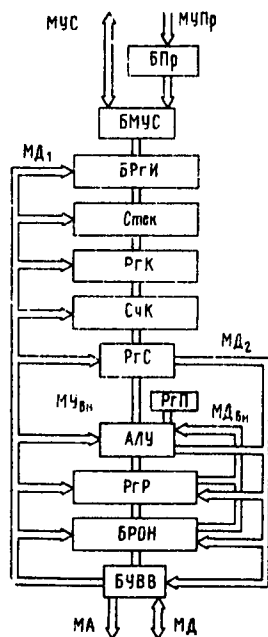


Рис. 1.3. Логическая структура развитого однокристального микропроцессора

команде местонахождение одного из операндов и (или) результата операции, т. е. уменьшить разрядность команд.

Для организации программного управления в состав МП входят счетчик команд СчК — для организации последовательного программного управления, регистр команд РгК — для хранения кода команды и блок индексных регистров БРгИ — для индексирования команд. Индексные регистры упрощают работу с массивами данных и позволяют при ограниченной разрядности адреса использовать память большого объема.

Формирование адреса следующей по порядку команды в СчК производится схемой инкремент — декремент (схема увеличения или

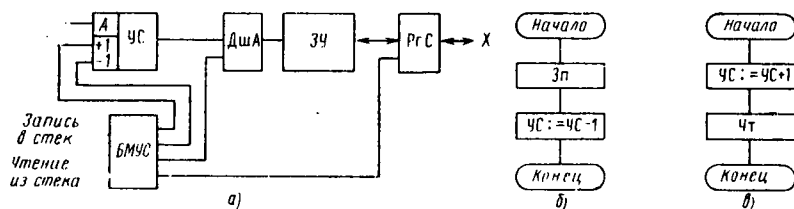


Рис. 1.4. Организация стека:

а — структурная схема; б — программа записи в стек; в — программа чтения из стека

уменьшения содержимого счетчика на единицу). Включение данной схемы в состав МП позволяет сократить время выполнения этой часто встречающейся процедуры и совместить во времени выполнение операции в АЛУ и формирование адреса следующей команды. Схема инкремент — декремент используется также для изменения содержимого индексных регистров.

Блок прерываний БПр предназначен для приема запросов на прерывание от внешнего источника, сохранения текущего состояния МП и перехода к прерывающей программе.

Для работы с подпрограммами организуется стек (от англ. stack), при обращении к которому не требуется указания адреса. Стек (магазинная память) — эффективное программно-аппаратное средство для сохранения и возврата команд данных и результатов при переходе с одной подпрограммы на другую и осуществления прерываний.

Возможная организация стека представлена на рис. 1.4, а. Выборка из ячеек запоминающего устройства ЗУ осуществляется через дешифратор адреса ДшА по адресу, находящемуся на реверсивном счетчике адреса, называемом указателем стека УС. Начальное значение адреса поступает в УС на вход А. В процессе работы состояние УС при каждой записи уменьшается, а при каждом чтении увеличивается на единицу. Поэтому для работы со стеком в системе команд МП предусматриваются следующие команды: запись в стек; чтение из стека; увеличение содержимого УС на единицу; пересылка содержи-

мого УС в регистр (память); пересылка содержимого регистра (памяти) в УС.

При записи информации входное слово X поступает на регистр слов RgC и записывается в ЗУ по адресу, установленному в УС. После записи информации содержимое УС уменьшается на единицу, подготавливаясь таким образом к следующей записи (рис. 1.4, б), так что УС постоянно указывает на свободную ячейку, в которую должно быть занесено очередное входное слово.

При чтении (рис. 1.4, в) первоначально содержимое УС увеличивается на единицу, а затем производится чтение информации из памяти. В результате в RgC появится слово, которое было записано последним.

После последовательной записи в стек слов X_1, X_2, X_3 при первом чтении будет выбрано слово X_3 , затем X_2 и, наконец, X_1 , т. е. в стеке осуществляется дисциплина обслуживания «последним пришел — первым ушел» (last in first out — LIFO). Возможны и другие способы организации магазинной памяти, но в МП обычно реализуется рассмотренный способ. В качестве ЗУ для стека может применяться оперативное ЗУ (ОЗУ) либо универсальные регистры из БРОН. Если для стека используются ОЗУ, то в МП достаточно иметь только УС. В качестве последнего, а также счетчика команд $CчК$ и индексных регистров могут быть задействованы также универсальные регистры из БРОН, что позволяет гибко использовать регистровые ресурсы МП программным путем и упростить структуру МП.

В микропроцессоре может быть один или несколько регистров результата RgP (накапливающих регистров или аккумуляторов). Блок универсальных регистров общего назначения БРОН (или регистровое сверхоперативное запоминающее устройство — СОЗУ) расширяет внутренние возможности микропроцессора по хранению данных и управляющей информации. В процессе функционирования АЛУ в БРОН хранятся промежуточные результаты вычислений и данные, требующиеся для последующих вычислений, с целью минимизации числа обращений к оперативной памяти. Эффективная загрузка БРОН в процессе функционирования микропроцессорного РЭС позволяет повысить производительность реализации алгоритмов без увеличения тактовой частоты работы микропроцессора.

Многокристалльные микропроцессоры получают путем разбиения логической структуры процессора на функционально законченные части и их реализации в виде БИС и СБИС. Функциональная законченность БИС многокристалльного микропроцессора означает, что его части выполняют заранее определенные функции и могут работать автономно, а для построения развитого процессора не требуется организация большого количества новых связей и каких-либо других БИС.

Многокристалльные секционные микропроцессоры состоят из набора микропроцессорных секций. Микропроцессорная секция — микропроцессорная интегральная микросхема, реализующая часть микропроцессора и обладающая средствами простого функционального

объединения с однотипными или другими микропроцессорными секциями для построения законченных микропроцессоров, МПУ или микроЭВМ. Управление секционными микропроцессорами осуществляется микропрограммным способом. К секционным МПК БИС относятся серии К583, К584, КР587, К589, К1800, КР1802, КМ1804 и др.

Основное назначение секционных МПК — создание высокопроизводительных многоразрядных МП и МПС, на базе которых реализуются управляющие вычислительные системы различного назначения. Требуемая разрядность МП достигается за счет выбора соответствующего числа секций.

С целью расширения возможностей как секционного, так и однокристалльного микропроцессора к ним подключают дополнительные схемы, выполняющие различные функции по сопряжению с внешними устройствами, ускорению выполнения ряда арифметических операций, организации системы памяти, а также по решению специфичных задач. Комплект вспомогательных схем вместе с микропроцессорными комплектами составляет микропроцессорный комплект БИС (МПК БИС).

МПК БИС — совокупность конструктивно и электрически совместимых ИМС, предназначенных для построения микропроцессора, МПУ или микроЭВМ с определенным составом и требуемыми техническими характеристиками. Основу МПК БИС составляет базовый комплект ИМС одной серии, который может состоять из микросхемы однокристалльного МП с фиксированной разрядностью и набором команд или комплекта БИС многокристалльного МП. Для расширения функциональных возможностей микропроцессора базовый МПК БИС дополняется другими типами БИС: ОЗУ, постоянного ЗУ (ПЗУ), программируемого ПЗУ (ППЗУ), интерфейсных ИМС, контроллеров внешних устройств и т. д.

По назначению МП делятся на универсальные и специализированные. Универсальные МП могут быть применены для решения широкого круга разнообразных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. Специализированные МП оптимизированы по некоторым параметрам на конкретные применения (БИС микрокалькулятора, однокристалльный контроллер для определения соответствия в реальном времени принимаемого сигнала изменяющейся формы путем сравнения с различными эталонными сигналами и др.).

По виду обрабатываемых входных сигналов различают цифровые и аналоговые микропроцессоры. При этом следует помнить, что в обоих видах микропроцессоров обработка информации — цифровая. В аналоговых МП специально для обработки аналоговых сигналов имеются встроенные аналого-цифровые и цифроаналоговые преобразователи. В системах с аналоговыми МП входные аналоговые сигналы передаются в МП через АЦП, обрабатываются в цифровой форме и после преобразования в аналоговую форму в ЦАП поступают на выход. С архитектурной точки зрения такие МП представляют собой аналоговые

функциональные преобразователи сигналов. Они могут выполнять функции любой аналоговой схемы (например, производят генерацию колебаний, модуляцию, фильтрацию, кодирование и декодирование сигналов в реальном времени и т. д.). Особенность аналоговых МП — способность переработки больших объемов числовой информации посредством выполнения арифметических операций.

Сравнение цифровых МП производится путем сопоставления времени выполнения ими основных операций, сравнение же аналоговых МП — по количеству эквивалентных звеньев аналоговых устройств. Производительность аналоговых МП оценивается скоростью выполнения операции умножения.

По разрядности различают микропроцессоры с фиксированной и с изменяемой разрядностью слова (модульные МП). При фиксированной разрядности наиболее распространены МП с длиной слова 8 и 16 бит. При модульном принципе возможно построение 8-, 16-, 24-, 32-разрядных МП из секций разрядностью 2, 4 или 8.

Технология изготовления является одним из определяющих факторов в микропроцессорной технике. От нее зависят параметры МП по быстродействию, температурному диапазону, потребляемой мощности, стоимости и др. Так, при использовании различных технологий (*p*-МОП, *n*-МОП, КМОП, И²Л, ТТЛШ, ЭСЛ) скорость выполнения операций типа «регистр — регистр» возрастает от 100 тыс. оп/с (для *p*-МОП технологии) до 10 млн. оп/с и более (для ЭСЛ-технологии).

По способу управления МП подразделяются на микропрограммируемые и с жестким управлением. Микропрограммное управление характерно для секционных МП с наращиваемой разрядностью и позволяет пользователю установить собственный набор операций и команд, оптимальных для реализации определенных задач. Жесткое (аппаратное) управление, характерное для однокристальных МП, принципиально такой возможности не допускает.

Спектр параметров, по которым можно классифицировать МП, чрезвычайно широк. Возможна классификация МП по характеру временной работы — синхронные и асинхронные МП (в синхронных МП начало и конец выполнения операций задаются устройством управления, а в асинхронных МП начало выполнения каждой следующей операции определяется по сигналу фактического выполнения предыдущей операции). По количеству выполняемых программ различают одно- и многопрограммные МП (в однопрограммных МП выполняется только одна программа, а в многопрограммных могут одновременно выполняться несколько программ). Можно классифицировать МП и по более мелким признакам, например по числу регистров общего назначения, источников питания и т. д.

§ 1.5. ОБОСНОВАНИЕ ПРИМЕНЕНИЯ И ВЫБОР МИКРОПРОЦЕССОРНОГО КОМПЛЕКТА ДЛЯ ПРОЕКТИРУЕМЫХ РЭС

Выбор оптимального МПК для конкретного применения является наименее решенной из многочисленных проблем развития микропроцессорной техники. Это определяется постоянным ростом количества МПК, расширением области их применения, а также отсутствием четкой методики, позволяющей сделать однозначный выбор МПК.

МП является функционально сложным программно управляемым устройством, выполненным в виде БИС, и характеризуется большим количеством параметров. Поэтому задача выбора оптимального с технической и экономической точек зрения МПК БИС для конкретной области применения является многокритериальной.

При выборе МПК БИС важным является формирование основных требований, предъявляемых к создаваемой аппаратуре. Аппаратура со встроенными микропроцессорами, как правило, должна удовлетворять следующим требованиям:

работа в реальном времени;

повышенная надежность, помехозащищенность, простота обслуживания;

наличие фиксированного набора задач, многократно решаемого на протяжении всего срока службы аппаратуры.

Выбор МПК БИС обычно производится с трех основных позиций:

1) с точки зрения разработки математического обеспечения следует анализировать: разрядность, число и использование регистров общего назначения, набор команд и способы адресаций, наличие и организацию стека;

2) с точки зрения системного проектирования нужно анализировать следующие характеристики МПК БИС: тип архитектуры МП (секционные или однокристалльные) и, как следствие этого, тип организации управления (микропрограммное или с жесткой логикой), наличие логически совместимых БИС из других комплектов, быстродействие МП, возможность прерывания и прямого доступа в память, наличие системы автоматизированного проектирования МПС;

3) с точки зрения разработки аппаратных средств МПС необходимо учитывать: электрическую совместимость БИС, число источников питания и рассеиваемую мощность, размер, тип корпуса и число выводов, диапазон рабочих температур и др.

Выбор МПК БИС для конкретного применения часто осуществляется на основе данных о технологии, по которой он изготовлен.

Проанализируем возможности различных технологий.

Микропроцессоры, изготовленные по *p*-МОП технологии, имеют низкую стоимость, высокую степень интеграции на кристалле, но они являются медленно действующими, не сопрягаемыми со схемами ТТЛ и обладают низкой нагрузочной способностью по выходам. Обыч-

но используются в бытовой радиоаппаратуре, фототехнике, микрокалькуляторах (К145).

Микропроцессоры, изготовленные по *n*-МОП технологии, имеют самую высокую плотность компоновки, высокое быстродействие, являются сопрягаемыми со стандартными схемами ТТЛ, но обладают низкой нагрузочной способностью по выходам. Используются в контрольно-измерительной аппаратуре, АСУТП, мини- и микроЭВМ, диалоговых вычислительных комплексах (КР580, К1801, КР1809, К1810 и др.).

Технологии *p*- и *n*-МОП позволяют получать МП на одном кристалле, такие МПК БИС обеспечены аппаратными средствами и имеют развитое программное обеспечение.

Микропроцессоры, изготовленные по КМОП-технологии (комментарной МОП-технологии), обладают умеренным быстродействием, невысокой степенью интеграции, повышенной стоимостью, но имеют такие достоинства, как простота сопряжения с ТТЛ, низкая потребляемая мощность и высокая помехоустойчивость. Находят применение там, где помехи и агрессивная окружающая среда являются важным фактором (военная техника, транспорт), или в автономных системах с малой потребляемой мощностью (КР588 и др.).

Микропроцессоры, изготовленные по биполярной ТТЛШ-технологии (ТТЛ-Шотки), обладают сверхвысоким быстродействием, но они дороги и имеют низкую степень интеграции. ТТЛШ-технология вследствие большой потребляемой мощности и малой сложности кристалла используется главным образом в многокристалльных МП. Применяются в быстродействующих ЭВМ, системах обработки информации, контрольно измерительной аппаратуре и т. д. (К589, КР1802, К1804 и др.).

Микропроцессоры, изготовленные на основе ЭСЛ-технологии (транзисторная логика с эмиттерными связями), имеют сверхвысокое быстродействие, но они дороги, потребляют большую мощность и сложно сопрягаются с интегральными микросхемами, изготовленными по другой технологии. Применяются для создания процессоров больших ЭВМ, в быстродействующих цифровых устройствах и точных приборах (К1800).

Микропроцессоры, изготовленные по И²Л-технологии (интегральная инжекционная логика), имеют высокую помехозащищенность, высокое быстродействие, высокую плотность заполнения кристалла и низкую стоимость. Используются в быстродействующих ЭВМ, системах обработки информации, системах управления верхнего уровня (КР583, КР584 и др.).

В табл. 1.2 приведены основные параметры МПК БИС, выпускаемых отечественной промышленностью.

Комплексный учет всех характеристик МПК БИС весьма затруднителен, так как одни характеристики относятся к МП как к интегральной схеме, а другие — как к устройству ЭВМ или системы обработки информации.

Таблица 1.2

Основные параметры МПК БИС

Серия	Технология	Число БИС	Разрядность (АЛУ)	Тип архитектуры	Применение, аналог
K145IK18	p-МОП	11	4,8	Однокристалльный	Бытовая техника
K145IK19	КМОП	14	4,8	»	То же
KP580*	n-МОП, ТТЛШ	21	8	»	18080 MSC-80
KP581	n-МОП	6	16	»	CM1800, CM1804
K583	И ² Л	11	8n	Секционный	«Электроника 60» Контроллеры
K584	И ² Л	4	4n	»	SBP0400
585	ТТЛШ	8	2n	»	K589 (13000)
K587	КМОП	4	4n	»	ЧПУ
K588*	КМОП	21	16n	»	ЧПУ
K589	ТТЛШ	8	2n	»	I3000
KB1013	КМОП	3	4	ОМЭВМ	Бытовая техника
K1518*	ЭСЛ-ТТЛШ	3	16	Однокристалльный	TDC-1010
K1800	ЭСЛ	10	4	Секционный	M10800
K1801	n-МОП	9	16	Однокристалльный	«Электроника НЦ», «Электроника» МС»
KP1802*	ТТЛШ	15	8n	Секционный	CM1300
KM1804*	ТТЛШ	29	4n	»	CM1600, CM1420, AMD2900, CM1700
K1806*	КМОП	5	16/32	Однокристалльный	Контроллер ЗУ ЦМД
KA1808	И ² Л	4	8	»	Бытовая техника
KA1809	n-МОП	4	—	—	БМК
KM1810*	n-МОП	22	16	Однокристалльный	CM1810, «Искра 1030», EC1840, «Нейрон И6.99», 18086, MSC-86
K1811	n-МОП	5	16	Однокристалльный	
KM1813BE1*	КМОП	1	25	ОМЭВМ	I2920, обработка аналоговых сигналов
K1814	p-МОП	2	4	ОМЭВМ	TMS1000
K1815*	ТТЛШ, И ² Л	9	8	Однокристалльный	МПК ЦОС (БПФ)
K1816*	n-МОП	4	8	ОМЭВМ	18748, 18049, бытовая техника
KP1818	n-МОП	2	8	Однокристалльный	Контроллер НМД
KP1820	n-МОП	2	4	ОМЭВМ	SOP400, бытовая техника
1821*	КМОП	12	8	Однокристалльный	18085
1822*	ТТЛШ	(1990) 10 (1989)	8n	Секционный	Супермини-ЭВМ

Серия	Технология	Число БИС	Разрядность (АЛУ)	Тип архитектуры	Применение, аналог
1824*	КМОП	3 (1989)	16n	Секционный	ЧПУ, бортовая аппаратура
K1827BE1	n-МОП	1	16	ОМЭВМ	
KP1828	ТТЛШ	2	8n	Секционный	Диагностика МПС
KM1831*	КМОП	2 (1989)	16/32	Однокристальный	ДСУ-11АС
1832*	ТТЛШ	12 (1989)	16	»	Конвейерная обработка
1838*	ЭСЛ-ТТЛШ	11 (1989)	16	»	Am2950DC
«Электроника 32П3»*	КМОП	3 (1989)	32	«	

Примечание: * — перспективные МПК БИС.

Одной из основных характеристик, отражающих функциональные возможности МП, может служить его разрядность. Диапазон требуемой разрядности в микропроцессорных РЭС довольно широк. Так, при построении контрольно-измерительных систем и систем сбора данных довольно часто используются 16-разрядные МП, а при построении цифровых фильтров и спектральных анализаторов требуемая разрядность МП возрастает до 32.

После определения допустимого класса МПК БИС, удовлетворяющих поставленным требованиям, производится оценка имеющихся программных возможностей МП. При этом следует учитывать наличие средств автоматизации программирования и отладки программного обеспечения (редактирующие и компилирующие программы, программы согласования и загрузки).

Системы команд МП различаются возможностями организации компактных цепочек команд при программировании различных алгоритмов решаемых задач. Поэтому сравнение лишь числа команд в системах команд или числа регистров общего назначения в МП не дает достаточного основания для правильного выбора МП. Действительно, большое число регистров МП с жестко определенными функциями может быть менее эффективным, чем малое число универсальных РОН. Например, выделение отдельных РОН для хранения адресов и размещения данных обеспечивает большие возможности для перемещения информации в вычислительном процессе.

Для оценки проектируемой микропроцессорной аппаратуры используется эталонное (пробное) программирование. Оно выполняется для заранее определенного набора эталонных задач, отражающих специфику области применения. Знание области применения МП позволя-

ет выделить наиболее специфичные и принципиальные части алгоритмов и составить программы для всех допустимых типов МП с целью получения эксплуатационных характеристик МПУ. Из них наиболее важны такие, как общее время выполнения программы, требуемая емкость ОЗУ, время реакции МПУ на внешние сигналы и др.

В результате эталонного программирования получаются реальные эксплуатационные характеристики применения конкретного МП в конкретной системе. Вместе с информацией завода-изготовителя полученные данные, как правило, достаточны для оценки МП и МПУ или микроЭВМ на его основе.

Важнейшим фактором, влияющим на выбор МПК, является наличие программного обеспечения с сопутствующими ему средствами отладки и трансляции. Это определяется тем, что без программного обеспечения МП не может использоваться, а стоимость его разработки может существенно превышать стоимость МПС. При оценке программного обеспечения как критерия выбора МП следует учитывать альтернативу аппаратурного решения задачи. Микропроцессорные системы, использующие аппаратные средства решения задач, являются более быстродействующими против МПС с программной реализацией их решения, но последние более гибкие. Кроме того, различны составляющие общих затрат на функционирование МПС: аппаратурные средства решения задач — низкие затраты на разработку и высокие — на производство; при программном решении задач наблюдается противоположная картина. При выборе МП важно учитывать также сложность программирования, которая определяется системой команд и архитектурой МП.

Микропроцессоры — слишком сложные устройства, чтобы их можно было однозначно охарактеризовать перечислением всевозможных технических характеристик, тем более что ряд характеристик не может быть оценен количественно (например, сложность программирования). В табл. 1.3 по материалам фирмы Integrated Computer System проиллюстрирована значимость показателей МП по четырем критериям выбора (здесь введены следующие обозначения: — безразлично, + не очень важно, ++ важно, +++ очень важно).

Рассмотрим некоторые методы использования количественных характеристик при оценке и выборе наилучшего варианта МПК БИС. Прежде всего это производительность решения заданного набора задач на конкретном МПК БИС, которая зависит от различных факторов: мощности программного обеспечения, возможностей аппаратной поддержки, максимально допустимой тактовой частоты и др. Среди множества существующих методов оценки и сравнения производительности различных МПК БИС выделим метод оценки по скорости обработки данных ν , которая измеряется количеством бит, обрабатываемых в секунду:

$$\nu = \sum_{j=1}^M g_j r_j \nu / \alpha_j.$$

Значимость показателей МП для различных критериев выбора

Показатели	Скорость обработки программы	Гибкость в применении	Затраты на разработку	Стоимость производства
Длина данных	+++	+++	++	+++
Набор команд и адресация	++	++	++	+
Архитектура центрального процессорного устройства	++	++	++	+
Микропрограммируемость	+++	+++	+++	+
Цикл выполнения команды	+++	+++	—	++
Возможность использования прерываний	++	+++	++	+
Набор схем	+	+	+++	+++
Выбор запоминающих устройств	—	—	+	+++
Питание	—	++	+	++
Средства автоматизации проектирования	—	+	+++	+
Наличие аналогов	—	+++	—	—
Стоимость	—	—	+	+++
Прочность	—	+++	—	+++

Здесь q_j — относительная частота появления j -й команды, $j = \overline{1, M}$; r_j — количество обрабатываемых бит в j -й команде; α_j — коэффициент сложности выполнения j -й команды по отношению к элементарной команде, например, к команде пересылки «регистр — регистр», $\alpha_j \geq 1$; v — скорость выполнения элементарной операции; для МП КР580 $v = 500$ тыс. оп/с, для МП КР1804 $v = 10$ млн. микрокоманд/с, для МП К1810 $v = 1$ млн. оп/с, для МП К1800 $v = 50$ млн. микрокоманд/с.

Наиболее полно производительность МП можно оценить по смеси команд через производительность P , определяемую количеством выполняемых команд в секунду:

$$P = \sum_{i=1}^m \sum_{j=1}^{s_i} g_i n_{ij} \left/ \sum_{i=1}^m \sum_{j=1}^{s_i} g_i \alpha_j n_{ij} \tau \right.,$$

где g_i — относительная частота появления i -й программы в течение определенного цикла работы МП; n_{ij} — число команд j -го типа, используемых при реализации i -й программы; s_i — число типов команд в i -й программе ($s_i \leq M$); τ — время выполнения элементарной команды (например, команды пересылки «регистр — регистр», $\tau_j = \alpha_j \tau$).

Очевидно, при выборе МПК по критерию производительности решения заданного класса задач предпочтение следует отдавать МПК, характеризующемуся максимальным значением v или P .

Оценка технического уровня МПК БИС по совокупности ряда количественных показателей может быть произведена с помощью свертки показателей, используемых для оценки качества проектируемой аппаратуры. Существуют различные формы представления показателя качества, одной из которых является оценка Φ_j по среднеквадратическому отклонению «взвешенных» значений параметров j -го типа МПК БИС от значений параметров «директивного» (желаемого) комплекта. Суть такой оценки заключается в следующем.

Любой МПК БИС характеризуется множеством параметров $\Theta = \{\vartheta_1, \dots, \vartheta_n\}$, каждый из которых может быть оценен количественно. Будем считать, что первые m параметров $\vartheta_1, \dots, \vartheta_m$ желательно максимизировать (быстродействие, среднее время наработки на отказ, емкость оперативной памяти и др.), а остальные $n - m$ параметров $\vartheta_{m+1}, \dots, \vartheta_n$ — минимизировать (стоимость, потребляемая мощность, габариты и др.). Другими словами, множество параметров Θ можно разбить на два непересекающихся подмножества $\Theta' = \{\vartheta_1, \dots, \vartheta_m\}$ и $\Theta'' = \{\vartheta_{m+1}, \dots, \vartheta_n\}$, причем $\Theta = \Theta' \cup \Theta''$; $\Theta' \cap \Theta'' = \emptyset$. Поскольку все параметры $\vartheta \in \Theta$ имеют различную размерность, для получения количественных характеристик, позволяющих сравнивать реальные значения параметров МПК с директивными, необходимо привести все параметры к безразмерному виду. Эти преобразования должны удовлетворять определенным требованиям: иметь общее начало отсчета, один порядок изменения относительных значений параметров и сохранять отношение предпочтения между абсолютными и относительными значениями параметров.

В качестве такого преобразования можно выбрать монотонную функцию вида

$$f_{ij} = \begin{cases} \vartheta_{ij}/\vartheta_{i0}, & \vartheta_i \in \Theta', \\ \vartheta_{i0}/\vartheta_{ij}, & \vartheta_i \in \Theta''. \end{cases} \quad (1.1)$$

Здесь f_{ij} — безразмерная форма задания i -го параметра j -го микропроцессорного комплекта; ϑ_{ij} и ϑ_{i0} — соответственно значения i -го параметра j -го и директивного комплектов БИС; $\vartheta_{ij} \in \Theta_j$, $\vartheta_{i0} \in \Theta_0$.

Очевидно, если хотя бы один из параметров $\vartheta_{ij} \in \Theta_j$ j -го микропроцессорного комплекта не удовлетворяет соответствующему директивному значению и директивные требования изменить нельзя, то j -й комплект следует исключить из рассмотрения. Поэтому будем рассматривать ситуацию, когда при выборе микропроцессорного комплекта все n параметров удовлетворяют директивным требованиям, т. е. для всех $\vartheta_i \in \Theta'$ справедливо условие $\vartheta_{ij} \geq \vartheta_{i0}$, а для $\vartheta_i \in \Theta''$ — соответственно $\vartheta_{ij} \leq \vartheta_{i0}$. При выполнении данных условий преобразования вида (1.1) всегда имеют значения, меньшие или равные единице.

Для любых МПК БИС улучшение одних параметров, как правило,

приводит к ухудшению других. Это можно проследить по характеру изменения различных функций, в частности быстродействие — стоимость, разрядность — потребляемая мощность, быстродействие — разрядность и т. д. Поэтому с точки зрения экономических факторов для оценки j -й серии МПК целесообразно использовать показатель качества Φ_j вида

$$\Phi_j = \sqrt{\sum_{i=1}^n [(1-f_{ij})(1-c_i)]^2}, \quad (1.2)$$

где c_i — весовой коэффициент i -го параметра, причем

$$0 \leq c_i \leq 1, \quad \sum_{i=1}^n c_i = 1. \quad (1.3)$$

Показатель качества Φ_j характеризует среднеквадратическое отклонение множества нормированных значений j -й серии МПК БИС от значений множества директивных параметров с учетом степени важности каждого параметра в данной конкретной области применения МПК. Поскольку улучшение любого из параметров МПК приводит к повышению стоимости комплекта, с экономической точки зрения целесообразно, чтобы значение величины f_{ij} стремилось к единице, а показатель Φ_j — к минимуму. Поэтому при выборе МПК БИС предпочтительнее следует отдавать серии с номером q , для которой $\Phi_q = \min_j \{\Phi_j\}$.

Для получения единственного решения в задаче выбора микропроцессорного комплекта требуется задать весовые коэффициенты c_i , $i = \overline{1, n}$, удовлетворяющие соотношению (1.3) и отражающие относительную важность параметров. В литературе, посвященной математическим методам принятия сложных решений, описано множество методов определения весовых коэффициентов, наиболее эффективными из которых для решения описываемой задачи оказались методы ранжирования и приписывания баллов.

Метод ранжирования заключается в том, что группа из l специалистов в исследуемой области производит экспертизу. Каждого эксперта просят расставить все параметры $\vartheta_i \in \Theta$ в порядке их важности. При этом цифрой 1 обозначают наиболее важный (с точки зрения эксперта) параметр, цифрой 2 — следующий по важности параметр и т. д. Эти ранги преобразуются таким образом: ранг 1 получает оценку n , ранг 2 — оценку $n - 1$ и т. д. до ранга n , который получает оценку 1. Зная преобразованный ранг $r_i^{(k)}$ параметра ϑ_i у k -го эксперта, весовые коэффициенты c_i находят из соотношения

$$c_i = \frac{\sum_{k=1}^l r_i^{(k)}}{\sum_{i=1}^n \sum_{k=1}^l r_i^{(k)}}, \quad i = \overline{1, n}. \quad (1.4)$$

Метод приписывания баллов состоит в том, что эксперты оценивают важность каждого параметра по некоторой шкале (например, по шкале 0—10). При этом им разрешается нескольким параметрам приписывать одну и ту же оценку (в том числе и дробными величинами). Зная балл $h_i^{(k)}$ i -го параметра у k -го эксперта, весовые коэффициенты c_i можно найти из соотношений (1.4) с учетом того, что значения $r_i^{(k)} = h_i^{(k)} / \sum_{i=1}^n h_i^{(k)}$. В этом случае величину $r_i^{(k)}$ называют весом, подсчитанным для параметра Φ_i на основе оценок k -го эксперта.

Каждый МПК содержит различное число БИС, что может вызвать значительные трудности при определении количественных значений параметров. Поэтому выбор МПК рекомендуется производить либо по характеристикам центрального процессорного элемента МПК, либо путем анализа параметров наиболее типовых структур МПУ, используемых в разрабатываемых РЭС.

Контрольные вопросы и задания

1. Объясните основные закономерности развития РЭС.
2. В чем состоят принципиальные отличия микропроцессорных РЭС от радиоэлектронной аппаратуры ранних поколений?
3. Укажите основные области применения микропроцессорных комплексов БИС.
4. Выделите основные уровни и аспекты проектирования конкретного образца микропроцессорных РЭС.
5. Приведите примеры нисходящего и восходящего проектирования.
6. Перечислите и охарактеризуйте основные технологии изготовления МПК БИС.
7. Объясните назначение магистралей в МП. Почему магистральный принцип является доминирующим при построении МПУ и МПС?
8. В чем заключается принципиальная разница в использовании универсальных микроЭВМ и встроенных МПУ?
9. Почему задача выбора МПК БИС для конкретных применений является многокритериальной?
10. Приведите примеры проектирования РЭС на базе детерминированного и статистического подходов к проектированию.
11. Как осуществляется взаимосвязь цифровых и аналоговых устройств в микропроцессорных РЭС?
12. Почему в настоящее время появилась тенденция отдавать предпочтение цифровым методам обработки информации по сравнению с аналоговыми?
13. Спрогнозируйте пути развития микропроцессорной техники в рамках создания новых однокристалльных МП и микроЭВМ.
14. Чем объясняется необходимость создания и развития секционированных МПК БИС?
15. В чем состоит принципиальное отличие систем с жесткой и программируемой логикой?
16. Какие методы экспертных оценок применяются в процессе выбора МПК БИС?
17. Предложите и обоснуйте критерии выбора МПК БИС для конкретного микропроцессорного радиоэлектронного средства.

АРХИТЕКТУРА МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ

§ 2.1. ДУАЛИЗМ «ПРОГРАММНЫЕ СРЕДСТВА — АППАРАТУРА» В МИКРОПРОЦЕССОРНОЙ ТЕХНИКЕ

Проектирование микропроцессорных РЭС производится на базе определенных МПК БИС. Разработчикам аппаратуры на основе МП недоступен уровень отдельных транзисторов, связей между ними. Микропроцессор воспринимается как нечто цельное, имеющее различные свойства, заложенные в его архитектуре.

Архитектура микропроцессора — это его логическая организация, определяемая возможностями МП по аппаратурной или программной реализации функций, возлагаемых на проектируемые МПУ или микроЭВМ. Архитектура отражает структуру МП, способы представления и форматы данных, набор операций, форматы управляющих слов, способы обращения ко всем доступным для пользователя элементам структуры, реакцию МП на внешние сигналы.

При разработке микропроцессорных РЭС необходимо определить, какая часть функций при создании конкретных РЭС должна быть реализована программным способом, а какая — с помощью дополнительных аппаратурных средств. Поэтому при проектировании микропроцессорных РЭС необходимо: 1) дать описание концептуальной модели функционального поведения РЭС и рекомендации по организации вычислительного процесса в микропроцессорной аппаратуре; 2) определить структуру, номенклатуру и особенности построения программных и микропрограммных средств; 3) описать характеристики внутренней организации потоков данных и управляющей информации; 4) провести анализ функциональной структуры и особенностей физической реализации устройств микропроцессорных РЭС с позиций сбалансированности программных, микропрограммных и аппаратурных средств.

При разработке архитектуры микропроцессорных устройств определяют форматы данных и команд, систему команд и методы адресации, обосновывают требования к интерфейсам. Правильный выбор архитектуры дает возможность оптимизировать вычислительный процесс реализации алгоритмов функционирования МПУ на выбранных аппаратурных средствах. В МПУ процесс оптимизации начинается с решения компромисса «программные средства — аппаратура», который заключается в том, что в микропроцессорной технике любое функциональное преобразование может быть осуществлено как программным путем, так и на аппаратурных средствах. При этом для реализации любых программных средств необходима аппаратурная поддержка и, наоборот, ни одно из аппаратурных средств МПУ не может функцио-

цировать без программного обеспечения. Иногда компромисс «программные средства — аппаратура» называют дуализмом в микропроцессорной технике.

Для построения МПУ и микроЭВМ на основе микропроцессоров состав МПК БИС должен содержать широкую номенклатуру микросхем, в частности: микропроцессоры однокристалльные либо секционные в виде модулей-секций на несколько двоичных разрядов; программируемые логические матрицы и более совершенные программируемые матричные логические схемы, матрицы логических элементов и специальные БИС (интерфейсные, умножители и т. д.). Решение проблемы дуализма «программные средства — аппаратура» является довольно сложным из-за наличия большого количества аппаратурных средств поддержки и широких возможностей создания разнообразного программного обеспечения. Чем больше сложных функций в МПУ или микроЭВМ реализуется аппаратурно, тем сложнее их архитектура и тем больше разновидностей БИС и ИМС требуется для их создания. Увеличение номенклатуры элементов в МПУ или микроЭВМ приводит к увеличению их стоимости и вызывает серьезные трудности при разработке и создании специальных БИС и ИМС. Упрощение аппаратуры связано с усложнением программного обеспечения. Для решения этой проблемы создают вычислительные модули, имеющие возможность реализовать в них прямое восприятие языков высокого уровня, что позволяет решить проблему автоматизации программирования.

Создание микропроцессоров привело к новой концепции построения РЭС. Если в традиционных РЭС использовались схемы с жесткими связями между элементами, за счет которых фиксировались определенные алгоритмы функционирования РЭС, то в микропроцессорных РЭС алгоритмы реализуются программным методом, а аппаратура остается практически одинаковой для разнообразных применений. Поскольку микропроцессор — программно-управляемый прибор, при проектировании МПУ необходимо закладывать оптимизацию уровней аппаратно-программных средств, а затем «настраивать» МПУ на решение конкретных задач по конкретным алгоритмам только за счет разработки программ, отражающих специфику данных РЭС.

В МПУ задачи решаются программным путем методами последовательных вычислений, что приводит к большим затратам времени по сравнению с функционированием РЭС, реализованных на основе только схемотехнических принципов. Для функционирования микропроцессорных РЭС в состав аппаратуры обязательно должны входить следующие функциональные части: подсистемы ввода—вывода данных, обработки данных, накопления и хранения данных об особенностях внутренней структуры и функционирования аппаратуры (объекта, технологического процесса и т. д.). Оборудование, в которое встраивается МПУ, должно быть оснащено датчиками, определяющими режимы его отдельных частей или технологического процесса, а также исполнительными механизмами, позволяющими влиять на управляемый процесс. При этом должны быть осуществлены согласования пара-

метров всех электрических сигналов, различные преобразования аналоговых сигналов в цифровые и, наоборот, световых в электрические и т. д.

§ 2.2. ТИПОВЫЕ СТРУКТУРЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Любая система обработки информации, построенная на основе микропроцессора, содержит большое число функциональных устройств, главным из которых является микропроцессор. Все устройства должны иметь стандартный интерфейс для подключения к информационной магистрали.

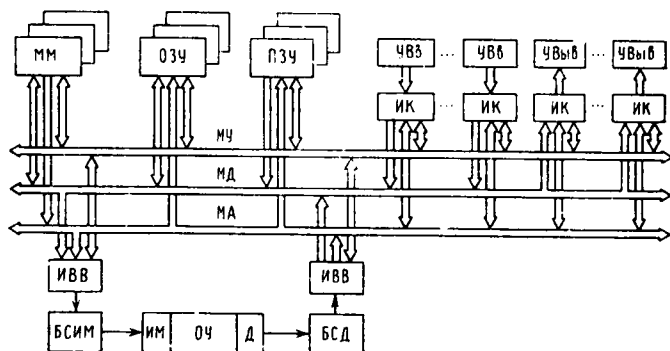


Рис. 2.1. Логическая структура МПС

В любой МПС и микроЭВМ микропроцессор выполняет роль центрального устройства управления и устройства арифметико-логического преобразования данных. В качестве устройства управления он генерирует последовательности синхронизирующих и логических сигналов, которые определяют последовательность срабатывания всех устройств и компонентов системы. Микропроцессор задает и последовательно осуществляет микрооперации извлечения команд программы из памяти системы, их расшифровку и исполнение, тем самым осуществляя реализацию принципа программного управления.

В состав любой МПС входят система памяти, система ввода—вывода информации и система сопряжения с объектом управления или контроля (рис. 2.1).

Память МПС физически реализуется в виде иерархической системы, состоящей из ряда уровней. Верхние уровни памяти строятся на основе полупроводниковых постоянных и оперативных запоминающих устройств, а нижние — на основе магнитных внешних запоминающих устройств.

Полупроводниковые постоянные запоминающие устройства (ПЗУ) предназначены для хранения заранее записанных данных и используются только в режиме чтения информации. Они энергонезависимы,

т. е. сохраняют информацию при выключении питания и имеют высокую скорость работы. Полупроводниковые оперативные запоминающие устройства (ОЗУ) работают в режимах оперативной записи и чтения данных в темпе работы МП. Полупроводниковые ОЗУ энергозависимы, т. е. теряют информацию при выключении питания.

Устройства ввода данных (У Вв) — технические средства, предназначенные для передачи данных извне в регистры МП или в память (клавиатура пульта управления, ввод с перфолент и перфокарт, внешние ЗУ на магнитных лентах, кассетах, дисках, дисплеи и т. д.). Устройства вывода данных (У Выв) — технические средства, способные воспринимать данные, передаваемые из регистров МП или ячеек памяти (дисплеи, печатающие устройства, внешние ЗУ, пульт управления, графопостроители и т. д.). Для подключения устройств ввода—вывода необходимо привести все их связи и сигналы к стандартному виду, т. е. провести согласование интерфейсов. Для этого используют специальный блок — интерфейс ввода—вывода (ИВВ), имеющий стандартный интерфейс со стороны подключения информационной магистрали и нестандартный интерфейс со стороны устройств ввода—вывода.

Для сопряжения объекта управления или контроля (технологического процесса, радиотехнической системы и др.) с МПС в состав объекта должны входить датчики состояния Д и исполнительные механизмы ИМ, при этом датчики выступают в качестве источников вводимой в МПС информации, а ИМ — в качестве приемников выводимой информации. Для согласования интерфейсов подключение датчиков и исполнительных механизмов в системе осуществляется через блоки сопряжения датчиков БСД и исполнительных механизмов (БСИМ).

Практика использования микропроцессоров позволяет выделить четыре основных направления в их применении:

- встроенные системы управления и контроля;
- локальные системы накопления и обработки информации;
- распределенные системы управления сложными объектами;
- распределенные высокопроизводительные системы параллельных вычислений.

Встраивание микропроцессора в любую систему принципиально изменяет качество функционирования всех ее компонентов (агрегатов технологической линии, отдельных устройств и т. д.). Встраивание микропроцессоров позволяет оптимизировать режимы работы управляемых объектов или процессов. За счет этого получается технико-экономический эффект и существенно снижаются требования к персоналу, работающему на оборудовании. Микропроцессоры, встраиваемые в оборудование, как правило, не комплектуются внешними устройствами и содержат только ПЗУ управляющих программ и специализированный пульт управления.

Локальные системы накопления и обработки информации технически просто и экономически доступно осуществляют информационное обеспечение специалистов непосредственно на их рабочих местах.

Распределенные системы управления сложными объектами предназначены для обработки информации в местах ее возникновения и для сбора обработанной информации с целью выработки управляющих воздействий. В распределенных системах достигается значительная экономия в количестве и распределении линий связи, повышается живучесть системы.

Распределенные высокопроизводительные системы параллельных вычислений состоят из множества однотипных или специализированных на определенные задачи МП, которые ведут параллельную обработку информации.

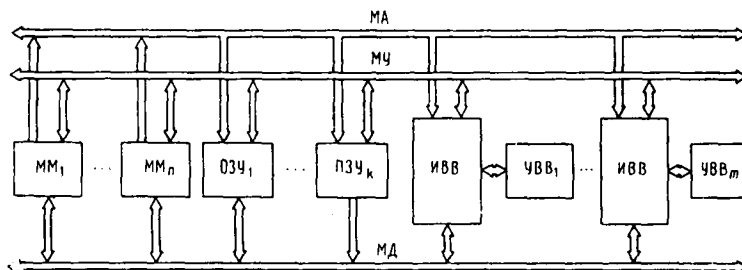


Рис. 2.2. Структурная схема магистральной МПС

С точки зрения территориально-пространственного построения МПС подразделяются на рассредоточенные территориально и на сосредоточенные в одном месте.

Территориально распределенные МПС для повышения живучести системы и эффективности обработки информации содержат микропроцессоры, рассредоточенные по всему объекту управления. Сосредоточенные МПС содержат множество одновременно работающих МП, действующих по принципу параллельной обработки данных. Такие системы обеспечивают высокую производительность за счет параллельного выполнения отдельных частей задачи и высокую живучесть за счет резервирования однотипных МП и распределения функций между отдельными МП. Рассмотрим основные структуры сосредоточенных и распределенных МПС.

Структуры сосредоточенных МПС. Сосредоточенные МПС используются, как правило, в системах параллельной обработки информации и состоят из ряда вычислительных модулей, решающих элементарные задачи, и каналов взаимодействия информацией между модулями. Все множество структур сосредоточенных МПС можно разбить на следующие структурные реализации или их комбинации.

Магистральные МПС. В МПС с магистральной структурой все устройства (микропроцессорные модули ММ, секции ОЗУ и ПЗУ, устройства ввода—вывода и др.) подключаются к магистралям данных, адреса и управления (рис. 2.2). Микропроцессорный модуль — типовая легко заменяемая часть МПУ или микроЭВМ, как правило, конст-

руктивно выполненная на одной плате и проблемно-ориентированная на решение определенных задач. Каждый ММ обычно содержит МП и локальную память, регистры и вспомогательные логические схемы для обеспечения передачи данных и управляющих сигналов, а также для осуществления связей между ММ. Стоимость реализации МПС с магистральной структурой невысока, но производительность ограничивается пропускной способностью магистралей. В МПС с магистральной структурой возможно наличие управляющего ММ.

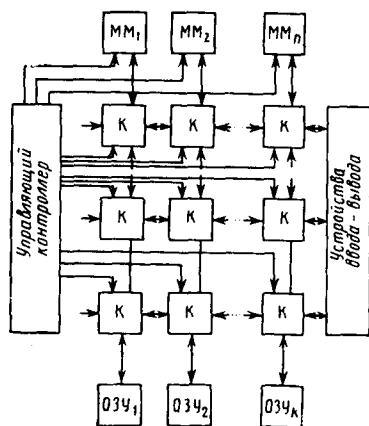


Рис. 2.3. Структурная схема матричной МПС

Матричные МПС. Системы этого типа являются регулярными МПС с изменяемой структурой, в которых одинаковые ММ соединяются

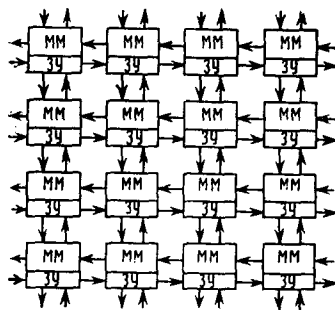


Рис. 2.4. Структурная схема двумерной матричной МПС

между собой с помощью координатного (коммутирующего) искателя К (рис. 2.3). В процессе функционирования ММ одновременно выполняют один и тот же поток команд над различными потоками данных под управлением управляющего контроллера (микроЭВМ).

Один из вариантов матричной МПС представляет собой двумерный массив ММ, каждый из узлов которого соединен с четырьмя соседними ММ. При этом каждый промежуточный ММ определяет, кто из соседних ММ будет следующим адресатом. Частным случаем такой структуры являются МПС, в которых отдельные ММ объединяются через запоминающие устройства (рис. 2.4). При этом каждый ММ соединен со своими ЗУ и ЗУ соседних ММ односторонними линиями связи.

Матричную МПС можно рассматривать как некоторый массив ММ и ЗУ, позволяющий построить граф для оптимального решения данной задачи. При этом выделять блоки, решающие части задач параллельно и последовательно, и составляют оптимальную структурную схему МПС в виде графа, вершинам которого соответствуют параллельные ветви алгоритма, а дугам — связи между ними. В подобной структуре можно одновременно выделить несколько конфигураций (под-

систем), предназначенных для решения различных задач. Планирование размещения задач на массиве ММ и присвоение задачам приоритетов осуществляют программно. Систему можно программировать таким образом, чтобы формировать произвольную комбинацию связей и динамически перестраивать ее в процессе работы МПС.

Преимущества матричных МПС: высокая производительность решения задач, допускающих высокую степень параллелизма; малое отношение «стоимость — производительность» за счет высокой степени модульности МПС; простота расширения системы за счет увеличения только емкости коммутаторов; высокая степень однородности и большая живучесть. Основные недостатки: трудность программной ор-

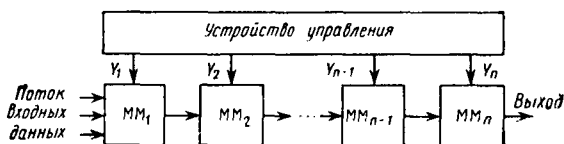


Рис. 2.5. Структурная схема конвейерной МПС

ганизации работы системы; фиксированность связей между отдельными ММ; большая избыточность аппаратных средств при решении нерегулярных задач.

Конвейерные МПС. В конвейерной МПС (рис. 2.5) поток данных проходит последовательно через одновременно работающие ММ, каждый из которых выполняет некоторую часть преобразований из общего преобразования и полученный частичный результат передает следующему ММ. Конвейерные МПС являются специализированными и ориентированными на реализацию однообразных операций Y_i в каждом $ММ_i$ над поступающим потоком данных.

Структуры распределенных МПС. Принципы построения МПС обработки и управления процессами с распределенной архитектурой основываются на функциональной и топологической децентрализации.

Целью *функциональной децентрализации* является снижение сложности управляющей системы путем декомпозиции (разделения, расчленения) сложного процесса или функции на составные части — подпроцессы или подфункции, с тем чтобы обеспечить повышенную надежность и эффективность системы. Оптимальная функциональная децентрализация подразумевает разбиение процесса на минимально связанные между собой подпроцессы.

Топологическая децентрализация предполагает пространственное распределение датчиков, исполнительных механизмов и аппаратуры обработки информации (МПУ, микроЭВМ и др.). Одним из критериев оптимальности при топологической децентрализации может служить критерий минимизации суммарной длины кабеля. Следует отметить, что, как правило, оптимальная функциональная децентрализация мо-

жет совпадать с оптимальной топологической децентрализацией. В этих случаях при выборе окончательного решения оценивают экономические факторы.

Распределенные МПС имеют следующие базовые структурные конфигурации: звездообразную (радиальную), кольцевую и смешанную. В настоящее время находит широкое применение реализация распределенных МПС в виде локальных вычислительных сетей.

Радиальные МПС. Системы подобного типа (рис. 2.6) содержат центральный ММ (ЦММ), к которому с помощью двунаправленных шин подключаются МПУ или микроЭВМ. Сообщения между периферийными МПУ и микроЭВМ передаются через ЦММ, обеспечивающий

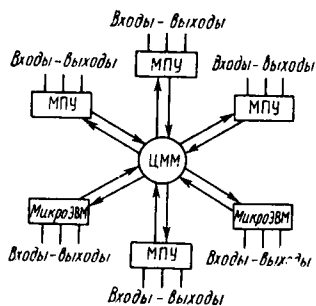


Рис. 2.6. Структурная схема радиальной МПС

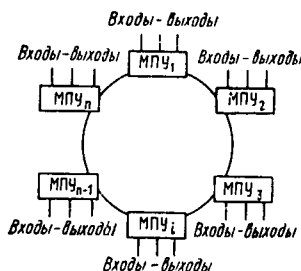


Рис. 2.7. Структура кольцевой МПС

независимость процессов, происходящих в отдельных МПУ, от структуры сообщения, а также защиту устройств друг от друга. В этой системе реализуются принципы параллельности и частичной модульности (для периферийных устройств), поэтому ее живучесть при отказе отдельных МПУ или микроЭВМ высока. Однако наличие ЦММ, выполняющего функции коммутатора и отказ которого приводит к отказу всей системы, снижает живучесть и мобильность системы.

Кольцевые МПС. Системы этого типа имеют кольцевое (контурное) соединение МПУ или микроЭВМ, при котором каждое МПУ связано с двумя соседними (рис. 2.7). В кольце может одновременно циркулировать одно или несколько сообщений фиксированной либо переменной длины. Сообщение циркулирует по кольцу от источников к адресатам на основании заданных критериев управления. Промежуточное МПУ выполняет несколько функций: транслирует сообщения к нужному МПУ, распознает и обрабатывает сообщения, направленные к нему самому, и т. п.

Кольцевая архитектура МПС использует свойства параллельности и модульности. При расширении системы дополнительные МПУ включаются в контур системы. Живучесть системы в целом определяется надежностью отдельного МПУ, хотя при правильной организации ее

структуры отказ одного МПУ не приводит к отказу всей системы, а лишь ухудшает ее функциональные характеристики (система становится линейной). Недостатками кольцевых МПС являются снижение их пропускной способности при росте числа сообщений в системе и сложность программного обеспечения, вызванная сложностью протоколов взаимодействия.

Смешанная структура МПС. Подобные МПС представляют собой комбинацию радиальной и кольцевой структур.

Вопросы построения локальных вычислительных систем рассмотрены в гл. 5.

§ 2.3. ИНТЕРФЕЙС МИКРОПРОЦЕССОРОВ

Для включения микропроцессора в любую систему необходимо установить единые принципы и средства его сопряжения с остальными устройствами системы, т. е. выбрать унифицированный интерфейс.

Унифицированный интерфейс — совокупность правил, устанавливающих единые принципы взаимодействия устройств МПС. В состав интерфейса входят аппаратурные средства соединения устройств (разъем и связи), номенклатура и характер связей, программные средства, описывающие характер сигналов интерфейса и их временную диаграмму, а также описание электрофизических параметров сигналов.

Связь между устройствами МПС осуществляется через информационные магистрали, функционально состоящие из магистралей адресов, данных и управления.

Магистраль адресов. В простых МПС, в которых все устройства, кроме микропроцессора, являются пассивными, только МП может вырабатывать адреса передаваемой в системе информации. Поэтому в таких МПС магистраль адресов МА — однонаправленная: микропроцессор генерирует сигналы кода адреса, а остальные устройства, подключенные к МА, могут только воспринимать их, выполняя непрерывно микрооперацию опознания кода адреса. Количество линий связи в МА совпадает с разрядностью передаваемого кода адреса.

Магистраль данных. По магистрали данных МД передаются данные, которыми обмениваются устройства МПС (МП, ОЗУ, дисплей и др.). Для обеспечения обмена данными между различными устройствами системы МД реализуется двунаправленной. Разрядность МД определяется разрядностью микропроцессора. Если в МП обрабатываются данные по программам двойной разрядности, то слово двойной длины пересылается за два цикла, т. е. имеется временное мультиплексирование.

Магистраль управления. По магистрали управления МУ передаются управляющие сигналы, предназначенные для синхронизации и определения операций в устройствах МПС. Эти сигналы передаются по совокупности линий связи, в целом образующих магистраль сигналов управления. Все сигналы управления в МПС согласованы с системными сигналами синхронизации. Эти сигналы задают начало и по-

следовательность срабатывания различных устройств системы, а также различных блоков и узлов внутри всех кристаллов БИС. Для задания главной последовательности синхронизирующих импульсов применяется внешний кварц или генератор на его основе. Выдаваемые микропроцессором сигналы синхронизации бывают однофазными, реже двухфазными. Каждый МП имеет уникальную систему сигналов управления. Поэтому конкретное описание всех шин МУ, так же как и цоколевки выводов корпуса, дается в технической документации на конкретный МП.

Интерфейсные схемы. Для подключения внешних устройств к центральной части МПУ (МП и ОЗУ) и организации обмена информацией между ними служат интерфейсные схемы. Эффективность и производительность МПУ в значительной степени зависят от возможностей схем сопряжения. В МПУ интерфейсные схемы выполняют следующие функции:

- буферизацию данных, адресов и команд, что необходимо для синхронизации обмена информацией;

- дешифрацию адресов, выбор внешнего устройства ВУ при работе в МПУ с несколькими ВУ;

- дешифрацию команд при работе с такими ВУ, которые, кроме собственно передачи данных, выполняют и другие действия, например обратную перемотку магнитной ленты;

- синхронизацию и формирование управляющих сигналов, необходимых для правильной реализации любой из упомянутых функций.

Помимо перечисленных каждая конкретная интерфейсная схема может выполнять также ряд специфичных функций.

Обмен информацией между МП, ОЗУ и ВУ реализуется через интерфейсные схемы с помощью:

- программно-управляемого обмена (по инициативе МП), при котором задача интерфейсной схемы состоит в организации обмена информацией между МП и ВУ параллельным, последовательным либо параллельно-последовательным способом;

- обмена в режиме прерывания (по инициативе ВУ), при котором интерфейсная схема осуществляет организацию запроса и выдачу начального адреса подпрограммы обслуживания прерывания;

- обмена в режиме прямого доступа к памяти (по инициативе ВУ), при котором интерфейсная схема осуществляет организацию запроса на прямой доступ к памяти и передачу данных, минуя МП.

Программно-управляемый обмен данными. В этом виде обмена МП программным путем должен определить, готово ли периферийное устройство к выполнению операций ввода-вывода до того, как начнется программная передача данных. Внешнее устройство должно иметь аппаратные средства для выработки информации о внутреннем состоянии устройства. Микропроцессор считывает эту информацию, передает ее во внутренний регистр — аккумулятор, анализирует и на основе результата анализа принимает решение о готовности устройства. Обобщенная структура интерфейсной схемы при программно-управляемом

обмене данными представлена на рис. 2.8, на котором показаны варианты подключения ВУ четырех типов: двух устройств ввода информации параллельным и последовательным кодами и двух устройств вывода информации также параллельным и последовательным кодами. На рис. 2.8 изображена единая интерфейсная схема для подключения ВУ при программно-управляемом обмене, однако на практике для управления ВУ с параллельной и последовательной передачей данных, как правило, используются различные интерфейсные схемы. Естественно, что при построении МПУ число используемых ВУ может варьироваться в достаточно широких пределах.

Обмен в режиме прерывания. При обмене данными по сигналам прерывания внешние устройства являются сами инициаторами обмена. Микропроцессор работает параллельно с ними над выполнением основной программы и не занимается анализом состояния внешних устройств. Для организации обмена по прерываниям в МП предусматриваются специальные аппаратные средства для анализа состояния ВУ, которые сигнализируют блоку управления микропроцессора о поступлении сигналов готовности к обмену какого-либо ВУ. При поступлении сигнала прерывания МП завершает текущую операцию, передает на хранение в память всю информацию внутренних регистров данных и управления и осуществляет переход к подпрограмме обслуживания прерывания. После осуществления процедуры обмена информацией по прерыванию происходит восстановление состояния микропроцессора, которое существовало к началу прерывания.

Структура типичной интерфейсной схемы для организации обмена информацией в режиме прерывания показана на рис. 2.9. В качестве входной информации на интерфейсную схему поступают запросы на

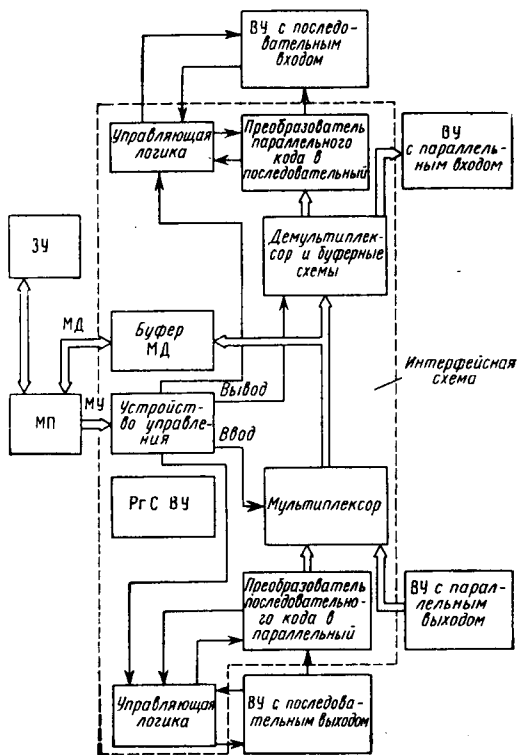


Рис. 2.8. Структура интерфейсной схемы

прерывание от ВУ и приоритет текущей программы. На выходе схемы формируются запрос на прерывание на МП и адрес первой ячейки подпрограммы обслуживания прерывания.

Обмен в режиме прямого доступа к памяти. Для осуществления обмена информацией между внешним устройством и памятью нет необходимости пересылать данные через МП. Обычно обмен данными между ОЗУ и ВУ заключается в пересылке массивов информации. Если проводить обмен массивов информации через МП, используя при этом регистр—аккумулятор АЛУ, то затраты времени МП будут очень большими. Поэтому в состав МПУ вводят контроллер прямого

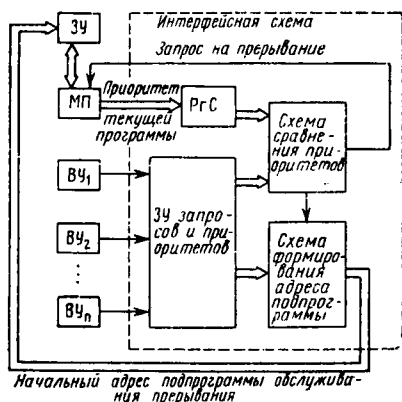


Рис. 2.9. Структура интерфейсной схемы для организации обмена информацией в режиме прерывания

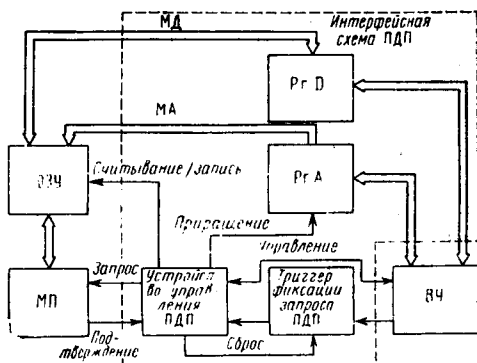


Рис. 2.10. Структура интерфейсной схемы для организации обмена информацией в режиме прямого доступа к памяти

доступа в память ПДП, который берет на себя управление передачей, освободив от этих функций основной МП. Прямой доступ в память при выполнении операций ввода—вывода значительно увеличивает скорость передачи данных и повышает эффективность использования средств МП. Построение канала ПДП является альтернативой программному обмену и иллюстрирует принцип дуализма в микропроцессорной технике: в данном случае процедура обмена с помощью дополнительной аппаратуры упрощает программное обеспечение, значительно сокращает затраты времени на выполнение функций по сравнению с их программной реализацией, но требует дополнительных аппаратных затрат с соответствующим увеличением стоимости, габаритов и потребляемой мощности системы и снижением ее надежности.

Структура интерфейсной схемы для организации обмена информацией в режиме прямого доступа к памяти (на примере ВУ, работающего в параллельном коде) приведена на рис. 2.10. В состав интерфейсной схемы входят два регистра для промежуточного хранения адресов

и данных PгА и PгD, устройство управления передачей данных, а также триггер фиксации запроса ПДП от ВУ. При обмене информацией с ВУ, работающими в последовательном коде, в состав схемы входят блоки преобразования кодов из последовательного кода в параллельный и обратно.

Средства прямого доступа к памяти подключаются параллельно процессору, причем в любой момент времени работает либо канал обмена ВУ—МП—ОЗУ, либо ВУ — канал ПДП—ОЗУ. Разделение этих каналов производится посредством использования свойств трехуровневого состояния информационных подмагистралей МА, МД и МУ. Микропроцессор во время передачи информации по каналу ПДП переводит выходные схемы управления магистралями данных, адресов и управляющих сигналов в высокоимпендансное состояние и тем самым изолируется от остальной части системы, что аналогично обрыву его информационного канала. При этом состояние внутренних регистров МП «замораживается», т. е. МП в режиме ожидания сохраняет то информационное состояние, которое возникло в нем к моменту удовлетворения запроса канала прямого доступа.

Существует много способов реализации ПДП, и все они обеспечивают более высокую скорость ввода—вывода по сравнению с методами программно-управляемого обмена. Наиболее распространенными являются методы организации ПДП с остановом МП и захватом цикла МП.

Напомним, что *машинный цикл* — это время, требуемое для извлечения одного байта информации из памяти или выполнения команды, определяемой одним машинным словом. Машинный цикл состоит из нескольких машинных тактов. *Машинный такт* — период сигналов синхронизации (интервал между двумя соседними моментами дискретного времени).

Метод останова основан на том, что в состоянии «останов» МП отключается от магистралей адресов и данных на время передачи всего массива данных. Поскольку МП должен завершить выполнение текущей команды перед переходом в состояние останова, может возникнуть задержка на несколько машинных тактов до того, как шины освободятся, т. е. до начала выполнения режима ПДП. При использовании метода останова из-за отключения МП от системных шин он не реагирует на сигналы прерывания, что может оказаться неприемлемым для некоторых режимов работы МПС.

Метод захвата цикла основан на пословном обмене данными. Если быстродействующие ВУ должны обмениваться с МП данными по одному слову, то их запрос на обслуживание может быть удовлетворен путем задержки исполнения текущей команды на один машинный цикл в то время, когда МП находится в процессе перехода от одного машинного цикла к другому. В режиме ПДП с захватом цикла работа МП приостанавливается только на один машинный цикл при передаче каждого слова данных, после чего управление опять возвращается микропроцессору.

§ 2.4. СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА

Команда микропроцессора — это предписание, определяющее шаг процесса выполнения программы работы МП. Команда содержит указание операции, адрес операндов и различные служебные признаки. Фиксированный набор команд конкретного МПК называют системой команд.

Длина команды как двоичного слова совпадает с длиной слова данных. Так, длина слова команды 8-разрядного МП равна 8 бит, а 16-разрядного МП — 16 бит. Однако команды могут иметь длину, равную не только одному, но также двум или трем словам. К примеру, длина команды 8-разрядного МП может равняться 8, 16 и 24 бит. Назначение отдельных битов команды оговаривается заранее.

Для выполнения команда посылается в регистр команд, дешифратор и схемы управления, где они идентифицируются, в результате чего формируются сигналы, направляемые в другие части микропроцессора. С помощью этих сигналов выполняются микрооперации, предписываемые командой. Микропроцессор загружает команду в регистр команд в течение цикла выборки. В течение следующего за ним цикла микропроцессор декодирует команду и создает сигналы управления процессом выполнения операций этой команды. Время, нужное для чтения команды из памяти, ее декодирования и выполнения, называется командным циклом. Ввиду того что команды могут состоять из одного, двух или трех слов и что время их выполнения может быть разным, командный цикл не является постоянной величиной.

Все команды МП имеют поля, которые определяют тип команды, ее длину, подлежащую выполнению операцию или передачу. В командах, связанных с обращением к памяти, могут быть предусмотрены поля, определяющие адрес ячейки памяти, номер регистра, номер индексного регистра и др.

В настоящем параграфе описание методов адресации и системы команд будет иллюстрироваться примерами из системы команд МП КР580ИК80А, структурная схема которого приведена на рис. 2.11. Не описывая подробно структурную схему микропроцессора, остановимся на основных деталях, представляющих интерес для программиста.

Блок регистров организован в шесть 16-битных регистров: программный счетчик РС, указатель стека SP, регистры временного хранения W, Z и три пары 8-битных регистров общего назначения В, С; D, E; H, L. Программный счетчик хранит адрес текущей ячейки программной памяти. Указатель стека адресует последнюю занятую ячейку стека. Регистры W, Z не доступны программисту. Узловой точкой всех преобразований данных и основным рабочим регистром микропроцессора является аккумулятор А. Три пары регистров общего назначения В — (В, С), D — (D, E) и H — (H, L) могут использоваться в качестве указателей памяти для организации косвенной адреса-

ции к памяти в однобайтных командах, причем эти три регистровых пары являются программно доступными.

Процесс выполнения программ зависит от особенностей (признаков) текущих результатов. Для индикации складывающейся ситуации МП формирует двоичные признаки (флажки), отражающие особенности результата выполненной команды. Значение каждого признака фиксируется в специальном триггере. Если какая-либо особенность имеет место, то в соответствующем триггере записывается 1 (флажок уста-

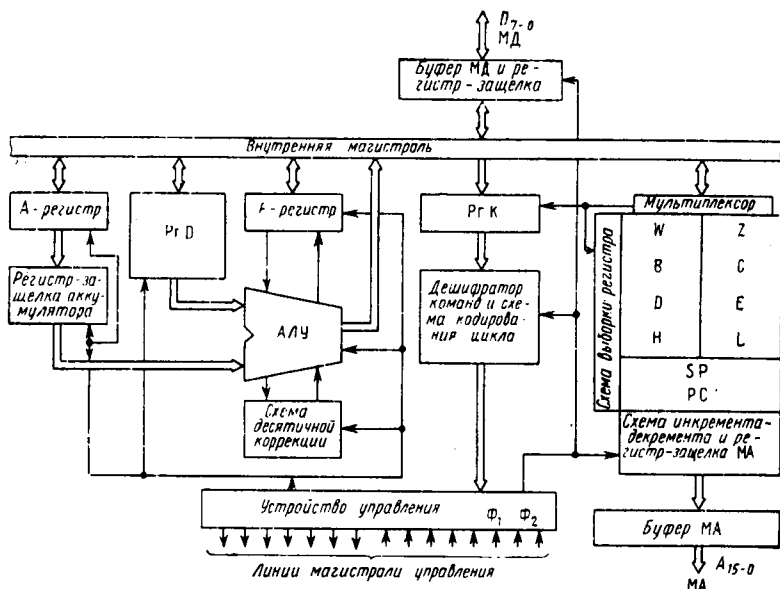


Рис. 2.11. Структурная схема микропроцессора КР580ИК80А

новлен), в противном случае триггер фиксирует 0 (флажок сброшен). Совокупность триггеров флажков образует F-регистр кода условия. Содержимое аккумулятора и F-регистра называют словом состояния процессора (ССП), для чего используют аббревиатуру PSW (processor status word — слово состояния процессора).

В МП КР580ИК80А имеются следующие флажки: С — переноса, S — знака, Z — нуля, P — паритета (устанавливается при наличии в результате четного числа единиц). AC — полупереноса (вспомогательного переноса, устанавливается при возникновении переноса из третьего бита в четвертый, т. е. из младшей тетрады в старшую).

В F-регистре флажки распределены таким образом: бит 7 — S, бит 6 — Z, бит 4 — AC, бит 2 — P, бит 0 — C. Биты 5 и 4 содержат нули, а бит 1 — единицу. Важнейшим является флажок С, позволяющий организовывать на малоразрядном микропроцессоре обработку дан-

ных любой длины путем последовательной обработки байтов операндов.

В микропроцессоре предусмотрен обширный набор команд условной передачи управления, осуществляющих выбор одного из двух направлений в зависимости от значений проверяемого флага.

Для программирования применяются mnemonicские обозначения команд — сокращенная форма записи названия команды. Для этой цели обычно используются три буквы названия операции, выполняемой командой. Мнемоника записывается, как правило, на английском языке. Например, в большинстве микропроцессоров используется код операции Clear (очистка, или сброс). Эта команда имеет mnemonicское обозначение CLA. Если микропроцессор содержит два аккумулятора (A и B), то команды их очистки могут записываться в виде CLA A и CLA B, где CLA — код операции; A и B — адреса местоположения обрабатываемых данных. Если же команда оперирует числовыми данными или адресами областей памяти, то целесообразно использование чисел в адресной части команды. Например, код операции с mnemonicским обозначением JMP (Jump — переход) требует указания адреса перехода. Подобная команда может иметь вид JMP 177756₈, где адрес выражен 6-разрядным восьмеричным числом, двоичный эквивалент которого 1111 1111 1110 1110₂ есть адрес ячейки памяти. Сочетание сокращенного буквенного обозначения кода операции с числовой формой записи адреса является одной из наиболее удобных форм записи команды. Мнемоническая запись является не только удобной формой представления кода операции, но и составной частью последней при использовании языка ассемблера. Программа ассемблера преобразует mnemonicское обозначение кодов операции в соответствующие двоичные эквиваленты.

Способы адресации микропроцессора. Исполнение большинства команд МП связано с обращением к памяти (ОЗУ, ПЗУ или ППЗУ), регистрам общего назначения, внешним устройствам через порты ввода-вывода. Организация адресации к перечисленным устройствам выполняется различными способами с помощью команд переменных форматов. Это позволяет уменьшить объем памяти, используемый для хранения программ, и повысить производительность работы МПС.

Применение различных способов организации адресов позволяет при минимальных формах команд упростить организацию вычислительного процесса. При любом методе адресации формируется исполнительный адрес, по которому производится обращение к любой ячейке адресного пространства и узлам МП. Адресное пространство — множество адресов оперативной памяти, доступных для программ, выполняемых микропроцессором данной МПС. Размер адресного пространства определяется максимальной длиной адреса и может выражаться в единицах минимальных адресуемых элементов памяти либо в более крупных единицах. Например, если исполнительный (эффективный) адрес формируется в виде 16-разрядного слова, то при минимальном адресуемом элементе памяти, равном 1 байт, максимальный размер

адресного пространства составит $2^{16} = 65536$ байт = 64 К байт = = 32 К слов (если в каждом слове помещается 2 байт). Исполнительный адрес может представлять собой также конкретное значение операнда. Рассмотрим основные методы адресации.

Наиболее естественный способ адресации — *прямая адресация*. При прямой адресации исполнительный адрес, указанный в адресной части команды (АЧ), передается в регистр адреса (рис. 2.12). Команды с прямой адресацией могут иметь длину, равную 2 или 3 байт. Первый байт предназначен для кода операции (КОП), второй и третий байты — для адреса. Прямая адресация является наиболее простой, но при этом и наименее экономичной формой адресации.

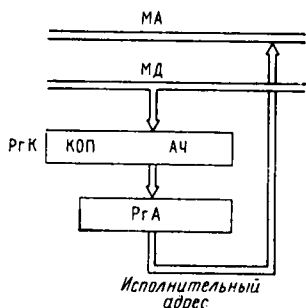


Рис. 2.12. Схема прямой адресации

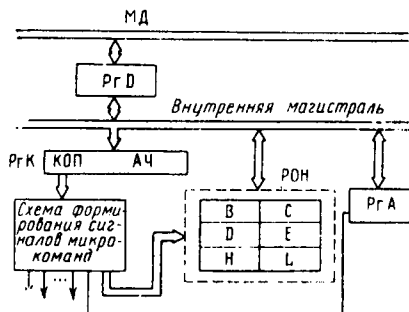


Рис. 2.13. Схема прямой регистровой адресации

Для адресации к внутренним регистрам МП используется *прямая регистровая адресация*. На рис. 2.13 показана схема выполнения однобайтной команды при прямой регистровой адресации. Сигналы микроопераций, формируемые по значению кода операции команды, определяют не только направление пересылки, но и адреса (названия) регистров. Группа команд с прямой регистровой адресацией осуществляет пересылку содержимого любого из регистров общего назначения через внутреннюю магистраль МП в регистр-аккумулятор.

Удобным средством, позволяющим компактно адресовать все пространство памяти, является *косвенная адресация*. При этом способе адресации команда указывает адрес ячейки, содержащей исполнительный адрес. Если исполнительный адрес хранится в РОН, то метод формирования исполнительного адреса именуется как *косвенная регистровая адресация*. На рис. 2.14 приведена схема, поясняющая порядок выполнения команд с косвенной регистровой адресацией. Перед началом исполнения команды с косвенной регистровой адресацией значение исполнительного адреса через МД с помощью отдельной команды заносится в одну из регистровых пар ВС, DE или HL. Исполнительный адрес из соответствующей пары РОН пересылается в регистр адреса и далее в МА.

Разновидностью косвенной регистровой адресации является *адресация к ячейкам памяти по содержимому указателю стека*. Команды этого типа производят либо запись содержимого соответствующей пары РОН в двух соседних ячейках памяти, либо считывание информации из двух соседних ячеек ОЗУ и пересылку ее в соответствующую пару РОН. На рис. 2.15 приведена схема выполнения операции обращения к ячейкам ОЗУ по содержимому SP. По команде обращения к стековой памяти в регистр адреса (PrA) пересылается адрес из указателя стека SP. При этом формируется соответствующее значение исполнительного адреса, определяющего первую ячейку стековой памяти.

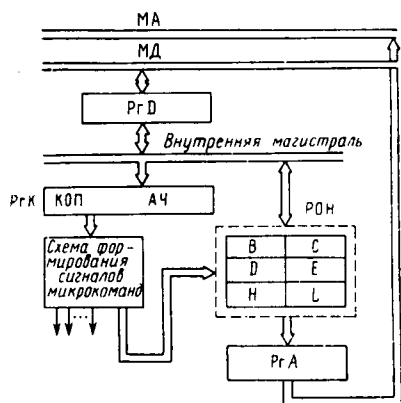


Рис. 2.14. Схема косвенной регистровой адресации

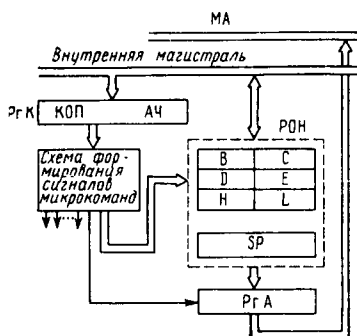


Рис. 2.15. Схема адресации с использованием указателя стека

При *непосредственной адресации* адресная часть команды представляет собой непосредственно операнд. На рис. 2.16 приведена схема выполнения команды с непосредственной адресацией. В зависимости от разрядности операнда команды с непосредственной адресацией могут быть двух- и трехбайтные. Команды с непосредственной адресацией используются для инициализации счетчиков, загрузки косвенных адресов и введения констант, необходимых при вычислениях. Непосредственная адресация осуществляется микропроцессором за два цикла: в течение первого цикла производится выборка команды, в течение второго — ее выполнение. Операции, задаваемые первым байтом команды (кодом операции), МП выполняет над данными, представленными ее вторым байтом. При использовании непосредственной адресации в команде предполагается размещение данных там же, где находится сама команда.

В однобайтных командах не отводится места ни для адреса операнда, ни для самих данных. Поэтому при выполнении однобайтных команд не происходит адресации к данным, расположенным в памяти. Однобайтные команды оперируют данными, загруженными в регистр

МП, регистрирую пару, или данными, хранимыми в области памяти, адрес которой находится в регистровой паре. Такая организация адресации называется неявной. Как правило, однобайтные команды выполняются за два цикла: за первый цикл производится выборка из памяти, за второй — выполнение операции. Однако имеются команды, выполняемые за большее число циклов.

В МПС, в которых емкость памяти превышает возможности адресной части (АЧ) команды, используется *страничная адресация*. При этом методе вся память разбивается на отдельные одинаковые страницы (участки), в пределах которых обращение производится с помощью адресной части команды. На рис. 2.17 приведена схема формиро-

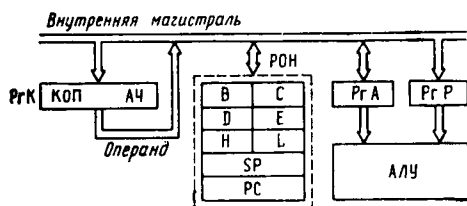


Рис. 2.16. Схема непосредственной адресации

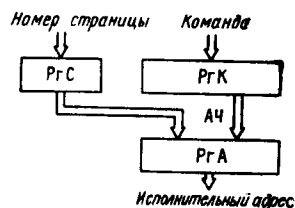


Рис. 2.17. Схема страничной адресации

вания исполнительного адреса при страничной адресации. Номер (адрес) страницы вместе с адресной частью команды составляют исполнительный адрес, по которому производится обращение к любой ячейке памяти. Адрес страницы записывается в отдельный регистр. Если программа и данные размещаются в пределах одной страницы, то в процессе исполнения программы содержимое регистра страниц остается постоянным.

В некоторых микропроцессорах исполнительный адрес может быть получен сложением исходного адреса с содержимым заданного регистра. Такой способ называется базовой адресацией, а соответствующий регистр — базовым регистром (РгБ). На рис. 2.18 приведена схема получения исполнительного адреса при базовой адресации. Базовая адресация медленнее прямой, так как для получения исполнительного адреса необходимо выполнить дополнительные операции, но в то же время, она является более гибкой, поскольку по одной и той же команде можно обрабатывать все элементы массива или таблицы. Для организации циклических вычислительных процессов содержимое базовых регистров автоматически увеличивается или уменьшается каждый раз, когда базовый регистр используется для получения исполнительного адреса. Один из вариантов модификации базового регистра приведен на рис. 2.19.

Адресация, при которой адресная часть команды, представляющая собой число со знаком, прибавляется к содержимому счетчика команд, называется *относительной адресацией*. В результате образуется ис-

полнительный адрес, по которому производится обращение к памяти. Учет знака, роль которого выполняет старший разряд, позволяет увеличить или уменьшить содержимое счетчика команд. Схема формирования исполнительного адреса при относительной адресации приведена на рис. 2.20. Данный метод удобен при организации передач управления, так как при этом не требуется задания полного адреса и перемещения программ в памяти.

Типы команд. Широкий набор команд и режимов адресации различных МП позволяет выполнять разнообразные арифметические и логические операции, операции ввода—вывода и специальные функции, а также организовывать передачу управления внутри програм-

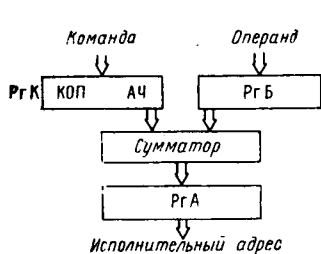


Рис. 2.18. Схема базовой адресации

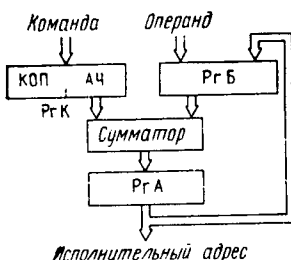


Рис. 2.19. Схема модификации базового регистра

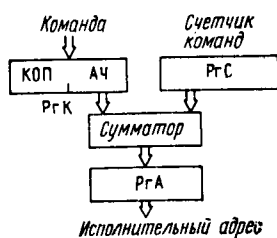


Рис. 2.20. Схема относительной адресации

мы и между программами. По функциональному назначению команды каждой МПС или микроЭВМ можно подразделить на такие группы: команды пересылки данных; команды выполнения арифметических и логических операций; команды управления; команды связи с подпрограммами; команды ввода—вывода; специальные системные команды. Для различных МПК и соответственно микроЭВМ возможна также другая функциональная классификация команд. Следует отметить, что с развитием микропроцессорной техники номенклатура системы команд постоянно расширяется. Так, если базовая система команд микропроцессора КР580ИК80А включает 78 команд, то система команд микропроцессора КМ1810ВМ86 содержит уже 135 базовых команд.

Команды пересылки данных. Эти команды служат для пересылки данных в различные устройства хранения информации, которыми располагает МПС или микроЭВМ, а также для пересылки данных из этих устройств.

Команда пересылки данных состоит из двух частей: кода операции и адресной части. Во всех командах пересылки данных должны быть указаны как источник, так и место назначения данных. Код операции в команде пересылки данных задает источник данных и способ адресации. Адресация может быть одного из следующих видов: непосредственная, прямая и косвенная. Восьмиразрядный МП может иметь одно-, двух- и трехбайтные команды пересылки данных.

Однобайтная команда пересылки из регистра S в регистр D имеет вид MOV D, S (move register — передать из регистра в регистр). По команде MOV D, S в регистр D загружается содержимое (копия данных) регистра S (выполняется операция $D \leftarrow (S)$, где (S) — содержимое регистра S). В качестве регистров D и S можно указывать любой РОН и аккумулятор. С помощью косвенной адресации команда MOV может обращаться к памяти, используя в качестве указателя адреса памяти регистры (H, L). Например, команда MOV M, C (move to memory — передать в память) заключается в пересылке данных из регистра C в память по адресу, хранимому в регистрах H и L (операция $[H, L] \leftarrow (C)$, где запись [ADDR] означает ячейку памяти с адресом ADDR). Команда MOV L, M (move from memory — передать из памяти) имеет смысл $L \leftarrow ([H, L])$. Однобайтная команда MOV реализуется за два машинных такта.

Однобайтные команды LDAX B и LDAX D (load A indirect — загрузить регистр A с косвенной адресацией) загружают в аккумулятор содержимое ячейки памяти, адресуемой соответственно парами регистров (B, C) и (D, E). Команды запоминания STAX B и STAX D (store A indirect — запомнить содержимое аккумулятора) осуществляют передачу содержимого аккумулятора в память. Команды SPHL и PCHL (jump H and L indirect — move H and L to SP — команда SPHL или to PC — команда PCHL — безусловно перейти по косвенному адресу) передают 16-разрядное содержимое регистров (H, L) в указатель стека и программный счетчик PC соответственно. Команда обмена XCHG (exchange H and L with D and E — обменять содержимое пары регистров H, L с содержимым пары регистров D, E) производит обмен данными между парами регистров (H, L) и (D, E) (операция $(D, E) \leftrightarrow (H, L)$), а команда XTHL (exchange stack top with H and L — обменять содержимое стека и регистра) осуществляет обмен между вершиной стека с адресами (SP), (SP) + 1 и парой регистров L, H, т. е. $(L) \leftrightarrow ((SP)), (H) \leftrightarrow ((SP) + 1)$.

В двухбайтных командах передач MVl r, <B2> (move immediate — передать непосредственный операнд) применяется непосредственная адресация: второй байт команды <B2> передается в адресуемый регистр: $r \leftarrow \langle B2 \rangle$. В качестве r можно указывать любой РОН или аккумулятор. Команда MVl M, <B2> (move to memory immediate — передать в память непосредственный операнд) позволяет занести байт данных в ячейку памяти, адресуемую регистрами (H, L).

В трехбайтных командах передач используется прямая и непосредственная адресация. Команда LDA (load direct — загрузить аккумулятор с прямой адресацией) загружает в аккумулятор содержимое ячейки памяти, адресуемой вторым и третьим байтами команды: $A \leftarrow ([\langle B3 \rangle \langle B2 \rangle])$, а команда запоминания STA (store A direct — запомнить содержимое аккумулятора) производит противоположную передачу: $([\langle B3 \rangle \langle B2 \rangle]) \leftarrow (A)$. Команда LHLD (load H and L direct — загрузить регистры H и L) применяется для загрузки основного указателя памяти — регистров (H, L) из двух смежных ячеек

памяти $L \leftarrow (l \langle B3 \rangle \langle B2 \rangle l)$, $H \leftarrow (l \langle B3 \rangle \langle B2 \rangle + 1)$). Команда SHLD (store H and L direct — запомнить содержимое регистров H и L) запоминает содержимое регистров (H, L) в двух смежных ячейках памяти.

Трехбайтные команды LXI rp (load register pair immediate — загрузить непосредственный 16-разрядный операнд), где rp означает регистровые пары B, D, H и указатель стека SP, применяются для загрузки начальных значений (инициализации) внутренних указателей памяти. Например, команда LXI B имеет вид $C \leftarrow \langle B2 \rangle$, $B \leftarrow \langle B3 \rangle$.

В группу команд пересылки данных включаются двухбайтные команды ввода IN (input — вход) и OUT (output — вывод). Второй байт $\langle B2 \rangle$ этих команд представляет собой 8-байтный адрес порта ввода (вывода), а получателем (источником) может быть только аккумулятор.

Команды арифметических операций. В различных типах МП состав команд арифметических и логических операций может иметь некоторые отличия. Так, например, арифметические операции умножения и деления схемно реализуются не во всех типах МП. Наиболее часто эти операции реализуются с помощью подпрограмм. В МП с микропрограммным управлением набор команд может существенно изменяться в зависимости от реализуемых задач. Вычислительные возможности МП КР580ИК80А ограничены простыми командами сложения и вычитания 8-битных операндов. Операции умножения и деления, а также операции с другими форматами данных реализуются с помощью подпрограмм, что приводит к значительному уменьшению производительности МПС.

Сложение и вычитание осуществляются однобайтными командами ADD r (add register — сложить с содержимым регистра: $A \leftarrow (A) + (r)$) и SUB r (subtract register — вычесть содержимое регистра: $A \leftarrow (A) - (r)$). Флажок C в операции вычитания устанавливается, если уменьшаемое меньше вычитаемого, т. е. он превращается во флажок заема.

Во многих применениях базовой длины слова микропроцессора недостаточно и приходится вводить многобайтные числа с последовательной обработкой отдельных байтов. В памяти такие числа хранятся в смежных ячейках и адресуются по младшему байту. При обработке многобайтных чисел очень удобны команды сложения с переносом ADC r (add register with carry — сложить с содержимым регистра с учетом переноса): $A \leftarrow (A) + (r) + C$ и вычитания с заемом SBB r (subtract register with borrow — вычесть содержимое регистра с заемом): $A \leftarrow (A) - (r) - C$, где C — величина переноса.

В системе команд МП КР580ИК80А имеются двухбайтные команды сложения и вычитания с непосредственной адресацией: адресуемым операндом служит второй байт $\langle B2 \rangle$ команды. Это команды ADI (add immediate — сложить с непосредственным операндом): $A \leftarrow (A) + \langle B2 \rangle$, SUI (subtract immediate — вычесть непосредствен-

ный операнд): $A \leftarrow (A) - \langle B2 \rangle$, AC1 (add immediate with carry — сложить с непосредственным операндом и переносом): $A \leftarrow (A) + \langle B2 \rangle + C$, SB1 (subtract immediate with borrow — вычесть непосредственный операнд с заемом): $A \leftarrow (A) - \langle B2 \rangle - C$.

Команды арифметического сравнения CMP r (compare register — сравнить с содержимым регистра): $(A) - (r)$ и CPI (compare immediate — сравнить с непосредственным операндом): $(A) - \langle B2 \rangle$ производят вычитание из содержимого аккумулятора значения адресуемого операнда, модифицируют по результату все флажки, но не изменяют содержимое аккумулятора. Эти команды используются для поиска в массиве заданного слова. Факт определения заданного слова фиксируется по флажку $Z = 1$.

Однобайтная команда десятичной коррекции аккумулятора DAA (decimal adjust accumulator — выполнить десятичную коррекцию содержимого аккумулятора) корректирует 8-битное значение в аккумуляторе на две цифры в двоично-десятичном коде.

При программировании часто возникает необходимость увеличения (инкремента) или уменьшения (декремента) значения операнда на единицу. Эти действия легко осуществляются командами AD1 и SU1, второй байт $\langle B2 \rangle$ которых равен единице. Для сокращения длин программ и времени их выполнения в системе команд предусмотрены однобайтные команды инкремента INR r (increment register — увеличить на единицу содержимое регистра r): $r \leftarrow (r) + 1$ и декремента DCR r (decrement register — уменьшить на единицу содержимое регистра r): $r \leftarrow (r) - 1$.

Для проведения операций с 16-битными операндами предусмотрен ряд команд. Команды инкремента INXrp (increment register pair — увеличить на единицу содержимое пары регистров) и декремента DCXrp (decrement register pair — уменьшить на единицу содержимое пары регистров) позволяют увеличить или уменьшить содержимое регистров (B, C), (D, E), (H, L) и указателя стека SP на единицу.

Команда двойного сложения DAD rp (add register pair to H and L — сложить содержимое регистров с двойной точностью): $H, L \leftarrow (H, L) + (rp)$ суммирует содержимое регистра (H, L) и адресуемого 16-битного регистра.

Команды логических операций и сдвигов. Логические операции являются поразрядными, т. е. выполняются независимо для всех 8 бит операндов. Неадресуемый операнд находится в аккумуляторе, туда же загружается результат операции. По результату операции модифицируются состояния всех флажков, кроме флажка переноса C.

Команда ANA r (AND register — логическое И с содержимым регистра r): $A \leftarrow (A) \wedge (r)$ является командой поразрядной конъюнкции операндов.

Команда ORA r (OR register — логическое ИЛИ с содержимым регистра r): $A \leftarrow (A) \vee (r)$ осуществляет поразрядную дизъюнкцию операндов.

Команда XRA r (exclusive OR register — исключаящее ИЛИ с содержимым регистра r): $A \leftarrow (A) \oplus (r)$ производит поразрядное сложение операндов по mod 2.

Двухбайтные команды с непосредственной адресацией ANI (AND immediate — логическое И с непосредственным операндом), ORI (OR immediate — логическое ИЛИ с непосредственным операндом) и XRI exclusive OR immediate — исключаящее ИЛИ с непосредственным операндом) производят соответствующие логические операции с

содержимым аккумулятора и вторым байтом $\langle B2 \rangle$ команды, а результат загружается в аккумулятор.

При выполнении операций сдвига в МП чаще всего содержимое регистра А сдвигается вправо или влево на один разряд. Для того чтобы произвести сдвиг на несколько разрядов, необходимо соответствующее число раз выполнить команду сдвига. Сдвиг информации может быть логическим и арифметическим. При логическом сдвиге все разряды числа сдвигаются на равных основаниях. При арифметическом сдвиге

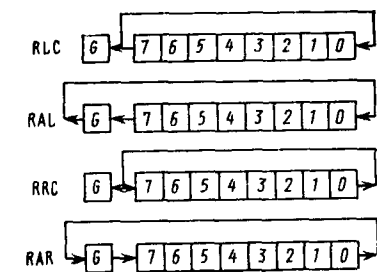


Рис. 2.21. Схема выполнения операций сдвига

знаковый разряд не сдвигается. В различных МП состав реализуемых схемно операций сдвига может быть различным. В системе команд МП КР580ИК80А имеются команды сдвига RLC (rotate left — циклически сдвинуть влево), RAL (rotate left through carry — сдвинуть влево), RRC (rotate right — циклически сдвинуть вправо) и RAR (rotate right through carry — сдвинуть вправо). Перечисленные команды выполняют сдвиг содержимого аккумулятора в соответствии со схемами, приведенными на рис. 2.21.

К логическим операциям можно отнести однобайтные команды инвертирования аккумулятора CMA (complement accumulator — инвертировать содержимое аккумулятора): $A \leftarrow (\bar{A})$, установки флажка переноса STC (set carry — установить флажок переноса): $C \leftarrow 1$ и инвертирования флажка переноса CMC (complement carry — инвертировать перенос): $C \leftarrow \bar{C}$.

Команды передачи управления. Для организации разветвляющихся вычислительных процессов служат команды передачи управления, с помощью которых выполняется анализ условий разветвления и производится запись в программный счетчик РС адреса следующей команды. Команды передачи управления образуют две группы. Первую группу составляют команды безусловной передачи управления, обеспечивающие запись информации в счетчик РС. Вторую группу составляют команды условий передачи управления. С помощью этих команд также выполняется запись информации в счетчик РС, если вы-

полнены определенные условия. Если условия не выполнены, то занесение информации в счетчик РС не производится, а его содержимое увеличивается на единицу. Условия, по которым организуется разветвление вычислительного процесса, фиксируются в F-регистре признаков. Рассмотрим ряд команд передачи управления из системы команд МП КР580ИК80А.

Команда безусловной передачи управления JMP (jump — перейти безусловно) состоит из байта кода операции и двух байтов полного 16-битного адреса перехода. При ее выполнении адрес перехода загружается в счетчик РС, а текущее содержимое счетчика РС теряется: $PC \leftarrow \langle B3 \rangle \langle B2 \rangle$.

Команды условного перехода осуществляют разветвление вычислительного процесса по текущему значению соответствующего флажка. Например, команда JZ (jump on zero — переход по нулевому результату) осуществляет загрузку в счетчик РС: $PC \leftarrow \langle B3 \rangle \langle B2 \rangle$, если флажок **Z** = 1, а команда JNZ (jump on not zero — переход по ненулевому результату), если **Z** = 0. Аналогичные парные команды имеются для флажка **C**: JC (jump on carry — переход по переполнению) и JNC (jump on not carry — переход по отсутствию переполнения), для флажка **S**: JM (jump on minus — переход по отрицательному результату) и JP (jump on positive — переход по положительному результату), для флажка **P**: JPE (jump on parity even — переход по четному результату) и JPO (jump on parity odd — переход по нечетному результату).

Трехбайтная команда безусловной передачи управления с возвратом CALL (call — вызвать) состоит из байта кода операции и адреса перехода, представляющего собой начальный адрес подпрограммы. При выполнении вызова необходимо временно запомнить текущее содержимое счетчика РС (т. е. адрес команды, следующей за командой CALL, называемый адресом возврата), загрузить адрес перехода команды CALL в счетчик РС, а после выполнения подпрограммы передать адрес возврата в счетчик РС. Возвращение к основной программе выполняется с помощью команды RET (return — вернуться). Таким образом, вызов одной подпрограммы и возврат из нее осуществляются с помощью 16-битного регистра для временного запоминания адреса возврата.

Для организации вложенных подпрограмм используется стековая память. По команде PUSH гр (push — поместить в стек) в ячейку памяти с адресом (SP) — 1 записывается содержимое восьми старших разрядов регистровой пары гр, а в ячейку с адресом (SP) — 2 — содержимое восьми младших разрядов этой регистровой пары. При этом содержимое регистра SP уменьшается на два. При выполнении команды POP гр (pop — вытолкнуть из стека) содержимое верхушки стека передается в младшие 8 бит регистровой пары гр, а содержимое ячейки стека, адрес которой на единицу больше адреса верхушки (записанного в регистр SP), — в старшие 8 бит этой регистровой пары. После этого содержимое регистра SP увеличивается на два.

В командах условных вызовов и возвратов проверяется состояние указанного в коде операции флажка, и если оно удовлетворяет условию, осуществляются вызов и возврат, а если не удовлетворяет, выполняется следующая по порядку команда. Имеются команды, проверяющие единичное и нулевое значения каждого из четырех флажков, например для флажка **Z**: команды вызовов **CZ** (call on zero) и **CNZ** (call on no zero); команды возвратов **RZ** (return on zero) и **RNZ** (return on no zero).

В системе команд МП имеется однобайтная команда вызова, предназначенная для обработки прерываний и введения контрольных точек при отладке программ. Она называется рестартом (повторным запуском) и имеет мнемоническое обозначение **RST** (restart — повторный пуск).

При отработке прерываний возникает необходимость запоминания текущего слова состояния процессора **PSW** — содержимого аккумулятора и содержимого **F**-регистра флажков. В системе команд МП имеется команда **PUSH PSW** (push processor status word — поместить в стек содержимое регистра признаков) загрузки в стек **PSW**, а также соответствующая команда **POP PSW** (pop processor status word — вытолкнуть из стека содержимое регистра признаков) извлечения **PSW** из стека и загрузки его в **F**-регистр флажков и аккумулятор. Выполнение этих команд сопровождается автоматической модификацией регистра **SP**.

Команды управления микропроцессором. Однобайтные команды этой группы команд применяются для задания режима работы МП.

Команда останова **HLT** (halt — стоп) вызывает прекращение выполнения программы. Команда разрешения прерываний **EI** (enable interrupts — прерывание разрешено) устанавливает триггер **INTE** разрешения прерываний в единичное состояние, при этом МП реагирует на запросы прерываний. Команда запрещения прерываний **DI** (disable interrupt — прерывание запрещено) устанавливает триггер **INTE** в нулевое состояние. Команда **NOP** (no operation — пустая команда) используется в программных циклах задержки, в которых МП генерирует сигналы программируемой длительности.

§ 2.5. АРХИТЕКТУРА ОДНОКРИСТАЛЬНОГО МИКРОПРОЦЕССОРА КМ1810ВМ86

Однокристалльный высокопроизводительный 16-разрядный микропроцессор **КМ1810ВМ86** (далее используется сокращенное обозначение **К1810**) предназначен для использования в качестве центрального процессорного устройства в различных электронно-вычислительных и радиоэлектронных средствах — от встроенных МПУ и одноплатных микроЭВМ до высокопроизводительных МПС.

Микропроцессор обладает высоким быстродействием (до $2,5 \times 10^6$ оп/с типа «регистр—регистр» при тактовой частоте 5 МГц) и обеспечивает возможность прямой адресации памяти емкостью до

1 М байт. Микропроцессор архитектурно и программно совместим с комплектом БИС КР580, обладает возможностью эффективной работы с языками высокого уровня, имеет гибкую и мощную систему команд. Для решения сложных задач в реальном времени микропроцессор имеет векторную структуру прерывания и обеспечивает до 256 запросов прерывания трех типов: внешних, внутренних и программных. Для вычисления адресов операндов, размещенных в памяти, используются 24 режима адресации.

Архитектурной особенностью микропроцессора К1810 является наличие аппаратно-программных средств, позволяющих упростить построение МПС на его основе. Эти средства обеспечивают синхронизацию работы нескольких независимых процессоров, имеющих общие ресурсы, а также синхронизацию параллельной работы микропроцессора и сопроцессоров специализированных процессоров, аппаратно реализующих команды сложных процедур. Микропроцессор К1810 представляет собой двухадресную ЭВМ, обеспечивающую следующие комбинации источника — приемника операндов: регистр—регистр, регистр—память, память — регистр, непосредственный операнд—регистр и непосредственный операнд—память.

Структурная организация.

Структурная схема микропроцессора К1810 ориентирована на параллельное выполнение функции выборки и исполнения команд и состоит из операционного устройства

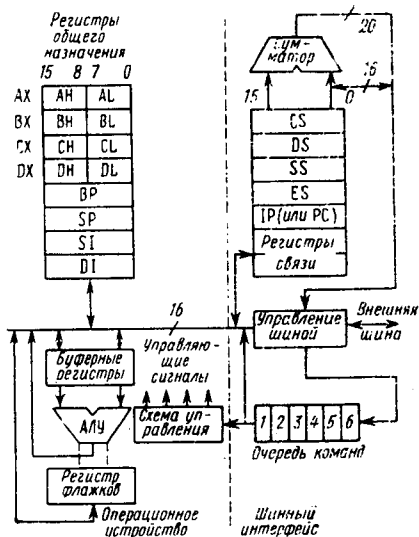


Рис. 2.22. Структурная схема МП КМ1810ВМ86

и шинного интерфейса (рис. 2.22).

Шинный интерфейс обеспечивает формирование 20-разрядного физического адреса памяти, выборку команд и операндов из памяти, формирование очереди команд, запоминания результатов выполнения команд в памяти. В соответствии с этим в состав шинного интерфейса входят: шесть 8-разрядных регистров очереди команд; четыре 16-разрядных сегментных регистра; 16-разрядный регистр адреса команды; 16-разрядный регистр обмена и 16-разрядный сумматор адреса.

Шинный интерфейс выполняет для операционного устройства все операции обмена. Данные передаются между МП и памятью или портами ввода—вывода по запросам операционного устройства. Когда операционное устройство занято выполнением команды, шинный интерфейс самостоятельно инициирует опережающую выборку из памяти очередных команд. Команды хранятся во внутренней регист-

ровой памяти, состоящей из шести 8-разрядных регистров и называемой очередью (буфером) команд. Очередь команд выполняет по существу функции регистра команды процессора. Очередь команд работает по принципу FIFO (first in — first out — первым пришел, первым ушел), который сохраняет на выходе порядок поступления команд. Шинный интерфейс инициирует выборку из памяти следующего командного слова, когда в очереди команд оказываются два свободных байта.

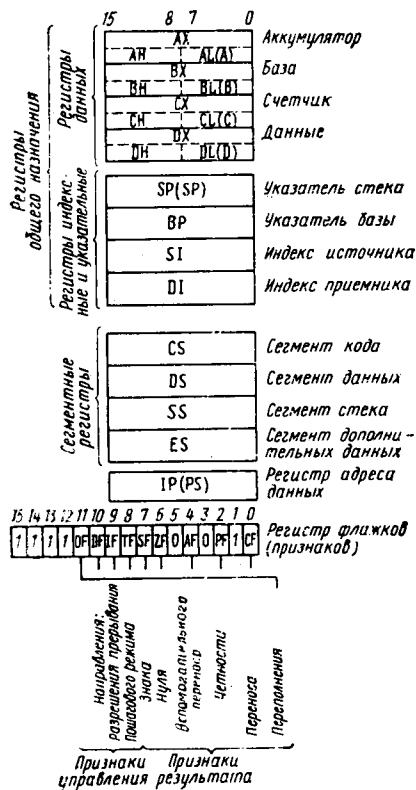


Рис. 2.23. Программно-доступные регистры МП KM1810BM86

Операционное устройство предназначено для выполнения операций по обработке данных. Команда, выбранная из памяти и записанная в регистры очереди команд шинного интерфейса по запросам от операционного устройства, поступает через 8-разрядную магистраль команд на микропрограммное устройство управления, которое декодирует команды и вырабатывает соответствующую последовательность микрокоманд, управляющую процессом выполнения текущей операции. Операционное устройство не имеет непосредственной связи с внешней магистралью системы и обменивается данными через регистры связи с шинным интерфейсом. В состав операционного устройства входят: арифметико-логическое устройство (АЛУ), основу которого составляет комбинационный 16-разрядный сумматор с последовательно-параллельным переносом, восемь 16-разрядных регистров общего назначения и регистр признаков состояния МП (регистр флажков).

Для программиста программно-доступными функциональными частями

МП являются регистры: общего назначения (используемые для хранения операндов и результатов выполнения команд), сегментные (используемые для хранения базовых адресов текущих сегментов памяти), адреса команд (IP) и признаков. На рис. 2.23 в скобках указаны соответствующие регистры МП KP580IK80A.

В состав регистров общего назначения входят регистры данных AX, BX, CX и DX. Отличительной особенностью этих регистров является то, что допускается разделять их старшую (H —

high) и младшую (L — low) половины, т. е. каждый из них можно использовать как 16-битный регистр или два 8-битных регистра. Это обеспечивает простую обработку 8- и 16-битных данных (байтов и слов). Остальные регистры МП можно использовать только как 16-битные.

Группа индексных и указательных регистров представлена адресными регистрами SP, BP, SI и DI. Они предназначены для хранения 16-битных адресов (внутрисегментных смещений) и обеспечивают при этом косвенную адресацию и динамическое вычисление эффективного адреса памяти. Гибкость вычислений достигается тем, что эти регистры могут участвовать в арифметических и логических операциях, как и регистры данных. Поэтому регистры данных, индексные и указательные регистры объединены вместе в качестве регистров общего назначения.

Указательные регистры SP и BP предназначены для упрощения доступа к данным, находящимся в текущем сегменте стека, а не в сегменте данных. Индексные регистры SI и DI содержат смещения, которые относятся к текущему сегменту данных.

Наличие в программной модели четырех сегментных регистров объясняется способом адресации памяти. Хотя МП имеет 20-битную шину физического адреса памяти, он оперирует 16-битными логическими адресами, состоящими из базового адреса сегмента и внутрисегментного смещения. Внутреннее устройство преобразования адресов превращает два логических адреса в 20-битный физический адрес.

Пространство памяти емкостью 1 М байт разделено на логические сегменты емкостью 64 К байт. Выполняя программу, МП может одновременно обращаться к четырем сегментам. Их базовые (начальные) адреса содержатся в сегментных регистрах CS (кода), DS (данных), SS (стека) и ES (дополнительных данных).

Регистр CS указывает на текущий сегмент кода (программы), откуда выбираются команды. DS указывает на текущий сегмент данных, в котором содержатся переменные. Регистр SS адресует текущий сегмент стека, в этом сегменте реализуются все стековые операции. Наконец, регистр ES определяет текущий дополнительный сегмент, который обычно используется для хранения данных. Сегментирование памяти совместно с позиционно-независимыми командами передачи управления позволяет создавать динамически перемещаемые программные модули.

Последнюю группу регистров МП образуют 16-битные указательные команды IP (аналог стандартного программного счетчика PC) и регистр флажков (признаков). Шинный интерфейс осуществляет модификацию регистра IP таким образом, что он содержит смещение следующей команды от начала текущего сегмента кода, т. е. указывает на следующую по порядку команду.

Регистр флажков (признаков) предназначен для фиксации различных признаков протекающего вычислительного процесса. Младший байт 16-разрядного регистра признаков полностью соответст-

вует F-регистру МП КР580, а старший байт содержит четыре флажка, которых нет в МП КР580.

Шесть арифметических флажков фиксируют определенные признаки результата арифметической или логической операции, а группа команд условных переходов позволяет изменить ход программы в зависимости от состояний флажков. Флажки отражают следующие условия:

1) флажок вспомогательного переноса **AF**, фиксирует перенос (заем) из младшей тетрады в старшую при реализации команд десятичной арифметики;

2) флажок переноса **CF** фиксирует значение переноса (заема), возникающего при сложении или вычитании;

3) флажок переполнения **OF** сигнализирует о потере старшего бита результата сложения или вычитания;

4) флажок знака **SF** повторяет значение старшего бита результата, который при использовании дополнительного кода соответствует знаку числа;

5) флажок паритета **PF** (или четности) фиксирует наличие четного числа единиц в младших 8 бит результата операции;

6) флажок нуля **ZF** сигнализирует о получении нулевого результата операции;

7) флажок направления **DF** определяет порядок сканирования цепочек в соответствующих командах: от меньших адресов к большим (**DF** = 0) или наоборот (**DF** = 1);

8) флажок прерывания **IF** определяет реакцию МП на запросы внешних прерываний по входу **INT** (запрос прерывания), при этом если **IF** = 1, то МП реагирует на запрос прерывания;

9) установка в состояние «1» флажка пошагового режима **TF** переводит МП в покомандный режим работы, применяемый при отладке программ.

Форматы команд и данных. На форматы команд и данных влияет выбор режима адресации. Назначением режима адресации является указание способа формирования эффективного (или исполнительного) адреса **EA**. Этот адрес является либо адресом операнда (в командах, оперирующих данными), либо адресом перехода (в командах передачи управления). Команды МП К1810 занимают в памяти 1...6 байт, а эффективный адрес памяти представляет собой 16-битное беззнаковое целое, являющееся смещением относительно базы некоторого сегмента. Полный (физический) адрес памяти формируется с привлечением одного из сегментных регистров.

Режимы адресации подразделяются на прямые и косвенные. При прямой адресации эффективный адрес либо содержится в команде, либо вычисляется с использованием значения, находящегося в команде, и содержимого, указанного в команде регистра (или двух регистров). При косвенной адресации эффективный адрес в команде определяет регистр или ячейку памяти, содержащую окончательный эффективный адрес.

Микропроцессор K1810 имеет организацию типа «регистр—память». С точки зрения адресации это означает, что его команды адресуют максимум два операнда и что не допускается одновременная адресация двух ячеек памяти. Первым операндом в двухоперандной (двухадресной) команде обычно является содержимое регистра или ячейки памяти, а вторым — содержимое регистра или непосредственный операнд в команде.

Наиболее общий формат двухоперандной команды приведен на рис. 2.24 (штриховыми линиями обозначены необязательные байты команды). Первый байт команды содержит код операции КОП и два однобитных поля—направления *d* (direction — направление) и слова *w* (word — слово). Поле *d* определяет направление передачи: если *d* =

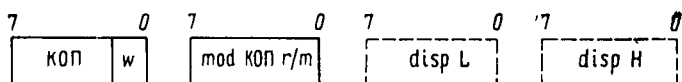


Рис. 2.24. Двухоперандная команда (второй операнд — регистр)

1, то направление в МП, если *d* = 0, то направление из МП. Само направление относится ко второму операнду—регистру, определяемому полем *reg* (register — регистр) второго байта команды. Этот байт называется постбайтом режима адресации. Поле *w* идентифицирует размер операнда: если *w* = 1, то команда оперирует словом, если *w* = 0, то — байтом.

Участвующие в операции регистры или регистр и ячейку памяти идентифицирует постбайт, состоящий из трех полей: *mod* — режим (mode — режим), *reg* — регистр и *r/m* — регистр/память (memory — память). Поле *reg* состоит из 3 бит и определяет второй операнд, обязательно находящийся в регистре. Поле *mod* состоит из 2 бит и определяет используемый режим адресации. Во втором и третьем байтах команды находится смещение *disp* (displacement — смещение). Поле *r/m* — трехбитное.

Описанная организация двухоперандной команды позволяет адресовать операнд в памяти прямо (16-битное смещение) или косвенно (с 8- или 16-битным смещением). Во втором случае память можно адресовать через базовый регистр (BP или BX), индексный регистр (SI или DI), а также через комбинацию базового и индексного регистров. Всего получается 24 режима адресации памяти — три способа интерпретации поля *mod* и восемь способов интерпретации поля *r/m*. Процедура вычисления адреса памяти представлена на рис. 2.25.

В двухоперандной команде с непосредственным операндом, формат которой приведен на рис. 2.26, второй операнд адресовать не нужно, поэтому поле *reg* постбайта режима адресации используется как расширение кода операции. Кроме того, здесь не нужен бит *d*, так как

результат можно поместить только на место первого операнда. Для определения размера непосредственного операнда служат биты *s* и *w*. На рис. 2.27 показан типичный формат однооперандной команды.

Помимо типичных форматов в МП К1810 реализована группа команд с упрощенным форматом, которые осуществляют наиболее часто применяемые операции. По быстродействию и занимаемой памяти они наиболее эффективны. Так, команды упрощенного формата для работы с РОН являются однобайтными, а поле *reg* помещено в байт с кодом операции. При работе с данными в памяти используется регистр АХ,

а данные в сегменте данных адресуются вторым и третьим байтами команды.

Микропроцессор К1810 обеспечивает аппаратную обработку элементарных и индексированных данных, стека, строки и таблицы. К элементарным данным относятся одно- и двухбайтные порядковые и целые числа, однобайтные упакованные, одно- и двухбайтные распакованные двоично-десятичные числа, а также четырехбайтные логические адреса.

Целые числа представляются в дополнительном коде. Реализация в микро-

процессоре признака переполнения **OF** практически позволяет использовать модифицированный дополнительный код. В связи с этим диапазон изменения целых однобайтных чисел от -255 до $+255$, двухбайтных от -65535 до $+65535$. Диапазон изменения однобайтных порядковых чисел $0...255$, двухбайтных $0...65535$.

Упакованные двоично-десятичные числа содержат две десятичные цифры в байте, распакованные — одну. Над порядковыми, целыми и распакованными двоично-десятичными числами выполняются операции сложения, вычитания, умножения и деления, над упакованными двоично-десятичными числами — операции сложения и вычитания.

В МП К1810 под стек отводится специальный сегмент, адресуемый сегментным регистром **SS**. Указатель стека, определяющий относительный адрес верхушки стека в сегменте, располагается в регистре **SP**. При записи в стек вначале содержится регистр **SP** уменьшается на два, а затем по адресу в **SP** записывается слово данных. При чтении по адресу, находящемуся в **SP**, сначала читается слово данных, а затем значение **SP** увеличивается на два. Имеются команды обмена данными между стеком и регистрами микропроцессора, включая сегментные ре-

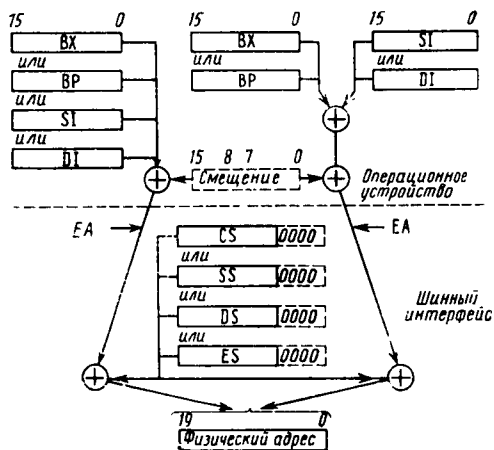


Рис. 2.25. Вычисление адреса памяти

гистры, а также между стеком и памятью. Кроме хранения адресов возврата из подпрограмм (что обеспечивается аппаратно), стек наиболее часто служит для передачи фактических параметров.

Отметим, что стандартные режимы адресации МП К1810 обеспечивают однокомпонентную прямую и косвенную адресацию элементарных данных, многовариантную двухкомпонентную адресацию, удобную, например, для доступа к элементам массивов и простых структур, и трехкомпонентную адресацию, удобную для доступа к элемен-

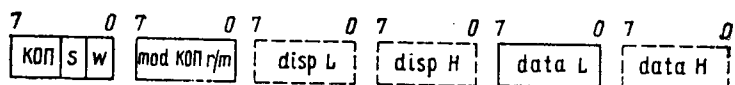


Рис. 2.26. Двухоперандная команда (второй операнд — константа)

там сложных структур данных, используемых в ячейках высокого уровня. Время вычисления эффективного адреса ЕА зависит от режима адресации. Оно равно числу тактов синхронизации, необходимому для определения эффективного адреса при условии, что команда находится во внутренней очереди команд. Зависимость времени вычисления эффективного адреса от используемого режима адресации приведена в табл. 2.1.

Поясним содержание табл. 2.1. *Прямая адресация* представляет собой простейший режим адресации — эффективный адрес ЕА берет-

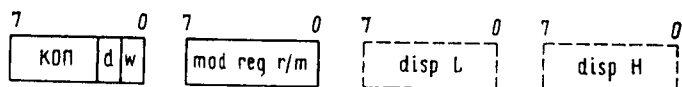


Рис. 2.27. Однооперандная команда

ся прямо из поля смещения команды и никакие регистры для его вычисления не применяются. В режиме *косвенной регистровой адресации* эффективный адрес ЕА операнда находится в одном из базовых или индексных регистров. При *базовой адресации* эффективный адрес операнда является суммой смещения и содержимого регистров ВХ или ВР, при *индексной адресации* — суммой смещения, находящегося в команде, и содержимого регистров SI или DI. При *базовой индексной адресации* эффективный адрес равен сумме содержимого базового регистра, индексного регистра и смещения, находящегося в команде. Этот режим адресации является наиболее гибким, так как два компонента адреса можно определять и варьировать при выполнении программы.

Отметим, что структурными единицами памяти являются: ячейка, слово, двойное слово и сегмент.

Таблица 2.1

Время вычисления эффективного адреса

Адресация	Формирование адреса	Число тактов
Прямая	disp	6
Косвенная регистровая	[BX], [BP], [SI], [DI]	5
Базовая или индексная	[BX, BP, SI, DI] + disp	9
Базовая индексная (без смещения)	[BP + DI], [BX + SI]	7
	[BP + SI], [BX + DI]	8
Базовая индексная (со смещением)	[BP + DI] + disp [BX + SI] + disp	11
	[BP + SI] + disp [BX + DI] + disp	12

Ячейка памяти — минимальная адресуемая единица памяти, используемая для запоминания 8-разрядных данных (байта данных).

Слово памяти — две последовательные ячейки памяти, которые используются для запоминания 16-разрядных данных, причем младшие восемь разрядов всегда хранятся в ячейке памяти с меньшим адресом, а старшие — с большим.

Двойное слово памяти — четыре последовательные ячейки памяти или два последовательных слова памяти, которые используются для запоминания 32-разрядных данных.

Сегмент памяти — участок памяти, емкость которого может изменяться от 16 до 65536 байт. Каждому сегменту соответствует непрерывная и отдельно адресуемая область памяти.

Микропроцессор К1810 позволяет независимо адресовать четыре программных сегмента в памяти, называемых текущими сегментами команд, данных, стека и текущим дополнительным сегментом. Сегментирование памяти позволяет создавать динамически перемещаемые программные модули.

Система команд. Система команд микропроцессора К1810 содержит 135 базовых команд, которые могут быть разделены на шесть категорий: команды пересылки данных, арифметические команды, команды

поразрядной обработки данных, команды обработки элементов строк данных, команды передачи управления, команды управления микропроцессором.

Команды пересылки данных предназначены для пересылки содержимого операнда-источника на место операнда-приемника. Существует четыре группы команд пересылки данных: общего назначения, ввода—вывода, логического адреса, признаков.

Арифметические команды предназначены для выполнения основных арифметических операций (сложения, вычитания, умножения и деления) над порядковыми и целыми двоичными числами, упакованными и распакованными двоично-десятичными числами, а также для преобразования форматов данных.

Команды поразрядной обработки данных предназначены для выполнения логических операций, а также операций линейного и циклического сдвигов (арифметических и логических) на один или n разрядов.

Команды обработки элементов строк данных предназначены для пересылки, сравнения, записи в память, загрузки в аккумулятор элементов строк данных. Команды этой группы позволяют пересылать и сравнивать строки (цепочки), производить поиск (или пропуск) элемента в строке и заполнять строки конкретным символом (например, пробелом). Команды обработки строк можно использовать и для обработки массивов данных различных типов (целых, порядковых и т. п.).

Команды передачи управления предназначены для организации перехода в программе. Существует четыре класса таких команд: безусловная передача управления, условная передача управления, управление циклами, команды прерываний.

Команды управления микропроцессором позволяют программно управлять его различными функциями и делятся на две группы: команды управления состоянием признаков, команды синхронизации работы микропроцессора с внешними событиями.

Команды микропроцессора обеспечивают выполнение операций над одним или двумя операндами, и результат операции может записываться по адресу любого из операндов. В зависимости от типа команды операнды могут быть расположены в программно-доступных регистрах, непосредственно в ходе команды, в памяти и регистрах ввода—вывода. Непосредственные данные могут быть типа байта или слова, операнды в программно-доступных регистрах — типа байта или слова, а для команд умножения и деления — типа двойного слова. Операнды в памяти могут быть типа байта, слова, двойного слова, а в регистрах ввода—вывода — типа байта и слова.

Особенности применения. Микропроцессорные системы на основе микросхем из МПК БИС КМ1810 реализуются по модульному принципу. В общем случае МПС состоит из центрального процессора КМ1810ВМ86 с необходимым комплектом микросхем серий КМ1810 и КР580, микросхем памяти (ПЗУ, ППЗУ, ОЗУ) и устройств ввода—вывода.

Рассмотрим основные мультипроцессорные конфигурации, организованные на базе МПК БИС КМ1810 (или его американского аналога Intel 8086).

Сопроцессорные конфигурации. Несмотря на развитую систему команд микропроцессора К1810, ее возможностей оказывается недостаточно для эффективной реализации некоторых сложных применений. В этих случаях в состав МПС необходимо включать сопроцессоры, которые расширяют систему команд в направлениях, обеспечивающих более эффективную реализацию требуемых специальных вычислений. Это часто необходимо при реализации алгоритмов обработки радиотехнической информации, в которых требуемая высокая точность вычислений наталкивается на трудности, связанные с недостаточной разрядностью представляемых данных.

Для увеличения разрядности обрабатываемых данных и выполнения арифметических операций с плавающей точкой в рамках микропроцессорного комплекта серии К1810 разработан вспомогательный процессор (сопроцессор) К1810ВМ87. Сопроцессор К1810ВМ87 позволяет работать с тремя типами данных, целыми числами, представленными двоичной или двоично-десятичной системой счисления, и вещественными числами.

Двоичные целые числа могут быть представлены в одном из трех форматов: слово (16-разрядное число со знаком), короткое целое (32-разрядное число со знаком) и длинное целое (64-разрядное число со знаком).

Для записи двоично-десятичных чисел отводится 10 байт, причем число представляется в виде последовательности 18 4-разрядных кодов цифр числа (72 разряда), а в старшем байте числа содержится знак числа.

Вещественные числа могут быть представлены в различных форматах, причем максимальный формат занимает 80 двоичных разрядов и состоит из 64-разрядной мантиссы, 15-разрядного порядка и знака мантиссы, что позволяет представлять числа в диапазоне значений от $\pm 1,2 \cdot 10^{4932}$ до $\pm 3,4 \cdot 10^{-4931}$.

Система команд сопроцессора включает в себя не только разнообразные формы сложения, вычитания, умножения и деления, но и специальные функции типа извлечения квадратного корня, возведения в степень, нахождения тангенса и др.

В МПС с сопроцессором центральный процессор и сопроцессор работают по одной программе. При этом, если команда выполняется сопроцессором, то центральный процессор кроме возможного считывания операнда для сопроцессора не предпринимает более никаких действий.

Сильно связанные конфигурации. В таких конфигурациях центральный процессор и параллельно работающий процессор реализуют свои потоки команд. В сильно связанных МПС центральный процессор активизирует независимый процессор, посылая приказ в один из его портов. Независимый процессор обращается к памяти, получает от

туда предназначенную ему задачу и выполняет ее параллельно с центральным процессором. После завершения задачи независимый процессор извещает об этом центральный процессор и ждет команды на выполнение следующей задачи. Примером независимого процессора может служить процессор ввода—вывода K1810BM89 (Intel 8089).

Слабо связанные конфигурации. В МПС невозможно непосредственно подключить два микропроцессора K1810. Для организации параллельной работы двух или нескольких центральных процессоров на базе МП K1810 каждый центральный процессор должен иметь свою логику управления магистралью, а арбитраж магистрали достигаться путем расширения этой логики и введения общей для всех процессоров внешней логики. При такой организации несколько центральных процессоров могут образовывать очень большую систему, а к каждому процессору можно подключить независимые процессоры и (или) сопроцессор. Подобная конфигурация носит название слабо связанной. Она обладает такими достоинствами:

- при наличии нескольких центральных процессоров повышается пропускная способность системы;

- система допускает модульное расширение; каждый модуль является независимым устройством и обычно размещается на отдельной печатной плате;

- отказ в одном модуле обычно не вызывает простоя всей системы;

- каждый центральный процессор может иметь собственную память и устройства ввода—вывода, а также средства доступа к ним.

Для организации синхронизации доступа множества центральных процессоров к системной шине в слабо связанных МПС используется арбитр системной шины KP1810BB89 (Intel 8289).

Сети микроЭВМ. Рассмотренные конфигурации имеют общее свойство — все процессоры разделяют одну и ту же системную шину. Следовательно, микропроцессорные взаимодействия осуществляются через общую память, а процессоры физически должны быть расположены близко друг к другу.

При использовании последовательных линий связи множество микроЭВМ может взаимодействовать друг с другом и разделять некоторые аппаратные и программные ресурсы. Такие системы называют сетями микроЭВМ. Каждый элемент сети (микроЭВМ или мультипроцессорный элемент) относительно независим от других, а расстояния между ними могут быть практически любыми.

Новые сверхбольшие интегральные схемы. Разработка новых типов интегральных схем возможна в рамках следующих направлений:

1. Увеличение числа рабочих регистров и (или) их разрядности. При этом существенно увеличивается объем логики в микросхеме, что в конечном итоге приводит к появлению нового семейства процессоров и вспомогательных микросхем.

В соответствии с этим направлением разработаны однокристалльные 32-битные микропроцессоры, примерами которых могут служить процессоры Intel 80386 и Intel 80486.

2. Улучшение вычислительной возможности процессора. Для этого обычно разрабатывается быстродействующий числовой сопроцессор, совместимый с базовым центральным процессором. Такой процессор, выполняющий операции с плавающей точкой и позволяющий достигать высокой точности вычислений, значительно повышает производительность МПС. Примером является процессор Intel 8087 (аналог — МП К1810ВМ87).

3. Усовершенствование средств доступа к быстродействующей внешней памяти. Примером такого устройства, способного выполнять операции ввода—вывода и передачи ПДП, является процессор Intel 8089 (аналог — МП К1810ВМ89).

4. Размещение специализированных и системных программных средств в ПЗУ, находящемся на кристалле МП. Подобную микросхему совместно с центральным процессором называют процессором операционной системы. В качестве примера может служить микросхема Intel 80130, разработанная для поддержки операционной системы. Главным компонентом в этой микросхеме является ПЗУ емкостью 16 К байт, в котором находится код, необходимый для различных системных вызовов операционной системы.

5. Размещение на кристалле МП таких компонентов, как интерфейсы ввода—вывода, таймеры, контроллеры ПДП, блоки управления прерываниями, секции памяти и др. Это сокращает число корпусов и контактов в МПС. Примером является микросхема Intel 80186.

6. Размещение на кристалле МП логики управления памятью и ее защиты. Примером может служить микросхема Intel 80286. Отметим, что при одинаковой конструкции корпусов микросхем 8086 и 80286 производительность процессора 80286 в мультипрограммной среде в 6 раз выше производительности МП 8086. Фирмой Intel разработан также процессор числовых данных 80287, совместимый с процессором 80286.

§ 2.6. ОРГАНИЗАЦИЯ УПРАВЛЕНИЯ МИКРОПРОЦЕССОРАМИ

Особенности программного и микропрограммного управления. В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов: программный и микропрограммный. Программный метод управления реализован в однокристалльных МП (с фиксированной разрядностью слова и с фиксированной системой команд). Микропрограммный метод управления характерен для секционированных многокристалльных микропроцессоров с дискретно наращиваемой разрядностью слова.

В однокристалльных МП реализованы аппаратурные (схемные) устройства управления, в которых каждой операции соответствует свой набор логических схем, вырабатывающих определенные функциональные сигналы для выполнения микроопераций в определенные моменты времени. При этом способе построения устройства управления реализация микроопераций достигается за счет однажды соединенных между

обой логических схем, поэтому такой способ организации управления называют управлением с *жесткой логикой*. Это понятие означает невозможность каких-либо изменений в системе команд МП.

В секционных МП реализован микропрограммный способ организации устройства управления, в его состав вводится ЗУ, каждый разряд выходного кода которого определяет появление определенного функционального сигнала управления. Поэтому каждой микрооперации ставится в соответствие свой информационный код — *микрокоманда*. Набор микрокоманд и последовательность их реализации обеспечивают выполнение любой сложной операции. Упорядоченный набор микроопераций называют *микропрограммами*. Способ управления операциями путем последовательного считывания и интерпретации микрокоманд из ЗУ, а также использование кодов микрокоманд для генерации функциональных управляющих сигналов называют микропрограммным. К микропрограммам предъявляют требования функциональной полноты и минимальности. Первое требование необходимо для обеспечения возможности разработки микропрограмм любых машинных операций, второе — связано с желанием уменьшить объем используемого оборудования. Учет фактора быстродействия ведет к расширению микропрограмм, поскольку усложнение последних позволяет сократить время выполнения команд программы. В микропрограммируемых МП управляющий автомат может быть конструктивно реализован либо в виде отдельной БИС с памятью микрокоманд, либо в виде набора БИС.

Управление в однокристалльных микропроцессорах. Основные элементы архитектуры однокристалльных МП с программным управлением (на примерах МП КР580ИК80А и КМ1810ВМ86) показаны на рис. 2.11, 2.22. Организованные подобным образом МП являются универсальными устройствами обработки данных с программным управлением.

Особенности архитектуры МП с «жестким» автоматом управления определяют два основных ограничения конструктивно-технологического порядка:

- 1) сравнительно низкая сложность процессорных функций, которые могут быть реализованы в пределах одного кристалла;
- 2) ограниченное число связей с внешней средой, что объясняется небольшим числом электрических контактов, которые могут быть размещены на полупроводниковом кристалле ограниченных физических размеров (40...50 мм²).

Первое ограничение носит временный характер, и с ростом степени интеграции сложность внутренней организации МП будет возрастать. Второе ограничение долговременное, преодолевается более трудно и, по-видимому, является принципиальным ограничением производительности систем, построенных на основе МП.

Управляющие автоматы однокристалльных МП могут строиться либо на базе произвольно-последовательной логики («жесткий» автомат), либо на основе программируемых логических матриц.

В результате того, что в большинстве случаев для выполнения одной процессорной команды требуется несколько элементарных действий (микроопераций), относительная независимость этих двух уровней управления позволяет при практически одинаковой архитектуре АЛУ иметь МП с различными наборами команд. Если организации АЛУ огромного множества МП практически не отличаются друг от друга и соответствуют архитектуре, показанной на рис. 2.11, 2.22, то системы команд имеют большое разнообразие как по числу, так и по содержательности.

Большинство однокристалльных МП имеют устройства управления, реализованные на основе «жесткого» управляющего автомата. Следует отметить, что микропрограммирование как атрибут архитектуры используется во всех без исключения МП — как в однокристалльных МП с программным управлением, так и в секционных МП с микропрограммным управлением. Однако в однокристалльных МП сама процедура микропрограммирования как средство упорядочения внутренней организации системы вылилась в псевдомикропрограммирование, поскольку программируемые логические матрицы недоступны пользователю. Иными словами, принцип псевдомикропрограммирования, реализованный в однокристалльных МП, не дает пользователю возможности составления микропрограмм, а предоставляет в его распоряжение только список команд для реализации процедур программного управления МП. Управляющий автомат в однокристалльных БИС под воздействием одной команды вырабатывает целую последовательность микроприказов, которые реализуют в МП операцию, определяемую кодом команды. Ввиду того что управляющий автомат однокристалльных МП занимает ограниченное место на кристалле, он относительно прост, а список команд МП не велик и содержит простые команды.

Организацию управления в однокристалльных МП рассмотрим на примере работы управляющего автомата МП КР580ИК80А.

Устройство управления МП под воздействием сигналов синхронизации инициирует начало выполнения команды, вырабатывая микроприказ выдачи содержимого программного счетчика РС в магистраль адреса (см. рис. 2.11). Выбранный из памяти байт информации идентифицируется как код операции и помещается в регистр команд. Дешифратор команд и устройство управления преобразуют код команды в последовательность микроприказов, реализующих в МП операцию, задаваемую кодом команды. В зависимости от сложности команды цикл команды реализуется за несколько (1...5) внутренних машинных циклов. Каждый машинный цикл может состоять из 3...5 тактов. Каждый такт машинного цикла образует пара сигналов синхронизации Φ_1 и Φ_2 , поступающих от внешнего генератора.

На рис. 2.28 показан алгоритм работы устройства управления (УУ). В начале каждого машинного цикла устройство управления генерирует сигнал синхронизации СИНХРО (SYNC). Одновременно с сигналом СИНХРО МП выдает на магистраль данных байт состояния, содержащий информацию о действиях МП в текущем машинном цикле.

Каждому такту сигналов синхронизации соответствует отдельное состояние Т УУ. На рис. 2.28 эти состояния обозначены символами $T_1 \dots T_5$, все они имеют одинаковую длительность (для МП КР580ИК80А период составляет 500 нс). Состояния ожидания ($T_{ож}$), прямого доступа к памяти ПДП ($T_{пдп}$) и останова ($T_{ост}$) не связаны с тактовой частотой сигналов синхронизации и их продолжительность неопределенная, так как зависит от внешних событий. Из схемы алгоритма функционирования УУ видно, что цикл команды может быть реализован УУ с числом переходов по внутренним состояниям от 4 до 18. В процессе реализации цикла команды участвуют сигналы управления состоянием микропроцессора: входные сигналы СБРОС (RESET) и ГОТОВ (READY), а также выходной сигнал ожидания (WAIT); сигналы управления ПДП: входной сигнал захвата магистралей (HOLD), выходной сигнал подтверждения захвата (HLDA); сигналы управления прерываниями: входной сигнал запроса прерывания ПР (INT), выходной сигнал разрешения прерывания РПР (INTE); сигнал ПОСТ (HLTA) означает подтверждение останова.

При выполнении команды в такте T_1 содержимое РС выдается на МА, на МД поступает 8-разрядное управляющее слово, формируется сигнал СИНХРО, по которому управляющее слово записывается в буферный регистр внешних схем управления. В такте T_2 содержимое РС увеличивается на единицу и осуществляется анализ управляющих сигналов ГОТОВ и ПДП (HOLD) и сигнала ПОСТ. В зависимости от значений этих сигналов производится переход либо к такту T_3 машинного цикла, либо в одно из состояний ожидания $T_{ож}$, $T_{ост}$, $T_{пдп}$ выход из которых возможен только при поступлении внешних управляющих сигналов. В такте T_3 данные принимаются из ЗУ или внешнего устройства на МД и записываются в регистр команд или в один из регистров МП. В тактах T_4 и T_5 , если они необходимы, выполняются действия над операндами. Если команда включает несколько

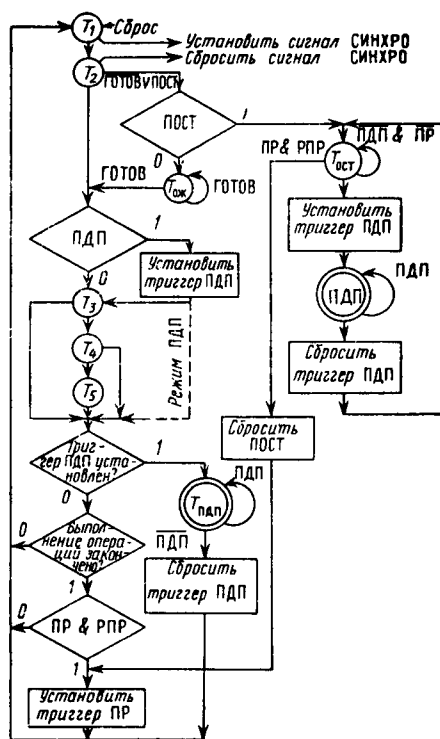


Рис. 2.28. Алгоритм работы устройства управления

циклов, то по завершении текущего машинного цикла процессор переходит к такту T_1 следующего цикла.

Средством управления МПС является слово состояния МП, которое индицирует имя машинного цикла, выполняемого микропроцессором. Индикация состояния производится на внешнем по отношению к МП регистре состояния. Слово, фиксируемое на этом регистре, определяет тип машинного цикла.

МП в первом такте каждого машинного цикла передает в МД слово состояния, информирующее МПС о процессах, происходящих в МП. Сигналы, которые представлены на выходе регистра состояния, используются в качестве сигналов управления периферийным оборудованием МПС. Всего МП КР580ИК80А имеет 10 типов машинных циклов и соответственно 10 кодов, идентифицирующих эти циклы. В табл. 2.2 приведены типы машинных циклов.

Таблица 2.2

Типы машинных циклов

Машинный цикл	Биты бита состояния							
	D_0 (INTA)	D_1 ($\overline{WO}$)	D_2 (STACK)	D_3 (HLTA)	D_4 (OUT)	D_5 (M_1)	D_6 (INP)	D_7 (MEMR)
ВЫБОРКА КОДА	0	1	0	0	0	1	0	1
ОПЕРАЦИИ								
СЧИТЫВАНИЕ ИЗ	0	1	0	0	0	0	0	1
ПАМЯТИ								
ЗАПИСЬ В ПАМЯТЬ	0	0	0	0	0	0	0	0
СЧИТЫВАНИЕ ИЗ	0	1	1	0	0	0	0	1
СТЕКА								
ЗАПИСЬ В СТЕК	0	0	1	0	0	0	0	0
ВВОД	0	1	0	0	0	0	1	0
ВЫВОД	0	0	0	0	1	0	0	0
ПРЕРЫВАНИЕ	1	1	0	0	0	1	0	0
ОСТАНОВ	0	0	0	1	0	0	0	1
ПРЕРЫВАНИЕ ПРИ	1	1	0	1	0	1	0	0
ОСТАНОВЕ								

Отметим, что биты бита состояния имеют следующее содержание: D_0 (INTA) — подтверждение прерывания; D_1 ($\overline{WO}$) — запись — вывод, причем при $\overline{WO} = 1$ МП является приемником информации, а при $\overline{WO} = 0$ — источником; D_2 (STACK) — стек; D_3 (HLTA) — подтверждение останова; D_4 (OUT) — вывод; D_5 (M_1) — первый ма-

Средством программной настройки интерфейсных БИС является управляющее слово. Для организации управления интерфейсными БИС в их состав введен регистр управления, и МП осуществляет через него программное управление периферийным устройством с использованием управляющих слов, которые не только определяют режим работы различных частей интерфейсной БИС, но и присваивают спе-

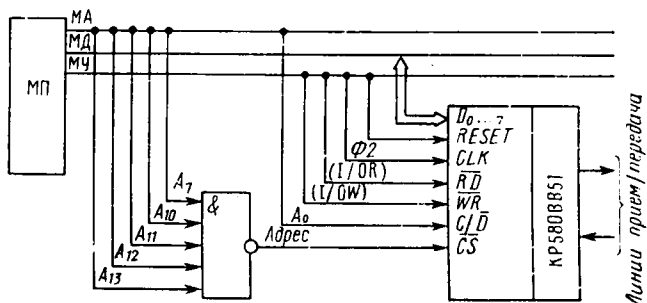


Рис. 2.29. Способ подключения МП и БИС последовательного интерфейса

Использование программируемых интерфейсных БИС привело к тому, что прием и передача данных в МПС реализуются с помощью достаточно сложного аппаратно-программного комплекса. На рис. 2.29 показан один из возможных вариантов соединения МП КР580ИК80А с БИС последовательного интерфейса КР580ВВ51. Ограниченное число внешних выводов корпуса БИС последовательного интерфейса позволяет подать на него только основные сигналы с магистрали управления МП. Среди них сигналы: RESET (СБРОС), CLK (ТАКТИРОВАНИЕ Φ_2), $\overline{RD}$ (ЧТЕНИЕ), $\overline{WR}$ (ЗАПИСЬ), а также сигнал CS (ВЫБОР КОРПУСА — разрешение работы), сформированный на внешнем адресном декодификаторе, и сигнал сопровождения информации, который передает по линии A_0 адресной шины и позволяет различать управляющие слово и байт данных ($C/\overline{D}$ — УПР./ДАННЫЕ). На рис. 2.29 обозначено: $D_{0...7}$ — двунаправленная МД, A_n — линия n МА $A_{0...15}$, I/OR, I/OW — ВВ/ВЫВ. В соответствии с представленной схемой адрес БИС последовательного интерфейса будет выглядеть так: 0011 1100 1000 0000 — при передаче данных и 0011 1100 1000 0001 — при передаче управляющих слов.

Таким образом, МП осуществляет управление МПС или микроЭВМ на трех уровнях:

программном путем генерации управляющих слов для интерфейсных БИС;

микропрограммном путем генерации слова состояния процессора в каждом машинном цикле;

микропрограммном в каждом состоянии управляющего автомата МП.

Принцип организации управления в однокристальных 16-разрядных МП (в том числе и МП КМ1810ВМ86) аналогичен описанному с

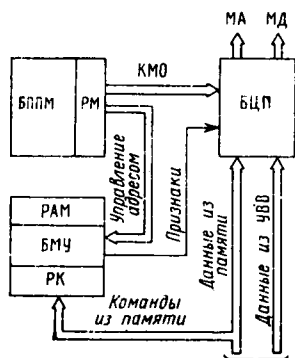


Рис. 2.30. Структура секционного МП с микропрограммным управлением

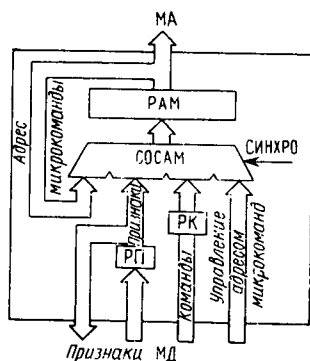


Рис. 2.31. Структура блока микропрограммного управления

той разницей, что система команд и режимы адресации у 16-разрядных МП намного разнообразнее. При этом следует учитывать, что система команд МП КР580ИК80А представляет собой подмножество команд МП КМ1810ВМ86 и обеспечивается программная совместимость «снизу вверх».

Управление в секционных микропроцессорах. Структура секционного МП с микропрограммным управлением представлена на рис. 2.30. В его состав входят блок центрального процессора БЦП, блок микропрограммного управления БМУ, блок постоянной памяти микропрограмм ВППМ, регистры микрокоманд РК и адреса микрокоманд РАМ; БЦП состоит из некоторого числа параллельно включенных БИС процессорных секций, что позволяет иметь АЛУ практически любой требуемой разрядности.

Секции БЦП объединены общей магистралью микропрограммного управления и соединены между собой по цепям межразрядных связей. В состав каждой секции БЦП входит дешифратор микроопераций, который преобразует код микроопераций в совокупность микроприказов, управляющих коммутацией узлов и блоков секции. Основ-

ное отличие секции БЦП от однокристалльного МП состоит в том, что здесь код микрооперации КМО не разворачивается во временную последовательность управляющих сигналов, а каждому входному коду соответствует одно-единственное слово микроприказа.

Структурная организация блока микропрограммного управления показана на рис. 2.31. Основное назначение БМУ — преобразование кода поступившей из внешней памяти команды в последовательность микрокоманд, т. е. передача в БППМ серии адресов, по которым расположены соответствующие микрокоманды, составляющие микропрограммную последовательность реализации данной команды. Сам принцип микропрограммного управления предполагает, что в каждом шаге перехода от предыдущей микрокоманды к последующей в БМУ подлежат анализу не только код изначальной системной команды, не только адрес текущей микрокоманды, являющийся базовым для перехода к последующей команде, но и система признаков, генерируемых БЦП под воздействием каждой очередной микрооперации и, кроме того, код управления следующим адресом, который содержится в специальном поле каждой микрокоманды. Все эти задачи решаются в БМУ комбинационной схемой определения следующего адреса микропрограммы СОСАМ.

Ценным качеством МП с микропрограммным управлением является возможность замены БИС (или нескольких БИС), в которых хранятся микропрограммы. При этом появляется возможность одну и ту же системную конфигурацию МП приспособлять для решения различных задач, причем для каждой из этих задач можно сформировать наиболее эффективную систему команд.

Использование секционного МП с микропрограммным управлением накладывает на разработчика МПС ряд ограничений и вынуждает принимать специфические решения проектных задач, возникающих в процессе разработки МПС: выбор структуры управления, распределение функций между аппаратными и программными средствами, выбор формата и способа кодирования микрокоманд, определение способов задания адресов и организация переходов по микропрограмме.

§ 2.7. АРХИТЕКТУРА СЕКЦИОННОГО МИКРОПРОЦЕССОРА КМ1804ВС2

Характеристика МПК К1804. Одним из наиболее распространенных микропрограммируемых МПК является МПК БИС серии 1804 (аналог Ат 2900). Микропроцессорный комплект серии КМ1804 (КР1804) выполнен на основе биполярной технологии (ТТЛШ) и является одним из быстродействующих и функционально полных комплектов. В состав комплекта входит более 20 БИС различного функционального назначения и проводятся работы по дальнейшему его расширению.

Основой для построения операционных блоков вычислительных устройств являются 4-разрядные микропроцессорные секции КМ1804ВС1 и КМ1804ВС2 с возможностью наращивания разрядности.

Микросхема КМ1804ВС2 по сравнению с микросхемой КМ1804ВС1 выполняет большее число арифметико-логических функций и, кроме того, дополнительно реализует умножение, деление, нормализацию и преобразование в дополнительный код, что позволяет существенно повысить быстродействие разработанных на ее основе систем и сэкономить объем требуемой памяти программ. Кроме того, БИС КМ1804ВС2 обеспечивает возможность расширения файла регистров общего назначения (РОН) за счет подключения внешней памяти и осуществляет контроль четности.

МПК серии КМ1804 имеет микропрограммное управление, реализуемое с помощью БИС КМ1804ВУ1 (КМ1804ВУ2) — схем управления адресом микрокоманды, БИС КМ1804ВУ3 — схем управления следующим адресом и внешнего ПЗУ микрокоманд. Для систем с большим набором команд вместо микросхем КМ1804ВУ1 и КМ1804ВУ3 целесообразно использовать микросхему КМ1804ВУ4—схему управления последовательностью микрокоманд. В состав комплекта входят также микросхемы: КМ1804ИР1, КМ1804ИР2 — параллельные регистры; КМ1804ИР3 — параллельный двунаправленный регистр; КМ1804ВА1, КМ1804ВА2, КМ1804ВА3 — магистральные приемопередатчики; КМ1804ВР1 — схема ускоренного переноса; КМ1804ГГ1 — генератор тактовых импульсов; КМ1804ВН — схема векторного приоритетного прерывания; КМ1804ВЖ1 — схема обнаружения и коррекции ошибок и др. Микросхемы серии КМ1804 выполнены в металлокерамических корпусах. Значительная часть МПК серии К1804 выпускается в пластмассовых корпусах. Такие микросхемы имеют обозначение КР1804. Их функциональное назначение и параметры соответствуют аналогичным по наименованию типонаминалам МПК серии КМ1804.

Микропрограммный способ управления позволяет создавать устройства с различными системами команд, однако пользователям при этом необходимо учитывать, что оригинальные системы команд требуют самостоятельной разработки программного обеспечения. На базе МПК серии К1804 строятся быстродействующие мини- и микроЭВМ, микропрограммируемые контроллеры, высокоскоростные внешние устройства, например накопители на магнитных дисках (НМД). В качестве примеров серийных микропроцессорных систем на основе комплекта К1804 можно привести миниЭВМ СМ1420, микроЭВМ «Электроника 81», «Электроника 82», контроллер НГМД «Электроника ГМД 7012», контроллеры НМД СМ 5400 и ЕС 5300.

Структурная схема БИС КМ1804ВС2. Рассмотрим более подробно организацию и основные принципы работы 4-разрядной микропроцессорной секции КМ1804ВС2. Она предназначена для построения операционных блоков цифровых устройств повышенного быстродействия с разрядностью, кратной четырем. Структурная схема этой секции приведена на рис. 2.32. Микропроцессорная секция КМ1804ВС2 состоит из следующих блоков: блока внутренней памяти БВП, ариф-

метико-логического блока АЛБ, блока рабочего регистра БРР и блока управления БУ.

Блок внутренней памяти состоит из 16-словного 4-разрядного регистрового запоминающего устройства (РЗУ) и двух регистров РгА, РгВ. Регистровое запоминающее устройство состоит из дешифратора адресов А ($A_0...A_3$) и В ($B_0...B_3$); 16 4-разрядных регистров общего назначения, схем считывания и записи. Регистры общего назначения предназначены для хранения данных внутри микропроцессорной секции. Каждый из 16 РОН может быть как источником операндов,

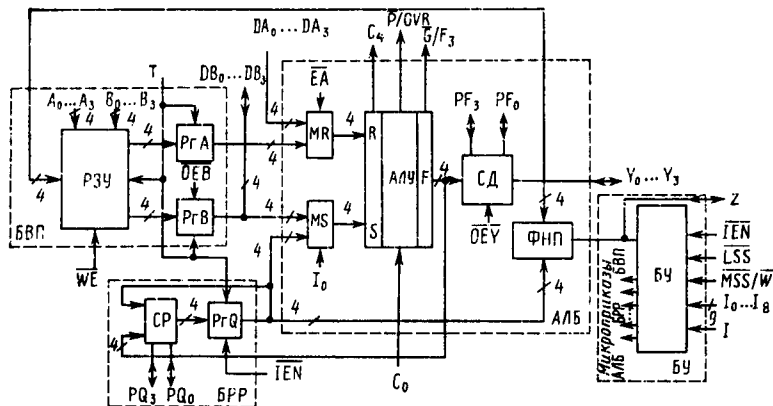


Рис. 2.32. Структурная схема микропроцессорной секции КМ1804ВС2

так и местом записи результата. Информация на вход данных РЗУ может поступать либо с выхода арифметико-логического блока, либо с двуправленных выходов $Y_0...Y_3$. Информация с выходов данных РЗУ поступает на регистры РгА и РгВ, которые управляются тактовыми сигналами Т. На выходе регистра РгВ (в отличие от РгА) установлены элементы с трехстабильными выходами, управляемые сигналом OEB . При $OEB = 1$ выходы регистра РгВ отключаются (состояние высокого сопротивления), а источником операнда на вход мультиплексора MS выбирается шина $DB_0...DB_3$.

Запись информации в РЗУ может производиться только по адресу В при наличии управляющих сигналов WE и Т, считывание информации из РЗУ — по адресам А и В одновременно. Если на адресных входах установлены одинаковые адреса, то по выходам А и В считывается одна и та же информация.

Блок рабочего регистра состоит из регистра Q (РгQ) и сдвигателя регистра Q (CP). Для управления блоком рабочего регистра используются управляющие сигналы с блока управления. Запись информации в регистр Q производится по положительному фронту тактового сигнала Т при $IEN = 0$. При $IEN = 1$ регистр РгQ находится в режиме

хранения информации. Регистр Q может служить источником операнда S арифметико-логического устройства (АЛУ) и приемником информации через сдвигатель регистра с выходов АЛУ (без сдвига) или с собственных выходов (со сдвигом). Сдвигатель регистра Q выполняет логические сдвиги на один разряд в сторону старших или младших разрядов содержимого регистра RgQ или передает информацию с выхода АЛУ либо регистра RgQ несдвинутой. Когда сдвиги не выполняются, выходы PQ_0 и PQ_3 отключаются. При выполнении сдвига в сторону младших разрядов шина PQ_3 становится входом, шина PQ_0 — выходом, а при сдвиге в сторону старших разрядов наоборот. Блок рабочего регистра используется главным образом при выполнении операций умножения и деления, а также в качестве аккумулятора.

Арифметико-логический блок состоит из двух входных мультиплексоров MR и MS, арифметико-логического устройства АЛУ, сдвигателя данных СД и формирователя нулевого признака ФНП. Входные мультиплексоры осуществляют выбор источников операндов R и S под воздействием управляющих сигналов $\overline{EA}$, IO и $\overline{OEB}$ в соответствии с табл. 2.3.

Таблица 2.3

Источники операндов АЛУ

$\overline{EA}$	I_0	$\overline{OEB}$	Источник операнда R	Источник операнда S
0	0	0	PrA	PrB
0	0	1	PrA	$DB_0 \dots DB_3$
0	1	X	PrA	PrQ
1	0	0	$DA_0 \dots DA_3$	PrB
1	0	1	$DA_0 \dots DA_3$	$DB_0 \dots DB_3$
1	1	X	$DA_0 \dots DA_3$	PrQ

Обозначения в таблице: X — значение сигнала безразлично (может быть как «1», так и «0»);

$DA_0 \dots DA_3$ — входная шина данных.

Арифметико-логическое устройство обеспечивает выполнение семи арифметических и девяти логических операций, а также девяти специальных функций над одним или двумя 4-разрядными входными операндами R и S, поступающими с выходов мультиплексоров MR, MS. АЛУ вырабатывает сигналы состояния $\overline{G}$, F_3 , $\overline{P}$, OVR, C_4 . Сигналы генерации переноса $\overline{G}$ и распространения переноса $\overline{P}$ необходимы для организации ускоренного переноса. Так как в старшей секции выходы $\overline{P}$, $\overline{G}$ не задействованы, а с другой стороны только в старшей секции используются выход старшего разряда результата АЛУ (F_3) и вы-

ход переполнения (OVR), то в микропроцессорной секции применяются объединенные выходы $\overline{G}/F_3$, $\overline{P}/OVR$, которые используются как $\overline{G}$, $\overline{P}$ для средних и младших секций и как F_3 , OVR — для старшей секции. АЛУ имеет также выводы для приема сигнала входного переноса C_0 и выдачи сигнала выходного переноса C_4 . Вход C_0 и выход C_4 предназначены для обеспечения распространения последовательного переноса. При соединении нескольких секций в качестве выходного переноса служит сигнал переноса C_4 старшей секции.

Информация с выхода АЛУ может передаваться на вход блока внутренней памяти БВП и на шину $Y_0...Y_3$ без сдвига или со сдвигом

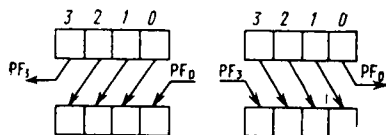


Рис. 2.33. Организация логических сдвигов

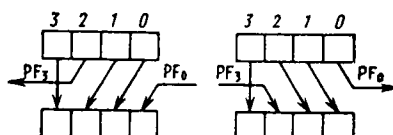


Рис. 2.34. Организация арифметических сдвигов

на один разряд в сторону младших или старших разрядов, для чего сдвигатель данных СД управляется сигналами, формируемыми блоком управления. Когда сдвиги не выполняются, информация с выхода АЛУ передается через СД немодифицированной, выходы PF_0 и PF_3 отключаются. При выполнении логического сдвига в сторону старших разрядов PF_0 становится входом, а PF_3 — выходом, а при выполнении логического сдвига в сторону младших разрядов — наоборот (рис. 2.33).

Сдвигатель данных АЛУ в отличие от сдвигателя регистра RgQ кроме описанных логических сдвигов, когда все разряды сдвигаются одинаково, может выполнять и арифметические сдвиги. Арифметические сдвиги выполняются так же, как и логические, за исключением того, что старший (знаковый) разряд не участвует в сдвигах (рис. 2.34)

Для управления трехстабильными выходами СД используется сигнал $\overline{OEY}$. При $\overline{OEY} = 0$ информация с выхода СД передается на шину $Y_0...Y_3$ и в БВП. При $\overline{OEY} = 1$ выходы СД отключаются (состояние высокого сопротивления), а шина $Y_0...Y_3$ может применяться в качестве шины данных для записи в БВП.

Формирователь нулевого признака ФНП участвует в выполнении арифметических и логических операций, специальных функций и формировании сигналов состояния. ФНП вырабатывает сигнал состояния Z . Наличие «1» на выводе Z означает, что все сигналы на выходах АЛУ и на выходах регистра RgQ нулевые. При выполнении некоторых специальных функций шина Z становится входом. Блок управления предназначен для формирования внутренних управляющих сигналов, оп-

ределяющих положение секции в системе, выполняемую функцию АЛУ и т. д.

Сигналы микрокоманды $I_0 \dots I_8$ определяют функцию, выполняемую микропроцессорной секцией, источники операндов и местоположение результата. При этом, если $I_0 = I_1 = I_2 = I_3 = I_4 = 0$, то сигналы $I_5 \dots I_8$ определяют одну из девяти специальных функций, выполняемых секцией. Если хотя бы один из сигналов $I_0 \dots I_4$ не нулевой, то сигналы $I_8 \dots I_5$ определяют местоположение результата операции, сигналы $I_1 \dots I_4$ — одну из 16 простых функций, выполняемых секцией, а сигналы I_0 , $\overline{EA}$ и $\overline{OEB}$ — источники операндов, подаваемых на вход АЛУ.

Сигнал разрешения записи в PgQ и разблокировки выхода $\overline{W}$ младшей микропроцессорной секции $\overline{IEN}$ используется для управления регистром Q и схемой выработки признака сравнения знаков.

Секционная наращиваемость — одна из архитектурных особенностей микропроцессорной БИС КМ1804BC2, которая дает возможность соединять микропроцессорные 4-разрядные секции для получения процессора любой разрядности, кратной четырем.

Функционирование микропроцессорной секции при выполнении арифметических операций зависит от ее положения в системе. Поэтому при соединении секций необходимо выполнить их настройку на заданное положение (старшая, средняя или младшая). Такая настройка выполняется с помощью шин $\overline{LSS}$ и $\overline{MSS/W}$. В младшей секции на входе $\overline{LSS}$ устанавливается «0», при этом шина $\overline{MSS/W}$ становится выходом $\overline{W}$. В средней и старшей секциях на входе $\overline{LSS}$ устанавливается «1» при этом шина $\overline{MSS/W}$ становится входом $\overline{MSS}$. Для старшей секции $\overline{MSS} = 0$, для средней $\overline{MSS} = 1$ (табл. 2.4).

Таблица 2.4

Положение секции в многоразрядной системе		
$\overline{LSS}$	$\overline{MSS/W}$	Положение секции
0	Выход $\overline{W}$	Младшая
1	0	Старшая
1	1	Средняя

Набор операций БИС КМ1804BC2. Рассмотрим более подробно набор операций, выполняемых микропроцессорной секцией КМ1804BC2. Выбор операции АЛУ осуществляется сигналами, которые формирует блок управления при поступлении на него сигналов микрокоманды $I_0 \dots I_8$.

Если хотя бы на один из входов $I_0...I_4$ подана «1», то выполняется одна из 16 операций, приведенных в табл. 2.5. Управление операндами на входах R и S в этом случае обеспечивается сигналами $\overline{EA}$, $\overline{OE}$ и $\overline{IO}$ в соответствии с табл. 2.3. Управление результатом операции осуществляется разрядами микрокоманды $I_5...I_8$ согласно табл. 2.6. При обозначении сдвигов в табл. 2.6 использованы следующие сокращения: САЛ и САП — сдвиги арифметические левый и правый; СЛЛ и СЛП — сдвиги логические левый и правый.

Таблица 2.5

Операции АЛУ

Номер операции	Код микрокоманды					Операция АЛУ
	I_4	I_3	I_2	I_1	I_0	
	0	0	0	0	0	См. табл. 2.7 «Специальные функции АЛУ»
1	0	0	0	0	1	$F=1111$
2	0	0	0	1	X	$F=S-R-I+C_0$
3	0	0	1	0	X	$F=R-S-I+C_0$
4	0	0	1	1	X	$F=R+S+C_0$
5	0	0	0	0	X	$F=S+C_0$
6	0	1	0	1	X	$F=S+C_0$
7	0	1	1	0	X	$F=R+C_0$
8	0	1	1	1	X	$F=\overline{R}+C_0$
9	1	0	0	0	X	$F=0000$
10	1	0	0	1	X	$F=\overline{R} \wedge S$
11	1	0	1	0	X	$F=R \oplus S$
12	1	0	1	1	X	$F=R+S$
13	1	1	0	0	X	$\overline{F}=\overline{R \wedge S}$
14	1	1	0	1	X	$\overline{F}=\overline{R \vee S}$
15	1	1	1	0	X	$F=F \wedge S$
16	1	1	1	1	X	$F=R \vee S$

Примечание. X — значение сигнала безразлично (может быть как «1», так и «0»).

Набор операций АЛУ включает семь арифметических операций (2...8) и девять поразрядных логических операций (1, 9, ..., 16). Арифметические операции используют вход переноса АЛУ — C_0 как дополнительный одноразрядный операнд, имеющий вес единицы младшего разряда. Его значение для младшей процессорной сек-

Таблица 2.6

Управление результатом операции АЛУ, СД и СР

Номер операции	Код микрокоманды				Функции сдвигов		$\bar{W}$
	I_6	I_7	I_8	I_9	СД	СР	
1	0	0	0	0	САП (F)→Y	Q→Q	0
2	0	0	0	1	СЛП (F)→Y	0→Q	0
3	0	0	1	0	САП (F)→Y	СЛП (Q)→Q	0
4	0	0	1	1	СЛП (F)→Y	СЛП (Q)→Q	0
5	0	1	0	0	F→Y	Q→Q	0
6	0	1	0	1	F→Y	СЛП (Q)→Q	1
7	0	1	1	0	F→Y	F→Q	1
8	0	1	1	1	F→Y	F→Q	0
9	1	0	0	0	САЛ (F)→Y	Q→Q	0
10	1	0	0	1	СЛЛ (F)→Y	Q→Q	0
11	1	0	1	0	САЛ (F)→Y	СЛЛ (Q)→Q	0
12	1	0	1	1	СЛЛ (F)→Y	СЛЛ (Q)→Q	0
13	1	1	0	0	F→Y	Q→Q	1
14	1	1	0	1	F→Y	СЛЛ (Q)→Q	1
15	1	1	1	0	PFO→Y0, Y1, Y2, Y3	Q→Q	0
16	1	1	1	1	F→Y	Q→Q	0

Примечание. Обозначения в таблице: САП — сдвиг арифметически правый; СЛП — сдвиг логический правый; САЛ — сдвиг арифметический левый; СЛЛ — сдвиг логический левый.

ции может задаваться специальным разрядом микрокоманды. Поразрядные логические операции выполняются над одноименными разрядами операндов R и S. Набор логических операций включает конъюнкцию, дизъюнкцию, функцию запрета, сложение по модулю два (неравнозначность) и его инверсию (равнозначность). Вход переноса C_0 на результат логических операций не влияет. Результат операции с выходов F АЛУ может выдаваться на выходную шину секции $Y_0...Y_3$, а также заноситься в регистры РЗУ по адресу $B_0...B_3$ и в регистр PqQ без сдвига или со сдвигом на один разряд вправо или влево (табл. 2.6).

Функции управления результатом (см. табл. 2.6) содержат сдвиговые операции, имеющие следующий смысл. Для регистра Q выполняется только логический сдвиг, при этом выполняемые преобразования имеют вид, показанный на рис. 2.33.

При выдаче с шины F на шину Y логические сдвиги для всех процессорных секций, а также арифметические сдвиги для младшей и

средней секций выполняются так, как показано на рис. 2.33, 2.34. Арифметический сдвиг реализует умножение на два (левый) или деление на два (правый) числа со знаком в дополнительном коде. При этом старший разряд трактуется как знак числа.

Выходной сигнал $\overline{W}$ младшей процессорной секции (см. табл. 2.6) можно использовать для управления записью результата в РЗУ по адресу $B_0...B_3$ и в регистр PrQ. Для этого его следует подать на входы WE и $\overline{TEN}$. Код микрокоманды $I_8...I_5 = 1110$ используется для расширения знака PF_0 на все выходы процессорной секции $Y_3...Y_0$. Это бывает необходимо, например, при расширении байтовых данных до 16-разрядных для выполнения арифметических операций.

Специальные функции, выполняемые микропроцессорной секцией KM1804BC2, определяются сигналами $I_8...I_5$ при наличии «0» на входах $I_1...I_0$. БИС может выполнять девять специальных функций, которые перечислены в табл. 2.7. Семь кодовых комбинаций сигналов $I_8...I_5$ не используются и являются запрещенными.

Таблица 2.7

Специальные функции АЛУ

Номер операции	Код микрокоманды				Операция АЛУ	Функция АЛУ
	I_8	I_7	I_6	I_5		
1	0	0	0	0	Умножение без знака	$F = \begin{cases} S + C_0 & \text{при } Z = 0, \\ S \div R \div C_0 & \text{при } Z = 1 \end{cases}$
2	0	0	1	0	Умножение в дополнительном коде	$F = \begin{cases} S \div C_0 & \text{при } Z = 0, \\ S \div R \div C_0 & \text{при } Z = 1 \end{cases}$
3	0	1	0	0	Инкремент на 1 или 2	$F = S + 1 \div C_0$
4	0	1	0	1	Преобразование числа со знаком $\leftrightarrow$ дополнительный код	$F = \begin{cases} S \div C_0 & \text{при } Z = 0, \\ S \div C_0 & \text{при } Z = 1 \end{cases}$
5	0	1	1	0	Последний цикл умножения в дополнительном коде	$F = \begin{cases} S \div C_0 & \text{при } Z = 0, \\ S - R - 1 & \text{при } Z = 1 \end{cases}$
6	1	0	0	0	Нормализация обычной длины	$F = S \div C_0$
7	1	0	1	0	Нормализация двойной длины	$F = S \div C_0$
8	1	1	0	0	Деление в дополнительном коде	$F = \begin{cases} S \div R \div C_0 & \text{при } Z = 0, \\ S - R - 1 \div C_0 & \text{при } Z = 1 \end{cases}$
9	1	1	1	0	Деление в дополнительном коде (коррекция)	$F = \begin{cases} S \div R \div C_0 & \text{при } Z = 0, \\ S - R - 1 \div C_0 & \text{при } Z = 1 \end{cases}$

Операция нормализации числа в дополнительном коде выполняется путем сдвига числа в сторону старших разрядов до тех пор, пока два старших разряда не будут иметь различные значения. Если нормализуемое число имеет K первых нулей или единицу, то для его нормализации необходимо выполнить команду нормализации $K - 1$ раз.

При выполнении функции «Преобразование числа со знаком $\leftrightarrow$ дополнительный код» положительные числа не модифицируются, а отрицательные преобразуются в положительные.

В результате выполнения команды «Умножение без знака» для двух n -разрядных сомножителей получится $2n$ -разрядное произведение, n старших разрядов которого находится в первом РОН, а n млад-

Поле управления операциями АЛУ										Поле адресов РОН								Поле входных данных											
T	TEN	WE	OEY	OEB	EA	I ₀ ...I ₈				A ₀ ...A ₃				B ₀ ...B ₃				C ₀	DB ₀ ...DB ₃				DA ₀ ...DA ₃						
T	TEN	WE	OEY	OEB	EA	I ₀	...	I ₈	A ₀	A ₁	A ₂	A ₃	B ₀	B ₁	B ₂	B ₃	C ₀	DB ₀	DB ₁	DB ₂	DB ₃	DA ₀	DA ₁	DA ₂	DA ₃				
(1)	(2)	(3)	(4)	(5)	(6)	(7)	...	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)	(32)				

Рис. 2.35. Полный формат микрокоманды управления работой микропроцессорной секции КМ1804BC2

ших — в регистре PgQ . Для выполнения умножения без знака требуется хранить только две микрокоманды. Команда «Умножение без знака» выполняется за $n + 1$ тактов.

Для выполнения команды «Умножение в дополнительном коде» потребуется хранить три микрокоманды. При этом умножение в дополнительном коде выполняется за $n + 1$ тактов.

Функция «Деление в дополнительном коде» выполняется путем реализации алгоритма деления без восстановления остатка. Предполагается, что делимое по модулю меньше делителя.

Таким образом, для использования всех функциональных возможностей микропроцессорной секции КМ1804BC2 необходимо подавать на нее из управляющей памяти следующие данные: сигналы адреса $A_0 \dots A_3$ и $B_0 \dots B_3$; сигналы микрокоманды $I_0 \dots I_8$; сигнал разрешения передачи в АЛБ — $\overline{EA}$; сигнал разрешения съема данных с БВП — $\overline{OEB}$; сигнал разрешения съема данных с АЛБ — $\overline{OEY}$; сигнал разрешения записи в БВП — $\overline{WE}$; сигнал разрешения записи в PgQ и разблокировки выхода $\overline{W}$ младшей секции — $\overline{TEN}$; сигнал переноса в АЛУ — C_0 ; входные сигналы данных $DA_0 \dots DA_3$; входные сигналы данных при расширении объема РЗУ $DB_0 \dots DB_3$; тактовый сигнал T .

Сигналы адреса $A_0 \dots A_3$ и $B_0 \dots B_3$ используются для выбора соответствующих регистров РЗУ. Данные могут быть записаны в РЗУ только по адресу $B_0 \dots B_3$. Сигналы $I_0 \dots I_8$ определяют функцию, выполняемую секцией, источники операндов и местоположение результата. Сигналы используются: $\overline{EA}$ — для выбора источника операнда R

АЛУ, $\overline{\text{ОЕВ}}$ — для отпираания трехстабильных выходов регистра В и выбора источника операнда S АЛУ; $\overline{\text{ОЕУ}}$ — для отпираания трехстабильных выходов АЛБ и выдачи информации в магистраль микропроцессорной системы; $\overline{\text{WE}}$ — для управления работой РЗУ; Т — для управления и синхронизации блоков микропроцессорной секции.

Полный формат микрокоманды управления работой микропроцессорной секции приведен на рис. 2.35. Длина микрокоманды — 32 двоичных разряда. Разряды $I_0 \dots I_8$, $V_0 \dots V_3$, $A_0 \dots A_3$, Т являются обязательными и неизменными. Оставшиеся разряды можно для конкретных применений иногда исключать (не все) и уменьшать длину микрокоманды, а следовательно, и объем управляющей памяти. Следует иметь в виду, что при использовании микропроцессорной секции КМ1804BC2 в микропроцессорных системах к формату микрокоманды, приведенному на рис. 2.35, добавятся поля управления блоком микропрограммного управления и интерфейсными устройствами.

§ 2.8. СТРУКТУРА ПАМЯТИ МПУ

Памятью МПУ или микроЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации. Отдельные устройства, входящие в эту совокупность, называют запоминающими устройствами (ЗУ).

Производительность и вычислительные возможности МПУ и микроЭВМ в значительной степени определяются составом и характеристиками используемых ЗУ. В составе МПУ используется одновременно несколько типов ЗУ, различающихся 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*. При обращении к памяти производится считывание или запись некоторой единицы данных для устройств разного типа. Такой единицей может быть, например, байт, машинное слово или блок данных.

Классификация запоминающих устройств. По *принципу действия* ЗУ можно разделить на полупроводниковые, магнитные с неподвижными магнитными запоминающими элементами и магнитомеханические с движущимся магнитным носителем информации.

По *функциональному назначению* ЗУ подразделяют на *оперативные* и *постоянные*. Последние, в свою очередь, делятся на ЗУ, в которых информация записывается однократно (либо в процессе изготовления, либо в процессе программирования) и не может быть изменена в процессе эксплуатации (ПЗУ), и программируемые ЗУ (ППЗУ), в которых хранящая информация может меняться в процессе эксплуатации.

По *способности сохранять или не сохранять содержимое* при включении питания различают *энергозависимые* и *энергонезависимые* ЗУ. Энергонезависимая память сохраняет свое содержимое при отключении

питания, а энергозависимая — не сохраняет. Примерами энергонезависимой памяти являются ПЗУ, а также ОЗУ на ферритовых сердечниках, примером энергозависимой памяти — оперативная память на полупроводниковых микросхемах. Поскольку ферритовая память требует громоздкого электронного обрамления и оказывается сравнительно дорогой, в МПУ для хранения информации при выключении питания предусматривается внешняя память или полупроводниковая память с резервным питанием.

По *методам доступа к информации* (выборки информации) устройства памяти подразделяют на ЗУ с произвольной выборкой (ЗУПВ), ассоциативной выборкой (АЗУ) и с последовательной выборкой. В ЗУПВ обеспечивается непосредственный доступ к любой заданной ячейке в любой последовательности как при записи, так и при считывании, при этом время обращения не зависит от адреса запоминающей ячейки и является одинаковым для всех ячеек ЗУ. Отметим, что ОЗУ, ПЗУ и ППЗУ относятся к классу ЗУПВ и работают с той же тактовой частотой, что и МПУ. В ЗУ с последовательным доступом для выборки информации по определенному адресу необходимо последовательно просмотреть все ячейки, предшествующие заданной, так что время обращения к информации в таких ЗУ зависит от адреса ячейки. В ассоциативных ЗУ поиск и извлечение информации производятся не по ее адресу, а по некоторому характеристическому признаку самой информации, содержащемуся в запоминающей ячейке.

По *принципу хранения информации* полупроводниковые ЗУ делятся на статические и динамические. В статических ЗУ информация хранится в форме устойчивого состояния запоминающего элемента все время, пока поддерживается напряжение питания. В динамических ЗУ информация хранится в форме заряда на запоминающем конденсаторе, и для хранения записанной информации требуется периодическое проведение процесса регенерации, во время которого происходит восстановление исходных логических уровней.

Независимо от типа полупроводникового интегрального ЗУ основными его блоками являются запоминающее поле (накопитель), представляющее собой матрицу ячеек памяти и выполняющее функцию хранения информации, и периферийные обслуживающие устройства, выполняющие функции выбора (дешифрации) адресов, формирования сигналов записи и считывания, усиления сигналов и т. д. Во всех полупроводниковых ЗУ на кристалле БИС содержатся все перечисленные блоки.

Основные характеристики системы памяти. Одной из основных характеристик системы памяти является *емкость памяти* — наибольший объем данных, выраженный в единицах информации, который одновременно может храниться в ЗУ. Емкость измеряется в битах, машинных словах, но большей частью в байтах.

Удельная емкость есть отношение емкости ЗУ к его физическому объему. *Ширина выборки* определяется количеством информации в битах, записываемой в ЗУ или извлекаемой из него за одно обращение.

Быстродействие ЗУ характеризуется временем обращения к ЗУ, которое определяется временным интервалом между двумя следующими друг за другом обращениями к ЗУ. *Время обращения* (длительность цикла обращения) определяет максимальную частоту обращения к ЗУ. В течение цикла можно выбрать информацию (чтение), ввести информацию (запись), обновить или модифицировать состояние некоторого элемента ЗУ. Не следует смешивать время обращения ЗУ и *время выборки* — отрезок времени от момента подачи сигнала записи или чтения до завершения соответствующей операции. В зависимости от принципа работы ЗУ время цикла может совпадать с временем выборки или несколько превышать его. Так, в статических ЗУ эти времена совпадают, а в динамических помимо времени выборки в длительность цикла считывания входит время, необходимое для подготовки внутренних схем к работе.

Для оценки экономических характеристик ЗУ применяется показатель *удельной стоимости*, определяемый отношением его стоимости к информационной емкости.

Потребляемая энергия очень важна для систем, которые работают от аккумуляторов или солнечных элементов. Основопологающим фактором для потребляемой каждым запоминающим элементом энергии является применяемая технология. Наиболее предпочтительной для изготовления ЗУ с минимальным потреблением энергии является КМОП-технология. К сожалению, потребляемая мощность и быстродействие связаны обратно пропорциональной зависимостью, поэтому, оптимизировать оба эти показателя довольно сложно.

Общая организация памяти. Память МПУ или микроЭВМ подразделяют на два класса: 1) область памяти, к которой в процессе функционирования процессор может обращаться за командами и данными непосредственно; 2) область памяти, к которой процессор непосредственно обращаться не может. Память первого класса часто называют внутренней (основной), а память второго класса — внешней (массовой). Для того чтобы использовать информацию, хранящуюся во внешней памяти, ее необходимо предварительно передать во внутреннюю память.

К внутренней памяти относится регистровая память МП, ОЗУ и ПЗУ, к внешней памяти — ЗУ с прямым доступом (ЗУ на гибких и жестких магнитных дисках) и с последовательным доступом (ЗУ на перфолентах, магнитных лентах). Иерархическая структура памяти МПУ представлена на рис. 2.36.

Внутренняя память МПУ обычно состоит из одной или нескольких печатных плат, которые подключены к системной шине. На каждой плате находится модуль памяти (ПЗУ или ОЗУ), адресуемый старшими битами шины адреса. Отметим, что, как показано на рис. 2.37, в большинстве МПУ и микроЭВМ имеются модули ПЗУ и ОЗУ, однако в малых МПУ типа контроллеров могут быть только модули ПЗУ, а оперативная память находится на одной и той же печатной плате (и даже в одной и той же микросхеме), что и процессор.

но включить параллельно восемь БИС памяти (т. е. $m = 8$). В этом случае совокупность восьми одноразрядных линий данных всех восьми БИС ЗУ образует 8-разрядную шину данных систем памяти. Линии адреса и чтения—записи всех БИС включаются параллельно для одновременного обращения ко всем БИС ЗУ.

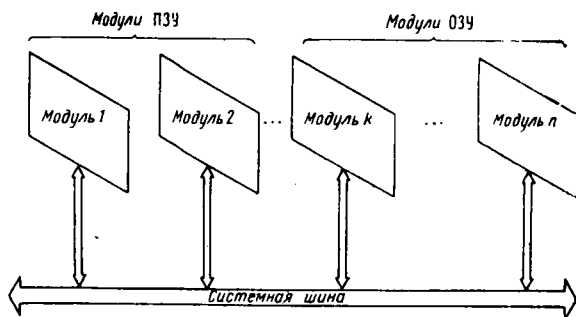


Рис. 2.37. Конструктивное разбиение внутренней памяти МПУ

В связи с ограничением количества внешних выводов корпуса БИС в ЗУПВ с информационной емкостью 16 К бит и выше используется мультиплексирование адресных каналов.

Организация интегральных ПЗУ. Полупроводниковые ПЗУ являются энергонезависимыми приборами с произвольной выборкой ин-

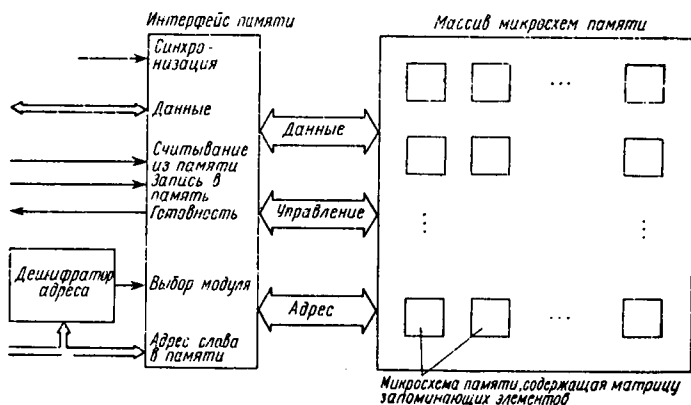


Рис. 2.38. Структурная схема модуля памяти

формации. Так же как и в БИС оперативной памяти, на кристалле БИС ПЗУ содержатся накопитель и периферийные обслуживающие схемы, необходимые для вывода информации.

Практически все промышленные типы БИС ПЗУ имеют байтовую (8-разрядную) или полубайтовую (4-разрядную) организацию, при-

чем наиболее экономичной и удобной является 8-разрядная организация.

Различают следующие виды ПЗУ: программируемые маской на предприятии-изготовителе и программируемые пользователем на специальных установках.

Масочные БИС ПЗУ используются при серийном выпуске отлаженных МПУ и микроЭВМ для хранения информации, не зависящей от конкретных приложений (системного программного обеспечения, тестовых программ, микропрограмм и др.). Программирование таких ПЗУ осуществляется на основе специальных фотошаблонов (масок), с помощью которых информация заносится на кристалл в ПЗУ в процессе его производства. Этот способ программирования является самым дешевым и предназначен для крупносерийного производства ПЗУ.

Среди разнообразных типов ПЗУ, *программируемых пользователем*, наибольшее распространение получили ПЗУ с плавкими перемычками, с ультрафиолетовым стиранием, с электрическим стиранием и записью информации.

Во всех типах ПЗУ на кристалле микросхемы размещается двумерная матрица запоминающих элементов, каждый из которых располагается на пересечении словарных линий, и разрядных линий, перпендикулярных

Рис. 2.39. Структурная схема полупроводникового ЗУПВ

идущих от дешифратора, и разрядных линий, перпендикулярных словарным и подсоединенных к мультиплексорам.

В ПЗУ с плавкими перемычками запоминающим элементом является транзистор типа *n-p-n*, база которого подсоединена к словарной линии, коллектор — к источнику положительного напряжения, а эмиттер через плавкую перемычку — к разрядной линии. В качестве плавкой перемычки используется поликристаллический кремний или нихром, напыленные в процессе изготовления микросхемы. Процесс программирования протекает таким образом. На адресные входы подается адрес выбранной ячейки. Напряжение питания микросхемы повышается до значения, необходимого для создания тока, достаточного для выплавления перемычки (обычно импульс тока 50...100 мкА длительностью 2 мс). Далее указываются разряды слова, в которых будет выплавляться перемычка. Процесс программирования обеспечивается программирующим устройством, которое параллельно производит контроль правильности программирования. Очевидно, микросхемы ПЗУ данного типа допускают только однократную запись ячейки.

ПЗУ с ультрафиолетовым стиранием (УФ-стиранием) и с электрическим стиранием являются *программируемыми ПЗУ* (ППЗУ), поскольку после стирания хранимой в ПЗУ информации возможно занесение в накопитель новых данных.

Микросхемы ПЗУ с У Ф - с т и р а н и е м характеризуются возможностью многократного программирования, достаточно малым временем выборки и энергопотреблением, большой емкостью. Все это делает их предпочтительными для применения в качестве памяти МПУ и микроЭВМ с сохранением информации после отключения питания.

Запоминающим элементом в ПЗУ с УФ-стиранием является МОП-транзистор, расположенный на пересечении соответствующих словарной и разрядной линий. Затвор этого транзистора подключен к словарной линии, сток — к источнику положительного напряжения, а исток — к разрядной линии. Информация о содержимом данной ячейки хранится в виде заряда на втором, полностью изолированном (плавающем) затворе МОП-транзистора. Плавающим затвор называется потому, что он размещен между управляющим затвором транзистора и его каналом и окружен высокоомным диэлектриком. Такое положение затвора позволяет формировать вольт-амперную характеристику транзистора, изменяя напряжение отпирания. При вводе в плавающий затвор отрицательного заряда будет формироваться смещенная характеристика транзистора. При программировании в результате поступления программирующего импульса на затвор транзистора последний включается, что приводит к появлению на разрядной шине низкого потенциала. Транзистор, который на программировался, будет иметь нулевой заряд и останется выключенным, что приведет к появлению на разрядной шине высокого потенциала.

При необходимости в программировании микросхемы предварительно записанную информацию стирают ультрафиолетовым излучением через прозрачное кварцевое окошко на поверхности корпуса микросхемы. УФ-излучение, попадая на плавающий затвор и выбивая из него фотоэлектроны, разряжает плавающий затвор МОП-транзистора. Время сохранения информации в ПЗУ данного вида составляет 10 лет и более.

В ПЗУ с э л е к т р и ч е с к и м с т и р а н и е м основу запоминающей ячейки составляет МОП-транзистор с плавающим затвором, такой же, как и в ПЗУ с УФ-стиранием. Но в микросхемах данного типа технологическими методами обеспечена возможность отбора электронов с плавающего затвора, что позволяет выборочно стирать занесенную информацию. В таких ПЗУ допустимое число циклов перезаписи информации составляет 10 000 и более.

Следует отметить, что при создании микропроцессорных устройств и систем емкости основных ЗУ может оказаться недостаточно и расширение памяти проводится за счет внешней памяти. В настоящее время наиболее распространенными устройствами внешней памяти являются гибкие и жесткие магнитные диски, кассетные магнитофоны и накопители на перфолентах. Перспективными внешними ЗУ являются де-

швые гибкие магнитные диски, высокоскоростные жесткие диски большой емкости (20...80 М байт) и оптические диски, созданные на базе мощных высокочастотных диодных лазеров. По прогнозам специалистов, двусторонний оптический диск диаметром 5,25 дюйма легко сможет хранить 1 Г байт информации.

Контрольные вопросы и задания

1. Сформулируйте принцип дуализма в микропроцессорной технике.
2. Приведите типовую структуру магистральной МПС.
3. Проанализируйте структуры сосредоточенных МПС по различным критериям (производительность, потребляемая мощность, габариты и др).
4. Почему в микропроцессорных ЭЭС широко используется функциональная децентрализация?
5. Почему в состав МПК БИС включают интерфейсные схемы?
6. Как работают адресные дешифраторы?
7. В чем заключается сущность программно-управляемого обмена данными?
8. Что подразумевается под прерыванием?
9. Объясните особенности организации режима прямого доступа к памяти.
10. Что характеризует слово состояния процессора?
11. Объясните функциональное назначение регистра флажков.
12. Чем вызвано обилие способов адресации микропроцессора?
13. В каких системах используется страничная память?
14. Зачем при программировании пользуются мнемоническими обозначениями и шестнадцатеричным кодом?
15. Объясните назначение сегментных регистров в программной модели микропроцессора.
16. Объясните сущность организации управления в однокристальных и секционных микропроцессорах.
17. Как формируются поля микрокоманд?
18. Как микропроцессор выполняет арифметические операции?
19. Что подразумевается под памятью статической, динамической и памятью, не сохраняющей информацию при отключении напряжения питания?
20. Как осуществляется занесение информации в полупроводниковые ПЗУ?

§ 3.1. СТРУКТУРА И СПОСОБЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Основные понятия и определения. Под программным обеспечением понимают совокупность программных средств, связанных с ними данных и необходимой документации. Разработка программного обеспечения (ПО) представляет собой формализованный процесс, посредством которого цели создания программ трансформируются в требования к ним; требования в свою очередь воплощаются в проектные решения, реализуются в виде машинных программ, которые отлаживаются, тестируются и в итоге вводятся в эксплуатацию. Для изложения дальнейшего материала напомним смысл важнейших терминов, используемых при создании программного обеспечения.

Алгоритм — точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату.

Программа — алгоритм, записанный в форме, воспринимаемой ЭВМ.

Программное обеспечение — совокупность программ и документов на них для реализации целей и задач на ЭВМ.

Программное изделие — программа на носителе данных, являющаяся продуктом промышленного производства.

Подпрограмма — часть программы, выделяемая в целях сокращения записи программы и выполняющая четко определяемую функцию.

Программный сегмент — часть программы, которая может выполняться, не требуя нахождения в оперативной памяти остальных модулей.

Программный модуль — часть программы, допускающая автономную трансляцию. Изменения, вносимые в модуль, или его замена на другой модуль, выполняющий те же функции, не влияют на остальную часть программы.

Программное обеспечение делится на системное и прикладное (специальное) ПО. *Прикладное ПО* (ППО) реализует алгоритмы решения конкретных прикладных задач, на реализацию которых проблемно ориентированы микроЭВМ и МПУ, работающие в составе РЭС. Системное ПО (СПО) предназначено для организации функционирования МПУ и микроЭВМ, планирования и управления процессом реализации прикладных программ, распределения имеющихся ресурсов, ввода—вывода и управления данными, организации взаимодействия с пользователем и др. При проектировании микроЭВМ, МПУ и МПС

обязательным является принцип совместного использования аппаратных средств и СПО. При этом СПО рассматривается как неотъемлемая часть микроЭВМ, которая поставляется одновременно с аппаратными средствами.

Общая характеристика СПО микроЭВМ и МПУ. Под СПО микроЭВМ, МПУ, МПС (в дальнейшем — только микроЭВМ) понимают комплекс программ, выполняющий функции:

- управления вычислительным процессом в микроЭВМ (соответствующая компонента СПО называется операционной системой — ОС);
- автоматизации разработки прикладного обеспечения микроЭВМ;
- организации взаимодействия с пользователем.

Особенностью СПО микроЭВМ является то, что большинство микроЭВМ (так же как МПУ и МПС) используется в различных системах управления, поэтому, как правило, к аппаратным ресурсам микроЭВМ предъявляются жесткие требования с точки зрения минимизации габаритных размеров, объема, стоимости и на СПО микроЭВМ расходуется только необходимый минимум аппаратных ресурсов. Кроме того, при включении микроЭВМ в состав РЭС она выполняет чаще всего один и тот же фиксированный набор программ и для разработки новых программ не применяется. Но циклическая реализация одного и того же набора готовых программ требует меньшего объема аппаратных ресурсов и значительно меньшей динамичности управления этими ресурсами по сравнению с реализацией переменного набора программ, поступающих из внешнего источника. Поэтому управление микроЭВМ, работающей в составе РЭС, сравнительно несложно. Кроме того, поскольку программы, реализуемые микроЭВМ при работе в составе РЭС, полностью готовы к исполнению (оформление в виде загрузочных модулей), никаких средств автоматизации программирования в СПО включать не нужно.

Помимо работы в составе РЭС микроЭВМ можно применять для разработки и отладки новых программ.

При использовании микроЭВМ для отладки программ, реализуемых в дальнейшем на микроЭВМ с той же архитектурой, но с сокращенной комплектацией, в состав СПО входит так называемая инструментальная операционная система и все средства автоматизации разработки программ.

При использовании микроЭВМ для разработки новых программ и реализации готовых программ в мультипрограммном режиме требуется наличие сложного СПО, способного обеспечить разработку и выполнение программ.

Поэтому СПО микроЭВМ содержит следующие основные компоненты:

- исполнительную ОС, служащую для управления выполнением готовых программ при работе микроЭВМ в составе РЭС;

- инструментальную ОС, создающую среду для реализации средств автоматизации программирования;

общefункциональную ОС, обеспечивающую одновременное выполнение готовых и разработку новых программ; средства автоматизации программирования.

На основании изложенного можно сделать вывод, что микроЭВМ, работающие в составе РЭС, состоят из аппаратных средств и исполнительной ОС, а микроЭВМ, служащие для одновременной разработки программ и выполнения готовых программ, — из аппаратных средств, общefункциональной ОС и средств автоматизации программирования.

Исполнительные операционные системы микроЭВМ. Наиболее распространенным и эффективным режимом работы микроЭВМ в составе РЭС является мультипрограммный режим. В этом случае в оперативной памяти одновременно находятся несколько программ, совместно использующих процессор микроЭВМ так, что когда одна программа не может выполняться на процессоре (например, вследствие ожидания поступления информации), процессор передается другой программе. Однако при этом программа должна предварительно получить все другие ресурсы микроЭВМ, необходимые для ее реализации. Таким образом, мультипрограммный режим работы микроЭВМ основывается на динамическом перераспределении ресурсов микроЭВМ между несколькими программами.

Использование мультипрограммирования основывается на применении современных аппаратных средств микроЭВМ (каналов ввода — вывода, контроллеров прямого доступа к памяти и т. д.), позволяющих совмещать работу процессора с операциями обмена между оперативной памятью и внешними устройствами.

В процессе функционирования от управляемого РЭС в случайные моменты времени на вход микроЭВМ поступают разнообразные сигналы, требующие выполнения различных автономных программ. Поэтому для обеспечения требуемого быстродействия прикладные программы следует оформлять в виде автономных задач, конкурирующих за ресурсы микроЭВМ в соответствии со своими приоритетами. Таким образом, при работе микроЭВМ в составе РЭС мультипрограммирование необходимо для организации параллельного выполнения программ, которые должны кооперироваться в ходе своего выполнения.

Исполнительные операционные системы должны обеспечивать следующие основные функции: распределение времени процессора в соответствии с заданными приоритетами задач; управление функционированием фиксированного (обычно ограниченного) набора периферийных устройств; обеспечение простейшего языкового интерфейса, который включает в себя директивы для взаимодействия задач друг с другом и несколько команд оператора (в основном команды запуска-завершения задач и команды для получения информации о работе РЭС).

Современные исполнительные ОС подразделяются на резидентные, целиком размещаемые в оперативной памяти (обычно от 2,5 до 20 К байт), и дисковые, размещаемые на магнитном диске, с которого

их соответствующие компоненты вызываются в оперативную память.

Рассмотрим функции исполнительных ОС по управлению различными ресурсами: процессором, оперативной памятью, внешними устройствами и данными.

Управление процессором заключается в анализе фиксированных или расчете динамических приоритетов задач и предоставлении процессора задаче с наибольшим текущим значением приоритета.

В исполнительных ОС используется статический и динамический методы *управления оперативной памятью*. При статическом методе распределение оперативной памяти между задачами фиксируется до начала выполнения прикладных программ и в дальнейшем не изменяется. При динамическом методе оперативная память постоянно перераспределяется между задачами в соответствии с запросами от них.

Одним из наиболее эффективных методов использования оперативной памяти является режим виртуальной памяти. Под виртуальной памятью понимается совокупность оперативной памяти и памяти на магнитных дисках, которая, с точки зрения программиста, рассматривается как память, допускающая произвольный доступ к любой ее ячейке. В основе организации виртуальной памяти лежит принцип динамического обмена информацией между внешней и оперативной памятью.

При управлении внешними устройствами (ВУ) центральной задачей является организация обмена информацией между ОЗУ и ВУ. Это связано с тем, что программирование операции ввода—вывода, осуществляемое на машинно-ориентированном языке, крайне трудоемко, поскольку для этого требуется детальное знание архитектуры ЭВМ и внешнего устройства. С другой стороны, программы управления обменом можно сделать унифицированными и не зависящими от прикладных программ. Отсюда и возникла необходимость включения в исполнительную ОС программ организации обмена ОЗУ — ВУ.

При управлении данными центральной проблемой является обеспечение языкового интерфейса для прикладных программ. Это объясняется тем, что способы записи и хранения информации во внешних запоминающих устройствах (ВЗУ) определяются физическими принципами работы этих устройств, в силу чего использование машинных адресов данных, хранимых в ВЗУ, практически невозможно. Кроме того, поскольку массивы данных, хранимые в ВЗУ, имеют обычно большой объем, прикладной программист осуществляет структурирование данных определенным образом, т. е. упорядочивает записи массивов относительно друг друга в соответствии с алгоритмами их обработки.

Структурирование данных осуществляется введением соответствующего их описания в прикладную программу. Это описание называют логической структурой данных. Естественно, что логическая структура данных, как правило, не соответствует способу их реального хранения в ВЗУ, называемому физической структурой данных.

Задача языкового интерфейса при управлении данными и заключается в отображении логической структуры данных в физическую.

Языковый интерфейс может обеспечивать следующие уровни символического обращения к данным. При организации массивов данных для каждой конкретной задачи создаются файловые системы, основное назначение которых — работа с данными на уровне наборов данных (файлов) с обеспечением простейшей структуризации наборов. При объединении всех наборов данных в единую базу данных возникают трудности отображения разных логических структур данных в одну и ту же физическую структуру базы данных. В этом случае в состав исполнительной ОС вводится система управления базой данных (СУБД).

В настоящее время широко применяются такие исполнительные операционные системы, как базовая резидентная система реального времени (БРС РВ), мультипрограммная операционная система реального времени (МОС РВ), мультимикропроцессорная операционная система реального времени со специализацией функций процессоров (ОС СФП и ОС СФП 1810). Все эти исполнительные

системы совместимы между собой снизу вверх, что обеспечивает преемственность прикладных систем реального времени.

Дадим краткую характеристику исполнительных операционных систем.

БРС РВ — простейшая ОС, требующая минимального объема памяти.

МОС РВ — более развитая система, которая обеспечивает ведение файлов на гибких магнитных дисках в реальном времени. В МОС РВ присутствует также специальный отладчик, пригодный для отладки систем на базе МОС РВ. Состав МОС РВ приведен на рис. 3.1. Ядро МОС РВ, управляющее функционированием системы, выполняет диспетчеризацию и синхронизацию вычислительных процессов, межзадачный обмен информацией, обработку прерываний и организацию работы в реальном времени.

На следующем уровне находятся системные задачи, драйверы, эмулятор терминала, файловая система, отладчик и распределитель памяти.

Верхний уровень — прикладные задачи пользователя. Следует отметить, что для ядра не существует отличий между этими уровнями, так как они с точки зрения ядра представляют собой задачи, обращающиеся к ядру во время работы.

МОС РВ является открытой системой. Имеются эффективные средства для написания пользователями собственных драйверов или дру-

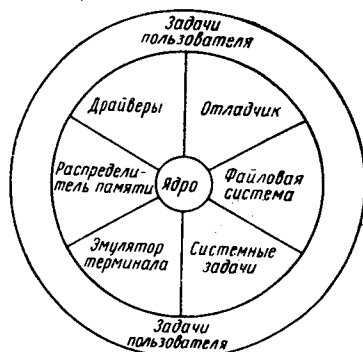


Рис. 3.1. Состав МОС РВ

гих нестандартных модулей. Возможна замена любых компонент и добавление новых системных задач.

Развитием МОС РВ для реализации мультимикропроцессорного режима являются ОС СФП и ОС СФП 1810.

Инструментальные операционные системы. Эти системы предназначены для разработки программ на той же микроЭВМ, но расширенной комплектации, где впоследствии будут выполняться готовые программы. В эту комплектацию обязательно входят магнитные диски, на которых хранятся библиотеки разрабатываемых программ, средства автоматизации программирования и большая часть программ самой инструментальной ОС.

Так как инструментальные системы предназначены для автоматизации процесса проектирования программ, то они включают эффективные трансляторы и гибкие редакторы текстов. Помимо диалогового поддерживается также пакетный режим.

К специфическим особенностям инструментальных ОС относятся: обработка прерываний, происходящих при работе средств динамической отладки программ, средств вывода содержимого памяти (дампа памяти) и трассировки; развитые средства управления данными, что необходимо для эффективного ведения библиотеки поддержки разработки; развитые средства языкового интерфейса в части команд оператора.

В число инструментальных систем входят такие ОС, как система программного обеспечения (СПО), операционные системы общего назначения (ОС 1800, МИКРОС-86), дисковые операционные системы для подготовки программ реального времени (ДОС 1800, ДОС 1810).

СПО — система программного обеспечения, ориентированная на разработку простейших программ (в пределах нескольких тысяч строк исходного текста). Характеризуется простотой и малым объемом резидентной части (монитор-отладчик MONID), записанной в системном ПЗУ. Языковая поддержка минимальная — Ассемблер (абсолютный), ПЛ/М (абсолютный) и интерпретатор БЭЙСИК. Режим работы — диалоговый, однопользовательский. СПО ориентирована на работу с гибкими магнитными дисками.

ОС 1800 — однопрограммная, однопользовательская диалоговая операционная система. Имеется также пакетный режим работы. ОС 1800 снабжена развитыми языковыми трансляторами — Макроассемблер, ПЛ/М, БЭЙСИК, ФОРТРАН, МИБОЛ, ПАСКАЛЬ. Кроме того, для близкой к ней ОС СР/М известно очень большое количество языковых средств (ЛИСП, ПЛ/1 и т. д.), которые также могут быть использованы. ОС 1800 как инструментальная система ориентирована на разработку программ общим объемом до нескольких десятков тысяч строк исходного текста. Под управлением ОС 1800 работает значительное число наиболее распространенных пакетов прикладных программ для 8-разрядных микроЭВМ. Состав ОС 1800 приведен на рис. 3.2.

МИКРОС-86 во многом повторяет ОС 1800, но эта операционная система ориентирована на 16-разрядные микроЭВМ. Однако в данных системах не предусмотрена поддержка разработки программ реального времени.

При разработке программ, работающих в реальном времени, важное значение приобретают вопросы надежности программ и эффективности процесса разработки. С учетом возрастающих объемов систем реального времени (до сотни тысяч строк исходного текста в случае 8-разрядных микроЭВМ) потребовалось создание дисковой операционной системы ДОС 1800.

ДОС 1810 является аналогом ДОС 1800, расширенной средствами поддержки 16-разрядного микропроцессора K1810BM86.

ДОС 1800 — однопользовательская, однопрограммная инструментальная операционная система, работающая в диалоговом и пакетном режимах. Основная область применения ДОС 1800 — создание программ реального времени, в том числе работающих под управлением МОС РВ и ОС СФП. Структура процесса подготовки программ показана на рис. 3.3.

Все трансляторы ДОС 1800 (ПЛ/М-80, ФОРТРАН-80, Макроасемблер) могут генерировать перемещаемые объектные модули. Эти модули с помощью компоновщика LINK объединяются с компонентами операционных систем МОС РВ или ОС СФП, которые хранятся в библиотеке объектных модулей. На этом этапе возможно также присоединение к программе перемещаемых объектных модулей, полученных ранее с помощью различных трансляторов и хранящихся в библиотеке объектных модулей. На втором этапе происходит настройка перемещаемого объектного модуля в формат абсолютного загрузочного модуля с помощью настройщика на адрес LOCATE. Абсолютный загрузочный модуль с помощью монитора выводится на носитель информации (перфоленту или дискету) и может быть использован как исполнительная ОС.

Средства автоматизации программирования. Эти средства включают в себя самонастраивающий загрузчик, ассемблер, один или несколько компиляторов или интерпретаторов, редактор связей, загрузчик, системные макрокоманды и подпрограммы. Общий вид процесса трансляции и выполнения программы показан на рис. 3.4. *Резидентный монитор* является ядром операционной системы, которое воспринимает приказы пользователя и управляет выполнением программ, необходимых для исполнения приказов.

Ассемблер представляет собой программу, которая транслирует язык ассемблера на машинный язык, а *компилятор* транслирует язык высокого уровня (ФОРТРАН, ПЛ/М и др.) на машинный язык. Время, в течение которого ассемблер или компилятор обрабатывает программу, называется соответственно временем ассемблирования или компилирования. Вход для одного выполнения ассемблера или компилятора называется *входным (исходным) модулем*, а результат их работы — *объектным модулем*. Несколько объектных модулей могут

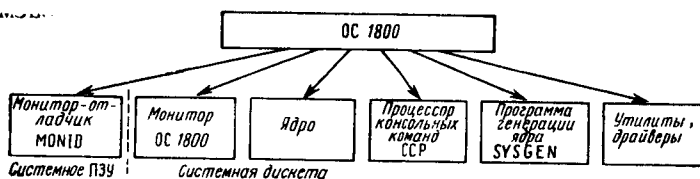


Рис. 3.2. Состав ОС 1800

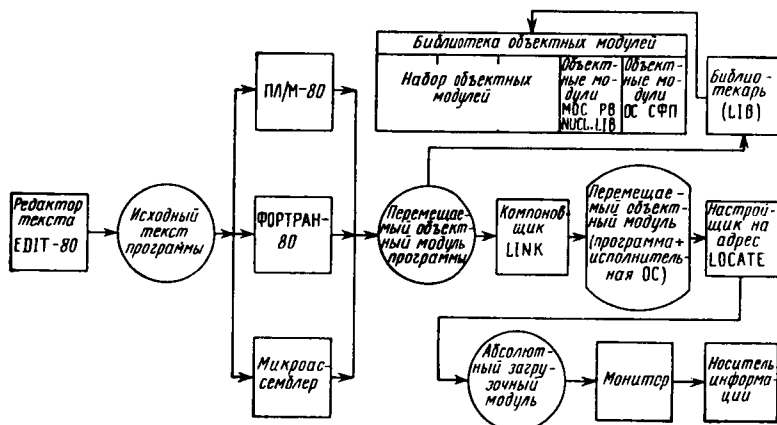


Рис. 3.3. Структура процесса подготовки программ

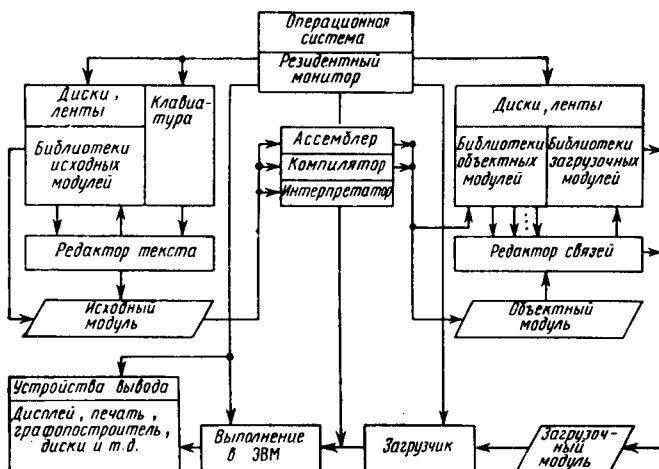


Рис. 3.4. Процесс трансляции и выполнения программы

объединяться в программу с помощью программы, называемой *редактором связей*. Объектные модули, порождаемые на одной и той же системе, устроены так, что язык, на котором они составлены, не играет роли и редактор связей всегда может связать их вместе. Наиболее часто используемые объектные модули хранятся в массовой памяти, где образуют *библиотеки*.

Выход редактора связей называется *загрузочным модулем*. Если он хранится в массовой памяти, то для загрузки его в память и подготовки к выполнению используется еще одна программа, называемая *загрузчиком*.

Практически основой технологического процесса разработки программ является совокупность библиотек, обеспечивающих долговременное хранение на магнитных дисках исходных, объектных и загрузочных модулей и их автоматический вызов по именам. Это хранение обеспечивается на основе файловых систем, входящих в состав инструментальных ОС.

Способы разработки программного обеспечения. Наиболее распространенными являются следующие подходы к процессу разработки ПО: *восходящее проектирование*, *нисходящее проектирование* и *структурное проектирование*.

Восходящее проектирование. В основе этого подхода к проектированию ПО лежит идея выделения достаточно крупных модулей, реализующих определенные функции в структуре ПО.

Каждый из выделенных модулей автономно программируется, отлаживается и проверяется. Модули характеризуются структурами данных, реализуемыми функциями и интерфейсом по входу и выходу.

Объединение модулей в программу или систему производится отдельно разрабатываемой управляющей программой, определяющей последовательность вызовов модулей, дополненных подпрограммами ввода—вывода, контроля данных и т. п. В процессе отладки проверяется информационная совместимость модулей, соблюдение распределения памяти, достоверность функционирования ПО.

Основным недостатком такого подхода является сложность процесса объединения отдельных модулей в единую систему, непредсказуемость этого процесса и трудности исправления ошибок, допущенных на ранних стадиях разработки ПО.

Нисходящее проектирование. Нисходящее проектирование основано на идее введения уровней абстракции при анализе решаемой задачи и их отображении на уровни модулей в иерархической структуре программы, т. е. декомпозиции решаемой задачи на ряд связанных между собой подзадач.

На эффективность процесса нисходящего проектирования в значительной мере влияют квалификация и его понимание задачи при выполнении последовательных декомпозиций.

Преимуществами нисходящего проектирования являются: отсутствие комплексной отладки программ, которая при восходящем проектировании занимает 25...35% общего времени разработки програм-

мы; возможность проверки и отладки управления в программе в самом начале ее разработки, что позволяет избежать трудно исправляемые ранние ошибки; возможность одновременного выполнения работ несколькими программистами, как, впрочем, и при восходящем проектировании, но с более тесным взаимодействием между ними.

Недостатками нисходящего проектирования являются неполная предсказуемость реализации компонентов нижних уровней (например, обработать входные сигналы в реальном времени или обеспечить требуемую точность), а также неполная осведомленность совместно работающих программистов при реализации одних и тех же функций и вычислений, которые могли бы быть реализованными одной подпрограммой.

Структурное программирование. Структурное программирование является воплощением системного подхода в процессе создания и эксплуатации ПО. В основу структурного программирования положены следующие принципы:

1. Программа должна состояться мелкими шагами. Размер шага определяется количеством решений, применяемых программистами на этом шаге.

2. Сложная задача должна разбиваться на достаточно простые, легко воспринимаемые части, каждая из которых имеет только один вход и один выход.

3. Логика программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

Фундаментом структурного программирования является доказанная Бемом и Джекопини *теорема о структурировании*. Эта теорема устанавливает, что как бы сложна ни была задача, блок-схема соответствующей программы всегда может быть представлена с использованием ограниченного числа элементарных управляющих структур, например: `if p THEN g; IF p THEN f ELSE; WHILE p DO f`.

Эти элементарные структуры могут соединяться между собой, образуя более сложные структуры, по тем же самым элементарным схемам. При этом *f* и *g* могут являть собой довольно сложные блок-схемы с одним входом и одним выходом, построенные из таких же элементарных структур. Следует подчеркнуть, что в структурном программировании можно обойтись без произвольных передач управления, т. е. без использования операторов `GO TO` и `RETURN`.

Структурное программирование прививает программисту процесс организации мышления таким путем, который приводит за разумное время к понятному описанию задачи для ЭВМ.

Языки программирования. Языки программирования для микроЭВМ можно разделить на три основных уровня: машинные, ассемблера и алгоритмические языки высокого уровня.

Машинный язык вычислительной машины представляет собой набор последовательностей (цепочек) битов, которые воспринимаются машиной как команды. Действия, выполняемые каждой машинной командой, определяются логикой работы ЭВМ. При создании прикладных

программ функционирования МПС на машинном языке все элементы программы (коды операций, адреса, данные) представляются в двоичной форме или в эквивалентных, но более удобных восьмеричной и шестнадцатеричной формах. Однако при программировании на машинном языке возникают определенные трудности:

необходимо помнить коды операций многочисленных команд, входящих в систему команд микропроцессора;

трудно следить за абсолютными адресами памяти, особенно в длинных программах с большим числом условных переходов;

сложно модифицировать составленную программу, в частности введение одной пропущенной команды может потребовать изменения операндов многих команд;

трудно разбираться в программе, состоящей только из числовой информации. Для документирования каждую команду следует сопровождать подробным комментарием, так как содержательный смысл (семантика) команд завуалирован в числах. Кроме того, работа исключительно с числами быстро утомляет программиста, что приводит к появлению в программе ошибок.

Достоинствами программирования на машинном языке является то, что программа оказывается наиболее эффективной, если она составлена опытным программистом. Поскольку программист максимально близок к аппаратным средствам, знает особенности микропроцессора и каждой команды, может выбрать наилучший режим адресации, программа оказывается короткой и быстродействующей. Программа работы большинства МПУ материализуется в программном ПЗУ, и ее длина влияет на экономические показатели системы, особенно в условиях массового производства.

Алгоритмические языки высокого уровня (ФОРТРАН, БЭЙСИК, ПЛ/М, ПАСКАЛЬ, АДА и др.) обеспечивают естественную форму описания вычислительных процессов; они просты и удобны в программировании, но не всегда позволяют в полной мере использовать технические возможности ЭВМ, а результирующие машинные программы, получаемые после трансляции программ с алгоритмических языков, обычно неэффективны с точки зрения объема или быстродействия. Для использования языков высокого уровня необходимы трансляторы, представляющие собой сложные программные комплексы и требующие для своей разработки большого труда.

Языки ассемблера занимают промежуточное положение между машинными языками и языками высокого уровня, будучи при этом гораздо ближе к машинным языкам. Свое название языки ассемблера получили от имени программы, преобразующей программу, написанную на таком языке, в машинные коды.

§ 3.2. ЯЗЫК АССЕМБЛЕРА

Характеристика языка ассемблера. Язык ассемблера, называемый также языком символического кодирования, допускает представление всех элементов программы в символической (буквенно-цифровой) фор-

ме, отражающей их содержательный смысл. Преобразование символических наименований в двоичные коды машинного языка возлагается на специальную программу, называемую ассемблирующей программой или ассемблером. Программа-ассемблер действует как «сборщик», объединяя элементы программы в конкретные машинные команды.

Язык ассемблера изоморфен машинному языку, т. е. каждой команде (оператору) языка ассемблера соответствует, как правило, одна машинная команда. Это позволяет программисту, составляющему программы на языке ассемблера, воспользоваться всеми возможностями системы команд микропроцессора. Именно поэтому язык ассемблера наиболее широко применяются при программировании больших задач, решаемых в МПС.

Язык ассемблера обеспечивает возможность использования символических имен в программе и избавляет программиста от утомительной работы по распределению памяти для переменных и констант. Язык ассемблера позволяет так же гибко и полно использовать технические возможности микроЭВМ и МПС, как и машинный язык, что особенно существенно для микроЭВМ с ограниченной емкостью основной памяти. На языке ассемблера обычно программируются операционная система и система реального времени, к которым предъявляются высокие требования с точки зрения скорости выполнения программ и емкости занимаемой памяти.

Каждый язык ассемблера является машинно-зависимым языком и отражает архитектурные особенности используемого МП. В языках ассемблера имеется множество символических обозначений, причем для каждого микропроцессора некоторые символические наименования фиксированы и иногда называются ключевыми словами. К ним относятся мнемоники кодов операций, директивы ассемблера и наименования внутренних регистров. Остальные символические наименования определяет программист, придавая им содержательный смысл, согласующийся с контекстом прикладной программы. Числовые данные в языке ассемблера допускается представлять в двоичной, десятичной, восьмеричной и шестнадцатеричной системах счисления.

В качестве алфавита допустимых символов в зарубежных ассемблерах принят код ASCII (американский стандартный код для обмена информацией), содержащий латинские буквы (в языке ассемблера — только прописные), цифры, разделители (двоеточие, точка с запятой, запятая), знаки математических операций и специальные символы, например CR — возврат каретки, LF — перевод строки. Каждый символ представляется семью битами (к которым иногда добавляется восьмой бит паритета) и хранится в одном байте памяти, а длина символических наименований, состоящих из букв и цифр, ограничена 5...8 символами, первым из которых обязательно должна быть буква.

Процесс разработки программы на языке ассемблера представлен на рис. 3.5. Обычно программист использует *редактор текста*, который формирует текстовый файл, содержащий программу на языке ассемблера. Названия *исходный файл* и *исходная программа* часто

употребляют для текстовых файлов, представленных на языке ассемблера и языках программирования высокого уровня. Ассемблер воспринимает исходную программу в качестве входной информации, проверяет корректность форматов и формирует объектный модуль, содержащий программу на машинном языке. *Загрузчик* производит загрузку объектного модуля в память ЭВМ. Впоследствии на этой машине выполняется прогон программы, иногда с помощью *отладчика*.

Поля ассемблерной строки. Программа на языке ассемблера представляет собой последовательность операторов, каждый из которых записывается в одну строку. Оператор языка ассемблера обычно имеет четыре поля: метки, кода операции, операндов и комментария. Из этих полей обязательно лишь поле кода операции, остальные поля могут отсутствовать.

Поле метки (LABEL) предназначено для записи символического имени данного оператора. Метка — обычный идентификатор (мнемоническое обозначение в языке ассемблера), представляющий собой последовательность букв и цифр, начинающуюся с буквы. Максимальная длина поля метки различна для разных ассемблеров и, как правило, не превышает восьми знаков. Метка не должна содержать знаков пунктуации и пробелов. За последним символом метки следует двоеточие. В качестве меток не допускаются использовать ключевые слова.

Примеры допустимых меток: AGAIN:, A15:, DONE:, C5FA:.

Поскольку метки ассоциируются с адресами памяти, их нельзя дублировать. Символическое имя требуется в тех случаях, когда на данный оператор есть ссылка в программе хотя бы в каком-нибудь другом операторе. При определении каждому мнемоническому обозначению в программе на языке ассемблера присваивается значение. Программа ассемблера следит за метками и их значениями с помощью внутренней таблицы символов или имен. Каждая метка может быть определена только один раз, но ссылка на нее может организовываться столько раз, сколько это необходимо.

Поле кода операции (OPCODE) содержит мнемоническое обозначение либо машиной команды, либо псевдооперации. Число знаков в мнемонической записи может варьироваться в пределах от одной буквы до некоторого максимального числа букв, определяемого конкретным языком. Для операторов машинных команд мнемонадом операции является мнемонакод соответствующей машинной команды. Большинство мнемоник представляют собой аббревиатуры предложений, характеризующих основные функции команд, например:

MOV (MOVE) — передать, переслать;

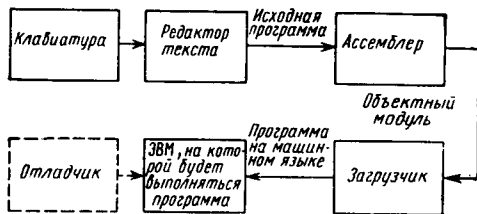


Рис. 3.5. Этапы разработки программы на языке ассемблера

ACI (Add with Carry Immediate) — сложение с переносом непосредственное;

JNZ (Jump if Non Zero) — перейти, если не ноль;

XCHG (eXCHanGe) — обменять.

Некоторые мнемоники явно определяют функцию команды, например команда PCHL означает передачу содержимого регистров (H, L) в программный счетчик PC, а команда PUSH выполняет загрузку в стек (в системе команд МП КР580).

Обычно длина поля мнемоники не превышает четырех позиций, а между ним и соседним справа полем операнда необходим минимум один пробел. Запоминать и использовать мнемоники значительно проще, чем численные коды команд. Мнемоники кодов операций являются ключевыми словами ассемблера, и если содержимое этого поля не входит в множество допустимых мнемоник, ассемблер выдает сообщение о недействительной команде.

Поле операнда (OPERAND) в зависимости от содержимого поля кода операции либо указывает несколько операндов, разделенных запятыми, либо не содержит ни одного операнда. Содержимое поля операнда интерпретируется в соответствии с функцией команды. В качестве операндов могут фигурировать адреса памяти, внутренние регистры МП, адреса портов ввода—вывода, числовые и символьные константы.

Проиллюстрируем способы задания операндов в стандартном языке ассемблера МП КР580.

Числовые данные могут задаваться в шестнадцатеричном коде, при этом число должно начинаться с цифр 0...9 и завершаться буквой H (Hex); в десятичном коде число необязательно заканчивается буквой D (Decimal); в восьмеричном коде число заканчивается буквой O (Octal) или Q, а в двоичном коде — буквой B (Binary). Примеры задания числовых данных:

Метка	Код	Операнд	Комментарий
	STA	8000 H ;	Запоминание в ячейке 8000
	MVI	B, 15 ;	Загрузка в регистр B кода 00001111
	IN	32 D ;	Ввод из порта с адресом 00100000
	SUI	IQ ;	Декремент аккумулятора
	OUT	1111 B ;	Вывод в порт с адресом 15

Помимо числовых данных в поле операнда могут задаваться символьные константы; идентификаторы внутренних регистров B, C, D, E, H, L, M, A с соответствующими двоичными значениями от 000 до 111; в командах, оперирующих 16-битными значениями, применяются идентификаторы внутренних 16-битных регистров B, D, H, PSW, SP; метки, используемые для реализации команд передачи управления;

текущее содержимое программного счетчика, необходимое для организации относительной адресации. Наиболее сложными конструкциями поля операнда являются *выражения*, содержащие в качестве аргументов все рассмотренные выше типы данных, которые связываются арифметическими и логическими операторами.

Поле комментария (COMMENTS), начинающееся с некоторого разделителя (точка с запятой или наклонная черта), полностью игнорируется ассемблером, поэтому в нем можно помещать любой текст. Содержание этого поля должно пояснять действия, которые выполняет команда в контексте конкретной прикладной программы.

Для иллюстрации программирования на языке ассемблера МП КР580 рассмотрим методику составления программ сложения и умножения двух 8-разрядных чисел.

Пример 3.1. Рассмотрим программу сложения двух чисел. Предположим, что нужно сложить содержимое ячеек 40_{16} , 41_{16} и поместить результат в ячейку 42_{16} (обозначение A_{16} поясняет, что число A представлено в шестнадцатеричной системе счисления). В этом примере для простоты не будем учитывать возможность возникновения переноса.

Процедура сложения выполняется за несколько шагов:

1. Послать в аккумулятор содержимое ячейки 40_{16} .
2. Прибавить к содержимому аккумулятора содержимое ячейки 41_{16} .
3. Послать содержимое аккумулятора в ячейку 42_{16} .

Программа на языке ассемблера выглядит таким образом:

Метка	Код	Операнд	Комментарий
	LDA	40 H	; Загрузка операнда 1
	MOV	B, A	; Пересылка содержимого аккумулятора в регистр В
	LDA	41 H	; Загрузка операнда 2 в аккумулятор
	ADD	B	; Сложение содержимого аккумулятора с содержимым В
	STA	42 H	; Запоминание результата в ячейке 42
	HLT		

Рассмотрим выполнение программы по шагам.

1. LDA 40 H . По этой команде содержимое ячейки памяти с адресом 40 (адрес представлен в шестнадцатеричной системе счисления) посылается в аккумулятор.

2. MOV B, A. По этой команде в регистр В посылается содержимое аккумулятора.

3. ADD B. По этой команде к содержимому аккумулятора прибавляется содержимое регистра В.

4. STA 42 H . По этой команде содержимое аккумулятора посылается в ячейку 42.

5. Команда HLT является командой останова.

Пример 3.2. Рассмотрим программу умножения 8-битных целых чисел. Процедура умножения реализуется младшими разрядами вперед со сдвигом частичной суммы. Имя программы — MUL 88. Предполагается следующее распределение внутренних регистров: множитель находится в регистре C, множимое — в регистре D, произведение образуется в регистрах (B, C), а регистр E выступает в качестве счетчика бит множителя.

Программа MUL 88 имеет вид:

Метка	Код	Операнд	Комментарий
MUL 88:	MVI	B, 0 ;	Сброс старшего байта произведения
	MVI	E, 8;	Инициализация
XBIT	MOV	A, C ;	Множитель — в аккумулятор
	RAR		Очередной бит во флажке C
	MOV	C, A ;	Возвращение множителя
	DCR	E ;	Декремент счетчика бит
	RM		Умножение закончено?
	MOV	A, B ;	Старший бит произведения
	INC	NOADD ;	Бит множителя равен нулю
	ADD		Суммирование множимого
NOADD	RAR		Сдвиг частичной суммы
	MOV	B, A ;	Возвращение старшего байта
	JMP	NXB IT ;	Умножение на следующий бит

В начале программы производится инициализация регистра B, в котором будут образованы старшие 8 бит произведения, и счетчика E. После этого множитель передается в аккумулятор и командой RAR очередной анализируемый бит множителя передается во флажок C. Если бит равен единице, то командой ADD осуществляется сложение регистров B и D, а если он равен нулю, сложение не производится. В данной программе младшие биты произведения замещают освобождающиеся старшие биты множителя.

Команды управления ассемблером предназначены для управления режимом и формой выдачи листинга, обеспечения выбора внешних устройств для ввода исходной программы и вывода результатов трансляции, осуществления приостановки и продолжения трансляции, а также для выполнения ряда других вспомогательных операций.

Если определенная последовательность команд часто встречается в программах, то оказывается удобным обозначать ее символическим именем. Такое средство ассемблера, которое позволяет подставить вместо символического имени соответствующую последовательность

команд, называется *макросредством*, а сама совокупность команд с присвоенным символическим именем — *макрокомандой*. Ассемблер автоматически заменяет каждую ссылку на макрокоманду последовательностью команд, взятой из макроопределения.

§ 3.3. СТРУКТУРА ДАННЫХ

Методы программирования в обязательном порядке должны учитывать методы структурирования данных. Программы представляют собой конкретные формулировки алгоритмов, основанные на конкретных представлениях и структурах данных. Другими словами, при разработке МПС следует учитывать, что структура программ функционирования МПС неразрывно связана со структурами данных.

При организации структур данных необходимо помнить следующее. В математике при классификации данных проводится строгое разграничение между вещественными, комплексными и логическими переменными, между целыми и дробными числами, между функциями, функционалами и множествами функций и т. д. При обработке данных средствами вычислительной техники классификация данных тем более важна. Искусством программирования является умение составлять не только последовательности операций, но также структуры данных.

Элементы данных обычно разделяют на две основные группы: *скаляры* и *структуры*. К скалярам относятся коды, флажки, числа и слова. Под структурами данных понимают наборы организованных некоторым образом скалярных данных. К структурам относятся массивы, очереди, стеки, файлы, таблицы и др. Характер организации структур может быть самым разнообразным. Выбор структуры данных влияет как на требуемую емкость памяти, так и на производительность решения задач в микропроцессорной системе.

Рассмотрим основные структуры данных.

Простые типы данных. Во многих программах целые числа используются в том случае, когда их собственно числовое значение несущественно и когда целое число указывает на выбор значения из небольшого множества возможных вариантов. В подобных случаях вводится новый, простой, неструктурированный тип T и перечисляется множество всех его возможных значений $C_1, C_2, \dots, C_n$:

$\text{type } T = (C_1, C_2, \dots, C_n).$

Кардинальное число T есть $\text{card}(T) = n$.

Примеры:

$\text{type фигура} = (\text{прямоугольник}, \text{квадрат}, \text{эллипс}, \text{круг})$

$\text{type Boolean} = (\text{false}, \text{true}).$

Стандартные простые типы данных. Стандартные простые типы — это типы, которые являются встроенными для большинства ЭВМ. Они

включают целые числа, логические значения и множество символов печати. Стандартные простые типы обозначаются идентификаторами:

integer, real, Boolean, char.

Тип **integer** содержит подмножество целых чисел. Предполагается, что все действия с данными этого типа являются точными и выполняются по обычным правилам арифметики.

Тип **real** обозначает подмножество вещественных чисел. Для арифметических действий со значениями типа **real** допускается неточность в пределах ошибок округления, так как в вычислениях участвует конечное число цифр.

Два значения стандартного типа **Boolean** (булевское) обозначаются идентификаторами **true** (истина) и **false** (ложь).

Стандартный тип **char** включает множество печатных символов. Тип **char** содержит 26 латинских букв, 10 арабских цифр и некоторое количество других графических символов, таких, как знаки препинания.

Ограниченные типы. Если переменная принимает значения некоторого типа только в определенном интервале, то это можно выразить, определив переменную ограниченного типа, который описывается так:

type T = min ... max,

где **min** и **max** — границы интервала.

Примеры:

type год = 1900 ... 1999

type цифра = 0 ... 9.

Массивы. Под *массивом* понимают многомерную структуру данных, в которой элементы упорядочены таким образом, что их описание однозначно определяет положение каждого элемента или путь доступа к нему.

Массив — регулярная структура со случайным доступом, все его компоненты могут выбираться произвольно и являются одинаково доступными. Для обозначения отдельной компоненты к имени всего массива добавляется так называемый индекс, позволяющий выбрать компоненту. Описание регулярного типа **T** задает базовый тип **T₀** и тип индексов **I**:

type T = array [I] of T₀

Пример:

type Row = array [1 ... 5] of real.

Конкретное значение переменной **var x : Row**, если каждая компонента удовлетворяет равенству $x_i = 2^{-i}$, имеет вид $x_1 = 0,5$; $x_2 = 0,25$; $x_3 = 0,125$; $x_4 = 0,0625$; $x_5 = 0,03125$.

Наиболее простой и распространенной структурой данных является массив (рис. 3.6). Он представляет собой набор элементов данных одинаковой длины, который размещается в области смежных ячеек памяти. Часто начальному адресу массива присваивается имя **BASE**. Число элементов в массиве называется его длиной. Положение любого элемента в массиве характеризуется его порядковым номером, называемым индексом. При этом адрес любого элемента равен сумме базового адреса **BASE** и индекса **IND**, умноженного на длину элемента в байтах.

Обработка массивов осуществляется циклическими программами, состоящими из таких частей: инициализации, обращения к текущему элементу, перехода к следующему элементу и проверки условия окончания цикла.

При организации программ обработки массивов необходимо хранить адрес текущего элемента массива и вести счет числу циклов выполнения программы с целью фиксации окончания процедуры обработки массива.

Для иллюстрации процедуры обработки одномерного массива рассмотрим программу поиска максимального числа в массиве 8-битных целых чисел. Программа представлена на языке ассемблера МП КР580. При этом адрес текущего элемента массива хранится в регистрах (**H**, **L**), называемых указателями, длина массива хранится в ячейке с адресом **LENGT**, а в качестве счетчика циклов используется регистр **B**.

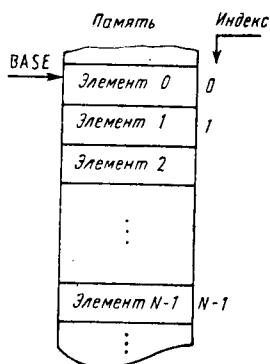


Рис. 3.6. Размещение массива в памяти

Метка	Код	Операнд	Комментарий
	LDA	LENGT ;	Инициализация счетчика
	MOV	B, A	
	LXI	H, BASE ;	Инициализация указателя
NEWMX	MOV	A, M ;	Загрузка максимума
EXT	DCR	B ;	Проверка
	JZ	DONE ;	окончания цикла
	INX	H ;	Переход к следующему элементу
	CMP	M ;	Сравнение его с максимумом
	JC	NEWMX ;	Новый максимум
	JMP	NEXT ;	Следующий элемент меньше
DONE	***		Максимум в аккумуляторе

Идея алгоритма поиска максимального элемента в массиве заключается в следующем. Сначала первый элемент массива выбирается

в качестве максимального и записывается в аккумулятор. Затем каждый следующий элемент сравнивается с ним, причем если текущий элемент больше ранее найденного максимума, он замещает его в аккумуляторе.

Более общую структуру данных представляет двумерный массив. Каждый элемент двумерного массива характеризуется двумя индексами: номером строки I и номером столбца J . Однако в памяти двумерный массив хранится в линейно упорядоченных смежных ячейках. Поэтому общий индекс IND зависит от способа размещения массива. В частности, в массиве из однобитных элементов значение IND определяется таким образом:

$IND = I \cdot n_r + J$, если массив хранится по строкам;

$IND = I + J \cdot n_c$, если массив хранится по столбцам.

Здесь n_r и n_c — соответственно число строк и столбцов в массиве, а нумерация строк и столбцов начинается с нуля. Когда длина элементов массива превышает 1 байт, значения IND следует умножить на длину элементов.

Обработка двумерных массивов осуществляется программами, содержащими два цикла. Во внешнем цикле производится инкремент I , а во внутреннем (вложенном) — инкремент J .

При организации массивов большой размерности хранение их в памяти также осуществляется линейно. Программы обработки многомерных массивов имеют соответствующее число вложенных циклов.

Записи. Самый общий метод получения составных типов — это объединение компонент, принадлежащих к произвольным, возможно, тоже составным типам, в один составной тип. Примеры из математики — комплексные числа, состоящие из двух вещественных чисел; координаты точек, состоящие из нескольких вещественных чисел. Пример из обработки данных — описание людей с помощью нескольких существенных характеристик, таких, как фамилия, имя, дата рождения, пол и семейное положение.

В математике такой составной тип называется декартовым произведением типов компонент. Это связано с тем, что множество значений составного типа состоит из всех возможных комбинаций значений, взятых по одному из каждого типа компонент.

Для обозначения совокупности данных используется слово «запись» (record). В общем виде составной тип T определяется следующим образом:

```

typ T = record
    s1:T1;
    s2:T2;
    . . . . .
    sn:Tn.
end

```

Кардинальное число T есть $\text{card}(T) = \prod_{i=1}^n \text{card}(T_i)$.

Пример:

```
type Date=record   day:1 . . . 31
                   month 1 . . 12
                   year 1 . . .2000
end
```

Множество. Кроме массива и записи имеется третья фундаментальная структура данных — *множество*. Соответствующий тип описывается таким образом:

type $T = \text{set of } T_0$.

Значениями переменной x типа T являются множества элементов типа T_0 .

Пример:

type $\text{intset} = \text{set of } 0 \dots 30$.

Здесь базовым типом является подмножество целых чисел от 0 до 30.

Кардинальное число множества типа T $\text{card}(T) = 2^{\text{card}(T_0)}$. Это следует из того, что каждый из элементов T_0 представляется в множестве T одним из двух значений: «присутствует» или «отсутствует», и эти элементы входят в множество T независимо друг от друга.

Файлы. *Последовательным файлом*, или просто *файлом*, называется последовательность записей, рассматриваемая в процессе пересылки и обработки как единое целое. Файловый тип определяется как **type** $T = \text{file of } T_0$. Это означает, что любой файл типа T состоит из 0 или более компонент типа T_0 .

Пример:

type $\text{text} = \text{file of char}$.

Доступ к компонентам файлов — строго последовательный. Смысл последовательного доступа заключается в том, что в каждый момент времени доступна лишь определенная компонента последовательности. При этом важно, что процессы формирования и просмотра последовательности не могут произвольно чередоваться. Таким образом, файл вначале строится с помощью последовательного добавления компонент (в конец), а затем может последовательно просматриваться от начала до конца. Поэтому принято считать, что файл находится в одном из двух состояний: либо формирования (записи), либо просмотра (чтения).

Файл состоит из отдельных записей (блоков), каждая из которых содержит информацию об одном объекте. Типичный файл включает в себя заголовок (метку), собственно информацию и маркер конца файла. Минимальная длина файла равна емкости одного сектора гибкого диска, а максимальная ограничена емкостью носителя.

Файлы классифицируют по виду информации, форме ее представления, а также по способу доступа.

По *виду информации* различают программные файлы и файлы данных. Следует отметить, что к одним и тем же файлам данных могут обращаться несколько программ, а одна программа может обращаться к нескольким файлам данных.

По *форме представления информации* различают текстовые и двоичные файлы. В текстовых файлах информация хранится в символьной форме и каждый символ кодируется байтом. Примером текстового файла служит прикладная программа на языке ассемблера. В двоичных файлах, примером которых может быть объектная программа, используются только логические «0» и «1».

По *способу доступа* выделяют последовательные (с последовательным доступом) и равнодоступные (с произвольным доступом) файлы. Равнодоступные файлы организуются только на дисках, последовательные — на магнитной ленте, а иногда и на диске.

Имя файла состоит из двух полей, разделенных точкой. В первом поле находится собственно имя файла, содержащее от одного до восьми символов, первым из которых должна быть буква. Второе поле, содержащее до трех символов, обычно идентифицирует форму информации в файле, например TXT обозначает символьную информацию, а BIN — двоичную.

Файлы, как правило, размещаются во ВЗУ с последовательной выборкой (в МПС и микроЭВМ — на гибких магнитных дисках).

На гибком диске кроме собственно файлов находится оглавление, представляющее собой таблицу с информацией о текущих файлах. Каждая запись в таблице содержит имя файла, дату его создания или модификации, длину и физический адрес файла. Кроме того, в оглавление обычно включается число файлов, занимаемое им пространство диска и свободное пространство.

Описанные структуры данных относятся к *статическим* структурам, поскольку заранее известен размер значений данных и соответственно размер выделяемой для них памяти. Однако многие задачи требуют более сложных структур данных. Для таких задач характерно, что используемые в них структуры данных изменяются во время выполнения. Поэтому они называются *динамическими* структурами; примерами таких структур являются рекурсивные типы данных, линейные списки (или очереди), древовидные и другие структуры данных.

§ 3.4. ЯЗЫКИ ВЫСОКОГО УРОВНЯ

Язык ассемблера, максимально близкий к аппаратным средствам и обеспечивающий эффективность прикладных программ, имеет ряд недостатков. Один из них заключается в том, что в распоряжении программиста имеются только элементарные машинные команды, поэтому программы сравнительно простых задач оказываются слишком

громоздкими. По мере усложнения прикладных программ этот недостаток проявляется более серьезно. Другой существенный недостаток — ориентация ассемблера на конкретный микропроцессор — приводит к тому, что при появлении каждого нового микропроцессора приходится заново осваивать его язык ассемблера.

Применение языков высокого уровня для программирования задач, решаемых для микроЭВМ и МПС, обусловливается такими требованиями, как повышение производительности труда программистов, упрощение сопровождения программных систем, повышение мобильности программного обеспечения и т. д.

Сроки программирования на языках высокого уровня сокращаются за счет их близости к естественному математическому языку. Считается, что каждый оператор языка высокого уровня эквивалентен 4 ... 6 операторам языка ассемблера.

Программы на языках высокого уровня обладают свойством самодокументирования, поскольку операторы этих языков почти не нуждаются в комментариях. Языки высокого уровня являются машинно-независимыми. Процедуры проверки и отладки прикладных программ на языках высокого уровня занимают гораздо меньше времени по сравнению с программами на языке ассемблера.

Использование языков высокого уровня связано со следующими негативными моментами. Во-первых, для их практического применения необходимы системы проектирования, оснащенные трансляторами с соответствующих языков. Во-вторых, порождаемые транслятором объектные программы оказываются менее эффективными, чем эквивалентные ассемблированные программы. Практика программирования показывает, что длина прикладных программ увеличивается на 20 ... 50% с соответствующим увеличением емкости программного ПЗУ.

Первый язык высокого уровня для МПС — язык ПЛ/М фирмы Intel — ориентирован на микропроцессоры 8080 и 8085 (КР580) и представляет собой сильно упрощенную версию языка ПЛ/1. В последнее время заметна ориентация на стандартные языки БЭЙСИК, ФОРТРАН, ПАСКАЛЬ, АДА и ряд других. Рассмотрим краткую характеристику языков БЭЙСИК, ПЛ/М и ПАСКАЛЬ.

1. Язык БЭЙСИК

Язык БЭЙСИК — диалоговый алгоритмический язык высокого уровня. Название языка БЭЙСИК (BASIC) возникло от сокращения английских слов Beginner's All — purpose Symbolic Instruction Code — многоцелевой язык символических инструкций для начинающих. Язык БЭЙСИК довольно близок к алгоритмическому языку ФОРТРАН. Рассмотрим описание языка применительно к ЭВМ «Электроника 60», ДВК-2, СМ-4.

Программа на языке БЭЙСИК состоит из пронумерованных строк. Одна строка содержит один или несколько операторов, разделенных символом \ (косая черта). Строки нумеруются, начиная с адреса 10,

с интервалом в 10, т. е. 10, 20, ... Номера строк используются в операторах передачи управления.

Символы языка. В языке БЭЙСИК содержатся следующие группы символов: прописные латинские буквы (A, ..., Z); прописные русские буквы (А, ..., Я); десятичные цифры (0 ... 9); знаки арифметических операций и специальные символы.

Константы. Запись чисел близка к естественной. В языке БЭЙСИК используются *целые* и *вещественные константы*. Константы выражаются десятичными числами в форматах с фиксированной или плавающей точкой:

n или $n \cdot t$ или $n \cdot tE \pm s$,

где n , t — последовательность десятичных цифр; E — признак десятичного основания; s — показатель степени (порядок). Например, число $-8,4 \cdot 10^{-4}$ на языке БЭЙСИК представляется как $-8.4E -4$ или $-.84E -3$, а число $0,15$ — в виде $0,15$ или $.15$.

Переменные. *Переменная* — величина, значение которой в процессе выполнения программ может меняться. Переменные обозначают идентификаторами, состоящими либо из одной латинской буквы, либо из латинской буквы с последующей цифрой. Например, A, A2, C. Переменные могут быть *целыми* и *вещественными*.

Массивы. Язык БЭЙСИК допускает использование *одномерных* и *двумерных* массивов, т. е. векторов и матриц. Идентификатором массива может быть одна латинская буква или латинская буква с десятичной цифрой. Элементы одномерных массивов снабжаются одним индексом, определяющим порядковый номер элемента в массиве, например A (1) — первый элемент массива A. Элементы двумерных массивов снабжаются двумя индексами, первый из которых определяет номер строки, а второй — номер столбца, на пересечении которых расположен в матрице соответствующий элемент, например M (5, J).

Перед использованием в программе массивы должны быть описаны. Для описания массивов необходим оператор DIM, например DIM M4 (5, 10).

Стандартные и определяемые функции. Язык БЭЙСИК содержит набор стандартных функций, для обозначения которых служат трехбуквенные идентификаторы (табл. 3.1). Аргумент функции заключается в круглые скобки, им может быть произвольное арифметическое выражение.

С помощью оператора

DEF FNa (параметр) = выражение

пользователь может определять в своей программе новые функции, причем a — однобуквенное окончание имени функции, назначаемое программистом, например

DEF FNE = 3 — 4 * X.

Стандартные функции

Запись на языке БЭЙСИК	Математическое определение	Запись на языке БЭЙСИК	Математическое определение
ABT (X)	$ x $	LOG (X)	$\ln x$
ATN (X)	$\arctg x$	SQN (X)	$\begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$
COS (X)	$\cos x$		
STN (X)	$\sin x$		
EXP (X)	e^x	SQR (X)	$\sqrt{x}$
INT (X)	$[x]$ — наибольшее целое, не превосходящее x	RND (X)	Датчик случайных чисел

Кроме того, пользователь может включить в БЭЙСИК-систему ту или иную программу на языке ассемблера.

Выражения. Арифметические выражения соответствуют общепринятым алгебраическим выражениям. В арифметическое выражение могут входить числа, переменные, стандартные или определяемые функции, соединенные знаками арифметических операций. Число или переменная также являются арифметическими выражениями. Формулы записываются в одну строку без каких-либо подстрочных или надстрочных индексов. Например, математическая запись выражения $(ax^2 + bx)/(c^2 - 5,7)$ на языке БЭЙСИК выглядит как

$$(A \cdot X \uparrow 2 + B \cdot X) / (C \uparrow 2 - 5.7).$$

Арифметические выражения внутри скобок выполняются-слева направо в соответствии со следующими приоритетами: вычисление значений функций; возведение в степень (знак операции $\uparrow$); умножение и деление ($\times$ и $/$); сложение и вычитание.

Операторы языка БЭЙСИК. Рассмотрим различные операторы языка.

Оператор присваивания LET служит для присваивания переменной значения арифметического выражения и имеет вид

m LET переменная = выражение.

К операторам *управления программой* относятся операторы GO TO, IF, FOR, NEXT, GOSUB.

Оператор GO TO (безусловного перехода) передает управление в то место программы, с которого следует продолжать вычисление на данном этапе. Общий вид оператора

GO TO N,

где N — номер строки.

Оператор IF (условного перехода) изменяет естественный порядок выполнения программы в случае истинности некоторого условия. Общий вид условного оператора

IF условие THEN N

или

IF условие GO TO N.

Циклы — участки программы, которые в процессе ее выполнения повторяются указанное число раз.

Операторы FOR и NEXT упрощают составление циклов и обеспечивают четкое их выделение в программе. Оператор FOR — оператор заголовка цикла — всегда предшествует повторяющейся группе операторов, составляющих так называемое тело цикла. По оператору FOR для параметра цикла вычисляются начальное и конечное значения, а также приращение параметра цикла. Операторы, составляющие тело цикла, повторяются при различных значениях параметра цикла в соответствии с оператором FOR. Заканчивается тело цикла оператором NEXT. По оператору NEXT вычисляется новое значение параметра цикла и производится анализ на завершение цикла. Общий вид операторов цикла

FOR V = A₁ TO A₂ STEP A₃

.....
NEXT V,

где V — любая неиндексированная переменная (переменная цикла); A₁, A₂, A₃ — начальное и конечное значения и шаг изменения переменной цикла V.

Оператор GOSUB служит для обращения к подпрограмме. Он не только передает управление первому оператору подпрограммы, но и запоминает то место, из которого произошло обращение.

К *операторам ввода — вывода данных* относятся операторы DATA, READ, RESTORE, INPUT, PRINT.

Операторы DATA и READ необходимы для ввода данных в программном режиме.

Оператор DATA имеет вид

DATA n₁ [, n₂, ...],

где n₁, n₂, ... — числа

Оператор DATA задает в программе числовую информацию в виде блока данных. Оператор READ извлекает числовую информацию из блока данных для присвоения значений переменным.

Оператор READ имеет вид

READ V₁ [, V₂, ...],

где V₁, V₂, ... — вводимые переменные ([] означает необязательные параметры).

Оператор RESTORE обеспечивает возможность повторного использования блока данных DATA.

Оператор INPUT служит для ввода данных пользователем с терминала по запросу программы. Общий вид оператора

INPUT $V_1[, V_2, \dots]$.

Выполняя оператор INPUT, машина печатает его текст на терминале и ожидает ввода пользователем числовых значений переменных.

Оператор PRINT используется для вывода на экран или на принтер результатов вычислений. Общий вид оператора

PRINT список

Элементами выводного списка могут быть числа, имена переменных, выражения, тексты. Тексты необходимы для печати заголовков и пояснений. Текст заключается в апострофы и может состоять из произвольной последовательности символов входного языка.

Оператор RETURN завершает действие подпрограммы, обеспечивая возврат в то место вызывающей программы, из которого произошло обращение к подпрограмме.

Оператор REM (REMARK) используется для записи комментариев. *Оператор* RANDOMIZE используется для обеспечения формирования случайных чисел с помощью функции RND.

Оператор STOP указывает логический конец программы. При этом на экране появляется сообщение

STOP AT LINE N,

где N — номер строки, в которой произошел останов. *Оператор* END указывает физический конец программы, он записывается только в конце программы.

Директивы. Для реализации языка БЭЙСИК служит интерпретирующая БЭЙСИК-система, которая преобразует программу, записанную на языке БЭЙСИК, во внутреннее представление конкретной ЭВМ и обеспечивает ее выполнение в режиме интерпретации. Системные команды интерпретирующей БЭЙСИК-системы не нумеруются и подлежат немедленному исполнению сразу после их ввода и синтаксического анализа. Каждая реализация БЭЙСИК-системы ориентирована на конкретный тип микроЭВМ или МПС и состав периферийного оборудования. Поэтому системные команды для различных систем могут отличаться по составу, форме записи и выполняемым с их помощью действиям.

2. Язык ПЛ/М

Язык ПЛ/М является подмножеством языка ПЛ/1. Он представляет пользователю достаточно мощные средства для создания эффективных программ. Язык ПЛ/М имеет блочную структуру. Описания в блоке позволяют задавать тип переменных, элементов массивов и

структур, их начальные значения, управлять распределением памяти и определять простые текстовые подстановки.

Среди выполняемых операторов языка наряду с конструкциями общего назначения (присваивания, цикла, перехода и т. д.) имеются машинно-зависимые конструкции, обеспечивающие ввод, вывод и выполнение встроенных функций. Использование этих возможностей приближает технологию программирования на языке ПЛ/М к технологии программирования на языке ассемблера. При этом ПЛ/М сохраняет для программиста удобства, предоставляемые операторами языков высокого уровня.

Программа на языке ПЛ/М — последовательность предложений, заканчивающаяся зарезервированным словом EOF (конец файла). При записи операторов используются буквы латинского и русского алфавитов, десятичные цифры, специальные символы, зарезервированные слова и так называемые предопределенные имена, или предопределенные переменные. Рассмотрим основные элементы языка.

Идентификаторы. Идентификаторы в ПЛ/М необходимы для присваивания имен переменным, процедурам, макрокомандам и меткам операторов. Идентификатор — последовательность букв и цифр, начинающаяся с буквы, длиной не более 31 символа. Примеры идентификаторов:

X Y Z10SUMMA ТАБЛИЦА ВХОДОВ

Комментарии. Комментарий — последовательность символов, ограниченная слева парой символов /*, а справа — парой символов */. Комментарии можно ставить в любом месте программы и они могут занимать любое число строк, например

/* ДАННЫЙ ТЕКСТ ЯВЛЯЕТСЯ КОММЕНТАРИЕМ */

Комментарии игнорируются компилятором ПЛ/М.

Метки. Предложение как элемент программы может быть помечено для последующих ссылок на него. В качестве меток записываются идентификаторы и числовые адреса, например,

НАЧАЛО ЦИКЛА : DO WHILE (A > 0);

Элементы данных. Любой набор знаков, над которым можно выполнять определенные действия, в программировании называют данными. Элементами данных являются константы и переменные.

Константы представляют собой величину, известную в момент компиляции программы. Константа может быть числовой или символьной.

Числовые константы — целые числа, которые могут быть записаны в любой из систем счисления: десятичной, двоичной, восьмеричной и шестнадцатеричной. Основание системы счисления указывается посредством букв В, О (или Q), D, H, записываемых за числом, например 1001 1101 В (двоичная константа), 33 Q (восьмеричная констан-

та); 65 D или 65 (десятичная константа); 0B 3H, 77AH (шестнадцатеричные константы).

Символьные константы или *строки* записываются в виде последовательности символов языка ПЛ/М, заключенной в апострофы, например 'AB', 'P1', '—5', '77TB'.

В языке ПЛ/М различают простые, индексированные переменные и селекторы поля структуры. *Простая переменная* определяет отдельное скалярное значение и изображается идентификатором. *Индексированная переменная* представляет компонент массива. *Селектор поля структуры* — конструкция, соответствующая некоторому компоненту структуры.

Каждая переменная в программе до ее первого использования в выражении должна быть описана. Описание определяет имя переменной, ее характеристики (атрибуты) и дает транслятору необходимую информацию для резервирования объема памяти при размещении значений переменной.

В языке ПЛ/М допускается два типа переменных: BYTE и ADDRESS. Переменная типа BYTE занимает в памяти 8 бит и является однобайтным элементом, переменная типа ADDRESS занимает 16 бит и является двухбайтным элементом. Описание переменных осуществляется с помощью оператора DECLARE. Общий формат объявления простой переменной имеет вид

DECLARE <идентификатор> <тип> [<начальное значение>]

Пример:

```
DECLARE X BYTE;  
DECLARE Y ADDRESS;  
DECLARE Z BYTE;
```

В одном операторе DECLARE можно описывать несколько переменных, например оператор

```
DECLARE (X, Z) BYTE, Y ADDRESS;
```

аналогичен трем операторам предыдущего примера.

Массивы. *Массив* — совокупность фиксированного числа компонентов одного и того же типа. Допускаются только одномерные массивы с нулевым начальным значением индекса. Общий формат объявления массива:

```
DECLARE <имя> (<размерность>) <тип>  
[<начальные значения>];
```

Например, массив, состоящий из 20 элементов адресного типа, описывается таким образом:

```
DECLARE X (20) ADDRESS;
```

Элементами этого массива будут переменные X (0), X (1), ..., X(19).

Структуры. Для представления логически связанных данных (в общем случае разных типов) в языке ПЛ/М используются *структуры*. Структура — совокупность фиксированного числа компонентов, возможно, разных типов. Прежде чем использовать структуру в программе, ее необходимо описать. Описание задает для каждого элемента структуры имя и атрибуты. Общий формат объявления структуры следующий:

```
DECLARE <имя> [(<размерность>)]  
STRUCTURE (<поле> [, <поле>] ...)  
[BY <имя структуры>]
```

Например, оператор

```
DECLARE X STPUCTURE (A1 ADDRESS, A2 BYTE);
```

описывает структуру X, содержащую компоненты A1 адресного поля и A2 — байтового поля.

Компонентами структур могут быть массивы. Например, оператор

```
DECLARE      GRUPPA      STRUCTURE  
FAM(20) BYTE  
IM (10) BYTE  
OT (10) BYTE  
NOM ADDRESS ;
```

описывает структуру с именем GRUPPA, состоящую из четырех компонент. Первая компонента состоит из 20 байтовых элементов, вторая и третья — из 10, а четвертый является простой адресной переменной.

Базированные переменные. *Базированные переменные* — переменные, местоположение которых в памяти определяется динамически в момент выполнения программы в зависимости от значения других переменных — ее базы. *База* — простая переменная, имеющая тип ADDRESS. Для задания базированных переменных используется атрибут BASED, который записывается сразу за именем базированной переменной. Например, оператор

```
DECLARE X ADDRESS, (Y BASED X) BYTE;
```

описывает базу X как переменную адресного типа, а затем — базированную переменную Y байтового типа. Так как база указывает на некоторую ячейку памяти, то она обязательно должна быть адресного типа.

Базированные переменные позволяют составить программу таким образом, чтобы она оперировала не с самими данными, а с их адресами.

Макроопределения. С помощью описания LITTERALLY можно составить макроопределение, подставляемое в программу на этапе компиляции. Объявление макроопределения имеет вид

DECLARE <макроимя> LITERALLY <строка>.

Строка является последовательностью произвольных символов длиной не более 255 символов, заключенной в апострофы.

Выражения. В языке ПЛ/М используются арифметические и логические выражения.

Арифметическое выражение использует числовые константы, переменные, функции, знаки арифметических операций и круглые скобки. Значением арифметического выражения является число. В языке определены семь арифметических выражений, использующих символы: + и —, что соответственно означает сложение и знак положительного числа, вычитание и знак отрицательного числа; * — умножение; / — целочисленное деление; MOD — получение остатка от деления; PLUS — сложение с учетом текущего значения признака переноса; MINUS — вычитание с учетом текущего значения знака переноса. Примеры записи арифметических выражений:

$(2 * A * X - B * X + C) / (2 * A)$

$(A \text{ MOD } 5) * 2 - 4.$

Арифметическое выражение не только вычисляет значение, но и определяет тип результата. При выполнении операций сложения вычитания, PLUS и MINUS результат имеет тип BYTE, если оба операнда имеют тип BYTE. В остальных случаях результат имеет тип ADDRESS. При выполнении операций MOD, умножения и деления результат всегда имеет тип ADDRESS независимо от типов участвующих в операции операндов.

Логическое выражение принимает одно из двух значений: истинно (OFFH) или ложно (OOH).

Логическое выражение строится с помощью символов логических операций, к которым относятся NOT — отрицание (НЕ), AND — логическое умножение (И), OR — логическое сложение (ИЛИ), XOR — исключающее ИЛИ. Оператор NOT — одноместный, остальные операторы — двуместные. Все операторы производят поразрядные логические операции.

Тип операции NOT совпадает с типом операнда. Операции AND, OR, XOR с двумя операндами типа BYTE имеют результат типа BYTE; в противном случае результат имеет тип ADDRESS.

Помимо логических операторов в языке ПЛ/М имеются операторы отношения < (меньше), > (больше), <= (меньше или равно), >= (больше или равно), = (равно), <> (не равно), обеспечивающие сравнение значений операндов. Результат операции сравнения всегда имеет тип BYTE.

Операторы для выражения алгоритмических действий. В языке ПЛ/М различают простые, условный и структурные операторы.

Простые операторы — операторы, в конструкции которых не входят никакие другие операторы. К ним относятся следующие восемь операторов.

О п е р а т о р п р и с в а и в а н и я предписывает вычислить выражение, записанное в его правой части (после знака =), и присвоить вычисленное значение переменной, имя которой записано в левой части. Формат оператора

[<переменная>], ... <переменная> = <выражение>;

Например, $X = A + B - 17$;

О п е р а т о р G O T O (перехода) передает управление по соответствующему указателю метки и имеет формат

G O T O <указатель метки>;

О п е р а т о р C A L L (вызова) передает управление процедуре, обозначенной указанным именем процедуры, и имеет формат C A L L <имя процедуры> [<список фактических аргументов>];

Например, запись

C A L L B (8)

означает, что при обращении к процедуре B передается один аргумент — число 8.

О п е р а т о р R E T U R N (возврата) завершает выполнение процедуры (поэтому должен находиться в теле процедуры) и имеет формат

R E T U R N <выражение>;

или

R E T U R N ;

Первая форма используется в функциях, вторая — в подпрограммах.

О п е р а т о р п у с т о й никак не обозначается и не выполняет никаких действий.

О п е р а т о р ы E N A B L E и D I S A B L E управляют механизмом прерываний в системе. Оператор E N A B L E разрешает обрабатывать прерывания в системе, а оператор D I S A B L E запрещает такую обработку.

О п е р а т о р H A L T приводит к останову процессора с последующим разрешением обработки прерывания.

Условный оператор I F служит для организации разветвлений в программе при выполнении некоторого условия. Оператор имеет формат

I F <условие> T H E N <оператор 1>; [E L S E <оператор 2>];

В качестве условия записывается логическое выражение языка. Порядок выполнения оператора такой. Если логическое выражение истинно (результат сравнения равен OFFH), то выполняется оператор 1, а оператор 2 пропускается. Если значение логического выражения ложно (равно 00H), то оператор 1 пропускается и выполняется оператор 2. Если конструкция ELSE <оператор 2> отсутствует, тогда при условии, что логическое выражение ложно, сразу выполняется оператор, следующий за условным. В качестве примера рассмотрим фрагмент программы вида

```
IF (X > Y) THEN Z = X + 1; ELSE Z = X - 1;
```

Здесь, если $X > Y$, то выполняется оператор $Z = X + 1$, в противном случае — оператор $Z = X - 1$.

Структурные операторы — операторы, в запись которых входят другие операторы. В языке ПЛ/М имеются четыре структурных оператора: составной, DO WHILE, DO CASE и итеративный цикл.

Составной оператор позволяет объединить несколько предложений посредством операторных скобок DO и END и имеет формат

```
DO; <оператор>, ..., <оператор>, END;
```

Ключевое слово DO означает «выполнить» и указывает начало, а ключевое слово END — конец составного оператора.

В качестве примера рассмотрим такую ситуацию: если $X \leq 5$, то переменной Y нужно присвоить значение 0, а переменной Z — значение 1. В противном случае переменной Y следует присвоить значение 1, а переменной Z — значение 2. Фрагмент программы имеет вид

```
IF X <= 5 THEN DO; Y = 0; Z = 1; END;
```

```
ELSE DO; Y = 1; Z = 2; END;
```

В теле составного оператора можно записывать любые операторы, в том числе и составные.

Оператор DO WHILE является одной из форм оператора цикла и предписывает многократное выполнение группы операторов, заключенной в нем. Оператор удобен, когда число повторений цикла заранее не известно. Оператор DO WHILE имеет формат

```
DO WHILE <логическое выражение>; <операторы>; END;
```

Если логическое выражение истинно, то выполняются операторы, составляющие тело цикла, в противном случае выполняется оператор, находящийся после оператора END. Телом цикла называются операторы, расположенные между логическим выражением и оператором E.

О п е р а т о р DO CASE предназначен для реализации разветвлений в программе и имеет структуру

DO CASE <выражение>; <операторы>; END;

Оператор DO CASE передает управление на оператор, выбранный из последовательности, заключенной в нем. Значение выражения в этом операторе является номером оператора в последовательности, на который передается управление. Первый оператор последовательности имеет номер 0, второй — 1 и т. д. Если значение выражения равно нулю, то выполняется первый оператор, если единице — второй и т. д.

(Если значение выражения равно n , а в списке операторов их меньше $n + 1$; то выполняется оператор, следующий за оператором END.

Итеративный цикл обеспечивает выполнение группы операторов (тела цикла) определенное, заранее известное число раз.

Процедуры. Процедура — часть программы, которая должна использоваться многократно в разных частях программы при различных значениях переменных.

Применение процедур составляет основу модульного программирования, где алгоритмически независимые части программы оформляются в виде процедур — модулей.

Использование процедуры предполагает наличие в программе описания процедуры, где оформляется алгоритм обработки данных, и обращение к процедуре, которое возбуждает однократное выполнение этого алгоритма. Объявление процедуры имеет формат

<заголовок процедуры>

<тело процедуры>

<выход>

Заголовок процедуры в общем случае имеет формат

<имя процедуры>: PROCEDURE [<список параметров>]

[<тип>]

[<атрибуты процедуры>;

и определяет имя процедуры, список формальных параметров, тип результата и атрибуты процедуры.

Если после выполнения типа процедуры в точку вызова передается результат, то такую процедуру называют *функцией*. Если выполнение процедуры не связано с вычислением определенного результата, то процедуру называют *подпрограммой*. Процедура завершается либо выполнением оператора RETURN в теле процедуры, либо доп. полнением слова END, заканчивающего объявление процедуры.

В языке ПЛ/М допускается возможность обращения также к процедурам, написанным на языке ассемблера.

В качестве примера рассмотрим способы использования встроенной процедуры TIME. При программировании процессов, выполняемых в реальном масштабе времени, необходимы средства управления. С помощью процедуры TIME организуется задержка выполнения операторов программы на интервал времени, определяемый значением аргумента.

Общий вид обращения к процедуре TIME:

```
CALL TIME <выражение>;
```

Аргумент этой процедуры имеет тип BYTE, а его значение определяет интервал времени, кратный 100 мкс.

В фрагменте программы

```
DECLARE (X, Y) BYTE
```

```
OUTPUT (5) = X;
```

```
CALL TIME (50);
```

```
OUTPUT (5) = Y
```

```
.....
```

вывод значения переменной Y будет задержан на 5 мкс относительно момента вывода значения переменной X.

Встроенные функции и предопределенные переменные. В языке ПЛ/М имеется множество встроенных функций (процедур).

Встроенные функции сдвига кодов позволяют выполнять сдвиги двоичных значений аргументов на заданное число разрядов. Встроенные функции с именами ROL и ROR производят циклический сдвиг кода первого аргумента на число сдвигов, задаваемых вторым аргументом влево (ROL) или вправо (ROR). Встроенные функции SCL и SCR выполняют циклический сдвиг кода первого аргумента с участием признака переноса CARRY влево (SCL) или вправо (SCR) на число разрядов, задаваемых вторым аргументом. Функции SHL и SHR выполняют операции логического сдвига.

Встроенные функции десятичной арифметики позволяют упростить арифметические операции с данными в десятичной системе счисления.

С помощью встроенных функций INPUT и OUTPUT реализуются операции ввода и вывода соответственно.

Среди других встроенных функций можно назвать функцию ввода символа с консоли INCHAR, функцию вывода символа на консоль OUTCHAR, функцию определения длины аргумента LENGTH и индекса последнего элемента аргумента LAST. Существуют и другие встроенные функции.

Предопределенные переменные используются в программе только для определенных целей. К таким переменным относятся переменные с именами CARRY, ZERO, SIGN, PARITY, MEMORY и STACKPTR.

Первые четыре переменных применяются для опроса соответствующих разрядов (флажков) регистра состояния процессора. Переменная MEMORY используется для объявления массива с нулевой длиной. Переменная STACKPTR служит для указателя стека. Значением этой переменной всегда является адрес вершины стека.

Структура программы. Язык ПЛ/М является языком с блочной структурой. Это означает, что определенные части программы, а именно блоки, можно программировать таким образом, чтобы исключить нежелательные взаимосвязи между другими блоками и остальной частью программы. Указанная возможность обеспечивается благодаря существующему в языке принципу локализации описанных в блоке объектов (переменных, массивов, структур, процедур, макромимен и др.), что делает эти объекты независимыми от текстуального окружения в программе.

Блочная структура программы позволяет вести раздельную (и почти независимую) разработку частей одной программы разными программистами и использовать в разрабатываемой программе готовые, ранее составленные и отлаженные части программы (блоки, процедуры).

3. Язык ПАСКАЛЬ

Язык ПАСКАЛЬ — структурный язык высокого уровня, который позволяет естественным образом определять данные и писать программы с использованием принципа иерархической организации. Язык ПАСКАЛЬ разработан Н. Виртом на базе языка АЛГОЛ W, основой которого послужил язык АЛГОЛ-60. Язык ПАСКАЛЬ, будучи языком умеренной сложности, содержит развитые средства для организации и обработки разнообразных структур данных и является весьма совершенным инструментом для создания надежного программного обеспечения.

Простейшие конструкции языка. Любая программа состоит из двух основных частей: описания данных и описания действий. Действия задаются операторами, а данные — описаниями и определениями. Основные типы данных, используемые в языке ПАСКАЛЬ, — скалярные. Существует четыре стандартных скалярных типа: целый (INTEGER), действительный (REAL), литерный (CHAR) и логический (BOOLEAN).

Числа в языке ПАСКАЛЬ могут быть целого или действительного (вещественного) типа. *Данные* могут быть логического или символьного (литературного) типа. Данные логического типа имеют два значения: TRUE или FALSE. Данные символьного типа принимают значение только одной литеры, имеющейся в наборе литер конкретной модели ЭВМ.

Имя (идентификатор) образуется из букв и цифр, начинающихся с буквы. С помощью числовых, логических, символьных значений

и имен записываются простейшие конструкции языка: константы, переменные, стандартные функции, выражения.

Константа представляет собой значение, которое не изменяется в программе. Константы могут быть целыми, действительными, логическими, символьными и текстовыми.

Переменные используются для записи значений, изменяющихся в программе.

Стандартные функции служат для облегчения программирования и обращения к наиболее часто встречающимся функциям при обработке данных.

Выражения определяют действия и последовательность вычисления значения. Они могут состоять из констант, переменных, стандартных функций, разделенных скобками и знаками операций. Все операции над данными объединяются в четыре группы: отрицания (NOT); мультипликативные (*, /, DIV, MOD, AND); аддитивные (+, —, OR); отношения (<=, <, =, <>, >, >=).

Операторы языка. С помощью операторов описываются действия над данными, выполняемые для реализации алгоритма решения задачи. По функциональному назначению операторы языка можно подразделить на группы: присваивания, ввода—вывода, управления, определения функций и процедур (подпрограмм).

В языке ПАСКАЛЬ по своему составу определены операторы двух типов: простые и структурные. Простым считается оператор, который не содержит других операторов, а структурным — оператор, содержащий в качестве компонентов один или несколько операторов. К простым операторам относятся операторы присваивания, перехода, процедуры, к структурным операторам — операторы условные, выбора, цикла, составные. Составной оператор представляет собой последовательность операторов, заключенную в ключевые слова BEGIN и END.

Операция программ линейной структуры. Программы линейной структуры не содержат никаких условий. Операторы такой программы выполняются в той последовательности, в которой они записаны. Для организации линейных программ требуются операторы присваивания, ввода информации и вывода результатов вычислений.

Оператор присваивания служит для вычисления значения выражения и присваивания его переменной или функции, расположенной слева от символа :=. Общий вид записи оператора

$a := b;$

где a — имя переменной; b — выражение.

Синтаксическая диаграмма оператора присваивания приведена на рис. 3.7.

Операторы ввода и вывода данных обеспечивают программу исходными данными, необходимыми для решения задачи, и осуществляют ввод и вывод результатов решения.

Для ввода используются операторы:

READ ($b_1, b_2, \dots, b_n$);

READLN ($b_1, b_2, \dots, b_n$);

READLN;

где $b_1, b_2, \dots, b_n$ — имена переменных, подлежащих вводу.

Оператор READ ($b_1, b_2, \dots, b_n$); обеспечивает выборку данных из стандартного входного файла INPUT. Оператор READLN ($b_1, b_2, \dots, b_n$) обеспечивает выборку данных из стандартного файла INPUT, но после окончания выборки последней переменной осуществляет переход к началу новой строки файла. Оператор READLN; обеспечивает пропуск одной строки в стандартном файле INPUT и переход к началу новой строки.

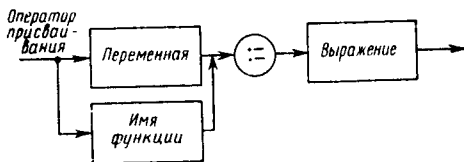


Рис. 3.7. Синтаксическая диаграмма оператора присваивания

Для вывода информации используются операторы:

WRITE ($b_1, b_2, \dots, b_n$);

WRITELN ($b_1, b_2, \dots, b_n$);

WRITELN;

где $b_1, b_2, \dots, b_n$ — имена переменных, подлежащих выводу.

Оператор WRITE ($b_1, b_2, \dots, b_n$); выполняет вывод значений, соответствующих именам $b_1, b_2, \dots, b_n$, в стандартный выходной файл OUTPUT, при этом выводимые значения размещаются в одной строке. Оператор WRITELN ($b_1, b_2, \dots, b_n$); выполняет вывод значений, соответствующих именам $b_1, b_2, \dots, b_n$, в стандартный файл OUTPUT и после вывода последнего значения осуществляет переход к новой строке файла OUTPUT. Оператор WRITELN; обеспечивает пропуск строки в файле и переход к началу следующей строки.

Форма представления выводимых переменных определяется типом переменных: значения величин целого типа выводятся в обычной форме; значения величин действительного типа — в виде нормализованного числа действительного типа с порядком; значения логического типа — в виде логических значений TRUE и FALSE; значения символьных переменных — в виде соответствующих символов.

При выводе значений действительного типа с фиксированной точкой указывается ширина поля, отводимого под все значения (m) и дробную часть числа (n), например:

WRITE ($b : m : n$);

При выводе нескольких значений на одной строке для наглядности они задаются необходимым количеством пробелов. Для этого за-

писывают оператор вывода в виде WRITE (":q); где q — константа целого типа, указывающая число пробелов.

Пример размещения информации при выводе:

```
PROGRAMM PRIM I(OUTPUT);
```

```
CONST PI=3.141592;
```

```
T=40 I;
```

```
W=TRUE;
```

```
SIM='D';
```

```
BEGIN
```

```
  WRITELN(PI:8:6);
```

```
  WRITELN(T:6, ":5, W:4, ":6, SIM:1);
```

```
END
```

Информация выводится в две строки в виде 3.141592

```
401 _____ TRUE _____ D;
```

Организация программ разветвляющейся структуры. Для организации разветвлений в программах используются операторы перехода, условные и выбора.

Оператор перехода имеет общий вид записи GO TO n , где n — метка. Синтаксическая диаграмма его представлена на рис. 3.8.

Условный оператор имеет общие формы записи:

```
IF  $b$  THEN  $a$ ; или IF  $b$  THEN  
 $a_1$  ELSE  $a_2$ ;
```

где b — логическое выражение; a , a_1 , a_2 — операторы.

Первая форма записи оператора позволяет организовывать вычисление оператора a , если логическое выражение имеет значение TRUE, в противном случае выполняется оператор, стоящий за условным.

Примеры записей условного оператора:

```
IF X<0 THEN P:=X*X;
```

```
IF D>0 THEN BEGIN X1:=R+1; X2:=R-1 END;
```

```
IF D>0 THEN BEGIN X1:=R+1; X2:=R-1 END;  
ELSE BEGIN F:=R; E:=1 END;
```

Синтаксическая диаграмма условного оператора приведена на рис. 3.9.

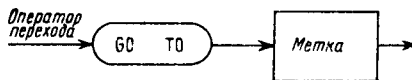


Рис. 3.8. Синтаксическая диаграмма оператора перехода

Оператор выбора CASE обеспечивает организацию разветвлений путем выбора одного из нескольких операторов. Синтаксическая диаграмма оператора выбора приведена на рис. 3.10.

Организация программ циклической структуры. Программы циклической структуры могут быть организованы с помощью условного оператора и оператора переходов, однако в языке ПАСКАЛЬ имеются специальные операторы цикла: с параметром (FOR), с предусловием (WHILE), с постусловием (REPEAT).

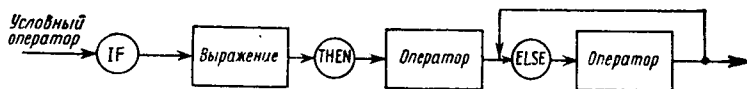


Рис. 3.9. Синтаксическая диаграмма условного оператора

Оператор цикла FOR служит для организации цикла с известным числом повторений. Общий вид записи оператора

FOR $i = m_1$ TO m_2 DO s ;

где i — параметр цикла; m_1, m_2 — начальное и конечное значения параметра цикла; s — тело цикла, которое содержит один или несколько операторов.

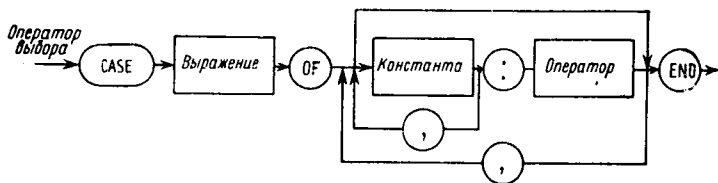


Рис. 3.10. Синтаксическая диаграмма оператора выбора

Заменив в операторе цикла ключевое слово TO на DOWNT0, изменим параметр цикла в обратном порядке с шагом -1 :

FOR $i = m_1$ DOWNT0 s ;

Синтаксическая диаграмма оператора цикла с параметром FOR приведена на рис. 3.11.

Оператор цикла WHILE позволяет организовывать цикл с неизвестным числом повторений, так как оно зависит от вычислений в операторе. Общий вид записи оператора

WHILE b DO s ;

где b — логическое выражение; s — тело цикла.

Если логическое выражение имеет значение TRUE, то выполняются операторы, входящие в тело цикла, в противном случае выполнение операторов цикла прекращается. Синтаксическая диаграмма оператора цикла с предусловием приведена на рис. 3.12.

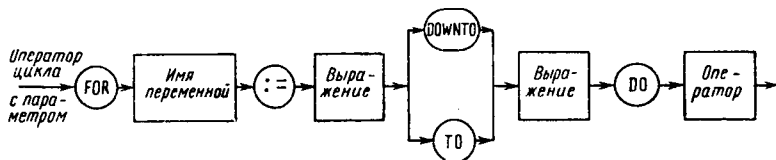


Рис. 3.11. Синтаксическая диаграмма оператора цикла с параметром FOR

Оператор цикла REPEAT позволяет организовать цикл с неизвестным числом повторений и действует подобно оператору WHILE, но при его выполнении проверка условия производится после каждой итерации. Отметим три основных отличия оператора REPEAT от оператора WHILE: оператор REPEAT обеспечивает выполнение, по край-



Рис. 3.12. Синтаксическая диаграмма оператора цикла с предусловием

ней мере, одного цикла; когда логическое условие имеет значение FALSE, выполнение цикла продолжается; цикл может содержать не один, а несколько операторов. Синтаксическая диаграмма оператора цикла с постусловием приведена на рис. 3.13.

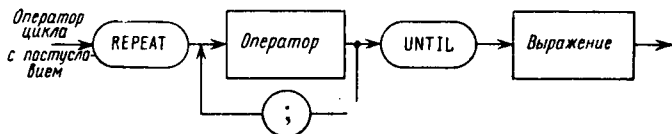


Рис. 3.13. Синтаксическая диаграмма оператора цикла с постусловием

Нестандартные скалярные типы данных. К нестандартным скалярным типам данных относятся перечисляемые и ограниченные типы, определяемые пользователем. Использование этих типов данных позволяет разработать более простую и наглядную программу.

Перечисляемый тип данных представляет собой упорядоченное множество значений, образуемое перечислением имен, обозначающих

эти значения. Синтаксическая диаграмма данного типа приведена на рис. 3.14. Применение перечисляемых типов данных позволяет проще запрограммировать нечисловую задачу.

Примеры записи определения типов:

TYPE

```
GEOMFIG = (TREYG, KVADR, PRYMYG, KRYG, MNOGOYG;  
MERA = (MMETR, CMETR, METR, KMETR);
```

После определения типа данных объявляются переменные данного типа следующим образом:

VAR

```
FIGURA : GEOMFIG;  
RASST : MERA;
```

Оба описания могут быть объединены в одно:

VAR

```
FIGURA : (TREYG, KVADR, PRYMYG, KRYG, MNOGOYG);  
RASST : (MMETR, CMETR, METR, KMETR);
```

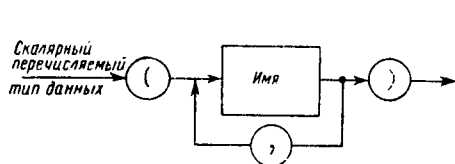


Рис. 3.14. Синтаксическая диаграмма скалярного перечисляемого типа данных

Над описанным типом данных можно производить действия согласно стандартным функциям языка ПАСКАЛЬ, при этом значением стандартной функции является порядковый номер переменной или константы перечисляемого типа.

Ограниченные (интервальные) *типы данных* определяют

элементы подмножества идущих подряд величин некоторого базового типа. Ограниченный тип представляется двумя константами, задающими диапазон значений, например:

TYPE

```
INDEX = 1 ... 100;  
MERA = MMETR ... KMETR;
```

Переменные ограниченных типов данных описываются в разделе описания переменных таким образом:

VAR

```
SIZE : INDEX;  
RASST : MERA;
```


что можно объединить:

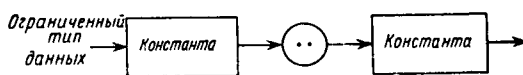
VAR

SIZE: 1 ... 100;

RASST:MMETR ... KMETR;

Синтаксическая диаграмма ограниченного типа данных показана на рис. 3.15.

Рис. 3.15. Синтаксическая диаграмма ограниченного типа данных



Структурированные типы данных. В языке ПАСКАЛЬ используются следующие классы структур данных: массивы, записи, множества, файлы.

Регулярные типы данных (массивы) представляют собой упорядоченную последовательность переменных одного типа. Переменные массива являются его компонентами. Каждая компонента имеет имя и индекс, заключенный в квадратные скобки. Синтаксическая диаграмма регулярного типа данных приведена на рис. 3.16.

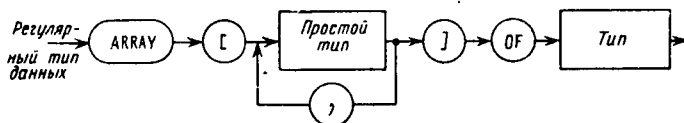


Рис. 3.16. Синтаксическая диаграмма регулярного типа данных

Записи представляют собой объединения фиксированного числа логически связанных компонент, называемых полями. Компоненты записи могут быть разных типов и различаются между собой именами полей. Синтаксическая диаграмма структуры записей показана на рис. 3.17.

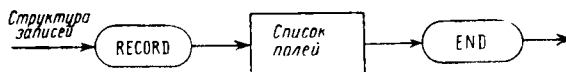


Рис. 3.17. Синтаксическая диаграмма структуры записей

Множественные типы данных (множества) представляют собой ограниченный набор различных элементов базового типа — скалярный или ограниченный тип данных. Синтаксическая диаграмма множественного типа данных представлена на рис. 3.18.

Файловые типы данных (файлы) представляют собой упорядоченную совокупность произвольного числа однотипных компонент с последовательным доступом. Синтаксическая диаграмма файлового типа данных показана на рис. 3.19.

Более подробно структурированные типы данных были описаны в § 3.3.

Динамические структуры данных. Неструктурированные и структурированные типы данных представляют собой статические объекты

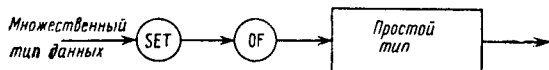


Рис. 3.18. Синтаксическая диаграмма множественного типа данных

программы. Они создаются до выполнения программы, не зависят от абсолютных адресов загрузки и существуют в течение всего времени выполнения блока, в котором используются.

Помимо статических объектов в языке ПАСКАЛЬ определены динамические объекты, которые возникают в момент выполнения программы и существуют до их явного уничтожения либо до прекращения вычисления. Динамические объекты образуются с помощью *ссылочного типа данных*, синтаксическая диаграмма которого представлена на рис. 3.20. Переменные ссылочного типа имеют значение адреса

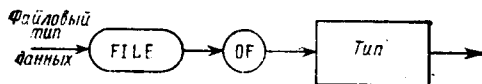


Рис. 3.19. Синтаксическая диаграмма файлового типа данных

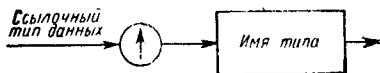


Рис. 3.20. Синтаксическая диаграмма ссылочного типа данных

создаваемых объектов. Они являются ссылками (указателями) на создаваемые объекты.

Определение данных ссылочного типа выглядит таким образом:

TYPE

SV = ↑ PT;

VAR

FD : SV;

и означает, что SV является ссылкой на PT. Символ ↑ указывает на то, что переменная SV является ссылкой. Переменная FD является ссылкой на объекты типа PT.

Динамические переменные в программе создаются в результате обращения к стандартной процедуре NEW.

Использование данных ссылочного типа позволяет описывать сложные структуры: списки, очереди, деревья и их разновидности.

§ 3.5. ПРИНЦИПЫ РАЗРАБОТКИ РАБОЧИХ ПРОГРАММ ДЛЯ МПУ

Этапы разработки рабочих программ. Разработка прикладного программного обеспечения для МПУ представляет сложный процесс, состоящий из ряда взаимосвязанных этапов.

Первым является *этап формулирования требований и постановки задачи*. На этом этапе вырабатываются требования, предъявляемые к проектируемому программному изделию, и производится формализация задачи. Устанавливается сущность процессов обработки информации, определяются входы и выходы программы, формулируются системные ограничения (время выполнения, точность, время реакции и др.).

На *этапе проектирования* определяется структура программы в соответствии с формулой Н. Вирта «Программа = алгоритм + структура данных». На этом этапе осуществляется декомпозиция общей задачи на отдельные фрагменты путем выделения некоторого набора взаимодействующих программных модулей высокого уровня. При проектировании программ используются такие методы, как нисходящее проектирование, структурное программирование, модульное программирование и создание структурных схем программ.

Этап кодирования часто называют собственно этапом программирования. На этом этапе спроектированная ранее программа переводится на язык машинных команд. Практика создания программного обеспечения показывает, что ошибки при проектировании программ совершаются гораздо чаще, чем при кодировании, так что качественное проектирование является важнейшей предпосылкой создания качественного и надежного прикладного программного обеспечения.

На *этапе отладки* осуществляются поиск и исправление программных ошибок. Процесс отладки программ довольно трудоемок и, как правило, требует специальных отладочных средств.

На *этапе тестирования* проверяется корректность программы. Тестирование позволяет убедиться в том, что программа правильно выполняет возложенные на нее функции. Для проведения тестирования необходимо составлять план тестирования, основным содержанием которого являются наборы входных данных и ожидаемые результаты работы программы. Статистика показывает, что трудозатраты на отладку программ и тестирование составляют примерно 40% от общих трудозатрат на создание прикладного программного обеспечения. Для сопровождения и эксплуатации программ производят документирование программ, которое должно соответствовать требованиям ЕСПД.

Для больших программ требуется организация их *сопровождения* во время эксплуатации. Это связано с необходимостью корректировки программ при изменении условий или места их использования, а также определения ошибок, не выявленных при отладке и тестировании.

В настоящем параграфе рассматриваются вопросы проектирова-

ния программ, а вопросы отладки и тестирования освещены в гл. 4 настоящего учебника.

Особенности проектирования программ для встроенных МПУ. Разработка программного обеспечения МПУ, встроенных в состав РЭС, имеет ряд специфических особенностей по сравнению с разработкой программ для микроЭВМ. Прежде всего определяется степень взаимодействия МПУ с различными составными частями системы: электронной, электрической, электромеханической и др., выявляется набор вычислительных процедур, реализуемых МПУ, а также перечень входных и выходных сигналов.

Первостепенным вопросом, решаемым при постановке задачи разработки программного обеспечения МПУ, является форма представления входных и выходных данных. При этом определяется, с какими устройствами будет взаимодействовать МПУ и в каком виде они будут посылать и принимать информацию. Основными характеристиками устройств ввода—вывода являются максимальная и средняя скорости обмена данными, процедуры обнаружения ошибок, управляющие сигналы, с помощью которых устройства ввода—вывода сообщают о наличии данных или о готовности принять данные, типичная длина слова, тактовые или стробируемые сигналы и протоколы.

Второй вопрос, решаемый при постановке задачи разработки программного обеспечения, касается требований к процессу обработки информации. Следует четко определить перечень операций над входными данными и порядок решения поступающих задач. Порядок выполнения операций в ряде случаев должен строго соблюдаться, так как сигналы ввода—вывода должны приниматься и посылаться в определенной последовательности. Система может предъявлять требования к временным соотношениям: скоростям обмена в периферийных устройствах; временам задержек в механических или электрических компонентах; временам, в течение которых данные должны оставаться стабильными; временам стабилизации, восстановления, перехода в состояние готовности и выхода из него. Для выполнения этих ограничений используются специальные системы синхронизации. На объем программ и данных, а также на размер буферов существенные ограничения накладываются размерами доступной памяти. Обработка данных во встроенных МПУ должна производиться в реальном времени, т. е. МПУ должны выдавать либо управляющие сигналы, либо результаты обработки информации в сжатые сроки, пока управляемый объект не изменил своего состояния.

Важным вопросом, решаемым при постановке задачи, является обработка ошибок. Необходимо предусмотреть средства восстановления при возникновении некорректных последовательностей операций или ошибочных сигналов. В программах должны предусматриваться средства блокировки типовых ошибок программирования.

Кроме того, при разработке программного обеспечения требуется учитывать вопросы взаимодействия различных программ друг с другом, вопросы влияния различных структур данных и т. д.

Технология проектирования рабочих программ. Традиционно проектирование программы связывается с созданием структурной схемы. Однако разработка подробной структурной схемы — такая же трудоемкая работа, что и описание первоначального варианта работающей программы. Техника разработки структурных схем программ полезна при описании структуры программы и объяснении ее работы. Однако на этапе разработки программы более эффективны следующие методы, нашедшие применение в технологии программного обеспечения.

Технология программного обеспечения — общий термин, используемый для обозначения применения технологических принципов при разработке систем программного обеспечения. Технология программирования включает в себя большое число различных технологических принципов и приемов программирования.

Основой технологии программирования является *структурное программирование*. В структурном программировании используются только простые логические структуры трех типов:

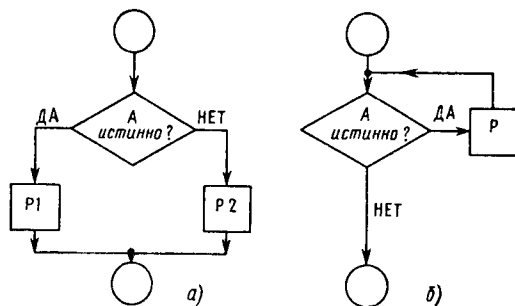


Рис. 3.21. Структурная схема программных структур:

а — ветвление; б — цикл

1) последовательная, в которой команды или программы выполняются последовательно в порядке, в котором они записаны;

2) условная типа IF A THEN P₁ ELSE P₂, где A — логическое выражение; P₁ и P₂ — программы;

3) циклическая типа DO P WHILE A (выполнять P, пока будет A), где A — логическое выражение; P — программа, состоящая только из допустимых структурных элементов.

Программа называется *структурированной*, если она написана с использованием только рассмотренных трех структур или некоторого другого оговоренного набора структур (например, конструкции DO CASE). На рис. 3.21, а приведена структурная схема условного оператора, а на рис. 3.21, б — структурная схема оператора цикла.

При разработке и реализации программного обеспечения для МПУ необходимо придерживаться следующих правил.

Правило 1. Описывайте все переменные. В языках ПЛ/М и ПАСКАЛЬ это правило выполняется достаточно легко, поскольку данные языки требуют описаний, однако оно должно быть принято даже для языков, которые позволяют опускать описания (например, для языка ФОРТРАН).

Правило 2. *Используйте процесс компиляции для установления значений переменных, которые являются константами, и не используйте его для переменных, которые будут изменять свои значения во время выполнения программы.* Для реализации этого правила могут использоваться атрибуты DATA, INITIAL и LITERALLY, допустимые в языке ПЛ/М.

Правило 3. *Составляйте программу так, чтобы в каждом блоке описания предшествовали выполняемой части программы и были представлены в читабельном формате.* Следует помнить, что программа должна читаться людьми так же хорошо, как и машинами.

Правило 4. *Для передачи управления в программе используйте управляющие конструкции структурного программирования: следование, ветвление и повторение. Безусловный переход GO TO должен использоваться крайне редко, поскольку он имеет тенденцию делать логику алгоритма более трудной для понимания.* В частности, в языке ПЛ/М для выражения потока управления имеется достаточное количество управляющих структур. Ветвление возможно в форме структур IF THEN ELSE и DO CASE. Структура DO WHILE и цикл DO возможны для выполнения повторений. Кроме того, процедурное обеспечение и оператор CALL могут применяться для организации модульного программирования. При использовании этих базовых структур программист быстро обнаруживает, что необходимость в операторе GO TO значительно снижается.

Правило 5. *Разбивайте задачу на модули.* Это позволяет длинные программы расчленить на короткие, называемые *модулями*. Модули могут быть спроектированы, закодированы и отлажены отдельно с привлечением минимальных сведений о других программах. Взаимодействия между модулями являются частыми источниками программных ошибок, поэтому они должны быть четко выделены.

Наиболее естественный подход к спецификации модулей дает метод *нисходящего проектирования*. Одно из достоинств этого метода заключается в том, что разработчику приходится иметь дело со связями между отдельными модулями на самом раннем этапе проектирования. При спецификации межмодульных связей разработчик должен по возможности определить характерные линии коммуникаций и изолировать модули друг от друга, насколько это возможно. Каждый модуль должен иметь доступ только к необходимым для него данным.

Модули целесообразно делать короткими. Желательно, чтобы максимальный размер модуля не превышал 50 ... 75 строк. Стандартные модули могут быть оформлены как макрокоманды или просто вставляться в программу как фрагменты.

И, наконец, следует помнить, что целью программирования для микропроцессоров является создание работающей программы. Поэтому первоначальная программа должна быть простой и неизощренной. Нужно избегать инициализации переменных в операторах DATA, использования команд не по их прямому назначению, длинных команд для обработки явно помеченных переменных, резуль-

татов предыдущей операции для инициализации переменных или выполнения вычислений параметров в качестве постоянных данных.

В заключение еще раз отметим, что наиболее важным правилом для программиста независимо от языка программирования, с которым он работает, является стремление в первую очередь к получению правильной программы и только потом — к ее оптимизации и сокращению времени выполнения.

Языки программирования и структурное программирование. Различные языки программирования обладают неодинаковой приспособленностью к написанию структурированных программ. К настоящему времени АЛГОЛ, ПАСКАЛЬ, ПЛ/М и АДА являются языками, которые позволяют наиболее естественным образом реализовать принципы структурного программирования. Наличие в этих языках блочной структуры позволяет программисту группировать несколько отдельных операторов и затем обращаться с ними, как с одним составным оператором. Эти языки содержат также в чистом виде основные управляющие конструкции структурного программирования.

Из широко распространенных языков высокого уровня ФОРТРАН является, по-видимому, менее всего приспособленным для реализации идей структурного программирования.

Во-первых, в ФОРТРАНе отсутствует блочная структура, что приводит к необходимости обходить группы операторов, применяя оператор GO TO.

Во-вторых, нет возможности составлять сложную конструкцию из вложенных один в другой операторов IF — THEN — ELSE. Конструкция IF вообще является слабым местом ФОРТРАНа с точки зрения структурного программирования.

В-третьих, конструкция DO в ФОРТРАНе не гибка и ее использование ограничивается организацией итерационных процессов.

При программировании на машинно-ориентированных языках ассемблера невозможно непосредственно выполнить требования структурного программирования, поскольку при отображении двумерной управляющей конструкции на одномерную структуру памяти невозможно обойтись без операторов условного перехода. Это не означает, что методы структурного программирования в этом случае неприменимы. Наоборот, они являются весьма эффективными и в первую очередь при программировании на машинно-ориентированных языках, поскольку программам, написанным на этих языках, присущи недостатки, ослабляемые структурным программированием: плохая обозримость и воспринимаемость, сложность внесения изменений и др. Программист должен иметь в виду указанное ограничение и искать пути его преодоления, которые могут быть самыми различными — от варьирования алгоритма до использования макросредств.

Идеология, получившая название «структурное программирование», все более распространяется, однако следует иметь в виду, что структурное программирование не есть система готовых рецептов для написания качественных программ и идеи технологии программиро-

вания должны применяться программистом творчески. Нужно помнить, что хотя в принципе любая программа может быть автоматически преобразована в эквивалентную программу, не содержащую операторы GO TO, на практике эта возможность никем не реализуется. Отсутствие оператора GO TO еще не означает улучшение программы. Отказ от оператора GO TO иногда может привести к довольно длинной последовательности взаимовложенных конструкций IF—THEN—ELSE, что делает программу столь же трудно воспринимаемой, как и ее неструктурированная версия: для внутренних IF довольно трудно определить, какие же условия должны выполняться.

§ 3.6. МИКРОПРОГРАММИРОВАНИЕ

Принцип микропрограммного управления. Одним из самых мощных инструментов разработчиков цифровых устройств является микропрограммирование. Микропрограммное управление является средством упорядоченной и гибкой реализации управления ЭВМ и МПС и альтернативой аппаратному управлению. Микропрограммным управлением называется способ организации работы устройства управления, при котором каждая команда, при вводе ее в регистр команд, определяет выполнение последовательности микрокоманд, в совокупности реализующих алгоритм выполнения команды. В системах с микропрограммным управлением каждая машинная команда интерпретируется стандартной микропрограммой — последовательностью микрокоманд, записанной в специальной управляющей памяти ЭВМ или МПС. Каждая микрокоманда (МК) состоит из микроопераций (МО), реализующих аппаратным способом некоторые простейшие операции переработки информации.

Пользователь имеет возможность использовать микропрограммирование в секционных микропроцессорах. В отличие от однокристальных МП с жестким управлением секционный МП реализуется в виде нескольких интегральных микросхем. Обычно это определенное число микропроцессорных секций, блок микропрограммного управления (БМПУ), ПЗУ микрокоманд и ряд других схем большой, средней и малой степеней интеграции.

Микропрограмма состоит из совокупности микрокоманд. Микрокоманда содержит в общем случае три поля: управления генерацией адреса следующей микрокоманды; управления длительностью такта; управления операционной частью микропроцессора.

Первое поле предписывает, где взять и как преобразовать информацию для генерации адреса следующей микрокоманды. Это поле управляет вычислением условий перехода, блоком прерываний и может содержать константы --- адреса перехода к фиксированным ячейкам микропрограммной памяти.

Второе поле управляет работой схем синхронизации микропроцессора, обеспечивающих выработку такта переменной длительности. Регулирование длительности такта — один из наиболее эффективных способов достижения повышенного быстродействия микропроцессора.

Третье поле состоит, в свою очередь, из подполей, управляющих работой арифметико-логического устройства адресной обработки вычисляющего адреса команд и операндов в основной памяти (выбор источников операндов АЛУ, выбор приемников результата операции АЛУ, организация сдвиговых операций), а также содержит непосредственные операнды-константы и пр. Методически может оказаться удобным выделение в структуре микрокоманды отдельного поля управления доступом к основной памяти команд и операндов.

Как правило, суммарная длина микрокоманды составляет 30 ... 100 бит и более. Такая значительная разрядность микрокоманды обусловлена тем, что ее поля (за исключением полей констант для АЛУ и схем формирования адреса микропрограммной памяти, а также полей специальных управляющих сигналов) не перекрываются, т. е. используется горизонтальный способ кодирования микрокоманды.

Способы кодирования микрокоманд. Способы кодирования МК и формат МК являются одними из основных характеристик микроЭВМ и МПС, построенных на микропрограммируемых микропроцессорах. Эти характеристики определяют быстродействие, аппаратные затраты на реализацию микропрограммного управления и гибкость микропрограммирования. Существует четыре основных способа кодирования МК: максимальное (вертикальное), унитарное (горизонтальное), кодирование полей совместимых микроопераций, двухуровневое.

При вертикальном кодировании МК операционная часть МК представляется в виде неизбыточного двоичного кода, разрядность которого определяется как наименьшее целое, не меньшее, чем двоичный логарифм от числа микроопераций ($\lceil \log_2 N \rceil$, где N — число МО). Этот способ кодирования МК требует наименьших аппаратных затрат в схеме БМПУ, но при этом требуется дешифратор, преобразующий код МК в соответствующие ей микрооперации. При большом числе МК дешифратор вносит значительную временную задержку, ограничивая общее быстродействие БМПУ. При вертикальном микропрограммировании микрокоманда разбивается на поля и каждая микрооперация определяется двоичным кодом, содержащимся в этих полях. Применение вертикального микропрограммирования на практике приводит к сокращению разрядности микрокоманд в 2—3 раза по сравнению с горизонтальным.

При горизонтальном кодировании МК каждой микрооперации микропрограммы соответствует один разряд в операционной части МК, что устраняет необходимость в дешифраторе. При горизонтальном микропрограммировании максимальное число разрядов микрокоманды $n_0 = n_{\max} = N$. Основное преимущество горизонтального кодирования — максимальные гибкость микропрограммирования и производительность. Любая модификация микропрограммы, не связанная с изменением системы микроопераций, осуществляется простым изменением содержимого управляющей памяти и не приводит к изменению аппаратуры вне управляющей памяти. Такой метод обеспечивает максимальный параллелизм реализации всей системы микроопе-

раций и позволяет микропрограммисту применять МК с произвольным сочетанием микроопераций. При этом следует иметь в виду, что одна горизонтальная МК позволяет параллельно управлять всеми устройствами микроЭВМ или МПС: чтением и записью из памяти, генерацией адреса условного перехода, выполнением операций в нескольких АЛУ и т. д. Основной недостаток горизонтального микропрограммирования — значительная избыточность реализации.

Горизонтальные и вертикальные микрокоманды относятся к крайним точкам широкого спектра возможных решений. Эффективным

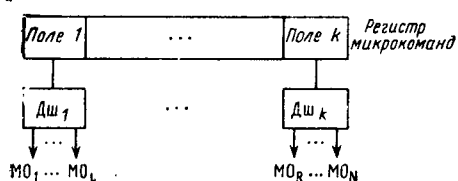


Рис. 3.22. Организация БМПУ с кодированием полей совместимых микроопераций

является метод кодирования полей совместимых микроопераций (прямое кодирование), который позволяет сочетать гибкость и потенциальный параллелизм унитарного кодирования с более экономичным кодированием микроопераций (рис. 3.22). При этом в множестве микроопераций выделяется подмножество совместимых МО, т. е. МО, которые никогда не встречаются вместе в одном слове управляющей памяти. При

разбиении обычно стремятся к минимизации разрядности управляющей памяти. Каждому блоку разбиения множества МО ставится в соответствие поле в формате МК. Внутри каждого поля микрооперации кодируются минимальным числом двоичных разрядов. Каждому полю МК соответствует индивидуальный дешифратор, который в силу небольшой разрядности поля требует небольших аппаратных затрат и вносит малую временную задержку. Этот способ кодирования обеспечивает меньшую гибкость, чем унитарное кодирование, так как изменение хотя бы одной МК может привести к нарушению условия совместимости в одном или нескольких полях МК и к необходимости нахождения нового разбиения множества МО. Однако частично он сохраняет присущий унитарному кодированию параллелизм реализации системы микроопераций.

Для минимизации разрядности слова управляющей памяти иногда используется двухуровневое (косвенное) кодирование. Для этого в МК может быть выделено специальное управляющее поле, служащее для определения других полей. Одним из вариантов косвенного кодирования может быть кодирование микрокоманд с «плавающим» дополнительным полем. При этом в МК дополнительное поле специально не выделяется, а его функции несут определенные коды в основных полях. Двухуровневое кодирование позволяет экономить емкость управляющей памяти, но при этом требуется дополнительное оборудование и уменьшается быстрдействие БМПУ за счет организации двухуровневой дешифрации микроопераций.

По временному признаку характеризуют одноктактное и много- тактное кодирование микрокоманд. В случае одноктактного микропро- граммирования каждая микрокоманда приводит к выработке опре- деленной совокупности сигналов микроопераций, выполняемых одно- временно. Многотактному микропрограммированию соответствуют мик- рокоманды, в которых закодированы серии сигналов микроопераций, выполняемых в течение двух и более тактов. Существуют две разно- видности многотактных микрокоманд: в первой каждому полю микро- команд соответствует определенный такт выполнения, во второй каждое поле содержит код, описывающий конечную последователь-

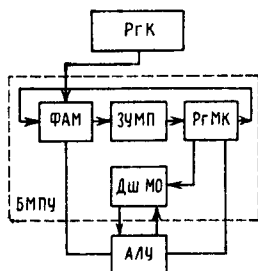


Рис. 3.23. Структурная схема БМПУ

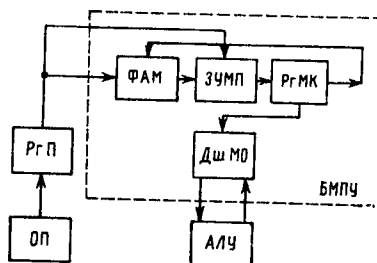


Рис. 3.24. Структурная схема динамического БМПУ

ность микроопераций, зачастую связанных с определенным функцио- нальным узлом. Например, последовательность микроопераций, свя- занных с АЛУ, может включать выборку операндов, суммирование, запись результата.

Структуры устройств микропрограммного управления. Обобщенная структурная схема блока микропрограммного управления приве- дена на рис. 3.23. Выполнение каждой микрокоманды в БМПУ сводится к следующему: расшифровка кода операции и формирование адреса очередной микрокоманды, обращение к памяти микропрограмм (ЗУМП), дешифрация содержимого регистра микрокоманд (РгМК) и операционный цикл.

Узел ФАМ формирует адрес очередной микрокоманды, хранимой в ЗУМП, который в общем случае зависит от кода выполняемой ко- манды, кода предыдущей микрокоманды и сигналов логических усло- вий АЛУ. Содержимое выбранной ячейки ЗУМП поступает в регистр микрокоманд РгМК, откуда код микрокоманды подается в дешифратор микроопераций (Дш МО), который вырабатывает некоторую совокуп- ность управляющих сигналов, подаваемых в АЛУ и вызывающих вы- полнение соответствующих микроопераций.

Структура и функциональные возможности микропрограммных устройств управления в значительной степени зависят от типа микро- программной памяти, метода формирования адресов микрокоманд и способа кодирования формата микрокоманд.

Тип микропрограммной памяти влияет на быстродействие и стоимость микропрограммного процессора, а также на гибкость микропрограммирования. В зависимости от типа микропрограммной памяти различают *статическое* и *динамическое* микропрограммирование. В первом случае в качестве ЗУМП применяется постоянное ЗУ, во втором — оперативное. Динамическое микропрограммирование в отличие от статического позволяет оперативно изменять микропрограммы в БМПУ, изменяя тем самым функциональные свойства процессора. Схема динамического БМПУ приведена на рис. 3.24, где показаны

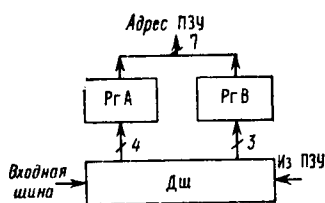


Рис. 3.25. Структура БМПУ на основе дешифратора

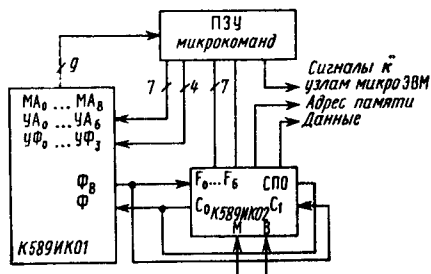


Рис. 3.26. Реализация БМПУ на МПК БИС К589

шины, по которым осуществляется запись микропрограмм из основной памяти (ОП) в микропрограммную. Одним из способов, совмещающих положительные свойства динамического и статического микропрограммирования, является использование ППЗУ.

БИС БМПУ входит в состав каждого МПК БИС с микропрограммным управлением. БМПУ могут иметь различную структурную организацию, но базовыми являются два варианта: на основе дешифратора и на основе счетчика. В первом варианте дешифратор получает из ПЗУ микрокоманд команду, указывающую на способ формирования адреса следующей МК. Во втором варианте адрес следующей МК вычисляется с помощью счетчика.

Структура простейшего БМПУ на основе дешифратора приведена на рис. 3.25. В частности, для ПЗУ микрокоманд емкостью 128 слов необходимо вырабатывать 7-разрядный адрес, который в БМПУ хранится на двух регистрах адреса: 4-разрядном РгА и 3-разрядном РгВ. ПЗУ может быть представлено в виде матрицы, имеющей $2^4 = 16$ строк и $2^3 = 8$ столбцов (адрес строки указывается РгА, а адрес столбца — РгВ). В этой схеме дешифратор получает из ПЗУ команду, содержащую указания для вычисления следующего адреса ПЗУ. После дешифрации команды в БМПУ генерируется новый адрес, который загружается в адресные регистры РгА и РгВ. Наиболее распространенными МПК, в состав которых входят БМПУ на основе дешифратора, являются К589 и Intel 3000. Структура устройства микропрограммного управления на МПК К589 показана на рис. 3.26. В состав

Структура БМПУ на основе счетчика приведена на рис. 3.27. Содержимое счетчика, т. е. адрес МК, изменяется в соответствии с командами, поступающими на логическую схему (ЛСх) из ПЗУ. Набор

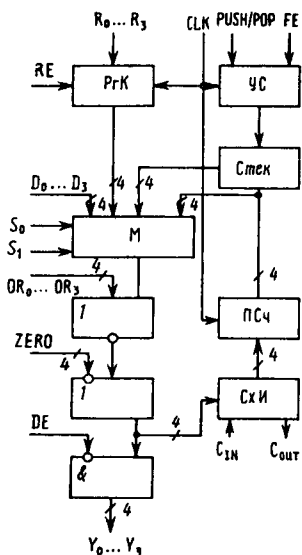


Рис. 3.28. Структурная схема БМПУ АМ2909 на основе счетчика

Структурная схема БМПУ на основе счетчика приведена на рис. 3.28. Устройство может обращаться с помощью 4-разрядной адресной шины Y к 16 ячейкам ПЗУ. Информация на эту шину поступает через мультиплексор M из программного счетчика ПСч, стека, регистра команд РРК или с входной шины $D_0 \dots D_3$. Выбор одного из четырех входов мультиплексора осуществляется из ПЗУ по управляющим входам S_0 и S_1 . Выходная информация на шине Y может быть модифицирована 4-разрядным кодом $OR_0 \dots OR_3$. Другая входная шина $ZERO$ позволяет установить нулевое значение на шине Y или на отдельных ее линиях.

Схема инкремента СХИ предназначена для увеличения на единицу кодов, передаваемых через нее на программный счетчик ПСч. Вход C_{IN} и выход C_{OUT} создают возможность каскадного соединения нескольких БМПУ в более сложные схемы. При единичном сигнале на входе C_{IN} в ПСч загружается увеличенное на единицу содержимое выходной адресной шины; в противном случае код записывается в ПСч без увеличения. Это позволяет производить безусловную передачу управления по следующему адресу.

Регистр команд РгК выполняет функции буферного регистра для входной информации, подлежащей выдаче на шину Y в качестве адреса следующей микрокоманды.

Стек состоит из 4-разрядных регистров и указателя стека (УС). УС работает как реверсивный счетчик, он имеет два управляющих ввода: PUSH/POP — записи—чтения и FE — разрешения обращения. Стек позволяет сохранять содержимое ПСч во время обращений к подпрограммам.

§ 3.7. МЕТОД ФОРМИРОВАНИЯ МИКРОПРОГРАММ

Микропрограммное управление позволяет проблемно ориентировать проектируемые микроЭВМ и МПС на конкретные области применения. Для организации микропрограммного управления в МПС необходимо определить формат управляющих слов с учетом противоречия между повышением производительности МПС за счет организации параллельной обработки информации и сокращением разрядности слова, а следовательно, и емкостью управляющей памяти.

Рассмотрим МПС, построенную в соответствии с принципами модульности, магистральности и микропрограммируемости и состоящую из некоторого множества функциональных модулей (ФМ) и информационных магистралей. Максимальное распараллеливание вычислительного процесса, а значит, и быстродействие МПС могут быть достигнуты, если микропрограммное управляющее слово (МУС) содержит поля микроинструкций всех ФМ (горизонтальное микропрограммирование). Однако при этом разрядность управляющей памяти оказывается неприемлемо большой. Наиболее рациональное использование памяти микропрограмм достигается тогда, когда управляющее слово содержит только два поля: поле микрооперации и поле, указывающее ФМ, которому предназначена микроинструкция (вертикальное микропрограммирование). При таком способе организации управления быстродействие МПС является минимальным. Задача обеспечения одновременного управления несколькими ФМ при ограниченной разрядности управляющей памяти решается путем использования смешанного микропрограммирования. В этом случае управляющее слово разбивается на фиксированное число полей. Для каждого поля назначается некоторое количество ФМ, не работающих одновременно. Разновидностью смешанного микропрограммирования

является применение плавающих форматов микроинструкций, при котором точное число полей в управляющем слове не задается.

Управление системой осуществляется последовательностью МУС длины w , определяемой разрядностью управляющей памяти, состоящей из двух полей: структуры и управления. Поле структуры задает подмножество ФМ системы, которые одновременно выполняют микрооперации, предписываемые микроинструкциями поля управления. Поле структуры может кодироваться двумя способами. Так, при позиционном кодировании оно занимает столько разрядов, сколько существует в системе микропрограммно-управляемых ФМ. Длину поля структуры можно уменьшить, используя не значение отдельных разрядов, а значение кода, записанного в этом поле.

Рассмотрим методику формирования последовательности МУС плавающего формата с ограничениями на разрядность управляющей памяти и форматы микроинструкций ФМ системы. При этом не будем учитывать поле структуры, так как для каждого МУС оно однозначно определяется полем управления.

Формальная постановка задачи. Пусть МПС состоит из множества $D = \{d_i\}$, $i = \overline{1, p}$ функциональных модулей, взаимодействующих через множество $S = \{s_i\}$, $i = \overline{1, q}$ информационных магистралей. Для каждого ФМ системы $d_i \in D$ известен набор МО $X^{(i)} = \{x_j^{(i)}\}$, $j = \overline{1, r}$ и время в тактах $t(x_k^{(i)})$, $k = \overline{1, r}$ их реализации данным ФМ.

Тогда $X = \bigcup_{i=1}^p X^{(i)}$ есть множество МО для МПС в целом. В общем случае условие $X^{(i)} \cap X^{(j)} = \emptyset$ при $i \neq j$ может не выполняться, т. е. существует некоторое перекрытие функций различных модулей и каждая МО может выполняться одним или несколькими ФМ. Например, для МПК БИС серии К583 логическая обработка байтовой информации может выполняться ФМ, построенным на основе как БИС логического микропроцессора К583ВМ1, так и БИС универсального микропроцессора К583ВС1. Для однородных МПС перекрытие функций модулей очевидно. Для микроинструкций каждого ФМ системы $d_i \in D$ известен формат F_i , который задает разрядность и положение управляющего поля (полей) в МУС.

При описании алгоритма микропрограммы на множестве $A = \{A_i\}$ $i = \overline{1, n}$ операторов преобразования информации задается отношение предшествования $<$, определяющее допустимый порядок выполнения операторов. Каждому оператору $A_i \in A$ однозначно соответствует реализующая его МО $x_i \in X$. В общем случае будем считать, что формат F_i определен для каждой микроинструкции, предписывающей реализацию МО x_i . Указанное отношение порождает ориентированный граф ограничения порядка $G = (A, U)$, где дуга $(a_i, a_j) \in U$, если для выполнения оператора A_j требуется результат оператора A_i или для реализации микроопераций x_i и x_j необходимы общие ресурсы МПС. Граф G имеет одну начальную a_1 и одну конечную a_n вершины, что соответствует описанию реальных микропрограмм, вы-

полнение которых начинается с МО перехода к начальному адресу и заканчивается загрузкой кода следующей операции или адреса перехода для линейных сегментов. Кроме того, будем считать, что граф G не содержит контуров и кратных дуг.

Требуется сформировать последовательность управляющих слов $MI = (MI_1, MI_2, \dots, MI_z)$ минимальной длины, т. е. $z \rightarrow \min$, чтобы каждое слово MI_i состояло из совместимых по формату МО. При этом под совместимыми понимают МО одновременно функционирующих ФМ, управляющие поля которых размещаются в МУС без пересечений. Если представить МУС в виде множества двоичных разрядов $F = \{f_i\}$, $i = \overline{1, \omega}$, где ω — число разрядов и элемент f_i соответствует одному двоичному разряду, то формат F_i можно описать подмножеством $F_i \subseteq F$, определяемым разрядами слова, занимаемыми полями управления МО x_i . Тогда для МО x_i и x_j , совместимых по формату микроинструкций, выполняется условие

$$F_i \cap F_j = \emptyset. \quad (3.1)$$

Формально необходимо построить последовательность MI минимальной длины, где $MI_i = \{x_k^{(i)}\}$, $k = \overline{1, h}$, чтобы выполнялись условия:

$$G^{-1}(x_s^{(i)}) \subseteq \{MI_j: j < i\}, \quad \forall x_s^{(i)} \in MI_i; \quad (3.2)$$

$$F_v \cap F_\lambda = \emptyset, \quad \forall x_v^{(i)} \in MI_i, \quad \forall x_\eta^{(i)} \in MI_i. \quad (3.3)$$

Очевидно, что для каждой пары МО, соответствующих упорядоченным вершинам графа G , т. е. вершинам, соединенным дугой или путем на графе, безразлично, являются ли они совместимыми по формату или нет, так как их размещение в одном управляющем слове невозможно. Однако для неупорядоченных вершин это утверждать нельзя. Поэтому требуется выполнение следующих условий:

$$\exists (x_v, x_\lambda) \in U \Rightarrow x_v \in MI_i, \quad x_\lambda \in MI_j, \quad i < j; \quad (3.4)$$

$$\exists \mu [x_v, x_\lambda] \Rightarrow x_v \in MI_i, \quad x_\lambda \in MI_j, \quad i = j - l(\mu), \quad (3.5)$$

где $\mu [x_v, x_\lambda]$ — путь из вершины x_v в вершину x_λ в графе G длиной $l(\mu)$.

Метод решения. Не уменьшая общности, будем считать, что алгоритм, который требуется реализовать на микропрограммном уровне, разбит на линейные блоки. Операторы, входящие в блоки, выполняются в своем естественном порядке последовательно один за другим или в порядке, определенном операторами безусловных переходов. Линейные блоки не содержат команд условного перехода, кроме одной, последней, и циклов. Поэтому будем считать, что линейный блок микропрограммы задается в виде упорядоченного множества $X = (x_1, \dots, \dots, x_N)$ микроопераций, каждая МО $x_i \in X$ определяется шестеркой $\langle OP_i, E_i, I_i, O_i, T_i, F_i \rangle$, где OP_i — идентификатор функции, выполняемой МО x_i ; E_i — элемент МПС, необходимый для выполнения x_i ; I_i — множество источников информации (регистры или магистра-

ли); O_i — множество приемников результата (регистры или магистрали); T_i — временные характеристики МО (множество машинных циклов синхронизации); F_i — множество полей МК, в которых располагаются управляющие коды и необходимые константы.

Однако строгая линейная упорядоченность МО не является обязательной. Например, некоторые МО можно поменять местами, а при наличии в МПС нескольких операционных элементов (АЛУ) — объединить в МК, иницировать (считывать из управляющей памяти) и выполнять одновременно. В связи с этим задача сводится к формированию последовательности микроинструкций $MI = (MI_1, \dots, MI_z)$, выполнение которой было бы эквивалентно (с точки зрения достижения получаемого результата) реализации множества X микроопераций в порядке их записи и осуществлялось бы за минимальное время. Микропрограммирование исходного алгоритма в целом осуществляется последовательным построением МУС его линейных участков.

Параллельное выполнение нескольких МО допускается в случае их независимости по данным и отсутствия конфликтов по ресурсам. Так, микрооперации x_i и x_j нельзя объединить в одну МК, если имеется хотя бы один из трех случаев пересечения по данным:

$$O_i \cap I_j \neq \emptyset; \quad O_i \cap O_j = \emptyset; \quad I_i \cap O_j \neq \emptyset.$$

Обозначим через V множество упорядоченных пар МО, зависящих по данным:

$$V = \{(x_i, x_j), i < j : (O_i \cap I_j) \wedge (O_i \cap O_j) \vee (I_i \cap O_j) \neq \emptyset\}. \quad (3.6)$$

Конфликты по ресурсам возникают при одновременном обращении к элементам памяти множеств I и O , при одновременном использовании элементов E и при одновременном использовании полей МК. Формально микрооперации x_i и x_j находятся в конфликте по ресурсам, если имеется хотя бы один из следующих случаев:

$$I_i \cap I_j \neq \emptyset; \quad O_i \cap O_j \neq \emptyset; \quad E_i \cap E_j \neq \emptyset; \quad F_i \cap F_j \neq \emptyset. \quad (3.7)$$

Условия (3.7) сгруппируем таким образом. В пределах одного линейного блока условие $O_i \cap O_j \neq \emptyset$ учитывается в выражении (3.6), определяющем зависимость микроопераций x_i и x_j по данным. Если учесть, что прием информации (считывание с магистралей) для МО, реализуемых более чем за один цикл синхронизации, выполняется в первом цикле, то все конфликты можно подразделить на две группы (по времени их разрешения) и представить в виде двух множеств V' , V'' конфликтующих пар микроопераций:

$$\begin{aligned} V' &= \{(x_i, x_j) | (x_i, x_j) \notin V, (x_j, x_i) \notin V, E_i \cap E_j \neq \emptyset\}; \\ V'' &= \{(x_i, x_j) \notin V' | (x_i, x_j) \notin V, (x_j, x_i) \notin V, (I_i \cap I_j) \cup (F_i \cap F_j) \neq \emptyset\}. \end{aligned}$$

Сформулируем задачу построения последовательности МУС в терминах теории расписаний. Для этого выберем в качестве переменных проектирования (искомых величин) $t(x_i)$ — время инициирования МО $x_i \in X$ или, что то же самое, номер машинного цикла, в котором МО x_i считывается из управляющей памяти. Поскольку для всех пар $(x_i, x_j) \in V$, удовлетворяющих требованию (3.6), известно, что МО x_j может быть реализована только после завершения МО x_i , должно выполняться неравенство

$$t(x_j) - t(x_i) \geq T_i, (x_i, x_j) \in V. \quad (3.8)$$

Если микрооперации x_i и x_j одновременно выполняться не могут, но отношение порядка их реализации не задано, что имеет место для пар $\{x_i, x_j\} \in V' \cup V''$, то должно выполняться одно из неравенств

$$t(x_j) - t(x_i) \geq T_i \text{ или } t(x_i) - t(x_j) \geq T_j \text{ для } \{x_i, x_j\} \in V', \quad (3.9)$$

либо

$$t(x_j) - t(x_i) \geq \tau \text{ или } t(x_i) - t(x_j) \geq \tau \text{ для } \{x_i, x_j\} \in V'', \quad (3.10)$$

где τ — длительность машинного цикла синхронизации.

Поскольку для формального представления задачи линейного программирования формы записи (3.9), (3.10) недопустимы, для модификации записей (3.9), (3.10) введем целочисленные переменные y_{ij} , принимающие значения:

$$y_{ij} = \begin{cases} 1, & \text{если } x_i \text{ иницируется раньше } x_j; \\ 0 & \text{— в противном случае.} \end{cases}$$

Тогда условия (3.9) и (3.10) можно переписать таким образом:

$$\begin{cases} M(1 - y_{ij}) + t(x_j) - t(x_i) \geq T_i, \\ My_{ij} + t(x_i) - t(x_j) \geq T_j \text{ для } \{x_i, x_j\} \in V' \end{cases} \quad (3.11)$$

и

$$\begin{cases} M(1 - y_{ij}) + t(x_j) - t(x_i) \geq \tau, \\ My_{ij} + t(x_i) - t(x_j) \geq \tau \text{ для } \{x_i, x_j\} \in V'', \end{cases} \quad (3.12)$$

где M — достаточно большое число.

Задача построения оптимальной по быстроедействию последовательности МУС в описанной постановке сводится к нахождению зна-

Пример 3.1. Управление в МПС осуществляется последовательно 32-разрядных управляющих слов. Граф микропрограммы и допустимые в системе форматы микроинструкций показаны на рис. 3.29. Так, например, для формата F_1 управляющее поле МО занимает с нулевого по 15-й разряды МУС. Длительность реализации операторов $A_1, \dots, A_{11}$ и форматы управляющих полей для соответствующих МО приведены в табл. 3.2.

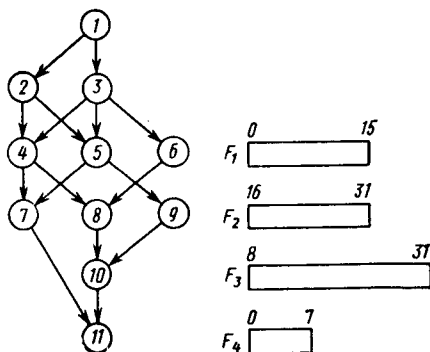


Рис. 3.29. Граф микропрограммы и допустимые форматы микроинструкций

На основе матрицы достижимости графа микропрограммы выписываются все пары неупорядоченных вершин: 2—3, 4—5, 4—6, 4—9, 5—6, 5—8, 6—7, 7—8, 7—9, 7—10, 8—9. Проанализировав эти пары на совместимость по формату МО, сформируем множество ребер $V = \{[5, 6], [6, 7], [7, 8]\}$. Результаты вычислений, определяющих ориентацию ребер из множества V и полученные в итоге дуги, представлены в табл. 3.3.

Таблица 3.2

Характеристики операторов

Параметр	Операторы										
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
F_i	F_3	F_2	F_4	F_4	F_2	F_3	F_1	F_4	F_2	F_2	F_3
t_i	1	1	1	1	1	2	1	1	1	1	2

Таблица 3.3

Результаты проектирования микропрограммы

Этап вычислений	Пометка вершины	Операторы											Дуга
		A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	
1	$T(a_i)$	0	1	1	2	2	2	3	4	4	5	6	(6,7)
	$NT(a_i)$	6	4	5	3	3	4	1	2	2	1	0	(6,5)
2	$T(a_i)$	0	1	1	2	3	2	4	4	4	5	6	(8,7)
	$NT(a_i)$	6	4	5	3	3	4	1	2	2	1	0	
3	$T(a_i)$	0	1	1	2	3	2	5	4	4	5	6	—

Результатом работы алгоритма 3.1 является граф — решение и собственно последовательность $MI = (MI_1, MI_2, \dots, MI_7)$ скомпонованных управляющих слов, которые показаны на рис. 3.30, где

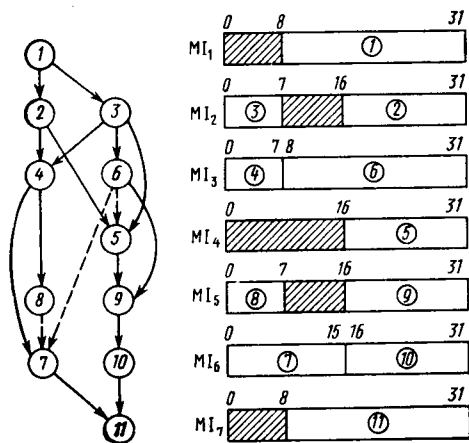


Рис. 3.30. Последовательность управляющих слов

пунктиром выделены дуги, полученные в процессе решения, и заштрихованы неиспользуемые зоны МУС.

Рассмотренный алгоритм 3.1 позволяет автоматизировать процесс синтеза последовательности микропрограммных управляющих слов для МПС при любом способе микропрограммирования: горизонтальном, вертикальном и смешанном. Такая универсальность достигается применением множества F , а способ микропрограммирования определяет подмножества $F_i \subset F$.

§ 3.8. СРЕДСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МПУ

Система разработки программного обеспечения. Разработка программного обеспечения микроЭВМ или МПУ может производиться либо на микроЭВМ и микропроцессорных устройствах, для функционирования которых это программное обеспечение создается, либо на универсальных вычислительных средствах.

Программное обеспечение (ПО), реализованное на той же микропроцессорной системе, что и вновь разработанное, или на программно-совместимой с ней системе, называется *резидентным* ПО.

Программное обеспечение, реализованное на микропроцессорных системах, не совместимых программно с той системой, для которой ведется разработка, а также на универсальных ЭВМ, называют *кроссовым* ПО разработки.

Кроссовое ПО часто реализуется на широко распространенном языке программирования, например на языке ФОРТРАН или ПЛ/1, что обеспечивает высокую мобильность этих программ.

Целесообразность применения резидентного или кроссового ПО разработки определяется такими факторами, как требования к разрабатываемому МПУ, стоимость резидентного и кроссового ПО, стоимость дополнительной аппаратуры для использования резидентного ПО, наличие доступной ЭВМ, пригодной для постановки кроссового ПО, и т. д.

К достоинствам кроссового ПО относят: низкую начальную стоимость; возможность использования развитого сервисного ПО; возможность разработки ПО до появления аппаратурных средств.

К достоинствам резидентного ПО разработки относят: низкую общую стоимость использования резидентных средств за счет низкой стоимости микропроцессорных средств по сравнению с ЭВМ других классов; невозможность во многих случаях полностью смоделировать на другой ЭВМ реальные временные отношения и условия внешнего окружения МПУ; отсутствие необходимости освоения разработчиками работы на двух машинах: на ЭВМ, реализующей ПО, и на самой микропроцессорной системе.

На практике обычно выбирают компромисс в использовании кроссовых и резидентных программных средств, например трансляцию программ — на кроссовом трансляторе, а отладку — с помощью резидентных средств отладки на прототипной или разрабатываемой микропроцессорной системе.

Отечественные кросс-системы. Как отмечалось, кросс-системы отличаются от резидентных систем тем, что архитектура инструментальной ЭВМ не совпадает с архитектурой разрабатываемого МПУ. Различают кросс-системы, работающие: на больших универсальных ЭВМ или миниЭВМ под управлением мощных, хорошо развитых операционных систем, на мощных автономных микроЭВМ общего назначения; на автономных мощных микроЭВМ, соединенных в локальную сеть с возможностью разделения общих ресурсов.

Цикл жизни ПО можно подразделить на следующие этапы: анализ и спецификация проблемы, проектирование (декомпозиция на модули и алгоритмизация), реализация (кодирование, отладка, тестирование, документирование) и эксплуатация. Статистика показывает, что затраты на разработку ПО составляют около 40 %, затраты на эксплуатацию — около 60 % от общих затрат.

Возможности отечественных кросс-систем отражены в табл. 3.4. Все рассматриваемые кросс-системы имеют библиотеки и наборы тестов, снабжены отладочными средствами, имеют возможность документировать программы и обеспечивают только этап реализации программ. Основное внимание во всех кросс-системах уделяется трансляции и автономной отладке модулей. Основной документацией, получение которой автоматизировано, являются распечатки программных модулей. Выпуск эксплуатационной документации и документирование процесса разработки поддерживается только в четырех системах. Однако следует учитывать, что существует множество самостоятельных систем подготовки документации, работающих в ОС ЕС, которыми можно пополнять функции кросс-систем.

Большинство кросс-систем работают на ЕС ЭВМ в пакетном режиме (табл. 3.5). Рассматриваемые системы написаны на ассемблере или ПЛ/1, что делает практически невозможным их перенос на другую ЭВМ или операционную систему.

Таблица 3.4

Технологические возможности кросс-систем

Тип кросс-системы	Трансляция программ	Редктирование связей модулей	Отладочное выполнение модулей	Выпуск эксплуатационной документации	Подготовка тестов	Тестирование функционирования программы	Тестирование временных характеристик программы	Документирование процесса реализации	Тестирование совместной работы аппаратуры и программы	Настройка на конфигурацию	Настройка на тип микропроцессора	Перевод программы в МПУ или микроЭВМ
КАС580/МОСК580	а		М									Л
СППМ-580	а		П									
САПО-«Э60»	ма	+	М			М						Л
CSDS/580	я	+	М			М						
КС 580/ЛГУ	я	+	М			М						
КОМПРИС	я		М									Л
ИСПО ПО АСНИ	я	+	М			М	М		М	+		
П 4.4	я		М									
КОСА	ма		М							+	+	
КОМ-2	я	+	М			М	М			+		
ТЕМП	ма	+	М	+		М		+		+	+	
СЕРП	ма	+	М		+	М	М	+	М	+	+	Л
АРМ 2-05	а (я)	+	П/М	+		П/М	П		П	+	+	Н
ПРА	ма		М	+	+		М	+				Л
РУЗА	ма	+	М	+	+	М	М	+			+	Л

Примечание. а — ассемблер; ма — макроассемблер; я — язык среднего или высокого уровня; М — программные модели аппаратуры; П — непосредственно на аппаратуре; Л — перфолента; Н — непосредственное управление программатором ПЗУ.

Сводные данные о характеристиках разрабатываемых программ для проектирования микро-ЭВМ и микропроцессорных устройств приведены в табл. 3.6.

По видам применения МПС подразделяют на:

автономные микроЭВМ (конструктивно законченные вычислительные устройства с собственно традиционной периферией);

встроенные МПУ (вычислительные устройства, являющиеся неотъемлемой частью технологического оборудования);

Таблица 3.5

Технические характеристики кросс-систем

Тип кросс-системы	ОС	Инструментальная ЭВМ	Объем ОЗУ, К байт	Режим работы	Объем программы кросс-системы К байт	Язык программирования	Наличие документации
КАС580/МОСК580	ОС ЕС	ЕС ЭВМ	128	Пакетный	130	ПЛ/1	+
СППМ-580	ОС ЕС	ЕС ЭВМ				Ассемблер	+
САПО-«Э60»	ОС ЕС	ЕС ЭВМ				Ассемблер, ФОРТРАН	+
CSDS/580	ОС ЕС	ЕС ЭВМ			170	Ассемблер, ФОРТРАН	+
КС 580/ЛГУ	ОС ЕС	ЕС ЭВМ	256	»	400	Ассемблер, ПЛ/1	ЕСПД
КОМПРИС	ОС ЕС	ЕС ЭВМ	120	»	200	То же	ЕСПД
ИСПО ПО АСНИ	ДОС ЕС, ОС ЕС	ЕС ЭВМ	256	Пакетный, диалоговый	1000	Ассемблер, ПЛ/1	
П4.4	ДОС СМ	СМ-3 СМ-4	48	»	94	Ассемблер	
КОСА	ОС ЕС	ЕС ЭВМ	256	Пакетный	200	»	
КОМ-2	ДОС АСВТ	М4030.1	192 Макс. 300	»	270	»	+
ТЕМП	ДИСПАК	БЭСМ-6				Автокод	+
СЕРП	ОС ЕС, ОС РВ	ЕС ЭВМ				Ассемблер	ЕСКД
АРМ 2-05	СПО СМ1800	СМ-4, СМ1800			500	Ассемблер	ЕСПД
ПРА	РАФОС	СМ1800, ВУМС025	64	Диалоговый, пакетный Диалоговый, пакетный	250	Ассемблер	ЕСПД, ЕСКД ЕСКД
РУЗА	ОС 6.1	ЕС ЭВМ	300	То же	500	»	ЕСПД

Характеристики разрабатываемых программ

Тип кросс-системы	МПК микроЭВМ	Вид применения	Характеристика ПО	Язык программирования	Используемые языки
КАС580/МОСК580	К580	Контроллер	Общего назначения	Ассемблер	ЯУЗ ОС ЕС
СППМ-580	К580	»	То же	»	ЯУЗ ОС ЕС, управление отладчиком
САПО-«Э60»	«Электроника 60»	Автономная микроЭВМ	»	«Ассемблер, «Электроника 60», макрогенератор	ЯУЗ ОС ЕС, управление отладчиком, управление редактором связей
CSDS/580	К580	Контроллер, автономная микроЭВМ	»	Макроассемблер 8080	То же
КС 580/ЛГУ	К580	»	»	Макроассемблер К580/ЛГУ ассемблер 8080, АССОЛ/М	ЯУЗ ОС ЕС, ассемблер ЕС ЭВМ, управление редактором связей
КОМПРИС	«Электроника 60»	»	»	ПАСКАЛЬ	ЯУЗ ОС ЕС, управление отладчиком
ИСПО ПО АСНИ	«Электроника 60»	»	Реального времени	Макроассемблер, «Электроника 60», ПЛ/П	ЯУЗ ОС ЕС, управление отладчиком, управление редактором связей
П4.4	К580	Встроенная микроЭВМ	»	Ассемблер	ЯУЗ ДОС СМ, управление редактором связей, управление отладчиком

КОСА	К580, К584, «Электроника С5»	»	Общего назначения	Настраиваемый макроассемблер (оптимизирующий)	ЯУЗ ОС ЕС, управление редактором связей, управление отладчиком, моделирование
КОМ-2	СМ 1800	»	Общего назначения, реального времени	Ассемблер, ПЛ/М, БЭЙСИК, ПАСКАЛЬ	ЯУЗ ДОС АСВТ, управление редактором связей, управление отладчиком
ТЕМП	К580	»	Общего назначения	Макроавтокод	ЯУЗ ДИСПАК, отладки, управление с информационной базой
СЕРП	К580, К581, К1801, К1810	»	»	Макроассемблеры 8080, МАС-11, К1801, К1810ВМ86	Отладки редакторских связей, моделирование
АРМ 2-05	К580 СМ1800	»	Общего назначения, реального времени	Ассемблер, ПЛ/М, БЭЙСИК, ПАСКАЛЬ	Метаязыки, язык документирования, диалоговый управления
ПРА	К580, К1801, К1810	»	»	Ассемблер	СУБД, отладки
РУЗА	К580, К1810, К1801	»	»	Ассемблер, ПЛ/М	СУБД, отладки, спецификаций

контроллеры (функционально самостоятельные узлы вычислительной или технологической аппаратуры с узкоспециализированными функциями управления).

Программное обеспечение разделено на ПО общего назначения и ПО реального времени (ПО, для которого предъявляются жесткие требования по временным характеристикам).

Следует отметить, что в настоящее время основной инструментальной ЭВМ является ЕС ЭВМ с ОС ЕС. Реальной альтернативой кросс-системы на больших инструментальных ЭВМ общего назначения являются инструментальные системы на мощных микроЭВМ и персональных ЭВМ, соединенные в локальную сеть с возможностью разделения дорогих периферийных устройств.

Программные средства разработки ПО. В соответствии с выполняемыми функциями ПО разработки может быть подразделено на четыре основных класса программных средств: трансляторы, загрузчика, системы отладки, сервисные программы.

Трансляторы преобразуют исходную программу, написанную на некотором входном языке — языке ассемблера, высокого уровня или микроассемблера, в результирующую, так называемую объектную программу на языке команд или микрокоманд МПУ или МПС. Общий вид процесса трансляции и выполнения программы описан в § 3.1 (см. рис. 3.4). Формируемый транслятором объектный модуль затем обрабатывается редактором связей и загрузчиком, который настраивает его на конкретные адреса и удовлетворяет его внешние ссылки — содержащиеся в программе данного модуля указания данных и команд, расположенных в других модулях.

Загрузчики — совокупность программ, переводящих объектные программы к виду, непосредственно готовому к выполнению на микропроцессорной системе, и заносящих эти программы в память МПС.

Преобразование объектной программы к виду, непосредственно готовому к выполнению, состоит в преобразовании перемещаемого варианта объектной программы в вариант программы в абсолютных адресах. Программа, реализующая такое преобразование, называется *размещающей программой*.

Процесс установления связей между отдельными объектами и программными модулями называется редактированием внешних связей, а производящая эти действия программа — *редактором связей*. Если при этом еще происходит объединение объектных модулей в единый модуль, то реализующая это объединение программа называется *объединителем*.

Программы-отладчики предназначены для облегчения отладки ПО. Отладчики могут быть кроссовыми и резидентными, причем, как правило, они являются диалоговыми системами.

Основные функции отладчика:

вывод на печать или пульт оператора содержимого задаваемой области памяти и регистров процессора;

изменение содержимого оперативной памяти или регистров процессора;

запуск программы на выполнение с указанного адреса;

приостановка выполнения программы по достижении одной из заданных команд или при выполнении заданного условия.

Чтобы кроссовое программное обеспечение разработки позволяло эффективно проводить работы по отладке, в его состав должна входить система моделирования и отладки, состоящая из имитатора микропроцессорной системы и программы управления отладкой.

К сервисным программам относятся программы, облегчающие подготовку исходной информации, создание документации, организацию хранения и использования отлаженных программ. В состав сервисных программ в обязательном порядке входят редакторы, библиотекари и программы преобразования форматов.

Редактор — обслуживающая программа для редактирования набора данных с целью представления их в виде, воспринимаемом средствами обработки, либо в соответствующем формате вывода. Редактор принимает исходную программу, вводимую с клавиатуры или некоторого носителя, и записывает в оперативную либо внешнюю память системы. Редактор оперирует с исходной программой как с текстом, не учитывая синтаксические правила, которые она должна удовлетворять.

Для хранения и использования отлаженного ПО служат библиотеки исходных и объектных программ. Программы, обеспечивающие организацию библиотек и работу с ними, носят название *библиотечарей*. В резидентных системах разработки ПО в библиотеках хранятся объектные программные модули. В кроссовых системах ПО разработки в библиотеки объединяются как исходные программы, так и объектные программные модули.

Программы *преобразования форматов* обеспечивают информационную совместимость компонентов систем программирования, реализованных на различных машинах или в рамках одной системы на разных этапах ее развития, а также информационную совместимость систем программирования и программного обеспечения (например, монитора), разрабатываемых МПС. Напомним, что *монитором* называется записанная в ПЗУ системная программа, реализующая операции обмена с внешними устройствами и помогающая осуществить отладку программ.

§ 3.9. МЕТОДЫ И СРЕДСТВА АВТОМАТИЗАЦИИ МИКРОПРОГРАММИРОВАНИЯ

Микропрограммное управление позволяет проблемно ориентировать разрабатываемые микроЭВМ и МПУ на целевые задачи конкретной области применения.

В связи с расширением функциональных возможностей встроенных МПУ объем их микропрограмм достигает нескольких тысяч микрокоманд и намечается тенденция его дальнейшего роста. На проекти-

рование микропрограммного обеспечения МПУ затрачивается до 90 % общего времени их проектирования. К эффективности микропрограмм и к срокам их разработки предъявляются довольно жесткие требования. Поэтому разработка микропрограмм больших объемов невозможна без применения кросс-систем автоматизации микропрограммирования, использующих более мощные ресурсы инструментальных ЭВМ.

Синтез микропрограмм — одна из наиболее сложных задач в процессе проектирования микроЭВМ и МПУ. Формально задача синтеза состоит в построении множества вариантов реализации микропрограмм и выборе лучшего из них по заданному критерию. Естественно, такой путь полного перебора неприемлем, даже если удастся формально синтезировать варианты, поскольку количество вариантов построения достаточно сложных микропрограмм для структуры среднего объема практически бесконечно.

Для синтеза микропрограмм необходимо построить формальную модель обобщенного вычислительного процесса, которая позволяла бы осуществить либо направленный перебор вариантов микропрограмм, либо синтез оптимального (или рационального) варианта посредством целенаправленного выбора действий в процессе синтеза.

При решении задачи синтеза микропрограмм требуется получать дополнительную информацию для выработки рекомендаций по изменению структуры, поскольку по синтезированной микропрограмме сложно определить, в каких именно местах структуры возникают конфликты при распределении общих ресурсов системы. Кроме того, при синтезе микропрограмм нужно иметь дополнительные средства, обеспечивающие «расширение» описания микропрограмм за счет введенных ранее операторов.

Для определения качества синтезированных микропрограмм используют методы моделирования. Моделирование реализации микропрограмм предполагает имитацию выполнения микропрограмм на выбранном варианте структуры. Система исследований должна обеспечивать как физическое, так и математическое моделирование. При физическом моделировании микропрограмма реализуется на реальной структуре. Основными вопросами, которые должны быть решены при физическом моделировании, являются потактовое выполнение микрокоманд и произвольных микроопераций, имитация микрокоманд и микроопераций, индикация по требованию пользователя любого набора из доступных регистров. При математическом моделировании в качестве исходных данных должны использоваться описания микропрограмм и структуры на одном из языков проектирования и моделирования дискретных систем. Результатом математического моделирования должно быть заключение о правильном либо неправильном выполнении микропрограмм.

Диагностика микропрограмм может представляться либо в качестве отдельной задачи, либо совмещаться с этапом математического моделирования. Цель диагностики — определение и указание ошибок

ных микрокоманд и микропрограмм. Здесь же разрабатываются диагностические тесты микропрограмм.

Оценка качества функционирования системы при выбранном наборе микрокоманд предполагает определение значений ряда показателей, таких, как реальная производительность системы на заданном множестве операторов или задач, требуемый объем микропрограммной памяти, статистические характеристики использования оборудования системы, стоимость системы, «узкие» и «широкие» места в системе, т. е. перегруженные элементы системы, сдерживающие увеличение ее реальной производительности, и недогруженные элементы системы. Некоторые из этих показателей используются как жесткие ограничения, определяющие возможность дальнейшего рассмотрения того или иного варианта структуры, другие — для оценки эффективности структуры.

Изменение структуры по результатам моделирования предполагает выявление элементов, введение которых в структуру существенно увеличивает ее производительность при синтезе наилучших для вновь полученной структуры микропрограмм.

Автоматизация микропрограммирования делает необходимым применение для исходного описания микропрограмм языков, подобных языкам высокого уровня. Такой язык должен отличаться от языка ассемблера большей наглядностью и простотой описания алгоритмов микропрограмм.

На систему автоматизированного проектирования микропрограмм возлагаются функции синтаксического и частично семантического контроля микропрограмм, компоновки микроопераций по микрокомандам (которая непосредственно влияет на время выполнения машинных операций МПУ и микроЭВМ с микропрограммным управлением), формирования двоичных микрокодов (трансляции) и документирования.

Характеристики ряда отечественных и зарубежных систем автоматизации микропрограммирования МПУ приведены в табл. 3.7. Кратко рассмотрим особенности известных отечественных систем.

Система автоматизации ТЕМП предполагает обеспечение полного цикла проектирования программ и микропрограмм для определенного класса микроЭВМ, а также адаптацию к особенностям конкретной машины из этого класса и связанным с ней конкретным условиям применения.

В качестве обобщающего принципа, с помощью которого достигается универсальность системы, используется принцип параметризации, при котором конкретные условия работы представляются в виде параметров (переменных). Конкретные условия, в которых работает система ТЕМП, представлены следующими классами данных:

данные, характеризующие архитектуру разрабатываемого комплекта управляющих программ;

данные, определяющие ассемблер, применяемый для программирования или микропрограммирования (микроассемблер);

Таблица 3.7

Основные характеристики систем автоматизации микропрограммирования МПУ

Наименование системы	Инструментальная ЭВМ	Язык реализации	Ориентация систем (комплект БИС или настраиваемая)	Язык микропрограммирования	Разделение фаз настройки и трансляции	Решение задачи размещения МК в ЗУМП	Максимальный объем памяти МК, бит
CROMIS AMDASM		ФОРТРАН, ПЛ/М-80 ПЛ/1	13000 (K589) AM 2900 (KPI804) AM 2900 (KPI804) K589	Микроассемблер »	Нет Есть	Есть »	64 128
DARL		ПЛ/М-80	Настраиваемая »	Четырех уровней Микроассемблер »	Нет	Нет	256
МИКРАС Система 589 АВТПРОМ ASMP	МикроЭВМ ЕС ЭВМ ЕС ЭВМ ЕС ЭВМ			Ассемблер ЕС ЭВМ, микроассемблер »	Есть Нет Есть »	» Есть Нет »	128 128 256
САМП	ЕС ЭВМ	ПЛ/1, Ассемблер	»	»	»	Есть	
ТЕМП	БЭСМ-6		»	Ассемблер, микроассемблер Микроассемблер »	»	Нет	
ГНОМ/ФЛОТ МЕТАМИКРО	СМ-4 ЕС ЭВМ, СМ ЭВМ, «Электроника 60» ЕС ЭВМ, СМ-1600	ФОРТРАН	» »		» »	Есть Нет	256 128
НЯМУНАС-91			»	Микроассемблер, язык высокого уровня МИКАЛУ Микроассемблер	»	Есть	
МИАСС	СМ-ЭВМ	СИ	»		»	»	
АСПРОМ	М4030	Метаассемблер МАКРО-Ф	»	МИКРОКОД	»	Нет	

данные, описывающие архитектуру МП или микроЭВМ, т. е. ресурсы (память, регистры, разрядность), структуру (типы устройств памяти, форматы команд и микрокоманд), систему команд и микрокоманд;

данные, определяющие конфигурацию технологической системы, т. е. количество, размеры и размещение ее архивов, библиотек и т. п.

Система ТЕМП реализована на ЭВМ БЭСМ-6. Имеются две версии, предназначенные для проектирования программ (ТЕМП-П) и микропрограмм (ТЕМП-М) микроЭВМ. Версия ТЕМП-М предназначена для автоматизации проектирования микропрограмм и настроена на микрокоманды микропроцессора КР589.

Основные функции, автоматизированные в системе ТЕМП-М при проектировании микропрограмм, сведены к накоплению, хранению и корректировке текстов микропрограмм в различной форме, настройке и трансляции микропрограмм. Для отладки микропрограмм имеется разработанный интерпретатор, позволяющий проверять использование микропрограмм и исследовать процесс обработки микропрограммного управления на ЭВМ БЭСМ-6.

В автоматизированной системе микропрограммирования (АСПРОМ) реализована идея метаассемблера. Применение в системе метаассемблера является компромиссом между требованием независимости системы от микропрограммной базы и использованием в системе языка, ориентированного на конкретную микропрограммную базу. Основными функциями АСПРОМ являются ведение архива, кодирование, анализ (в том числе имитационное моделирование) и документирование микропрограмм. АСПРОМ обладает возможностью простой настройки на различные МПК БИС. Система включает в себя входной язык микропрограммирования, архив данных с программными средствами доступа и обслуживания, подсистемы трансляции (кодирования), документирования и моделирования. В качестве входного используется язык символического микропрограммирования МИКРОКОД к основным особенностям которого можно отнести символическую адресацию, возможность задания указаний о размещении микрокоманд в микропрограммной памяти, а также графических координат для выпуска схем алгоритмов микропрограмм.

Каждая символическая микрокоманда языка МИКРОКОД описывает одну микрокоманду. Язык МИКРОКОД является универсальным в том смысле, что обладает сменным синтаксисом и семантикой. Под синтаксисом понимается количество полей символической микрокоманды, их длина и относительное расположение на бланке описания, а также перечень допустимых микроопераций и условий для соответствующих полей. Семантика определяет правила отображения символических микропрограмм в кодовую микропрограмму в микропрограммной памяти.

Информация, находящаяся в архиве системы, позволяет выпускать весь комплект конструкторской документации на микропрограммную память, а именно: таблицы прошивки, определяющие правила изго-

товления матриц, таблицы размещения, отображающие заполнение ячеек микрооперациями; таблицы содержимого, отображающие шестнадцатеричные коды полей микрокоманд и их мнемонические эквиваленты всех ячеек матриц и др.

АСПРОМ включает в себя следующие программные комплексы:

ТРАНСЛЯТОР — центральный комплекс в системе, осуществляющий перевод символических микропрограмм, составленных на языке МИКРОКОД, в микропрограмму. Микропрограмма в дальнейшем размещается в памяти микрокоманд. Этот же комплекс производит синтаксический и частичный семантический контроль программ. Параметрическое задание синтаксиса и семантики языка осуществляется на метаязыке МАКРО-Ф;

АНАЛИЗ, осуществляющий полный семантический контроль корректности микропрограмм;

МОДЕЛИРОВАНИЕ, предназначенное для выявления ошибок функционального характера; основой этого комплекса является система функционального моделирования интерпретационного типа ФОРΟΣ, описывающая аппаратные средства на уровне регистровых передач;

НАСТРОЙКА и УПРАВЛЕНИЕ, служащие для настройки системы, формирования и обслуживания информационных массивов архивного и рабочего типов;

ДОКУМЕНТИРОВАНИЕ, обеспечивающее выпуск эксплуатационных, конструкторских и технологических документов.

Система АСПРОМ реализована на ЭВМ М4030 и функционирует под управлением монитора ДОС АСВТ. Недостатками системы являются ограничение задач только микропрограммированием (без учета конкретных аппаратных средств) и отсутствие трансляции программ в микропрограммы.

Настраиваемый инструментальный комплекс МЕТАМИКРО предназначен для автоматизации проектирования МПС на секционных МП. В состав комплекса входят четыре основных компонента:

программа-генератор трансляторов;

настраиваемый транслятор с языка символьного микропрограммирования;

программный монитор для управления отладкой микропрограмм и аппаратуры в реальном масштабе времени;

аппаратно-программный стенд (АПС) для отладки микропрограмм в реальном масштабе времени.

Генерация трансляторов и собственно трансляция осуществляются на кроссовой ЭВМ (ЕС ЭВМ или СМ ЭВМ). Окончательная отладка производится с помощью АПС, выполненного на базе микроЭВМ «Электроника-60».

Работа с инструментальным комплексом МЕТАМИКРО состоит из трех этапов:

1. **Н а с т р о й к а** н а с и м в о л ь н ы й я з ы к. Язык разрабатывается пользователем и записывается по формальным прави-

лам, с тем чтобы система могла автоматически сгенерировать с него транслятор. Сгенерированный транслятор является самостоятельным программным продуктом.

2. Написание и трансляция микропрограмм. С помощью транслятора в процессе его работы ведется синтаксический контроль микрокоманд. Результатом работы транслятора является перфолента с двоичными кодами микропрограмм.

3. Отладка микропрограмм и аппаратуры на АПС. Стенд работает под управлением монитора, занимающего 8 К байт ОЗУ микроЭВМ, и позволяет использовать микропрограмму в четырех режимах;

автоматическом с пуском по заданному адресу;

автоматическом с пуском и остановом по заданным адресам;

автоматическом с пуском по заданному адресу и остановом по событию;

пошаговом, начиная с заданного адреса.

В распоряжении оператора АПС имеется 13 директив, позволяющих эффективно отлаживать микропрограммы и аппаратуру.

Микрокоманда, описываемая в комплексе МЕТАМИКРО на символьном языке микропрограммирования, состоит из полей, отделенных друг от друга разделителями `__` (пробел) или `!`. Число полей в микрокоманде не превышает 40, длина одного поля — 31 бит, длина всей микрокоманды — 128 бит. Допускаются поля переменных (VAR), числовые поля (DIG) и поля адреса перехода (SYM) по символьной метке или текущему значению счетчика команд.

Однозначное задание формата микрокоманд в языке символьного микропрограммирования определяется такими параметрами:

длина микрокоманды;

число полей;

порядок следования полей;

символическое имя каждого поля.

На основе указанного описания генерируется транслятор, используемый для трансляции микрокоманд.

Система функционально-логической отладки МПС (ФЛОТ) предназначена для разработки и отладки микропрограммного обеспечения МПС. Возможности системы:

1) определение форматов микрокоманд с указанием набора полей, количества разрядов в них и значения, принимаемого по умолчанию;

2) выделение субполей в рамках заданных полей формата микрокоманды и задание их мнемонического обозначения;

3) генерация по символической микропрограмме объекта микрокода, готового для загрузки в ПЗУ;

4) задание схемы соединений БИС МПС;

5) автоматическое формирование программной модели технических средств на базе библиотеки функциональных описаний БИС;

6) отладка микропрограмм на программной модели с указанием режима отладки и перечня выводимой информации. Отладка микро-

программ осуществляется в диалоговом режиме на микроасsemblerе либо в объектном коде.

По своей организации программа моделирования является таблично управляемым интерпретатором, что позволяет легко модифицировать схему технических средств.

В качестве языкового средства для отладки микропрограммного обеспечения в системе ФЛОТ используется универсальный микроасsembler ГНОМ (генератор объектного микрокода), реализованный на мини-ЭВМ СМ-4 и построенный по принципу настройки на микроархитектуру технических средств.

Микроасsembler ГНОМ применяется при переводе символических команд в объектные.

Результатом трансляции символических микропрограмм являются объектные микрокоды, готовые к записи в управляющее ПЗУ микропроцессорной системы. Процессы настройки и трансляции выполняются раздельно (после одной настройки трансляция микропрограмм может производиться многократно). Максимальная разрядность слова 256 бит, микрокоманда 1 ... 7 слов, идентификаторы могут содержать до 128 символов (первый символ — латинская буква).

Универсальный микроасsembler ГНОМ может использоваться при разработке микропрограмм для микропрограммируемых секционных микропроцессоров любой серии.

Отладка микропрограмм осуществляется как на технических средствах (макете), так и на программной модели. При этом обязательной является совместная проверка на макете микропрограммного обеспечения и технических средств.

Отладка на программной модели производится на ранних стадиях проектирования. При составлении программной модели применяется автоматическое формирование модели технических средств в соответствии со схемой соединений на основе библиотеки функциональных описаний микропроцессорных БИС.

Система ФЛОТ реализована на мини-ЭВМ СМ-4 и функционирует под управлением операционной системы РАФОС.

Система автоматизации микропрограммирования (САМП) предназначена для автоматизации процессов кодирования микропрограмм, размещения их в микропрограммной памяти и получения технической документации.

САМП состоит из следующих элементов:

- языковых средств;

- блока настройки, выполняющего процедуру генерации транслятора САМП;

- транслятора микропрограмм, осуществляющего перевод операторов входного языка микропрограммирования на язык машинных микрокодов;

- архива символических микропрограмм и машинных микрокодов; средств отладки микропрограмм.

Язык символического микропрограммирования состоит из двух

частей. Первая часть включает директивы управления размещением микропрограмм в управляющей памяти, представления микрокодов в виде констант, организации ветвлений. Вторая (переменная) часть включает операторы описания символических микрокоманд. Структура этих операторов определяется грамматикой входного языка, выбранного пользователем.

Микропрограммы, представленные на языке символического кодирования, транслируются в последовательность машинных микрокоманд (микрокодов), готовых для загрузки в микропрограммную память. Преобразование микропрограмм в последовательность микрокодов включает этапы синтаксического контроля исходных операторов, перекодировки микрокоманд и обработки директив, размещения микрокоманд в микропрограммной памяти и заполнение полей управления переходом, формирование файла, представляющего собой копию микропрограммной памяти (архива машинных микрокодов), и вывод документации.

Система САМП работает под управлением ОС ЕС на ЭВМ ЕС с объемом оперативной памяти не менее 384 К байт. Программы САМП написаны на языке ПЛ/1 с применением подпрограмм на ассемблере.

Микроассемблер МИКРАС позволяет увеличить наглядность микропрограмм путем объединения связанных по смыслу отдельных полей МК в единое символическое выражение при сохранении уровня языка. Едиными выражениями задаются поля МК, управляющие подачей информации на блок центрального процессора и ее обработкой в этом блоке, и поля, управляющие формированием адреса следующей МК. Нестандартные поля МК, реализующие управление другими элементами процессора, описываются при настройке МИКРАС на конкретную структуру МПС.

Кросс-система НЯМУНАС-91 реализует две альтернативные технологии микропрограммирования, использующие микроассемблер и язык высокого уровня МИКАЛУ, разработанный на базе языка ПЛ/1. Микропрограмма синтезируется из готовых фрагментов, находящихся в каталоге типовых микропрограмм. При генерации объектных микропрограмм каждый оператор языка МИКАЛУ переводится в последовательность микроопераций, стандартизованную для описанной структуры МПС и минимизирующую время выполнения оператора. Полученная последовательность МО, реализующая исходную микропрограмму, используется для непосредственного формирования МК, причем ограничением является заданное распределение МО по полям управляющего слова.

Применение языков высокого уровня для целей автоматизации микропрограммирования обеспечивает сокращение сроков разработки и отладки микропрограммного обеспечения проектируемых вычислительных структур в 2 ... 2,5 раза по сравнению с использованием микроассемблеров. Однако недостаточно высокое качество получаемых объектных микропрограмм (по сравнению с микропрограммами, получаемыми с применением языков ассемблера, длина микропрограмм

увеличивается на 30 ... 40 %) не позволяет использовать языки высокого уровня для разработки микропрограммного обеспечения специализированных МПС, работающих в условиях жестких временных ограничений. Этим объясняется тот факт, что все рассмотренные системы автоматизации микропрограммирования ориентированы на применение языков микроассемблера.

Таким образом, системы автоматизации микропрограммирования МПУ и МПС в общем случае реализуют следующие функции: трансляцию символических микрокоманд в объектные коды, размещение МК в адресном пространстве управляющей памяти, отладку микропрограмм и выпуск документации на управляющую память.

Контрольные вопросы и задания

1. Опишите основные функции операционных систем микроЭВМ.
2. Зачем и когда применяется программа ассемблера?
3. Объясните необходимость многообразия типов данных.
4. С помощью каких процедур осуществляется обработка массивов?
5. Приведите структурную схему программы поиска минимального элемента двумерного массива.
6. Объясните физику организации доступа к разнородным и последовательным файлам.
7. В чем заключаются основные преимущества программирования на языке ассемблера?
8. Объясните содержание работ на различных этапах проектирования программ.
9. Объясните смысл термина «технология программного обеспечения».
10. Объясните преимущества структурированных программ.
11. Какие факторы влияют на выбор способа кодирования микрокоманд?
12. Объясните назначение стека и его организацию в схеме БМПУ на основе счетчика.
13. Какие ресурсы МПС влияют на эффективность формирования микропрограмм?
14. Объясните целесообразность выбора кроссовых и резидентных систем при разработке ПО МПС.
15. Объясните основные функции загрузки.
16. Объясните назначение и способ организации редактора связей.
17. Чем объясняется многообразие систем автоматизации программирования МПС?
18. Почему языки высокого уровня неэффективны в системах автоматизации микропрограммирования?

ОТЛАДКА МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ

§ 4.1. НЕОБХОДИМОСТЬ ОТЛАДКИ АППАРАТНЫХ
И ПРОГРАММНЫХ СРЕДСТВ МПУ

Этапы разработки МПУ. Микропроцессорные средства, входящие в состав РЭС и технологического оборудования, могут существенно отличаться друг от друга сложностью аппаратуры и объемом программных средств. В связи с этим процесс проектирования МПУ может иметь свои особенности в зависимости от роли его функционирования в системе.

Процесс создания МПУ и МПС характеризуется тесной взаимосвязью этапов разработки аппаратного и программного обеспечения и сложностью отладки программ при ограниченном наборе оборудования.

На *первом этапе* (этапе составления ТЗ) производится формализация требований к системе. При этом определяются все внешние связи разрабатываемых микропроцессорных средств, перечисляются функции системы, формализуется ТЗ на систему. На этом этапе возможны ошибки из-за несогласованности требований к системе и неточностей алгоритмов функционирования МПУ.

На *втором этапе* производится разработка структуры и архитектуры системы, а также распределение функций между аппаратурными и программными средствами. Выбираются микропроцессорные наборы, на базе которых будет реализована система, определяются взаимодействия между аппаратурными и программными средствами. На этом этапе ошибки в проекте могут быть из-за несогласованности протокола взаимодействия аппаратуры и программ, неверного выбора микропроцессорных наборов, неверной интерпретации технических требований и др.

На *третьем этапе* производится разработка и изготовление аппаратурных средств и программного обеспечения системы. После определения функций, реализуемых аппаратурой, и функций, реализуемых программами, схемотехники и программисты приступают к работам.

Проектирование аппаратных средств включает в себя разработку подробной логической схемы каждого модуля, построение временных диаграмм управляющих сигналов, конструирование каждого модуля на плате, тестирование схем путем прослеживания сигналов и сравнения их с логическими схемами и временными диаграммами и сборку модулей в законченную систему, называемую прототипом.

Разработка программ состоит из разработки алгоритмов, декомпозиции общей программы на подпрограммы, разработки структурной

схемы каждого программного модуля, написания программ на языке ассемблера или языке высокого уровня, трансляции с помощью ассемблера или компилятора в объектные модули, создания загрузочных модулей и их автономной отладки, объединения модулей в законченный программный модуль и его отладки.

Основные ошибки, возникающие на этом этапе, следующие. При разработке аппаратуры возможны упущения некоторых функций, недоработка в схемах синхронизации, нарушение правил проектирования, при разработке программных средств — упущения и неверная интерпретация отдельных функций, неточности в алгоритмах и кодировании.

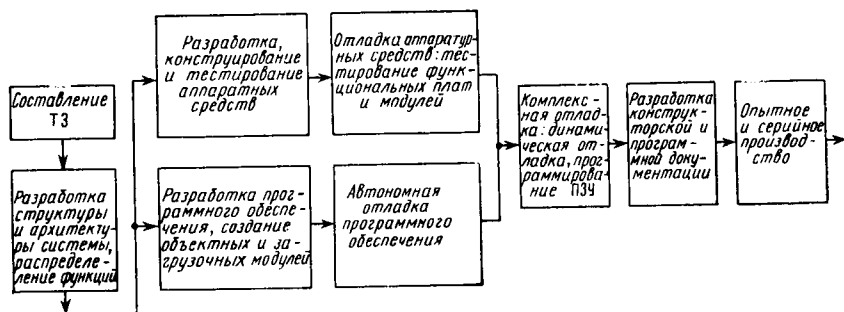


Рис. 4.1. Модель процесса разработки МПУ и МПС

Разработка аппаратных и программных средств МПУ производится параллельно, причем по мере завершения проекта взаимодействие между группами разработчиков становится все более тесным. С помощью кросс-систем, моделирующих программ и систем автоматизированного проектирования можно отладить почти все программы. Однако системные моменты типа прерываний можно проверить только на прототипе.

Окончательный этап проектирования -- интеграция прототипа и программы. Прототип здесь еще может иметь неисправности, поэтому на этом этапе производится комплексная отладка аппаратурных и программных средств.

Завершается процесс проектирования разработкой документации, опытным и серийным производством.

Последовательность разработки МПУ и МПС представлена на рис. 4.1.

Проектирование — итерационный процесс. Неисправности, обнаруженные на этапе приемосдаточных испытаний, могут привести к коррекции всей документации и перепроектированию всей системы.

Методы контроля правильности проекта МПУ. Основными методами контроля правильности проекта МПУ является верификация, моделирование и тестирование.

Под верификацией понимают формальные методы доказательства корректности проекта. Однако они являются довольно трудоемкими для использования в инженерных разработках.

Для контроля корректности проекта МПУ на каждом этапе проектирования необходимо производить моделирование аппаратурных и программных средств. При моделировании аппаратурных средств широкое распространение получили методы имитационного моделирования.

Имитационное моделирование — метод исследования, основанный на том, что изучаемая динамическая система заменяется ее имитатором и с ним проводятся эксперименты с целью получения информации об изучаемой системе. Напомним, что динамическими являются системы, которые изменяются во времени. Очевидно, МПУ, а вернее законы их функционирования, относятся к классу динамических систем. Имитаторы МПУ, как правило, реализуются в виде программных моделей на ЭВМ.

При имитационном моделировании процессов не требуется преобразовывать аналитические выражения в специальную систему уравнений относительно искоемых величин. Для имитационного моделирования характерно воспроизведение на ЭВМ явлений, описываемых математической моделью, с сохранением их логической структуры и последовательности чередования во времени.

Имитационное моделирование МПУ реализуется моделирующим алгоритмом, в соответствии с которым в ЭВМ имитируется функционирование проектируемого МПУ с учетом выбранного уровня детализации для получения нужных характеристик. Эти характеристики выводятся на печать и используются в качестве прямых или косвенных результатов проектирования. Таким образом, в процессе имитационного моделирования проверяется и отлаживается модель проектируемого МПУ.

Для проектирования и отладки программных средств МПУ используются кроссовые, резидентные и комбинированные системы проектирования программ.

Кроссовые системы предназначены для моделирования и отладки системных и прикладных программ, а также для моделирования и оценки алгоритмов в процессе разработки и эксплуатации МПУ. Широкие аппаратные и программные возможности универсальных ЭВМ позволяют создавать кроссовое математическое обеспечение с помощью языков высокого уровня и использовать мощные резидентные средства отладки. Применение кроссовых средств снижает стоимость и время разработки МПУ.

Кроссовые системы проектирования, однако, имеют ряд недостатков, ограничивающих их применение:

интерпретация системы команд микропроцессора в кроссовой системе осуществляется сравнительно просто, но моделирование вычислительной среды, в которой работает создаваемое МПУ, требует дополнительных средств и не всегда возможно;

отсутствуют какие-либо возможности отладки аппаратной части МПУ.

В *резидентных системах* программное обеспечение автоматизированного проектирования реализуется на микроЭВМ такого же типа, что и проектируемое МПУ, и выполнение разрабатываемой и отлаживаемой программ происходит одновременно с работой системы проектирования. Таким образом, МПУ разрабатывается в реальных или близких к реальным условиям. Однако резидентным системам также свойственны недостатки:

программное обеспечение системы проектирования достаточно сложное, поэтому для создания резидентной системы необходимо иметь некоторые средства разработки (чаще всего кроссовые);

для обеспечения универсальности резидентных систем микроЭВМ должна быть оснащена развитым набором периферийного оборудования, что является сдерживающим фактором широкого распространения резидентных систем;

программное обеспечение резидентной системы машин независимо.

Комбинированные (комплексные) системы проектирования представляют собой, как правило, многопроцессорный комплекс, в котором основные функции системы проектирования реализует универсальная ЭВМ (в том числе и микроЭВМ), а остальные аппаратные средства используются для поддержки различных проектных процедур. Комбинированная система может использоваться как кроссовая система и предоставлять возможности резидентной системы по отладке программ на реальной аппаратуре. Разработчик получает средство для нисходящего модульного проектирования аппаратной и программной частей МПУ. Начиная с некоторого ядра программного обеспечения, которое полностью интерпретируется системой, можно постепенно достраивать МПУ, добавляя программные модули и перенося функции выполнения на реальные устройства (МП, ЗУ, внешние устройства). При этом используются все возможности программного обеспечения инструментальной ЭВМ, ее периферийное оборудование и вычислительные ресурсы разрабатываемых МПУ.

Роль тестирования при отладке МПУ. Отладка аппаратных и программных средств связана в первую очередь с поиском неисправностей. Поиск неисправностей в аппаратуре МПУ сильно осложнен целым рядом причин, наиболее важными из которых являются: высокая сложность БИС; малое число контрольных точек схемы, физически доступных для непосредственного контроля и воздействия; неразделимость аппаратуры и программного обеспечения; сложность и неразделимость аппаратуры МПУ на функциональные узлы; необходимость одновременного контроля состояния шин адреса, данных и управления на протяжении значительного временного интервала; особенности архитектурно-структурной организации однокристальных и микропрограммируемых секционных микропроцессоров.

Усложняет поиск неисправностей и двойственная природа МПУ: если отказ не очевиден, то необходимо решить, скрыт ли отказ в ап-

паратных средствах или вызван ошибкой в программе. Основным методом поиска неисправностей в МПУ является тестирование.

Принципы тестирования МПУ по существу не отличаются от тестирования обычных цифровых устройств. Для поиска неисправностей и отладки МПУ необходимо определить состав требуемого контрольно-измерительного оборудования и составить логическую схему поиска неисправности. Диапазон контрольно-измерительной аппаратуры для отладки МПУ довольно широк — от обычных осциллографов и цифровых вольтметров до специальных приборов типа сигнатурных и логических анализаторов. Помимо аппаратурных средств для поиска неисправностей в МПУ используются программные средства. Тест-программы либо встраиваются в состав программного обеспечения МПУ или МПС и выполняются как часть процедуры включения системы, либо вызываются по мере необходимости из ПЗУ.

Остановимся на основных проблемах тестирования МПУ. Прежде всего, поскольку управление проверкой МПУ скрыто в программах, это затрудняет прослеживание прохождения сигналов. Кроме того, все процессы в МПУ происходят очень быстро, что делает невозможным и наблюдение, и анализ функционирования МПУ в реальном времени. Измерения нужно производить, когда МПУ работает с номинальным быстродействием, что делает невозможным использование традиционных систем контроля.

§ 4.2. МЕТОДЫ ТЕСТИРОВАНИЯ МПУ

Задачи тестирования. Особенности организации процесса обработки информации, введение новых технологических операций на стадии производства и оригинальные схмотехнические решения позволяют выделить МПУ в особый класс цифровых устройств, требующих разработки специальных процедур определения их работоспособности. Это, однако, не означает отказ от широко применяемых в настоящее время методов обнаружения и поиска неисправностей цифровых устройств. Целесообразным представляется подход, основанный на оптимальном использовании результатов, полученных за последние годы в области контроля и технической диагностики, с учетом особенностей архитектуры и логики функционирования МПУ.

Под тестированием МПУ будем понимать процесс установления исправности или работоспособности устройства с помощью определенных входных воздействий и анализа соответствующих выходных реакций. Тестирование является одной из основных процедур технической диагностики (контроля), задачами которой являются определение технического состояния объекта контроля и, в случае его неработоспособности, — обнаружение и локализация неисправностей.

Совокупность входного воздействия и соответствующей ему выходной реакции называется тестом, а упорядоченная последовательность тестов — тестовой программой. Процедура контроля МПУ состоит из разработки тестовой программы, последующей подачи вход-

ных воздействий на контролируемый прибор, наблюдения выходных сигналов и анализа полученных результатов с целью установления годности изделия. Процедура контроля обеспечивает полный (неполный) контроль МПУ, если она обнаруживает любую (не обнаруживает хотя бы одну) неисправность рассматриваемого класса нарушений. Полнота контроля является одним из основных требований, предъявляемых к разрабатываемой тестовой программе устройства. Другим, противоречащим первому требованию, является максимизация скорости контроля, что обычно интерпретируется как минимизация длины тестовой программы. В зависимости от того, что является исходной информацией для создания тестовой программы МПУ, различаются два вида контроля: функциональный и структурный.

При *функциональном контроле* в качестве исходной информации для построения тестовой программы используется алгоритм функционирования МПУ. Необходимость функционального контроля вызывается отсутствием полной информации о причинах отказов, повышенной сложностью контролируемого устройства, пониженными требованиями к полноте контроля и т. п. Функциональный контроль чаще всего применяется пользователями МПУ. На стадии производства функциональный контроль используется главным образом для проверки полупроводниковых ЗУ.

Методы построения тестов для *структурного контроля* ориентируются на принципиальную схему (структуру) проверяемого МПУ. Они используются главным образом для контроля простейших МПУ или даже отдельных узлов таких МПУ на стадии производства. Эти методы в настоящее время разработаны наиболее полно и на практике зарекомендовали себя при контроле и диагностике устройств, состоящих из типовых элементов замены (ТЭЗ). Структурные методы, как правило, обеспечивают достаточную полноту контроля. Однако они малоэффективны при отсутствии адекватных моделей некоторых характерных для МПУ нарушений и в случае проверки сложных схем, насчитывающих тысячи логических и запоминающих элементов.

В зависимости от детализации рассматривания МПУ как объекта тестового контроля различают модульный и системный подходы.

Модульный подход предусматривает разбиение МПУ на блоки (модули), для которых по отдельности определяются тестовые программы. Полученные тестовые программы блоков соответствующим образом объединяются (склеиваются), образуя тестовую программу всего МПУ. Одной из распространенных разновидностей модульного подхода является метод «раскрутки», сущность которого заключается в выборе определенной последовательности проверки отдельных блоков МПУ, позволяющей в первую очередь установить исправность узлов МПУ, нарушение функционирования которых может влиять на достоверность результатов контроля последующих узлов.

Модульный подход обладает тем преимуществом, что сложная структура МПУ рассматривается как композиция более простых структур, для которых трудоемкость построения тестов может быть

значительно меньше, чем для всего устройства в целом. Недостатком этого подхода является то, что исправная работа составляющих узлов не гарантирует правильность работы МПУ.

Системный подход предполагает, что МПУ является единым законченным функциональным устройством, для которого строится тестовая программа.

Из сказанного следует, что при модульном и системном подходе могут быть использованы и функциональные, и структурные методы контроля.

Особенности тестового контроля МПУ. Формализация методов построения тестов требует разработки математических моделей технических устройств как объектов диагноза и моделей неисправностей, характерных для этих объектов.

Построение моделей, адекватно описывающих функционирование объектов диагноза в условиях возможных нарушений, требует глубокого исследования структуры, логики работы и анализа причин, вызывающих отклонение алгоритма функционирования от заданного. Сравнительно небольшой срок, прошедший со дня выпуска первого микропроцессора (1971 г.), не дает оснований утверждать, что адекватная модель МПУ получена.

Основные трудности при разработке МПУ, следующие:

МПУ резко различаются по структурной организации, функциональным возможностям, назначению, принципу действия и т. п. Каждому из этих узлов соответствует, как правило, свой класс возможных неисправностей, что затрудняет получение единой модели неисправности, охватывающей все возможные нарушения работы МПУ;

достигнутая современной технологией производства степень интеграции (плотность компоновки элементов на кристалле) вызвала появление нарушений, связанных со взаимным влиянием соседних элементов. Часть причин, вызывающих нарушение логики функционирования, остается невыявленной. Все это затрудняет получение подходящих моделей неисправностей МПУ;

большая часть процесса обработки информации в МПУ недоступна для наблюдения извне, что в значительной степени затрудняет установление причины отказа;

по мнению некоторых специалистов, если неисправности логических схем достаточно хорошо описывались моделью константных логических неисправностей в линиях, т. е. «катастрофическими» отказами, то в МПУ на первый план выступают постепенные отказы в результате изменения электрических параметров;

нарушения в работе МПУ могут быть вызваны как отказами аппаратуры, так и ошибками, допущенными при составлении программы, а также совокупностью этих причин;

работа МПУ требует высокой степени согласованности временных параметров функциональных узлов. Рассогласование временных сигналов может вызвать сбой в работе, необходимость учета которых ведет к усложнению модели МПУ.

Поясним сказанное. Сложность тестирования типичной микросхемы оценим на примере микропроцессора КР580ВМ80А. Система его команд состоит из 78 базовых команд ($m = 78$), длина слова данных $n = 8$. Общее число тест-комбинаций для проверки МП с учетом всех возможных двоичных наборов для каждой команды определяется равенством $C = 2^{mn} = 2^{8 \cdot 78} = 2^{624}$, что в десятичной системе счисления составляет величину $C = 10^{187.84}$. Предположим, что каждый тест длится 1 мкс (значение почти недостижимое для МП КР580). Тогда на проведение всех тестов потребуется $10^{181.84}$ или $3,171 \cdot 10^{174.84}$ лет. Нетрудно оценить реальную достижимость полного контроля такой микросхемы, если примерный возраст Земли оценивается в $4,7 \cdot 10^9$ лет. Из-за невозможности полного контроля микросхемы микропроцессора проводится ее функциональное тестирование, при котором каждая функция проверяется с привлечением ограниченного множества двоичных наборов.

Простейший вид тестирования МПУ в условиях эксплуатации — проверка центрального процессора МПУ в режиме свободного счета. При этом определяется правильность считывания команды с шины данных, формирования адресных наборов на шине адреса и корректности реагирования на сигналы системной синхронизации. Контролируя сигналы в линиях системной шины, можно частично проверить шину управления. Обычно при проверке в условиях эксплуатации центральный процессор МПУ считается работоспособным, если он функционирует в режиме свободного счета. При отказе центрального процессора МПУ для обнаружения и локализации неисправности необходимо использовать другую вычислительную систему. Поэтому большинство сложных контрольно-измерительных приборов для проверки МПУ сами имеют встроенные микропроцессоры.

§ 4.3. СРЕДСТВА ОТЛАДКИ АППАРАТУРЫ МПУ

Для технического обслуживания и отладки аппаратурных средств МПУ может применяться обычная контрольно-измерительная аппаратура (осциллографы, цифровые вольтметры, частотомеры и др.), ручные инструментальные средства (логические пробники, логические компараторы и др.), логические анализаторы, сигнатурные анализаторы и эмуляторы.

1. Обычные контрольно-измерительные приборы

Под обычной контрольно-измерительной аппаратурой будем понимать приборы, которые использовались для контроля электронных схем до появления микропроцессоров — осциллографы, цифровые вольтметры и частотомеры.

На первых этапах отладки аппаратуры МПУ может быть использован *осциллограф*. С помощью осциллографа можно проверить работу источников питания, тактового генератора, а также правильность

функционирования микропроцессора, наблюдая осциллограммы на его выходных шинах при циклическом повторении несложной последовательности из нескольких команд. Проверка периодических сигналов в МПУ возможна с помощью осциллографа, поскольку после согласования частоты периодического сигнала и скорости развертки на экране осциллографа появляется неподвижное изображение вследствие наложения друг на друга последовательных сигналов.

Ограниченность использования осциллографов при отладке МПУ объясняется тем, что в МПУ при выполнении программ нет периодических сигналов. Поэтому при индикации на осциллографе видны почти случайные последовательности двоичных сигналов, которые интерпретировать практически невозможно.

Для контроля редких и однократных сигналов применяются *запоминающие осциллографы*, которые при запуске воспринимают входной сигнал и индицируют его либо до естественного затухания, либо до стирания с целью воспроизведения нового сигнала. В современных запоминающих осциллографах информация о входном сигнале преобразуется в цифровую форму и хранится в ЗУПВ. Для индикации на электронно-лучевой трубке информация о сигнале преобразуется в аналоговую форму. Хранимые в ЗУПВ цифровые значения не подвержены затуханию и изображение на экране может индицироваться практически сколь угодно долго. Осциллографы с цифровым хранением информации называют *цифровыми осциллографами*.

В микропроцессорных РЭС *цифровые вольтметры* применяются для измерения напряжений питания и потребляемых токов. Из-за малого быстродействия преобразователей, входящих в состав цифровых вольтметров, последние не обнаруживают быстрые выбросы напряжения, которые возникают в МПУ при отказах.

Для измерения частоты генератора синхронизации МПУ, а также для измерения временных интервалов используются *частотомеры*. С их помощью можно определять продолжительность управляющего сигнала, посылаемого в выходной порт, следовательно, частотомер может являться средством проверки правильности программы, которая формирует этот сигнал.

Ограниченность применения обычных контрольно-измерительных приборов при отладке микропроцессорных РЭС объясняется невозможностью одновременного контроля ими параллельного представления информации в магистралях МПУ и высокой скоростью изменения информации.

2. Ручные инструментальные средства

Ручные инструментальные средства применяются при проверке МПУ путем реализации проверок по принципу «стимул—реакция». К ним относятся логические пробники, пульсаторы, компараторы, бесконтактные индикаторы тока и т. д.

Логический пробник контролирует поведение одной точки в МПУ и с помощью индикатора сообщает пользователю о том, в каком состоянии находится проверяемая точка («1», «0» или промежуточный уровень). Поэтому главное назначение логического пробника — проверка подачи питания на микросхемы, контроль статических уровней в логических элементах и определение наличия импульсов в проверяемых цепях.

Логические пульсаторы — схемные стимулирующие приборы, предназначенные для введения в узел МПУ коротких и мощных импульсов или последовательностей импульсов, которые переводят узел из одного состояния в другое и возвращают его в первоначальное состояние.

Метод контроля логических схем, при котором применяются логический пробник и логический пульсатор, называется тестированием по принципу «стимул — реакция». С помощью пульсатора в узел вводится стимулирующее воздействие, а получающаяся реакция логической схемы прослеживается логическим пробником. Зная проверяемую схему, исследователь может проследить логическим пробником тракт распространения сигнала.

Бесконтактные *индикаторы тока* предназначены для контроля токов в защищенных изоляционными покрытиями проводниках печатных плат. Индикатор фиксирует наличие изменяющегося тока в печатном проводнике без разрыва проводника и разрушения защитных покрытий с помощью индуктивной приемной катушки.

Некоторые интегральные микросхемы можно проверить в МПУ, пользуясь *логическим компаратором*. Компаратор содержит плату с микросхемой, аналогичной проверяемой, и имеет клипсу, с помощью которой он подключается к проверяемой микросхеме. В процессе работы компаратор воспринимает все входные сигналы проверяемой микросхемы и подает их на встроенную в него микросхему. Если на выходах обеих микросхем обнаружатся разные сигналы, то в логическом компараторе формируется сигнал ошибки. Недостатком компаратора является необходимость в наличии отдельной эталонной платы для каждой проверяемой микросхемы.

Ручные инструментальные средства довольно эффективны при анализе отказов в обычных логических схемах, но они почти бесполезны при анализе МПУ и МПС с шинной структурой, где информация обновляется последовательно во времени на большом числе линий одновременно.

3. Сигнатурные анализаторы

Формирование сигнатуры. Сигнатурный анализ базируется на методике контроля информации с помощью циклических избыточных кодов.

Известно, что любое двоичное слово можно представить в виде двоичного полинома. Например, 8-битное двоичное слово можно представить полиномом степени 7:

$$B_7 2^7 + B_6 2^6 + B_5 2^5 + B_4 2^4 + B_3 2^3 + B_2 2^2 + B_1 2^1 + B_0 2^0,$$

где $B \in \{0,1\}$. Обозначая переменную в полиноме символом X (вместо конкретного числа 2), можно представить n -разрядное двоичное слово в виде $A(X) = \sum_{i=0}^{n-1} B_i X^i$. Например, двоичное слово 11000001 представляется полиномом

$$A(X) = X^7 + X^6 + X^0 = X^7 + X^6 + 1.$$

Циклический избыточный контроль производится следующим образом. Входная кодовая последовательность $A(X)$ снимается путем деления ее на порождающий полином $G(X)$. Если разделить полином $A(X)$ на $G(X)$, то получится частное $Q(X)$ и остаток $R(X)$:

$$A(X) = G(X)Q(X) + R(X) \text{ или } A(X) - R(X) = G(X)Q(X). \quad (4.1)$$

При циклическом избыточном контроле передаваемая двоичная последовательность $A(X)$ делится на порождающий полином $G(X)$ и получающийся остаток $R(X)$ добавляется в передаваемый двоичный поток, т. е. передается информация, определяемая суммой полиномов $A(X) + R(X)$.

В приемнике входной двоичный поток вида $A(X) + R(X)$ делится на полином $G(X)$. При правильной передаче кодовой последовательности $A(X) + R(X)$ остаток в результате деления должен быть равен нулю. Отметим, что в выражении (4.1) остаток вычитается из двоичного потока $[A(X) - R(X)]$, но в арифметике по модулю 2 операции сложения и вычитания дают одинаковые результаты, поэтому передача $A(X) + R(X)$ производит то же самое действие.

Для реализации циклического избыточного контроля применяется 16-разрядный регистр сдвига, на вход которого подается контролируемый двоичный набор $A(X)$, а из соответствующих разрядов регистра сдвига, определяемых порождающим полиномом $G(X)$, берутся сигналы обратной связи. Входной сигнал и все сигналы обратной связи суммируются по модулю 2 (операция исключающее ИЛИ), в результате чего получается линейная последовательностная схема, так как сумматор по модулю 2 придает один и тот же вес каждому входному биту. В результате работы сдвигающего регистра с обратными связями входной поток $A(X)$ делится на выбранный порождающий (или характеристический) полином $G(X)$. По окончании двоичного потока остаток представляется состояниями 16 триггеров, остаток извлекается из регистра и добавляется к передаваемому потоку данных. Содержимое регистра сдвига, преобразованное в 4-разрядное число в шестнадцатеричной системе, называется сигнатурой. Сигнатурный метод позволяет снимать длинные последовательности логических уровней в отдельных точках цифровой аппаратуры в сигнатуры, которые могут приводиться в схемах.

Сигнатурные анализаторы. Основным элементом сигнатурного анализатора является сдвигающий регистр с обратными связями, пре-

образующий двоичную последовательность выбранной длины от конкретного узла проверяемой схемы в сигнатуру. Обычно в сигнатурных анализаторах используется 16-битный сдвигающий регистр с обратными связями, преобразующий входную двоичную последовательность в 4-разрядную шестнадцатеричную $(16)^4 = 2^{16}$ «метку» — сигнатуру проверяемого узла. Обработываемые двоичные последовательности суммируются по модулю 2 с сигналами четырех разрядов обратной связи в соответствии с выбранным полиномом $G(X)$. Выбор вида полинома $G(X)$ производится из условия недопустимости группирования ошибок. По этой причине не рекомендуется делать выводы

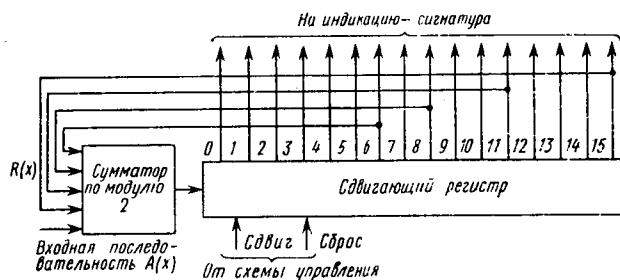


Рис. 4.2. Сдвигающий регистр с обратными связями

через 4 или 8 разрядов, так как они соответствуют наиболее вероятным размерам слов в микропроцессорах, и применять четное число входов. Фирма Hewlett—Packard остановилась на нечетном числе входов, применив выражение обратной связи вида $X^{16} + X^{12} + X^9 + X^7 + 1$ — разряды 6, 8, 11, 15 сдвигового регистра (рис. 4.2). Регистр синхронизируется теми же тактовыми сигналами, что и обрабатываемая двоичная последовательность. Входные последовательности могут быть любой фиксированной длины, но в конце обработки они превращаются в 16-битное слово. Эти 16-битные слова, воспроизведенные в шестнадцатеричном формате (в алфавите 0123456789 ACFHPU), и являются искомыми сигнатурами.

В сигнатурном анализаторе производятся следующие преобразования входной последовательности $A(X)$. Код входной последовательности делится на порождающий полином $G(X)$, определяемый обратными связями. Деление происходит с помощью сдвигающего регистра, а остаток, зафиксированный в этом регистре после окончания деления всей последовательности на порождающий полином, и будет сигнатурой этой последовательности. Деление в сдвигающем регистре осуществляется за счет обратных связей и сложения по модулю 2. В арифметике по модулю 2 нет межразрядных переносов, операции сложения и вычитания эквивалентны.

Достоверность сигнатурного анализа. Последовательность ошибок представляет собой результат поразрядного сложения по модулю 2 эталонной (исходной) и искаженной последовательностей. Последовательность ошибок будет содержать 1 в тех и только тех разрядах, в которых искаженная последовательность отличается от исходной, во всех остальных разрядах последовательности ошибок будут содержаться 0.

Сдвигающий регистр с сумматором по модулю 2, представленный на рис. 4.2, является линейной системой, следовательно, для него справедлив принцип суперпозиции: реакция регистра на сумму двух входных воздействий равна сумме реакций на каждое из этих воздей-

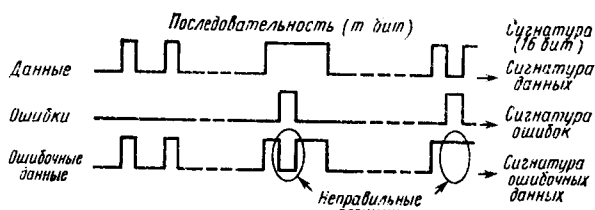


Рис. 4.3. Формирование сигнатуры ошибочных данных

ствий. Отсюда следует, что если входной сигнал регистра рассматривать как сумму по модулю 2 двух двоичных последовательностей (исходной последовательности и последовательности ошибок), то полученная сигнатура будет равна поразрядной сумме по модулю 2 сигнатур каждой из этих последовательностей (рис. 4.3). Из этого следует, что для того, чтобы ошибка была опознана сигнатурным анализатором, сигнатура ошибочных данных не должна быть такой же, как сигнатура правильных данных, или, другими словами, сигнатура ошибки не должна быть равна нулю.

При использовании 16-разрядного регистра сигнатура последовательности ошибок не может быть нулевой для последовательности длиной $m \leq 16$, содержащей хотя бы одну 1, потому, что первая 1, попавшая в регистр, не успеет выйти из него до окончания формирования сигнатуры и не может быть уничтожена из-за сложения с битом обратной связи.

При длине последовательности $m = 17$ одна из возможных последовательностей может быть упущена, а именно последовательность, начинающаяся с 1 и содержащая 1 во всех разрядах, соответствующих отводам в обратной связи. При нумерации бит с возрастанием слева направо такой последовательностью будет 10000001010010001, сигнатура этой последовательности равна нулю.

Среди последовательностей длиной $m = 18$ «невидимых» последовательностей будет $3^{2m-n} - 1$, т. е. последовательности, начинающиеся с комбинаций 10, 01 или 11 и имеющие 1 только в разрядах,

соответствующих отводам в обратной связи. При $m = 19$ «невидимых» последовательностей будет 7, $(2^{(19-16)} - 1)$, при $m = 20$ их будет $2^{(20-16)} - 1 = 15$ и т. д.

Определим вероятность ошибки сигнатурного анализа. Для этого рассмотрим таблицы соответствия между кодами входной последовательности (в окне данных длиной m) и сигнатурами размера $n < m$

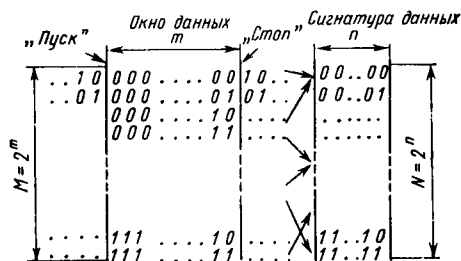


Рис. 4.4. Таблица соответствия между данными и сигнатурами

анализатора и размера m окна данных. Коды данных, находящиеся в одной и той же строке таблицы (рис. 4.5), образуют класс эквивалентности, соответствующий одной и той же сигнатуре. Отсюда видно, что коды любого одного класса могут отличаться друг от друга только кодом нераспознаваемой ошибки, т. е. кодом, находящимся в первой строке таблицы (рис. 4.5). Из этой же таблицы видно, что чем больше неоднозначность $P = 2^{m-n}$ кодирования, тем больше «сжатие» данных.

Если любой ошибочный код в окне данных длиной m может встретиться с равной вероятностью, то вероятность ошибки анализа $\delta_{\text{ош}}$ равна отношению числа кодов в классе эквивалентности (без одного правильного) к общему числу кодов (без одного правильного):

$$\delta_{\text{ош}} = (2^{m-n} - 1) / (2^m - 1) \approx 1/2^n. \quad (4.2)$$

Вероятность правильного результата анализа

$$P_{\text{пр}} = 1 - 2^{-n}.$$

При $m \leq n$ $P_{\text{пр}} = 1$. При $n = 16$ и $m > n$ вероятность $P_{\text{пр}} = 0,99998$ независимо от длины m последовательности.

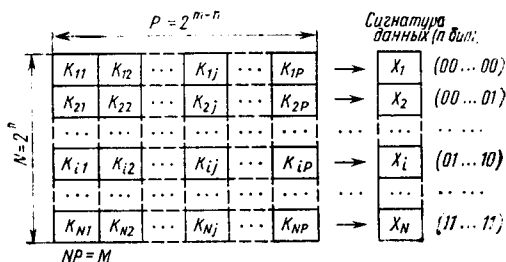


Рис. 4.5. Таблица соответствия между сигнатурами и данными с выделенными классами эквивалентности

Требования к проведению сигнатурного анализа. Сигнатурный анализ — простой и достоверный метод сравнения двух битовых последовательностей произвольной длины, анализом которых и занимаются при отыскании неисправностей в микропроцессорных устройствах и системах с помощью традиционных измерительных приборов. Однако его применение требует включения в МПУ еще на стадии разработки специальных средств, прежде всего средств, позволяющих в режиме контроля разрывать цепи обратной связи в контролируемой схеме, поскольку при выходе из строя одного из элементов, входящих в контур обратной связи, локализовать неисправность методом сигнатурного анализа не удастся.

Кроме того, на стадии разработки должны быть предусмотрены:

аппаратные средства формирования импульсов «Такт», «Пуск» и «Стоп»;

тестовые (стимулирующие) программы, вырабатывающие контрольные сигналы для всех узлов системы и управляющие формированием импульсов «Пуск», «Стоп»;

ПЗУ для хранения тестовой программы;

принципиальная схема устройства (системы) или другая документация с указанными на ней сигнатурами, снятыми в контрольных точках при работе стимулирующей программы на исправно работающем устройстве;

конструктивные средства, позволяющие легко перевести устройство в тестовый режим работы;

рекомендации разработчиков устройства по последовательности проведения сигнатурного анализа при отыскании неисправностей.

Упрощенная структурная схема МПУ, построенного на однокристальном МП и разработанного с учетом требований применения сигнатурного анализа при испытаниях, приведена на рис. 4.6.

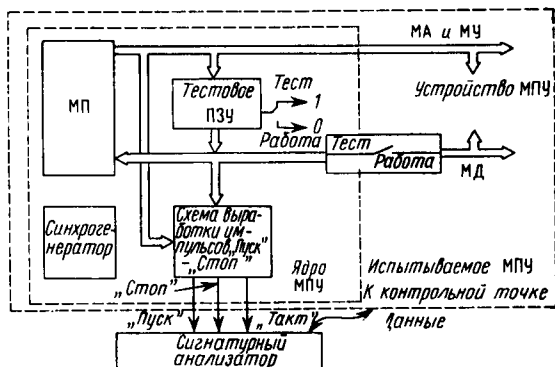


Рис. 4.6. Структурная схема МПУ, приспособленного для сигнатурного анализа

4. Логические анализаторы

Особенности логических анализаторов. Контрольно-измерительные приборы, предназначенные для сбора данных о поведении дискретных систем (в том числе МПС), обработки этих данных и представления их человеку на различных уровнях абстракции, получили

название *логических анализаторов* (ЛА). Для ЛА характерны следующие особенности: возможность наблюдения и обработки сигналов одновременно по нескольким каналам; параллельная регистрация и запоминание данных по всем каналам; нормирование входных сигналов по логическим уровням; возможность реализации различных вариантов запуска развертки; возможность регистрации предельно коротких помех по сравнению с периодом повторения сигнала импульсных помех; возможность машинной обработки, декодирования и сравнения полученных данных с опорными.

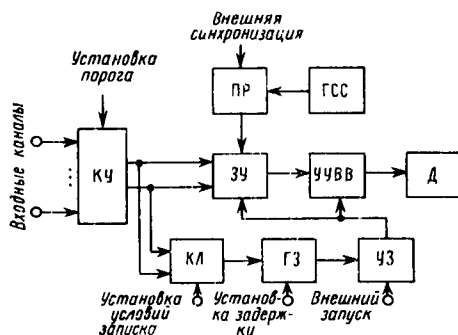


Рис. 4.7. Обобщенная структура логического анализатора (ГСС — генератор синхронизирующих сигналов; ПР — переключатель режима)

Обобщенная структура ЛА представлена на рис. 4.7. На входные каналы ЛА поступают сигналы с отлаживаемой или диагностируемой МПС, которые распределяются компараторами уровня КУ на соответствующие логические уровни. Сформированный в КУ набор значений сигналов подается на входы запоминающего устройства ЗУ и логического компаратора КЛ.

ЛА во время работы может находиться в одном из трех режимов настройки, регистрации и индикации.

В режиме настройки КЛ настраивается (программируется) на обнаружение определенной последовательности наборов значений сигналов. После поступления запрограммированной последовательности входных наборов КЛ выдает сигнал на генератор задержки ГЗ, который через заданное время выдает сигнал на устройство запуска УЗ, инициирующее или прекращающее запись наборов значений входных сигналов в ЗУ. Устройство управления визуальным выводом ЧУВВ транслирует информацию на экран дисплея Д в виде временных диаграмм, таблиц, графов и т. д.

ЛА при определении значений сигналов отображают на экране дисплея нормированные по уровню цифровые сигналы: если значения входного сигнала превышают порог срабатывания входного компаратора КУ, то значения сигнала на входе ЗУ соответствуют «1», если не превышают, то «0».

Из всего многообразия логических анализаторов можно условно выделить две группы приборов: анализаторы логических временных диаграмм и анализаторы логических состояний. Они отличаются способами визуального вывода, числом входных каналов и степенью эффективности при проверке цифровых схем с обычной логикой и схем на основе микропроцессора.

Анализаторы логических временных диаграмм. Это такие ЛА, в которых реализован асинхронный режим записи. Источником синхронизирующих сигналов является внутренний генератор (ГСС), при этом входные данные ВД стробируются синхронизирующим сигналом СС внутреннего генератора (рис. 4.8). В результате стробирования в память записывается последовательность данных ПД.

Анализаторы подобного типа обычно используются для нетактируемых информационных сигналов, регистрации переходных процессов, т. е. там, где тактовая частота регистрации может не синхронизироваться с моментами изменения входного сигнала.

В анализаторах логических временных диаграмм стробирование

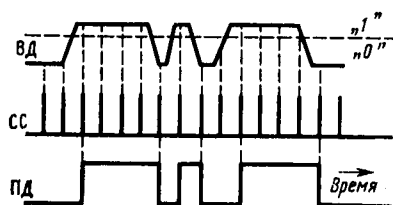


Рис. 4.8. Временные диаграммы записи данных в асинхронном режиме

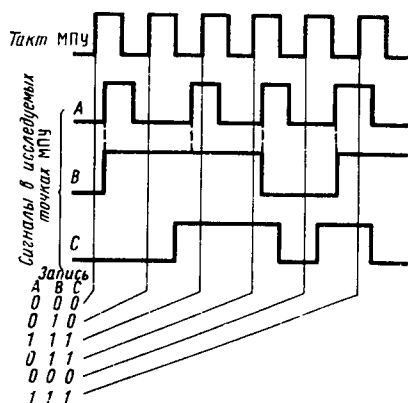


Рис. 4.9. Временные диаграммы записи данных в синхронном режиме

данных при передаче в память осуществляется с более высокой частотой, чем тактовая частота проверяемого устройства. Благодаря этому указанные анализаторы дают довольно точную информацию о сигналах на входных шинах и особенно полезны для тестирования асинхронных схем. Такие анализаторы позволяют обнаруживать кратковременные импульсные помехи, неодновременность прихода сигналов и перемежающиеся нарушения синхронизации.

Анализаторы логических состояний. Это разновидность ЛА, имеющих синхронный режим записи. При синхронном режиме записи анализатор воспринимает только установившиеся значения логических сигналов в контрольных точках, не давая никакой информации о временных соотношениях между сигналами тактового интервала исследуемого устройства (рис. 4.9). Поэтому результаты синхронного анализа, как правило, удобнее анализировать в табличной форме, в то время как результаты асинхронного анализа — в виде временных диаграмм.

Для анализаторов логических состояний характерно двоичное представление данных и отсутствие представления в виде временных диаграмм, так как эти приборы рассчитаны на работу с тактиро-

ванными логическими схемами. Двоичное представление удобно для восприятия и сравнения с текстовыми документами.

Следует отметить, что при контроле одного и того же МПУ методом асинхронного анализа требуется быстроедействие ЛА в 5 ... 10 раз выше, чем при синхронном анализе. Современные логические анализаторы МПУ сочетают в одном приборе возможности временного и логического анализа, имеют большое число входных каналов, позволяют представлять данные в виде временных диаграмм и в двоичном коде, используют матричное, шестнадцатеричное, восьмеричное и мнемоническое представления. Для управления работой МПУ и возможностью контроля результатов анализа в ЛА обеспечивается диалоговый режим работы.

Для обслуживания логических анализаторов всех видов требуются операторы высокой квалификации, умеющие интерпретировать достаточно большие массивы выходных данных, имеющие хорошее представление о работе проверяемого устройства и владеющие методами отыскания неисправностей.

5. Внутрисхемные эмуляторы

Эмуляция — процесс, в котором одна система используется для копирования действий другой системы. Для организации эмуляции различных компонентов разрабатываемого МПУ используются *внутрисхемные эмуляторы*. Они предназначены для организации комплексной отладки аппаратуры и программного обеспечения МПУ в процессе разработки. Промышленностью выпускаются эмуляторы в виде автономных приборов, которые входят в состав тестеров, предназначенных для функционального контроля МПУ в процессе производства и эксплуатации изделия. В функции внутрисхемного эмулятора входят эмуляции поведения микропроцессора и ЗУ проектируемого МПУ, периферийных устройств и различных контроллеров МПУ, управление поведением проектируемых микропроцессорных РЭС. При отладке МПУ на однокристальных МП с фиксированной системой команд необходима организация генерации программных воздействий на различные компоненты МПУ. Для отладки МПУ на микропрограммируемых МП помимо внутрисхемного эмулятора требуется имитатор ПЗУ с записанным в нем набором микропрограмм.

Внутрисхемный эмулятор может работать в режимах опроса состояния различных узлов МПУ, пошагового исполнения программы (микропрограммы) пользователя и эмуляции исполнения программы (микропрограммы) пользователя в реальном времени. С помощью эмулятора проверяются ядро МПУ, магистрали, а также выполняются тесты ПЗУ и ЗУПВ.

Внутрисхемные эмуляторы весьма эффективны при обработке и проверке микропроцессорных схем. При этом методе применяется внешняя по отношению к проверяемому прибору система для имитации МП и обеспечивается диагностика отказов при функционировании в

реальном времени. Использование сменных узлов позволяет легко адаптировать прибор к конкретному типу МП. Для такой аппаратуры требуется персонал высокой квалификации, но она не позволяет диагностировать источники неисправностей с точностью до компонента. При объединении методов и аппаратных средств внутрисхемной эмуляции и сигнатурного анализа появляется возможность создать чрезвычайно мощное средство для проверки цифровых микропроцессорных схем. С помощью внутрисхемного эмулятора анализатор вырабатывает цифровые коды, благодаря чему оказывается возможным проверять фактически неработающую схему. Единственное, что необходимо для выполнения проверки, — это функционирование генератора тактовых сигналов проверяемой системы. Микропроцессор анализатора подключается с помощью кабеля эмуляции вместо микропроцессора проверяемой системы. В ППЗУ анализатора хранятся программы испытаний. Анализатор должен иметь разные режимы запуска для выполнения программ с целью проверки содержимого регистров ячеек памяти, а также правильности функционирования устройств ввода — вывода. Встроенные в анализатор средства сигнатурного анализа позволяют отыскивать неисправные компоненты.

Оценивая рассмотренную аппаратуру и реализованные в ней методы диагностики, можно сделать заключение, что наиболее простыми в применении приборами, позволяющими производить высококачественную и быструю проверку сложных цифровых устройств в условиях производства и эксплуатации, являются приборы, использующие метод сигнатурного анализа. К тому же это пока единственный метод, позволяющий локализовать поиск неисправных компонентов в микропроцессорных схемах. Этот метод применим для цифровых, аналоговых и аналого-цифровых схем.

§ 4.4. САМОТЕСТИРОВАНИЕ

Концепция самотестирования. По мере усложнения структур и функциональных возможностей МПУ и МПС первостепенное значение приобретает необходимость встроенного самоконтроля и самотестирования. Как отмечалось, тестирование и отладка МПУ требуют сложной контрольно-измерительной и отладочной аппаратуры с использованием микроЭВМ, а также обширный набор программных средств для реализации процесса отладки.

Эффективным решением этой проблемы представляется концепция самотестирования, которая заключается в том, что МПУ или микропроцессорные РЭС разрабатываются таким образом, чтобы обеспечивалась возможность самотестирования без участия какого-либо внешнего оборудования или программных средств. В идеальном случае при включении питания в МПУ должно осуществляться самотестирование и при обнаружении отказа в МПУ должен определяться отказавший узел.

Средства самотестирования должны быть спроектированы в процессе разработки МПУ с целью удовлетворения потребностей заказчиков в удобной профилактике и сокращении времени отладки. Рассмотрим некоторые аппаратные методы организации самотестирования, применяемые в микропроцессорных РЭС.

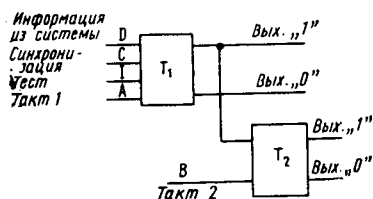


Рис. 4.10. Схема элементарного сдвигового элемента памяти (ЭСЭП)

Подобная организация ЭСЭП позволяет объединить их в сдвигающий регистр большой длины, для чего выход триггера T_2 первого каскада соединен со входом тестовой информации I триггера T_1 следующего каскада и т. д. до последнего ЭСЭП. Наличие сдвигающего регистра позволяет устанавливать и считывать состояние любого триггера путем простого сдвига информации. Таким способом обеспечивается подача установочных последовательностей, фиксирующих исходное состояние схемы. После этого на входы схемы подается тестовая информация, результатом действия которой является изменение состояний внутренних логических блоков. Выходные сигналы заносятся в соответствующие ЭСЭП и оттуда выводятся на выход последовательной информации, где сравниваются с ожидаемыми их состояниями.

Метод встроенного логического блока наблюдения. Одним из удачных методов самодиагностирования является метод BILBO (Built in Logic Block Observation — встроенный логический блок наблюдения). Этот метод предусматривает добавление к тестируемой логической схеме дополнительного оборудования.

В общем случае BILBO является многофункциональным регистром, 8-разрядная схема которого показана на рис. 4.11.

В зависимости от состояния управляющих входов B_1 и B_2 возможны следующие режимы работы:

$B_1 = B_2 = 0$ — линейный сдвиговый регистр, у которого S_{IN} и S_{out} — вход и выход сканирования соответственно;

$B_1 = B_2 = 1$ — системный регистр, в котором значения Z_i загружаются в триггер T_i , а системные операции проводятся с выходами Q_i ;

$B_1 = 0, B_2 = 1$ — параллельный регистр с обратными связями с суммированием по модулю 2 (Q_3, Q_4, Q_7 и Q_8), который может функционировать как сигнатурный анализатор либо как генератор псевдослучайной последовательности (кодов). Генератор псевдослучайных

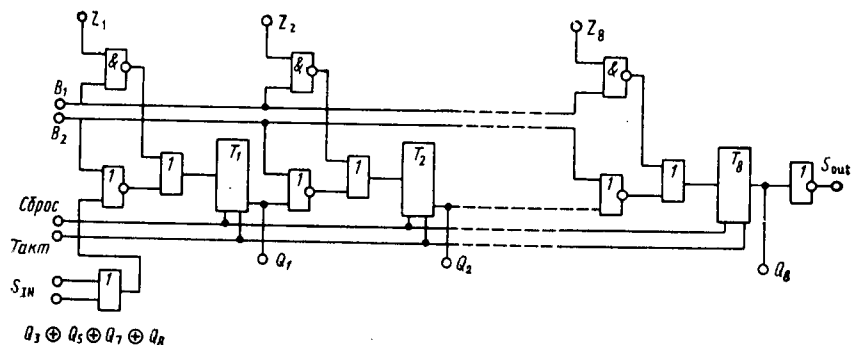


Рис. 4.11. Схема регистра BILBO

кодов выдает параллельный код на схемы испытуемых устройств. Выходные сигналы последних поступают в сигнатурный регистр, выход которого в свою очередь подключен к дешифратору, позволяющему определить, прошел или не прошел данный тест.

Одним из вариантов реализации метода BILBO является модуль BIDCO (Built in Digital Circuit Observer), схема которого включает регистр BILBO = A, работающий в режиме генератора псевдослучайных последовательностей, регистр BILBO = B, работающий в режиме сигнатурного анализатора, и схемы оценки результата проверки по критерию «годен — не годен» (рис. 4.12). Перед началом контроля осуществляется установка в начальное состояние схемы BIDCO и испытуемой схемы, а также запись в регистр BILBO = A начального слова смещения. По истечении цикла синхронизации результат дешифрации выдается на индикаторы состояний. Метод BILBO особенно эффективен при диагностике комбинационных схем. Достоинством метода является значительное сокращение объема тестовых данных (а следовательно, и времени самоконтроля) по сравнению с методом LSSD.

Метод самотестирования MICROBIT. Метод MICROBIT (микропроцессорные встроенные средства тестирования) реализуется следующим образом. В стандартную конфигурацию МПС, совместимую с ши-

ной Multibus, включены два регистра BILBO, реализующие функции генератора псевдослучайных последовательностей и сигнатурного анализатора, а также делитель частоты и кнопка управления началом проверки. Кроме того, в программируемом ПЗУ процессора должна храниться специальная тест-программа, обеспечивающая последовательное тестирование всех функциональных узлов МПУ.

Метод MICROBIT эффективен при тестировании однокристальных микроЭВМ и МПУ, оценка состояния которых производится по критерию «годен --- не годен».

Организация полного самотестирования МПС. Сочетание таких встроенных диагностических средств, как сквозной сдвиговый регистр для последовательного сканирования состояний, генератор псевдослучайных тест-кодов и регистр сигнатурного анализа, позволяет реализовать самотестирование и поиск внутренних ошибок на уровне кристаллов, плат, микросхем, МПУ и МПС. Поскольку стоимость средств самотестирования остается относительно постоянной, а затраты на тестирование увеличиваются в геометрической прогрессии, можно полагать, что уже в ближайшее время с ростом насыщенности СБИС средства самотестирования и самодиагностики станут для них обязательными.

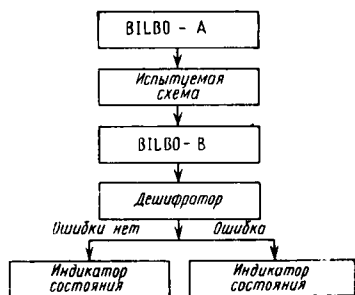


Рис. 4.12. Схема модуля BIDCO

Прогнозы относительно того, что СБИС могут самостоятельно решать проблемы собственного тестирования и диагностики, начинают сбываться. Эти схемы можно сделать самотестируемыми, сочетая несколько видоизмененный метод сквозного сдвигового регистра с последовательным сканированием состояний (LSSD) со средствами генерации псевдослучайных тест-кодов и сигнатурного анализа. Кроме того, такие методы и средства самотестирования можно иерархически распределить между различными уровнями — от кристаллов до схемных плат и в конечном счете до систем в целом, что предусматривает метод LSSD в своем первоначальном виде.

СБИС с самотестированием можно создавать при минимальных усилиях разработчика дополнительных логических схем, если по периферии кристалла добавить кольцо элементарных сдвиговых элементов памяти (ЭСЭП). Последовательный прием или выдача данных из такого кольца сканирования обеспечивает в дальнейшем возможность воздействия на состояния входов или выходов кристалла (или и тех и других одновременно) и наблюдения за ними.

Чтобы ЭСЭП можно было подключить к входному или выходному контакту кристалла, этот элемент выполняют в виде пары триггеров, содержащей главный (управляющий) триггер T_1 и подчиненный (управляемый) триггер T_2 , причем включены они по схеме LSSD

(рис. 4.13). Когда синхронизирующий сигнал SKIN или SKOUT поступает на вход триггера T_1 , последний устанавливается в состояние, соответствующее входу данных D, а при поступлении синхронизирующего сигнала A — в состояние, соответствующее входу I. Аналогично синхронизирующий сигнал B устанавливает триггер T_2 в состояние, соответствующее состоянию триггера T_1 .

Триггеры T_1 и T_2 не влияют на работу кристалла (являются «прозрачными»), когда на их входах синхронизации высокие уровни. Подключая выход каждого триггера T_2 к входу I следующего триггера T_1 , можно объединить все ЭСЭП в сквозной сдвиговый регистр. Переменная выдача синхронизирующих сигналов A и B будет обеспечивать сдвиг данных через этот регистр.

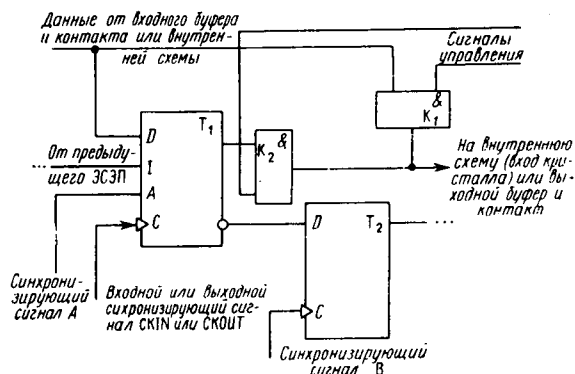


Рис. 4.13. Схема подключения ЭСЭП к кристаллу

В вентильной матрице эти ЭСЭП и соответствующие логические вентили K_1 и K_2 могут образовать единую часть буферов ввода — вывода на периферии кристалла. Вентили передачи K_1 и K_2 — это простые ключи, которые в открытом состоянии разрешают прохождение данных или блокируют их в закрытом состоянии.

Одним из последних достижений в развивающейся области самотестирования является сочетание методов LSSD и BILBO. Здесь псевдослучайные тест-коды генерируются благодаря объединению некоторых элементарных сдвиговых регистров LSSD в сдвиговый регистр максимальной длины. Затем выходные коды расположенных на кристалле логических схем анализируются с помощью встроенного сигнатурного регистра.

При использовании сдвигового регистра для последовательного сканирования состояний, генератора псевдослучайных тест-кодов и регистра сигнатурного анализа возможно строить СБИС с высокой степенью тестируемости. При этом возможно построение генератора псевдослучайных кодов на входных ЭСЭП, а сигнатурного анализатора — на выходных.

В режиме нормальной работы (или LSSD) внутренние триггеры можно сканировать независимо от входных и выходных триггеров, причем кристалл выглядит как типичный кристалл, построенный по методу LSSD.

В режиме тестирования по выводам кристалла периферийное кольцо сканирования можно опрашивать независимо от внутренних ЭСЭП,

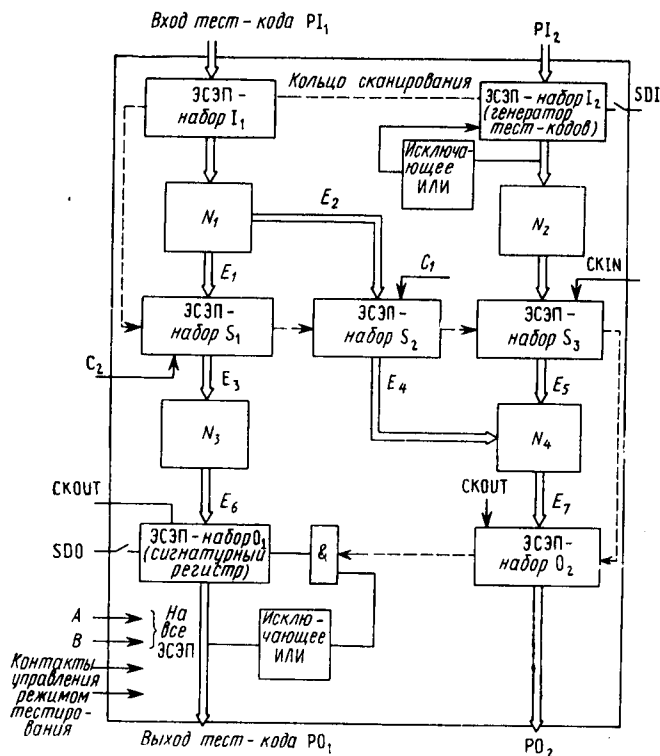


Рис. 4.14. Конфигурация кристалла, построенного по методу LSSD и находящегося в режиме самотестирования

а сигнатурный регистр блокируется. Выходные ЭСЭП передают свое содержимое на внутренние схемы кристалла. Режим тестирования по выводам кристалла используется для проверки связей между кристаллами с помощью внешних тест-векторов.

Режим сканирования по кольцу — вариант режима тестирования по выводам кристаллов, позволяющим извлекать содержимое кольца сканирования ввода — вывода, в то время как все кристаллы системы продолжают работать. Наибольший интерес представляет действительный режим самотестирования кристалла, в котором все входные, внутренние и выходные ЭСЭП становятся одним непрерывным сдвиговым

регистром, начинает работать генератор псевдослучайных тест-кодов и включается сигнатурный регистр. Для работы кристалла в режиме самотестирования требуются только внешние синхронизирующие сигналы, поскольку внутри кристалла генерируются псевдослучайные коды, а для самодиагностики используется сигнатурный анализ.

На рис. 4.14 показана конфигурация кристалла, построенного по методу LSSD и находящегося в режиме самотестирования.

Наборы элементарных сдвиговых элементов памяти S_1 , S_2 и S_3 — это ЭСЭП, которые используются при нормальной работе кристалла и стробируются неперекрывающимися системными синхронизирующими сигналами C_1 , C_2 . Блоки $N_1, \dots, N_4$ — комбинационные схемы, а шины $E_1, \dots, E_7$ — внутренние сигнальные шины между схемами. Порты PI_1 , PI_2 — основные входы кристалла, а PO_1 , PO_2 — основные выходы, каждый из которых содержит множество сигнальных линий. Наборы I_1 , I_2 представляют собой единый набор входных ЭСЭП, а O_1 , O_2 — единый набор выходных ЭСЭП.

Все ЭСЭП образуют многоразрядное кольцо сканирования — от SD1 до SD0: сначала идут входные ЭСЭП, затем внутренние ЭСЭП и, наконец, выходные ЭСЭП.

Рассматривая методы LSSD, BILBO и MICROBIT в сравнении, можно отметить следующее:

наиболее простым в реализации является метод BILBO, требующий введения в структуру МПУ двух сдвиговых регистров, однако без дополнительных программных средств он малоэффективен;

метод LSSD требует введения избыточности в структуру БИС на этапе проектирования, что в свою очередь снижает их надежность и увеличивает стоимость;

при разработке метода MICROBIT были учтены недостатки метода BILBO, разработаны программные средства, обеспечивающие режим самотестирования методом «раскрутки». Благодаря этому метод MICROBIT может быть взят за основу при построении системы программной диагностики.

§ 4.5. МЕТОДЫ ОРГАНИЗАЦИИ РЕЖИМА САМОТЕСТИРОВАНИЯ

Рассмотрим методы организации режима самотестирования на примере самотестирования кристалла, построенного по методу LSSD (см. рис. 4.14).

Для инициирования самотестирования во входные ЭСЭП генератора псевдослучайных тест-кодов (набор I_2) и сигнатурный регистр (последние 8 выходных ЭСЭП) загружаются начальные значения, определяющие конкретную исходную точку. Это делается путем перевода кристалла в режим тестирования по выводам и последовательной загрузки нужных значений. Все внутренние ЭСЭП устанавливаются в «нуль» путем перевода кристалла в режим нормальной работы и последовательного сбрасывания в «нуль», поскольку в этом случае генератор тест-входов будет заблокирован в «нулевом» состоянии. После

такой инициализации всех ЭСЭП кристалл переводится в режим самотестирования. В этот момент первый тест-код поступает на внутренние схемы кристалла. Например: ЭСЭП — набор I_1 воздействует на блок N_1 , набор S_1 — на блок N_3 , набор S_2 — на блок N_4 и т.д. Поступление системных синхронизирующих сигналов C_1 , C_2 позволяет наборам S_1 , ..., S_3 принимать коды реакций блоков N_1 , N_2 . Затем по синхронизирующему сигналу SKOUT производится прием всех значений сигналов на основные выходные ЭСЭП всего кристалла. Поскольку сигнатурный регистр включен, при поступлении синхронизирующего сигнала SKOUT параллельно формируется сигнатура. Если текущим тестовектором внутри кристалла обнаруживаются какие-либо ошибки, то в некоторые из внутренних или выходных ЭСЭП будет принят неправильный ответный код. Следующий шаг процедуры заключается в том, чтобы выдать достаточное количество синхронизирующих сигналов А и В для полного сканирования всех внутренних и выходных ЭСЭП. Если в этих ЭСЭП были записаны какие-либо ошибочные ответные коды, то в конце концов они попадут в сигнатурный регистр, так что сформируется неправильная сигнатура. Когда в сигнатурный регистр поступают последовательные данные, сигнатура формируется последовательно. Одновременно с этим производится сдвиг содержимого входных ЭСЭП и генератор тест-кодов выдает последовательные псевдослучайные коды, которые поступают на входные ЭСЭП и перемещаются во внутренние и выходные ЭСЭП. После завершения цикла сканирования в наборах I_1 , I_2 , S_1 , ..., S_3 оказываются новые псевдослучайные коды, которые воздействуют из комбинационной схемы. Теперь весь цикл тестирования может быть повторен с выдачей системных синхронизирующих сигналов SKOUT и т.д.

Для достаточно полной проверки схемы кристалла следует выполнить необходимое число циклов тестирования, которое можно определить путем моделирования кристалла. После прохождения последнего цикла тестирования можно проверить значение сигнатуры, сдвигая содержимое сигнатурного регистра и сравнивая его с ожидаемым значением сигнатуры, также определяемым с помощью моделирования. Чтобы обеспечить полностью автономное самотестирование, такое ожидаемое значение сигнатуры можно записать в память кристалла, а затем сравнить его с фактической сигнатурой и выдать признаки правильного и неправильного прохождения теста.

Параллельное тестирование. В конечном итоге, поскольку последовательности задаваемых синхронизирующих сигналов сходны между собой для любых кристаллов (они отличаются только количеством системных синхронизирующих сигналов и синхронизирующих сигналов А, В), тестирование многих кристаллов можно осуществлять параллельно; при этом число синхронизирующих сигналов в цикле должно соответствовать максимальному значению, требуемому для какого-либо из кристаллов. Если содержимое кольца сканирования кристалла будет сдвигаться большее число раз, чем необходимо, это не нарушит правильности самотестирования, а приведет просто к потере части

сформированных псевдослучайных тест-векторов. Псевдослучайные тест-векторы, генерируемые внутри кристалла (см. рис. 4.14), позволяют эффективно тестировать этот кристалл, поскольку вся последовательная логика сведена к комбинационной благодаря архитектуре LSSD. В свою очередь, комбинационная логика разделена на непрерывающиеся блоки, такие, как $N_1, \dots, N_4$, которыми можно управлять с помощью соседних ЭСЭП-наборов.

Динамическое тестирование. Если системные синхронизирующие сигналы и синхронизирующие сигналы сканирования в цикле самотестирования имеют те же параметры (длительности, периоды и фазовые соотношения), которые они имели бы при работе с максимальной частотой в обычном режиме, то самотестирование будет представлять не только статическую проверку фиксированных ошибок на кристалле, плате или в системе, но и динамическую проверку кристаллов.

Предположим, например, что для схемы рис. 4.14 время между положительным фронтом последнего синхронизирующего сигнала А в сканирующей части цикла и отрицательным фронтом синхронизирующего сигнала C_1 равно наихудшей задержке на блоке N_1 . Этот последний синхронизирующий сигнал А вызывает загрузку нового псевдослучайного вектора в набор I_2 и стимулирование блока N_2 . Если блок N_2 имеет ошибку по задержке или из-за каких-то изменений в технологическом процессе получился кристалл с меньшим быстродействием, то в некоторый момент времени один или более ЭСЭП в наборе S_3 во время выполнения теста будет установлен неправильно, что в итоге вызовет появление неправильной сигнатуры. Те же соображения распространяются на любой из блоков внутри кристалла, а также на ЭСЭП-наборы. Таким образом, если кристалл успешно проходит самотестирование в динамическом режиме, это гарантирует, что он не содержит ошибок и удовлетворяет требованиям по производительности для худшего случая, т. е. речь идет о проверке типа «идет — не идет». Однако можно также измерить быстродействие данного кристалла по отношению к другим кристаллам того же типа, если связать внутренние ЭСЭП в кольцевой генератор и измерить его частоты.

MICROBIT — метод самотестирования сложных МПУ. С ростом сложности систем и устройств на базе микропроцессоров затраты на их испытания резко возрастают, что делает практически обязательным внедрение функционального самотестирования на уровнях как схемных плат, так и микросистем в целом. Попыткой решить эту проблему явился метод самотестирования микропроцессорных систем MICROBIT, который требует введения на плату некоторых дополнительных схем.

MICROBIT — общий метод испытания печатных схемных плат, содержащих микропроцессор и представляющих собой какое-либо МПУ. Его можно применять независимо от типа микропроцессора, однако для проверки возможностей аппаратурной и программной реализации метода MICROBIT был выбран 8-разрядный микропроцессор 8085. Последующее испытание опытного образца МПС подтвердило

потенциальные достоинства этого метода, в числе которых можно назвать такие:

исключение потребности в оборудовании для функционального тестирования — проверка осуществляется полностью автономно;

ускорение поиска неисправностей благодаря встроенным средствам диагностики;

исключение затрат, необходимых для подготовки соответствующих тест-векторов, поскольку все тест-программы хранятся в постоянной памяти МПС;

существенное упрощение испытаний в условиях эксплуатации благодаря автономной природе встроенных тестов;

относительно малое время проверки.

Метод MICROBIT предусматривает введение в микропроцессорную систему дополнительных аппаратурных и программных средств для обеспечения возможностей самотестирования и исключения обычной потребности во внешнем оборудовании. Цель метода — обеспечить испытание системы на функциональном уровне в предположении, что различные приборы, входящие в ее состав, ранее проверялись и успешно выдержали параметрические испытания. Для ускорения поиска неисправностей в системе предусмотрены также соответствующие диагностические средства. Точность обнаружения ошибок посредством встроенных диагностических средств зависит от объема дополнительных аппаратурных и программных средств, на который может пойти разработчик.

На рис. 4.15 представлена структурная схема микропроцессорной системы, используемой для проверки концепции MICROBIT.

Для реализации метода MICROBIT в схему базовой МПС были введены четыре функциональных блока: генератор псевдослучайных кодов (ГПСК), сигнатурный регистр, делитель частоты и кнопка «Пуск испытаний» для управления началом проверки. Хотя все эти блоки состоят из готовых приборов и их функции почти очевидны, существуют некоторые особенности, связанные с включением логики MICROBIT в микросистему. Поэтому необходимо более подробно рассмотреть схему, приведенную на рис. 4.15.

Блок синхронизации и сброса управляет основной переменной диаграммой работы системы. Триггеры в системе переводятся в начальное состояние по сигналу сброса, вырабатываемому этим блоком.

В систему введены расширители (усилители сигналов шины), предотвращающие возможное ухудшение параметров сигналов, поскольку аппаратурные средства системы в целом размещаются на нескольких платах. Если бы все компоненты МПС размещались на одной плате, никаких усилителей сигналов шины не потребовалось бы.

Блок управляющей логики фактически состоит из нескольких субблоков, которые вырабатывают синхронизирующие и управляющие импульсные сигналы. Имеются следующие субблоки: схемы готовности, псевдопамяти, подтверждения адресов памяти и ввода — вывода. Эти схемы выполняют две функции: 1) дают возможность микропро-

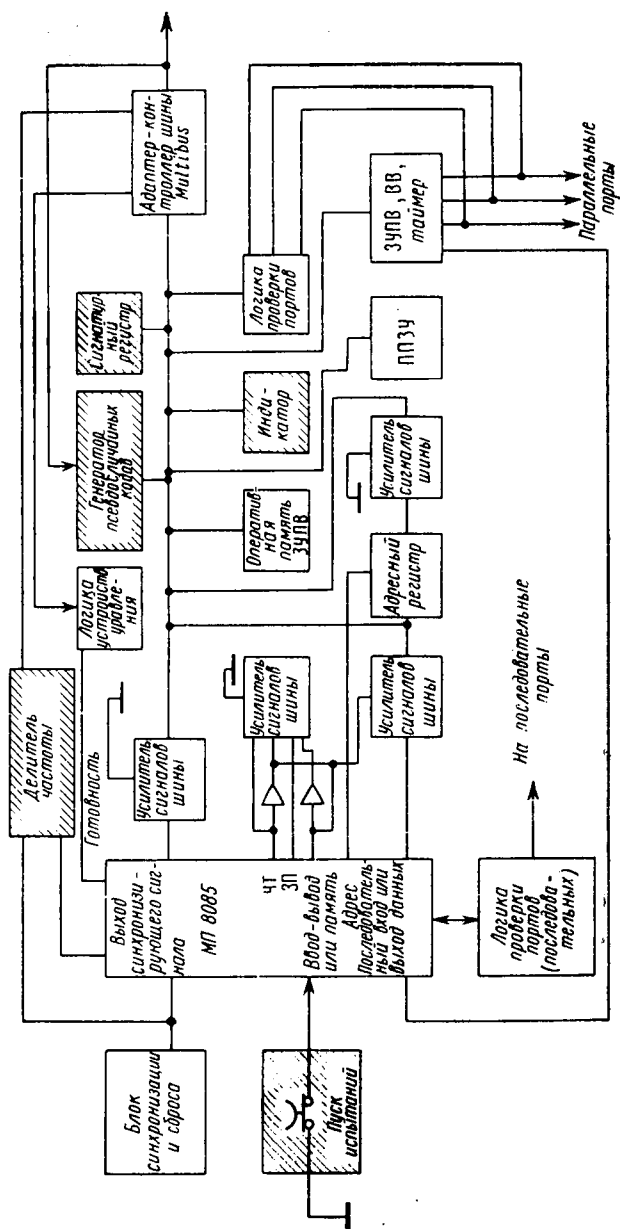


Рис. 4.15. Структурная схема МПС с самодиагностикой по методу MICROBIT

цессору различать сигналы, поступающие изнутри и извне платы; 2) контролируют адреса памяти для дополнительных аппаратных средств самопроверки. Кроме того, в логическом блоке проверки портов содержится несколько схем декодирования адресов, которые формируют сигналы разрешения и синхронизации для демультиплексора и выходного буфера. Логика тестирования последовательных портов позволяет принимать данные последовательного ввода — вывода и возвращать их в микропроцессор. Эта логика содержит также мультиплексор-коммутатор, позволяющий дифференцированно обрабатывать и выбирать реальные или тестовые данные. Схема делителя содержит три каскада делителя для уменьшения основной частоты синхронизации системы до значений, необходимых для проверки. Генератор псевдослучайных кодов — четырехфункциональный линейный регистр из восьми триггеров с тремя цепями обратной связи. В зависимости от логического уровня из управляющей линии этот регистр может выполнять функции либо генератора псевдослучайных кодов, либо сигнатурного регистра.

Для реализации метода MICROBIT регистр в режиме ГПСК вырабатывает повторяющуюся последовательность из 256 псевдослучайных чисел.

Вторая составляющая часть метода MICROBIT — его программные средства. Вся программа, написанная на языке ассемблера, разбита на девять автономных и независимых модулей. Таким образом, если в МПС отсутствует какой-то конкретный функциональный блок, то соответствующий модуль можно обойти или опустить. Цель разработчиков программного обеспечения — объединить эти модули с прикладной программой и разместить их вместе в ППЗУ.

В процессе работы (когда включено питание МПС, все триггеры сброшены и МПС находится в начальном состоянии) прежде всего выполняется основной тест — тест ядра. Он обеспечивает проверку микропроцессора, шин, блока синхронизации и сброса, линий адресов и данных. Это делается путем сложения двух специально выбранных чисел 10101010 и 01010101 и сравнения полученного результата с числом 11111111.

Для проверки модуля последовательного ввода — вывода в самый старший разряд накапливающего регистра последовательно помещаются «1» и «0» (по одному значению за цикл). Затем эти значения выдаются на контакт выхода последовательных данных микропроцессора, которые принимаются логикой тестирования последовательных портов и возвращаются на контакт входа последовательных данных микропроцессора. Каждый раз принятые данные сравниваются с ранее выданными значениями. Если несовпадения нет, это означает, что последовательный порт работает правильно.

Практика применения метода MICROBIT выявила определенные недостатки этого метода самодиагностики. Одна из проблем заключается в том, что такое тестирование имеет ограниченную разрешающую способность. Другим недостатком метода является то, что он преду-

считывает автономное тестирование, в то время как во многих случаях желательно производить проверку системы в процессе выполнения основных функций. Оперативное тестирование по существу повышает надежность системы, поскольку позволяет обнаруживать функциональные ошибки сразу же при их возникновении.

Анализ особенностей метода MICROBIT показывает следующее: диагностические тесты должны быть ориентированы на структуру МПС, тогда проигрыш во времени, получающийся при диагностике, восполняется сокращением времени на локализацию неисправности; для полной проверки МПС важно сочетать функциональные тесты и тесты, ориентированные на структуру МПС; для сокращения времени на диагностику тесты должны располагаться в ПЗУ МПС и иметь микропрограммную реализацию.

§ 4.6. ОРГАНИЗАЦИЯ ВСТРОЕННОЙ ДИАГНОСТИКИ МПУ

Общие принципы самотестирования. Постоянно растущая функциональная сложность интегральных микросхем, а также микроЭВМ и МПУ, построенных на их основе, делает необходимым реализацию тестовой диагностики на всех этапах жизненного цикла устройств. Средства самотестирования прежде всего нужны для микроЭВМ и МПУ, выпускаемых большим тиражом, например для персональных ЭВМ. В § 4.4, 4.5 были проанализированы основные методы и средства самотестирования. Очевидно, не существует четких рекомендаций относительно выбора методов самодиагностики в самом общем случае, но ясно, что в основном выбор метода зависит от элементной базы.

Решение проблемы тестовой диагностики для МПУ длительное время возлагалось на внешние средства — автоматизированные стенды контроля. Но в связи со все возрастающей сложностью и насыщенностью микропроцессорных плат эффективность методов внешнего диагностирования перестала устраивать разработчиков, поэтому стали делать изменения в структуре микропроцессоров.

Структура микропроцессора модифицируется таким образом, чтобы непосредственно, без выполнения команд МП, производить запись и чтение всех регистров. При этом большая часть регистров МП и логики устройств управления становится доступной для непосредственного наблюдения в режиме тестовой диагностики. Аппаратурные затраты возрастают на 10%, а вероятность обнаружения одиночных неисправностей достигает 98 %.

Организация автономного процесса тестовой диагностики предполагает наличие схем, задающих тестовые воздействия, и схем, анализирующих реакции диагностируемого устройства. При этом удобно сочетать небольшие по объему тестовые программы, ориентированные на проведение установочных экспериментов, и программы генерации достаточно длинных псевдослучайных последовательностей (2^{16} и более) причем в качестве методов анализа используется арифметическое суммирование или сигнатурное сжатие. Такой состав дополнительного

оборудования позволяет организовать самотестирование сложных устройств, реализованных на базе однокристалльных или секционных МП.

Диагностирование ОЗУ требует иной реализации метода самотестирования. Широкий класс неисправностей ЗУ с произвольной выборкой обнаруживается с помощью аппаратно реализованных алгоритмических тестов проверки.

Таким образом, к числу обязательных функций, которые необходимы при реализации любого метода встроенного самотестирования, следует отнести генерацию псевдослучайных последовательностей (кодов), сигнатурное сжатие, арифметическое суммирование и синдромное сжатие (подсчет числа переходов, единиц и т. п.), а также один или несколько алгоритмов проверки ОЗУ.

Структуры БИС для встроенных средств контроля. Для тестового диагностирования процессоров, устройств памяти, процессоров ввода вывода и схем произвольной логики разработан комплект БИС серии КР1828. Интегральная реализация тест-структур позволяет конструктивно объединять их с объектами контроля и осуществлять проверку по принципу «годен — не годен» на реальных частотах работы устройств, обеспечивая эффективное обнаружение не только контактных неисправностей и замыканий, но и параметрических отказов.

Наибольший эффект достигается при встраивании тест-структур в каждый типовой элемент замены (ТЭЗ), что позволяет быстро обнаруживать и заменять отказавший ТЭЗ. Кроме того, тест-структура может быть использована для поиска неисправности в ТЭЗе, исключая необходимость в применении сложных и дорогостоящих стендов.

Микросхемы серии КР1828 изготовлены по ТТЛШ-технологии и совместимы по уровням сигналов и номиналам питающего напряжения с широко распространенными МПК БИС. Рассмотрим структурные схемы БИС, предназначенных для построения ядра схем встроенной диагностики процессоров различного типа, ПЗУ/ППЗУ и схем произвольной логики (БИС КР1828ВЖ1), а также БИС для построения схемы встроенной диагностики ОЗУ (БИС КР1828ВЖ2).

В БИС КР1828ВЖ1 устройство управления УУ на основе заданной команды и внешних сигналов формирует необходимые сигналы для работы всех узлов БИС (рис. 4.16). Операционное устройство ОУ содержит сумматор с аккумуляторами и выполняет все необходимые преобразования получаемых данных. Компаратор сравнивает полученные результаты с эталонными значениями, хранящимися во внешнем ПЗУ. Схема входного переноса обеспечивает работу операционного устройства в режиме сигнатурного анализа. Схема индикации состояний предназначена для формирования результата тестирования в виде сигналов Да и Нет. БИС построена по разрядно-модульному принципу, что позволяет практически неограниченно увеличивать длительность формируемых и анализируемых сигналов. Микросхема имеет восемь разрядов с возможностью наращивания разрядности.

БИС может функционировать в одном из следующих режимов: генератора псевдослучайных кодов с максимальным периодом 255;

генератора инкрементируемых и декрементируемых последовательностей (от 0 до 255 и обратно); сигнатурного анализатора; счетчика числа переходов и сумматора со сравнением полученной и контрольной сумм.

БИС КР1828ВЖ2 предназначена для применения в качестве ядра схемы встроенной диагностики ОЗУ. Следует отметить, что для современных БИС ОЗУ не существует универсального проверяющего теста практически приемлемой длины. Положение еще более осложняется, если объектом контроля служит ячейка, содержащая несколько десятков БИС ОЗУ.

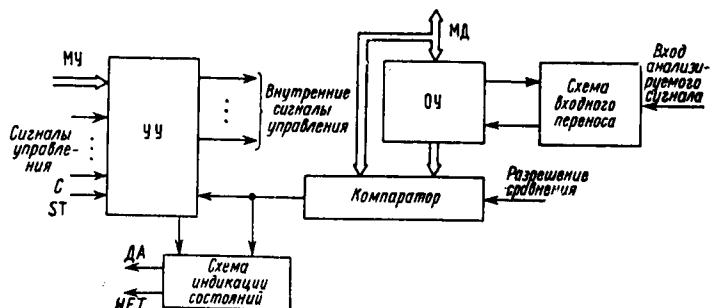


Рис. 4.16. Структурная схема БИС КР1828ВЖ1

Для аппаратной реализации в виде БИС КР1828ВЖ2 (рис. 4.17) выбраны такие тесты: обращение по прямому и дополнительному адресам; пропорциональное деление; бегущие по разрядам «1» и «0». Первые два теста предназначены для проверки накопителя и схем дешифрации ОЗУ, последний — для диагностики информационного тракта. Суммарная длина тестов $40n$, где n — число ячеек ОЗУ. На рис. 4.16, 4.17 буквой C обозначен вход тактовых импульсов, $\overline{ST}$ — вход включения микросхемы в рабочий режим.

Пример построения МПУ со встроенной диагностикой. Схемы самотестирования, основу которых составляют рассмотренные в этом параграфе БИС, целесообразно размещать на сменных узлах (платах) МПУ. Схемы самотестирования (иногда называемые тест-структурами) по сигналу с инженерного пульта осуществляют тестовую диагностику сменного узла, в котором они размещены, на рабочей частоте по принципу «годен — не годен».

Для самотестирования процессоров, имеющих стандартную архитектуру, и контроллеров с микропрограммным управлением может быть использована тест-структура, представленная на рис. 4.18. Генератор тестовых воздействий формирует тестовую последовательность определенной длины (в данном случае 24-разрядную), которая отождествляется с микрокомандой тестируемого БМПУ проверяемого процессора. Поля микрокоманды с запрещенными кодами формируются

с помощью ПЗУ, которое может адресоваться генератором псевдослучайных кодов ГПСК. Максимальное число тестовых последовательностей Т, формируемых ГПСК при использовании двух схем КР1828ВЖ1, равно 64770. Обязательное условие при тестировании — установление тестируемого процессора в некоторое заранее известное состояние.

Анализ реакций на тестовые воздействия осуществляется при регистрации значений сигналов в N заранее выбранных контрольных точках КТ на всех тестовых воздействиях. Получаемая в контрольной

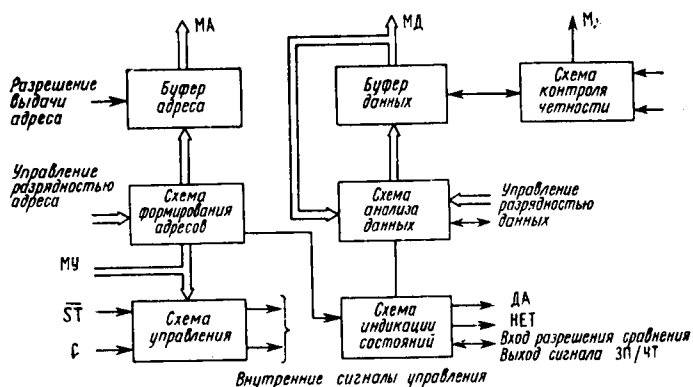


Рис. 4.17. Структурная схема БИС КР1828ВЖ2

точке КТ Т-разрядная последовательность превращается в процессе эксперимента в 8-разрядную сигнатуру, которая сравнивается со значением эталонной сигнатуры.

Схема анализа реакций состоит из коммутатора, разрядность которого равна числу контролируемых точек (в данном случае $N=32$), счетчика на основе БИС КР1828ВЖ1, ПЗУ эталонных сигнатур (возможна их реализация на БИС КР556РТ5) и 8-разрядного сигнатурного анализатора на БИС КР1828ВЖ1. Таким образом, показанная на рис. 4.18 схема 32 раза формирует тестовую последовательность и сравнивает эталонные 8-разрядные реакции с полученными в каждой из 32 точек. Схема управления тест-структурой состоит из двух вентилей и триггера. Запуск тестовой последовательности для контроля следующей КТ осуществляется после получения результата контроля предыдущей КТ (по сигналам «Годен» или «Не годен» и сигналу «Пуск» с пульта).

В заключение сформулируем основные требования к организации самотестирования микропроцессорных устройств:

в процессе тестирования обеспечить разрыв обратных связей, которые могут привести к неоднозначности при смене состояний МПУ под воздействием тестовой последовательности;

по возможности добиться уменьшения числа фронтов синхронизирующих сигналов, по которым происходит изменение логических уровней диагностируемой схемы;

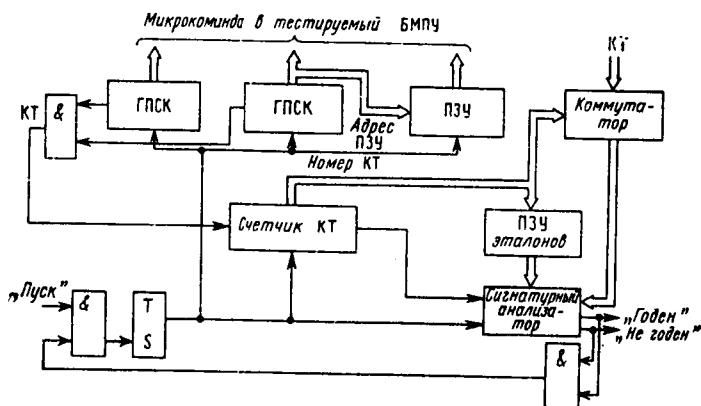


Рис. 4.18. Схема тест-структуры процессора

Частично такая информация может быть получена из номера контрольной точки, по которой произошло несовпадение с эталоном. Для получения более точной информации следует организовать поиск неисправностей с помощью щупа, в котором совмещен сигнатурный анализатор и счетчик числа переходов.

§ 4.7. КОМПЛЕКСНАЯ ОТЛАДКА МПУ

213

Исходными данными для работы программного обеспечения встроенных МПУ служат параметры электрических сигналов, поступающих на вход МПУ. Критериями оценки правильности функционирования МПУ и его программного обеспечения являются последовательность генерируемых МПУ сигналов и их параметры. Отладка программного обеспечения с помощью кроссовых средств позволяет выявить только ошибки логического характера, а ошибки в сложных временных зависимостях между различными выходными сигналами МПУ остаются невыявленными. Поэтому окончательная отладка программного обеспечения встроенных МПУ возможна только на реально функционирующем прототипе аппаратуры. Необходимость отладки программного обеспечения на реальной аппаратуре, функционирующей в реальном времени, требует использования в процессе комплексной отладки специальных аппаратно-программных средств, выполняющих функции имитации различных воздействий на МПУ, управления вычислительным процессом и имеющих возможность сбора и индикации информации о состоянии МПУ в любой момент его функционирования.

Как отмечалось, проектирование МПУ представляет собой сложный многоступенчатый итерационный интерактивный процесс. Для его осуществления требуются разнообразные методы и средства диагностики и отладки. При разработке программного обеспечения эти средства применяются при вводе и редактировании текста, в процессе трансляции и отладки программы. При создании прототипа МПУ средства отладки используются при отладке узлов и блоков макета. При совместной отладке аппаратурных и программных средств МПУ отладочные средства используются при стыковке различных блоков МПУ и МПУ с управляемыми РЭС, программировании ПЗУ и последующем тестировании всех узлов МПУ.

Назначение отладочных средств — обеспечение минимальных затрат времени на проектирование. В качестве критерия оценки этого главного требования к отладочным средствам часто применяют обобщенный показатель — производительность отладочных средств, под которой понимают величину, обратную общему времени проектирования МПУ. При этом очевидно, что время, затрачиваемое на разработку конкретного МПУ, зависит не только от характеристик и функциональных возможностей отладочных средств, но и от сложности МПУ, квалификации проектировщика и ряда случайных факторов.

С точки зрения обеспечения отладочными средствами всю номенклатуру МП и однокристальных микроЭВМ можно условно подразделить на три группы:

- 1) процессорно-ориентированные МП с СМ — системой команд (МПК БИС серий К581, К588, К1801 и др.). В качестве отладочных средств для этих МП могут служить одна из модификаций микроЭВМ «Электроника 60», ДВК или ЭВМ с системой команд СМ ЭВМ. В этом случае отладочный комплекс используется как инструментальная ЭВМ;

2) секционные микропроцессорные микропрограммируемые наборы (например, К583, К584, К589, К1802, К1804 и др.). Для МП этой группы существуют универсальные программные и аппаратные комплексы, в частности система МЕТАМИКРО;

3) МП и однокристальные микроЭВМ с фиксированной системой команд (К1814, К1820, К1816, КР580, КР1810 и др.). Для разработки МПУ и микроЭВМ этой группы требуются универсальные отладочные комплексы с развитым программным обеспечением.

По составу технических средств, программного обеспечения и выполняемым функциям отладочные средства можно подразделить на три класса: отладочные устройства, отладочные комплексы и отладочные программные средства.

Отладочные устройства. Для отладки аппаратных и программных компонентов МПУ используются отладочные устройства. Они характеризуются прежде всего автономностью, низкой стоимостью, малыми габаритными размерами. К отладочным устройствам предъявляются следующие требования: обеспечение работы в реальном времени, относительная универсальность, наличие канала связи с персональной ЭВМ, обеспечение автоматического и пошагового режимов, вывод на индикацию содержимого внутренних регистров.

Типовая архитектура отладочного устройства приведена на рис. 4.19. Отладочная модель обеспечивает выполнение программ в реальном времени, а также создает канал связи с устройством пользователя (канал В). Канал позволяет в отсутствие реального МПУ набирать физическую модель МПУ из отдельных элементов. Отладочная модель строится на том типе микропроцессора или микроЭВМ, на котором разрабатывается МПУ. Блок загрузки и управления БЗУ осуществляет запись и чтение программ пользователя, обеспечивает необходимые режимы выполнения программ и индикацию содержимого РОН. Через БЗУ осуществляется связь с инструментальной микроЭВМ или персональной ЭВМ (канал А). Структурная схема типового автономного отладочного устройства приведена на рис. 4.20. Оно может работать как отдельно, так и в составе комплексной универсальной отладочной системы (рис. 4.21). В состав отладочной модели (см. рис. 4.20) входит ПЗУ, в котором хранятся служебные команды. ПЗУ подключается к отладочной БИС в точке останова программы пользователя и обеспечивает программу чтения регистров и восстановления статуса. Служебное ПЗУ подключается к отладочной БИС внутри командного цикла после завершения считывания программы из ОЗУ, которое является основной программной памятью. Важно (хотя для однокристальных микроЭВМ это технически затруднительно) зафиксировать статус в момент останова и восстановить его при возобновлении прогона.

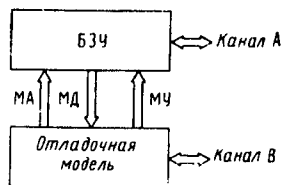


Рис. 4.19. Типовая архитектура отладочного устройства

В состав БЗУ для реализации возлагаемых на устройство функций включены ОЗУ и микроЭВМ. В ОЗУ хранятся программы пользователя и перечень контрольных точек. МикроЭВМ управляет работой ОЗУ, дисплея, а также через интерфейс — связью с инструментальной микроЭВМ. Наличие на выходе БЗУ магистралей адреса, данных и управления позволяет подключать программаторы различных типов.

При выполнении БЗУ на базе персонального компьютера и дополнении отладочного устройства кросс-ассемблером отладочное устройство трансформируется в отладочную систему с контрольно-диагностическими функциями и возможностью выпуска технической документации.

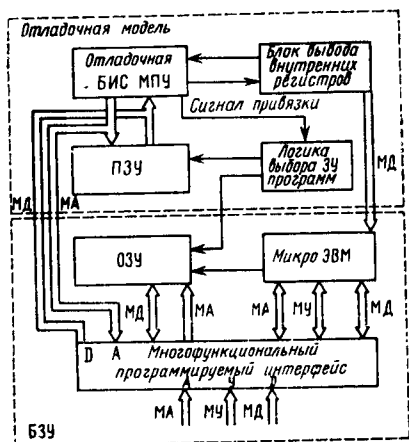


Рис. 4.20. Структурная схема автономного отладочного устройства

имеют более высокую производительность по сравнению с отладочными устройствами, которая обеспечивается сопряжением всех аппаратных и программных средств по конструкции, интерфейсам, электрическим уровням, форматам команд и данных и т. п. Использование схемных и системных эмуляторов в отладочных комплексах позволяет приступить к отладке программных средств параллельно с разработкой и изготовлением физической модели МПУ, а по мере ее готовности постепенно наращивать ресурсы отлаживаемого устройства, последовательно подключая цепи синхронизации, магистраль, устройства ввода — вывода, ОЗУ и ПЗУ, что также повышает производительность при разработке МПУ.

Построение универсального отладочного комплекса с максимальными функциональными возможностями при наименьших затратах на разработку и умеренной стоимости требует создания аппаратных и программных средств по функционально-модульному принципу, ориентации на использование в качестве базового вычислительного комплекса серийно выпускаемых микроЭВМ, дополнения его смежными блоками эмуляторов и комплексами отладочных программ для каждого

целевого процессора, применения стандартных операционных систем и интерфейсов, унификации всех аппаратных и программных средств по конструкции, интерфейсам, системной магистрали и системе команд, максимальной реализации функциональных возможностей с помощью программных средств.

Типовая структура универсального отладочного комплекса для однокристальных микроЭВМ и МПУ показана на рис. 4.21. Отладочный комплекс состоит из ведущей стандартной ЭВМ и дополнительно аппаратно-программного блока, который исполняет общие (для отладки всех однокристальных микроЭВМ) функции и учитывает особенности настраиваемых микроЭВМ или МПУ. Такой блок в комплексе отладки, промежуточный между ведущей ЭВМ и устройством пользователя, называется системным эмулятором (СЭ). Системный эмулятор должен обеспечивать в реальном времени достоверную эмуляцию, так как функции ведущей ЭВМ и системного эмулятора разделены. В отладочном комплексе ведущая ЭВМ и системный эмулятор представляют собой самостоятельные блоки, программное обеспечение которых создается в большей мере независимо. СЭ построен по модульному принципу, его аппаратная и программная части должны допускать несколько конфигураций с различной степенью сложности, имеющих максимум удобств при отладке, тестировании и диагностике устройств с фиксированной системой команд.

Рассмотрим назначение и основные функции блоков отладочного комплекса.

Интерфейс к ведущей ЭВМ и контроллер СЭ принимают команды и данные от ведущей ЭВМ и передают данные обратно; декодируют адреса и принятую информацию; вводят комплекс в один из режимов: диалога с ведущей ЭВМ, выполнения мониторинговых программ СЭ и выполнения (прогона) программы пользователя. Кроме того, интерфейс содержит аппаратные средства для исполнения программы пользователя в пошаговом режиме.

Логика задания точек останова автоматически вырабатывает команды останова при совпадении заданных с пульта оператором условий останова с текущей информацией о состоянии комплекса. В про-

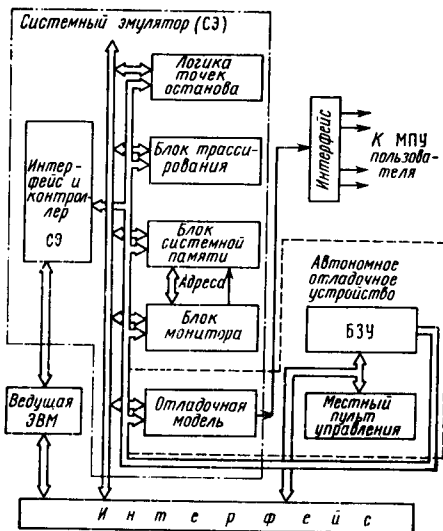


Рис. 4.21. Структура универсального отладочного комплекса

стейшем случае сравниваются заданные адреса остановов с состояниями счетчика команд.

Монитор (блок управляющих подпрограмм) распределяет память системного эмулятора, отображая виртуальный адрес программной памяти пользователя в физический адрес системной памяти. Он включает также различные подпрограммы: распечатывает содержимое внутренних регистров при останове, а также восстанавливает их при переходе к исполнению программы; управляет доступом к информации о состоянии периферийных устройств системы пользователя. Это придает отладочной системе черты диагностической системы.

Блок системной памяти загружается объектными модулями, полученными в результате работы кросс-ассемблера, а также служебными программами для выполнения системных команд.

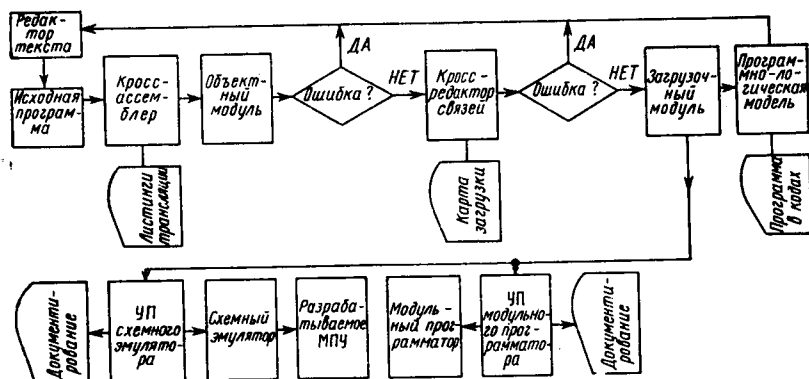


Рис. 4.22. Структура отладочных программных средств

Блок трассирования служит для записи состояний системы (включая логические узлы устройства пользователя). Глубина трассировки должна быть не меньше числа адресов на одной странице командной памяти отлаживаемого процессора.

Отладочная модель, входящая также в состав автономного отладочного устройства, — единственный блок СЭ, который полностью заменяется при смене типа отлаживаемых МПУ или микроЭВМ. Его основа — отладочная БИС МПУ (см. рис. 4.20). В режиме прогона эта БИС задает выборку адресов и интерпретирует полученные из программной памяти команды. Другой его элемент — ПЗУ, в котором записаны все признаки отлаживаемой микроЭВМ, а также служебные подпрограммы для прочтения содержимого внутренних регистров при останове.

Отладочные программные средства. Для отладки программного обеспечения МПУ, а также аппаратурных средств МПУ на отладочном комплексе предназначены программные отладочные средства. Ти-

новой пакет отладочных программ содержит редактор текста, кросс-ассемблер, кросс-редактор связей, программно-логическую модель и управляющие программы (УП) модульного программатора и схемного эмулятора (рис. 4.22).

Кросс-ассемблер формирует объектный модуль, который преобразуется в загрузочный с помощью кросс-редактора связей. В случае необходимости кросс-ассемблер может выдать листинг программ, а кросс-редактор связей — карту загрузки.

Программно-логическая модель осуществляет покомандную интерпретацию (моделирование) загрузочного модуля с целью проверки выполнения. При необходимости модель может выдать листинг программы в кодах, а также справочную информацию. В режиме трассировки программно-логическая модель обеспечивает индикацию или вывод на печать адреса выполняемой команды, кода команды и операндов, состояние рабочих регистров модели МПУ и слова состояния. Трассировка может выполняться как по всей программе, так и по ее части. При моделировании команд ввода — вывода индицируется или печатается вводимая или выводимая информация.

Окончательная отладка программы должна производиться с помощью управляющих программ схемного эмулятора и программатора, образующих программы поддержки аппаратуры МПУ.

Контрольные вопросы и задания

1. Чем объясняется необходимость разработки аппаратурных и программных средств для отладки МПС?
2. В чем заключается сущность функционального и структурного контроля?
3. Что такое сигнатура и почему она может быть использована в диагностике МПС?
4. Какие факторы влияют на точность сигнатурного анализа?
5. Почему требуется дополнительное оборудование для проведения сигнатурного анализа?
6. Что такое самотестирование БИС?
7. Какие дополнительные аппаратные средства необходимы для организации самотестирования БИС?
8. Сравните эффективность организации динамического и параллельного тестирования.
9. Почему диагностирование ОЗУ не может быть осуществлено с помощью средств, диагностирующих логические схемы?
10. Чем вызвана необходимость проведения комплексной отладки МПУ?
11. Какие функции выполняет схемный эмулятор?
12. Какие средства требуются для отладки работы системного интерфейса?

Глава 5

ОРГАНИЗАЦИЯ МИКРОЭВМ

§ 5.1. АРХИТЕКТУРА МИКРОЭВМ

Основные понятия и определения. Под *архитектурой* ЭВМ понимают абстрактное представление машины в терминах основных функциональных модулей, языка ЭВМ, структуры данных. Архитектура не определяет особенностей реализации аппаратной части ЭВМ, времени выполнения команд, степени параллелизма при выполнении программы, ширины магистралей и других аналогичных характеристик. Она отображает аспекты структуры ЭВМ, которые являются видимыми для пользователя: систему команд, режимы адресации, форматы и длину данных, набор регистров ЭВМ, доступных пользователю. Одним словом, термин «архитектура» используется для описания возможностей, предоставляемых ЭВМ, а термин «организация» определяет, как эти возможности реализованы.

Все ЭВМ, в том числе и микроЭВМ, содержат следующие функциональные блоки, имеющие свою микроархитектуру: процессор, состоящий из АЛУ и устройства управления, память, устройства ввода и вывода информации. Объединение функциональных блоков в ЭВМ осуществляется посредством системы магистралей (шин): шины данных, по которой осуществляется обмен информацией между блоками ЭВМ; шины адреса, используемой для передачи адресов, по которым осуществляется обращение к различным устройствам ЭВМ, и шины управления для передачи управляющих сигналов.

Для связи пользователя с ЭВМ предусмотрен пульт управления, позволяющий выполнить такие действия, как пуск ЭВМ; останов, под действием которого прекращается поступление сигналов с генератора тактирующих сигналов и процессор переходит в состояние ожидания; загрузка начального адреса программы в программный счетчик; индикация содержимого ячеек памяти и регистров процессора; пошаговое выполнение команд программы при ее отладке.

Конфигурации микроЭВМ. Все множество выпускаемых микроЭВМ можно классифицировать по их конфигурациям таким образом: встраиваемые микроЭВМ, конструктивно приспособленные для встраивания в технологическое и другое оборудование (станок, прибор, машина, РЭС и др.) в качестве сборочной единицы и не имеющие индивидуального пульта управления, источника электропитания, декоративного конструктивного оформления. Если встраиваемые микроЭВМ выполнены в виде одной БИС, то они называются однокристальными, если в виде одной печатной платы — одноплатными;

портативные сервисные микроЭВМ, содержащие небольшой дисплей, размещающиеся в небольшом чемоданчике и легко транспортируемые;

микроЭВМ настольной конструкции со встроенными, как правило, в единую конструкцию дисплеем, пультом, малогабаритным устройством печати; память на гибком диске или кассетном магнитофоне. Их часто используют в качестве персональных микроЭВМ;

микроЭВМ настольной и стоечной конструкций, содержащие набор отдельных конструктивно законченных модулей: процессор, экранный пульт, печатающее устройство, память на гибких дисках и малогабаритных винчестерских твердых дисках, функционально объединенных в целостную автономную микросистему. Их можно использовать в качестве персональных микроЭВМ, интеллектуальных терминалов вычислительных систем коллективного пользования, абонентских или рабочих станций локальных вычислительных сетей.

Организация микроЭВМ на однокристальных МП. Однокристальные МП включают в себя все аппаратурные средства центрального процессора — регистры, АЛУ, схемы управления, а в некоторых случаях даже память. Поэтому микроархитектура МП определяет многие

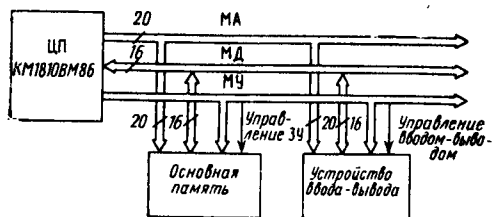


Рис. 5.1. Структурная схема микроЭВМ на МП КМ1810ВМ86

из черт микроархитектуры микроЭВМ (емкость и расположение регистров, разрядность и трассировку магистралей данных и управления, наличие и функции буферных регистров и магистралей для передачи всех сигналов). Однокристальные МП имеют ограниченную разрядность и жесткую систему команд. Наиболее распространенными однокристальными микропроцессорами являются центральные процессорные элементы КР580ИК80А (8 разрядов), КМ1810ВМ86 (16 разрядов) и ряд других.

При разработке структур микроЭВМ на однокристальных МП основными задачами являются: определение состава интерфейсных устройств ввода — вывода; определение состава памяти; разработка системы прерываний; разработка системы адресации и внутреннего интерфейса микроЭВМ; обеспечение допустимой нагрузочной способности компонентов микроЭВМ.

Возможно множество вариантов структур микроЭВМ, построенных на однокристальных МП. В качестве примера рассмотрим способ построения микроЭВМ на однокристальном МП КМ1810ВМ86, упрощенная структурная схема которой приведена на рис. 5.1.

В микроЭВМ предусматривается использование изолированного ввода — вывода, и таким образом, по крайней мере логически, магистрали ввода — вывода и магистрали памяти могут быть разделены. Однако в интегральной схеме КМ1810ВМ86 и для магистрали ввода — вывода, и для магистрали памяти предусмотрены одни и те же зажимы.

Чтобы указать, относится ли передача данных, поступающих на совместно используемые зажимы, к магистрали ввода — вывода или к магистрали памяти, микропроцессор генерирует сигналы управления магистралями. Как показано на рис. 5.1, магистраль данных МД имеет 16 разрядов, а магистраль адреса МА — 20 разрядов. Благодаря этому можно адресоваться к памяти емкостью до 1 М байт. При выполнении операций ввода — вывода используется 16-разрядный адрес, поэтому центральный процессор ЦП кроме доступа к основной памяти может обращаться к регистрам интерфейса ввода — вывода (портам), суммарный объем которых составляет 64 К байт.

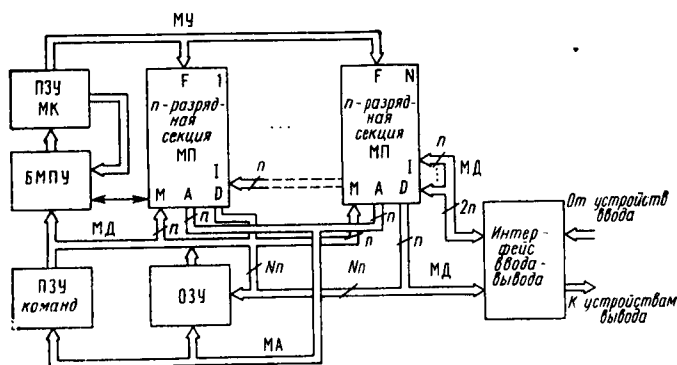


Рис. 5.2. Структурная схема микроЭВМ на секционных МП

Организация микроЭВМ на секционных МП. Построение микроЭВМ на базе микропрограммируемых секционных МП отличается более широкими возможностями модификации структуры микроЭВМ в соответствии с требованиями конкретного применения. Как правило, такие микропрограммируемые микроЭВМ требуют разработки двухуровневого управления — микропрограммного и программного. Это позволяет иметь произвольную систему команд, однако усложняет как структуру и архитектуру микроЭВМ, так и сам процесс проектирования. Напомним, что многокристальные секционные МП имеют разрядность $n = 2, 4, 8, 16$ бит. Упрощенная структурная схема микроЭВМ на секционных МП приведена на рис. 5.2.

В микроЭВМ с двумя уровнями управления программа размещается во внешней программной памяти, в ПЗУ команд или в ОЗУ. В качестве программного счетчика обычно используется один из Nn -разрядных РОН, расположенных в блоке из N микропроцессорных секций. Адрес команды выдается по выходной адресной шине блока секций на адресные входы программной памяти.

Секционные микропроцессоры различных серий позволяют создавать разнообразные архитектуры высокопроизводительных процессоров ЭВМ.

§ 5.2. ОСОБЕННОСТИ ПОСТРОЕНИЯ СОВРЕМЕННЫХ МИКРОЭВМ

Влияние элементной базы на архитектуру микроЭВМ. Технология изготовления БИС оказывает существенное влияние на архитектуру микроЭВМ, так как каждая из технологий имеет определенные ограничения. Элементы логических схем характеризуются рядом параметров, в частности амплитудой переключаемых сигналов, напряжениями питания и др. В силу тепловых ограничений для каждой элементной базы существует предельное значение степени интеграции N_p . Полагая, что в наихудшем случае в схеме могут работать все логические элементы, расположенные на кристалле, имеем

$$N_p = P_{кр} S / P,$$

где $P_{кр}$ — допустимая удельная рассеиваемая мощность кристалла, Вт/см²; S — площадь кристалла, см²; P — средняя мощность, рассеиваемая логическим элементом, Вт. Следует отметить, что значение величины N_p максимально для КМОП-технологии и минимально для ЭСЛ-технологии.

Современный уровень развития технологии позволяет изготовить СБИС со степенью интеграции в несколько десятков элементов на кристалл, что зачастую приводит к необходимости многокристальной организации систем (особенно для ЭСЛ- и ТТЛШ-технологий), накладывая определенные ограничения на архитектуру. Многокристальная организация при низких значениях предельных степеней интеграции возможна при микропрограммной организации управления и разбиении операционного устройства на секции малой разрядности. Примерами такой организации могут служить микроЭВМ на базе МПК К589, КР1804 и др. Можно сказать, что такой путь — это путь создания относительно несложных микроЭВМ, обладающих высоким быстродействием.

Другой путь использования ЭСЛ- и ТТЛШ-технологий — построение микроЭВМ на базовых кристаллах. При таком подходе центральный процессорный элемент разбивается на некоторое количество частей, не имеющих, как правило, самостоятельного функционального значения. Каждая из частей реализуется на индивидуальном базовом кристалле, затем кристаллы объединяются в систему. Этот путь позволяет получить наиболее высокопроизводительные и мощные процессорные элементы, однако требует больших материальных затрат и используется для создания мощных ЭВМ.

Для создания однокристалльных микроЭВМ наиболее пригодны элементные базы типов n -МОП, И²Л и КМОП, имеющие высокие значения N_p .

Таким образом, архитектурно-структурная организация микроЭВМ определяется типом элементной базы, на основе которой она реализуется, и возможностью применения тех или иных архитектурно-структурных методов повышения производительности, главными из которых являются распараллеливание процесса выполнения операций,

аппаратная реализация программных функций и увеличение числа уровней иерархии памяти.

Распараллеливание процесса выполнения операций. Под распараллеливанием понимают одновременную обработку двух или более частей одного либо нескольких заданий (алгоритмов, команд, микрокоманд) двумя или более модулями (микроЭВМ, микропроцессорами и т. д.). В зависимости от того, осуществляется ли одновременная обработка нескольких частей одного задания или сразу нескольких заданий, различают следующие типы распараллеливания:

1. Параллельное выполнение различных независимых частей одного задания несколькими модулями и объединение частных результатов для получения общего результата по окончании выполнения задания. В случае применения данного типа распараллеливания производительность системы

$$P_1 = P_0 \frac{\sum_{i=1}^n t_i}{\max_i \{t_i\} + t_{\Sigma}}, \quad (5.1)$$

где P_0 — производительность микроЭВМ без распараллеливания; t_i — время, необходимое модулю i для выполнения своей части задания; t_{Σ} — время, необходимое для объединения частных результатов; n — глубина распараллеливания, равная числу частей задания или соответственно числу модулей.

Максимальная производительность может быть достигнута при одинаковых временах выполнения всех частей задания. Объем оборудования при этом возрастает более чем в n раз по сравнению с вариантом микроЭВМ без распараллеливания.

2. Конвейерный режим работы модулей, в котором осуществляется параллельное выполнение частей различных заданий различными модулями. При этом процесс выполнения каждого задания разделяется на несколько приблизительно одинаковых по времени исполнения последовательных частей, выполняемых соответственно в различных специализированных модулях. Работа модулей организуется таким образом, что любое последующее задание начинает выполняться системой модулей до того, как будет выполнено предыдущее задание. При распараллеливании конвейерного типа производительность

$$P_2 = P_0 \frac{\sum_{j=1}^k \tau_j}{\max_j \{\tau_j\} + t_{\Pi}}, \quad (5.2)$$

где τ_j — время, необходимое модулю j для выполнения соответствующей части задания; k — число модулей; t_{Π} — время запоминания (с целью буферизации) промежуточных результатов каждого модуля.

Формула (5.2) справедлива для распараллеливания, при котором в любой период времени работают все модули (за исключением, разумеется, начальных и конечных периодов выполнения задания). Если в процессе работы часть модулей простаивает, то в формулу (5.2) следует ввести коэффициент, учитывающий относительное число простоев. Конвейерный метод позволяет достичь повышения производительности практически без дополнительных аппаратных затрат.

Отметим, что распараллеливание первого типа может иметь место только в случае, если задание допускает возможность разделения на *n* независимых частей. Конвейерное распараллеливание допускает подобную зависимость.

Одной из перспективных тенденций в области архитектуры универсальных микроЭВМ является тенденция к децентрализации обработки информации. Децентрализация функций ведет в повышению производительности системы в целом благодаря тому, что центральный процессор освобождается от такой непроизводительной работы, как управление работой внешних накопителей, обеспечение работы графических дисплеев, осуществление синхронизации процессов обмена и т. п.

Следует отметить, что наиболее эффективным способом повышения производительности является распараллеливание на программном уровне (на уровне задач). Повышение производительности в данном случае достигается распараллеливанием вычислений с помощью МПС и комплексов. Для этого несколько микроЭВМ объединяют в систему с макроархитектурой, соответствующей тому или иному классу выполняемых задач. В настоящее время существует стандартная классификация таких макроархитектур, основанная на соотношении потоков команд, выполняемых системой, и потоков данных, с которыми работают потоки команд.

Различают четыре вида макроархитектур.

SISD(Single Instruction Single Data Stream) — архитектура, в рамках которой один поток данных обрабатывается одним потоком команд, т. е. осуществляется обычная последовательная обработка. Такая архитектура характерна для однопроцессорных универсальных микроЭВМ, не имеющих ограничений на классы решаемых задач.

MISD (Multiple Instruction Single Data Stream) — архитектура, при которой несколько потоков команд обрабатывают один поток данных. К данному типу относится магистральная архитектура, обеспечивающая повышенную производительность для задач, алгоритмы которых допускают распараллеливание. Она предполагает наличие магистралей и нескольких процессорных модулей, которые работают одновременно и автономно, получая необходимые директивы и информацию. Магистраль служит для передачи от одной микроЭВМ к другой сообщений, содержащих адрес микроЭВМ, для которой предназначена информация, код операций и информацию, над которой данную операцию необходимо выполнить.

SIMD (Single Instruction Multiple Data Stream) — архитектура, при которой один поток команд обрабатывает несколько потоков данных. К данному типу архитектуры относятся матричные и ассоциативные системы из микроЭВМ. Матричные системы представляют собой матрицу процессоров, в которых осуществляется одновременное выполнение одной и той же команды с различными данными. Матричные системы используются при решении задач математической физики, радиолокации, распознавания образов и др.

Ассоциативные процессоры осуществляют поиск данных по содержанию, а не по адресам и представляют собой особый класс матричной архитектуры.

MIMD (Multiple Instruction Multiple Data Stream) — архитектура, при которой параллелизм в МПС достигается путем выполнения независимых частей задач над различными наборами данных в нескольких микроЭВМ и объединения результатов в процессе решения. К данному типу относится мультипроцессорная архитектура.

Отметим, что MISD-и SIMD-архитектуры являются специализированными архитектурами и позволяют эффективно решать достаточно специфичные классы задач, которые трудно, а иногда и просто невозможно решать с помощью универсальных архитектур.

Аппаратная реализация программных функций. Развитие технологии позволяет широко использовать аппаратную реализацию программных функций для повышения производительности микроЭВМ. На микропрограммном уровне вводятся микрокоманды умножения и деления и соответствующие им блоки аппаратного умножения и деления, позволяющие существенно сократить времени выполнения данных операций. На программном уровне возможна реализация арифметических операций с плавающей запятой, реализация специальных функций (преобразования Фурье, преобразования тригонометрических функций и др.), трансляторов и компиляторов с языков высокого уровня.

Организация работы памяти. Для высокопроизводительных микроЭВМ обязательным является наличие виртуальной памяти, позволяющей предоставлять в распоряжение программистов практически неограниченные пространства виртуальных адресов при достаточно быстром времени доступа к информации.

СОЗУ и ОЗУ микроЭВМ могут адресоваться непосредственно из программы. Для работы с информацией, находящейся в ВЗУ, необходимо предварительное перемещение ее из ВЗУ в ОЗУ. Организация виртуальной памяти в микроЭВМ предполагает наличие достаточно сложного программно-аппаратного механизма преобразования полного пространства виртуальных (математических) адресов в реальные физические адреса ОЗУ микроЭВМ, а также механизма автоматического перемещения требуемых массивов информации из ВЗУ в ОЗУ и наоборот. Виртуальная память микроЭВМ действует под управлением операционной системы и делает неразличимой для пользователя информацию, хранящуюся в ОЗУ или ВЗУ. Как правило, виртуальная

память реализуется с помощью страничного механизма. При страничной организации виртуальной памяти вся основная и внешняя память разбивается на части обычно емкостью по 512 или 4096 слов, называемых страницами. Внутри микроЭВМ, т. е. в ОЗУ, формируется таблица страниц, содержимое которой контролируется операционной системой. Таблица страниц состоит из признаковой части, части, идентифицирующей адрес данной страницы в ВЗУ, и части, идентифицирующей адрес данной страницы в ОЗУ. Логика работы механизма виртуальной адресации поясняется рис. 5.3. В процессе выполнения программы процессор формирует виртуальные адреса. Из формата виртуального адреса выделяется часть, соответствующая номеру виртуальной страницы. Операционная система с помощью признаковой части таблицы страниц проверяет, имеется ли данная виртуальная страница в ОЗУ. Если имеется, то операционная система формирует ее физический адрес путем объединения поля «номера виртуальной страницы» в ОЗУ, извлекаемого из таблицы страниц, и поля «адреса внутри виртуальной страницы», извлекаемого из виртуального адреса. Если требуемая виртуальная страница отсутствует в ОЗУ, то операционная система инициирует страничное прерывание, в процессе которого осуществляется ввод требуемой страницы в ОЗУ и ВЗУ, регистрация в таблице страниц осуществленного перемещения страницы и, наконец, повторное выполнение адресации. Следует отметить, что если инициировано страничное прерывание, а все ОЗУ занято страницами выполняемой программы, то операционная система должна произвести обмен страницами между ОЗУ и ВЗУ по какой-либо стратегии обмена страниц.

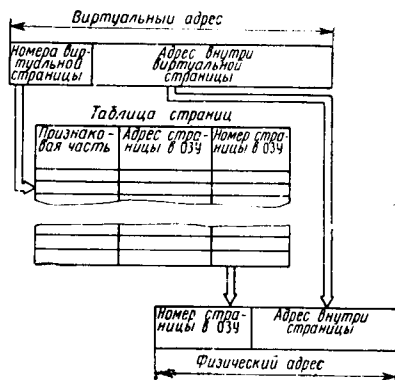


Рис. 5.3. Схема преобразования виртуального адреса

Кроме страничного механизма возможен сегментный механизм организации виртуальной памяти, при котором все виртуальное пространство адресов разбивается на несколько независимых пространств — сегментов. Сегменты могут быть различной длины, и в каждый из них записывается информация определенного типа. Любому виртуальному адресу ставится в соответствие определенный адрес сегмента и адрес внутри сегмента. Методами, аналогичными используемым в страничном механизме, формируется таблица сегментов, элементами которой являются «физический стартовый адрес», «длина сегмента», и «виртуальный стартовый адрес». По заданному виртуальному адресу в таблице с сегментов выбирается сегмент, которому принадлежит данный виртуальный адрес. Физический адрес рассчитывается путем вычитания

виртуального стартового адреса сегмента из виртуального адреса и прибавления физического стартового адреса сегмента. В случае отсутствия требуемого сегмента в ОЗУ происходит сегментное прерывание и операционная система выполняет с сегментами действия, аналогичные действиям при страничном механизме. Механизмы сегментации используются в ряде микропроцессоров, в частности, Intel 8086, KP1810, и др.

§ 5.3. ОРГАНИЗАЦИЯ ИНТЕРФЕЙСА В МИКРОЭВМ

Требования к интерфейсу. В процессе функционирования микроЭВМ осуществляется обмен информацией между центральным процессором, оперативной памятью и периферийными устройствами (дисплеями, электрическими пишущими машинками, накопителями на магнитных лентах, дисках, устройствами сопряжения с аналоговыми РЭС и др.). Все периферийные устройства требуют для функционирования набора управляющих сигналов, поступающих в определенной последовательности, протокола обмена и способа обмена с микроЭВМ, определенных кодов. Поэтому для подключения периферийных устройств к микроЭВМ используют специальные электронные схемы, называемые интерфейсными модулями, номенклатура которых зависит от числа типов периферийных устройств, физической природы устройств (электронные, электромеханические и т. д.), а также от расстояния между периферийными устройствами и микроЭВМ. Помимо аппаратной части для организации интерфейса необходимо разрабатывать программное обеспечение, включающее в себя программы идентификации типа информации, преобразования форматов, обработки запросов прерывания и т. д.

Как отмечалось, под интерфейсом понимают совокупность программных и аппаратных средств, с помощью которых все компоненты микроЭВМ или МПС объединяются таким образом, чтобы можно было решать требуемые задачи. В функции интерфейса входят дешифрация адреса устройств, синхронизация обмена информацией, согласование форматов слов, дешифрация кода команды, связанной с обращением к периферийному устройству, электрическое согласование сигналов и другие операции. Интерфейс должен осуществлять совместимость между всеми компонентами микроЭВМ по быстродействию, форме представления обмениваемой информации, архитектуре центрального процессора и электрическим характеристикам.

Согласование по быстродействию вызвано различным быстродействием микроЭВМ и периферийных устройств. Согласование производится путем контроля над состоянием признаков готовности (флажков) периферийных устройств к обмену и введения буферных регистров для временного хранения передаваемых кодов.

Согласование по форме представления обмениваемой информации производится путем использования специальных стандартных кодов,

содержащих алфавитно-цифровые, специальные и управляющие символы.

Архитектура процессора с точки зрения шинной организации (организации магистралей управления, данных и адреса) и набора управляющих сигналов определяет метод синхронизации процессора и периферийных устройств.

Электрические характеристики процессора должны быть совместимы с характеристиками логических схем интерфейса, которые, в свою очередь, должны быть согласованы с электрическими характеристиками периферийных устройств.

Возможна различная организация обмена информацией между процессором, памятью и периферийными устройствами. На организацию обмена влияют состав и назначение шин, правила их использования, способы функционального и физического разделения при передаче данных, адресов и управляющих сигналов.

Оценки эффективности интерфейсов. Оценка эффективности интерфейсов является сложной многофакторной задачей.

Структура интерфейса определяется размерностью и схемой соединений, а также топологией связей. *Размерность соединений* — число независимых информационных каналов и степень совмещения во времени процессов информационного взаимодействия в интерфейсе. Существующие стандартные интерфейсы микроЭВМ имеют в основном одномерную структуру. *Схема соединения* определяет структуру информационного и управляющих каналов и сводится к параллельному, последовательному и параллельно-последовательному вариантам. *Топология связей* характеризует взаимное размещение блоков микроЭВМ и интерфейсных блоков в пространстве. Она зависит от конструктивного размещения плат интерфейсных блоков и разделяется на внутриблочную и межблочную.

Функциональная организация интерфейса характеризует способы построения и порядок взаимодействия интерфейсных блоков.

Набор структурных параметров (размерности, схемы соединения, топологии связей) и интерфейсных операторов (функционально независимых интерфейсных блоков, выполняющих функции селекции, обмена, координации и др.) позволяет систематизировать возможные варианты интерфейсов. При этом не следует забывать, что структурные параметры характеризуют информационный канал, а интерфейсные операторы — управляющий.

Среди технико-экономических показателей можно выделить пропускную способность, вместимость, надежность и стоимость.

Пропускная способность определяет время передачи единицы информации между устройствами интерфейса. Она зависит от времени, необходимого для установления связи, передачи единицы информации и разъединения связи, а также от длины массива данных, передаваемого за один сеанс связи. Очевидно, пропускная способность зависит также от схемы соединения, топологии шин, способа реализации операторов селекции и синхронизации.

Вместимость характеризуется максимальным числом устройств, которое может быть подключено к интерфейсу без использования дополнительных средств его расширения. Она зависит от системы адресации интерфейса, нагрузочной способности приемопередающих элементов и условий конструктивной реализации. Вместимость тесно связана с остальными характеристиками интерфейсов — ее увеличение снижает значения пропускной способности и надежности и повышает стоимость интерфейса.

Надежность интерфейса обычно определяется средним временем наработки на отказ.

При оценке *стоимости* интерфейса обычно включают затраты на интерфейсные блоки, систему электропитания, конструктивы и кабели связи.

При оценке эффективности интерфейса можно использовать обобщенные критерии, одним из которых является добротность интерфейса:

$$Q = \frac{(I + \log_2 A + \log_2 K + F) d}{(T_1 + T_2/M) h},$$

где I — разрядность информационной магистрали; A — используемое число адресов; K — используемое число команд; F — эквивалентное число команд, которое надо выполнить при реализации эквивалентных аппаратных функций устройства сопряжения; d — коэффициент, учитывающий ускорение обмена за счет введения режима прямого доступа в память; T_1 — время передачи одного слова; T_2 — дополнительное время обмена, определяемое интервалами подготовки и завершения операции обмена и идентификации запроса; M — число слов; h — общее число шин интерфейса.

На практике довольно часто используют критерий отношения производительности к стоимости, где производительность определяется средней пропускной способностью интерфейса. Однако оценка альтернативных вариантов интерфейсов на основе обобщенных критериев иногда встречает ряд трудностей принципиального характера. Это связано с тем, что некоторые характеристики интерфейса невозможно связать аналитическими соотношениями с основными параметрами, интерфейса, а ряд характеристик (технологичность изготовления, уровень стандартизации и др.) вообще не поддается количественной оценке. Поэтому в последнее время при оценке эффективности интерфейсов получил распространение метод экспертных оценок.

Из всего многообразия типов микроЭВМ можно выделить две основные организации интерфейса между процессором, памятью и периферийными устройствами: двухшинную, или интерфейс с изолированной системой шин, и одношинную, или интерфейс с общей шиной.

Интерфейс с изолированной шиной. Особенностью этого интерфейса является раздельная адресация памяти и периферийных устройств при обмене информацией путем использования отдельных групп команд для обмена информацией с периферийными устройствами и па-

мью. Проиллюстрируем данный тип интерфейса на примере его применения в микроЭВМ, построенных на основе МП КР580 (рис. 5.4). Обмен с периферийными устройствами производится с помощью двухбайтных команд ввода IN PORT и вывода OUT PORT, а управление обменом — под воздействием управляющих сигналов ввода I/OR и вывода I/OW, формируемых системным контроллером при выполнении соответствующих команд. Адрес из МП передается по МА, передача данных выполняется по МД между аккумулятором А и буферным регистром интерфейсного модуля (портом) периферийного устройства.

Обмен информацией между МП и памятью производится по командам передачи данных. Так, команда MOV г, М осуществляет передачу из ячейки памяти, адресуемой 16-битной регистровой парой (Н, L), в аккумулятор или любой регистр общего назначения, кодируемый в поле г. Команда MOV, М, г выполняет передачу из регистра МП в ячейку памяти.

Рассмотрим пример ввода кода из периферийного устройства I в регистр В:

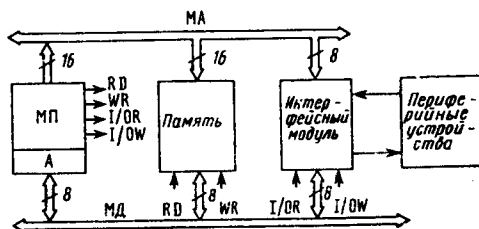


Рис. 5.4. Интерфейс с изолированной шиной

Метка	Код	Операнд	Комментарий
	MOV	С, А ;	Временное запоминание содержимого аккумулятора в регистре В
	IN	001 Q ;	Ввод кода из периферийного устройства в аккумулятор
	MOV	В, А ;	Передача содержимого аккумулятора в регистр В
	MOV	А, С ;	Восстановление аккумулятора

Основным недостатком организации интерфейса с изолированной шиной является осуществление обмена обязательно через аккумулятор МП, что, как правило, требует выполнения четырех команд.

Интерфейс с общей шиной. При данной организации интерфейса все адресное пространство делится между периферийными устройствами и ячейками памяти, причем обращение к периферийным устройствам осуществляется посредством набора команд, используемых для передачи данных с памятью (рис. 5.5). Так, например, при построении интерфейса с общей шиной в микроЭВМ на основе МП КР580 обычно выделяется старший разряд A_{15} МА в качестве обращения к памяти ($0 \dots 32 K - 1$, $A_{15} = 0$) или периферийным устройством ($32 \dots 64 K$, $A_{15} = 1$).

В качестве управляющих сигналов для обращения к памяти используются сигналы чтения RD и записи WR, а к периферийным устройствам — сигналы I/OR и I/OW.

Достоинствами данного интерфейса являются:

возможность расширения набора команд для обращения к периферийным устройствам, что позволяет сократить объем программ и повысить быстродействие;

возможность обмена информацией между любыми регистрами МП;

возможность внепроцессорного обмена данными между ячейками памяти.

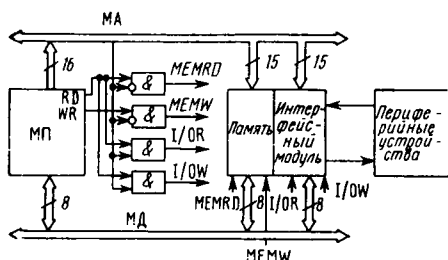


Рис. 5.5. Интерфейс с общей шиной

Недостатком интерфейса с общей шиной является усложнение дешифрирующих схем и сокращение области памяти. Интерфейс с общей шиной является наиболее распространенным в мини- и микроЭВМ.

Стандартные интерфейсы.

Динамика развития интерфейсов мини- и микроЭВМ направлена в сторону увеличения номенклатуры и числа периферийных устройств, увеличения числа операций ввода — вывода, осуществления

возможности функционирования в нескольких режимах взаимодействия (ввод — вывод по программному каналу, каналу прямого доступа в память и др.). Особенностью интерфейсов микроЭВМ является влияние жестких конструктивных ограничений, связанных с необходимостью минимизации внешних выводов БИС и низкой мощностью выходных сигналов. В структуре связей микроЭВМ разделяют *внутренний интерфейс*, объединяющий БИС процессора, модули ОЗУ, ПЗУ и схемы управления вводом — выводом, и *внешний интерфейс*, обеспечивающий сопряжение между внутренней магистралью и периферийными устройствами. Наиболее популярными и перспективными являются интерфейсы с минимальным числом шин на основе использования принципа мультиплексирования.

Системные интерфейсы можно классифицировать по времени взаимодействия Т, определяемому как время доступа процессора к периферийному устройству для пересылки единицы информации, и максимально возможной длине магистрали L (табл. 5.1).

Наиболее распространенными интерфейсами мини- и микроЭВМ являются интерфейсы с единой магистралью (общей шиной), представляющие собой совокупность совместимых и частично совместимых магистралей. Основу всех модификаций составила система Unibus, аналогом которой является интерфейс «Общая шина» СМ ЭВМ.

Основные характеристики интерфейсов ЭВМ с единой магистралью приведены в табл. 5.2.

Классификация системных интерфейсов

Группа системных интерфейсов	Т. мкс	Л. м	Интерфейс
Внутриплатные	1	0,3	Microbus; модификация Q-bus, МПИ
Сосредоточенные	1	0,8	Q-bus персональных ЭВМ. магистраль Nova, Multibus 16-разрядных ЭВМ, И41 СМ
Локально-сосредоточенные	5	3	Q-bus; И41 СМ; МПИ
Локальные	10	15	Unibus PDP-11, «Общая шина» СМ ЭВМ
Локально-распределенные	15 и более	20 и более	Последовательная магистраль Micro Nova

Таблица 5.2

Характеристики интерфейсов ЭВМ с единой магистралью

Параметр интерфейса	Мини-ЭВМ	МикроЭВМ	Персональные ЭВМ
Число функциональных линий Ст/Доп	56/60	34/46	46
Разрядность данных, бит	16	16	16
Разрядность шин адреса, Ст/Доп	18/22	18/22, М	22, М
Число линий управления	22	16	24
Число уровней приоритетов, Ст/Доп	5	2/5	3
Число подключаемых устройств	18	15	8
Длина магистрали, м	20	1	0,5
Минимальное время цикла, мкс	0,4	0,5	0,5

Примечание. Доп — допустимое; М — мультиплексируемые; Ст — стандартное.

В системных интерфейсах микроЭВМ используется мультиплексирование шин адреса и данных в целях реализации магистрали в БИС с корпусом, имеющим 40 выводов. В магистрали массовых микроЭВМ 16 двунаправленных линий мультиплексируются во времени относительно шин адресов и данных. Передачи по ним упорядочиваются с помощью управляющих линий, обеспечивающих идентификацию и направление обмена по информационным шинам, различение передачи слова и байта данных, синхронизацию обмена.

Следует отметить, что интерфейс Microbus разработан для 8-разрядных МП и применяется в качестве внутриплатной магистрали; интерфейс Q-bus рассчитан на упрощенную магистраль для внутриплатных и расширенную для межплатных и межблочных соединений.

§ 5.4. СРЕДСТВА СОПРЯЖЕНИЯ МИКРОЭВМ С ОБЪЕКТАМИ

Компоненты сопряжения микроЭВМ с объектами. Функционирование сложных объектов управления и контроля различной физической природы (технологических процессов, движущихся объектов, разнообразных РЭС и т. д.) обусловили необходимость высокой степени автоматизации процессов обработки информации для проведения процедур управления и контроля с помощью МПУ или микроЭВМ. Сопряжение микроЭВМ и МПУ с объектами управления и контроля осуществляется автоматизированными системами сбора, предварительной обработки (для ввода информации) и регистрации (для вывода информации). В состав аппаратуры сопряжения входят датчики, различные контроллеры, преобразователи аналоговых сигналов в цифровые коды (АЦП), цифровые и аналоговые коммутаторы и каналы связи, преобразователи цифровых сигналов в аналоговые (ЦАП), регистрирующие устройства.

Контроллер — устройство, выполняющее функции управления передачей данных и освобождающее от этих функций процессор. В различных управляющих комплексах могут использоваться самые разнообразные контроллеры. Так, например, в состав аппаратуры сопряжения управляющей системы «Электроника НЦ-31», предназначенной для оперативного управления универсальным токарным станком, входят контроллеры электроавтоматики, привода и фотопульсных преобразователей. В состав аппаратуры сопряжения управляющей системы «Электроника МС 2101», предназначенной для управления различными типами металлообрабатывающего оборудования, помимо названных контроллеров входит АЦП для приема информации с аналоговых датчиков управляемого оборудования.

В микропроцессорных РЭС одной из основных функций системы сопряжения являются преобразования информации из аналоговой формы в цифровую и наоборот. Поэтому одними из основных компонентов аппаратуры сопряжения в микропроцессорных РЭС являются цифроаналоговые и аналого-цифровые преобразователи. *Цифроаналоговый преобразователь* — устройство для автоматического декодирования входных величин, представленных числовыми кодами, в эквивалентные им значения заданной физической величины. Количественная связь между входной числовой величиной и ее аналоговым эквивалентом $A(t_i)$, характеризующая алгоритм цифроаналогового преобразования, имеет вид $A(t_i) = N_i/\Delta A + \delta A_i$, где ΔA — аналоговый эквивалент единицы младшего разряда кода; δA_i — погрешность преобразования.

Аналого-цифровой преобразователь — устройство для автоматического преобразования непрерывно изменяющихся во времени аналоговых величин в эквивалентные значения числовых кодов. Количественная связь между аналоговой величиной $A(t_i)$ и соответствующей ей цифровой величиной N_i , характеризующая алгоритм аналого-цифрового преобразования, имеет вид $N_i = A(t_i)/\Delta A \pm \delta N_i$, где ΔA —

шаг квантования, т.е. аналоговый эквивалент единицы младшего разряда кода; δN_i — погрешность преобразования на данном шаге.

В любом из преобразователей можно выделить цифровую и аналоговую части. В цифровой части производится кодирование или декодирование, запоминание, сравнение, сдвиг, счет и сложение цифровой информации, прием и обработка команд на преобразование, выработка сигналов управления, а также выполнение других логических функций, причем все чаще в цифровой части преобразователей используются микропроцессоры. В аналоговой части осуществляются сравнение, усиление, выборка и хранение, коммутация аналогового сигнала, а также операции по его сложению и вычитанию, делению, интегриро-

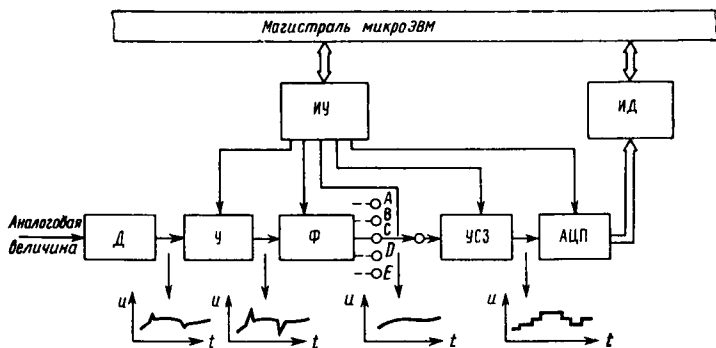


Рис. 5.6. Схема подсистемы аналогового ввода

ванию и выделению в промежуточную величину. Элементы, используемые при построении преобразователей, также подразделяются на цифровые (логические схемы, регистры сдвига, счетчики и т. д.) и аналоговые (операционные усилители, ключи, коммутаторы, резистивные матрицы и т.д.). Электрические и эксплуатационные характеристики АЦП и ЦАП во многом зависят от характеристик содержащихся в них элементов. В микропроцессорных РЭС применяются преобразователи, реализованные в микросхемотехническом исполнении.

Подсистема аналогового ввода. Процесс получения аналогового сигнала с какого-либо датчика, его предварительная обработка, преобразование в цифровую форму и ввод в МПС называется сбором данных. На рис. 5.6 представлена общая схема подсистемы сбора аналоговых данных. Контролируемый параметр с помощью соответствующего датчика Д преобразуется в электрический сигнал и усиливается усилителем У до необходимого уровня. С помощью активного фильтра Ф из сигнала удаляются помехи и отфильтрованный сигнал через аналоговый мультиплексор поступает на устройство слежения — запоминающий элемент УСЗ. Аналого-цифровой преобразователь преобразует уровень напряжения на входе в соответствующую цифровую величину. Да-

лее цифровые данные через интерфейс управления ИУ и интерфейс данных ИД поступают в общую магистраль МПС.

Аналого-цифровые преобразователи. АЦП решают задачу поиска однозначного соответствия аналоговому сигналу цифрового кода. На вход АЦП поступает аналоговый сигнал и после определенного конечного времени преобразования на его выходе появляется цифровой код. В настоящее время разработано много методов аналого-цифрового преобразования, в частности

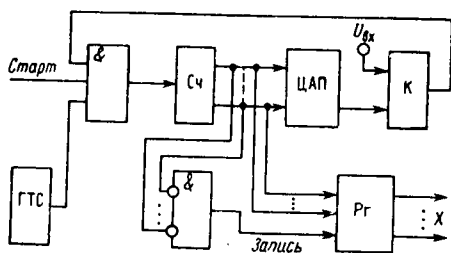


Рис. 5.7. Структура АЦП последовательного счета

методы последовательного счета, поразрядного кодирования, двойного интегрирования, преобразования напряжения в частоту, параллельного преобразования и др. В микропроцессорных РЭС наибольшее распространение получили АЦП «напряжение (ток) — код». Рассмотрим схемы построения АЦП по методам последовательного счета, поразрядного кодирования и двойного интегрирования.

При построении АЦП по методу последовательного счета входная аналоговая величина $U_{вх}$ уравнивается суммой одинаковых и минимальных эталонов, называемых квантами. Момент уравнивания определяется компаратором К (рис 5.7). Результат преобразования характеризуется числом квантов, используемых при преобразовании. Это число квантов представляется в виде последовательного единичного кода, преобразуемого с помощью счетчика Сч в позиционный двоичный код X. Время преобразования T_n — переменное и зависит от входного сигнала (рис. 5.8), однако такт работы всего устройства постоянен: $T_p = T_0 2^n$, где T_0 — период генератора тактовых сигналов ГТС; n — разрядность АЦП.

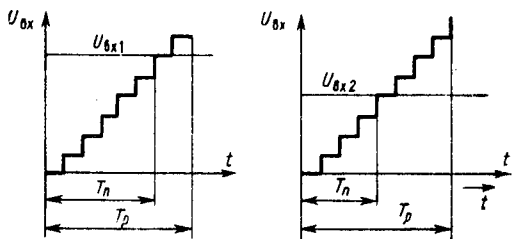


Рис. 5.8. Временные диаграммы работы АЦП последовательного счета

В АЦП, построенных по методу поразрядного кодирования (рис. 5.9), входная величина $U_{вх}$ последовательно сравнивается с суммой эталонов, имеющих значение 2^i квантов, $i = 0, n - 1$. При поступлении сигнала «Старт» ГТС выдает первый единичный сигнал на регистр сдвига Пр Сдв. Уравнивание на компараторе К начинается с эталона, дающего максимальное значение выходного напряжения ЦАП $U_{вых1} = 2^{n-1} \Delta A$. При $U_{вх} \geq U_{вых1}$ с регистра Пр Сдв через схему И

в старший разряд регистра выхода Рг А записывается «1», при $U_{вх} < U_{вых1} - «0»$. В следующем такте работы ГТС производится сдвиг единичного сигнала в Рг Сдв и осуществляется уравнивание сигнала $U_{вх}$ с эталоном, дающим на выходе ЦАП напряжение $U_{вых2} = 2^{n-2} \Delta A$, если в старшем разряде РгА записан «0», и величину $U_{вх} - U_{вых1}$, если в старшем разряде записана «1». Процесс преобразования проводится итеративно с последовательно убывающими эталонами в течение n тактов работы ГТС, т. е. время цикла работы АЦП $T_p = nT_0$. Следует отметить, что частота $1/T_p$ является параметром, определяющим максимально допустимую частоту отслеживания входного сигнала $U_{вх}$.

В АЦП, построенных по методу двойного интегрирования, используется предварительное преобразование напряжения во временной интервал, а затем измерение временного интервала. Преобразователь «напряжение—временной интервал» строится на основе интегратора, так как при постоянном входном напряжении входное напряжение интегратора представляет собой линейную функцию времени:

$$U_{вых}(t) = -[1/RC] \int_0^t U_{вх}(t) dt = -[1/RC] U_{вх} t.$$

На рис. 5.10, а, б приведены схема и график работы АЦП двойного интегрирования. На первом этапе преобразования на вход интегратора, построенного на основе операционного усилителя ОУ, через входной коммутатор КВх подается напряжение $U_{вх}$ и в течение фиксированного времени T_1 реализуется первое интегрирование, в результате которого на выходе интегратора устанавливается напряжение, пропорциональное $U_{вх}$, т. е. $U_{вых} = -U_{вх} T_1 / RC$.

На втором этапе преобразования на вход интегратора подключается опорное напряжение E_0 , знак которого противоположен знаку $U_{вх}$,

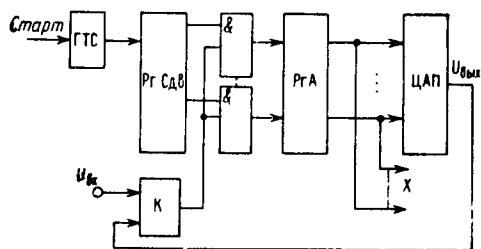
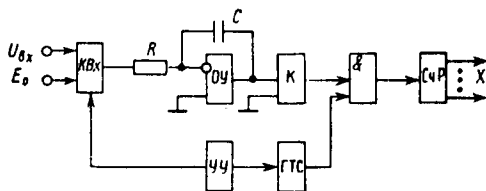
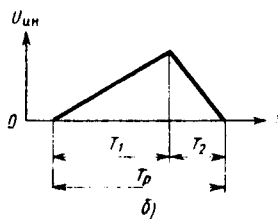


Рис. 5.9. Структура АЦП поразрядного кодирования



а)



б)

Рис. 5.10. Структура АЦП двойного интегрирования (а) и временная диаграмма его работы (б)

напряжение на выходе интегратора изменяется от $U_{\text{вых max}}$ до нуля в течение времени $T_2 = U_{\text{вх}} T_1 / E_0$. При достижении на выходе интегратора нулевого уровня «нуль-компаратор» К прекращает поступление тактовых импульсов на вход счетчика результат СчР.

Цифроаналоговый преобразователь. ЦАП — функциональный узел, однозначно преобразующий кодовые комбинации цифрового сигнала в значения аналогового сигнала. Основой для нахождения однозначного соответствия может служить выражение

$$U_{\text{вых}} = E_0 (x_1 2^{-1} + x_2 2^{-2} + \dots + x_n 2^{-n}), \quad (5.3)$$

где $U_{\text{вых}}$ — напряжение на выходе ЦАП; E_0 — опорное напряжение; $X = (x_1, x_2, \dots, x_n)$ — цифровой код, $x \in \{0, 1\}$.

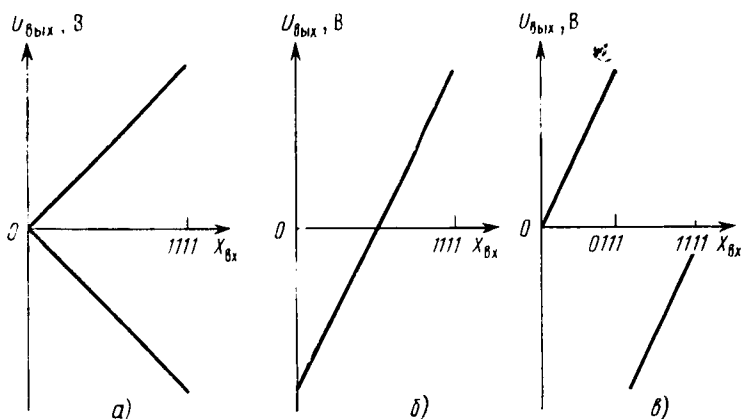


Рис. 5.11. Графики соответствия цифровых кодов и аналоговых напряжений

При определенном E_0 каждому X_i на выходе ЦАП соответствует напряжение $U_{\text{вых}i}$. В ЦАП используют три основных двоичных кода: прямой, смещенный и дополнительный. Графики соответствия цифровых кодов $X_{\text{вх}}$ и аналоговых напряжений $U_{\text{вых}}$ при прямом (а), смещенном (б) и дополнительном (в) кодах показаны на рис. 5.11.

Прямой код удобен при преобразовании сигналов следящих систем, так как при переходе через нуль меняются старшие разряды кода, что позволяет реализовывать линейный переход от малых положительных к малым отрицательным выходным напряжениям. При прямом коде в процессе преобразования используется знаковый разряд, который управляет переключением выходного напряжения ЦАП. Для исключения коммутирующих элементов из схемы ЦАП используют *смещенный код*, являющийся наиболее простым для реализации в схеме преобразователя. В случае применения *дополнительного кода* положительные числа преобразуются так же, как для прямого кода, а отрицательные — двоичным дополнением соответствующего положительного

числа (инверсия всех разрядов с последующим добавлением единицы в младший разряд). Из рис. 5.11 видно, что прямой код дает возможность использовать в 2 раза большее разрешение по сравнению со смещенным и дополнительным кодами (за счет дополнительного знакового разряда).

Базовая схема ЦАП, реализующая выражение (5.3), содержит источник опорного напряжения E_0 , матрицу двоично-весовых резисторов, набор ключей $K_1, \dots, K_n$ и дифференциальный операционный усилитель ОУ (рис. 5.12). Весовые коэффициенты x_i уравнения (5.3) реализуются сопротивлениями; если $x_i = 1$, то соответствующий ключ подсоединяет резистор 2^{i-1} ко входу ОУ, если $x_i = 0$, то к земле. Основные недостатки этой схемы определяются большим диапазоном номиналов резисторов и трудностями их подбора с высокой точностью, что особенно существенно при интегральном исполнении ЦАП.

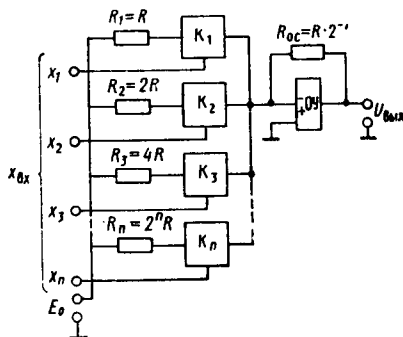


Рис. 5.12. Схема ЦАП с двоично-взвешенными сопротивлениями

Рациональным способом уменьшения количества номиналов резисторов является использование *резистивной (лестничной) матрицы* $R - 2R$, изображенной на рис. 5.13. Выражение (5.3) этой схемы ЦАП реализуется непосредственно, так как соотношение R_{oc}/R_i равно весу

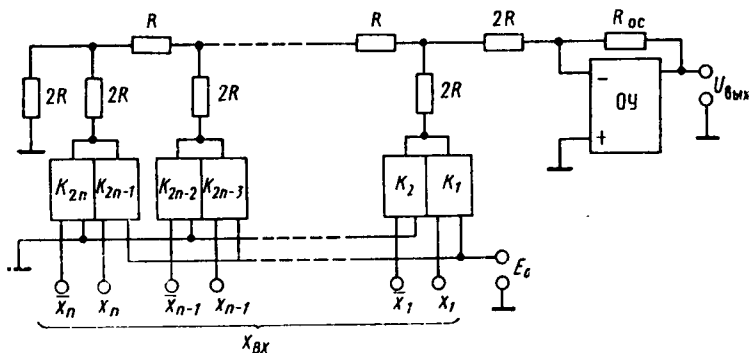


Рис. 5.13. Схема ЦАП на основе резистивной матрицы

соответствующего x_i , где R_i — сопротивление матрицы при условии равенства единице только одного разряда x_i цифрового кода.

Аналоговые микропроцессоры. Решение задачи рационального построения микропроцессорных РЭС может базироваться на использовании аналоговых микропроцессоров (АМП), структура которых позволяет принимать аналоговые сигналы, преобразовывать их в цифровую

форму для целей цифровой обработки, а затем результаты обработки формировать в аналоговые выходные сигналы. Системы обработки данных, построенные на базе АМП, обладают высокими эксплуатационными характеристиками за счет отсутствия промежуточных АЦП и ЦАП, выполненных в отдельных конструктивах, и цифровой обработки сигналов, что повышает надежность системы, увеличивает точность и быстродействие обработки информации, существенно упрощает разработку и монтаж системы, а также ее отладку. На рис. 5.14 представлена структура программируемого АМП 2920, разработанного спе-

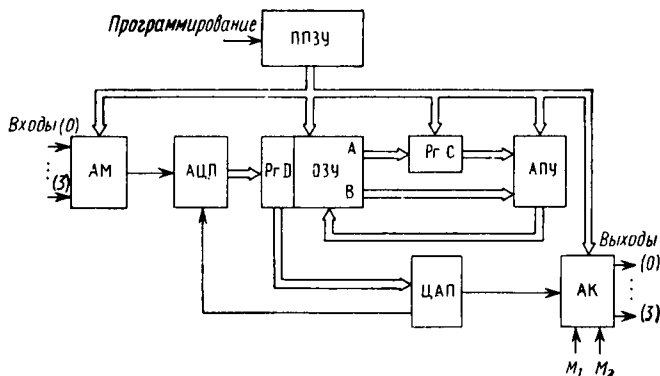


Рис. 5.14. Структура аналогового микропроцессора 2920

циалистами фирмы Intel. В состав АМП входят: программируемое постоянное запоминающее устройство ППЗУ; ОЗУ, имеющее два выхода А и В; многофункциональный регистр РгД, реализующий функции регистра АЦП последовательного счета и регистра ЦАП; обращение к регистру РгД для записи или считывания информации идентично обращению к ОЗУ, т. е. адрес РгД содержится в соответствующих командах; сдвиговый регистр РгС; арифметико-логическое устройство АЛУ, реализующее арифметические и логические операции с фиксированной точкой; аналоговый мультиплексор АМ входных аналоговых сигналов; аналоговый коммутатор АК выходов АМП. В АМП предусмотрена возможность вывода последовательных цифровых сигналов, управление выводом (аналоговым или цифровым) реализуется в зависимости от состояний входов M_1 и M_2 .

Команды АМП 24-разрядные, двухадресные, содержат два адреса: А и В. Формат команды АМП 2920 приведен на рис. 5.15. Содержимое команды условно разделено на пять зон: зону 4 — код операции АЛУ (три разряда), зону 3 — управление регистром сдвига РгС (четыре разряда), зону 2 адресов А и В (12 разрядов), зону 1 — код аналоговых операций АДЕ (два разряда), зону 0 — код аналоговых операций АДК (три разряда). Состав операций АЛУ содержит операции сложе-

ния, вычитания, сдвига, сложения по модулю 2, умножения, определения абсолютного значения, определения знака и др. Операции управления регистром сдвига РгС включают в себя операции сдвига влево на один или два разряда и операции сдвига вправо на разряды от 0 до 13. Аналоговые операции позволяют осуществить ввод аналоговой информации с одного из четырех входов АМ, вывод аналогового сигнала на один из четырех выходов АК, формирование знакового разряда при АЦП, формирование разрядов ЦАП и ряд других операций. Конструктивно АМП размещен в одном 28-выводном корпусе.

Способы сопряжения АЦП и ЦАП с МП. АЦП и ЦАП развивались независимо от микропроцессоров, поэтому часто возникают трудности их сопряжения. Работа с МП потребовала логической и электрической

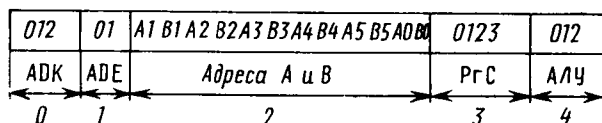


Рис. 5.15. Формат команды АМП 2920

совместимости АЦП и МП, а также использования МП для выполнения логических функций АЦП, которые обычно реализовывались его цифровой частью.

Способ сопряжения АЦП с МП обычно зависит не от типа АЦП, а от способа передачи данных — последовательным или параллельным кодом — и от возможностей, которые предусматриваются схемами используемых АЦП и МП. Выбор последовательной или параллельной передачи данных чаще всего определяется расстоянием от АЦП до МП.

Для обеспечения логической совместимости АЦП и МП должны работать в одинаковых или близких кодах и иметь одинаковые логические уровни «0» и «1». Электрическая совместимость предусматривает одинаковые уровни сигналов в цепях управления и на шине данных. Кроме того, выходные сигналы от АЦП должны поступать на шину данных МПС, как правило, с выходов трехстабильных буферов (третье состояние характеризуется высоким значением выходного сопротивления), чтобы не загружать шину данных в нерабочем состоянии.

При взаимодействии преобразователя и МП используются три вида сигналов: адреса, управления и данных, которые требуются для выбора канала преобразования, инициирования и управления процессом преобразования, передачи данных. Поэтому важное значение имеет временное согласование диаграмм работы МП и преобразователя. Необходимым условием совместной работы преобразователей и МП является их совместимость по виду цифрового кода и формату данных. Цифровые коды в преобразователях и МП в общем случае могут не совпадать. Тогда для сопряжения нужно применять внешние кодовые

преобразователи или встроенные логические элементы трансформации кодов.

Соответствие числа разрядов АЦП или ЦАП и МП является желательным, но не обязательным условием для сопряжения. Если длина слова преобразователя превышает длину слова шины данных МП, то при обмене данными потребуются дополнительные операции по формированию из нескольких слов МП одного полного слова преобразователя и сложное с точки зрения реализации сопряжение последовательного формата данных. Если большей длиной слова обладает шина данных МП, то его избыточные старшие разряды не учитываются и возможно применение параллельного формата данных.

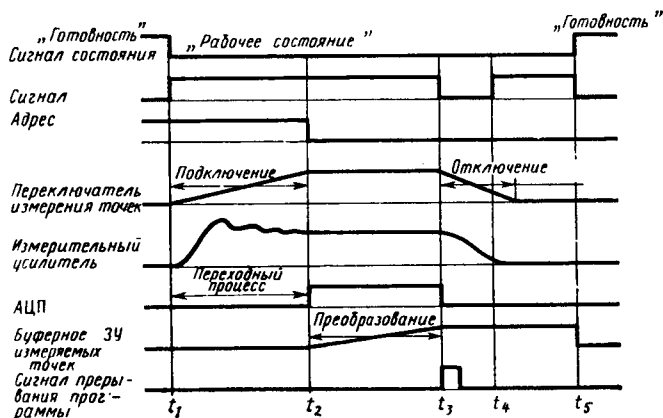


Рис. 5.16. Временной процесс ввода аналоговых данных

При организации совместной работы АЦП с микропроцессорами необходимо осуществлять программное сопряжение, заключающееся в проверке сигнала преобразования и в организации прерываний.

Проверка сигнала преобразования заключается в том, что МП после выдачи на АЦП команды начала преобразования «входит в цикл», т. е. ждет сигнала АЦП об окончании преобразования, после которого «выходит из цикла», считывает данные с АЦП и в соответствии с ними приступает либо к следующему преобразованию, либо к выполнению основной программы. Этот способ почти не нуждается в дополнительной аппаратуре, но применяется только в МПС с малой загрузкой, поскольку полезная работа МП прекращается на время преобразования.

Однако для большинства типов сопряжений АЦП с МП требуется возможность работы МП с прерыванием.

При простом прерывании МП, выдав команду о начале преобразования, продолжает работу по основной программе. После окончания преобразования специальное устройство формирует сигнал прерывания, который прерывает вычисления в МП и включает процедуру по-

иска периферийного прибора, пославшего сигнал прерывания. Эта процедура состоит в переборе всех периферийных устройств, пока не будет найден нужный. Преимущество данного способа по сравнению с предыдущим проявляется в большем числе преобразований за одно и то же время при использовании медленнодействующих АЦП.

При организации прямого доступа в память управление по сигналу прерывания передается на специальный интерфейс, который и производит перезапись данных преобразования в память, минуя регистр МП. Это позволяет сократить длительность прерывания до одного такта.

При сопряжении АЦП с МП важно обеспечить соблюдение всех временных соотношений, чтобы данные преобразователя устанавливались на магистрали данных МП за время, равное времени обращения к системе памяти, с которой работает МП. Обобщенный временной процесс ввода аналоговых данных из опрашиваемого канала представлен на рис. 5.16.

В момент времени t_1 в блок управления аналоговым вводом помещаются адрес измерительной точки и установленный коэффициент усиления. Функциональный блок сообщает о «готовности» с помощью разрешающего сигнала. Формируется команда запуска. Переключатель измеряемых точек циклически либо путем программного управления переключает адресуемые измеряемые точки, производится переключение усилителя.

После выполнения в переключателе измеряемых точек процессов включения заканчиваются время переключения и переходный процесс усилителя (момент времени t_2). Сигнал готовности переключателя измеряемых точек включает АЦП.

В момент времени t_3 процесс преобразования завершается. АЦП выдает сообщение «готовности» для передачи данных в МПС, откуда в момент времени t_4 поступает пусковой сигнал для приема данных от АЦП. Процесс заканчивается в момент времени t_5 , после чего аналоговый ввод оказывается готовым для нового процесса ввода.

Рассмотрим пример организации совместной работы АЦП с 10-разрядным выходом и МП КР580. На рис. 5.17 представлена организация связи между АЦП и МП по принципу интерфейса с изолированной шиной.

АЦП имеет аналоговый вход, цифровой выход, вход запуска преобразования информации, после завершения которого формируется

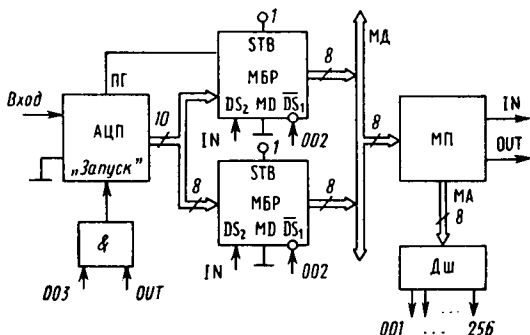


Рис. 5.17. Интерфейс микропроцессора с АЦП

признак готовности (ПГ). Так как 10-разрядный код не может быть введен в параллельной форме в 8-разрядный МП, то ввод осуществляется последовательно-параллельно из многоцелевых буферных регистров (МБР) типа К589ИР12, имеющих трехстабильный выход. Сигнал «Запуск» на входе АЦП инициирует преобразование, он может быть сформирован при выполнении команды вывода по адресу, присвоенному данному входу (в рассматриваемом примере 003).

ПГ принимает значение «1», когда преобразование завершено, и может анализироваться МП или использоваться в качестве сигнала прерывания.

Подпрограмма в системе команд МП КР580 имеет вид:

Метка	Код	Операнд	Комментарий
	PUSH	PSW	; Временное запоминание содержимого аккумулятора и флажков в стеке
	OUT	003 Q	; Запуск преобразования
TEST	IN	002 Q	; Ввод ПГ и двух старших разрядов кода
	ADI	200 Q	; Формирование ПГ на триггере
	INC	TEST	; Если ПГ=0, переход к TEST
	MOV	B, A	; Передача 2-разрядного кода в регистр В
	IN	001 Q	; Ввод в аккумулятор восьми младших разрядов кода
	MOV	C, A	; Передача 8-разрядного кода в регистр С
	POP	PSW	; Восстановление аккумулятора и флажков
	RET		Возврат из подпрограммы

Таким образом, 10-разрядный код с АЦП после выполнения данной программы передается в регистровую пару В, С. Как видно из рис. 5.17, для подключения АЦП к МП требуются три кода адреса.

Подсистема аналогового вывода. Аналоговый вывод представляет собой процесс, обратный аналоговому вводу. В микропроцессорных РЭС подсистема аналогового вывода предназначена для выдачи в различные устройства сигналов в виде напряжений или токов, изменяющихся во времени по заданному закону, а также может быть использована для вывода из МПС результатов обработки информации в виде графиков и диаграмм. Для этой цели в качестве внешних устройств служат самописцы и графопостроители.

Преобразование цифровых выходных данных в аналоговый сигнал выполняется с помощью ЦАП. При сопряжении ЦАП с МП возникают определенные трудности. На практике, когда разрядность ЦАП превышает разрядность МП, цифровая информация на вход ЦАП должна

поступать в виде двух последовательных байтов — старшего и младшего. Эти два байта в ЦАП должны сформироваться в одно слово, прежде чем начнется собственно преобразование. Формирование указанного слова в запоминающем регистре, непосредственно соединенном с входами ЦАП, нежелательно, так как это может привести к большим выбросам в аналоговом сигнале на выходе. Поэтому ЦАП, совместимый с МП, обычно имеет два буферных запоминающих регистра (рис. 5.18). Наличие двух последовательно расположенных регистров позволяет фиксировать данные на шинах в короткие интервалы, в течение которых они присутствуют в МП. Первый регистр разделен на две части PrC и PrM для приема старшего и младшего байтов. В каждой части предусмотрен вход для стробирования в моменты приема данных. Вто-

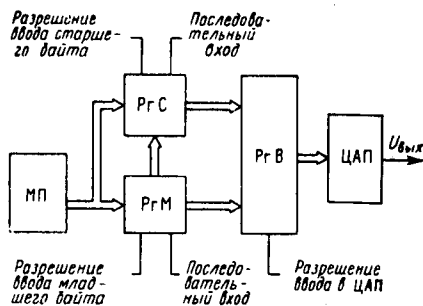


Рис. 5.18. Сопряжение ЦАП с МП с побайтным вводом данных

Таблица 5.3

Основные параметры микросхем АЦП и ЦАП

Микросхема	Назначение	Параметры			Зарубежный аналог
		n	$t_{пр. мкс}$	$U_{и.п. В}$	
K572ПВ1	АЦП поразрядного кодирования	12	130	+5,0; +15,0 -10...+10 (E_0)	AD7570
KP572ПВ2	АЦП двойного интегрирования	13...14	—	+5,0; -5,0 -5...+5 (E_0)	ICL7107
K1113ПВ1	АЦП поразрядного кодирования	10	30	+5,0; -15,0	AD571
K1107ПВ1	Одноотсчетный АЦП	6	0,1	+5,0; -6,0 -0,075...+1 (E_0) -1,9...+2,1 (E_0)	TDC1007
KP572ПА1	ЦАП с матрицей $R-2R$	10	5	+15,0; -22,5...+22,5 (E_0)	AD7520
KP572ПА2	То же	12	10	+5,0; -15,0 -22,5...22,5 (E_0)	—
KP594ПА1	ЦАП с матрицей токов	12	3,5	+5,0; -15,0 10 (9...11) (E_0)	AD562
K1108ПА1	То же	12(А) 10(Б)	0,4	+5,0; -15,0 10 (2...10,5) (E_0)	H1562
K1118ПА1	»	8	0,02	-5,2; 10 (E_0)	MC10318
K1107ПВ2	Одноотсчетный АПП	8	0,1	+5; -6 +0,1...-1,0 (E_0) -1,9...2,1 (E_0)	TDC1007J
K1107ПВ3	То же	6	0,02	-0,1...+0,1 (E_0) -1,9...-2,1 (E_0)	TDC1007J
K1108ПВ1	»	10	0,9	$\pm 5; \pm 5\%$	TDC1013J

рой регистр RгВ принимает слово, сформированное в первом регистре, и передает его в ЦАП при наличии разрешения. Обычно предусматривается также возможность использовать в качестве первого регистра сдвигаемый регистр с последовательным вводом байтов. При этом уменьшается быстродействие, но резко сокращается количество соединений между ЦАП и МП. Цифровые данные передаются от первого буферного регистра к регистру RгВ либо автоматически при записи второго байта, либо по сигналу МП, который обновляет данные по мере необходимости, либо по внешнему стробу.

Обеспечивая преобразования информации из аналоговой формы в цифровую и наоборот, микроэлектронные АЦП и ЦАП стали одними из важнейших элементов микропроцессорных РЭС. В табл. 5.3 приведены основные параметры микросхем, на основе которых реализуются аналого-цифровые и цифроаналоговые преобразователи (n — число двоичных разрядов, $t_{пр}$ — время преобразования, $U_{и.п}$ — напряжение питания).

§ 5.5. ПЕРСОНАЛЬНЫЕ ЭВМ

Термин «персональные ЭВМ» (ПЭВМ) характеризует средства вычислительной техники индивидуального пользования, удовлетворяющие двум основным требованиям: общедоступность (низкая стоимость, компактность, отсутствие специальных требований как к условиям эксплуатации, так и к степени подготовленности пользователя, малая потребляемая мощность и др.) и универсальность, обеспечивающая ориентацию этих ЭВМ на широкий круг задач, решаемых одним пользователем с помощью удобных технических и программных средств.

Можно выделить пять основных признаков, характеризующих ПЭВМ:

- 1) развитый человеко-машинный интерфейс, обеспечивающий простое и наглядное управление непрофессиональным пользователем;
- 2) большое число готовых программных средств прикладного характера для всех областей применения, избавляющее пользователя от необходимости разрабатывать программы самостоятельно;
- 3) малогабаритные накопители информации значительной емкости на сменных носителях, обеспечивающие взаимозаменяемость и эксплуатацию вновь приобретенных программных средств;
- 4) малые габариты и масса, позволяющие устанавливать ПЭВМ на любом рабочем месте (письменный стол, объект исследования и т.д.), а также малое энергопотребление;
- 5) эргономичность конструкции, привлекательность цвета и формы элементов конструкции ПЭВМ.

В состав любой ПЭВМ, как правило, входят следующие функциональные устройства:

- 1) 8-, 16- или 32-разрядный процессор;
- 2) ОЗУ емкостью 64 ... 265 К байт;
- 3) системное ПЗУ емкостью 32 ... 64 К байт;

4) контроллеры для связи с клавиатурой и с периферийными устройствами через стандартные параллельные или последовательные интерфейсы, а также контроллеры для создания локальных вычислительных сетей;

5) черно-белый или цветной растровый дисплей для вывода символьной и графической информации;

6) внешние запоминающие устройства: один или два накопителя на гибких магнитных минидисках емкостью 400 ... 800 К байт и накопители на жестких магнитных минидисках (винчестерские накопители) емкостью до 40 М байт.

К 1988 г. в мире произведено 60 млн. ПЭВМ. Среди зарубежных ПЭВМ следует выделить 8-разрядные ПЭВМ образца 1984—1986 гг.: HP-87, Правец, Commodore 64 и др., 16-разрядные ПЭВМ образца 1986—1987 гг.: IBM PC XT, IBM PC AT, Apple IIGS и др. Новейшие ПЭВМ — это семейство ПЭВМ PS/2 фирмы Intel, старшая модель 80 которого использует 32-разрядный микропроцессор 80386 на частоте до 20 МГц; ПЭВМ Toshiba 5100, реализованная также на микропроцессоре 80386; ПЭВМ MAC-2 фирмы Apple, реализованная на 32-разрядном микропроцессоре Motorola 68020 на частоте 16 МГц; ПЭВМ Macintosh, Amiga и Atari ST, построенные на 32-разрядном микропроцессоре Motorola 68000.

Среди отечественных ПЭВМ наиболее широкое распространение имеют ПЭВМ ЕС 1840, ЕС 1841, «Электроника МС 0585» диалоговые вычислительные комплексы ДВК (ПЭВМ «Электроника МС 0501», «Электроника МС 0502»), «Искра-1030», «Электроника ТЗ-29МК», «Нейрон И9.66» и ряд других.

Классификация ПЭВМ. Все многообразие ПЭВМ можно классифицировать как с точки зрения применения, так и с точки зрения принципов их построения. ПЭВМ подразделяют:

а) по назначению — домашние (бытовые), общего назначения и профессиональные;

б) по структурной организации — однопроцессорные (с 8-, 16- или 32-разрядным микропроцессором) и многопроцессорные;

в) по способу проблемной ориентации — с программной либо аппаратной ориентацией;

г) по способу использования — используемые автономно либо в вычислительных сетях;

д) по конструктивному исполнению — изготовленные в едином конструктиве либо в виде набора модулей;

е) по режиму работы — однопользовательные и мультимедийные.

Рассмотрим структуру отдельных классов ПЭВМ.

ПЭВМ общего назначения. ПЭВМ этого класса находят самое широкое применение для научных и инженерных расчетов, обучения, обработки текстов, игр, ведения домашнего хозяйства. Однако основное назначение ЭВМ — стать вычислительным и управляющим ядром автоматизированных рабочих мест различного назначения и личной оконечной станцией, позволяющей пользователю в любой момент по-

лучить доступ к информационной вычислительной сети. Проблемная ориентация ПЭВМ этого класса задается либо аппаратным, либо программным способом. Аппаратная ориентация создается соответствующим набором периферийных и других специальных технических устройств, а программная — проблемно-ориентированными пакетами программ.

Программное обеспечение ПЭВМ общего назначения является достаточно мощным и включает в свой состав несколько операционных систем, позволяющих пользователю работать в различных режимах, языки высокого уровня, пакеты прикладных программ. Все програм-

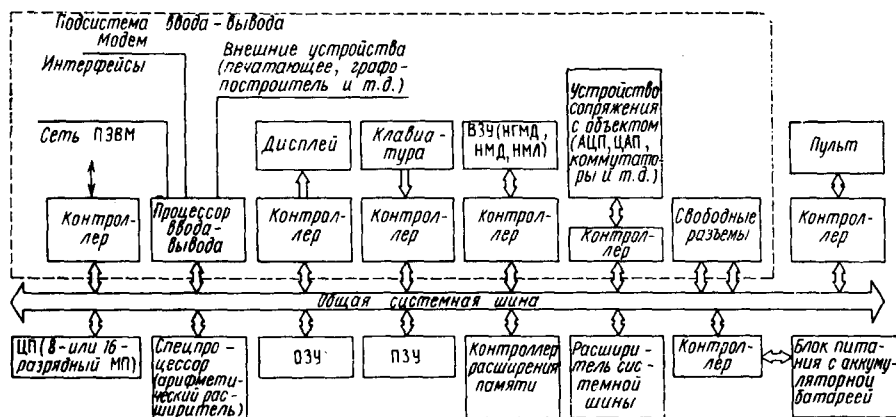


Рис. 5.19. Обобщенная структурная схема ПЭВМ общего назначения

мное обеспечение ПЭВМ общего назначения находится на носителях ВЗУ (НГМД или НМД винчестерского типа). Структуры ПЭВМ общего назначения строят в основном по магистрально-модульному принципу с общей системной шиной, что позволяет получать различные конфигурации ПЭВМ в зависимости от их конкретного применения. Обобщенная структурная схема ПЭВМ этого класса приведена на рис. 5.19.

Среди ПЭВМ общего назначения широкое распространение получили диалоговые вычислительные комплексы (ДВК). Обобщенная структурная схема ПЭВМ типа ДВК приведена на рис. 5.20. Интеллектуальным центром ДВК является одноплатная микроЭВМ, которая через систему контроллеров взаимодействует с дисплеем (монитором и клавиатурой), накопителями на гибких и жестких магнитных дисках и стандартной периферийной аппаратурой (графопостроителем, печатающим устройством). По мере совершенствования ПЭВМ типа ДВК в качестве периферийных устройств можно включать устройства ввода — вывода речевой информации, подключаемые к одноплатной микроЭВМ через соответствующий контроллер. Контроллеры пред-

ставляют собой управляющие устройства, содержащие собственный микропроцессор, ОЗУ, ПЗУ, регистры внешних устройств, буферные схемы. Контроллеры предназначены для подключения к магистрали ДВК различных по принципу действия, интерфейсу и конструктивному исполнению периферийных устройств и осуществляют следующие функции:

согласуют интерфейс магистрали ДВК со специфическим интерфейсом периферийного устройства;

выполняют роль буферной памяти между периферийным устройством и микроЭВМ ДВК;

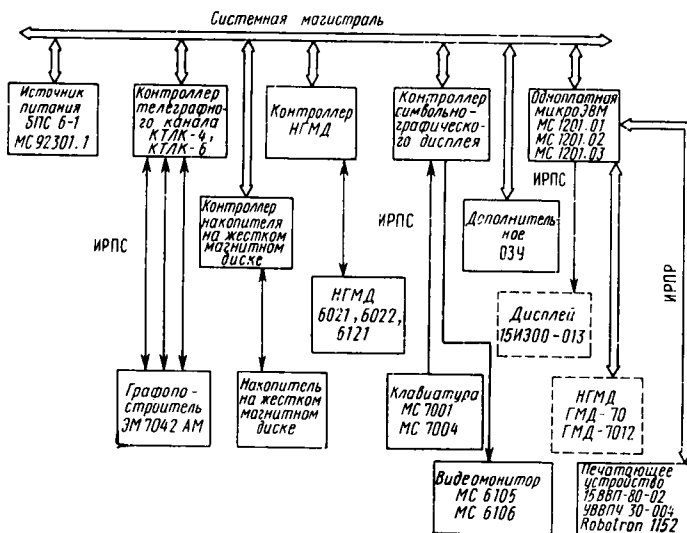


Рис. 5.20. Обобщенная структура ПЭВМ типа ДВК

освобождают от выполнения конкретных задач передачи информации центральный процессор ДВК.

Например, процессор контроллера цветного графического дисплея служит для генерации изображения на экране монитора по командам центрального процессора, а процессор контроллера минидисков выполняет все функции по блочному обмену информацией с накопителем на гибком магнитном диске (НГМД).

Роль центрального процессора в ДВК выполняет одноплатная микроЭВМ «Электроника МС 1201», различные модификации которой реализованы на однокристальных 16-разрядных микропроцессорах КР1801ВМ1 и К1801ВМ2. Характеристики различных модификаций микроЭВМ «Электроника МС 1201» приведены в табл. 5.4.

Особенности одноплатных микроЭВМ ряда «Электроника МС 1201» рассмотрим на примере микроЭВМ «Электроника МС 1201.01». Принцип работы основных устройств — параллельный, основной формат

Таблица 5.4

Характеристики микроЭВМ «Электроника МС 1201»

Техническая характеристика	Модификация микроЭВМ		
	МС 1201	МС 1201.01	МС 1201.02
Микропроцессор	КР1801ВМ1	КР1801ВМ1	К1801ВМ2
Число команд	64	64	72
Быстродействие выполнения команды СЛОЖЕНИЕ при регистровом методе адресации, тыс. оп/с	400 ± 100	400 ± 100	800 ± 100
Быстродействие выполнения команды СЛОЖЕНИЕ при косвенно-регистровом методе адресации, тыс. оп/с	180 ± 40	180 ± 40	350 ± 50
Быстродействие выполнения команды УМНОЖЕНИЕ, тыс. оп/с	—	—	80 ± 20
Микросхемы памяти	КР565РУ3	КР565РУ6	КР565РУ6

представления чисел и команд — 16 двоичных разрядов. Типы команд — безадресные, одно- и двухадресные, методы адресации — регистровая, косвенно-регистровая, автоинкрементная, косвенно-автоинкрементная, автодекрементная, косвенно-автодекрементная, индексная, косвенно-индексная. Число РОН — 8.

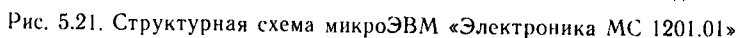
В качестве базового программного обеспечения микроЭВМ приняты тест-мониторная ОС и операционная система с разделением времени ОС ДВК.

Структурная схема микроЭВМ «Электроника МС 1201.01» приведена на рис. 5.21. В ее состав входят следующие функциональные устройства: процессор ПР, ОЗУ, системное ПЗУ (СПЗУ), устройства байтового параллельного интерфейса (УБПИ), устройства последовательного ввода — вывода (УПВВ), устройства интерфейса накопителя на гибких магнитных дисках (УИГМД), контактирующие устройства для ПЗУ пользователя (КУ ПЗУ), корректор сигналов управления системной магистралью (КС), регистр режима начального пуска (РНП), генератор тактовых импульсов (ГТИ), блок управления приемопередатчиками сигналов канала (БУПП). Связь между функциональными устройствами осуществляется через внутреннюю системную магистраль.

Связь между устройствами, подключенными к системной магистрали, осуществляется по принципу «активный — пассивный», суть которого заключается в том, что на сигнал активного устройства, инициирующий обмен данными, в активное устройство должен поступить ответный сигнал от адресуемого пассивного устройства. Поэтому про-

В микроЭВМ «Электроника МС 1201» обмен информацией с внешними устройствами осуществляется через интерфейсы периферийных устройств:

2) интерфейс локальных сетей на базе дуплексного последовательного канала с протоколом интерфейса для радиального подключения устройств с последовательной передачей информации (ИРПС), реализуемый с помощью УПВВ;



В настоящее время семейство диалоговых вычислительных комплексов является довольно распространенным. Первая модель ДВК (ДВК-1) была разработана в 1981 г. За прошедшее время создано несколько моделей ДВК, основные характеристики которых приведены в табл. 5.5.

251

Таблица 5.5

Характеристики семейства ДВК

Техническая характеристика	Модель, год разработки/год выпуска ДВК					
	«Электроника МС 0501»			«Электроника МС 0502»		
	ДВК-1	ДВК-2	ДВК-2М	ДВК-3	ДВК-3М2	ДВК-4
	1981/83	1981/83	1984	1984/85	1985/86	1986/87
МикроЭВМ	МС 1201.01			МС1201.01 МС1201.02	МС1201.02 МС1201.03	МС1201.02 МС1201.03
Быстродействие, тыс. оп/с типа RR	500			500 800	800 1200	800 1200
Число команд	64			64, 72		
Адресуемая память, К байт	64			64, 4096		
Информационная емкость ОЗУ, К байт	56			64, 248		
Информационная емкость НГМД, К байт	—	512	440	800, 440	800, 440	1600, 800
Тип дисплея	Символьный			Монохроматический графический		Цветной графический

магнитных дисках. Программное обеспечение содержит операционную систему, языки высокого уровня, специальные диалоговые языки, прикладные программные средства реализации графических и других процедур. Профессиональные ПЭВМ применяются обычно в составе комплекса технических средств САПР, локальных вычислительных сетей и т. д., но могут использоваться и автономно.

Характерной особенностью структурной организации ППЭВМ является их выполнение на базе нескольких 16- или 32-разрядных микропроцессоров с использованием кроме системной других шин (высокоскоростной, локальной) для подключения памяти и спецпроцессоров различных типов. Обобщенные структурные схемы ПЭВМ этого класса приведены на рис. 5.22 и 5.23.

Подключение периферийных устройств, ВЗУ и другого оборудования, а также связь с сетью в ПЭВМ различных классов обеспечиваются через разъемы интерфейсов и системную шину.

Особенности архитектурно-структурной организации ППЭВМ рассмотрим на примере ППЭВМ ЕС 1841.

Персональная ПЭВМ ЕС 1841 является дальнейшим развитием семейства ПЭВМ ЕС 1840 и представляет собой микрокомпьютерную систему, предназначенную для работы в автономном режиме и решения широкого круга научно-технических и экономических задач, автома-

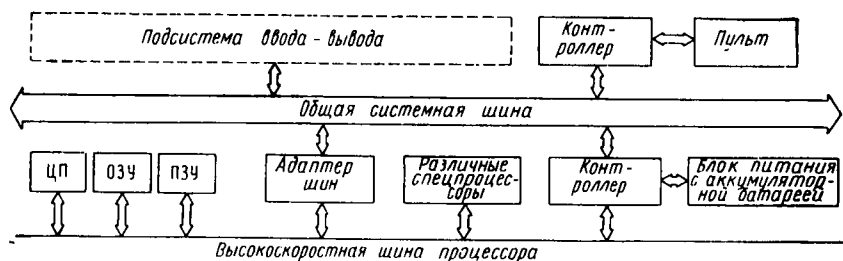


Рис. 5.22. Структурная схема ППЭВМ

тизации проектирования, управления, делопроизводства, а также для работы в локальных и глобальных вычислительных сетях. Кроме того, ЕС 1841 может использоваться также в качестве интеллектуального терминала ЭВМ серии ЕС.

При создании ПЭВМ ЕС 1841 учитывались два основных фактора: 1) своеобразие линии персональных ЭВМ (микропроцессорная, однопользовательская система) и сложившаяся линия развития архитек-

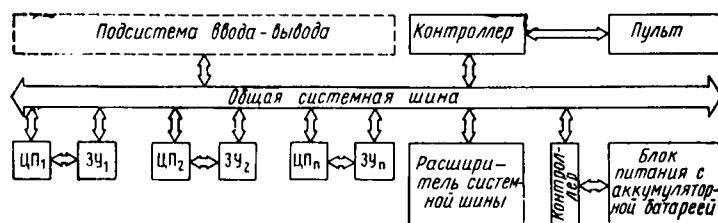


Рис. 5.23. Структурная схема высокопроизводительной ППЭВМ

туры; 2) особенности производства и использования семейства ЕС ЭВМ. ППЭВМ ЕС 1841 основного исполнения содержит следующие конструктивно-независимые модули: базовый электронный модуль, блоки накопителей на гибких и жестких магнитных дисках, печатающее устройство, монитор, клавиатуру.

Электронный модуль построен по принципу расширяемого процессора с системной шиной, которая представляет собой набор линий для передачи данных, адреса и сигналов управления, обеспечиваю-

щих универсальный интерфейс связи между модулями. Процессор ЕС 1841 построен на базе 16-разрядного микропроцессора КМ1810ВМ86.

В базовый состав ЕС 1841 включен цветной алфавитно-цифровой дисплей и предусмотрена возможность подключения монохромного и цветного графического дисплея.

В ЕС 1841 реализован принцип программной смены знакогенераторов дисплея и печати, что является основой для применения различных алфавитов и версий прикладного программного обеспечения. ЕС 1841 может работать с периферийными устройствами, имеющими связь по интерфейсам CENTRONICS или C2 (RS 232C).

Системное программное обеспечение (СПО) ППЭВМ ЕС 1841 включает операционную систему М86 по типу CP/M-86, сервисные программы, обеспечивающие работу с носителями программы оперативной коррективки и настройки на параметры конкретной установки, систему программирования БЕЙСИК М86 и др. Операционная система М86 позволяет использовать большинство пакетов прикладных программ, созданных для операционной системы CP/M-86, а также системы программирования, созданные для этой ОС: ПАСКАЛЬ, ФОРТ-РАН, СИ и др.

На предприятиях страны выпускаются ППЭВМ различных моделей: процессиональная ПЭВМ «Нейрон И9.66», предназначенная для обработки, управления, контроля и документирования алфавитно-цифровой и графической информации, решения задач автоматизации измерений и создания интегрированных цифровых систем; процессиональная ПЭВМ «Электроника ТЗ-29МК», ориентированная на применение в информационно-поисковых системах, системах подготовки программ и данных, в составе вычислительных сетей, а также в качестве интеллектуального терминала. Начато производство высокопроизводительных многопроцессорных ПЭВМ «Истра 4816».

Системное программное обеспечение ПЭВМ. В состав СПО входит набор программных модулей, реализующих функции организации и прохождения прикладных задач на конкретной конфигурации ПЭВМ и называемых операционной системой (ОС), а также программные модули трансляторов, ассемблеров, компиляторов и интерпретаторов, обеспечивающих эффективное взаимодействие пользователей с ПЭВМ на языках высокого уровня.

Можно выделить три типа ОС, используемых в ПЭВМ: однопрограммные, мультипрограммные и реального времени. Наиболее распространенными ОС в первых ПЭВМ являлись системы CP/M-80, CP/M-86, отличающиеся концептуальной простотой, надежностью, легкостью освоения и работы с ними. Их развитием являются системы MP/M-80, MP/M-86 и CP/Net, которые имеют файловую структуру, совместимую с файловой структурой системы CP/M. Система MP/M представляет собой мультитерминальную версию CP/M, обеспечивающую более эффективное использование таких ресурсов ПЭВМ, как процессор, ОЗУ и ВЗУ. Система CP/Net ориентирована на создание локальных сетей ПЭВМ и имеет средства генерации для различных

протоколов обмена между ПЭВМ. Перспективными ОС являются машинно-независимые системы UNIX, XENIX и OASIS.

В качестве языков высокого уровня широкое распространение в ПЭВМ получили языки БЕЙСИК, ПАСКАЛЬ, ФОРТРАН, АДА, ПЛ/М, СИ. Особое место в программном обеспечении ПЭВМ занимает прикладное программное обеспечение, с помощью которого осуществляется ориентация ПЭВМ на конкретные применения. К нему предъявляются требования, аналогичные требованиям к изделиям широкого применения: простота в обращении, доступность для неквалифицированного пользователя, постоянная готовность к применению.

Перспективы развития ПЭВМ. Микропроцессорная элементная база развивается в следующих направлениях. Созданы 32-разрядные кремниевые микропроцессоры Intel 80386, Motorola 68030, Motorola 68020 (с частотой 33 МГц), использующие сложные наборы команд и содержащие сотни тысяч транзисторов на одну кремниевую пластину.

Диаметрально противоположной концепцией является создание микропроцессоров с архитектурой РИСК (RISC — Reduced Instruction Set Computers — компьютер с сокращенным набором команд). В отличие от Intel 80386 и других МП, использующих сложные наборы команд, РИСК-процессоры программируются меньшим и упрощенным набором команд.

В основу архитектуры РИСК положено утверждение о том, что чем меньше функций выполняется на уровне аппаратных средств, тем лучше. В традиционных микроЭВМ команды представляют собой инструкции, интерпретируемые непосредственно аппаратными средствами. С целью упрощения процесса программирования в наборы команд большинства современных ЭВМ входят достаточно сложные команды, каждая из которых может быть реализована в виде последовательности более простых команд. В архитектуре РИСК ограниченность набора команд является следствием исключения из него сложных команд.

В современных МП команды реализованы в виде последовательностей микрокоманд, образующих микропрограммы, размещаемые в ПЗУ непосредственно на кристалле центрального процессора. Наборы команд МП с традиционной архитектурой состоят из 100 и более команд, в то время как наборы команд микропроцессоров с архитектурой РИСК содержат 30 ... 50 команд.

Чем больше (а значит, сложнее) набор команд, тем больше объем микропрограмм, которые необходимо разместить в ПЗУ непосредственно на кристалле центрального процессора. Но известно, что сложные команды используются относительно редко. Кроме того, часто последовательное выполнение ряда простых шагов оказывается более эффективным, чем непосредственное выполнение одной сложной команды. В соответствии с этим архитектура РИСК обеспечивает уменьшение набора команд до минимума.

Использование сокращенного набора команд освобождает место на кристалле, на котором размещается центральный процессор. В ре-

в результате этого производительность МП с архитектурой РИСК может быть повышена, поскольку на освобождающемся месте можно разместить дополнительное число регистров. Большее число регистров позволяет уменьшить число операций обращения к ОЗУ для хранения промежуточных результатов. Однако при использовании микроЭВМ с архитектурой РИСК реализация сложных функций, выполняемых в ЭВМ с традиционной архитектурой на аппаратном уровне, перекладывается на уровень программного обеспечения, в частности на трансляторы, с различных языков программирования. Примером микропроцессора с архитектурой РИСК может служить МП MC88000 фирмы Motorola. Отличительными особенностями этого МП являются встроенная арифметика над числами с плавающей точкой, регистровая память емкостью 16 К байт. На базе микропроцессора с архитектурой РИСК фирмой IBM выпущена ПЭВМ модели IBM PC RT.

Перспективным направлением развития архитектур ПЭВМ является разработка систем параллельной обработки. Для этой цели разрабатываются специальные параллельные процессоры — транспьютеры. Высокое быстродействие транспьютеров связано с тем, что они строятся по архитектуре РИСК.

Большое будущее прогнозируется за арсенидом галлия — металлическим сплавом, способным заменить кремний в МП, что позволит увеличить скорость работы МП по меньшей мере в 3 раза.

§ 5.6. МИКРОПРОЦЕССОРНЫЕ ЛОКАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ

Микропроцессорные локальные вычислительные сети (ЛВС) предназначены для реализации распределенной обработки информации в объектах различной природы. К таким объектам прежде всего относятся системы управления технологическими процессами, системы контроля (в том числе и контроля РЭС), системы автоматизированного проектирования, гибкие перепрограммируемые автоматизированные производства и т.д. Это связано с тем, что распределенная обработка данных является единственным реальным методом, на основе которого создаются высокопроизводительные системы оперативной обработки информации.

Микропроцессорные ЛВС представляют собой вычислительные системы, объединяющие с помощью каналов связи от десятков до нескольких сотен узлов (ЭВМ, микроЭВМ, терминалов, микропроцессорных модулей — ММ и др.), расположенных на небольших расстояниях друг от друга (10 ... 15 км).

Основное назначение ЛВС — распределение дорогостоящих ресурсов (программ, запоминающих устройств большой емкости, быстродействующих печатающих устройств и т.д.) для эффективного решения поставленных задач. ЛВС должна иметь надежную, быструю и дешевую систему передачи данных (СПД), а стоимость передаваемой единицы информации должна быть значительно ниже стоимости обработки

информации. Для достижения этого ЛВС как система распределенных ресурсов должна выполняться на основе следующих принципов.

Принцип единых протоколов. Протоколы межмашинной связи с ЛВС предназначены для организации обмена информацией между компонентами сети. Протоколы сети определяют форму сообщения или пакета сообщений (длину, заголовок, знак окончания, дополнительную информацию для повышения достоверности передачи и др.). Все процедуры управления и соответствующие им протоколы едины для всей сети и не зависят ни от типа ЭВМ, подключенных к сети, ни от происходящих в них процессов.

Принцип единой передающей среды. При построении СПД ЛВС используется активная или пассивная структура передающей среды.

Активная структура выполняется на основе распределенных усилителей и преобразователей, обеспечивающих передачу информации в параллельном и последовательном кодах. Пассивная структура выполняется на основе пассивного носителя — коаксиального либо плоского кабеля — и использует преобразователи усилителя одного типа, обеспечивающие возможность работы в параллельном либо в последовательном коде. Структура передающей среды может быть реализована с применением либо моноканала, либо многопроводной связи. Более дешевой является структура с моноканалом, поскольку существенно снижаются издержки на эксплуатацию и прокладку соединений, поэтому для ЛВС эта структура является более предпочтительной.

Моноканалом является физическая среда, аппаратные и, возможно, программные средства, предназначенные для параллельной передачи информации одновременно (с точностью до времени распространения сигнала) всем абонентским системам. Моноканал предназначен для коллективного использования большим числом абонентских систем, поэтому он должен обладать высокой пропускной способностью передачи информации. Физическая среда моноканала реализуется посредством волоконно-оптических линий связи, коаксиальных или плоских кабелей, скрученных пар проводов и т.д.

Принцип единого метода управления. Протоколы ЛВС могут применять централизованные и децентрализованные формы управления одноузловой структурой моноканала. Принцип единого метода управления проявляется в выборе одной из этих форм, обеспечивающей достаточную надежность работы СПД и максимальную загрузку каналов связи. При этом для определения метода управления следует учитывать структуру соединений, их длину, число абонентов и сложность обработки информации с помощью ресурсов ЛВС.

Для *централизованных* форм управления характерны обилие служебной информации и приоритетность подключения к моноканалу станций. Защита от конфликта в моноканале реализуется центральной управляющей машиной.

В *децентрализованных* формах управления, допускающих одинаковый приоритет всех станций, подключаемых к моноканалу, применя-

ются многоступенчатые тракты защиты от конфликтов, учитывающие противоречивые требования надежности и максимальной загрузки моноканала.

При использовании в ЛВС нескольких методов управления средой передачи данных существенно увеличивается сложность схемных решений контроллеров, с помощью которых станции ЛВС подключаются к этой среде.

Принцип информационной и программной совместимости. Этот принцип предусматривает совместимость операционных систем, программ и систем управления базами данных (СУБД), рассредоточенных в рамках ЛВС. Его особенностями являются возможность адаптации процессов к видам пересылаемой информации и применение единых систем кодирования и контроля информации.

Принцип гибкой модульной организации. Данный принцип предусматривает проектирование СПД ЛВС на основе набора гибких конструктивно законченных модулей.

Локальные вычислительные сети могут быть *классифицированы*:

- а) по топологическим признакам — ЛВС произвольной (иерархической), кольцевой конфигурации, конфигурации типа «Общая шина», звездообразной конфигурации;
- б) по методам управления ресурсами среды передачи данных — ЛВС с детерминированным и случайным доступом к моноканалу;
- в) по программному обеспечению — ЛВС с единой операционной поддержкой и едиными методами теледоступа, ориентированными на конкретную ЛВС, а также ЛВС с различными наборами тех и других компонентов операционной поддержки;
- г) по техническому обеспечению — гомогенные и гетерогенные ЛВС. Первые предусматривают применение в станциях однотипного оборудования, например только комплексов отображения информации на дисплейных станциях. Вторые дают возможность подключения любых абонентских комплексов — от устройств выдачи документации о результатах обработки информации до высокопроизводительных вычислительных комплексов с развитой терминальной сетью.

Различают одно- и многоузловые ЛВС. Под многоузловыми сетями будем понимать сети, в которых одновременно несколько узлов могут вести передачу с несколькими коммуникационными узлами, а под одноузловыми сетями — сети, где возможность коммуникации имеет только один узел.

Конфигурация каналов СПД определяет топологию ЛВС. В высокопроизводительных ЛВС, реализованных на базе микропроцессоров и микроЭВМ с использованием в СПД волоконно-оптических линий связи или коаксиальных кабелей, наибольшее распространение получили топологии с упорядоченным расположением узлов по типу «шина» и «кольцо». ЛВС с топологией «шина» и «кольцо» позволяют исключить центральный узел, используемый в топологии «звезда», и расширить возможность взаимодействия остальных узлов. Примером самой известной ЛВС типа «Общая шина» является сеть Ethernet фирмы Xerox

Corp, а кольцевой ЛВС — маркерное кольцо фирмы IBM и сеть типа Cambridge Ring (CR).

Основным достоинством локальной вычислительной сети с общей шиной (рис. 5.24) являются малые затраты на расширение сети при подсоединении дополнительных абонентов и простота методов управления потоками информации. Недостаток такой сети — технические сложности передачи информации на большие расстояния без ретрансляторов и последовательный характер работы сети, при котором в определенный момент возможно передавать только один пакет информации — короткий сегмент информации, содержащий информацию пользователя и служебную информацию, позволяющую различать пакеты и выявлять возникающие при передаче ошибки.

В кольцевой сети сообщения проходят от одного узла к другому по одностороннему каналу передачи. Здесь также не требуется выбирать маршрут: каждый узел связи, подключенный к каналу передачи, передает сообщение по кольцу к узлу назначения. В таких сетях возникают проблемы передачи управления от узла к узлу и определения узла, имеющего право на передачу сообщения в данный момент времени. Эту функцию выполняет механизм управления передачей данных. Кольцевые СПД обладают потенциальной возможностью дальнейшего наращивания скорости передачи и децентрализации управления доступом к среде и передачей данных.

В качестве физической среды передачи данных в ЛВС используются витые пары проводников, коаксиальный кабель и волоконно-оптические линии связи. Витые пары отличаются низкой стоимостью и простотой подключения новых абонентов, но при этом невысоки скорость передачи и число подключаемых абонентов, поэтому витые пары применяются в основном в сетях персональных и персонально-профессиональных ЭВМ. Коаксиальный кабель дороже витой пары, но обеспечивает большие скорости передачи (10 М бит/с) и помехозащищенность. Наивысшие скорости (до 150 М бит/с), помехозащищенность и протяженность магистралей без специальных повторителей позволяют получить волоконно-оптические линии связи. К недостаткам этих линий следует отнести большие сложность и стоимость подключения новых абонентов.

В локальных сетях каждый абонент связывается с другим абонентом через общий канал. Поэтому здесь возникают проблемы совместно-

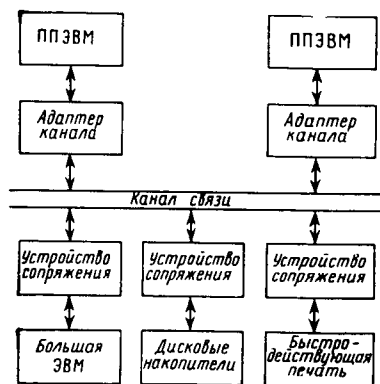


Рис. 5.24. Структурная схема локальной вычислительной сети типа «Общая шина»

го использования канала, а также разрешения конфликта и столкновений в канале. Методы организации доступа абонентов к каналу связи можно разделить на две основные группы: методы с детерминированным доступом и методы со случайным доступом.

В сетях с детерминированным доступом общий канал распределяется между абонентами по специальным алгоритмам в детерминированные моменты времени. При этом распределение во времени может быть как синхронным, так и асинхронным.

В случае *синхронного распределения* число интервалов времени использования канала соответствует числу абонентов в сети. Каждому абоненту выделяется определенный интервал времени, в течение которого он имеет доступ к каналу для передачи информации. При таком подходе могут возникнуть ситуации, когда одни абоненты не успевают передать информацию, а другие не имеют информации для передачи. Синхронный метод разделения канала отличается простотой организации, но используется при малом числе абонентов в сети.

Асинхронное распределение учитывает интенсивность и объем информации, передаваемой абонентом, что обуславливает различные интервалы доступа для различных абонентов.

В сетях со случайным доступом для эффективного использования каналов применяются методы, позволяющие уменьшить объем управляющей информации и увеличить скорость доступа. При случайном доступе каждый абонент может передавать информацию в любой момент времени, что приводит к возникновению конфликта за захват канала. Повышение коэффициента использования канала при случайном доступе обеспечивает ряд модификаций этого метода.

Случайный доступ с обнаружением передачи предоставляет каждому абоненту возможность прослушивать канал перед выдачей информации. Если абонент обнаруживает передачу каких-либо сообщений, то он ожидает окончания этой передачи. Прослушивание канала перед передачей позволяет уменьшить вероятность возникновения конфликтов и повысить относительную пропускную способность канала до 80 %.

Случайный доступ с контролем столкновений обеспечивает одновременно с передачей информации прослушивание канала. Использование этого метода позволяет довести пропускную способность канала до 90 %.

Эффективность работы ЛВС зависит от многих факторов, связанных с реализацией протоколов, топологией сети, числом узлов-абонентов сети, параметрами контроллеров и блоков доступа к каналам связи. Существуют различные подходы к оценке эффективности функционирования ЛВС. Первая группа методов оценки основана на обобщенном математическом аппарате и не всегда учитывает многие реально действующие факторы, вторая группа — на построении моделей функционирования ЛВС с учетом обобщенных характеристик СПД ЛВС и соответствующей поддержкой характеристик СПД.

Определенный интерес представляет критерий оценки качества моноканальных ЛВС (в мкс·К байт/км) вида

$$Q_c = C_1 T_n N_0 / C_2 L N_n.$$

Здесь C_1 — стоимость разработки и эксплуатации СПД, в которую включается стоимость оборудования среды передачи данных, контроллеров доступа к этой среде (аппаратуры и программ), обеспечивающих устойчивость функционирования СПД; T_n — время передачи кадров информации от станции-передатчика к станции-приемнику с учетом упаковки блоков информации (пакетов) в кадры и обратной распаковки, работы блоков сопряжения с моноканалом, задержек, связанных со столкновениями в моноканале, а также любых других задержек в случае применения иных методов управления доступом к моноканалу; N_0 — емкость буферной памяти, необходимой для размещения очереди передаваемых и принимаемых кадров информации; C_2 — стоимость разработки, приобретения и эксплуатации оборудования и программ абонентских станций для обработки информации пользователя; L — допустимая длина кабельных соединений между станциями (приемопередатчиками); N_n — допустимое число станций ЛВС (допустимая нагрузка).

Очевидно, что Q_c стремится к минимуму, поскольку идеальная система передачи данных имеет минимальные значения C_1 , T_n , N_0 и максимальные значения C_2 , L , N_n . Анализируя выражение для Q_c , следует отметить, что ряд параметров, а именно C_1 , C_2 и L , отражает содержание технического задания на систему передачи данных ЛВС, а остальные параметры (T_n и N_0) являются сложными функциями от ряда аргументов. Так, например, параметр T_n отражает время доставки кадра информации от станции-отправителя к станции-получателю, однако он должен учитывать время упаковки пакета в кадр в соответствии с протоколом сети, время доступа к моноканалу, зависящее от применяемого алгоритма и его схематического решения. Параметр N_0 , с одной стороны, испытывает влияние параметра T_n , так как увеличение T_n приводит к росту N_0 , а с другой — увеличение N_0 возможно вследствие влияния факторов функциональной среды более высоких уровней. Поскольку функциональная связь между параметрами, входящими в выражение для Q_c , является довольно сложной, для определения качества функционирования ЛВС целесообразно проводить имитационное моделирование, использующее описание ЛВС как модели массового обслуживания на верхних уровнях; модели же нижних уровней в основном описывают работу элементов СПД с учетом технических показателей интегральных микросхем.

Контрольные вопросы и задания

1. Объясните основные особенности организации микроЭВМ на однокристальных процессорах.

2. Какие факторы и параметры элементной базы влияют на архитектуру микроЭВМ?

3. Какие существуют способы распараллеливания вычислений?

4. Какие требования предъявляются к интерфейсам микроЭВМ?

5. Проведите качественный анализ различных способов построения аналого-цифровых преобразователей.

6. Объясните необходимость применения прямого, смещенного и дополнительного кодов в цифроаналоговых преобразователях.

7. Объясните принцип действия аналоговых микропроцессоров.

8. Какие факторы влияют на выбор разрядности АЦП?

9. Какие факторы выделяют персональные ЭВМ из всего множества микроЭВМ?

10. Спрогнозируйте перспективы развития профессиональных ПЭВМ.

11. Прокомментируйте основные принципы построения локальных вычислительных сетей.

12. Какие критерии эффективности функционирования локальных вычислительных сетей применяются при оценке микропроцессорных РЭС?

13. Почему в ЛВС используются различные методы доступа к каналу?

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА В РАДИОЭЛЕКТРОННЫХ СРЕДСТВАХ

§ 6.1. ОБОСНОВАНИЕ ПРИМЕНЕНИЯ МП В СОСТАВЕ РЭС

Появление МП создало благоприятные условия для реализации в РЭС многих оптимальных алгоритмов обработки сигналов, ранее отвергаемых из-за сложности и больших аппаратных затрат. В результате применения МП улучшились тактико-технические характеристики РЭС, их масса, размеры, потребляемая мощность и надежность. Существенно сократились сроки разработки аппаратуры и появилась возможность ее модификации в процессе эксплуатации за счет изменения алгоритма функционирования, хранящегося в ПЗУ.

Микропроцессорные устройства в составе РЭС функционально ориентированы на решение одной конкретной задачи в течение всего жизненного цикла.

Для нормального функционирования МПУ в составе РЭС должно быть осуществлено сопряжение их с аппаратурой РЭС посредством применения АЦП и ЦАП. Кроме того, должна быть обеспечена возможность приема, хранения, преобразования и выдачи информации.

Поскольку в течение жизненного цикла аппаратные и программные средства МПУ, как правило, не претерпевают изменений, в процессе проектирования должны быть решены задачи достижения максимальной производительности реализации алгоритмов, выбора и обоснования серии МПК БИС, выбора микропроцессора с оптимальным быстродействием реализации заданного алгоритма с учетом временных характеристик управляемого объекта (РЭС), а также целого ряда задач оптимизации (распределения памяти, решения задачи дуализма «аппаратные средства — программное обеспечение» и т. д.). Решение этих задач при проектировании МПУ позволяет добиваться получения структурно-архитектурных решений с оптимальными характеристиками.

К примеру, решение задачи выбора минимально допустимого быстродействия МП при реализации заданного алгоритма в реальном времени позволяет одновременно улучшать такие характеристики МПУ, как энергоемкость и надежность. Решение задачи интенсификации загрузки регистров общего назначения позволяет минимизировать число РОН в МПУ (минимизация оборудования) и увеличивать производительность МПУ за счет минимизации числа обращений к оперативной памяти.

Решение различных задач оптимизации в процессе проектирования МПУ направлено на сокращение избыточности аппаратных средств,

что имеет широкое распространение в универсальных микроЭВМ, и на согласование структурно-архитектурных решений МПУ с характеристиками и структурой реализуемых алгоритмов.

§ 6.2. ОПРЕДЕЛЕНИЕ НИЖНЕЙ ОЦЕНКИ БЫСТРОДЕЙСТВИЯ МИКРОПРОЦЕССОРА

При проектировании микроцессорных РЭС одним из решающих факторов является правильный выбор типа МПК БИС в соответствии с совокупностью требований к аппаратуре, программному обеспечению и вспомогательным средствам. Микропроцессорная аппаратура в составе РЭС должна функционально обеспечивать обработку данных, как правило, при большом потоке вводимых и небольшом потоке выводимых данных. К используемым в РЭС микропроцессорным устройствам предъявляются требования высокой надежности и необходимой производительности.

Повышение производительности микропроцессора ухудшает его надежные и энергетические характеристики. Поэтому для построения конкретной аппаратуры следует определять нижнюю оценку быстродействия МП, гарантирующую выполнение требований к производительности обработки потока входных данных.

Для определения нижней оценки быстродействия МП рассмотрим формальную модель вычислительного процесса в МПУ.

Функционально МПУ должно формировать множество выходных величин $Y = \{Y_p\}$, $p = \overline{1, n_1}$ не позже чем в заданные моменты времени $T_Y = \{t_Y^{(p)}\}$, путем преобразования в соответствии с функцией F информации NUX , состоящей из данных N , хранящихся в памяти МПУ, и множества входных величин $X = \{X_q\}$, $q = \overline{1, n_2}$, появляющихся в моменты времени $T_X = \{t_X^{(q)}\}$. Преобразования информации $Y = F(NUX)$ производятся посредством выполнения различных совокупностей функциональных операторов $L = (A_p^{(1)}, \dots, A_q^{(k)})$, $A_i \in A = \{A_1, \dots, A_n\}$. Обозначение $A_q^{(k)}$ означает, что оператор A_q стоит в последовательности L на k -м месте.

Рассмотрим формализованное представление алгоритмов в виде граф-схемы. Всякий алгоритм можно представить в виде некоторой последовательности $L = (A_p^{(1)}, \dots, A_q^{(k)})$ операторов, осуществляющей преобразование информации $Y = F(XUN)$.

Свяжем операторы A_i с вершинами графа. Будем соединять вершину, соответствующую функциональному оператору A_i , с вершиной, соответствующей функциональному оператору A_j , только в том случае, когда результат, полученный после реализации функционального оператора A_i , является одним аргументом для оператора A_j . После этого получим функциональный граф. На рис. 6.1 приведена функциональ-

ная граф-схема алгоритма, определяемого такой последовательностью функциональных операторов:

$$\begin{aligned} A_1(X_1) &= (Y_1^{(1)}, Y_2^{(1)}, Y_3^{(1)}); & A_2(Y_2^{(1)}) &= (Y_1^{(2)}, Y_2^{(2)}); \\ A_3(X_2, Y_3^{(1)}) &= (Y_1^{(3)}); & A_4(Y_1^{(1)}, Y_1^{(2)}, Y_1^{(3)}) &= (Y_1^{(4)}); \\ A_5(Y_2^{(2)}) &= (Y_1^{(5)}); & A_6(Y_1^{(4)}, Y_1^{(5)}) &= (Y_1). \end{aligned}$$

Функциональная граф-схема отражает не структуру конкретной программы, а структуру выбранного метода решения (алгоритма решения). Структура функциональной граф-схемы предопределяет структуру всех возможных программных алгоритмов, реализующих данный алгоритм решения.

Для определения модели, отражающей структуру функциональных связей в выбранном методе решения задачи функционирования МПУ в составе РЭС с учетом ее решения в реальном времени, строят граф-схему алгоритма таким образом.

Элементы последовательности операторов $L = (A_p^{(1)}, \dots, A_q^{(k)})$ отождествляют с вершинами графа, а информационные связи между операторами — с его дугами. Вводят множества начальных и конечных вершин граф-схемы, соответственно отождествляемых с множеством входных X и множеством выходных Y сигналов МПУ. Эти вершины можно интерпретировать как операции связи с функционирующими РЭС (внешней средой), причем начальные вершины интерпретируются как операции восприятия сигналов контроля, а конечные — выдачи результатов контроля. Каждая начальная вершина обозначается T_{X_i} , а конечная — T_{Y_j} .

Поскольку величины T_{X_i} и T_{Y_j} формируются внешней средой (функционирующими РЭС), тем самым задается еще и временное отношение порядка выполнения операторов. Полученный в результате граф G ограничений информационного и временного порядков называется функциональным графом системы.

Граф G описывает ограничения порядка между элементами множества $\{A_1, \dots, A_n, X_1, \dots, X_2, Y_1, \dots, Y_{n_1}\}$ и может быть задан соответствующей ему матрицей смежности $\alpha = \|\alpha_{ij}\|$ размерности $(n + n_1 + n_2) \times (n + n_1 + n_2) = m \times m$.

В общем случае в функциональном графе системы порядок выполнения операторов — частичный, и некоторые моменты времени готовы к выполнению несколько операторов. Это обуславливает наличие множества допустимых последовательностей выполнения операторов, реализующих закон F , что, в свою очередь, дает возможность осуществ-

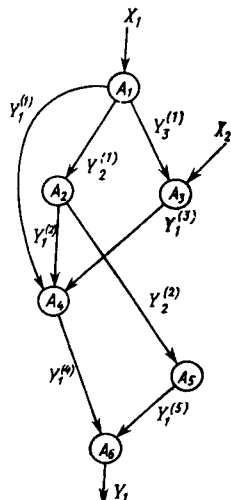


Рис. 6.1. Функциональная граф-схема алгоритма

вить "поиск" оптимального в некотором смысле вычислительного процесса реализации алгоритмов функционирования РЭС.

Исходными данными для определения нижней оценки быстродействия МП являются функциональный граф G системы со множествами моментов времени $T_X = \{T_{X_q}\}$ и $T_Y = \{T_{Y_p}\}$ поступления входных X и выдачи выходных данных Y соответственно, а также временные характеристики реализации операторов $A_i \in A$, входящих в состав алгоритма L функционирования МПУ. При реализации в МПУ множества несвязанных алгоритмов обработки информации $L = \{L_k\}$ следует выбирать алгоритм L_0 с самыми жесткими ограничениями в пересчете

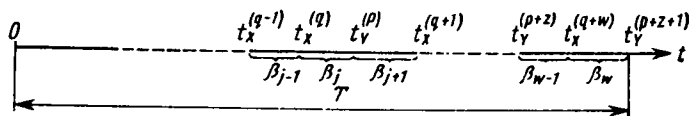


Рис. 6.2. Последовательность реализации операторов в МПУ

на один элементарный оператор, причем в зависимости от уровня записи алгоритма под A_i можно понимать как отдельную команду, так и целую процедуру. Очевидно, время выполнения каждого $A_i \in A$ можно представить в виде

$$\tau_i = k_i \tau,$$

где τ — время выполнения некоторого оператора, принятого за единицу.

Задача определения нижней оценки быстродействия МП заключается в выборе максимально допустимого времени $\tau = \tau_{доп}$ расписания работы ММ, построенного на основе данного МП, минимизирующих целевую функцию

$$\Delta = T - \sum_{i=1}^w k_i \tau_i, \quad T = T_{Y_d} - T_{X_1},$$

где Δ — время простоя ММ.

На временной оси работы МП расставим элементы множества $T_X \cup T_Y$, перенумеруем полученные временные отрезки и обозначим длину j -го отрезка ($j = \overline{1, w}$) через β_j (рис. 6.2). Требуется найти условия и метод заполнения временных отрезков по критерию минимума времени простоя МП.

Построим исходную допустимую последовательность выполнения операторов графа G и перепишем целевую функцию в виде

$$\Delta = T - \sum_{i=1}^w \Delta_i, \quad (6.1)$$

где Δ_i — время простоя МП на отрезке β_i .

Будем увеличивать τ до тех пор, пока какой-либо отрезок β_i не будет заполнен полностью. Тогда допустимое время определится из соотношения

$$\tau_{\text{доп}} = \beta_i / \sum_i k_i^{(i)}. \quad (6.2)$$

Здесь $\sum_{i \in \beta_j} k_i = \sum_i k_i^{(j)}$ (суммирование производится по всем операторам, которые выполняются на отрезке β_j).

Время простоя МП Δ_j на отрезке β_j можно представить в виде

$$\Delta_j = \beta_j - \sum_i k_i^{(j)} \tau_{\text{доп}}. \quad (6.3)$$

Подставляя (6.2) и (6.3), получаем

$$\Delta_j = \beta_j \left(1 - \frac{\sum_i k_i^{(j)} \beta_i}{\sum_i k_i^{(i)} \beta_j} \right). \quad (6.4)$$

Из (6.4) следует, что для того, чтобы значение целевой функции 6.1) было равно нулю, необходимо выполнить условие

$$\beta_i / \sum_i k_i^{(i)} = \beta_j / \sum_i k_i^{(j)} = \tau_{\text{доп}}^{\text{н}},$$

где $\tau_{\text{доп}}^{\text{н}}$ — значение $\tau_{\text{доп}}$ в идеальном случае (при $\Delta = 0$).

С другой стороны,

$$\tau_{\text{доп}}^{\text{н}} = T / \sum_{i=1}^w k_i.$$

Тогда идеальное значение $\sum_i k_i^{(j)}$ на отрезке β_j

$$\sum_i k_i^{(i)} = \frac{\sum_{i=1}^w k_i}{T} \beta_j.$$

Обозначив $\sum_{i=1}^w k_i = K$, запишем

$$\sum_i k_i^{(j)} = \frac{K}{T} \beta_j = \xi_j.$$

В результате конкретного заполнения временных отрезков может оказаться, что на некоторых отрезках β_j будет недобор, т. е. $\sum_i k_i^{(j)} = \xi_j - \eta_j$, на других — перебор, т. е. $\sum_i k_i^{(j)} = \xi_j + \delta_j$.

Учитывая, что $\sum_i k_i^{(i)} = \beta_j / \tau_j$, получим:

в первом случае

$$\tau_j' = \beta_j / (\xi_j - \eta_j), \quad (6.5)$$

во втором случае

$$\tau_j'' = \beta_j / (\xi_j + \delta_j). \quad (6.6)$$

Выражения (6.5) и (6.6) показывают, что всегда $\tau_j'' < \tau_j'$, т. е. τ_j'' определяет быстродействие МП.

Пусть имеется несколько отрезков β_j , в которых получились переборы δ_j . Тогда

$$\tau_{\text{доп}} = \min_i \{\tau_i''\}, \quad (6.7)$$

где τ_j'' — время, найденное по формуле (6.6), если на отрезке имеется перебор δ_j .

Преобразуем выражение (6.6) следующим образом:

$$\tau_j'' = \frac{\beta_j}{\frac{K}{T} \beta_j + \delta_j} = \frac{T}{K + \frac{\delta_j}{\beta_j} T}. \quad (6.8)$$

Из (6.8) непосредственно получаем

$$\tau_{\text{доп}} = \min_i \{\tau_i''\} = \frac{T}{K + \max_i \left\{ \frac{\delta_j}{\beta_j} \right\} T}. \quad (6.9)$$

Следовательно, величина $\tau_{\text{доп}}$ определяется отрезком, на котором относительный перебор δ_j/β_j максимален.

Методику определения нижней оценки быстродействия МП проиллюстрируем на примере.

Пример 6.1. В аппаратуре оперативного контроля сложных РЭС реализуется алгоритм пересчета входного параметра, функциональный граф которого приведен на рис. 6.3.

Работа алгоритма заключается в обработке и преобразовании входных сигналов X_2, X_3, X_4 , их сравнении с эталонным сигналом X_1 и выдаче информации о качестве контролируемых РЭС в виде сигналов Y_1, Y_2 .

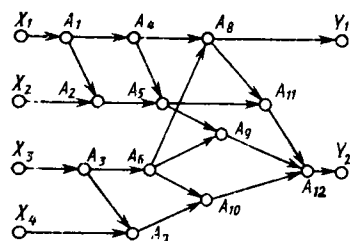


Рис. 6.3. Функциональный граф алгоритма пересчета входных параметров

Входные сигналы $X = \{X_q\}$, $q = \overline{1,4}$, появляются в определенные моменты времени $T_{X_1} = 0$, $T_{X_2} = 6\tau$, $T_{X_3} = 24\tau$, $T_{X_4} = 50\tau$.

Выходные сигналы $Y = \{Y_p\}$, $p = \overline{1,2}$, должны быть выработаны к моментам времени $T_{Y_1} = 42\tau$, $T_{Y_2} = 72\tau$. Значение параметра τ зависит от типа контролируемых РЭС и обычно лежит в пределах от 0,5 до 20 мкс.

Все операторы A_i , $i = \overline{1,12}$, реализуют шесть операций O_{ξ} , относительное время выполнения которых соответственно равно $k_1 = 5$,

$k_2 = 1, k_3 = 7, k_4 = 9; k_5 = 3, k_6 = 6$. Операции O_1 соответствуют операторы A_2, A_9 , операции $O_2 — A_1, A_4, A_6$, операции $O_3 — A_3, A_{10}$, операции $O_4 — A_5, A_{11}$, операции $O_5 — A_7$ и операции $O_6 — A_8, A_{12}$.

Необходимо определить нижнюю оценку быстродействия процессора, реализующего данный алгоритм.

Построим табл. 6.1, столбцы которой соответствуют временным отрезкам j с длинами β_j , а строки — операторам A_i с относительными длительностями выполнения k_i .

Таблица 6.1

Временные возможности реализации операторов алгоритма

A_i	j		1	2	3	4	5
	k_i	β_j	6	18	18	8	22
A_1	1		×				
A_2	5			×	×	×	×
A_3	7				×	×	
A_4	1		×	×	×	×	
A_5	9			×	×	×	×
A_6	1				×	×	
A_7	3						×
A_8	6				×	×	
A_9	5				×	×	×
A_{10}	7						×
A_{11}	9				×	×	×
A_{12}	6						×

На первом временном отрезке ($\omega = 1$) длительностью $\beta_1 = 6$ может быть реализован либо только оператор A_1 , либо операторы A_1 и A_4 .

Набор операторов, реализуемых на втором отрезке ($\omega = 2$), следует выбирать из множества вариантов: $\{A_2, A_4, A_5\}$ или $\{A_2, A_5\}$.

Аналогично, анализируя все временные отрезки, нетрудно убедиться, что на остальных трех временных отрезках ($\omega = 3, 4, 5$) возможен перебор операторов.

Используя процедуру максимального заполнения временных отрезков на временной оси, получим последовательность выполнения операторов вида $L = (A_1, A_2, A_4, A_5, A_3, A_6, A_8, A_9, A_7, A_{10}, A_{11}, A_{12})$, при этом имеется следующее распределение операторов на временной оси: 1 — A_1 ; 2 — A_2, A_4, A_5 ; 3 — A_3, A_6, A_8 ; 4 — A_9 ; 5 — $A_7, A_{10}, A_{11}, A_{12}$. На рис. 6.4 изображена временная диаграмма реализации алгоритма пересчета входного параметра.

В последовательности выполнения операторов L только один временной отрезок ($\omega = 5$) имеет перебор. В соответствии с выражением (6.6) находим величину τ_j'' , определяющую минимально допустимое быстродействие МП:

$$\tau_j'' = \frac{\beta_j}{\xi_j + \delta_j} = \frac{22}{25} \tau = 0,88\tau.$$

В табл. 6.2 приведены длительности выполнения элементарной операции O_2 , соответствующей операторам A_1, A_4, A_6 в микропроцессорных модулях на основе МП К1804BC1, К1804BC2, К1801BC1, К1801BC2, К583ИК1, К588BC1, К580ИК80, К587ИК1.

Таблица 6.2

Длительность выполнения элементарной операции

Тип МП	$\tau_{МП}$, мкс	Тип МП	$\tau_{МП}$, мкс
К1804BC1	0,2	К583ИК1	3,3
К1804BC2	0,3	К588BC1	10,0
К1801BC1	2,0	К580ИК80	10,0
К1801BC2	1,0	К587ИК1	20,0
К589ИК02	2,0		

Если в нашем примере $\tau = 1$ мкс, то $\tau_{доп} = 0,88$ мкс. Это означает, что для реализации заданного алгоритма следует выбрать МП, для

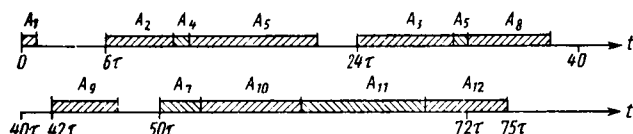


Рис. 6.4. Временная диаграмма реализации алгоритма пересчета входных параметров

которого $\tau_{МП} \leq 0,88$ мкс, т. е. МП К1804BC2. Если же, например, окажется, что $\tau = 15$ мкс, то $\tau_{МП} \leq 0,88 \cdot 15 = 13,2$ мкс и можно выбрать любой тип МП, кроме К587ИК1, в частности К580ИК80 или К588BC1.

§ 6.3. ОПТИМИЗАЦИЯ ПРОГРАММ ФУНКЦИОНИРОВАНИЯ МПУ

Алгоритм функционирования МПУ реализуется посредством взаимодействия аппаратных средств (АЛУ, РОН, ОЗУ и др.), при программном или микропрограммном управлении. Программа реализации алгоритма представляет собой упорядоченную последовательность выполнения операторов, построенную в соответствии с информационным отношением порядка между операторами. Порядок следования операторов в программах оказывает существенное влияние на загрузку СОЗУ и ОЗУ, а следовательно, и на время реализации программы. Любой порядок следования операторов в программе характеризуется числом информационных связей (промежуточных результатов) между операторами. Очевидно, что среди множества вариантов упорядоченных последовательностей операторов имеется вариант,

обеспечивающий максимальную загрузку РОН МПУ промежуточными результатами. Это ведет к уменьшению потерь времени, затрачиваемого в процессе реализации алгоритма на обращения к оперативной памяти для записи промежуточных результатов и их последующего чтения для выполнения операторов — потребителей этих промежуточных результатов.

Постановка задачи. Исходная информация для задачи оптимизации программы функционирования МПУ — алгоритм функционирования МПУ, заданный в виде функционального графа G , на котором определено отношение предшествования $<$, причем запись $A_i < A_j$ означает, что оператор A_j использует результирующую величину оператора A_i . Это отношение порождает ориентированный граф (орграф) $G = (X, U)$, где $U = \{(A_i, A_j) \mid (A_i < A_j)\}$. Для удобства решения задачи будем считать, что в графе G имеется единственная входная вершина (что не влияет на общность результатов) и отсутствуют контуры и параллельные дуги.

Введем некоторые определения.

Укладка графа $G = (X, U)$ — такая последовательность $L(G) = (A_{i_1}, A_{i_2}, \dots, A_{i_n})$ всех вершин графа, при которой для каждой дуги $(A_{i_k}, A_{i_p}) \in U$ справедливо неравенство $k < p$. Будем считать, что вершины A_{i_k} и $A_{i_{k+1}}$ находятся в отношении соседней связности в укладке L , если $(A_{i_k}, A_{i_{k+1}}) \in U$.

Число соседней связности укладки L — величина

$$\kappa(L) = \sum_{k=1}^{n-1} \delta(A_{i_k}, A_{i_{k+1}}),$$

где $\delta(A_{i_k}, A_{i_{k+1}}) = \begin{cases} 1, & \text{если } (A_{i_k}, A_{i_{k+1}}) \in U, \\ 0 & \text{— в противном случае.} \end{cases}$

Компонента укладки — последовательность вершин этой укладки $C = (A_{j_1}, \dots, A_{j_t})$, удовлетворяющих условиям

$$\delta(A_{j_{i-1}}, A_{j_i}) = 0; \quad \delta(A_{j_t}, A_{j_{t+1}}) = 0,$$

$$\delta(A_{j_p}, A_{j_{p+1}}) = 1, \quad p = \overline{1, t-1}.$$

Очевидно, всякая укладка L может быть представлена в виде последовательности компонент $L = (C_1, C_2, \dots, C_q)$. Будем называть C_i компонентой i -го ранга. Отметим, что если q — число компонент в укладке L , то $\kappa(L) = n - q$.

Число соседней связности графа G — величина $\kappa(L) = \max_{L \in R(G)} \{\kappa(L)\}$,

где $R(G)$ — множество всех упадок графа G .

Задача заключается в определении оптимальной укладки $L_* \in R(G)$, для которой $\kappa(L_*) = \kappa(G)$.

Метод решения. Структура рассматриваемой задачи позволяет определить точное решение оптимальной укладки графа G программы с

помощью одного из «ветвящихся» методов решения комбинаторных задач (динамического программирования, ветвей и границ, последовательного конструирования, анализа и отсеивания вариантов). Однако при использовании точных методов вычислительная сложность и требования к памяти растут экспоненциально с увеличением размерности задачи. Поэтому для решения практических задач укладки графов программ целесообразно применять эвристические методы. Дадим следующие определения.

Дуга $(A_i, A_j) \in U$ называется *избыточной*, если в графе G существует хотя бы один путь μ из $A_i \in X$ в A_j длиной $l(\mu) \geq 2$. Здесь и далее под длиной $l(\mu)$ пути μ следует понимать количество дуг на этом пути.

Назовем *A-преобразованием графа* G граф $G' = A(G)$, полученный удалением из G всех избыточных дуг.

Путь $\mu = (A_{i_1}, \dots, A_{i_p})$ графа G является *проникающим*, если:

а) $d^-(A_{i_1}) = 0$; б) в графе G отсутствуют дуги вида (A_u, A_{i_q}) , где $q = \overline{1, p}$, $A_u \in X \setminus \bigcup_{q=1}^p A_{i_q}$.

Проникающий путь $\mu = (A_{i_1}, \dots, A_{i_p})$ будет *максимальным*, если не найдется такой вершины A_u , что путь $\mu' = (A_{i_1}, \dots, A_{i_p}, A_u)$ также будет проникающим.

Предлагаемый ниже алгоритм 6.1 укладки графа основан на отыскании максимальных проникающих путей.

А л г о р и т м 6.1. Начать с графа $G = A(G)$, где A — символ A -преобразования.

1. На k -м шаге в графе G_k определяется множество M_k проникающих путей и среди них выбирается путь $\mu_k^* \in M_k$, имеющий максимальную длину.

2. Из графа G_k удаляется путь μ_k^* и образуется новый граф $G_{k+1} = G_k \setminus \mu_k^*$.

Алгоритм заканчивается, когда на некотором шаге $G_m = \emptyset$.

Полученные на каждом шаге пути μ_k^* отождествляются с компонентами k -го ранга S_k искомой укладки. Тогда последовательность $\Delta = (\mu_1^*, \dots, \mu_{m-1}^*)$ представляет собой укладку графа G , определенную с помощью алгоритма 6.1.

При определении максимального проникающего пути μ_k^* удобно использовать алгоритм 6.2, который основан на пометке вершин графа G_k .

А л г о р и т м 6.2. Первоначально все вершины графа $G_k = (X_k, U_k)$ считаются непомеченными и непросмотренными.

1. В графе G_k входные вершины A_X , у которых $d^-(A_X) = 0$, получают метку $\lambda(A_X) = 1$. После этого вершины A_X считаются помеченными и непросмотренными.

2. Пусть A_X — некоторая помеченная и непросмотренная вершина в графе G_k . Рассмотрим множество вершин $Y = \{A_Y | (A_X, A_Y) \in U_k\}$. Если $d^-(A_Y) = 1$, то вершина $A_Y \in Y$ получает метку $\lambda(A_Y) = \lambda(A_X) + 1$ и считается помеченной и непросмотренной. Если $d^-(A_Y) > 1$, то вершина не помечается. После анализа всех $A_Y \in Y$ вершин A_X считается просмотренной. Процесс пометки заканчивается, когда все помеченные вершины просмотрены.

3. В графе G_k выбирается вершина A_{i_p} , для которой величина метки $\lambda(A_{i_p})$ максимальна. Тогда путь $\mu_k^* = (A_{i_1}, \dots, A_{i_p})$ с последовательно возраста-

тающими метками вершин будет являться искомым максимальным проникающим путем.

Алгоритм 6.2 решает задачу с оценкой $O(n^2)$ действий.

Описанный ниже алгоритм 6.3 является усложнением алгоритма 6.1 и позволяет во многих случаях повысить точность получаемых решений за счет введения операции прогноза на один шаг.

А л г о р и т м 6.3. Начать с графа $G_1 = A(G)$.

1. В графе G_h определяется множество проникающих путей M_h .

2. Для каждого пути $\mu \in M_h$ образуется граф $G_h(\mu) = G_h \setminus \mu$ и в этом графе отыскивается максимальный проникающий путь $\pi_k^*(\mu)$.

3. Среди всех путей $\mu \in M_h$ выбирается путь μ_k^* , для которого максимальна величина $l(\mu_k^*) + l(\pi_k^*(\mu_k^*))$, где $l(\mu_k^*)$ и $l(\pi_k^*(\mu_k^*))$ — длины соответствующих путей.

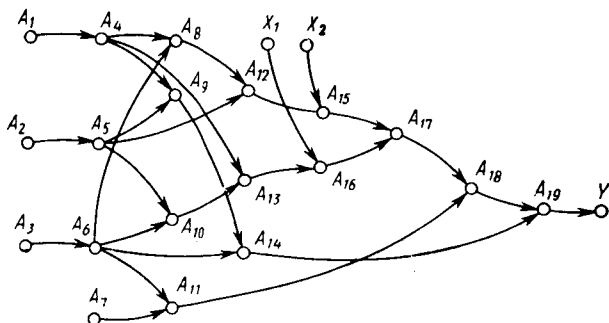


Рис. 6.5. Граф алгоритма статистической интерполяции

После этого образуется новый граф $G_{h+1} = G_h \setminus \mu_k^*$. Описанная процедура заканчивается на шаге m , для которого $G_m = \emptyset$.

Аналогично могут быть построены эвристические алгоритмы укладки графов с прогнозом на большее число шагов, однако при этом резко возрастает их трудоемкость.

Отметим, что алгоритм 6.1 дает точное решение для укладки программ, являющихся прадеревьями с одним корнем.

Проиллюстрируем методику оптимизации алгоритмов функционирования МПУ по критерию минимума числа обращений к оперативной памяти на следующем примере.

Пример 6.2. В микропроцессорных РЭС обработки сигналов (радиолокационных и др.) широко применяется алгоритм статистической интер- или экстраполяции по двум точкам, реализующий закон

$$y(t) = \sum_{i=1}^n p_i(t - t_i)[x^*(t_i) - E(x)] + E(x),$$

где $p_i(t - t_i)$ — коэффициенты многочлена, определяемые из условия минимизации погрешности интер- или экстраполяции; $E(x)$ — м.о. величины x , $n = 2$.

Граф G алгоритма статистической интерполяции представлен на рис. 6.5. В этом графе X_1 и X_2 — входные сигналы ($t_{X_1} < t_{X_2}$), Y —

выходной сигнал ($t_Y > t_{X_2}$), вершины $A_1, \dots, A_{19}$ соответствуют функциональным операторам, семантический смысл которых приведен в табл. 6.3, а дуги графа — ограничениям порядка выполнения операторов.

Т а б л и ц а 6.3

Семантика функциональных операторов алгоритма
статистической интерполяции

Опера- тор	Выполняемое действие	Опера- тор	Выполняемое действие
A_1	$y_1 = t - t_1$	A_{11}	$y_{11} = y_7 - k_x^2 (t_2 - t_1) = y_7 - y_8^2$
A_2	$y_2 = t - t_2$	A_{12}	$y_{12} = D_x k_x (t - t_2) - y_8 = D_x y_5 - y_8$
A_3	$y_3 = t_2 - t_1$	A_{13}	$y_{13} = D_x y_4 - y_{10}$
A_4	$y_4 = k_x (t - t_1) = k_x (y_1)$	A_{14}	$y_{14} = y_9 / D_x + y_6$
A_5	$y_5 = k_x (t - t_2) = k_x (y_2)$	A_{15}	$y_{15} = y_{12} x(t_2)$
A_6	$y_6 = k_x (t_2 - t_1) = k_x (y_3)$	A_{16}	$y_{16} = y_{13} x(t_7)$
A_7	$y_7 = D_x D_x, D_x = k_x (0)$	A_{17}	$y_{17} = y_{16} + y_{15}$
A_8	$y_8 = y_4 y_6$	A_{18}	$y_{18} = y_{17} / y_{11}$
A_9	$y_9 = y_4 + y_5$	A_{19}	$y_{19} = x^*(t) = y_{18} - m_x (y_{14} - 1)$
A_{10}	$y_{10} = y_5 y_6$		

Оптимальная укладка графа G определяется по алгоритму 6.3 таким образом.

Выполнение программы можно начать с любого входного оператора A_1, A_2 или A_3 , но анализ прогноза на первом шаге показывает, что более перспективным является начало выполнения программы с оператора A_1 , так как далее появляется возможность реализации последовательности операторов (A_2, A_5, A_9) длиной $l(\mu) = 3$, а на третьем шаге — длиной $l(\mu) = 5$.

Оптимальная укладка графа G имеет вид (в укладке вместо операторов A_i стоят их номера i)



Стрелки обозначают отношение соседней связности между вершинами в укладке $L_*(G)$.

Значение числа соседней связности, соответствующее укладке $L(G)$, $\kappa(G) = 13$.

Физический смысл любой последовательности $(\overbrace{j \rightarrow k}, \dots, \overbrace{n})$

поясним на примере упорядоченной последовательности операторов

(2, 5, 9). Стрелки означают, что результирующая величина после выполнения оператора A_2 требуется для реализации оператора A_5 и хранится только в аккумуляторе АЛУ, так как у оператора A_2 нет информационных связей с другими операторами. Результат выполнения оператора A_5 не только хранится в аккумуляторе АЛУ для выполнения оператора A_9 , но и засылается в один из РОНов для информационной поддержки выполнения операторов A_{10} , A_{12} . После выполнения оператора A_9 укладка программы обрывается, так как следующий далее в графе G оператор A_{14} выполняться не может из-за отсутствия результата выполнения оператора A_6 . Отсутствие стрелки между операторами A_9 и A_3 свидетельствует о том, что для выполнения оператора A_3 требуется освобождение аккумулятора АЛУ и обращение к памяти (ОЗУ или СОЗУ) для записи результата выполнения оператора A_9 .

На третьем шаге возможны два варианта: выполнение последовательностей $\mu_{31} = (3, 6, 10, 13, 16)$ или $\mu_{32} = (3, 6, 14)$, где μ_{3i} означает i -й вариант укладки на третьем шаге. При этом получается, что $l(\mu_{31}) = 5$, $l(\mu_{32}) = 3$. Поскольку $l(\mu_{31}) > l(\mu_{32})$, выбирается максимальный проникающий путь $\mu_{31} = \mu_3$. Решение задачи оптимизации программы заканчивается выбором последней нереализованной укладки $\mu_6 = (7, 11, 18, 19)$. Отсутствие пяти стрелок между парами операторов (4,2), (9,3), (16,8), (17,14) и (14,7) в укладке графа $L_*(G)$ говорит о том, что результирующие значения пяти операторов A_4 , A_9 , A_{16} , A_{17} , A_{14} обязательно должны быть записаны в память ОЗУ или СОЗУ и после их запоминания аккумулятор АЛУ должен быть обнулен.

§ 6.4. ОПТИМИЗАЦИЯ ЗАГРУЗКИ СВЕРХОПЕРАТИВНОЙ ПАМЯТИ

Оптимизация загрузки СОЗУ (программно-доступных РОНов) заключается в определении минимально необходимого объема СОЗУ, порядка реализации операторов и распределения данных по регистрам СОЗУ, обеспечивающего минимальное число обращений к ОЗУ в процессе реализации алгоритма функционирования МПУ. В структурном плане СОЗУ предназначено для хранения наиболее часто используемых данных, в частности промежуточных результатов вычислений, некоторых констант, операндов и т. д. Количество обращений к СОЗУ зависит от его объема, класса решаемых задач и способа организации загрузки. Сверхоперативная память вообще слишком мала, чтобы ее объем существенно сказывался на общем объеме памяти МПУ. Однако эффективная организация загрузки СОЗУ позволяет добиться значительного увеличения производительности реализации алгоритмов.

Развитие техники больших интегральных схем позволило организовать в структурах микропроцессоров сверхоперативное ЗУ (программно-доступные РОН) довольно значительного объема, что дало возможность организовать реализацию программ работы МПУ практически без обращения к ОЗУ с целью чтения и записи исходных и промежуточных данных, констант и др.

Постановка задачи. Задача оптимизации загрузки СОЗУ заключается в определении минимально необходимого числа программно-доступных РОН для хранения промежуточных результатов и данных, определении порядка реализации операторов и организации такой загрузки регистров, чтобы количество обращений АЛУ к оперативной памяти в процессе реализации алгоритма функционирования МПУ было минимальным.

Для решения этой задачи требуется исходная информация:

функциональный граф системы G (или матрица смежности графа α) с заданными множествами X входных и Y выходных величин;

множество $M = \{m_1, \dots, m_r\}$ операндов и констант, необходимых для реализации алгоритма функционирования МПУ.

В качестве критерия оптимальности естественно выбрать критерий минимизации числа обращений к ОЗУ в процессе реализации алгоритма (критерий максимальной интенсификации загрузки СОЗУ).

Метод решения. Идея оптимизации загрузки СОЗУ заключается в следующем. Для интенсификации загрузки СОЗУ промежуточными результатами вычислений в процессе реализации алгоритма нужно максимизировать число обращений АЛУ к СОЗУ и тем самым минимизировать число обращений АЛУ к ОЗУ. С другой стороны, для своевременного обеспечения вычислительного процесса операндами и константами следует заранее загрузить регистры СОЗУ необходимой информацией из ОЗУ (или из ПЗУ).

Поскольку поставщиками информации для СОЗУ являются различные устройства, информационная связь которых осуществляется с СОЗУ по самостоятельным трактам АЛУ — СОЗУ и ОЗУ — СОЗУ или ПЗУ — СОЗУ, задачу оптимизации общей загрузки СОЗУ целесообразно свести к задаче интенсификации загрузки соответствующих информационных трактов.

Объем сверхоперативной памяти настолько мал, что говорить о ее разделении между различными программами и подпрограммами работы вообще не имеет смысла. Однако при реализации каждой конкретной программы оказывается возможным выделить в СОЗУ двух непересекающихся зон: 1) для хранения промежуточных результатов, поступающих из АЛУ; 2) для хранения операндов и констант, поступающих из ОЗУ (или ПЗУ). Очевидно, при такой стратегии использования СОЗУ следует осуществлять прямой доступ к его ячейкам. Порядок использования объема СОЗУ определяется в процессе разработки системы программирования МПУ и осуществляется операционной системой МПУ.

Решение задачи проводится поэтапно.

Первоначально определяется последовательность операторов $L_0 = (A_1^{(i)}, \dots, A_n^{(in)})$, обладающая оптимальной укладкой по критерию максимальной загрузки СОЗУ промежуточными результатами. Запись $A_k^{(i_k)}$ означает, что оператор A_k в последовательности L_0 находится на i_k -й позиции.

Для выполнения алгоритма L должно быть задано отображение $\Gamma: A \rightarrow M$, где $\Gamma(A_k) = M_k \subseteq M$ — множество данных, требующихся для выполнения оператора A_k . Очевидно, в общем случае множество $M_k \cap M_l \neq \emptyset$, $k, l = \overline{1, n}$, а мощность этого множества $|M_k| = C_k$.

Назовем сечением C_s в последовательности $L = (A_1, \dots, A_s, A_{s+1}, \dots, A_n)$ совокупность информационных передач от операторов $A_1, \dots, A_s$ к операторам $A_{s+1}, \dots, A_n$.

Очевидно, на каждом шаге s выполнения алгоритма L возникает необходимость в хранении множества промежуточных результатов выполнения операторов $A_i^{(k)} \in L_1$, $k \leq s$, требующихся для реализации частичного алгоритма $L_2 \subset L$. При этом число промежуточных данных b_s в сечении C_s определяется количеством информационных связей между различными $A_i^{(k)} \in L_1$ и $A_j^{(l)} \in L_2$ ($L_1 \cup L_2 = L$) при условии, что результирующие величины $A_i^{(k)} \in L_1$ использует хотя бы один потребитель $A_j^{(l)} \in L_2$.

Минимальный объем СОЗУ $E_{\text{СОЗУ}}$, позволяющий осуществить реализацию алгоритма L без обращений за операндами к ОЗУ и промежуточными данными в процессе вычислений, определяется неравенством

$$E_{\text{СОЗУ}} \geq b_0 + e_0,$$

где b_0 — минимально необходимое число регистров СОЗУ для хранения промежуточных результатов вычислений, определяемое посредством описанной выше сетевой модели; $e_0 = \max_{A_k \in L} \{C_k\}$ — максималь-

ное число операндов и констант, требующихся для обеспечения выполнения операторов $A_k \in L$, $k = \overline{1, n}$.

Кроме того, в СОЗУ целесообразно бронировать часть регистров для хранения базовых адресов и индексов, которые многократно используются при обработке команд и при этом часто модифицируются путем выполнения над ними арифметических операций.

К сожалению, не всегда оказывается возможным обеспечить в структурах ММ достаточный для реализации алгоритмов функционирования МПУ объем СОЗУ. Поэтому в каждой конкретной разработке аппаратной части микропроцессорных РЭС и их программного обеспечения требуется тщательная проработка моделей и методов, позволяющих определять оптимальное (или рациональное) соотношение между объемами зон СОЗУ для хранения промежуточных результатов (b), операндов и констант (e) и их загрузку.

Поскольку обмен информацией между устройствами АЛУ — ОЗУ и СОЗУ — ОЗУ осуществляется по отдельным информационным трактам, с целью минимизации общего времени простоя АЛУ (из-за ожидания данных из ОЗУ) нужно осуществлять по возможности равномерную загрузку этих трактов. Поэтому объемы b и e зон СОЗУ должны быть пропорциональны количеству промежуточных результатов, а также операндов и констант соответственно.

С целью минимизации числа обращений к ОЗУ за промежуточными данными предварительно следует провести синтез алгоритмов функционирования МПУ по критерию максимальной соседней связности.

Пусть $\kappa(L)$ — число соседней связности алгоритма L . Тогда число обращений за промежуточными результатами в процессе реализации алгоритма L составит $\sum_{i=1}^{n-1} f_i - \kappa(L)$, где f_i — количество промежуточных результатов, получаемых после выполнения оператора $A_i \in L$. Из соотношения

$$\frac{\sum_{i=1}^{n-1} f_i - \kappa(L)}{r} = \frac{b}{e}$$

видно, что

$$b = \frac{\sum_{i=1}^{n-1} f_i - \kappa(L)}{\sum_{i=1}^{n-1} f_i - \kappa(L) + r} E'_{\text{СОЗУ}}, \quad e = \frac{r}{\sum_{i=1}^{n-1} f_i - \kappa(L) + r} E'_{\text{СОЗУ}}. \quad (6.10)$$

В простейшем случае, когда каждый оператор $A_i \in L$ выдает только по одному промежуточному результату, величины объемов b и e определяются соотношениями

$$b = \frac{n-1-\kappa(L)}{n-1-\kappa(L)+r} E'_{\text{СОЗУ}}, \quad e = \frac{r}{n-1-\kappa(L)+r} E'_{\text{СОЗУ}}. \quad (6.11)$$

В связи с тем что в каждую зону СОЗУ информация поступает по самостоятельным информационным трактам, а ячейки каждой зоны сверхоперативной памяти предназначены для хранения различной информации, оказывается корректной раздельная оптимизация загрузки этих зон по критерию минимума числа обращений к ОЗУ в процессе реализации алгоритма L .

Оптимизация загрузки СОЗУ промежуточными результатами. Исходной информацией для решения задачи оптимизации загрузки СОЗУ промежуточными результатами является последовательность операторов $L_0 = (A_1, \dots, A_n)$, характеризующаяся максимальным числом соседней связности. Последовательности L_0 соответствует матрица смежности $\alpha = ||\alpha_{ij}||$ размерности $n \times n$.

Задачей проектирования являются определение минимально необходимого объема b_0 СОЗУ, обеспечивающего реализацию алгоритма L_0 без обращений к ОЗУ (исключая обращения к начальным данным), и организация загрузки регистров СОЗУ.

Из матрицы α образуем новую матрицу $\tilde{\alpha} = ||\tilde{\alpha}_{ij}||$, элементы которой определяются по правилу

$$\tilde{\alpha}_{ij} = \begin{cases} 1, & \text{если } j \geq \delta_i, \\ 0 & \text{— в противном случае,} \end{cases} \quad (6.12)$$

где δ_i — номер такого столбца матрицы α , что

$$(\alpha_{i\delta_i} = 1) \ \& \ [(\forall \gamma > \delta_i) (\alpha_{i\gamma} = 0)] = 1.$$

Матрица $\tilde{\alpha}$ является верхней треугольной, и ее можно рассматривать как матрицу смежности нового графа $\tilde{G} = (\tilde{I}, \tilde{U})$, в котором дуги $(i, j) \in \tilde{U}$. Элементы $\tilde{\alpha}_{ij} = 1$ i -строки матрицы $\tilde{\alpha}$ указывают операторы A_j , результирующие величины $Y(A_j)$ которых можно отсылать на хранение в тот же самый регистр СОЗУ, в котором хранилась результирующая величина $Y(A_i)$.

Назовем *активностью* оператора A_i величину $d_i = \sum_{j \geq i} \alpha_{ij}$, показывающую, какое число раз используется результирующая величина $Y(A_i)$ при выполнении алгоритма функционирования МПУ, и поместим каждую вершину $i \in \tilde{I}$ графа $\tilde{G}$ числом d_i .

Очевидно, что в графе $\tilde{G}$ каждому пути $\mu_k = (k_1, k_2, \dots, k_{s(k)})$, где $s(k)$ — число вершин в пути μ_k , соответствует распределение загрузки $N(\rho_k) = (Y_{k_1}, Y_{k_2}, \dots, Y_{k_{s(k)}})$ регистра ρ_k СОЗУ результирующими величинами операторов A_{i_q} в порядке, определяемом последовательностью $N(\rho_k)$. При этом величина

$$\lambda(\rho_k) = \sum_j d_j = \sum_{t=1}^{s(k)} d_{k_t} \quad (6.13)$$

определяет загрузку регистра ρ_k . Величина $\lambda(\rho_k)$ соответствует числу обращений АЛУ к регистру ρ_k в процессе реализации алгоритма функционирования МПУ.

Вычеркнем из матрицы $\tilde{\alpha}$ все i -е строки и столбцы, для которых $d_i = 0$, после чего получим матрицу $\alpha^* = ||\alpha_{ij}^*||$ размерности $t \times t$ ($t < n$) и соответствующий ей граф $G^* = (I^*, U^*)$. Для решения задачи расширим граф $G^* = (I^*, U^*)$ до графа $G_1^* = (\tilde{I}_1, \tilde{U}_1)$ путем добавления к G^* двух вершин i_0, i_z и всех дуг $(i_0, j), (j, i_z), j \in I^*$.

В сформулированном виде задача оптимизации загрузки СОЗУ промежуточными результатами сводится к определению наименьшего числа непересекающихся по вершинам путей $\mu = \{\mu_1, \dots, \mu_b\}$, ведущих из вершины i_0 через смежные вершины $j \in G^* \cap G_1^*$ в вершину

i_z , покрывающих все вершины графа G_1^* ровно по одному разу (кроме вершин i_0 и i_z).

Величина b_0 является искомым минимально необходимым количеством регистров СОЗУ, а каждый путь $\mu_k \in \mu$ определяет распределение загрузки $N(\rho_k)$ регистра ρ_k , $k = \overline{1, b_0}$.

В случае, если объем СОЗУ $b < b_0$, то задача определения оптимальной загрузки СОЗУ может быть сформулирована следующим образом: в графе G^* найти b непересекающихся по вершинам путей $\mu_1, \dots, \mu_b$, таких, чтобы величина

$$\lambda_b = \sum_{k=1}^b \lambda(\rho_k) = \sum_{\substack{j \in U \mu_k \\ k=1}}^b d_j \quad (6.14)$$

достигала максимального значения. При этом число обращений к ОЗУ будет минимальным.

Оптимальную загрузку регистров СОЗУ можно найти, решив задачу определения минимального числа непересекающихся по вершинам путей, покрывающих все вершины графа.

Рис. 6.6. Матрица смежности графа алгоритма статистической интерполяции

Методику организации оптимальной загрузки СОЗУ промежуточными результатами проиллюстрируем на примере.

Пример 6.3. Определить минимально необходимый объем СОЗУ и оптимальное распределение его регистров для хранения результатов значений операторов при реализации алгоритма статистической интерполяции по двум точкам, граф G которого изображен на рис. 6.5.

Для решения задачи представим реализуемый алгоритм в виде упорядоченной последовательности операторов $L = (A_1, A_4, A_2, A_5, A_9, A_3, A_6, A_{10}, A_{13}, A_{16}, A_{14}, A_7, A_{11}, A_8, A_{12}, A_{15}, A_{17}, A_{18}, A_{19})$. Число соседней связности данной последовательности $\kappa = 13$. Это значит, что результаты, хранимые в аккумуляторе АЛУ, могут быть использованы для выполнения последующих операторов 13 раз (результаты выполнения операторов $A_1, A_2, A_5, A_3, A_6, A_{10}, A_{13}, A_7, A_8, A_{12}, A_{15}, A_{17}, A_{18}$).

Матрица смежности α графа G (без учета входных и выходных сигналов) приведена на рис. 6.6. В матрице α по строкам и столбцам вместо операторов A_i стоят их индексы i .

При определении столбца активностей операторов d_i не учитываются информационные связи вида $A_{i_k} \rightarrow A_{i_{k+1}}$, которым соответст-

вуют элементы $\alpha_{i_k i_{k+1}} = 1$, поскольку в этом случае результирующая величина оператора A_{i_k} хранится в аккумуляторе АЛУ.

Сокращая размерность задачи путем удаления из матрицы α строк и столбцов, имеющих активности $d_i = 0$, и используя выражение (6.12), переходим к модифицированной матрице $\tilde{\alpha}$ (рис. 6.7, а) и соответствующему ей графу G^* с введенными вершинами i_0 и i_z (рис. 6.7, б).

Следует отметить, что суммарная активность операторов $\lambda = \sum_j d_j = 12$. Это значит, что в процессе реализации алгоритма АЛУ

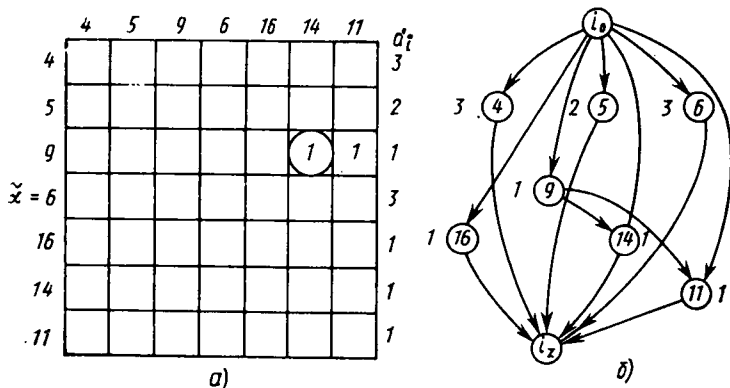


Рис. 6.7. Модифицированная матрица (а) и граф алгоритма статистической интерполяции (б)

обратится к памяти 12 раз ($\lambda = 12$), а к своему аккумулятору — 13 раз ($\kappa = 13$), причем суммарное число обращений за промежуточными результатами $\lambda + \kappa = 25$, что соответствует числу дуг в функциональном графе или числу единиц в матрице α .

Наименьшее число непересекающихся по вершинам путей в графе G^* составляет 6, следовательно, минимально необходимое число регистров СОЗУ, обеспечивающих выполнение алгоритма без обращения к ОЗУ, $b_0 = 6$. Этими путями являются: $\mu_1 = (i_0, 4, i_z)$; $\mu_2 = (i_0, 5, i_z)$; $\mu_3 = (i_0, 9, 14, i_z)$; $\mu_4 = (i_0, 6, i_z)$; $\mu_5 = (i_0, 11, i_z)$; $\mu_6 = (i_0, 16, i_z)$.

Эти же пути можно получить непосредственно из матрицы α следующим образом.

Находим верхнюю строку, для которой $d_i > 0$ (строка 4, соответствующая оператору A_4). Результирующая величина Y_4 оператора A_4 может храниться в одном из регистров СОЗУ до начала выполнения оператора, соответствующего крайней правой единице в строке 4. Им является оператор A_8 , но так как $d_8 = 0$, то надобности в хранении величины Y_8 в этом регистре нет. Аналогичная ситуация наблюдается и в строке 5. Анализ строки 9 показывает, что после выполнения

оператора A_{14} ($d_{14} = 1$) в регистре СОЗУ, в котором хранилась величина Y_9 , может храниться величина Y_{14} до конца реализации алгоритма и т. д.

Оптимальное распределение загрузки регистров при $b_0 = 6$ имеет вид

$$\begin{aligned} N(\rho_1) &= (Y_4), \lambda(\rho_1) = 3; & N(\rho_4) &= (Y_6), \lambda(\rho_4) = 3; \\ N(\rho_2) &= (Y_5), \lambda(\rho_5) = 2; & N(\rho_5) &= (Y_{11}), \lambda(\rho_5) = 1; \\ N(\rho_3) &= (Y_9, Y_{14}), \lambda(\rho_3) = 2; & N(\rho_6) &= (Y_{16}), \lambda(\rho_6) = 1. \end{aligned}$$

Пусть $b = 3$. Тогда оптимальное распределение загрузки регистров СОЗУ дадут три непересекающихся по вершинам пути, имеющих наибольшую суммарную активность:

$$\begin{aligned} N(\rho_1) &= (Y_4), \lambda(\rho_1) = 3; \\ N(\rho_2) &= (Y_6), \lambda(\rho_2) = 3; \\ N(\rho_4) &= (Y_9, Y_{14}), \lambda(\rho_4) = 2, \end{aligned}$$

или $N(\rho_3) = (Y_5), \lambda(\rho_3) = 2;$

$$\lambda_b = \sum_{i=1}^b \lambda(\rho_i) = 8.$$

Это означает, что при реализации алгоритма статистической интерполяции по двум точкам при наличии в структуре ММ СОЗУ объемом $b = 3$ ячейки к ОЗУ будет произведено всего четыре обращения за промежуточными результатами (величинами $Y_9, Y_{14}, Y_{11}, Y_{16}$ или Y_5 (2 раза), Y_{11} , и Y_{16}).

Оптимизация загрузки СОЗУ операндами и константами. Для определения точности решения задачи загрузки зоны e СОЗУ, предназначенной для хранения операндов и констант, обоснуем оценки числа обращений к ОЗУ при реализации алгоритма **L**.

При наличии в структуре ММ зоны СОЗУ объемом e_0 ($e_0 = r$) обращения к ОЗУ в процессе выполнения алгоритма **L** отсутствуют ($N_{e_0} = 0$), а верхняя оценка числа обращений $\bar{Q}_{e_0}$ устройства обмена (УО) к ОЗУ для загрузки СОЗУ определяется выражением

$$\bar{Q}_{e_0} = \sum_{k=1}^r q_k - \sum_{k=1}^r \sum_{p=1}^{q_k} (L_k^{(p)} - 1) \xi_k^{(p)},$$

где q_k — количество обращений к операнду или константе m_k ($q_k \geq 1$); $L_k^{(p)}$ — количество операторов в алгоритме **L**, находящихся в отношении соседней связности и являющихся потребителями величины m_k ; $\xi_k^{(p)}$ — количество цепочек соседне-связанных операторов длиной $L_k^{(p)}$.

При реализации алгоритмов функционирования МПУ число обращений УО к памяти можно уменьшить по сравнению с $\bar{Q}_{e_0}$ за счет действия незагруженных регистров. Действительно, если при вы-

полнении оператора $A_{i+\xi}$, $1 \leq \xi < z$ оказывается, что $c_{i+\xi} < e_0$ и справедливо условие

$$(m_k \in \Gamma(A_i) \& m_k \in \Gamma(A_{i+z}) (m_k \notin \Gamma(A_{i+\xi})) = 1, \quad 1 \leq \xi < z, \quad (6.15)$$

то с целью минимизации числа обращений УО к ОЗУ оказывается возможным содержимое регистра СОЗУ, в котором при выполнении оператора A_i хранилась величина m_k , сохранять вплоть до начала реализации оператора A_{i+z} . При этом количество обращений УО к ОЗУ будет составлять $\bar{Q}_{e_0}' = \bar{Q}_{e_0} - h$, где h — количество групп операторов, удовлетворяющих условию (6.15). Эти группы операторов определяются из таблицы, в которой задано отображение $\Gamma: A \rightarrow M$.

В случае, если объем зоны СОЗУ для хранения операндов и констант $e < e_0$, то нижней оценкой $\underline{N}_e$ числа обращений ОУ к ОЗУ в процессе выполнения алгоритма L будет величина

$$\underline{N}_e = \sum_{\forall i: c_i > e} (c_i - e).$$

Верхняя и нижняя оценки $\bar{Q}_e$ и $\underline{Q}_e$ числа обращений УО к ОЗУ при организации загрузки СОЗУ определяются соотношениями

$$\bar{Q}_e = \sum_{k=1}^r q_k \dots \underline{N}_e, \quad \underline{Q}_e = \bar{Q}_e - \sum_{k=1}^r \sum_{p=1}^{q_k} \xi_k^{(p)} (L_k^{(p)} - 1).$$

Очевидно, что $\bar{Q}_e \geq Q_e \geq \underline{Q}_e$ и для упрощения структуры и алгоритма функционирования УО следует стремиться, чтобы $Q_e \rightarrow \underline{Q}_e$.

Рассмотрим алгоритм 6.4 определения рациональной загрузки зоны СОЗУ для хранения операндов и констант. Работа алгоритма основана на последовательном определении цепочек соседне-связанных операторов максимальной длины, являющихся потребителями одной и той же величины m_k , $k = \overline{1, r}$, и построении сети специального вида, содержащей e непересекающихся путей из источника сети в сток. При этом каждый из e путей определяет загрузку одного из e регистров СОЗУ.

А л г о р и т м 6.4.

1. Формирование множества вершин I сети $G = (I, U)$. Множество I состоит из источника сети i_0 , стока i_z и n ярусов $D_1, \dots, D_n$. Каждый ярус D_k соответствует множеству операндов и констант, необходимых для выполнения оператора A_k , и поэтому содержит $e = |M_k|$ вершин ($e_k \leq e_0$).

2. Раскраска вершин сети $G = (I, U)$. Каждому операнду и константе m_k присваивается свой цвет, в который раскрашиваются все вершины $i \in I$, соответствующие данной величине m_k . Таким образом, все множество вершин I раскрашивается в r цветов, а вершины яруса D_k — в e_k цветов.

3. Пометка вершин сети $G = (I, U)$ индексами. Индекс φ_i у вершины $i \in D_k$ характеризует увеличенное на единицу количество соседних ярусов $D_{k+1}, \dots, D_{k+\varphi_i-1}$, содержащих одинаково окрашенные с $i \in D_k$ вершины.

Пометка вершин начинается с яруса D_n . Всем вершинам $i_k \in D_n$, $k = \overline{1, c_n}$, присваивается индекс $\varphi_{i_k} = 1$.

Вершине $i_k \in D_p$, $1 \leq p < n$, присваивается индекс

$$\varphi_{i_k} = \begin{cases} \varphi_{i_l} + 1, & \text{если, } i_l \in D_{p+1} \text{ и } i_k \in D_p, \text{ окрашены в один цвет;} \\ 1 & \text{— в противном случае.} \end{cases}$$

4. Формирование цепочки дуг, соединяющей множество соседне-связанных одноцветных вершин максимальной длины.

Отыскивается вершина $i \in D_p$, не входящая ни в одну из цепочек и имеющая индекс $\varphi_i = \max_{j \in I} \{\varphi_j\}$. Вершина $i \in D_p$ соединяется дугой (i, j) с одноцветной вершиной $j \in D_{p+1}$, $\varphi_j = \varphi_i - 1$.

Процедура формирования множества дуг, связывающих одноцветные вершины $i \in D_p$, $j \in D_{p+1}, \dots$, заканчивается образованием дуги (x, y) , $x \in D_{p+\varphi_i-2}$, $y \in D_{p+\varphi_i-1}$, $\varphi_y = 1$.

5. Поярусная проверка числа вершин, инцидентных сформированным дугам.

Если в ярусе D_p число вершин, инцидентных сформированным дугам, равно e , то ярус D_p из дальнейшего рассмотрения исключается, а цифровые индексы φ_i у свободных вершин оставшихся ярусов переобозначаются согласно п.3.

Формирование цепочек дуг заканчивается, когда либо все ярусы исключены из рассмотрения, либо в оставшихся ярусах не осталось свободных вершин.

6. Формирование множества непересекающихся по вершинам путей.

6.1. Вершины $i \in D_1$, для которых $\varphi_i > 1$, включаются в множество $D' \subseteq D_1$. Проводятся дуги вида (i_0, i) . Если $|D'| < e$, то производится переход к п.6.2, в противном случае — к п. 6.3.

6.2. Выбирается $e - |D'|$ (при $e < c_1$) произвольных вершин $j \in D_1$, для которых $\varphi_j = 1$, и проводятся дуги вида (i_0, j) .

6.3. Все вершины $i \in D_p$, $1 < p \leq n$, для которых $|U_i^+| = 1$ и $|U_i^-| = 0$, и все вершины $j \in D_{p+1}$, для которых $|U_j^+| = 0$ и $|U_j^-| = 1$, связываются дугами (i, j) .

6.4. Все вершины $i \in D_n$, для которых $|U_i^+| = 1$, связываются дугами (i, i_z) со стоком i_z .

7. Конец алгоритма.

Из логики построения алгоритма 6.4 видно, что в качестве критерия оптимальности при организации загрузки СОЗУ операндами и константами выбран критерий максимальной связности одноцветных вершин, находящихся в соседних ярусах. Для рассматриваемой сетевой модели проектирования это эквивалентно минимизации числа обращений УО к ОЗУ для осуществления загрузки СОЗУ при условии, что число обращений АЛУ к ОЗУ определяется величиной $\frac{N_e}{e}$.

Полученное решение задачи можно улучшить, если выполняется условие (6.15) и имеется возможность задействования незагруженных регистров СОЗУ. При этом нижняя оценка Q_e может быть уменьшена.

Рассмотренную методику организации загрузки СОЗУ проиллюстрируем на примере.

Пример 6.4. Определить минимально необходимый объем зоны СОЗУ для хранения операндов и констант и оптимальную загрузку ее регистров при реализации алгоритма $L = (A_1, \dots, A_{10})$. Известно, что для выполнения алгоритма L требуется множество данных $M = \{m_1, m_2, m_3, m_4\}$, отображение $\Gamma: A \rightarrow M$ задано табл. 6.4.

Из табл. 6.3. видно, что $e_0 = \max_k \{c_k\} = 3$. Оценки числа обращений к ОЗУ: $N_z = 0$, $\bar{Q}_3 = 10$.

В табл. 6.5 приведено оптимальное распределение загрузки трех регистров СОЗУ (здесь m_k^* обозначен момент засылки данного m_k из ОЗУ в СОЗУ, а m_k^* — дополнительная загрузка свободного регистра данным m_k). Условие (6.15) выполняется для трех групп операторов ($h = 3$), поэтому $\bar{Q}_3 = 7$. Действительное число обращений к ОЗУ $Q_3 = 7$.

Таблица 6.4

Данные, требуемые для выполнения алгоритма

m_k	A_i									
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
m_1	×	×			×	×	×	×		×
m_2		×	×	×	×				×	×
m_3	×			×			×	×	×	
m_4			×	×	×					×

Таблица 6.5

Оптимальное распределение загрузки трех регистров СОЗУ

Pr_i	A_i									
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
1	m'_1	m_1	$+m'_4$	m_4	m_4		$+m'_3$	m_3	m_3	$+m'_4$
2		m'_2	m_2	m_2	m_2	m_2^+	m_2^+	m_2^+	m_2	m_2
3	m'_3	m_3^+	m_3^+	m_3	$+m'_1$	m_1	m_1	m_1	m_1^+	m_1

Для зоны СОЗУ объемом $e = 2$ регистра оценки числа обращений к ОЗУ составляют: $\underline{N}_2 = 3$, $\overline{Q}_2 = 19$, $Q_2 = 8$.

С помощью алгоритма 6.4. получается сеть G , определяющая загрузку двух регистров СОЗУ (рис. 6.8). В сети G каждый вид зачернения вершин соответствует своему цвету.

Оптимальная загрузка СОЗУ объемом $e = 2$, регистра приведена в табл. 6.6.

Таблица 6.6

Оптимальное распределение загрузки двух регистров СОЗУ

Pr_j	A_i									
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
1	m'_3	$+m'_2$	m_2	m_2	m_2		$+m'_3$	m_3	m_2	$+m'_4$
2	m'_1	m_1	$+m'_4$	m_4	$+m'_1$	m_1	m_1	m_1	$+m'_2$	m_2

На рис. 6.9 представлены функциональные зависимости $N(e)$ и $Q(e)$ для рассматриваемого примера. Некоторое уменьшение значе-

ния функции $Q(e)$ при $e = 3$ объясняется возможностью улучшения загрузки СОЗУ за счет дополнительного использования свободных регистров [см. условие (6.11)].

Сложность программы или аппаратуры, управляющей обменом информацией между СОЗУ и устройствами, зависит от характера загрузки СОЗУ. При этом следует отметить, что на общее время реализа-

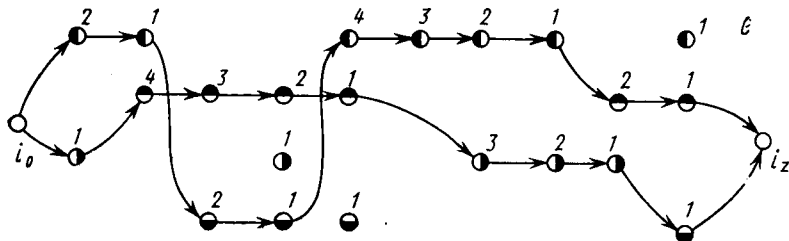


Рис. 6.8. Загрузка регистров СОЗУ данными ($e=2$)

ции алгоритма основное влияние оказывает число обращений N ОУ к ОЗУ. Время, необходимое на осуществление Q обращений СОЗУ к ОЗУ за операндами и константами, частично или полностью перекрывается временем работы АЛУ по реализации предыдущего оператора.

Методика оптимального распределения загрузки СОЗУ применима и для более общего случая, когда программа работы МПУ содержит команды условного перехода (УП), циклы неопределенной длины, а в структуре внутреннего языка МПУ предусмотрены принципы относительной адресации.

При наличии в программах работы ММ команд УП в СОЗУ следует добавить специальный регистр для хранения адреса начала второй ветви программы, что позволит сократить время при переходе программы на эту ветвь. Этот же регистр может быть использован и при реализации циклов неопределенной длины, поскольку после каждого выполнения цикла требуется реализовать команду УП на окончание или продолжение цикла.

В МПУ с переменной относительной адресацией следует предусматривать блок формирования исполнительных адресов, причем полноразрядный регистр, предназначенный для хранения базового адреса, должен быть вынесен в СОЗУ.

После внесения этих изменений для оптимизации загрузки СОЗУ могут быть применимы описанные выше методы.

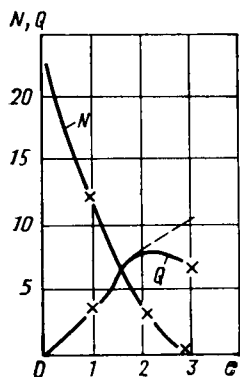


Рис. 6.9. Функциональные зависимости $N = \varphi(e)$ и $Q = \psi(e)$

§ 6.5. МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА И МИКРОЭВМ В ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНОЙ АППАРАТУРЕ

Контрольно-измерительная техника поистине оказалась питательной средой для широкого и эффективного внедрения микропроцессорной техники. Использование встроенных в измерительную аппаратуру МПУ и микроЭВМ позволило существенно улучшить характеристики приборов (точность, надежность, экономичность и др.). Появилась возможность осуществления автоматизации статистической обработки результатов измерений, разработки и выпускаются «интеллектуальные» приборы — полностью автоматизированные, радикально повышающие производительность труда и позволяющие решать совершенно новые задачи.

Основные функции МПУ в измерительных приборах. Встроенное в состав измерительного прибора МПУ позволяет модернизировать приборы, предназначенные для измерения одного параметра сигнала, в *многофункциональные* приборы для измерения нескольких параметров сигнала. Преобразование прибора из однофункционального в многофункциональный осуществляется путем объединения нескольких функциональных узлов совместно с коммутирующими устройствами в одном конструктиве. МПУ, введенное в состав многофункционального устройства измерения, сделало его программно-управляемым прибором. Функциональные возможности такого прибора определяются выполняемой программой и могут быть легко видоизменены путем перехода к другой программе, хранимой в ПЗУ микропроцессорного устройства.

Применение встроенных МПУ позволяет *повысить точность* измерительных приборов за счет автоматической компенсации систематической погрешности, в частности автоматической установки нуля перед началом измерений; автоматического выполнения градуировочной операции (самокалибровки); выполнения самоконтроля; проведения автоматической статистической обработки результатов измерений; вывода на дисплей информации о результатах процесса измерений.

Применение микропроцессоров позволяет расширить *измерительные возможности* приборов за счет использования косвенных и совокупных измерений. Напомним, что косвенное измерение заключается в нахождении искомого значения физической величины по известной математической зависимости между этой величиной и физическими величинами, подвергаемыми прямым измерениям (например, измерение цифровым вольтметром рассеиваемой на нагрузочном резисторе мощности согласно формуле $P = U^2/R$, измерение сопротивлений резисторов на основе формулы закона Ома $R = U/I$ и др.). При совокупных измерениях одновременно производят измерения нескольких одноименных физических величин, при которых искомые значения величин находят путем решения системы уравнений, получаемых при прямых измерениях различных сочетаний этих величин. МПУ програм-

мируют при этом на реализацию необходимых аналитических зависимостей.

В измерительных приборах очевидны следующие функции МПУ: *упрощение управления* прибором за счет уменьшения числа органов управления на панели прибора, возможность проведения различных *математических преобразований* над результатами измерений; получение *статистических характеристик* измеряемых величин (математическое ожидание, дисперсия, коэффициент корреляции взаимозависимых случайных переменных и др.); *миниатюризация и экономичность* аппаратуры за счет уменьшения числа компонентов в схеме прибора; *повышение надежности* прибора; *сокращение сроков разработки* за счет использования типовых структурных решений и типовых прикладных программ.

При проектировании микропроцессорных средств измерений возможно подключение измерительного прибора к стандартной интерфейсной шине. Это позволяет осуществлять дистанционное управление измерительными приборами, организовывать их в информационно-измерительную систему, которую в процессе проектирования и эксплуатации можно наращивать, дополнять новыми функциональными блоками и при необходимости заменять в ней одни блоки другими. При таком построении система получается функционально гибкой. Следует помнить, что каждому прибору, который может работать в системе, объединенной интерфейсом, присущи приборные и интерфейсные функции.

Основные *приборные функции* следующие. Первая функция, определяемая назначением прибора, — формирование сигнала измерительной информации. Она включает все слагаемые измерительной процедуры, выполняемой прибором: масштабные преобразования исследуемого сигнала (усиление, ослабление); воспроизведение единицы; преобразование исследуемого сигнала к виду, удобному для сравнения; операцию сравнения с единицей; фиксацию результата сравнения; отображение результата измерения, его запоминание и статистическую обработку и т.п.

Вторая приборная функция — программное управление прибором, заключающееся в выполнении программы, хранимой в ПЗУ и частично в ОЗУ микроЭВМ или МПУ.

Набор *интерфейсных функций*, выполняемых конкретным прибором, определяется задачами, которые должен решать этот прибор, и условиями его применения. Обычно в микропроцессорной измерительной аппаратуре осуществление интерфейсных функций возлагается на МПУ или микроЭВМ. Такое использование встроенного МПУ особенно целесообразно при большом числе интерфейсных присоединений и сложности операции сопряжения. Однако организация микропроцессорного управления интерфейсом требует хорошо обоснованного выбора микропроцессора (прежде всего по быстродействию и возможности прерывания) и рационального программирования работы МПУ.

ступают в формирователь временных ворот ФВВ, содержащий делитель частоты, у которого устанавливается одно из значений коэффициента деления $q_i = 2^k i$.

В режиме косвенного измерения измерительная процедура заключается в измерении периода T_x . Для этого микроЭВМ подает сигнал на управляющие входы 3 обоих мультиплексоров M_1 и M_2 , а на временной селектор ВС поступают импульсы с ФВВ и генератора счетных импульсов ГСИ. Одновременно у делителя частоты ФВВ устанавливается

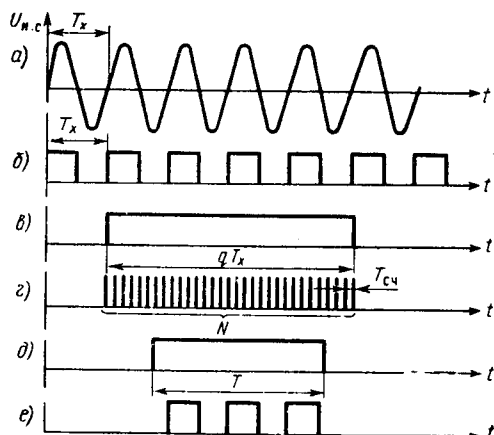


Рис. 6.11. Временные диаграммы работы микропроцессорного частотомера

выбранный коэффициент деления q . В результате на вход 1 ВС поступает стробирующий импульс — временные ворота, длительностью qT_x (рис. 6.11, б), где $T_x = 1/f_x$, а к входу 2 ВС подводятся счетные импульсы от ГСИ, заполняющие стробирующий импульс с частотой $F_{сч}$ (рис. 6.11, в). Число счетных импульсов в стробе $N = qF_{сч}T_x$. Величина N определяется с помощью счетчика СЧ и после преобразований в микроЭВМ в десятичном виде индицируется на экране дисплея Д.

В режиме прямого измерения частоты микроЭВМ подключает входы ВС к входам 2 M_1 и M_2 . После этого на вход 1 ВС из микроЭВМ подается стробирующий импульс (временные ворота) постоянной длительности T , например 1 с (рис. 6.11, б), а на вход 2 ВС с выхода ФИ поступает периодическая последовательность импульсов с частотой следования f_x . Счетчик СЧ подсчитывает число импульсов, попадающих во временные ворота (рис. 6.11, в), а результат измерения после обработки в микроЭВМ индицируется на дисплее.

В микропроцессорном частотомере может быть осуществлен автоматический контроль нормального функционирования прибора. Для этого по команде микроЭВМ выход ГСИ, импульсы которого следуют с частотой $F_{сч}$, подключается к входу работающего прибора. При нормальном функционировании частотомера дисплей будет показывать значение $F_{сч}$.

Широкодиапазонный автоматический частотомер. Для достижения широкого диапазона измерений в частотоизмерительную аппаратуру включают программируемый синтезатор частот ПСЧ и встроенную микропроцессорную систему МПС (рис. 6.12). Следует отметить, что приборы подобного типа могут без перестроек измерять час-

тоты, значения которых заключены в очень широком диапазоне, например от 10 Гц до 32 ГГц.

ПСЧ – устройство, преобразующее сигнал с фиксированным значением частоты, вырабатываемый высокостабильным источником, в сигнал, дискретные значения частоты которого устанавливаются с требуемым шагом в определенном диапазоне частот. Переход от одного значения частот к другому осуществляет МПС.

Для нахождения значения частоты в поддиапазоне от 10 Гц до 300 МГц напряжение исследуемого сигнала подается на вход 1 и измерение выполняет непосредственно цифровой частотомер ЦЧ. При более высоких значениях частоты $f_{изм}$ сигнал подается на вход 2 прибора.

Работа схемы заключается в следующем. Напряжение исследуемого сигнала поступает через входной блок ВБ на вход 1 смесителя, ко входу 2 которого подводится напряжение выходного сигнала ПСЧ. МПС изменяет частоту $f_{син}$ выходного сигнала синтезатора до такого значения $f'_{син}$, при котором разность измеряемой частоты $f_{изм}$ и n -й гармоники выходного сигнала ПСЧ $nf'_{син}$ примет значение $f'_{пр}$, находящееся в полосе пропускания усилителя промежуточной частоты УПЧ. При этом детектор формирует сигнал, по которому МПС прекращает дальнейшую перестройку частоты синтезатора. Значение $f'_{пр}$ измеряется ЦЧ и результат измерения направляется в память МПС. Искомое значение частоты $f_{изм}$ определяется как сумма или разность величин $nf'_{син}$ и $f'_{пр}$ в зависимости от того, больше или меньше $f_{изм}$ по сравнению с $f'_{пр}$.

Цифровые вольтметры. Цифровые вольтметры получили широкое распространение в технике измерения напряжений постоянного и переменного токов. Вольтметры оказались наиболее благодатными приборами для реализации в них идей и принципов микропроцессорной техники. Типовая структурная схема микропроцессорного цифрового вольтметра представлена на рис. 6.13.

Входной блок ВБ содержит аналоговые преобразователи — прежде всего аттенуатор и усилитель, но в некоторых приборах в состав блока может входить также измерительный преобразователь напряжения переменного тока в напряжение постоянного тока. Такое схемное решение применяется тогда, когда быстродействие микропроцессора недостаточно для вычисления среднеквадратического значения измеряемого напряжения. Обязательным узлом каждого цифрового вольт-

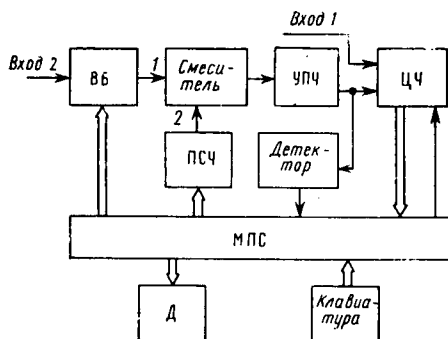


Рис. 6.12. Структурная схема широкодиапазонного частотомера

метра, в том числе микропроцессорного, является аналого-цифровой преобразователь АЦП.

Микропроцессорные (цифровые) вольтметры представляют собой многопредельные приборы с автоматическим и ручным переключением полярности и диапазонов измерений, которые проводят статистическую обработку результатов наблюдений, выполняют ряд специфических подпрограмм, существенно расширяющих возможности прибора. В микропроцессорных вольтметрах осуществляется автокалибровка, а сами приборы отличаются высокими метрологическими характеристиками.

В цифровых вольтметрах МПС организует процедуру измерений, управляет работой всех узлов, производит арифметические и логические операции с данными, поступающими из АЦП через интерфейс

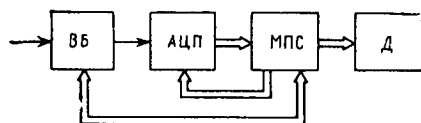


Рис. 6.13. Структурная схема микропроцессорного цифрового вольтметра

на шину данных. Программа работы МПС, составленная изготовителем цифрового вольтметра, хранится в ПЗУ МПС. Измеряемые величины и промежуточные результаты вычислений хранятся в ОЗУ и в регистрах общего назначения. Результаты измерений, получаемые в итоге статистической обработки входных данных, отображаются на цифровом дисплее Д.

Следует отметить, что цифровые вольтметры, построенные на основе микропроцессорных средств, являются многофункциональными программируемыми приборами. Применяемые программы и подпрограммы определяют возможные функции вольтметра. Они неодинаковы у различных приборов, но многие из них осуществляют такие функции: умножение на масштабный коэффициент, вводимый с клавиатуры прибора; сдвиг; определение относительного отклонения результата измерения от заданного значения; определение максимального, минимального и среднего (за время измерения) значений напряжения, причем продолжительность измерений может быть большой (дни, недели), что важно при очень медленных изменениях исследуемой физической величины; статистическая обработка данных наблюдений и т.д.

Измерительные генераторы с микропроцессорным управлением. Измерительные генераторы предназначены для формирования сигналов различных форм и частот. Среди них в настоящее время широкое распространение получили функциональные генераторы, вырабатывающие различные формы сигналов (треугольную, прямоугольную синусоидальную и др.) с нормируемыми метрологическими характеристиками. Диапазон частот функциональных генераторов довольно широк — $10^{-8} \dots 50 \cdot 10^6$ Гц. Это наиболее универсальные и широко-

диапазонные измерительные генераторы аналогового типа. Существуют измерительные генераторы, построенные на принципе цифроаналогового преобразования — это цифровые синтезаторы сигналов. Использование микропроцессорного управления в измерительных генераторах обоих типов значительно расширяет возможности и области применения этих приборов. Приборы со встроенным микропроцессором становятся «интеллектуальными» и служат для автоматизации измерений и контроля параметров РЭС. Применение микропроцессоров в измерительных генераторах позволило создать новый тип источника колебаний — генератор сигналов произвольной формы (ГСПФ).

Рассмотрим принцип построения ГСПФ на основе цифрового синтезатора сигналов (ЦСС) с микропроцессором. Основное требование, предъявляемое к ГСПФ, — полная программируемость формы сигнала без наличия специальной подготовки у оператора. Структурная схема ГСПФ представлена на рис. 6.14.

В состав генератора входят: МПС, обеспечивающая вычисления и управление генераторов;

ОЗУ; система обращения к памяти, в состав которой входят программируемые блок счетчиков БСч и генератор тактовых импульсов ГТИ; ЦАП с выходными усилителями. МПС запрограммирована на управление каждой функцией прибора и на диалог с оператором. После ввода оператором через терминал функции $f(t)$ для генерирования сигнала той же формы программа вычисляет отсчеты сигнала $f(t_i)$ на интервале одного периода с заданной частотой выборки. Отсчеты сигнала записываются в ОЗУ, содержание которого периодически адресуется для запуска ЦАП, генерирующего выходной сигнал требуемой формы. ГТИ программно управляется от МПС с помощью системы фазовой автоподстройки частоты, входящей в состав ГТИ. Выходной сигнал ГТИ поступает в БСч, где формируется адресный сигнал для ОЗУ. Опрос адресов обеспечивает подачу соответствующих отсчетов для запуска ЦАП. МПС управляет работой ГТИ, БСч, ОЗУ, связана с портом ввода — вывода от терминала, имеет дополнительный порт для блока кассет.

Таким образом, микропроцессорное управление в измерительных генераторах является весьма перспективным направлением, так как существенно расширяет их функциональные возможности, улучшает метрологические характеристики и позволяет создавать автоматизированные системы измерений и контроля для серийного производства разнообразных РЭС. Применение МП в измерительных генераторах по-

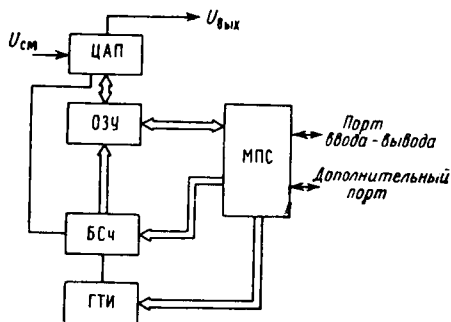


Рис. 6.14. Структурная схема ГСПФ

звolyет наряду с типовыми сигналами генерировать сигналы произвольной формы, в том числе задаваемые оператором, использовать новые режимы работы, принципиально невозможные для традиционных измерительных генераторов.

§ 6.6. МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В ТЕХНИКЕ СВЯЗИ

Основные функции отрасли связи состоят в представлении потребителям технических возможностей для обмена информацией в виде сообщений и почтовых отправлений. К техническим средствам связи относятся сети электрической и почтовой связи, средства радиосвязи и телевидения, а также автоматизированные системы управления и технического обслуживания средств связи.

Внедрение средств вычислительной техники в отрасль связи стимулирует развитие новых служб, возникающих в последнее время. К ним относятся электронная почта, которая осуществляет передачу рукописных или машинописных сообщений по телефонной сети с документированием на терминале абонента или предприятия связи; справочная служба, позволяющая передавать информацию из некоторых информационных центров по телефонным каналам на терминалы абонентов; справочное вещание, обеспечивающее передачу справочной и газетно-журнальной информации по телевизионному каналу, и т. д.

Особенностью технических средств связи является их территориальная распределенность, поэтому одно из основных требований, предъявляемых к вычислительным средствам, — возможность реализации принципа распределенной обработки информации, что позволяет существенно снизить требования к пропускной способности каналов связи. Естественной реализацией систем управления сетями связи являются распределенные вычислительные системы, состоящие из совокупности элементарных машин (ЭМ), объединенных сетью связи и сопрягаемых с объектами контроля и управления посредством микропроцессорных устройств связи с объектами. ЭМ состоит из вычислительного модуля, в качестве которого используется одна из серийно выпускаемых микроЭВМ, и модуля системного интерфейса, обеспечивающего функционирование набора ЭМ как единой системы. Система снабжается типовым набором средств хранения и отображения информации, а также интерфейсными платами для сопряжения с каналами связи и объектами управления и обслуживания. Локальные вычислительные системы сетей связи объединяются в региональные распределенные вычислительные системы, образуя многоуровневые системы с распределенной обработкой информации и децентрализованным управлением. Подобная концепция позволяет использовать микропроцессорные средства с ограниченными ресурсами для создания управляющих систем, обладающих широким спектром функций и большими возможностями.

Рассмотрим ряд применений микропроцессорных средств в технике связи.

Узел коммутации цифровой сети связи. Для коммутации каналов, пакетов и сообщений в сетях связи применяются узлы коммутации, являющиеся основой электронных и квазиэлектронных АТС. Структура узла коммутации приведена на рис. 6.15. Вычислительное ядро узла составляют элементарные машины (микроЭВМ) ЭМ_і с системой команд, ориентированной на задачи коммутации. Сеть связи, объединяющая все микроЭВМ в единую систему, образована модулями системного интерфейса (СИ), коммутационно-настроечными автоматами (КНА) и множеством магистралей. Функциями КНА являются коммутация элементарных машин и реализация общесистемных взаимодей-

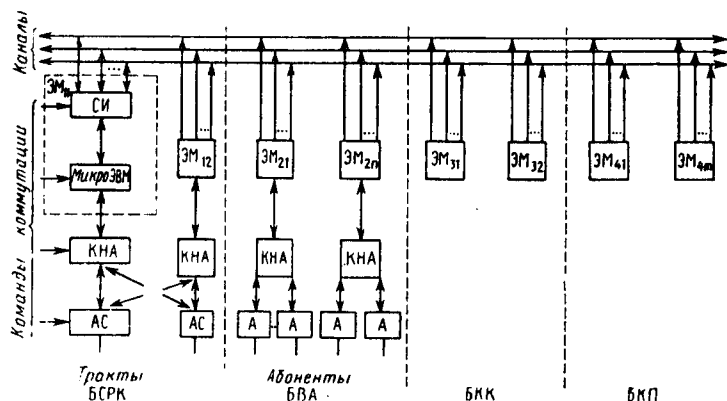


Рис. 6.15. Структура узла коммутации

ствий, позволяющих множеству ЭМ функционировать в виде вычислительной системы, а также организация связи с объектами. Функционально узел коммутации делится на четыре блока. Число ЭМ в каждом блоке определяется требуемой надежностью и пропускной способностью узла. ЭМ всех блоков идентичны. Специализация ЭМ отдельных блоков осуществляется посредством специализации программного обеспечения и интерфейсов с внешними устройствами. Интерфейс блока сборки — разборки кадров (БСРК) представляет собой микропроцессорный адаптер связи (АС) с цифровым высокоскоростным трактом. Элементарные машины блока взаимодействия с абонентами (БВА) имеют микропроцессорный интерфейс для подключения цифровых абонентских линий с относительно невысокой скоростью передачи информации. В процессе работы узла коммутации происходит функциональное распараллеливание (каждый блок реализует свои функции одновременно с другими) и распараллеливание по данным (все ЭМ функционируют параллельно).

Микропроцессорный блок автоматики абонентской радиостанции. В абонентских радиостанциях эффективно применяются средства микропроцессорной техники для построения блока автоматики, функции

которого заключаются в реализации алгоритмов цифровой фильтрации, статистической обработки сигнала с выхода приемника радиостанции и управления режимами работы радиостанции.

Задача обработки сигнала с выхода приемника радиостанции состоит в следующем. Имеется множество частот $\Omega = \{\omega_1, \dots, \omega_m\}$ взаимодействия радиостанции, причем частоты $\omega_1, \dots, \omega_r$ кодируют индивидуальный номер станции, частота $\omega_1 \in \Omega$ — циркулярный вызов, $\omega_p \in \Omega$ — маркерный сигнал, а $\omega_n \in \Omega$ — сигнал отбоя. Пусть $\Delta\Omega = \omega_{q+1} - \omega_q$ — частотный интервал частот ω_q взаимодействия

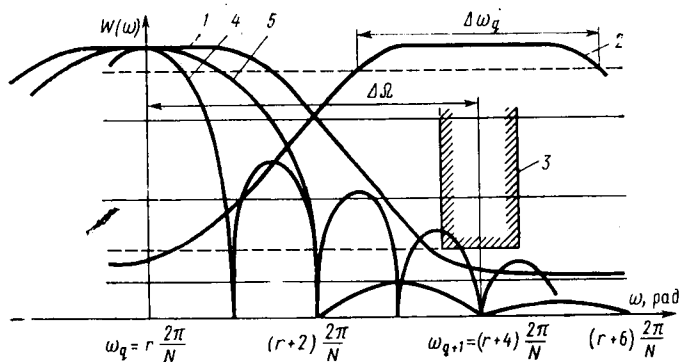


Рис. 6.16. Частоты взаимодействия радиостанции

радиостанции (рис. 6.16), $\Delta\omega_q$ — полоса пропускания фильтра по уровню 0,7. Примем, что избирательность при отстройке на частоте $\pm (\Delta\Omega - \Delta\omega_q/2)$ составляет не менее η дБ (кривая 3 на рис. 6.16). Исследуемый временной сигнал $x(t)$ на выходе приемника радиостанции представляет собой сумму m вещественных синусоид сигнала и аддитивной помехи:

$$x(t) = \sum_{q=1}^m A\omega_q \sin(2\pi f_q t + \Theta_q) + n(t), \quad (6.16)$$

где $A\omega_q$ — амплитуда сигнала на частоте ω_q . Здесь случайные фазовые сдвиги Θ_q взаимно независимы и равномерно распределены на интервале $[-\pi, \pi]$, а помеха $n(t)$, возникшая из-за шумов радиостанции и радиоэфира, представляет собой шум с нулевым математическим ожиданием и дисперсией σ^2 .

Режим работы радиостанции можно определить, если за время T_d сигналы частот взаимодействия ω_q обнаружены во временном сигнале $x(t)$ j -го радиоканала при $j = 1, \dots, d$ (d — количество радиоканалов), т. е. когда выполняются условия

$$A\omega_q \geq A_n \text{ для } \forall \omega_q \in \Omega, \quad (6.17)$$

где A_n — пороговое значение сигнала.

Таким образом, на блок автоматики абонентской радиостанции возлагаются задачи обнаружения индивидуального и циркулярного вызовов маркерного сигнала и сигнала отбоя в выходном сигнале $x(t)$ приемника радиостанции, а также включения соответствующих режимов работы станции после обнаружения данных сигналов.

Используемые для частотной селекции аналоговые резонаторы с полосой пропускания $\Delta\omega_q$ (кривые 1, 2 на рис. 6.16) не надежны и не позволяют достаточно оперативно изменять номер радиостанции, который кодируется центральными частотами резонаторов из множества Ω , и перестраивать алгоритм функционирования блока автоматики в процессе модернизации радиостанции. Применение микропроцессоров для цифровой обработки аналоговых сигналов делает систему управления универсальной и гибкой, при этом повышается точность обработки сигнала, увеличивается устойчивость схемы к внешним воздействиям и отпадает необходимость в применении прецизионных компонентов.

Цифровая обработка сигнала $x(t)$ может производиться разными методами, но наиболее экономичной по вычислительным затратам является обработка сигнала $x(t)$ в частотной области с помощью метода оценивания спектральной плотности мощности по периодограмме для m частот в соответствии с выражением

$$X(q\Delta\omega) = \left[\frac{1}{M} \frac{1}{N} \sum_{r=1}^M \sum_{i=1}^N x_r(i\Delta t) \omega(i\Delta t) e^{-i\varphi} \right]^2. \quad (6.18)$$

Здесь $X(q\Delta\omega)$ — спектральное отображение реализации $x(t) \omega(t)$; $x_r(i\Delta t)$ — отсчеты входного сигнала; $\omega(i\Delta t)$ — отсчеты весовой функции временного окна; $M = T_d/N\Delta t$ — число усреднений по выборке сигнала для получения статистически состоятельных оценок спектров периодограмм; N — объем выборки сигнала $x(t)$; $\varphi = (2\pi/N)qi$.

При программной реализации выражения (6.18) за интервал времени Δt производится m операций умножения отсчета $x(i\Delta t)$ на значение системы ортогональных функций, одно умножение $x(i\Delta t)$ на отсчет весовой функции временного окна, а также m операций сложения, так как обработка ведется сразу по всем частотам ω_q . Операция возведения в квадрат и проверка условий (6.17) могут выполняться в конце интервала T_d , причем $T_d \gg \Delta t$.

Следует отметить, что при соответствующем выборе частотного разрешения $\Delta\omega$ частотная селекция сигнала $x(t)$ будет практически совпадать с результатами обработки сигнала аналоговыми фильтрами (кривая 5 на рис. 6.16). Структурная схема блока автоматики абонентской радиостанции представлена на рис. 6.17. Блок автоматики с общей шиной состоит из блока цифровой обработки сигналов, на который возлагаются задачи обнаружения индивидуального и циркулярного вызовов маркерного сигнала и сигнала отбоя в выходном сигнале $x(t)$, а также блока цифрового автомата, включающего соответствующие режимы станции согласно алгоритму ее функционирования.

Блок автоматики может быть реализован на 16-разрядных МП (в частности, на малознергоемком МПК серии К588).

Все устройства, подключенные к общей шине, используют ее раздельно во времени, что позволяет процессору рассматривать внешние устройства как активные ячейки памяти, выполняющие специальные функции. При адресации внешних устройств три старших разряда регистра адреса RgA (0...2) управляют выборкой нужного внешнего устройства, а остальные разряды RgA (3...15) задают адрес, по которому идет запись — чтение информации в выбранном устройстве.

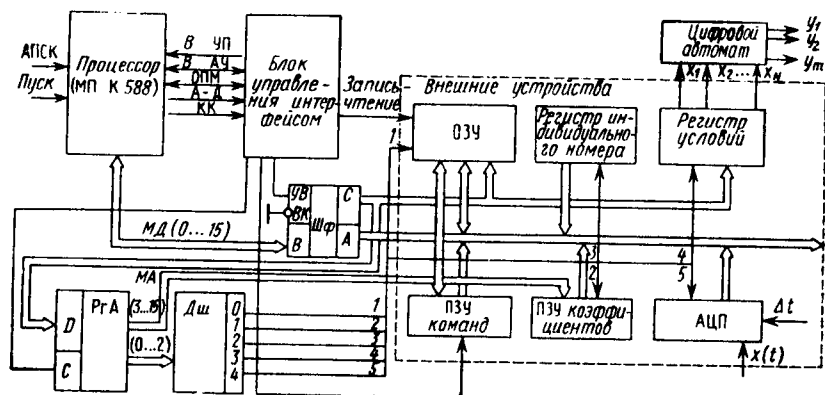


Рис. 6.17. Структурная схема блока автоматики абонентской радиостанции

Блок управления интерфейсом анализирует и выдает следующие сигналы:

сигнал «Пуск», инициирующий команду «Пуск начальный»;
сигнал «КК» (признак конца команды), по которому производится выбор команды из ПЗУ команд;

сигнал «В УП» (выдано в управляющую память) сопровождения (квартирования), по которому команда принимается в регистр команд управляющей памяти (УП);

сигнал «В АУ» (выдано в АУ), выдаваемый процессором или внешним устройством информации;

сигнал «ОПС» (окончание приема), являющийся ответным сигналом квартирования принимаемой информации;

сигнал «А — Д» (адрес — данные) управления интерфейсом, идентифицирующий тип выдаваемой АУ информации в общую шину.

Микропроцессорный блок автоматики работает таким образом. Анализ выходного сигнала $x(t)$ приемника станции по алгоритму (6.18) продолжается до прихода сигнала АПСК (автоматический по-

иск свободного радиоканала из d каналов). На выходе БИС УП формируется код команды прерывания, по которой процессор выполняет пересылку содержимого регистра состояния и счетчика команд в стек по указателю стека и принимает адрес вектора прерывания. Происходит переход к подпрограмме обнаружения индивидуального и циркулярного вызовов маркерного сигнала. Период следования сигналов АПСК равен T_d .

Результаты обнаружения сигналов записываются в регистр условий и являются логическими условиями $x_1, \dots, x_N$ для работы цифрового автомата (ЦА). Когда хотя бы один сигнал обнаружен, ЦА фиксирует j -й радиоканал на выходе y_j .

Использование микропроцессорного блока автоматики позволяет исключить из состава радиостанции аналоговые резонаторы. При этом увеличиваются надежность работы автоматики радиостанции и универсальность применения этого блока за счет возможных изменений алгоритма его работы.

§ 6.7. МИКРОПРОЦЕССОРЫ В СИСТЕМАХ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

Цифровая обработка сигналов средствами вычислительной и микропроцессорной техники позволяет решить задачу унификации тракта обработки сигналов на фоне помех. Один и тот же вычислитель при соответствующей его настройке может осуществить необходимую адаптацию для обработки сигналов с получением результатов, близких к оптимальным. В процессе цифровой обработки сигналов решаются задачи фильтрации, спектрального анализа, поиска сигнала и т. д.

Программная реализация алгоритмов цифровой обработки сигналов в реальном времени на универсальных ЭВМ часто затруднена из-за малого быстродействия ЭВМ общего назначения при выполнении специфических алгоритмов для обработки сигналов (преобразование Фурье, фильтрация и т. п.). Поэтому в системах цифровой обработки сигналов широко используют специализированные цифровые устройства, реализующие алгоритм дискретной фильтрации, дискретного преобразования Фурье, быстрого преобразования Фурье и другие функции. Указанная реализация специализированных вычислителей требует, чтобы быстродействие аппаратно реализованного цифрового вычислителя превышало на один-два порядка ширину полосы частот обрабатываемого сигнала. Построение специализированных вычислителей на микропроцессорной элементной базе с учетом возможности использования многопроцессорных архитектур позволяет на два-три порядка увеличить информационную производительность систем и существенно улучшить массогабаритные, надежность, энергетические и стоимостные характеристики. Рассмотрим основные аспекты применения микропроцессоров в системах цифровой обработки сигналов.

1. Аналоговые и цифровые сигналы

Пусть $x(t)$ — аналоговый сигнал, а соответствующий ему цифровой сигнал, полученный в результате аналого-цифрового преобразования, есть $x(nT)$. Здесь T — тактовый период; n — номер отсчета аналогового сигнала при его преобразовании в цифровую форму; nT — моменты отсчета аналогового сигнала. При этом должен быть известен спектр аналогового сигнала, а тактовый период должен удовлетворять условию $T \leq 1/2 F_{\max}$, где $F_{\max}$ — максимальная частота спектра.

В дальнейшем будем считать, что при $t < 0$ $x(t) = 0$, следовательно, отличные от нуля значения $x(nT)$ могут иметь место лишь при $nT \geq 0$ и n представляет собой последовательность 0, 1, 2, ... чисел натурального ряда.

Из теории сигналов известно, что операция получения спектральной функции $X(j\omega)$ аналогового сигнала $x(t)$ и обратная операция получения сигнала $x(t)$ по известной спектральной функции $X(j\omega)$ производятся с помощью пары преобразований Фурье:

$$X(j\omega) = \int_0^{\infty} x(t) e^{-j\omega t} dt, \quad x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) e^{j\omega t} d\omega.$$

Для цифрового сигнала спектральная функция последовательности $x(nT)$ обозначается $X(e^{j\omega T})$, а преобразования Фурье определяются выражениями

$$X(e^{j\omega T}) = \sum_{n=0}^{\infty} x(nT) e^{-jn\omega T},$$

$$x(nT) = \frac{T}{2\pi} \int_{-\pi/T}^{\pi/T} X(e^{j\omega T}) e^{jn\omega T} d\omega.$$

Преобразование Фурье, независимо от того, проводится ли оно над аналоговым или дискретным сигналом и является ли прямым или обратным, характеризуется следующим свойством: *преобразование Фурье, выполняемое над периодической функцией, приводит к дискретной функции и, наоборот, преобразование Фурье дискретной функции является периодической функцией.*

При решении практических задач вычисление выражения для $X(e^{j\omega T})$ не может быть реализовано из-за необходимости суммирования бесконечного числа слагаемых. При использовании конечного числа отсчетов аналогового сигнала переходим от обычного преобразования Фурье к дискретному преобразованию Фурье.

2. Дискретное преобразование Фурье

Дискретным спектром последовательности $x(nT)$ называют комплексную функцию

$$X(k\Omega) = \sum_{n=0}^{N-1} x(nT) e^{-jkn\Omega T}. \quad (6.19)$$

В свою очередь, если известна функция $X(k\Omega)$, то временная последовательность, соответствующая данной функции, может быть определена из выражения

$$x(nT) = \frac{1}{N} \sum_{k=0}^{N-1} X(k\Omega) e^{jkn\Omega T}. \quad (6.20)$$

Формулы (6.19) и (6.20) представляют собой пару преобразований Фурье. При этом преобразование (6.19) называется прямым дискретным преобразованием Фурье (ДПФ), а преобразование (6.20) — обратным дискретным преобразованием Фурье (ОДПФ). В данных выражениях $X(k\Omega)$ — последовательность из N частотных отсчетов, $k = 0, 1, 2, \dots, N-1$; $x(nT)$ — последовательность из N временных отсчетов с периодом T , $n = 0, 1, 2, \dots, N-1$.

Довольно часто в литературе обозначают $W = e^{-j\frac{2\pi}{N}}$, $j = \sqrt{-1}$. Тогда выражения (6.19) и (6.20) могут быть записаны в виде

$$X(k\Omega) = \sum_{n=0}^{N-1} x(nT) W^{kn}, \quad k = 0, 1, 2, \dots, N-1,$$

$$x(nT) = \frac{1}{N} \sum_{k=0}^{N-1} X(k\Omega) W^{-kn}, \quad n = 0, 1, 2, \dots, N-1.$$

Дискретное преобразование Фурье вводится для представления как периодических последовательностей с периодом N отсчетов, так и последовательностей конечной длины N . Коэффициенты ДПФ конечной последовательности равны значениям ее z -преобразования в N точках, равномерно распределенных по единичной окружности, т. е.

$$X(k\Omega) = X(z) \big|_{z=e^{j\frac{2\pi k}{N}}}, \quad k=0, 1, 2, \dots, N-1.$$

Процесс вычисления коэффициентов Фурье по N отсчетам сигнала заключается в перемножении каждого вновь поступившего отсчета на

соответствующий весовой коэффициент и суммировании результата с ранее полученными значениями предыдущих произведений. Структурная схема устройства, реализующего вычисление четырех коэффициентов Фурье по четырем входным отсчетам, представлена на рис. 6.18.

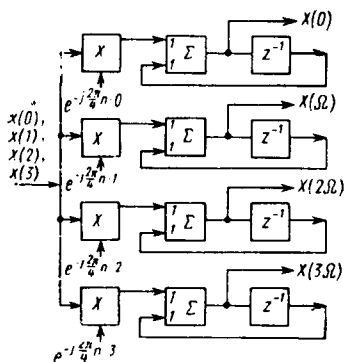


Рис. 6.18. Структурная схема устройства, реализующего алгоритм ДПФ

Очевидно, подобная организация вычислительного процесса в однопроцессорной ЭВМ требует на каждый вновь поступивший входной отсчет N операций комплексного перемножения и суммирования. Для поступивших N входных отсчетов количество этих операций будет равно N^2 . Поэтому из-за ограничений на быстродействие ЭВМ всегда увеличение числа эффективно используемых входных отсчетов будет сопровождаться сужением диапазона анализируемых частот. Последнее связано с тем, что между n -м и $(n+1)$ -м входными отсчетами нужно успеть

проделать все n операций комплексного умножения и сложения, а с возрастанием n увеличивается интервал временной дискретизации и, как следствие, уменьшается частота $F_{\max}$ в спектре сигнала.

3. Быстрое преобразование Фурье

При организации вычислительного процесса по реализации комплексной функции $X(k\Omega)$, согласно выражению (6.19), необходимо произвести N^2 комплексных умножений и сложений, что при больших N требует значительных затрат машинного времени. Если входная последовательность $x(nT)$ имеет строгое ограничение по длительности, а момент ее появления известен, для экономии затрат машинного времени применяют алгоритмы быстрого преобразования Фурье (БПФ). В этих методах используется искусственный прием, основанный на том, что сумма произведений по выражению (6.19) может быть упрощена за счет приведения подобных членов. Это позволяет выполнить операции сложения до умножения или заменить комплексные операции на действительные.

Алгоритмы БПФ находят применение в случае, когда число отсчетов временной последовательности является сложносоставным, причем предполагается, что все отсчеты перед началом вычислений находятся в памяти вычислителя и пригодны для обработки. Заметим, что в том важном частном случае, когда N является степенью числа 2, алгоритмы БПФ требуют проведения $N \log_2 N$ вычислений. Так, в частности, использование БПФ при $N = 1024$ составляет 99 %-ную экономию в вычислениях по сравнению с ДПФ. Для эффективной ре-

лизации алгоритмов БПФ необходимо, чтобы N разлагалось на возможно большее число отсчетов.

Существует два больших класса алгоритмов БПФ — алгоритмы с прореживанием по времени и алгоритмы с прореживанием по частоте.

При дальнейшем изложении будем пользоваться обозначениями в виде индексов. Тогда выражение для ДПФ можно записать в виде

$$X_k = \sum_{n=0}^{N-1} x_n W^{nk}.$$

Рассмотрим алгоритм БПФ с *прореживанием по времени*. Предположим, что число отсчетов делится на два. Тогда последовательность x_n можно представить в виде объединения двух более коротких последовательностей: q_l , состоящей из четно пронумерованных отсчетов, и h_l , состоящей из нечетно пронумерованных отсчетов. Эти последовательности формально задаются как

$$q_l = x_{2l}, h_l = x_{2l+1}, l = 0, 1, 2, \dots, (N/2) - 1. \quad (6.21)$$

ДПФ этих последовательностей также являются последовательностями, состоящими из $N/2$ точек, и могут считаться непериодическими последовательностями с периодом $N/2$. Они могут быть записаны в виде

$$G_k = \sum_{l=0}^{N/2-1} q_l (W^2)^{lk}, H_k = \sum_{l=0}^{N/2-1} h_l (W^2)^{lk}, \quad (6.22)$$

а ДПФ всей последовательности имеет вид

$$X_k = \sum_{l=0}^{N/2-1} (q_l W^{2lk} + h_l W^{(2l+1)k}). \quad (6.23)$$

Используя выражения (6.21) и (6.22) для ДПФ вида (6.23), получаем

$$X_k = \sum_{l=0}^{N/2-1} q_l (W^2)^{lk} + W^k \sum_{l=0}^{N/2-1} h_l (W^2)^{lk} = G_k + W^k H_k. \quad (6.24)$$

Соотношение (6.24) между X_k , G_k и H_k имеет важное значение для организации вычислительного процесса. Очевидно, что последовательности G_k и H_k могут быть вычислены прямым ДПФ за $(N/2)^2$ операций каждая, а соединение их, дающее X_k , требует дополнительно N операций, что в сумме составляет $N + (N/2)^2$ операций (под операцией понимают комплексное умножение и сложение, которые состоят из четырех операций умножения над действительными числами, сложения и вычитания). В то же время, как указывалось, трудоемкость прямого вычисления X_k составляет N^2 операций, поэтому реализация (6.24) экономит вычисления почти в 2 раза при больших N . В выра-

жении (6.24) индекс k изменяется от 0 до $N - 1$. Однако G_k и H_k имеют период $N/2$ и вычисляются только в диапазоне значений от 0 до $(N/2) - 1$. Поэтому при вычислении в диапазоне значений от 0 до $N - 1$ применяют выражения

$$X_k = \begin{cases} G_k + W^k H_k, & 0 \leq k \leq (N/2) - 1; \\ G_{k-N/2} + W^k H_{k-N/2}, & N/2 \leq k \leq N - 1. \end{cases} \quad (6.25)$$

На рис. 6.19 изображен граф восьмиточечного БПФ, приведенного к двум четырехточечным ДПФ с помощью прореживания по времени, который отражает соотношения между X_k , G_k и H_k в (6.25).

В случае, когда $N/2$ также делится на два, вычисление G_k и H_k может быть сокращено тем же способом, каким было уменьшено вы-

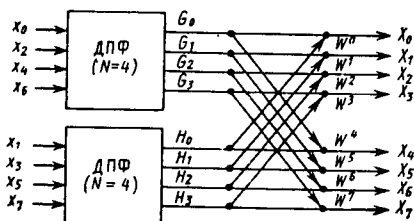


Рис. 6.19. Реализация восьмиточечного БПФ четырехточечными ДПФ

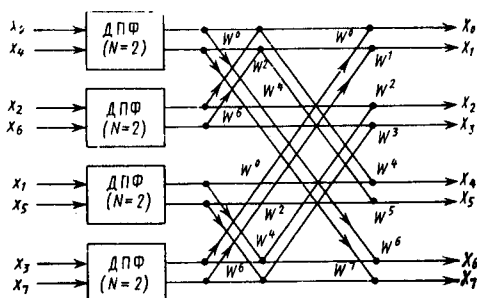


Рис. 6.20. Реализация восьмиточечного БПФ двухточечными ДПФ

числение X_k . Следуя этим путем, объем вычислений X_k можно уменьшить, если увеличивать число ДПФ от все меньшего числа точек со все большей экономией вычислительных операций. В этом и заключается сущность метода БПФ.

На рис. 6.20 два четырехточечных ДПФ из рис. 6.19 приведены к двум двухточечным, а на рис. 6.21 каждое двухточечное ДПФ — к двум одноточечным. На этих рисунках каждый узел соответствует операции сложения или вычитания, а каждая стрелка — операции умножения. Из рис. 6.21 видно, что при реализации восьмиточечного БПФ имеется 8×3 узлов и $2 \times 8 \times 3$ стрелок, соответствующих $N \log_2 N$ сложениям и $2 N \log_2 N$ умножениям. Половина умножений может быть опущена, поскольку линии передачи, обозначенные стрелкой без надписи, имеют коэффициент умножения, равный единице. Половину оставшихся умножений также можно исключить, если учесть, что $W^{N/2} = 1$.

Поэтому, если $N = 2^p$, то для вычисления ДПФ последовательно из N точек требуется $N \log_2 N$ сложений и $N/2 \log_2 N$ умножений.

Из графа организации вычислений коэффициентов Фурье X_k , представленного на рис. 6.21, видно, что после окончания каждой ите-

рации запись результатов может быть произведена в те же ячейки памяти, из которых были взяты исходные значения, так как они не будут больше нужны. Это позволяет организовать вычислительный процесс при существенной экономии в затратах оборудования.

Реализация БПФ с *прореживанием по частоте* производится следующим образом. Пусть последовательность x_l с четным числом точек N разделена на две последовательности по $N/2$ точек каждая, например q_l и h_l , где последовательность q_l образована из первых $N/2$ точек последовательности x_l , h_l — из последних $N/2$ точек последовательности x_l . Формально можно записать $q_l = x_l$, $h_l = x_{l+N/2}$, $l = 0, 1, 2, \dots, (N/2) - 1$.

Тогда N -точечное ДПФ может быть записано в виде

$$X_k = \sum_{l=0}^{N/2-1} (q_l W^{lk} + h_l W^{(l+N/2)k}) = \sum_{l=0}^{N/2-1} (q_l + e^{-j\pi k} h_l) W^{lk}. \quad (6.26)$$

Рассмотрим четные и нечетные отсчеты X_k отдельно (отсюда и название — прореживание по частоте). Заменив в (6.26) k на $2k$ и $2k+1$, получим соответственно

$$X_{2k} = \sum_{l=0}^{N/2-1} (q_l + h_l) (W^2)^{lk}, \quad (6.27)$$

$$X_{2k+1} = \sum_{l=0}^{N/2-1} [(q_l - h_l) W^l] (W^2)^{lk}. \quad (6.28)$$

Выражения (6.27) и (6.28) представляют собой $(N/2)$ -точечные ДПФ от функций $q_l + h_l$ и $(q_l - h_l) W^l$. Таким образом, определен другой порядок вычисления N -точечного ДПФ через результаты двух $(N/2)$ -точечных ДПФ. На рис. 6.22 представлено разложение восьмиточечного ДПФ на два четырехточечных. Аналогично можно выполнить деление на меньшее число ДПФ до тех пор, пока число точек в последовательности будет делиться на два. Как и в алгоритме с разде-

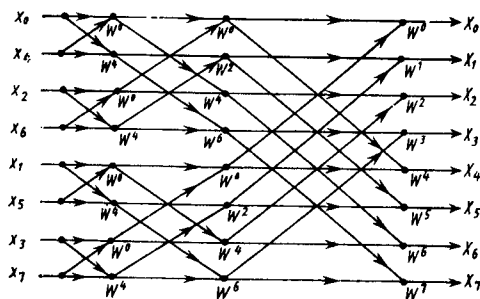


Рис. 6.21. Реализация восьмиточечного БПФ односточечными ДПФ

лением по времени, здесь количество комплексных сложений и умножений пропорционально $N \log_2 N$.

Следует отметить, что можно получить алгоритмы БПФ не только по основанию 2, но и по другим, в частности 4, 8 и т. д. В случае, когда N не является сложносоставным числом, массив входных отсчетов x_l дополняют нулями до той величины N , когда она станет равной требуемой.

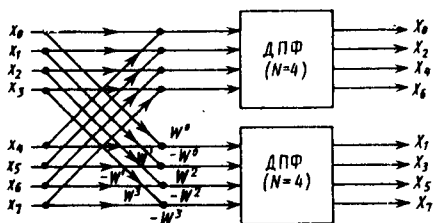


Рис. 6.22. Реализация восьмиточечного БПФ с прореживанием по частоте

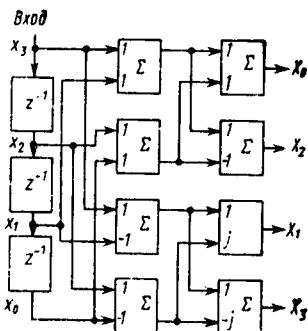


Рис. 6.23. Структурная схема устройства, реализующего алгоритм четырехточечного БПФ с прореживанием по частоте

На рис. 6.23 изображена структурная схема устройства, реализующего алгоритм четырехточечного БПФ с прореживанием по частоте. В данном устройстве операция умножения выполняется путем соответствующего соединения по входам сумматора действительных и мнимых чисел (операция умножения на j) и подачи их на суммирующие или вычитающие входы.

4. Цифровая фильтрация

Цифровой фильтр — устройство, осуществляющее преобразование одного дискретного сигнала x_n в другой дискретный сигнал y_n , причем сами сигналы x_n и y_n представляют собой двоичные цифровые коды.

В общем случае выходной сигнал цифрового фильтра в момент времени $t = nT$ определяется значением входного сигнала в тот же момент времени, а также значениями входных и выходных сигналов в предшествующие моменты времени:

$$y_n = f(x_n, x_{n-1}, \dots, x_{n-m}, y_{n-1}, \dots, y_{n-l}). \quad (6.29)$$

Если эта зависимость является линейной, то цифровой фильтр называется линейным, при этом выходная величина y_n определяется выражением

$$y_n = a_0 x_n + a_1 x_{n-1} + \dots + a_m x_{n-m} - b_1 y_{n-1} - \dots - b_l y_{n-l}. \quad (6.30)$$

Линейный дискретный фильтр обычно описывают с помощью передаточной функции, под которой понимают отношение z -преобразования выходной величины к z -преобразованию входной величины:

$$W_D^*(z) = Y^*(z)/X^*(z). \quad (6.31)$$

Выражение для передаточной функции можно получить из уравнения (6.30), если обе его части подвергнуть операции z -преобразования. Принимая во внимание, что

$$Z\{x_{n-k}\} = z^{-k} X^*(z) \text{ и } Z\{y_{n-k}\} = z^{-k} Y^*(z),$$

получаем

$$W_D^*(z) = \frac{a_0 + a_1 z^{-1} + \dots + a_m z^{-m}}{b_0 + b_1 z^{-1} + \dots + b_l z^{-l}}. \quad (6.32)$$

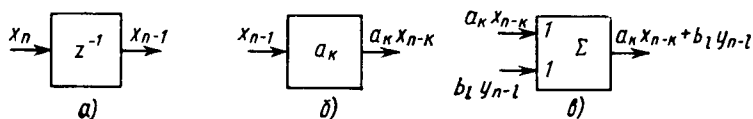


Рис. 6.24. Элементы цифровых фильтров:

a — задержки на период квантования; b — умножения на постоянный коэф. фициент; Σ — сложения

Из выражения (6.32) видно, что в общем случае передаточная функция линейного цифрового фильтра представляет собой отношение двух многочленов от z .

При построении цифровых фильтров существенным является вопрос их физической реализации, т. е. вопрос о том, любая ли передаточная функция вида (6.32) может быть реализована в виде схемы, построенной из физически осуществимых элементов, либо она может быть запрограммирована для микропроцессорной реализации фильтра.

Из уравнения (6.30) видно, что для получения y_n необходимо выполнить следующие операции.

1. Получение сигналов $x_{n-1}, \dots, x_{n-m}, y_{n-1}, \dots, y_{n-l}$. Эти сигналы можно получить из x_n и y_n , используя элементы задержки на один период квантования (рис. 6.24, а), в качестве которых могут служить запоминающие устройства. Последовательное включение нескольких ЗУ дает возможность задержать сигнал на произвольное число периодов квантования.

При микропроцессорной реализации цифровых фильтров для получения задержанных сигналов удобно использовать стек.

2. Умножение полученных на элементах задержки сигналов на постоянные коэффициенты a_k и b_k (рис. 6.24, б).

3. Суммирование полученных сигналов, что может быть осуществлено программным путем или на сумматорах (рис. 6.24, в).

Очевидно, единственным ограничением физической реализации разностного уравнения (6.30) является невозможность получения какого-либо из слагаемых правой части по той причине, что соответствующее слагаемое еще не появилось и, следовательно, не может быть получено путем его запоминания с целью задержки на заданное число периодов квантования. Таким образом, цифровой фильтр может быть физически реализован, если в правую часть уравнения (6.30) входят только настоящие и прошлые значения входной величины, но не входят будущие значения.

Покажем вид передаточной функции физически неосуществимого цифрового фильтра. Для этого в правую часть уравнения (6.30) должно входить слагаемое вида Ax_{n+s} , соответствующее входной величине, которая будет получена через s шагов квантования. Очевидно, что z -преобразование величины Ax_{n+s} равно $AX^*(z)z^s$. При этом передаточная функция принимает вид

$$W_D^*(z) = \frac{Az^s + a_0 + a_1 z^{-1} + \dots + a_m z^{-m}}{b_0 + b_1 z^{-1} + \dots + b_l z^{-l}}. \quad (6.33)$$

В качестве нормальной формы записи передаточной функции обычно принимают форму, при которой многочлены в числителе и знаменателе содержат только отрицательные степени z . Для приведения к нормальной форме разделим числитель и знаменатель передаточной функции (6.33) на z^s :

$$W_D^*(z) = \frac{A + a_0 z^{-s} + a_1 z^{-(s+1)} + \dots + a_m z^{-(s+m)}}{b_0 z^{-s} + b_1 z^{-(s+1)} + \dots + b_l z^{-(s+l)}}. \quad (6.34)$$

Особенностью передаточной функции (6.34) является отсутствие в знаменателе свободного члена. Это и является признаком физической нереализуемости цифрового фильтра.

При представлении передаточной функции цифрового фильтра в виде отношения многочленов, содержащих только положительные степени z , признаком физически реализуемого фильтра является выполнение условия, того, что степень многочлена, стоящего в числителе передаточной функции, не должна превышать степени многочлена, стоящего в ее знаменателе.

5. Цифровой спектральный анализ

Важной разновидностью цифровых устройств обработки сигналов являются цифровые спектральные анализаторы различных видов. Отметим, что спектральный анализ находит применение в задачах обнаружения сигнала, классификации типа сигнала, локализации источника сигнала, а также при решении траекторных задач.

Наиболее важными характеристиками спектрального анализа являются количество частот, на которых измеряется спектр сигнала, и разрешающая способность измерителя спектра.

При спектральном разложении некоторой функции времени предполагают, что она может быть представлена суммой синусоидальных компонент

$$x(t) = \sum_k C_k \sin(\omega_k t + \varphi_k). \quad (6.35)$$

При этом распределение амплитуд C_k по частотам называют амплитудным спектром функции $x(t)$ или просто спектром, а распределение фаз φ_k по частотам — спектром фаз. Как правило, практическому определению подлежит амплитудный спектр функции $x(t)$.

На практике цифровой спектральный анализ осуществляется по дискретной последовательности x_n , которая представляет собой выборки исходного непрерывного процесса $x(t)$, взятые для моментов времени $t = nT$, где T — период квантования. При этом довольно эффективно аппаратурный непрерывный анализ амплитудного спектра сигналов проводить согласно выражению

$$X_n(\omega_i) = \left| \sum_{k=0}^{\infty} \omega_k x_{n-k} e^{-j\omega_i T(n-k)} \right|, \quad (6.36)$$

где $\omega_k = \omega[kT]$ — весовая функция, $k = 0, 1, 2, \dots$

Величина $X_n(\omega_i)$ для случая, когда весовая функция $\omega[kT]$ отлична от нуля на бесконечном отрезке времени, интерпретируется как оценка модуля текущего спектра сигнала $x(t)$, определяемого выражением

$$X_t(j\omega) = \int_0^t x(t) e^{-j\omega t} dt,$$

а для случая, когда весовая функция $\omega[kT]$ отлична от нуля лишь на конечном интервале времени, — как оценка модуля мгновенного спектра

$$X_T(j\omega) = \int_{t-T}^t x(t) e^{-j\omega t} dt.$$

С другой стороны, вычисление оценки (6.36) можно рассматривать как осуществление полосовой амплитудно-частотной фильтрации спектра исследуемого сигнала, причем частотная характеристика устройства, вычисляющего данную оценку, будет определяться видом весовой функции $\omega[kT]$.

Аппаратурный анализ спектра сигнала подразделяют на последовательный, параллельный и смешанный. Последовательный анализ проводится в случае, если имеется перестраиваемый по резонансной частоте анализатор. При этом время анализа заданного диапазона частот спектра сигнала будет равно $T_{\Sigma} = NT_a$, где N — число полос, на которые разбивается анализируемый диапазон частот спектра сигнала; T_a — время сигнала в одной частотной полосе. Недостатками последо-

вательного метода анализа являются невозможность проведения текущего анализа спектра сигнала, непрерывно поступающего на вход вычислителя, а также большое общее время анализа.

Параллельный анализ предполагает одновременную обработку сигнала для всех анализируемых участков спектра и, следовательно, $T_{\Sigma} = T_a$. При этом возможно проводить текущий анализ спектра сигнала.

Цифровой спектральный анализ осуществляется в режимах «скачущего» и «скользящего» окна. Техническая реализация может быть осуществлена с помощью устройств, выполняющих как параллельный, так и последовательный спектральный анализ. В первом случае необходимы существенные затраты оборудования, в то время как последовательный спектральный анализ требует выполнения условий эргодичности и стационарности анализируемых случайных процессов. Организация режимов «скачущее» и «скользящее» окно не зависит от вида используемого вычислителя, а определяется лишь алгоритмом обработки.

6. Микропроцессорные средства цифровой обработки сигналов

Анализ методов и алгоритмов цифровой обработки сигналов показывает, что алгоритмы реализации ДПФ, БПФ, цифровой фильтрации и цифрового спектрального анализа хорошо подвергаются распараллеливанию. При этом основные алгоритмы и операции цифровой обработки сигналов могут быть эффективно реализованы на функционально законченных узлах в микропроцессорном исполнении.

В настоящее время имеется широкая номенклатура зарубежных и отечественных микропроцессорных средств для цифровой обработки сигналов. В качестве примера рассмотрим архитектурно-структурные особенности построения микропроцессора К1815ВФЗ.

Микропроцессор К1815ВФЗ для цифровой обработки сигналов. Однокристалльный микропроцессор К1815ВФЗ предназначен для построения устройств быстрого преобразования Фурье, цифровых фильтров и систем цифровой обработки сигналов на их основе. Машинный язык микропроцессора является языком высокого уровня, а набор команд обеспечивает выполнение таких крупных операций, как БПФ, умножение и сложение комплексных чисел, а также операций не рекурсивной и рекурсивной фильтрации, вычисления полиномов и ряда логических операций.

В основу построения микропроцессора положены идеи реализации крупных операций методами программирования структуры, а также способ обработки цифровой информации старшими разрядами вперед.

Структурная схема рассматриваемого микропроцессора изображена на рис. 6.25. Микропроцессор состоит из трех блоков: блока арифметико-логических операций БАЛО, блока умножения БУ и блока управления операциями БУО. В МП имеется пять информационных входов (LA, LB, LC, LD, LG) и два информационных выхода (LS, LP), что обеспечивает одновременное выполнение арифметико-логи-

ческих операций над пятью операндами (A, B, C, D, G) с одновременной выдачей двух результатов (S и P). Совокупность этих операций представляет собой унифицированную часть крупных операций. Операнды и результаты в МП могут иметь произвольную разрядность, точность умножения МП составляет 26 двоичных разрядов.

БАЛО в зависимости от заданного кода команды F (вход LF БУО) осуществляет сложение или вычитание операндов A и B и выдает на выход либо их сумму (разность) с взятием (без взятия) модуля, либо больший из операндов. БАЛО работает по конвейерному принципу. Операнды A, B в последовательном коде, начиная со старших разрядов, поступают одновременно на входы LA, LB и через одноктактные приемные регистры PrA и PrB подаются на одноразрядное АЛУ. С выхода АЛУ результат S может поступать на выход LS через одноктактный выходной регистр PrS или через один или два шеститактных регистра задержки результата Prd. Пары операндов A, B могут следовать друг за другом через необходимые промежутки времени или потоком. Управление масштабированием и задержкой выдачи результата осуществляется на регистрах Prd и PrS сигналами управления АЛУ (УАЛУ) и сигналами управления PrS (УПС), поступающими из БУО.

Блок умножения БУ обеспечивает получение произведения суммы (разности) операндов C и D на третий операнд G. Операнды подаются на выходы LC, LD и LG одновременно в форме последовательных кодов старшими разрядами вперед. БУ работает по конвейерному принципу. Операнды C, D проходят одноктактные регистры PrC, PrD и суммируются (вычитаются) на одноразрядном сумматоре Σ . Операнд G через четырехтактный регистр PrG поступает на вход умножителя У одновременно с результатом из сумматора ($C \pm D$). Произведение ($C \pm D$)G с умножителя поступает на выход LP через одноктактный регистр PrP. Масштабирование результата, если это необходимо по команде, осуществляется на умножителе и регистре PrG с помощью соответствующих сигналов управления, формируемых в БУО.

Блок БУО представляет собой регистр кода операций PrO с дешифратором ДШ. БУО по заданному коду команды F вырабатывает сиг-

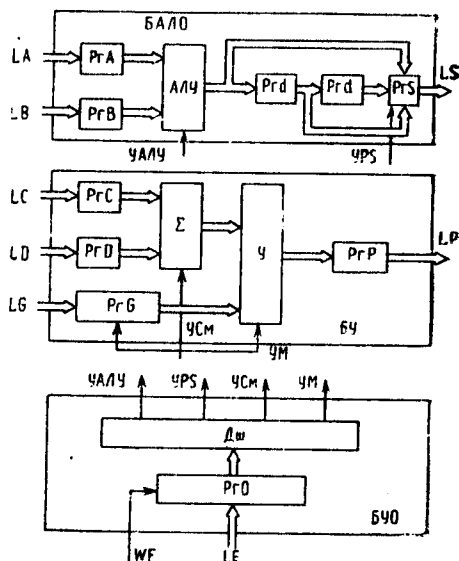


Рис. 6.25. Структурная схема микропроцессора K1815ВФ3

налы управления для всех узлов МП. Шестиразрядный код F подается на шину LF и по сигналу WF расшифровывает состояние PrO в необходимый набор сигналов.

Как было отмечено в этом параграфе, в цифровой обработке сигналов используются в основном операции БПФ и цифровой фильтрации, базу которых составляют умножение и сложение комплексных чисел.

Для осуществления операций сложения (вычитания) комплексных чисел $\underline{X}$ и $\underline{Y}$ в виде

$$\underline{Z} = \underline{X} \pm \underline{Y}; \operatorname{Re} \underline{Z} + j \operatorname{Im} \underline{Z} = \operatorname{Re} \underline{X} \pm \operatorname{Re} \underline{Y} + j (\operatorname{Im} \underline{X} \pm \operatorname{Im} \underline{Y})$$

и умножения комплексных чисел $\underline{U}$ и $\underline{V}$ в виде

$$\begin{aligned} \underline{W} = \underline{U} \underline{V}; \operatorname{Re} \underline{W} + j \operatorname{Im} \underline{W} = & \operatorname{Re} \underline{U} \operatorname{Re} \underline{V} - \\ & - \operatorname{Im} \underline{U} \operatorname{Im} \underline{V} + j (\operatorname{Re} \underline{U} \operatorname{Im} \underline{V} + \operatorname{Im} \underline{U} \operatorname{Re} \underline{V}) \end{aligned}$$

необходимо объединить четыре микропроцессора МП₁ ... МП₄ в один, как показано на рис. 6.26.

На БАЛО МП₁, МП₂ образуется сумма комплексных чисел, на БУ МП₁, МП₂ — произведения соответственно действительных и мнимых частей комплексных сомножителей, а на БУ МП₃, МП₄ — произведения действительной части на мнимую соответственно первого-второго и второго-первого сомножителей. Вычитание первой пары произведений осуществляется на БАЛО МП₃ — действительная часть комплексного произведения. Сложение второй пары произведений осуществляется на БАЛО МП₄ — мнимая часть комплексного результата.

Аналогично строится процессор (рис. 6.27), реализующий базовую операцию БПФ («бабочку») с прореживанием по частоте 2:

$$\begin{aligned} \underline{Z} = \operatorname{Re} \underline{Z} + j \operatorname{Im} \underline{Z} = \underline{X} + \underline{Y} = & \operatorname{Re} \underline{X} + \operatorname{Re} \underline{Y} + j (\operatorname{Im} \underline{X} + \operatorname{Im} \underline{Y}); \\ \underline{V} = \operatorname{Re} \underline{V} + j \operatorname{Im} \underline{V} = (\underline{X} - \underline{Y}) \underline{W} = & (\operatorname{Re} \underline{X} - \operatorname{Re} \underline{Y}) \operatorname{Re} \underline{W} - \\ & - (\operatorname{Im} \underline{X} - \operatorname{Im} \underline{Y}) \operatorname{Im} \underline{W} + j [(\operatorname{Re} \underline{X} - \operatorname{Re} \underline{Y}) \operatorname{Im} \underline{W} + \\ & + (\operatorname{Im} \underline{X} - \operatorname{Im} \underline{Y}) \operatorname{Re} \underline{W}]. \end{aligned}$$

Отметим, что структура, изображенная на рис. 6.27, является базовой для построения устройств, реализующих БПФ. Так, в частности, устройство восьмиточечного БПФ может быть построено на 12 процессорах БПФ и для его реализации требуется 48 корпусов МП K1815ВФ3. Производительность такого устройства для 20-разрядных чисел достигает 6 млн. базовых операций БПФ в секунду.

Применяя для построения цифровых фильтров соответствующие методы распараллеливания и каскадные схемы реализации фильтров высокого порядка, можно эффективно осуществлять разнообразную обработку сигналов.

Перспективы развития систем цифровой обработки сигналов. Средства цифровой обработки сигналов широко применяются в техни-

ке связи, радио, телевидении, а также в военной технике, где используются сигналы различной физической природы. При этом в большинстве практических задач полоса частот превышает 20 кГц и продолжает постоянно расширяться. Усложнение задач цифровой обработки сигналов привело к тому, что для их решения уже оказывается недостаточной производительность систем порядка нескольких миллионов операций в секунду.

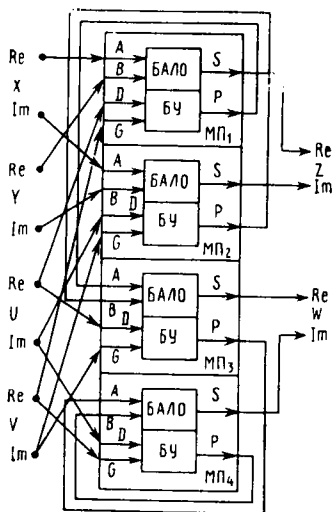


Рис. 6.26. Организация сложения и умножения комплексных чисел на МП К1815ВФ3

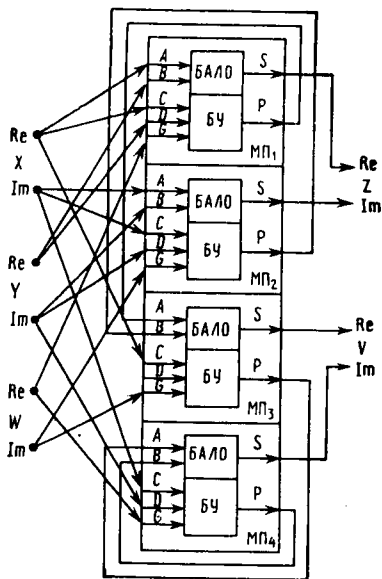


Рис. 6.27. Структура процессора БПФ на МП К1815ВФ3

Высокая производительность систем цифровой обработки сигналов может быть достигнута за счет технологических и архитектурных решений, позволяющих спроектировать быстродействующие БИС и СБИС со степенью интеграции до 200 тыс. вентилях на кристалл. В таких СБИС увеличение производительности реализации алгоритмов, цифровой обработки сигналов возможно благодаря аппаратной реализации операции умножения и операций над полем комплексных чисел путем организации параллельного выполнения операций адресации, переадресации и ветвлений программы, а также за счет применения аппаратно встроенных средств организации циклических вычислений, преобладающих в задачах цифровой обработки сигналов. Кроме того, в СБИС возможно создание специализированных процессоров, работающих над 32-разрядными словами в формате с плавающей точкой.

§ 6.8. МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА В БЫТОВОЙ РАДИОАППАРАТУРЕ

Использование микропроцессорных средств в бытовой радиотехнике обеспечивает наиболее оптимальные и экономичные режимы работы радиоаппаратуры, улучшенное восприятие информации, снижение массогабаритных показателей, различное сервисное обеспечение.

Радиоприемные устройства. Применение МП позволило реализовать надежные и недорогие цифровые системы настройки, исключив из приемника наименее надежные узлы — переменные конденсаторы и потенциометры. Такие системы являются модульными, поэтому сигнальные цепи можно размещать вблизи узлов настройки, а цепи управления и индикации приблизить к клавиатуре и индикатору, которые находятся на передней панели. Это позволяет уменьшить паразитное излучение и снизить расходы на монтаж и экранирование. В результате применения МП в цепях настройки уменьшаются размеры и повышается надежность приемника.

МПУ в радиоприемных устройствах управляют включением — выключением радиоприемника, осуществляют переключение диапазонов, регулируют громкость, приглушают звук при настройке и выполняют другие функции. В ряде приемников на МПУ возлагаются функции поиска частоты станции в обоих направлениях, управления синтезатором частот, управления цифровым вводом частот, запоминания и выдачи данных о частотах нескольких станций УКВ- и АМ-диапазонов, управления цифровой индикацией, коррекции частотных характеристик, переключения моно — стерео и др.

Магнитофоны. Применение микропроцессоров позволяет обеспечить автоматизацию и упрощение процессов настройки, автоматический контроль режимов работы и адаптацию магнитофонов к условиям эксплуатации, а использование МПУ — автоматическую подстройку характеристик магнитофона к используемой ленте. Это не только улучшает параметры тракта записи — воспроизведения, но и существенно сокращает общее время регулировки.

Все современные магнитофоны оборудуются системами автоматического поиска, которые обеспечивают нахождение нужного участка ленты, воспроизведение всей кассеты без перерыва и проигрывание отдельных фрагментов в заданной последовательности с повторением или без повторения. В перспективных магнитофонах поиск осуществляется не по интервалам между фрагментами, а по специальным кодовым послылкам, которые записываются перед каждым музыкальным отрывком. В этом случае даже при очень тихом звучании исключено ложное срабатывание. Отличительной особенностью таких магнитофонов является возможность автоматического поиска с любого места ленты без предварительной ее перемотки в начальное положение. Так, например, в системе CCS (Computer Code System) фирмы Philips (рис. 6.28) семиразрядные кодовые комбинации записываются на тех же звуковых дорожках, что и звуковые сигналы. Считывание кодовых

сигналов производится специальной головкой, а результаты обрабатываются микроЭВМ, которая определяет место нужного фрагмента записи. После этого лента остается на месте, если нужный участок совпадает с данным или ускоренно перематывается вперед или назад.

В комбайнах, где имеется магнитофон и радиоприемная часть, МПУ обеспечивает запись на ленту с радиоприемника заранее выбранной программы без участия и даже в отсутствие пользователя.

В отечественных микропроцессорах для контроля режимов работы различных механических устройств применяются базовые БИС К145ИК19. На основе БИС К145ИК19 создаются специализирован-

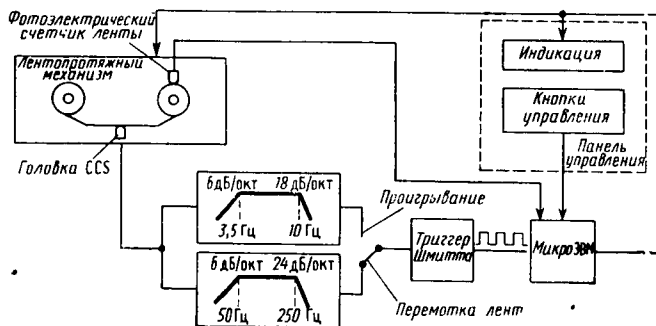


Рис. 6.28. Схема управления перематкой ленты магнитофона

ные микроконтроллеры, работающие по жесткой программе, занесенной во внутреннюю память микроконтроллера.

Следует отметить, что в БИС серии К145 разработан ряд однокристалльных микроконтроллеров с малой потребляемой мощностью, ориентированных на бытовое применение. Они выполнены по P -канальной МОП-технологии с напряжением питания 27 В и имеют относительно высокую помехозащищенность. Для БИС этой серии характерно, что логический «0» задается напряжением высокого уровня (0 ... 28 В), а логическая «1» — напряжением низкого уровня (—8 ... —27 В).

Для разнообразных практических применений в БИС серии К145 разработан ряд базовых однокристалльных микроЭВМ, в том числе микроЭВМ К145ИК19, ориентированная на решение задач управления с привязкой управляющих сигналов к текущему значению времени. Алгоритм работы в зависимости от конкретного исполнения вводится в память схемы при ее изготовлении. Схема БИС допускает подключение индикатора, помогающего контролировать ввод необходимых данных и результаты обработки.

В состав БИС К145ИК19 входят: ПЗУ программ объемом 128 20-разрядных слов; ПЗУ синхропрограмм (объемом 160×10), программы которого синхронизируют обработку информационного слова с движе-

нием информации в сдвиговых регистрах оперативной памяти; ПЗУ микрокоманд, осуществляющее непосредственное управление АЛУ; АЛУ с одноразрядным сумматором и набором вспомогательных регистров; устройство синхронизации и ряд регистров.

Для управления магнитофоном на базе БИС К145ИК19 созданы микроконтроллерные БИС К145ИК1906, К145ИК1913, К145ИК1914 и ряд других. В частности, при использовании БИС К145ИК1906 в составе магнитофона можно осуществить семь основных (останов, рабочий ход, перемотка вправо, перемотка влево, пауза, подготовка к записи, запись) и четыре дополнительных (автостоп, программный автостоп и две программы автоматического изменения режимов работы) режима работы. БИС К145ИК1913 в составе магнитофона позволяет обеспечить присвоение с помощью клавиатуры определенного номера (от 0 до 15) музыкальной программы с последующим поиском в режиме «Поиск» и прослушиванием. В режиме «Обзор» осуществляется последовательное прослушивание фонограммы в течение 15 с, затем происходит автоматическая перемотка до появления ближайшей паузы в фонограмме и т. д. При работе в режиме «Автоповтор» после воспроизведения последней фонограммы происходит переход к проигрыванию фонограммы, номер которой записан с помощью клавиатуры в первой ячейке памяти. БИС К145ИК1914 помимо использования в бытовых магнитофонах для контроля расхода ленты и управления лентопротяжным механизмом может служить для отсчета дискретных значений некоторого параметра с привязкой к реальному времени.

Телевизионные приемники. В телевизионных приемниках МПУ применяются для настройки и селекции каналов, а также для автоматического поддержания индивидуально установленных уровней громкости, яркости, цветовой насыщенности и контрастности. МПУ повышают надежность работы телевизионного приемника за счет исключения электромеханических деталей и значительного уменьшения числа дискретных элементов, гибкость управления, увеличивают эксплуатационные возможности и расширяют базу унификации различных моделей. Становится возможным производить модификацию функций управления простой доработкой программы МПУ.

На МПУ часто возлагается выполнение следующих функций:

- дешифрация команд дистанционного управления;
- автоматический поиск работающего передатчика и запоминание кода, соответствующего номеру его канала;
- запоминание кодов, соответствующих частотам передатчиков различных телевизионных программ, с возможностью их произвольного выбора;
- автоматический опрос телевизионных каналов с остановкой на боа-томе в данный момент канале;
- переключение на предыдущую программу и обратно;
- запоминание установленных пользователем уровней громкости, яркости, контрастности и цветовой насыщенности;
- индикация текущего времени и даты;

индикация номера программы и номера телевизионного канала;
переключение телевизора для работы с видеомagneтофоном;
программирование режима работы телевизионного приемника
(включение, переключение с программы на программу) на определенное время вперед;

автоматическое выключение звука при отсутствии работающего передатчика;

ступенчатое переключение тембра (речь — музыка).

Микропроцессорные системы управления имеются в отечественных телевизорах и в телевизорах фирм Zenith, Philips, SABA, Toshiba, Sony и др. В последнее время начинается внедрение способа управления голосом. Для его реализации в состав системы вводят анализатор речи, схему совпадения, которая преобразует квантированные характеристики голоса в цифровую форму, а также микроЭВМ. Характеристики голоса сравниваются с хранящимся в ЗУ эталоном. В случае совпадения происходит срабатывание системы. При управлении голосом возможна реализация выполнения таких функций, как включение, выключение, регулировка громкости и выбор телевизионного канала.

Использование микропроцессоров позволяет широко внедрить цифровые методы обработки сигналов и тем самым уменьшить потери качества изображения при всевозможных преобразованиях (ретрансляция, перезапись, коммутация и т.д.), повысить точность коррекции дефектов изображения, значительно снизить уровень контурных искажений путем повышения частоты кадров, а также реализовать дополнительные преобразования изображения.

Рассмотрим некоторые конкретные применения цифровых методов в телевидении.

Одним из первых блоков, в котором были внедрены цифровые методы обработки, явился селектор телевизионных каналов. Известно, что в обычных телевизорах применяются селекторы с механическим или сенсорным управлением, в которых реализуется плавная настройка в пределах одного канала или частотного диапазона, объединяющего несколько каналов. Такие селекторы сложны по схеме и конструкции, имеют низкие стабильность и надежность. Избавиться от этих недостатков удастся заменой традиционного гетеродина с непрерывной перестройкой частоты синтезатором сетки частот. В телевизорах применяется косвенный метод синтеза на основе фазовой автоподстройки частоты (ФАПЧ), так как он позволяет также довольно просто осуществить перестройку контуров с варикапами в усилительном тракте селектора каналов. Структурная схема синтезатора частот приведена на рис. 6.29. В нем необходимая сетка частот создается с помощью автогенератора АГ и делителя частоты ДЧ с изменяемым коэффициентом деления. Стабильность АГ обеспечивается системой ФАПЧ, в состав которой входят частотный детектор ЧД, кварцевый (опорный) генератор КГ и фильтр нижних частот ФНЧ. Коммутацию коэффициента деления ДЧ обеспечивает микроконтроллер МК, выполняющий роль

программируемого автомата системы управления. В цифровое запоминающее устройство МК вводятся данные о телевизионных каналах, задействованных в данном регионе, и путем нажатия соответствующей кнопки пульта управления ПУ включается требуемый канал или производится обзорный автопоиск работающих станций.

Достоинствами селектора каналов на основе синтезатора частоты являются:

особо точная настройка с долговременным сохранением параметров (обеспечивается кварцевым резонатором);

настройка на канал путем нажатия кнопки;

возможность настройки на канал без наличия телевизионного сигнала (настройка программируется заранее), а также путем использования программ различных телевизионных стандартов, в том числе по нестандартным каналам.

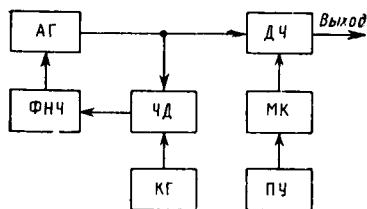


Рис. 6.29. Структурная схема синтезатора частот

Отечественная промышленность выпускает комплект из четырех ИМС для селектора каналов с синтезатором частоты:

КМ1814ВЕ5 — микроконтроллер, выполняющий функции коммутации, сохранения коэффициентов деления, управления индикацией;

К1015ХКЗ — синтезатор, включающий делитель частоты и частотный детектор;

КР558ХП2 — энергозависимое программируемое устройство памяти 16×24 , служащее для хранения информации и вызова отобранных телезрителем программ;

К193ПЦ1 — масштабный делитель частоты, снижающий частоту сигнала автогенератора с целью возможности обработки его в синтезаторе.

В современных телевизорах широкое применение находят микропроцессорные системы управления. Примером простой цифровой системы управления может служить схема, приведенная на рис. 6.30. В ней управление осуществляется всеволновым селектором каналов СК (через контроллер СК) и декодером телетекста. Под телетекстом понимают систему передачи дополнительной информации справочного характера по сети телевизионного вещания. Справочная информация передается в виде страниц текста. В декодере телетекста в процессе идентификации и синхронизации сигнала телетекста компаратор выделяет нужные страницы и загружает их в запоминающее устройство с произвольной выборкой ЗУПВ. С помощью знакогенератора на экране телевизора синтезируется текст выбранной страницы. Клавиатура пульта дистанционного управления ДУ позволяет производить поиск нужной информации в соответствии с принятой системой кодирования.

Структурная схема одного из вариантов блока декодирования видеосигналов приведена на рис. 6.31. Входной видеосигнал преобразуется во входном видеопроцессоре ВВП в последовательный цифровой поток, который через декодер телетекста ДТ заносится в ЗУПВ. Управление ДТ осуществляется микроконтроллером МК по двухпроводной цифровой шине (так называемая шина I²C — Inter Integral Circuit).

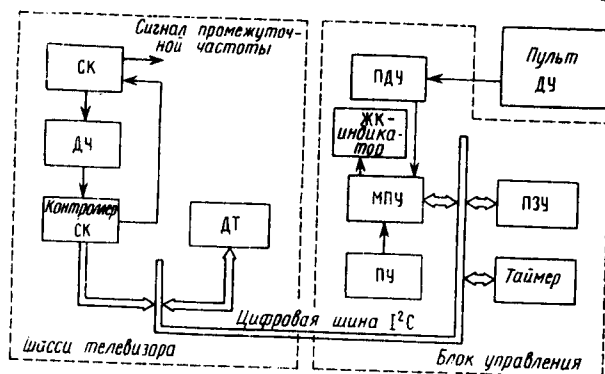


Рис. 6.30. Структурная схема микропроцессорной системы управления телевизором

В системе управления телевизором (см. рис. 6.30) команды переключения диапазонов и настройки на выбранный телевизионный канал передаются по цифровой шине I²C с блока управления. Номера каналов и программ индицируются на жидкокристаллических (ЖК) индикаторах. В состав блока управления в качестве основного элемента входит МПУ (или микроЭВМ), а также ПЗУ и таймер. Для приема команд с пульта дистанционного управления ДУ имеется приемник дистанционного управления ПДУ. В телевизор встроена панель управления ПУ.

Следует отметить, что развитие цифровых методов обработки сигналов привело к созданию цифрового телевидения на базе широкого использования микропроцессоров и микроЭВМ.

Современный телевизор становится ядром домашнего информационного и развлекательного центра.

С помощью встроенных микроЭВМ в бытовой радиоаппаратуре можно объединить домашнюю технику и комплексы, что позволяет лучше расходовать ресурсы аппаратуры, исключить некоторые дублирующие звенья (например, акустические системы, дисплей «домашней»

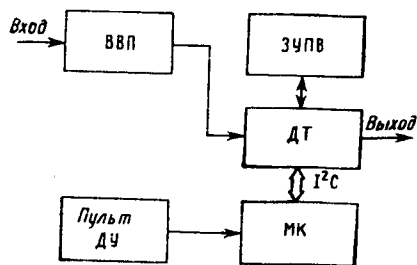


Рис. 6.31. Структурная схема блока декодирования видеосигналов

ЭВМ, органы настройки и регулировки), сократить число проводных соединений.

На основе однокристалльных микроЭВМ серий К1814, К1820, К1816, а также микроЭВМ других типов возможно построение комплексов бытовых РЭС. При этом каждое из устройств комплекса должно содержать управляющую систему на основе микроЭВМ, обеспечивающую работу аппаратуры в автономном режиме или от центральной микроЭВМ комплекса, которая выполняет функции программируемого таймера и диспетчера. Диспетчер определяет необходимый для выполнения данного режима набор аппаратуры, формирует и рассылает исполнительным устройствам команды управления и коммутации сигнальных цепей.

Контрольные вопросы и задания

1. Каковы принципиальные особенности функционирования МПУ в составе микропроцессорных РЭС?

2. Почему целесообразно выбирать в качестве элементной базы для микропроцессорных РЭС МПК БИС повышенной производительности по сравнению с требуемой?

3. Поясните понятие «укладка» и «компонента укладки» на примерах алгоритмов функционирования микропроцессорных РЭС.

4. С чем связана необходимость интенсификации загрузки регистров СОЗУ специализированных МПУ?

5. Почему встроенные МПУ позволяют повысить точность измерительных приборов?

6. Приведите дополнительные примеры применения встроенных МПУ в составе измерительных приборов.

7. Объясните назначение и особенности функционирования микроЭВМ в составе коммутационной аппаратуры сетей связи.

8. Опишите структурные особенности алгоритмов ДПФ и БПФ. За счет каких свойств эти алгоритмы допускают распараллеливание при их реализации?

9. Сформулируйте условия физической реализации цифровых фильтров.

10. Поясните особенности структурной реализации микропроцессоров цифровой обработки сигналов.

11. Перечислите основные функции микропроцессоров в бытовых радиоприемниках.

12. Чем вызвана необходимость создания специальных МПК БИС для бытовой радиоаппаратуры?

МИКРОПРОЦЕССОРЫ И МИКРОЭВМ В СИСТЕМАХ
КОНТРОЛЯ РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

§ 7.1. ЗАДАЧИ АВТОМАТИЗИРОВАННОГО КОНТРОЛЯ РЭС

Назначение автоматизированных систем контроля РЭС. Автоматизированные системы контроля (АСК) РЭС предназначены для проверки радиоэлектронного оборудования сложных систем различного назначения, в частности летательных аппаратов, корабельной аппаратуры, радиотехнических комплексов, систем автоматизированного управления и т.д.

Увеличение сложности функций, возлагаемых на РЭС, ужесточающиеся требования к достоверности и времени контроля вызвали необходимость разработки и создания автоматизированных комплексов контроля объектов в реальном времени. На автоматизированные системы контроля возлагаются задачи поиска неисправностей и установления причин отказов, сокращение общих трудозатрат на техническое обслуживание объекта контроля и времени подготовки техники к работе, повышения достоверности результатов контроля, коэффициента использования технического ресурса объектов контроля, а также слежения за аварийными параметрами и др.

Обобщенная структурная схема АСК представлена на рис. 7.1. В состав аппаратуры АСК входят системы сбора, преобразования и обработки информации о контролируемых параметрах РЭС, по которым можно судить о техническом состоянии объекта контроля на этапе его изготовления или эксплуатации. Под техническим состоянием понимают совокупность свойств объекта контроля, подверженных изменению в процессе производства или эксплуатации. Описание технических состояний объекта осуществляется конечной совокупностью параметров $X = (x_1, \dots, x_n)$. Каждому сочетанию значений контролируемых параметров X_k соответствует точечная параметрическая оценка технического состояния.

К внешним и внутренним возмущающим факторам $\xi(t)$, под воздействием которых изменяется техническое состояние объекта контроля, можно отнести изменение климатических условий, старение с течением времени, операции регулировки и настройки в ходе изготовления или ремонта, замену отказавших элементов и т.п.

Система сбора и преобразования информации выдает значение вектора контролируемых параметров X_k в систему обработки информации, где принимается решение о техническом состоянии объекта, а также о необходимости и характере управляющих воздействий Y_k , позволяющих удерживать значения контролируемых параметров в области допустимых значений S_k .

В автоматизированных системах контроля информация о состоянии объекта контроля поступает от системы измерительных датчиков. В АСК первого поколения в качестве элементной базы применялась дискретная полупроводниковая техника, а получение количественных характеристик при контроле осуществлялось с помощью специализированных устройств, работающих по жесткой программе.

С созданием управляющих вычислительных машин («Днепр», «Электроника 100», М-6000, М-400 и др.) оказался возможным переход

от выполнения ряда операций измерения и обработки информации аппаратным путем по жесткой программе к их выполнению по изменяющимся программам и варьированию программным путем в процессе работы структуры и состава работающих функциональных устройств системы. Объединение измерительной части и ЭВМ привело к появлению второго поколения АСК, которые строились по централизованному принципу.

Широкое внедрение средств микропроцессорной техники привело к появлению автоматизированных систем контроля третьего поколения. В этих системах с помощью микропроцессорных средств выполняются обработка и промежуточное хранение информации в местах

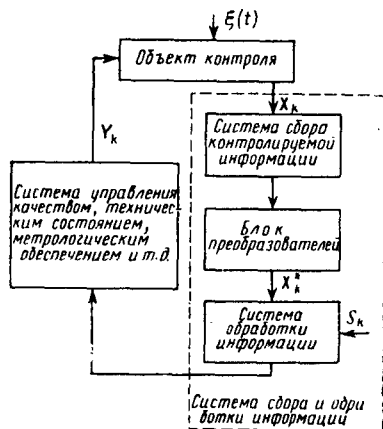


Рис. 7.1. Обобщенная структурная схема автоматизированной системы контроля

ее получения. Из-за распределенного характера процессов сбора информации, ее обработки и хранения эти системы называются децентрализованными. В составе АСК нашли широкое применение мини- и микроЭВМ, микропроцессорные системы, универсальные коммутаторы, преобразователи «аналог — код», «код — аналог» и другое оборудование. В современных АСК широко используются магистральные линии связи, уровень стандартизации которых доведен до государственных и международных стандартов. Таким образом, автоматизированная система контроля РЭС представляет собой комплекс аппаратных и программных средств, предназначенный для осуществления контроля сложного радиоэлектронного оборудования. Наличие в составе современной АСК развитого программного обеспечения и ЭВМ позволяет реализовать различные исполнения программ контроля, которые отличаются объемом функций, выполняемых человеком-оператором.

Анализ алгоритмов контроля РЭС. При оценке работоспособности радиоэлектронных средств возможны следующие методы контроля: качественный функциональный контроль;

количественный допусковый контроль;
диагностический допусковый контроль;
профилактический контроль.

При качественном *функциональном контроле* на вход проверяемой аппаратуры подается возмущающий сигнал, а работоспособность оценивается либо по срабатыванию исполнительных устройств, либо по отображениям сигналов на рабочих индикаторах проверяемой аппаратуры. При функциональном контроле количественные характеристики выходных параметров контролируемой аппаратуры не изменяются, поэтому данный метод контроля обладает ограниченной достоверностью, поскольку по его результатам нельзя оценить степень соответствия характеристик аппаратуры требованиям технического задания или паспортным данным.

Количественный допусковый контроль осуществляется путем количественных измерений выходных параметров РЭС по принципу «в норме — не в норме» или «меньше — норма — больше». Он получил широкое распространение при периодическом контроле сложных РЭС длительного использования.

Диагностический допусковый контроль обеспечивает обнаружение неисправных узлов и блоков РЭС, являющихся причиной отказов, с целью их замены на исправные.

Профилактический контроль обеспечивает проверку выходных параметров контролируемых РЭС и их основных блоков с целью определения временного дрейфа параметров и обнаружения с заданной вероятностью узла или блока, являющегося причиной отказа на данном временном интервале.

Контролируемые параметры РЭС представляют собой, как правило, стационарные случайные последовательности, и повышение достоверности их измерений можно свести к решению таких задач:

учету и исключению систематических ошибок путем введения поправок на нелинейность датчиков (тарировка), а также учету шкалы измерений параметра (калибровка);

фильтрации случайных погрешностей, вносимых датчиком и элементами тракта измерения;

отработке аномальных ошибок.

Эти задачи составляют класс задач *предварительной обработки информации*. Кратко поясним основные задачи предварительной обработки.

Тарировка заключается в приведении измеренных относительных значений параметров к их абсолютным значениям путем введения поправок, представляющих собой тарировочные характеристики датчика. В общем случае тарировочная характеристика имеет нелинейный характер и при обработке в микроЭВМ (или МПС) хранится в памяти в табличном виде.

Отбраковка аномальных измерений представляет собой задачу исключения ложных выбросов при анализе измерений, характеризующих поведение данного параметра. В зависимости от характера пове-

дения параметра и способов получения данных о нем можно использовать статистические либо детерминированные методы отбраковки.

Калибровка применяется для уменьшения требований к стабильности измерительного тракта в процессе работы путем передачи по отдельным каналам калибровочных сигналов от датчиков калибровочных направлений, которые позволяют определить коэффициент передачи измерительного тракта.

Фильтрация необходима для повышения точности и достоверности измерений путем ослабления случайной помехи. При обработке данных задача сглаживания реализуется путем аппроксимации параметра на интервале приближения какой-либо функцией: степенными полиномами, рядами Фурье, полиномами Чебышева и др. При выполнении операции сглаживания целесообразно одновременно проводить отбраковку аномальных измерений.

Помимо задач предварительной обработки информации аппаратура контроля РЭС должна обеспечивать также решение косвенных и организационных задач.

Косвенные задачи служат для измерения отдельных параметров косвенным путем, использованием значений их производных или составляющих. Это необходимо для удовлетворения требований обеспечения простоты и надежности компонентов системы контроля, унификации ее блоков и узлов путем сокращения номенклатуры датчиков и сигналов от них за счет исключения некоторого количества специализированных компонентов.

К организационным задачам относятся организация процесса контроля РЭС, формирование определенной последовательности адресов измеряемых параметров, согласование потоков данных, самоконтроль, диагностика и реконфигурация, связь с оператором и др.

Процесс обработки контролируемой информации с целью получения с заданной вероятностью оценки технического состояния контролируемых РЭС является *алгоритмическим*. Алгоритм функционирования автоматизированной системы контроля представляет собой совокупность частных алгоритмов преобразования, распределения, передачи, арифметической и логической обработки информации, а также алгоритмов представления информации в требуемой форме и управления работой всех устройств в процессе выполнения операции по оценке технического состояния контролируемых РЭС.

Способы и правила получения, преобразования, передачи и обработки информации в процессе контроля весьма разнообразны и качественно отличаются друг от друга. Вследствие этого на практике для решения задач контроля существует довольно много вариантов алгоритмов, которые дают похожий результат различными способами и в то же время отличаются друг от друга количеством операций, временем их выполнения, точностью и, следовательно, эффективностью.

Рассмотрим основные классификационные признаки алгоритмов контроля.

По числу различаемых состояний каждого контролируемого параметра известны три разновидности алгоритмов: «годен — не годен» (анализ двух состояний); «меньше—норма—больше» (анализ трех состояний); «меньше на сколько — норма с каким запасом — больше на сколько» (анализ многих состояний). Они просты в реализации, однако первые два алгоритма нельзя использовать при прогнозе технического состояния контролируемой аппаратуры.

По виду обслуживания объекта различают непрерывные, периодические и разовые алгоритмы.

Непрерывные алгоритмы присущи встроенным системам контроля сложных автоматически восстанавливаемых РЭС. Их особенность состоит в том, чтобы, не прерывая функционирования объекта контроля, обнаруживать неисправности и автоматически включать резерв или проводить регулировку вышедших за установленный допуск параметров. Периодические алгоритмы используются для периодического контроля сложных РЭС, входящих в состав летательных аппаратов, космических объектов и другой техники. Как правило, в состав аппаратуры периодического контроля входят ЭВМ. Разовые алгоритмы применяются для контроля уникальных высоконадежных объектов и реализуются на основе нескольких ЭВМ.

По режимам работы объекта во время контроля алгоритмы контроля подразделяются на статические и динамические.

По порядку выполнения операций контроля алгоритмы делятся на циклические, адресные и комбинированные. В циклических алгоритмах заранее установлена последовательность опроса датчиков (контрольных точек), которая в процессе функционирования не меняется. Адресные (приоритетные) алгоритмы позволяют производить опрос контролируемых параметров в любой последовательности, зависящей от технического состояния контролируемых РЭС. Комбинированные алгоритмы содержат как адресные, так и циклические части.

Проблемы выбора оптимального алгоритма функционирования системы, а также рационального соотношения между программными и аппаратными методами его реализации являются определяющими при оптимальном синтезе аппаратуры автоматизированного контроля сложных РЭС.

Роль и место микропроцессоров в АСК. Непрерывное увеличение вычислительной мощности микропроцессоров, сопровождаемое столь же непрерывным снижением их стоимости, привело к тому, что МП стали рассматривать как универсальные базисные элементы для построения сложных вычислительных и управляющих систем, в том числе систем автоматизированного контроля РЭС.

В зависимости от сложности контролируемых РЭС и требований, предъявляемых к достоверности и времени контроля, в состав аппаратуры контроля могут входить ЭВМ различной производительности (обычно мини- или микроЭВМ). Для организации непрерывного контроля сложных РЭС возникает необходимость построения систем обработки информации на основе микропроцессорных систем.

Использование МПС в составе аппаратуры контроля направлено на достижение следующих основных целей:

повышение производительности системы благодаря параллельному выполнению независимых задач или частей одной задачи;

повышение надежности системы контроля путем реконфигурации структуры МП и перераспределения задач между ММ при отказах некоторых из них в целях сохранения работоспособности системы в целом, может быть, при ухудшенных, но допустимых показателях производительности или качества контроля;

обеспечение высокой гибкости системы, т. е. возможности изменения ее конфигурации в целях реализации различных алгоритмов одним и теми же аппаратными средствами, а также обеспечение таких свойств, тесно связанных с гибкостью, как модульность и легкая расширяемость системы.

Высокая надежность МПС достигается благодаря высокой надежности ее модулей, строящихся на основе микропроцессорных комплектов БИС, которые реализуют большое число функций, ранее выполнявшихся отдельными логическими схемами.

При проектировании МПС основную роль играют такие факторы, как структура и характеристики ММ, распределение задач по ММ и взаимодействие между ММ системы.

Под распределением задач по ММ понимают сопоставление с каждым ММ множества тех задач контроля, которые он должен выполнять в процессе функционирования системы. Задачи контроля РЭС могут быть независимыми или связанными между собой. Распределение задач может быть статическим (фиксированным) или динамическим. В первом случае распределение задач по ММ производится до начала функционирования системы контроля и в процессе работы системы перераспределения задач не предусматривается. Во втором случае возможно перераспределение задач контроля по ММ в процессе функционирования системы. Динамическое распределение задач может быть вызвано необходимостью в реконфигурации МПС при отказах отдельных ММ или изменении распределения нагрузки между ММ в ходе работы аппаратуры контроля.

Таким образом, если реализация алгоритмов контроля РЭС возможна на базе одной ЭВМ, микроЭВМ или однопроцессорного МПУ с требуемой достоверностью в реальное время, то выбирают однопроцессорный вариант аппаратуры АСК. В противном случае аппаратуру контроля строят по многопроцессорному принципу.

§ 7.2. ОРГАНИЗАЦИЯ ВНУТРИСИСТЕМНОГО ИНТЕРФЕЙСА В МПС

Назначение стандартных интерфейсов. При проектировании аппаратуры контроля РЭС следует учитывать требования, предъявляемые к организации информационных связей между компонентами МПС,

в частности требования унификации, необходимой пропускной способности, «наращиваемости» с целью подключения дополнительных устройств, логической гибкости и высокой надежности системы связей. Наиболее полно приведенным требованиям отвечает организация связей между компонентами МПС по принципу «каждый с каждым». Анализ ее различных модификаций (связь по симметричной схеме, через центральный коммутатор, по итеративной, радиальной или радиально-магистральной схемам, по магистральной схеме и др.) показывает, что следующие факторы свидетельствуют в пользу магистральной структуры: умеренные требования к внутренним связям; использование в основном блочного обмена; применение распределенного типа связей; оправданность центрального управления магистральной структурой с помощью аппаратных или программных средств; наличие многообразных возможностей для функционального и пространственного расширения в пределах коммуникационных возможностей магистрали без изменения каких-либо ее элементов.

Простота магистральной схемы, незначительное количество кабелей шин и требуемых контактных соединений в каждом устройстве делают данную организацию связей наиболее приспособленной к унификации, а также для реализации в БИС-технологии.

К недостаткам магистральной структуры следует отнести сложность реализации значительного количества связей и их произвольного распределения, сложность организации приоритетного обмена между устройствами, уязвимость устройства управления ИМ, использование длинных шинных соединений.

Сущность магистрально-модульного принципа построения аппаратуры заключается в том, что проектируемая МПС komponуется из стандартных и унифицированных ММ, функциональных блоков и узлов, функционально и конструктивно законченных, совместимых друг с другом.

Конструкция ММ обеспечивает возможность их механического объединения в соответствующем корпусе. Обычно модули выдвигаются по направляющим в корпус при одновременном сочленении соединителя на задней кромке модуля с ответной частью на задней стенке объединяющего корпуса. Электрическое и информационное объединение ММ осуществляется через соединитель по магистральному системному интерфейсу. Преимуществами магистрально-модульного принципа построения аппаратуры являются совместимость и гибкость. Совместимость дает возможность комплексировать МПС из ММ, приобретенных у различных изготовителей, за счет строгой стандартизации конструкции и интерфейса. Гибкость обеспечивает возможность быстрого построения, наращивания, модернизации или реконфигурации МПС. Модули объединяются в единую систему посредством стандартного интерфейса.

Стандартизации в интерфейсе обычно подлежат: форматы передаваемой информации; команды и состояния; состав и типы линий связи; алгоритм функционирования; передающие и приемные электронные

схемы; параметры сигналов и требования к ним; конструктивные решения.

При создании внутрисистемного интерфейса необходимо учитывать требования обеспечения заданного быстродействия обработки информации, минимизации стоимости, учета возможности и простоты наращиваемости МПС, диагностики системы и т. п. В настоящее время разработка интерфейсов, процедура их внедрения и расширения сферы действия превратились в своеобразную научно-техническую и экономическую стратегию.

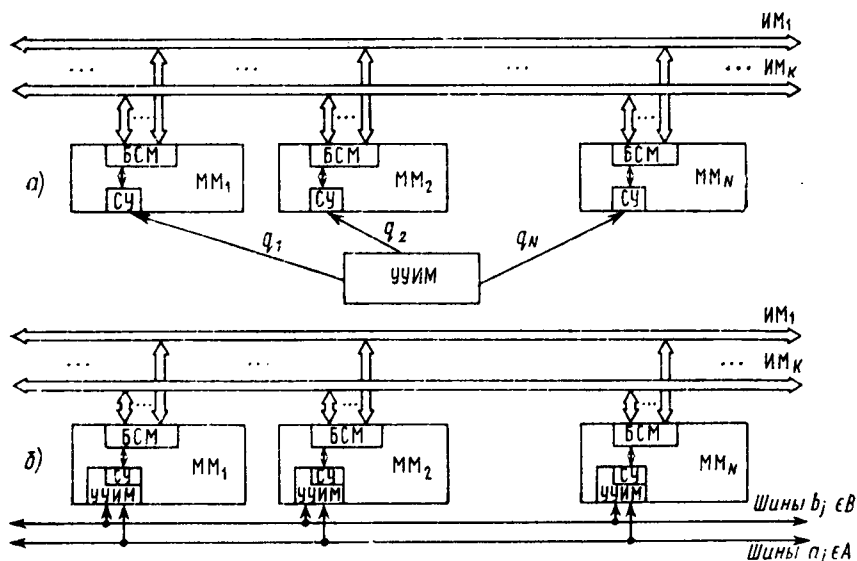


Рис. 7.2. Магистральная схема связей с централизованным (а) и децентрализованным (б) управлением

Магистральная схема связей. МПС с магистральной структурой системы связей состоит из набора модулей MM_i , $i = \overline{1, N}$, в качестве которых могут выступать процессорные модули, секции оперативной и постоянной памяти, мультиплексоры, функциональные преобразователи, контроллеры периферийных устройств и т. д. Варианты организации магистрального интерфейса представлены на рис. 7.2. Сопряжение устройств с информационной магистралью (ИМ) осуществляется через блок связи с магистралью (БСМ), в котором вырабатывается управляющая информация для устройства управления информационными магистралями (УУИМ) (рис. 7.2, а). Эта управляющая информация поступает в УУИМ либо параллельным кодом, либо набором одиночных сигналов и является достаточной для установления связи между любой парой устройств системы по соответствующему правилу. На

рис. 7.2, б представлена магистральная схема связей с децентрализованным управлением.

В зависимости от набора входных задач, интенсивности их поступления и сложности решения в состав МПС может входить различное число N_m комплекслируемых устройств. Следует отметить, что при увеличении этого числа возрастают физические размеры МПС и увеличивается длина ИМ, что приводит к снижению производительности системы связей и увеличению количества управляющих шин в ИМ.

Режимы активной и пассивной памяти. При организации обмена информацией между общей оперативной памятью и системой процессорных модулей возможна реализация режимов активной и пассивной памяти. В режиме активной памяти каждая секция ОЗУ может не только выполнять запросы от процессора на чтение или запись информации, но и самостоятельно обращаться со считываемой из ОЗУ информацией к процессору либо другим устройствам. В режиме пассивной памяти ОЗУ самостоятельно обращаться к другим устройствам не может.

В режиме активной памяти каждое устройство МПС имеет свой адрес, и при обращении процессора к любой секции памяти происходит выполнение следующей последовательности операций: захват ИМ, поиск нужной секции памяти, запоминание в секции ОЗУ адреса обратившегося процессора, отключение секции ОЗУ от ИМ, поиск и считывание нужной информации в памяти, обращение секции памяти к ИМ для передачи информации в запросивший ее процессор. В этом режиме в пределах цикла обращения процессора к памяти происходит совмещение работы ИМ по обслуживанию нескольких пар «процессор — секция ОЗУ».

В режиме пассивной памяти ИМ занята процессором в течение всего цикла обмена информацией между процессором и ОЗУ.

Пропускная способность F внутрисистемного интерфейса характеризует количество информационных сообщений N , передаваемых через него устройствами МПС в единицу времени. При последовательно-параллельном способе передачи информации одно сообщение передается Q последовательными посылками. Время $T_{ц}$, необходимое на передачу одной такой посылки, называют циклом передачи. При $Q = 1$ цикл передачи $T_{ц}$ есть не что иное, как длительность такта системы связей, определяемая выражением

$$T_{ц} = l_m u + \tau + t_y, \quad (7.1)$$

где l_m — максимальная длина кабельных шин связи между устройствами МПС; u — среднее время распространения сигнала по кабельным шинам; τ — длительность импульсного сигнала, передаваемого по ИМ; t_y — время, затрачиваемое схемой управления на установление связи между устройствами МПС.

В зависимости от способа управления ИМ значение величины $T_{ц}$ может изменяться. Так, например, при централизованном управлении ИМ работа схем управления обычно совмещается с передачей сообще-

ний и время t_y частично или полностью перекрывается со временем передачи ($l_{mu} + \tau$). При децентрализованном управлении возможность совмещения различных операций отсутствует, поэтому производительность системы связей несколько меньше, чем при централизованном управлении.

В режиме пассивной памяти производительность системы связей определяется выражением

$$F_n = [(2T_n + T_0)\gamma + T_n(1 - \gamma)]^{-1}, \quad (7.2)$$

где T_0 — длительность цикла обращения к памяти; γ и $(1 - \gamma)$ — соответственно относительные частоты обращения процессоров к памяти и друг к другу.

Производительность F_A системы связей, работающей в режиме активной памяти, выше F_n , поскольку величина T_0 в определении F_A участия не принимает. Однако вследствие того, что невозможно обеспечить четкую синхронизацию обращений устройств к каналу связи, исключаящую интерференцию этих обращений, производительность МПС с активной памятью будет снижаться из-за потерь времени на ожидание устройствами доступа в канал связи. Пропускная способность системы связей F_A в режиме активной памяти (в пересчете на одну посылку в паре «процессор—секция ОЗУ») определяется

$$F_A = T_n^{-1}. \quad (7.3)$$

При проектировании системы памяти в МПС следует учитывать, что режим активной памяти выгоден при $T_n/T_0 < 1$, а режим пассивной памяти — при $T_n/T_0 > 1$, что характерно для МПС с полупроводниковыми ОЗУ. При промежуточных значениях этого отношения выбор того или иного режима может определяться другими факторами (простотой контроля системы связи, сложностью технической реализации и др.).

В современных многопроцессорных системах контроля РЭС все большее распространение получает режим активной памяти.

Стандартные интерфейсы МПС. Современные МПС реализуются структурно и конструктивно как машинно-независимые системы. Принципы построения таких МПС определяются способами реализации машинонезависимых системных интерфейсов, имеющих в основном магистральную организацию.

Рассмотрим наиболее перспективные интерфейсы МПС, которые условно разделим на два больших класса: с отдельными и совмещенными шинами адреса и данных.

Интерфейсы семейства Multibus. К ним относятся интерфейсы с отдельными шинами адреса и данных различного уровня стандартизации, совместимые логически снизу вверх по числу используемых линий адреса (от 16 до 24), данных (от 8 до 16), арбитража и запроса прерываний. Эти интерфейсы имеют различные варианты конструктивной реализации. В отечественной технике широкое распространение получили варианты исходного интерфейса Multibus — интерфейсы И41 СМ ЭВМ и ИК1.

Интерфейс И41 предназначен для организации системы связей в микроЭВМ и многопроцессорных конфигурациях на базе МПК КР1810 и КР580. Шина интерфейса И41 состоит из линий передачи необходимой информации и управляющих сигналов, а также линий электропитания. Общее число линий — 73 (в том числе 20 линий адреса, 16 линий данных, 8 линий запросов прерываний). К шине можно подключить до 20 модулей с длиной связей не более 3 м, учитывая отводы от линий связи с передатчиком или приемником. Длительность цикла обмена ≥ 200 нс. Единица обмена — слово длиной 1 или 2 байт. В обмене всегда участвуют два модуля, связанные между собой как «здатчик» (управляющий) и «исполнитель» (управляемый). Структура интерфейса допускает использование 8- и 16-разрядных задатчиков и исполнителей. Задатчики работают поочередно, каждый может формировать адресные сигналы и сигналы управления для соответствующего исполнителя. Одновременные запросы управления упорядочиваются арбитражем.

Наиболее очевидное применение интерфейса И41 — многопроцессорные системы и высокоскоростные устройства прямого доступа. На базе стандарта И41 выпускается набор модулей широкой номенклатуры для организации объектно-ориентированных АСУ ТП, микроЭВМ СМ1800 и СМ1810 различных модификаций.

Интерфейс ИК1 стандартизован в рамках систем Микродат (комплекс технических средств ЛИУС-2). Он содержит отдельные 8-разрядные шины данных и 19-разрядные шины адреса, может обеспечивать многопроцессорный режим. Запросы и разрешения доступа к магистрали передаются по индивидуальным линиям. На основе этого интерфейса создано несколько объектно-ориентированных систем для АСУ ТП различного назначения.

Одним из перспективных интерфейсов этого класса является интерфейс VME — bus, который предназначен для использования в МПС на базе МП типов М6800 и М68000. За основу принят стандарт IEEE-961 (Versabus). Интерфейс позволяет строить 8-, 16- и 32-разрядные МПС различного функционального назначения. Практически он не накладывает ограничений на число подключаемых процессоров и других устройств, обеспечивает программную и аппаратную совместимость на всех уровнях системы. В интерфейсе предусмотрены два уровня реализации: для малых и средних систем 16-разрядная шина данных и 23-разрядная шина адреса; для мощных систем 32-разрядная шина данных и 31-разрядная шина адреса. Использование отдельных шин адреса и данных при достаточно высокой частоте тактового генератора (до 16 МГц) обеспечивает не только высокую скорость передачи данных, но и достаточный запас на допуски разброса параметров стандартных передатчиков, печатного монтажа основной панели и согласующих регистров.

Интерфейсы с совмещенными шинами адреса и данных. Эти интерфейсы разработаны в последние годы и наиболее полно отвечают тре-

бованиям построения современных экономических МПС с расширенными возможностями.

Интерфейс Eurobus является национальным стандартом Великобритании и ориентирован на использование в МПС, построенных на базе МП M6800 и M68000. Стандарт обеспечивает построение систем, использующих 8-разрядные МП с передачей 8-разрядных данных и реализацией на малых Европлатах; 16-разрядные МП и несколько «интеллектуальных» периферийных устройств с логической связью и реализацией на средних Европлатах; 24-разрядные МПС третьего поколения и 32-разрядные системы следующего поколения. Интерфейс содержит интерфейсные БИС данных и управления, обеспечивающие хорошие динамические характеристики (время цикла 400 нс) и компактность размещения модулей.

Интерфейс IEEE P-896 является единым процессорно-независимым интерфейсом для промышленных МПС. В нем реализованы децентрализованный арбитраж по пяти магистральным линиям, совмещенные 32-разрядные линии адреса и данных с переменной разрядностью, последовательная магистраль, режим блочной передачи, асинхронный и синхронный принципы передачи данных, супервизор магистрали. Состав линий на магистрали оптимизирован.

Интерфейс Fastbus характеризуется высокой скоростью передачи данных в асинхронном и синхронном режимах (время цикла 0,1 мкс). Он имеет единый протокол операций во всей системе, в том числе возможность операций с несколькими исполнителями одновременно; 32-разрядное адресное пространство и 32-разрядные линии данных; гибкую логическую и географическую адресацию; децентрализованный арбитраж; модульность конструкции и надежное тестирование системы. Предназначен для использования в высокопроизводительных МПС сбора и обработки больших объемов информации.

Интерфейс МПС с совмещенными шинами адресов и данных (межмодульный параллельный интерфейс для внутрислатного и межслатного обмена) разработан на основе интерфейса Q-bus фирмы DEC. Он использует 16-разрядную шину, обеспечен производством МПК серий K1801, K1809, K581, K588, K1811 и др. В интерфейсе МПС адрес и данные передаются по одним и тем же шинам. Мультиплексирование адресов и данных снижает пропускную способность интерфейса, но значительно уменьшает число линий связи, упрощая и удешевляя шину.

§ 7.3. МОДЕЛЬ ВЫЧИСЛИТЕЛЬНОГО ЯДРА МПС

В системах оперативного контроля РЭС объемы информации, подлежащие обработке при контроле радиоэлектронных изделий различной сложности, сильно меняются. Это требует создания гибких модульных структур систем обработки информации, отличающихся производительностью, а следовательно, составом процессоров и секций памяти в вычислительном ядре МПС.

Под *вычислительным ядром* МПС понимают минимальную совокупность технических средств (процессоров, секций памяти, каналов внутрисистемной связи), необходимых для решения множества задач контроля в реальном времени.

Для определения оптимального соотношения (например, производительности или аппаратных затрат) между количеством процессоров, секций памяти и каналов связи рассмотрим модель процессора и модель вычислительного ядра МПС.

Модель процессора представлена на рис. 7.3, а, б. В процессоре может быть предусмотрен опережающий просмотр команд за счет расслоения ОЗУ с целью минимизации простоя АЛУ. Следует иметь в виду, что данный способ повышения производительности эффективен при длительности цикла работы процессора, и малом количестве ветвлений в программе обработки, но малоэффективен для коротких программ обработки измерительной информации. Входящее в состав процессора ПЗУ (или ППЗУ) обычно используется для хранения микропрограмм управления процессором и макрокоманд обработки поступающей информации. Блок связи с магистралью БСМ предназначен для подключения устройств к информационным магистралям внутрисистемного интерфейса.

Модель вычислительного ядра МПС изображена на рис. 7.4. На вход ядра от активного источника информации (объекта контроля) по общему каналу связи поступает множество $S = \{s_1, \dots, s_m\}$ сигналов — заявок на обработку поступающей информации. В дальнейшем под s_i будем подразумевать как заявку на решение i -й задачи, так и саму задачу. Для решения каждой задачи $s_i \in S$ имеется программа L_i , состоящая из n_{ij} команд (операций) j -го типа. Коэффициент сложности выполнения каждой операции по отношению к элементарной (простейшей) есть α_j ($\alpha_j > 1$). Элементарная операция реализуется процессором за время t .

Время t выполнения элементарной операции зависит от архитектуры МП и технологии его изготовления. В справочниках приводятся характеристики быстродействия для операций типа «регистр — регистр».

Определение времени выполнения арифметических операций производится в соответствии с форматами команд процессоров и организацией их выполнения. Так, например, в центральном процессорном элементе КР580ИК80А серии КР580 каждая команда требует для выборки и выполнения 1 ... 5 машинных циклов, каждый машинный цикл включает 3 ... 5 тактов, а каждый такт длится в течение одного

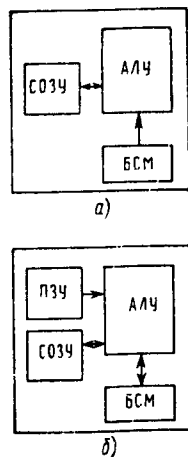


Рис. 7.3. Модель процессора без ПЗУ (а); с ПЗУ (б)

периода синхронизирующего сигнала (длительность такта при частоте 2 МГц составляет 0,5 мкс). В целом для выполнения команд в МП КР580ИК80А требуется от 4 (для команд без обращения к памяти типа арифметических операций с аккумуляторами или регистрами) до 18 машинных тактов (для наиболее сложных операций). При частоте синхронизирующего сигнала 2 МГц это означает, что для выполнения команды требуется от 2 до 9 мкс.

Следует отметить, что для ускорения выполнения ряда арифметических операций в состав различных МПК БИС введены специальные

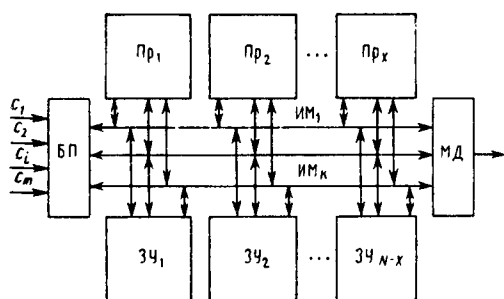


Рис. 7.4. Модель вычислительного ядра МПК

микросхемы, в частности в составе МПК КР588 имеется арифметический умножитель К588ВР2, предназначенный для аппаратного выполнения операции умножения двух 16-разрядных двоичных чисел за 2 мкс, в составе МПК КР1804 — схема ускоренного переноса КР1804ВР1 и т. д.

Входные заявки поступают в буферную память (БП) емкостью $Z=2^q$ ячеек и встают в общую очередь в порядке поступления.

Важным моментом при диспетчеризации входных информационных потоков является распределение заявок на обработку по процессорам. Возможны следующие способы распределения заявок на обработку по процессорам: по кольцу, групповое распределение, по мере освобождения процессоров. Последний способ является наиболее эффективным с точки зрения загрузки системы и минимизации потерь из-за простоя процессоров. В общем случае на последовательность измерений при контроле РЭС накладывается отношение порядка. Поэтому в оперативных системах контроля РЭС, как правило, реализуется дисциплина обслуживания FIFO — «первым пришел — первым обслужен». В процессе функционирования аппаратуры заявка, стоящая в очереди БП первой, направляется на обработку в любой свободный в данный момент времени процессор d_j , $j = \overline{1, X}$, который обслуживает ее до конца. Программы обработки могут храниться либо полностью в секциях ОЗУ b_k , $k = \overline{1, Y}$, либо частично в ОЗУ, частично в ПЗУ. Исходные данные всегда хранятся в ОЗУ системы, промежуточные данные — в СОЗУ процессора и только в случае ограниченного количества регистров СОЗУ и невозможности их наращивания — частично в ОЗУ системы. Обмен информацией между устройствами осуществляется по информационным магистралям. Результаты обработки выдаются из процессоров для вывода абонентам в мультимплексор данных (МД).

§ 7.4. АНАЛИЗ КОНФЛИКТНЫХ СИТУАЦИЙ В МПС

Природа конфликтных ситуаций в МПС. В магистрально-модульных МПС наличие общих ресурсов — общей магистрали и общей памяти — приводит к конфликтам. Конфликтные ситуации в МПС возникают при одновременном (в пределах одного и того же цикла памяти) обращении нескольких активных ММ к одной секции памяти или при одновременном (в пределах одного и того же цикла информационной магистрали) обращении нескольких активных ММ к одной и той же информационной магистрали. При большом числе таких конфликтов производительность снижается, что особенно заметно при сравнимых циклах работы ОЗУ и процессоров.

Следует отметить, что при $T_0 \geq T_{\text{ц}}$ (ОЗУ на ферритовых сердечниках) преобладают потери времени на разрешение конфликтных ситуаций при обращениях к секциям ОЗУ и с целью их уменьшения требуется увеличивать количество секций ОЗУ. При этом оказывается возможным пренебречь влиянием $T_{\text{ц}}$ и конфликтами в системе связей на общую производительность МПС. При значениях $T_{\text{ц}}$ близких к T_0 и тем более при $T_{\text{ц}} > T_0$ необходимо принимать меры для уменьшения потерь времени на разрешение конфликтных ситуаций при одновременных обращениях к каналам связи; увеличение количества секций ОЗУ выигрыша не дает. Это справедливо для систем с полупроводниковой памятью.

Для сокращения потерь времени на разрешение конфликтных ситуаций в магистрально-модульных МПС имеется несколько альтернатив. В частности, можно идти по пути построения структур с несколькими магистралями, в которых электрическая связь между ММ устанавливается с помощью многошинного коммутатора. Для сокращения потерь времени на разрешение конфликтов из-за общей памяти нужно разбивать ОЗУ на независимые секции, уменьшать количество обращений к ОЗУ за счет широкого использования принципа микропрограммирования, рационально распределять информацию в поле памяти, увеличивать соотношение цикла процессора к циклу ОЗУ, применять частичное дублирование информации в секциях ОЗУ.

Математическое ожидание (м. о.) времени ожидания $E [T_{\text{ож}}]$ на разрешение конфликтных ситуаций можно определить аналитически путем определения вероятностей возникновения всевозможных конфликтных ситуаций.

Пусть МПС комплексируется из N ММ (X однотипных процессоров и $Y = N - X$ секций оперативной памяти). Каждый процессор имеет свою локальную память (ОЗУ или ПЗУ), в которой хранятся микропрограммы, наиболее часто используемые программы и константы, а редко встречающиеся программы, константы, исходные и промежуточные данные записаны в Y секциях общей памяти.

Поступающие задачи распределяются по процессорам произвольным образом: задача, стоящая первой в очереди на обслуживание, поступает в любой освободившийся процессор.

Связь между процессорами и секциями общей памяти осуществляется через K шин с временным разделением, причем каждая шина может соединять любую пару «процессор — секция памяти», $K \leq \min \{X, Y\}$. Правило коммутации устанавливается арбитром. Существует множество различных правил коммутации, в частности такое: процессорам, запрашивающим одну и ту же секцию памяти, назначается одна и та же шина; если запрашиваемые секции памяти свободны, а все шины заняты, то процессорам назначаются шины в циклическом порядке. Порядок обслуживания запросов от процессоров секциям памяти — FIFO.

Поскольку любая поступающая задача может быть распределена на любой процессор и информация в секциях общей памяти занесена без учета специализации процессоров, будем считать, что обращение процессоров к секциям памяти асимптотически равновероятно, причем в один и тот же момент времени любой процессор может обратиться только к одной секции общей памяти. В стационарном режиме работы МПС полностью загружена и обращение процессоров к секциям памяти равновероятно. Вероятность p обращения процессора к секциям общей памяти определяется выражением

$$p = \frac{\sum_{i=1}^m g_i h_i}{\sum_{i=1}^m g_i (h_i + n_i)} \quad (7.4)$$

где g_i — относительная частота поступления задачи c_i ; h_i и n_i — соответственно число обращений процессора к общей и локальной памяти при решении задачи c_i ; m — число типов обрабатываемых задач.

Проанализируем потери производительности в МПС за счет наличия конфликтных ситуаций.

Конфликтные ситуации из-за общей памяти. Определим соотношение для оценки среднего времени ожидания $E[T_{ож}]$ доступа каждым процессором к секциям общей памяти в процессе решения поступившей задачи.

Методика оценки основывается на анализе всевозможных комбинаций взаимосвязей между процессорами и секциями памяти, поэтому в качестве математического аппарата для решения поставленной задачи будем использовать комбинаторный анализ.

Обозначим для некоторой конфликтной ситуации максимальное число процессоров, обратившихся одновременно к одной секции памяти, через k , $k = \overline{2, X}$. Тогда конфликтную ситуацию данного типа можно описать упорядоченным множеством

$$Z_k = (z_k, \dots, z_j, \dots, z_1),$$

где z_j — число групп, каждая из которых содержит j конфликтующих процессоров, $j = \overline{2, k}$; z_1 — число процессоров, обратившихся к общей памяти и не конфликтующих между собой.

Значения z_k , $k = \overline{1, X}$ определяются с помощью равенств

$$z_k = 1, \left\lfloor \frac{X}{k} \right\rfloor, z_j = 0, \left\lfloor \left(X - \sum_{i=j+1}^k iz_i \right) / j \right\rfloor, j = \overline{1, k-1}. \quad (7.5)$$

В (7.5) квадратные скобки означают выделение целой части числа, заключенного в них.

Конфликтные ситуации различных типов можно представить множествами (z_2, z_1) , (z_3, z_2, z_1) , ..., $(z_k, \dots, z_j, \dots, z_1)$, ..., $(z_X, \dots, z_j, \dots, z_1)$, причем для всех ситуаций должно выполняться условие

$$\sum_{j=1}^k iz_j \leq X. \quad (7.6)$$

учитываемое равенствами (7.5).

Пусть, например, в МПС число процессоров $X = 5$, число секций памяти $Y = 4$. При $k = 2$ $z_2 = 1, 2$, $z_1 = 0, 5 - 2z_2$. Тогда все воз-

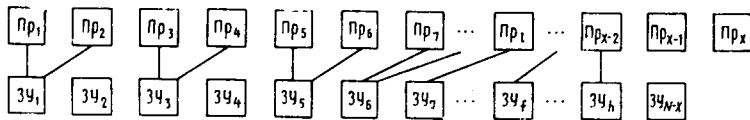


Рис. 7.5. Конфликтная ситуация типа $Z_2 = (4, X-10)$

можные конфликтные ситуации типа $Z_2 = (z_2, z_1)$ будут иметь вид $(1, 0)$, $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 0)$, $(2, 1)$. При $k = 3$ $z_3 = 1$, $z_2 = 0, 1$, $z_1 = 0$, $X - 2z_2 = 3$ и все конфликты типа $z_3 = (z_3, z_2, z_1)$ принимают вид $(1, 0, 0)$, $(1, 0, 1)$, $(1, 0, 2)$, $(1, 1, 0)$. При $k = 4$ $z_4 = 1$, $z_3 = 0$, $z_2 = 0$, $z_1 = 0, 1$ и имеются только две конфликтные ситуации типа $Z_4 = (z_4, z_3, z_2, z_1)$: $(1, 0, 0, 0)$ и $(1, 0, 0, 1)$. При $k = 5$ $z_5 = 1$, $z_4 = z_3 = z_2 = z_1 = 0$ и имеется только одна конфликтная ситуация типа $Z_5 = (z_5, z_4, z_3, z_2, z_1) : (1, 0, 0, 0, 0)$.

Выведем соотношение для оценки вероятности возникновения конфликтной ситуации типа Z_k . Для иллюстрации методики проанализируем конфликт типа $Z_2 = (4, X - 10)$, характеризующийся тем, что в произвольный момент времени в МПС, содержащей X процессоров, конфликтуют четыре пары процессоров, два процессора не обратились к общей памяти, а остальные к общей памяти обратились, но в конфликт из-за общих ресурсов не вступили (рис. 7.5).

Вероятность появления такой конфликтной ситуации ($p_{кс}$)

$$p_{кс} = \left(\frac{p}{Y} \right)^{z_1 + 2z_2} q^{X - z_1 - 2z_2},$$

где $q = 1 - p$ — вероятность обращения процессора к секциям локальной памяти: $z_1 = X - 10$ — число групп процессоров, обратившихся к общей памяти и не конфликтующих между собой (для данного случая $z_1 = X - 10$); z_2 — число групп, содержащих по два конфликтующих процессора (для данного случая $z_2 = 4$).

Для определения вероятности p_{z_2} возникновения конфликтных ситуаций рассматриваемого типа найдем их число.

Проанализируем все возможные комбинации конфликтных ситуаций данного типа между процессорами без учета нумерации секций памяти и при постоянных номерах необратившихся процессоров ($X - 1$, X). В исходной конфликтной ситуации (см. рис. 7.5) в первых двух группах поменяем местами процессоры так, чтобы группы стали содержать один процессор с новым номером. Число таких комбинаций равно C_3^1 . Произведем аналогичные действия в первых двух группах, заменяя последовательно в третьей группе номер одного процессора на первые четыре номера (1, 4). Число рассматриваемых комбинаций равно $C_3^1 C_5^1$. Проведем такие же операции в первых трех группах, заменяя последовательно в четвертой группе номер одного из процессоров на первые шесть номеров (1, 6). Общее число рассмотренных комбинаций равно $C_3^1 C_5^1 C_7^1$ или $\prod_{l=0}^{z_2-1} C_{2z_2-2l-1}^1$. Таким образом, найдено число

конфликтных ситуаций типа Z_2 ($4, X - 10$) с возможными группами z_2 при постоянных номерах остальных процессоров.

Число всех возможных комбинаций, которые отличаются хотя бы одним номером конфликтующего процессора, равно $C_{z_1+2z_2}^{2z_2}$, а число ситуаций, отличающихся хотя бы одним номером необратившегося процессора, $C_X^{X-z_1-2z_2}$.

Проанализируем исходную конфликтную ситуацию, но при обращении процессоров к другим секциям памяти. Число комбинаций групп с Y секциями равно $A_Y^{z_1+z_2}$.

Таким образом, число всех возможных конфликтных ситуаций рассматриваемого типа Z_2 ($4, X - 10$) равно

$$A_Y^{z_1+z_2} C_X^{X-z_1-2z_2} C_{z_1+2z_2}^{2z_2} \prod_{l=0}^{z_2-1} C_{2z_2-2l-1}^1,$$

а вероятность их возникновения

$$p_{z_2} = \left(\frac{p}{Y}\right)^{z_1+2z_2} q^{X-z_1-2z_2} A_Y^{z_1+z_2} \times \\ \times C_X^{X-z_1-2z_2} C_{z_1+2z_2}^{2z_2} \prod_{l=0}^{z_2-1} C_{2z_2-2l-1}^1 \quad (7.7)$$

при любых значениях z_1, z_2 .

Введем понятие вероятности возникновения конфликта между процессорами p_k , под которой будем понимать вероятность появления конфликтных ситуаций типа $(z_k, z_{k-1}, \dots, z_1)$ при

$$z_k = \overline{1, \left\lfloor \frac{X}{k} \right\rfloor}, \quad k = \overline{2, X}$$

и

$$z_j = \overline{0, \left\lfloor \frac{X - \sum_{i=j+1}^k iz_i}{j} \right\rfloor}, \quad j = \overline{1, k-1}.$$

Примем $k = 2$, тогда вероятность возникновения конфликта между двумя процессорами

$$\begin{aligned} p_2 &= \sum_{z_2=1}^{\left\lfloor \frac{X}{2} \right\rfloor} \sum_{z_1=0}^{X-2z_2} \left(\frac{p}{Y} \right)^{z_1+2z_2} q^{X-z_1-2z_2} A_Y^{z_1+2z_2} C_X^{X-z_1-2z_2} \times \\ &\times C_{z_1+2z_2}^{2z_2} \prod_{l=0}^{z_2-1} C_{2z_2-2l-1}^1. \end{aligned} \quad (7.8)$$

Аналогично при $k = 3$ вероятность появления конфликта между тремя процессорами

$$\begin{aligned} p_3 &= \sum_{z_3=1}^{\left\lfloor \frac{X}{3} \right\rfloor} \sum_{z_2=1}^{\left\lfloor \frac{X-3z_3}{2} \right\rfloor} \sum_{z_1=0}^{X-2z_2-3z_3} \left(\frac{p}{Y} \right)^{z_1+z_2+3z_3} q^{X-z_1-2z_2-3z_3} \times \\ &\times A_Y^{z_1+z_2+3z_3} C_X^{X-z_1-2z_2-3z_3} C_{z_1+2z_2}^{2z_2} \sum_{l=0}^{z_3-1} C_{2z_2-2l-1}^1 \times \\ &\times C_{z_1+2z_2+3z_3}^{3z_3} \prod_{l=1}^{z_3-1} C_{3z_3-3l-1}^2. \end{aligned} \quad (7.9)$$

Используя метод математической индукции, нетрудно получить следующее соотношение для оценки вероятности p_k :

$$\begin{aligned} p_k &= \sum_{z_k=1}^{a_k} \sum_{z_{k-1}=0}^{a_{k-1}} \dots \sum_{z_j=0}^{a_j} \dots \sum_{z_1=0}^{a_1} \left(\frac{p}{Y} \right)^b \times \\ &\times q^{X-b} A_Y^c C_X^{X-a} \prod_{i=2}^k C_d^{iz_i} \prod_{l=0}^{z_i-1} C_{iz_i-2l-1}^{i-1}, \end{aligned} \quad (7.10)$$

где

$$a_h = \left\lfloor \frac{X}{k} \right\rfloor; \quad a_j = \left\lfloor \frac{X - \sum_{i=j+1}^k iz_i}{j} \right\rfloor, \quad j = \overline{1, k-1};$$

$$b = \sum_{i=1}^k iz_i; \quad c = \sum_{i=1}^k z_i; \quad d = \sum_{n=1}^i nz_n.$$

В слагаемых формулы (7.10), содержащих $z_i = 0$, полагаем сомножители

$$C_d^{iz_i} \prod_{l=0}^{z_i-1} C_{iz_l}^{l+1} = 1.$$

На равенство (7.10) наложено ограничение $X - 1 \leq Y$, так как число групп обратившихся процессоров не должно превышать числа секций общей памяти. Если $X - 1 > Y$, то в формуле (7.10) слагаемые, у которых $\sum_{i=1}^k z_i > Y$, обращаются в нуль.

Потери времени на ожидание для каждого процессора, участвующего в конкретной ситуации типа $(z_k, z_{k-1}, \dots, z_j, \dots, z_1)$ определяются выражением

$$E|T_{z_k}| = \sum_{i=2}^k z_i \sum_{j=1}^{i-1} \frac{j \tau_{обм}}{X} = \frac{\tau_{обм}}{2X} \sum_{i=2}^k z_i i(i-1),$$

где $\tau_{обм}$ — время обмена единицей информации между процессором и секцией памяти.

Среднее время ожидания, необходимое на разрешение конфликтных ситуаций, определяется как

$$E|T_{ож}| = u \sum_{k=2}^X \sum_{z_k=1}^{a_k} \sum_{z_{k-1}=0}^{a_{k-1}} \dots \sum_{z_j=0}^{a_j} \dots \sum_{z_1=0}^{a_1} E|T_{z_k}| p_k, \quad (7.11)$$

где

$$u = \sum_{i=1}^m g_i h_i.$$

Анализируя выражение (7.11) для конкретных значений X и применяя метод математической индукции, нетрудно прийти к выводу, что все слагаемые в (7.11) имеют общий сомножитель

$$\left(\frac{p}{Y}\right)^2 A_Y^1 C_Y^2 \frac{\tau_{обм}}{X},$$

который можно вынести за скобки, при этом в скобках остается выражение, соответствующее биному $(p + q)^{X-2}$.

Таким образом, выражение (7.11) принимает вид

$$E[T_{ож}] = u \frac{\tau_{обм}}{X} \left(\frac{p}{Y} \right)^2 A_Y C_X^2 (p + q)^{X-2}$$

или

$$E[T_{ож}] = u \frac{X-1}{2Y} \tau_{обм} p^2. \quad (7.12)$$

Если все процессоры МПС не имеют локальной памяти, то $p = 1$ и

$$E[T_{ож}] = u \frac{X-1}{2Y} \tau_{обм}. \quad (7.13)$$

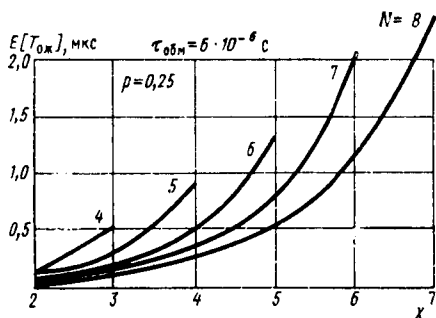


Рис. 7.6. Графики аналитических зависимостей $E[T_{ож}] = f(X)$ при различных N

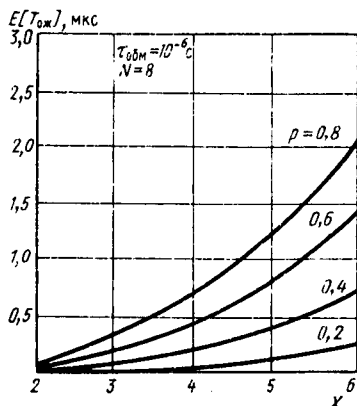


Рис. 7.7. Графики аналитических зависимостей $E[T_{ож}] = f(X)$ при различных вероятностях обращения процессоров к секциям памяти

На рис. 7.6 приведены графики аналитических зависимостей среднего времени ожидания, необходимого на разрешение конфликтных ситуаций в МПС, от числа процессоров в МПС. При этом МПС характеризуется такими параметрами: $p = 0,25$, $\tau_{обм} = 6$ мкс, $N = 8$.

3.8. Из приведенных графиков видно, что при увеличении числа процессоров в МПС при любом N среднее время ожидания $E[T_{ож}]$ возрастает, поскольку возрастает вероятность возникновения конфликтов между процессорами.

Характер изменения функции (7.12) при различных вероятностях обращения процессоров к секциям памяти показан на рис. 7.7.

Конфликтные ситуации из-за общей шины. Будем считать, что рассматриваемая МПС содержит K общих шин, $1 \leq K \leq \min\{X, Y\}$.

Нетрудно заметить, что при условии $\sum_{i=1}^k z_i \leq K$ конфликтные ситуации

возникают только из-за секций памяти, при $\sum_{i=1}^k z_i > K$, $K = 2, \overline{X}$

— как из-за общей памяти, так из-за общей шины, при $k = 1$ и $\rightarrow K$ — только из-за общей шины.

Оценим вероятность появления конфликтной ситуации z_1 из-за общих шин. Рассмотрим общий случай, когда число процессоров, обратившихся к общей памяти и не конфликтующих между собой, составляет z_1 , а число процессоров, не обратившихся к общей памяти, — $X - z_1$.

Число всех возможных комбинаций типа z_1 , каждая из которых отличается хотя бы одним номером обратившегося процессора, равно $C_X^{z_1}$ без учета нумерации секций памяти. При обращении процессоров к другим секциям памяти число комбинаций z_1 будет составлять $A_Y^{z_1}$. Таким образом, число комбинаций конфликтных ситуаций типа z_1 равно $A_Y^{z_1} C_X^{z_1}$, а вероятность возникновения ситуации z_1

$$p_1 = \left(\frac{p}{Y}\right)^{z_1} q^{X-z_1} A_Y^{z_1} C_X^{z_1}. \quad (7.14)$$

Вывод соотношения для определения вероятности p_k , $k = \overline{2, X}$ аналогичен выводу формулы (7.10). Нетрудно показать, что формула для определения вероятности возникновения любой конфликтной ситуации типа z_k , $k = \overline{1, X}$ имеет вид

$$p_k = \left(\frac{p}{Y}\right)^b q^{X-b} A_Y^b C_X^b \prod_{i=2}^k C_d^{iz_i} \prod_{l=1}^{z_i-1} C_{iz_i-l-1}^{i-l-1}. \quad (7.15)$$

При $k = 1$ формула (7.15) принимает вид (7.14).

При $k = 1$ в МПС используется одна шина с временным разделением и, следовательно, имеются только конфликты из-за общей шины. В этом случае величина $E|T_{\text{ож}}|$ определяется согласно выражению

$$E|T_{\text{ож}}| = u \frac{X-1}{2} p^2 \tau_{\text{обм}}. \quad (7.16)$$

В случае, когда процессоры не имеют локальной памяти, соотношение для оценки среднего времени ожидания принимает вид

$$E(T_{\text{ож}}) = u \sum_{k=1}^X \sum_{z_k=1}^{a_k} \sum_{z_{k-1}=0}^{a_{k-1}-1} \dots \sum_{z_j=0}^{a_j} \dots \sum_{z_2=0}^{a_2} E|T_{z_k}| p_h,$$

где $z_1 = X$ при $k = 1$ и $z_1 = X - \sum_{i=2}^k iz_i$ при $k = \overline{2, X}$.

Потери времени на ожидание $E|T_{z_k}|$ каждым процессором и вероятность p_k возникновения конфликтной ситуации z_k , $k = \overline{1, X}$ оцениваются аналогично выражениям (7.10), (7.12).

Вероятность возникновения любой конфликтной ситуации при обращении всех процессоров к общей памяти

$$p_k = Y^{-X} A_Y^c \prod_{i=2}^k C_d^{iz_i} \prod_{l=0}^{z_i-1} C_{iz_i-l-1}^{i-l-1}. \quad (7.17)$$

Если в (7.17) $z_i = 0$, то $\prod_{l=0}^{z_i-1} C_{iz_i-l-1}^{i-l-1} = 1$.

Для формул (7.15), (7.17) должны выполняться ограничения $\sum_{i=1}^k z_i \leq Y$, в противном случае $p_k = 0$.

Соотношение (7.17) можно преобразовать к виду

$$p_k = Y^{-X} \frac{X!}{z_1!} \prod_{s=0}^{c-1} (Y-s) \prod_{i=2}^k [(i!)^{z_i} z_i!]^{-1}.$$

При $k = 1$ $\prod_{i=2}^k [(i!)^{z_i} z_i!]^{-1} = 1$ и $p_1 = Y^{-X} \prod_{s=0}^{c-1} (Y-s)$.

В МПС с одной магистралью ($K = 1$) среднее время ожидания

$$E [T_{ож}] = u \frac{X-1}{2} \tau_{обм}. \quad (7.18)$$

Вид аналитических зависимостей $E [T_{ож}] = f(X)$ при различных N (см. рис. 7.6) является явно нелинейным. Так, в частности, по кривой $E [T_{ож}] = f(X)$ при $N = 8$ можно установить, что при каждом увеличении числа процессоров на единицу (в пределах от $X = 2$ до $X = 7$) значение величины $E [T_{ож}]$ увеличивается примерно в 2 раза, т. е. характер изменения функции $E [T_{ож}] = f(X)$ при различных N и заданной вероятности p обращения процессоров к секциям памяти можно аппроксимировать квадратичной функцией.

Вид аналитических зависимостей $E [T_{ож}] = f(X)$ при различных вероятностях обращения процессоров к секциям памяти и заданном N позволяет сделать заключение, что с увеличением p в 2 раза время ожидания $E [T_{ож}]$ увеличивается более чем в 2 раза. Поэтому для уменьшения $E [T_{ож}]$ следует так размещать программы в секциях ОЗУ, чтобы вероятность обращения процессоров к секциям памяти была по возможности минимальной.

Проведенный анализ показывает, что общее время ожидания на разрешение конфликтных ситуаций следует определять дифференцировано в зависимости от структуры МПС, а именно:

при $K = 1$ конфликты между процессорами возникают только из-за общей шины, поэтому при наличии у процессоров локальной памяти величина $E [T_{ож}]$ определяется по формуле (7.16), при отсутствии локальной памяти — по формуле (7.18);

при $K = \min \{X, Y\}$ конфликты возникают из-за общей памяти, поэтому в зависимости от наличия и отсутствия локальной памяти у

процессоров величина $E\{T_{ож}\}$ определяется согласно выражениям (7.12), (7.13):

при $1 < K < \min\{X, Y\}$ конфликты возникают как из-за общей памяти, так и из-за общих шин, поэтому в МПС с локальной памятью у процессоров нижняя оценка величины среднего времени ожидания определяется выражением (7.12), а верхняя — (7.16). При отсутствии локальной памяти у процессоров эти оценки определяются выражениями (7.13) и (7.18) соответственно.

Метод имитационного моделирования. При проектировании МПС контроля РЭС часто оказывается целесообразным использование методов имитационного моделирования для определения среднего времени ожидания на разрешение конфликтных ситуаций МПС.

Рассмотрим методику моделирования конфликтных ситуаций в МПС. Каждый из X процессоров системы в произвольный момент времени находится в одном из состояний $s_i \in S$, $S = \{s_1, s_2, \dots, s_Y, s_{Y+1}\}$. Состояния s_i , $i = 1, \dots, Y$ соответствуют тому, что процессор обратился к i -й секции памяти, а состояние s_{Y+1} — тому, что процессор не обратился к общей памяти. Закон, распределения вероятностей состояний приведен в табл. 7.1.

Таблица 7.1

Закон распределения вероятностей состояний

s_i	1	2	...	i	...	Y	$Y+1$
p_i	p/Y	p/Y	...	p/Y	...	p/Y	$1-p$

Работа моделирующего алгоритма состоит в следующем (рис. 7.8). Программным способом вырабатываются X псевдослучайных квазиравномерных чисел, которые используются для получения состояний (по состоянию у каждого процессора) с заданным законом распределения (см. табл. 7.1). Зная состояние каждого процессора, можно определить число состояний всех видов и число конфликтующих процессоров, которое регистрируется в счетчике.

Описанную процедуру повторяют до тех пор, пока текущее число опытов N не превысит заданного числа опытов N_0 . После этого вычисляют вероятности конфликтов p_k , $k = 2, \dots, X$ и среднее время ожидания $E\{T_{ож}\}$ на разрешение конфликтов по соотношениям $p_k = \frac{N_k}{N_0}$, $k = 2, \dots, X$, где N_k — число опытов с k конфликтующими процессорами:

$$E\{T_{ож}\} = \tau_{обм} \sum_{k=2}^X p_k (k-1).$$

В качестве примера рассмотрим МПС с хранением всех программ в ИЗУ процессоров, а всех констант, исходных и промежуточных дан-

ных — в секциях общей памяти. МПС характеризуется следующими параметрами: $p = 0,25$, $\tau_{обм} = 6$ мкс, $X = 2,7$, $Y = 1,4$.

Результаты моделирования и вычисления среднего времени ожидания, необходимого на разрешение конфликтных ситуаций при $N_0 = 10^4$, представлены на рис. 7.9. Результаты моделирования показывают, что при увеличении значений p и $\tau_{обм}$ среднее время ожидания $E[T_{ож}]$ будет возрастать.

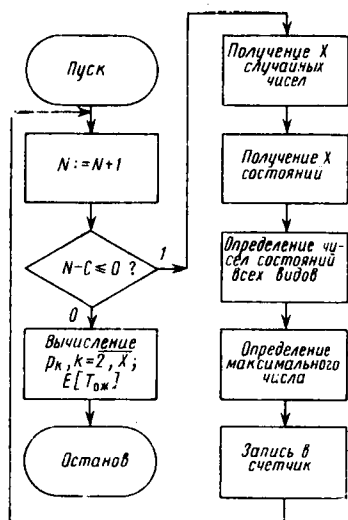


Рис. 7.8. Структурная схема моделирующего алгоритма

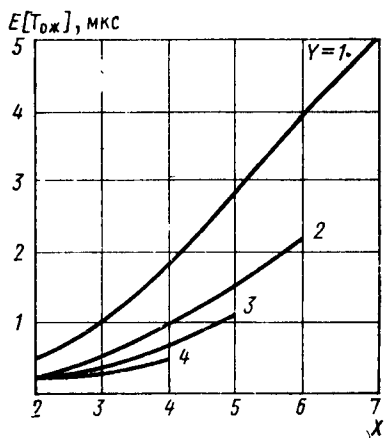


Рис. 7.9. Результаты моделирования конфликтных ситуаций в МПС

Описанная методика определения среднего времени ожидания, необходимого на разрешение конфликтных ситуаций, может быть использована как при анализе производительности спроектированной МПС, так и при синтезе магистрально-модульной структуры МПС.

§ 7.5. АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛИТЕЛЬНОГО ЯДРА МПС

Модель функционирования вычислительного ядра. Вычислительное ядро магистрально-модульной МПС состоит из ограниченного величины N числа устройств, в состав которых входит X процессоров и Y секций памяти, причем $X + Y \leq N$. Взаимодействие и обмен информацией между устройствами МПС осуществляются с помощью параллельно работающих информационных магистралей (см. рис. 7.4).

Существенное влияние на структуру вычислительного ядра МПС оказывают характеристики входных потоков заявок и их обслуживание. При организации контроля функционирования сложных РЭС не-

обходимо получать информацию из множества контролируемых точек аппаратуры. Эта информация подается с помощью системы датчиков и преобразователей формы информации в виде множества независимых потоков заявок C_i , как правило регулярных, либо простейших с интенсивностью их поступления λ_i . Общий поток обрабатываемых заявок C является пуассоновским с параметром $\lambda = \sum_{i=1}^m \lambda_i$. Длительность

реализации программы решения задач $c_i \in C$ является также случайной величиной (из-за случайного количества каждого из этапов обслуживания, случайного времени реализации циклов и асинхронной реализации сложных арифметических операций). Однако известно, что если в какой-то момент t_0 происходит обслуживание заявки, то закон распределения оставшегося времени обслуживания не зависит от того, сколько времени обслуживание уже продолжалось. Поэтому можно считать, что длительность реализации программы $c_i \in C$ распределена по экспоненциальному закону с м. о. μ_i^{-1} времени обслуживания задачи c_i программой L_i .

При проектировании МПС контроля РЭС принципиальным является вопрос выбора оптимальной (или рациональной) дисциплины обслуживания потоков входных заявок. В случае, если между поступающими заявками отсутствует отношение порядка (поступают независимые заявки), то возможно как приоритетное, так и беспriorитетное обслуживание.

Отметим, что в отсутствие внесистемных приоритетных дисциплин оптимальным с точки зрения минимизации средней длительности пребывания заявок в системе в классе приоритетных дисциплин без прерываний является выбор кратчайшей заявки (SPT), а среди дисциплин с прерываниями — выбор заявки с кратчайшей длительностью до обслуживания (SPRT).

Сравнение дисциплины SPT и SPRT показывает, что прерывание, которое не всегда легко осуществить, дает не слишком большое увеличение эффективности.

Однако стратегия, основанная на длительности обслуживания, оптимальна, если длительность обслуживания каждой заявки может быть точно предсказана в момент ее поступления, когда относительно будущих заявок известно только, что они образуют простейший поток с интенсивностью λ и задана плотность распределения длительности их обслуживания. Если известно еще что-нибудь, например могут быть предсказаны интервалы между поступлениями и длительности обслуживания нескольких будущих заявок, то эти стратегии, вообще говоря, уже не будут оптимальными, хотя часто остаются хорошими.

Кроме того, в МПС со сложной конфигурацией часто довольно трудоемкой оказывается техническая реализация процедуры назначения приоритетов, связанная с анализом множества поступивших в МПС заявок. При этом может оказаться, что выигрыш, получаемый для усредненных временных характеристик за счет введения приоритетных

дисциплин, значительно ухудшает эти характеристики вследствие затрат на назначение приоритетов. При этом следует иметь в виду, что характеристики МПС можно улучшить в случаях, когда моменты поступления и длительности обслуживания заявок становятся известными еще до их поступления в систему. Однако проблема использования такой прогнозирующей информации пока не получила удовлетворительного решения.

В связи с этим при поиске моделей проектирования магистрально-модульных МПС с общей памятью целесообразно исследование простейших дисциплин обслуживания типа «первым пришел — первым обслужен» (FIFO) и «последним пришел — первым обслужен» (LIFO), которые довольно легко реализуются с технической точки зрения.

Для определения производительности вычислительного ядра синтезируемой МПС необходимо сконструировать временной функционал, характеризующий м. о. времени $E\{T\}$ реакции системы на поступающую заявку:

$$E\{T\} = E\{T_0\} + E\{T_{обр}\}, \quad (7.19)$$

где $E\{T_0\}$ — м.о. времени пребывания заявки в буферной памяти; $E\{T_{обр}\}$ — м.о. времени обработки заявки $c_i \in C$ в системе.

Проанализируем функциональные зависимости компонентов функционала (7.19) от переменных X, Y, K, Z , где Z — емкость буферной памяти.

Определение м.о. времени пребывания заявки в буферной памяти. Буферная память МПС необходима для временного хранения поступающих заявок c_i , если в момент их поступления все процессоры системы оказываются занятыми. Если в МПС буферная память отсутствует, то $E\{T_0\} = 0$ и в системе возможна потеря поступающих заявок, что для реальных систем контроля РЭС недопустимо. Методы расчета оптимальных (с точки зрения конкретных критериев) емкостей буферной памяти находятся в прямой зависимости от назначения всей системы обработки информации и структуры устройства сопряжения МПС с объектом контроля.

Для определения величины $E\{T_0\}$ воспользуемся моделью одноканальной системы массового обслуживания (СМО) с ограниченной очередностью, пуассоновским входом и экспоненциальным распределением времени обслуживания (рис. 7.10). Напомним, что в случае, когда пуассоновский поток стационарен (простейший поток), интервал времени T между событиями в этом потоке есть случайная величина, распределенная по показательному закону

$$f(t) = \lambda e^{-\lambda t}, \quad t > 0,$$

где λ — интенсивность потока событий.

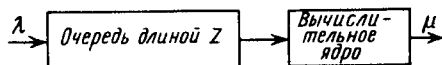


Рис. 7.10. Модель МПС в виде одноканальной СМО с ограниченной очередью

Обслуживание заявки продолжается в течение случайного времени, распределенного по показательному закону с параметром μ :

$$\varphi(t) = \mu e^{-\mu t}, \quad t > 0.$$

Интенсивность обслуживания μ характеризует среднее число μ обслуженных заявок в единицу времени в стационарном режиме работы вычислительного ядра, т. е. в режиме непрерывно занятых и постоянно нагруженных процессоров МПС. При таком допущении модель одно-канальной СМО с ограниченной очередью является корректной для анализа параметров функционирования вычислительного ядра, а величина μ определяется равенством $\mu = E[T_{обп}]^{-1}$. Относительная за-

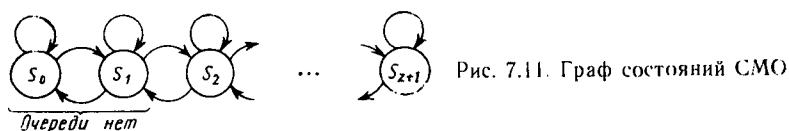


Рис. 7.11. Граф состояний СМО

грузка системы $\rho = \lambda E[T_{обп}]$. Заявка, поступившая в момент, когда канал (вычислительное ядро) занят, становится в очередь и ожидает обслуживания.

Состояния рассматриваемой СМО:

- S_0 — канал свободен;
- S_1 — канал занят, очереди нет;
- S_2 — канал занят, одна заявка стоит в очереди;
- ...
- S_k — канал занят, $k - 1$ заявка стоит в очереди;
- ...
- S_{Z+1} — канал занят, Z заявок стоит в очереди.

Граф состояний описываемой СМО показан на рис. 7.11. Пользуясь общим решением схемы гибели и размножения для графа рис. 7.11, напишем выражения предельных вероятностей состояний системы:

$$\begin{aligned} p_1 &= \rho p_0, \quad p_2 = \rho^2 p_0, \quad \dots, \quad p_k = \rho^k p_0, \\ p_{Z+1} &= \rho^{Z+1} p_0, \\ p_0 &= [1 + \rho + \rho^2 + \dots + \rho^{Z+1}]^{-1} = \frac{1 - \rho}{1 - \rho^{Z+2}}, \end{aligned} \quad (7.20)$$

где $\rho = \lambda/\mu$, $\rho < 1$.

Выражение для м. о. времени пребывания заявки в буферной памяти $E\{T_б\}$ определяются исходя из следующих соображений. Пусть заявка приходит в систему в какой-то момент времени. С вероятностью p_0 канал не будет занят, т. е. время ожидания будет равно нулю. С вероятностью p_1 она придет в систему во время обслуживания первой пришедшей заявки, но перед ней не будет очереди и заявка будет ждать начала своего обслуживания в течение времени $1/\mu$ и т. д. Во-

обще, с вероятностью p_k пришедшая заявка застанет в системе k заявок и будет ждать в среднем k/μ единиц времени ($k = \overline{1, Z}$). При $k = Z + 1$ заявка теряется, так как в очереди свободных мест нет (вероятность этого события p_{Z+1}). Поэтому можно написать

$$E[T_0] = \sum_{i=1}^Z p_i \frac{i}{\mu} = \frac{p_0 \rho}{\mu} \sum_{i=1}^Z i \rho^{i-1}.$$

С учетом формулы суммы геометрической прогрессии выражение для $E[T_0]$ принимает вид

$$E[T_0] = \frac{\rho [1 - \rho^Z (Z + 1 - Z\rho)]}{\mu (1 - \rho^{Z+2}) (1 - \rho)}. \quad (7.21)$$

В формуле (7.21) неизвестными являются Z , μ и $\rho = \lambda/\mu$.

В МПС с большой емкостью буферной памяти можно считать, что $Z \rightarrow \infty$. В этом случае выражение (7.21) принимает вид

$$E[T_0] = \frac{\rho^2}{\lambda (1 - \rho)}. \quad (7.22)$$

Определение м. о. времени обработки заявки. При наличии свободного процессора в вычислительном ядре МПС заявка, стоящая в буферной памяти первой, поступает на свободный процессор для обслуживания. В процессе обслуживания над сигналом, соответствующим поступившей заявке c_i , производится последовательность арифметических и логических преобразований в соответствии с программой обработки заявки. Для обеспечения процесса обработки информации в вычислительном ядре производится обмен информацией между процессорами и секциями памяти. В процессе обмена информацией в ядре неизбежны конфликтные ситуации за счет одновременного обращения нескольких процессоров к одной и той же секции памяти или к одной и той же информационной магистрали. Поэтому м.о. времени обработки заявки $E[T_{обp}]$ определяется в виде суммы составляющих:

$$E[T_{обp}] = E[T_{выч}] + E[T_{обм}] + E[T_{ож}], \quad (7.23)$$

где $E[T_{выч}]$ — м.о. времени, необходимого на обработку информации (вычисления) в процессорах; $E[T_{обм}]$ — м.о. времени, затрачиваемого на обмен требующейся для решения задачи информацией между процессорами и ЗУ; $E[T_{ож}]$ — м.о. времени, необходимого на разрешение конфликтных ситуаций, возникающих в процессе работы МПС при одновременном в пределах цикла памяти обращении различных процессоров к одним и тем же информационным магистралям или секциям ОЗУ.

Задачи оперативного контроля различных параметров РЭС, как правило, реализуются с помощью коротких алгоритмов. Поэтому каждая из поступивших заявок обрабатывается на одном процессоре без распараллеливания вычислительного процесса и без передачи заявки

на доработку другому процессору. Для проведения вычислений в соответствии с алгоритмом L_i решения задачи c_i требуется время

$$E[T_{\text{выч}i}] = \tau_0 + \sum_{j=1}^{s_i} n_{ij} \alpha_j \tau, \quad (7.24)$$

где τ_0 — время входа процессора в программу обработки и выхода из нее; s_i — число типов операций в программе L_i решения задачи c_i , $i = \overline{1, m}$; n_{ij} — среднее число операций j -го типа, необходимых для обработки заявки c_i ; величину n_{ij} для каждой задачи определяют исходя из количества ветвей v в алгоритме решения и вероятностей p их выполнения; α_j — коэффициент сложности выполнения операции j -го типа по отношению к элементарной ($\alpha_j \geq 1$); τ — время выполнения элементарной операции, например передачи «регистр — регистр».

В стационарном режиме работы МПС все X процессоров вычислительного ядра обрабатывают X находящихся в ядре заявок. Поэтому величина $E[T_{\text{выч}}]$ может быть определена как

$$\begin{aligned} E[T_{\text{выч}}] &= \sum_{i=1}^m q_i E[T_{\text{выч}i}] X^{-1} = \\ &= \sum_{i=1}^m g_i \left(\tau_0 + \sum_{j=1}^{s_i} n_{ij} \alpha_j \tau \right) X^{-1}, \end{aligned} \quad (7.25)$$

где $g_i = \lambda_i \lambda^{-1}$ — относительная частота появления в системе задачи c_i .

В процессе обмена информацией между процессорами и секциями оперативной памяти процессор запрашивает из памяти очередную j -ю команду из программы обработки заявки c_i , считывает r_j операндов, требуемых для выполнения j -й операции, и в случае необходимости отсылает на хранение в ЗУ f_i слов.

Величина $E[T_{\text{обм}}]$ зависит от способа хранения программ решения задач, режимов работы памяти и выбранного количества K параллельно работающих информационных магистралей.

В случае, если общая память МПС состоит только из секций оперативной памяти, то м.о. времени обмена информацией $E[T_{\text{обм}}]$ между системой процессоров и секциями оперативной памяти определяется соотношением

$$E[T_{\text{обм}}] = \sum_{i=1}^m \sum_{j=1}^{s_i} g_i n_{ij} (1 + r_j + a f_i) \tau_{\text{обм}i} K^{-1}. \quad (7.26)$$

Здесь r_i — число операндов, необходимых для выполнения j -й операции, $r_j = 0, 1, 2, \dots$; a — коэффициент, характеризующий соотношение длительности обменных операций при считывании и записи; f_i — число операций записи в ОЗУ при решении задачи c_i ; $\tau_{\text{обм}i}$ — время обмена процессора с секцией ОЗУ одним словом (с учетом времени

пересылки через канал связи); K — число информационных магистралей, причем $K_{\max} = \min \{X, N - X\}$.

Время обмена $\tau_{\text{обм}_1}$ процессора с секций ОЗУ, одним словом зависит от режима работы памяти. Для систем с активной памятью

$$\tau_{\text{обм}_1}^{(a)} = 2E[T_{\text{ож}}^{(a)}] + 2T_{\pi} + T_0, \quad (7.27)$$

где $E[T_{\text{ож}}^{(a)}]$ — м. о. времени ожидания заявок от X процессоров и $Y = N - X$ секций памяти на доступ к информационным магистралям.

Величина $E[T_{\text{ож}}^{(a)}]$ определяется следующим образом. Поскольку источниками заявок в борьбе за K информационных магистралей являются N устройств (X процессоров и $Y = N - X$ секций памяти), формальной моделью анализа описываемой ситуации является K -канальная замкнутая система массового обслуживания с N источниками заявок. При условии простейших потоков заявок с интенсивностями обращения устройств к системе связей λ_1 среднее время ожидания

$$E[T_{\text{ож}}^{(a)}] = \frac{\rho_a^k N!}{K! K \mu_a} \left\{ \sum_{i=0}^{N-K} \frac{(i+1) \rho_a^i}{(N-K-i)! K^i} \right\} p_0, \quad (7.28)$$

где

$$p_0 = \left[1 + \sum_{i=1}^K \frac{\rho_a^i}{i!} \prod_{i=1}^K (N-i+1) + \frac{N!}{K!} \sum_{i=K+1}^N \frac{\rho_a^i}{(N-i)! K^{i-K}} \right]^{-1};$$

$$\rho_a = \lambda_1 / \mu_a; \quad \mu_a = (T_{\pi})^{-1}. \quad (7.29)$$

Для систем с пассивной памятью время обмена одним словом определяется выражением

$$\tau_{\text{обм}_1}^{(n)} = E[T_{\text{ож}}^{(n)}] + 2T_{\pi} + T_0, \quad (7.30)$$

где $E[T_{\text{ож}}^{(n)}]$ — м. о. времени заявок от X процессоров на доступ к K информационным магистралям.

Величину $E[T_{\text{ож}}^{(n)}]$ находят из анализа K -канальных замкнутых систем массового обслуживания с X источниками заявок:

$$E[T_{\text{ож}}^{(n)}] = \frac{\rho_n^K X!}{K! K \mu_n} \left\{ \sum_{i=0}^{X-K} \frac{(i+1) \rho_n^i}{(X-K-i)! K^i} \right\} p_0. \quad (7.31)$$

Здесь

$$p_0 = \left[1 + \sum_{i=1}^K \frac{\rho_n^i}{i!} \prod_{i=1}^K (X-i+1) + \frac{X!}{K!} \sum_{i=K+1}^X \frac{\rho_n^i}{(X-i)! K^{i-K}} \right]^{-1};$$

$$\rho_n = \frac{\lambda_1}{\mu_n}; \quad \mu_n = (2T_{\pi} + T_0)^{-1}. \quad (7.32)$$

Выражения (7.28), (7.29), (7.31), (7.32) позволяют оценить величину средних потерь времени на ожидание процессоров и секций памяти обслуживания их заявок каналами связи при решении МПС комплекса задач контроля РЭС. Следует отметить, что эти выражения не учитывают конфликтных ситуаций, возникающих из-за одновременного обращения к одной секции памяти нескольких процессоров.

Для систем, общая память которых состоит из секций ОЗУ и ПЗУ,

$$E[T_{обм}] = \sum_{i=1}^m \sum_{j=1}^{s_i} g_i |n_{ij}| (1 + r_{j_1} + a f_i) \tau_{обм1} + n_{ij_2} (1 + r_{j_2}) \tau_{обм2} K^{-1}, \quad (7.33)$$

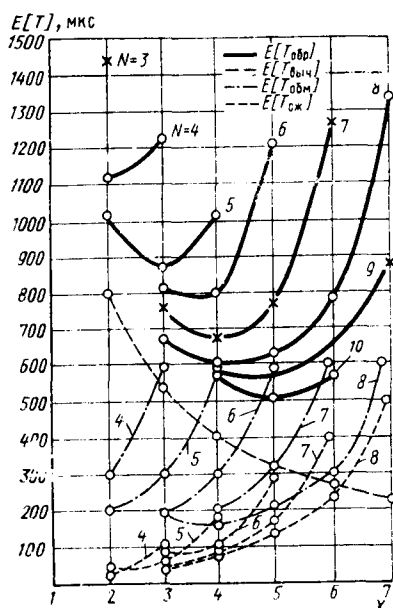


Рис. 7.12. Временные характеристики МПС при различных N

где n_{ij} , r_{j_1} — соответственно количество команд и операндов, хранимых в секциях ОЗУ; n_{ij_2} — n_{ij} , $r_{j_2} = r_j - r_{j_1}$ — количество команд и операндов, хранимых в секциях ПЗУ; $\tau_{обм}$ — время обмена процессора с секцией ПЗУ одним словом.

Для одновременного обмена между собой $\min\{X, N - X\}$ устройств нужно обеспечить достаточную пропускную способность системы связей. Требуемое количество параллельно работающих информационных магистралей определяется из условия $K \geq 2\tau_{обм}F$, где F — пропускная способность одной информационной магистрали.

Очевидно, что независимо от способа организации хранения информации для существования стационарного режима работы системы обязательно выполнение условия

$$E[T_{обм}] < \lambda^{-1}, \quad \rho < 1. \quad (7.34)$$

Величина $E[T_{ож}]$, характеризующая м.о. времени, необходимого на разрешение конфликтных ситуаций, определяется с помощью соотношений, полученных в § 7.4.

Исследуем характер изменения функционала (7.19) при изменении переменных X и Z .

Составляющий член $E[T_{выч}]$ функционала (7.23) является монотонно убывающей функцией от X (при $Z = \text{const}$), $E[T_{ож}]$ — монотонно возрастающей функцией, $E[T_{обм}]$ — симметричной выпуклой функцией, а $E[T_б]$ — монотонно убывающей функцией от X (рис. 7.12). Следовательно, функционал (7.19) при $Z = \text{const}$ превращается в уни-

модальную по X выпуклую функцию, определенную на целочисленных значениях аргументов X и Z . При различных $Z = 2^q, q = 0, q_{\max}$ функционал (7.19) превращается в множество $\{E(T)\}$ унимодальных выпуклых функций, причем с увеличением Z значение функции $E(T)$ возрастает.

§ 7.6. СИНТЕЗ СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНОГО ЯДРА МПС

Задача синтеза структуры вычислительного ядра магистрально-модульной МПС заключается в определении оптимального соотношения между числом процессоров X , числом секций памяти Y , количеством информационных магистралей K и объемом буферной памяти Z по выбранному или заданному критерию оптимальности с учетом системы ограничений на показатели аппаратуры (в том числе на общее число устройств N).

Синтез любой технической системы, в том числе и МПС контроля РЭС, неразрывно связан с общесистемной оценкой эффективности. Наиболее полную оценку эффективности проектируемой аппаратуры дают обобщенные (интегральные) критерии, отражающие в виде одной функции степень соответствия различных вариантов аппаратуры своему назначению и связанные со всеми ее характеристиками. Однако использование обобщенных критериев при проектировании аппаратуры контроля РЭС встречается с серьезными трудностями ранжирования частных критериев. Поэтому задачу синтеза аппаратуры контроля следует решать на основе системы частных критериев, наиболее важными из которых являются показатели достоверности контроля, производительности обработки контрольно-измерительной информации, стоимости, энергоемкости, надежности аппаратуры контроля и т.д.

Рассмотрим методику формирования частных критериев при синтезе вычислительного ядра магистрально-модульных МПС. При постановке и решении задачи оптимального синтеза один из частных критериев должен быть выбран в качестве целевой функции, а остальные участвовать в качестве ограничений в задаче оптимального проектирования.

Производительность МПС. Производительность МПС оценивается функционалом (7.19), составляющие которого были детально проанализированы в § 7.5.

Емкость буферной памяти. Необходимое число ячеек Z буферной памяти можно определить по задаваемой допустимой вероятности ϵ потери заявки на входе МПС, которая для рассматриваемой в § 7.5 одноканальной СМО с ограниченной очередью определяется из условия

$$\frac{\rho^{Z+1} (1-\rho)}{1-\rho^{Z+2}} \leq \epsilon. \quad (7.35)$$

Требования организации системы адресации буферной памяти и ограничения на общее число устройств N могут быть записаны в виде

$$X + Y \leq N, \quad Z = 2^q, \quad (7.36)$$

где q — целое число (в современных МПС $0 < q \leq 10$).

Функция (7.35), характеризующая вероятность потери заявки на входе МПС от м.о. времени обработки $E[T_{обp}]$ и объема буферной памяти Z , уменьшается как при увеличении X ($Z = \text{const}$), так и при увеличении Z ($X = \text{const}$).

Аддитивные параметры МПС. Массогабаритные, стоимостные, энергетические и ряд других параметров МПС являются аддитивными, а функциональные зависимости этих параметров от переменных X , K , Z — линейными.

Постановка задачи оптимального синтеза. Задача проектирования оптимальной структуры вычислительного ядра магистрально-модульной МПС с конечной буферной памятью заключается в нахождении системы целых чисел X , $Y = N - X$, K и Z , оптимизирующих выбранную целевую функцию при наличии системы ограничений на ряд параметров МПС.

В случае, если в качестве целевой функции выбрано время пребывания заявки в системе, описываемое выражением (7.19), и присутствуют ограничения (7.34) — (7.36) и ограничения на аддитивные параметры МПС, то задача оптимального проектирования формулируется в виде задачи целочисленного программирования с нелинейной целевой функцией и нелинейными ограничениями, которую можно решить либо методом ветвей и границ, либо методом дихотомии. Однако более эффективно использование специально разработанного для решения рассматриваемой задачи алгоритма адаптивного поиска.

Исследуем область существования задачи оптимизации.

Поскольку функции (7.34) и (7.35) являются выпуклыми как по X (при $Z = \text{const}$), так и по Z (при $X = \text{const}$), то область существования рассматриваемой задачи определена на выпуклой целочисленной оболочке, которая образуется из целочисленной решетки $X \times Q$ (где $X = \{1, \dots, N-1\}$, $Q = \{0, 1, \dots, q_{\max}\}$, $q = \log Z$, $Z = 1, 2^{q_{\max}}$) посредством наложения ограничений (7.34) и (7.35), а также ограничений на аддитивные параметры МПС.

Работа алгоритма 7.1 адаптивного поиска решения задачи математического программирования (7.19), (7.34) ... (7.36) заключается в следующем.

Будем обозначать решение на r -м шаге в виде упорядоченной пары (точки на плоскости) $O = (X_r, q_r)$. На первых шагах алгоритма (шаги 1 ... 10) ищется первоначальное допустимое решение $O_0 = (X_0, q_0)$, удовлетворяющее условиям (7.34) ... (7.36). В силу характера изменения функций (7.19) и (7.35) с целью ускорения сходимости алгоритма 7.1 поиск исходного допустимого решения целесообразно начинать либо с точки ($X_0 = N/2$, $q = q_{\max}$) при N — четном, либо с точки ($X_0 = (N-1)/2$, $q = q_{\max}$) при N — нечетном.

После определения точки O_0 и соответствующего ей значения целевой функции $E[T_0]$ проводится последовательное улучшение решения задачи в направлении глобального оптимума (шаги 11 ... 28 алгоритма). Опишем последовательность выполнения этого алгоритма.

А л г о р и т м 7.1.

1. Положить $r = 0$, где r — номер итерации.
2. Выбрать исходное решение O_0 ($X_0 = N/2$, $q_0 = q_{\max}$) или $O_0 := (X_0 = (N - 1)/2, q_0 = q_{\max})$.
3. Вычислить $E[T_r]$ согласно формуле (7.19).
4. Проверить выполнение условий (7.34) и (7.35). Если оба условия выполняются, перейти к шагу 11, в противном случае — к шагу 5.
5. Положить $r := r + 1$.
6. Выбрать новое решение $O_r = (X_r = X_{r-1} + 1; q_r = q_{r-1})$.
7. Исследовать X_r . Если $X_r < N$, перейти к шагу 3, в противном случае — к шагу 8.
8. Выбрать новое решение $O_r = (X = X_{r-1}; q_r = q_{r-1} - 1)$.
9. Проверить условие $q_r \geq q_{\min}$ (в частности, $q_r \geq 0$). Если это условие выполняется, перейти к шагу 3, в противном случае — к шагу 10.
10. Конец алгоритма. Решения задачи при заданных ограничениях не существует. Для получения решения следует изменить исходные данные с целью ослабления ограничений (7.34) и (7.35).
11. Включить точку O_r в множество допустимых решений O , а соответствующее ей значение целевой функции $E[T_r]$ — в множество T .
12. Определить, не является ли пустым множество $O_r = \{O_r^{(i)}\}$ точек, соседних с точкой O_r . Очевидно, одна или обе координаты точки $O_r^{(i)} = (X_r^{(i)}, q_r^{(i)})$ должны отличаться от координат точки $O_r = (X_r, q_r)$ на единицу при условии выполнения ограничений (7.36).
- Если множество $O_r \neq \emptyset$, перейти к шагу 13, в противном случае — к шагу 27.
13. Упорядочить элементы $O_r^{(i)} \in O_r$, $i = \overline{1, i_r}$.
14. Положить $i = 1$.
15. Вычислить $E[T_r^{(i)}]$ согласно выражению (7.19).
16. Проверить выполнение условия (7.35). Если оно выполняется, перейти к шагу 17, в противном случае — к шагу 18.
17. Проверить выполнение условия (7.34). В случае его выполнения перейти к шагу 21, в противном случае — к шагу 18.
18. Из множества O исключить $O_r^{(i)}$.
19. Положить $i := i + 1$.
20. Проверить номер проверяемого элемента множества O_r . Если $i \leq l_r$, то перейти к шагу 15, в противном случае — к шагу 22.
21. Включить точку $O_r^{(i)}$ в множество O , а соответствующее ей значение $E[T_r^{(i)}]$ — в множество T . Перейти к шагу 19.
22. Определить точку $O_m^{(k)}$ или множество точек $O_m^{(k)} = \{O_m^{(k)}\}$, $1 \leq k \leq l_m$, $0 \leq m \leq r$, $i = \overline{1, l_n}$, $n = \overline{0, r}$, для которых $E[T_m^{(k)}] = \inf_{in} T$. В силу направленности работы алгоритма возможно, что либо $m = r$, либо $m = r - 1$.
23. Если для всех точек $O_m^{(k)} \in O_m^{(k)}$, $m = r - 1$, то включить точку $O_m^{(k)}$ в множество O_{r-1} и перейти к шагу 24. Если же имеются точки $O_m^{(k)} \in O_m^{(k)}$, для которых $m = r$, то включить их в множество $O_r^{(k)}$ и перейти к шагу 25.
24. Все точки множества $O_{r-1}^{(k)}$ представляют собой решение задачи оптимизации. Поэтому из множества $O_{r-1}^{(k)}$ выбирают любую точку $O_{r-1}^{(i)} = (X_{r-1}^{(i)}, q_{r-1}^{(i)}) \in O_{r-1}^{(k)}$, которая является решением задачи. При этом $X_{\text{opt}} = X_{r-1}^{(i)}$, $Z_{\text{opt}} = 2^{q_{r-1}^{(i)} - 1}$. Перейти к шагу 28.

25. Проверить наличие точек, соседних с точками $O_r^{(k)} \in O_r^{(k)}$ и не входящих в множество O .

Если такие точки существуют, то перейти к шагу 26, в противном случае — к шагу 27.

26. Положить $r := r + 1$, переобозначить $O_r^{(k)}$ как O_r . Перейти к шагу 13.

27. Все точки множества $O_r^{(k)}$ представляют собой решение задачи оптимизации. Поэтому из множества $O_r^{(k)}$ выбирают любую точку $O_r^{(i)} = (X_r^{(i)}, q_r^{(i)}) \in O_r^{(k)}$,

которая является решением задачи. При этом $X_{\text{opt}} = X_r^{(i)}$, $Z_{\text{opt}} = 2^{q_r^{(i)}}$.

28. Конец алгоритма.

Сходимость алгоритма следует из того факта, что целевая функция (7.19) является либо унимодальной выпуклой, либо монотонно убывающей функцией, определенной в выпуклой целочисленной области допустимых решений, а работа алгоритма основана на направленной процедуре поиска глобального оптимума задачи. Поэтому за конечное число итераций возможно нахождение точки, которая соответствует решению рассматриваемой задачи. Отметим, что при выполнении процедуры полного перебора число итераций равно $(N - 1) q_{\text{max}}$, а верхняя оценка числа итераций выполнения алгоритма 7.1 определяется суммой $N/2 + q_{\text{max}}$.

Поскольку в структуре любой МПС допускается использование произвольного числа устройств $N \leq N_{\text{max}}$, для поиска оптимального соотношения между числом процессоров и секций общей памяти алгоритм 7.1 нужно реализовать $N_{\text{max}} - 2$ раз со всеми возможными значениями ($2 \leq N \leq N_{\text{max}}$), а затем выбрать наилучшее решение с учетом заданных массо-габаритных и стоимостных ограничений.

Описанная методика проектирования правомочна при проектировании оптимальных структур МПС и по другим критериям оптимальности, в частности по критерию минимума массо-габаритных (стоимостных) параметров. При этом в алгоритме 7.1 необходимо ввести новую целевую функцию (шаги 3, 15, 21) и заданные ограничения (шаги 4, 16, 17), предварительно исследовав характер поведения целевой функции задачи.

Методику проектирования вычислительного ядра МПС аппаратуры контроля РЭС проиллюстрируем на следующем примере.

Пример 7.1. Исходной информацией для проектирования вычислительного ядра МПС являются: множество $S = \{c_1, \dots, c_m\}$ сигналов—заявок на обработку контролируемых параметров; интенсивности λ_i поступления каждой заявки c_i ; программы L_i обработки заявки c_i . Из анализа программ определяют величины n_{ij} — количество команд j -го типа.

Время τ выполнения элементарной операции (типа «регистр — регистр») и коэффициенты сложности α_j выполнения каждой операции определяют при выборе МПК БИС.

На рис. 7.12 приведены функциональные зависимости изменения м.о. времени обработки контролируемой информации, времени вычислений, обмена и ожидания от числа X комплексируемых ММ. Класс решаемых задач — задачи предварительной обработки информации при контроле сложных РЭС. Структура МПС — магистральная, число комплексируемых устройств N может изменяться от 3 до 10, система элементов МПК К588. Оптимальное значение X комплексируемых ММ для каждого N определяется точкой, соответствующей кривой $F(X) = E [T_{\text{обп}}](X)$, в которой $dF(X)/dX = 0$.

В зависимости от заданных ограничений на массогабаритные, стоимостные и энергетические параметры выбирают максимально допустимое число N комплексируемых устройств, а затем по функциональной зависимости $F(X) = E[T_{\text{обp}}](X)$ при заданном N определяют структуру МПС максимальной производительности при заданной системе ограничений на параметры МПС.

§ 7.7. МЕТОДИКА ПРОЕКТИРОВАНИЯ МПС ПОВЫШЕННОЙ ЖИЗНЕСПОСОБНОСТИ

Мультипроцессорная системная организация является наилучшим средством достижения высокой готовности любой вычислительной системы, функционирующей в условиях большой рабочей нагрузки.

Основным требованием, которому должна удовлетворять в этих случаях МПС, является функциональная гибкость, т. е. способность системы оперативно перестраивать свою конфигурацию. Перестройка системы может предполагать либо разделение ее на ряд независимых подсистем, либо включение в ядро системы тех или иных устройств за счет исключения других, либо изменение специализации ММ (настройка на эффективное решение каких-либо отдельных задач) путем изменения содержимого ППЗУ ММ и др. Иными словами, управление структурой системы дает возможность устанавливать наиболее эффективную рабочую конфигурацию системы. Очевидно, что система связей в таких МПС должна быть программно-управляемой.

Высокие показатели надежности вычислительных систем, работающих в реальном времени, в настоящее время могут быть обеспечены лишь структурными методами. В МПС, построенных по модульному принципу, необходимая надежность может быть достигнута автоматической реконфигурацией системы.

Техническая реализация систем, устойчивых к отказам, предполагает применение нескольких устройств каждого типа, скомплексированных таким образом, что в случае выхода из строя отдельного устройства исходная конфигурация системы продолжает работать. Однако при этом несколько снижается ее эффективность вследствие реконфигурации и перераспределения нагрузки между оставшимися исправными устройствами. Такое свойство систем противостоять воздействию отказов оборудования характеризует живучесть вычислительной системы при выполнении ею своего целевого назначения. При этом под «живучестью» (жизнеспособностью) понимают свойства программной организации структур, в которых отказы и восстановления любых ММ не нарушают процесс решения сложных задач, а могут изменять время либо качество их решения.

Автоматическое восстановление работоспособности МПС предполагает выполнение следующих основных действий: обнаружение отказа, останов системы, локализацию отказа, изменение конфигурации, повторный пуск.

Обнаружение и локализация отказа предполагает наличие развитого аппаратного и тестового контроля, позволяющего определить вышедший из строя функциональный блок с точностью до автоматически заменяемой конструктивной единицы. Изменение конфигурации дает возможность отключить отказавшее устройство, перераспределить вычислительную нагрузку между оставшимися устройствами и при недостатке вычислительной мощности восстановленной системы либо исключить решение некоторых задач, либо перейти на использование менее точных, но более коротких программ обработки данных.

При анализе и синтезе жизнеспособных МПС используются два типа моделей: функциональная и топологическая.

Функциональная модель описывает функционирование МПС в различных конфигурациях, порождаемых выходом из строя некоторых из ее компонентов. Предполагается, что МПС содержит f типов устройств (модулей), причем однотипные модули полностью взаимозаменяемы. Отказавший модуль может быть заменен аналогичным исправным модулем, находящимся в горячем резерве. Каждый модуль в этом случае имеет свою схему коммутации, обеспечивающую подключение его к информационным магистралям и являющуюся составной частью аппаратуры модуля. Минимальное ядро МПС — минимальный набор модулей (в пределе по одному модулю каждого типа), обеспечивающих функционирование МПС.

Под коэффициентом жизнеспособности МПС будем понимать величину

$$Q = W/W_0 = \sum_{j=1}^M p_j W_j / W_0, \quad (7.37)$$

где W — м.о. производительности МПС с учетом отказов устройств; W_0 — производительность МПС без отказов; p_j — вероятность j -го состояния; W_j — производительность системы в j -м состоянии; M — число работоспособных состояний системы.

Оценка жизнеспособности по формуле (7.37) аналогична оценке технических средств с помощью коэффициента готовности, характеризующего вероятность того, что в произвольный момент времени устройство является исправным. Показатель Q характеризует вычислительную способность МПС для произвольного момента времени.

Топологической моделью для описания структурных свойств МПС является функциональный граф системы. При исследовании топологической модели жизнеспособных систем обычно используют топологические критерии жизнеспособности, являющиеся теми или иными характеристиками графов. Следует отметить, что в структуре жизнеспособной МПС не должно быть ни одного модуля или межмодульной связи, отказ которых ведет к выходу из строя всей МПС. Отсюда вытекает, что минимальная избыточность аппаратуры — наличие двух устройств каждого типа и двух маршрутов передачи информации между устройствами, а сама МПС должна быть построена по агрегатно-модульному принципу.

Рассмотрим методику синтеза МПС повышенной жизнеспособности.

Как и в § 7.6, будем рассматривать однородную МПС, состоящую из множества $D = \{d_j\}$ $j = \overline{1, X}$ однотипных ММ, множества $B = \{b_k\}$, $k = \overline{1, Y}$ секций памяти и множества $M = \{h_l\}$, $l = \overline{1, K}$ внутрисистемных каналов связи.

От активного источника информации в МПС поступает множество потоков $C = \{C_i\}$, $i = \overline{1, m}$ сигналов — заявок на обработку контролируемых параметров одного уровня приоритета, причем каждый поток $C_i \in C$ может быть либо регулярным, либо пуассоновским с интенсивностью поступления λ_i (в частности, такими свойствами обладают потоки телеметрической информации в автоматизированных системах контроля сложных РЭС).

Каждая задача $c_i \in C_i$ (входная заявка) может быть решена с помощью одного из алгоритмов $L_i = \{L_i^{(l)}\}$, $l = \overline{1, w}$. В свою очередь, каждый из алгоритмов L_i может быть реализован s_i методами, различающимися точностью и сложностью (множество методов реализации алгоритма L_i обозначим $U_i = \{u_i^{(q)}\}$, $q = \overline{1, s_i}$). Тогда каждой заявке $c_i \in C_i$ можно поставить соответствие множество упорядоченных пар $R_i = \{(L_i^{(l)}, u_i^{(q)})\}$, любая из которых указывает алгоритм решения задачи и идентифицирует программу ее обработки.

Программа обработки заявки c_i назначается перед использованием системы и динамически изменяется в случае необходимости реконфигурации системы, если ее «новая» производительность не обеспечивает выполнение «старых» задач.

Задача синтеза структуры МПС повышенной жизнеспособности заключается в определении такого набора ММ, секций ЗУ и информационных магистралей, чтобы при заданных расходах на аппаратные средства можно было максимизировать коэффициент жизнеспособности системы.

Очевидно, при заданных расходах на аппаратные средства жизнеспособность системы оказывается максимальной при минимизации числа типов различных устройств. К тому же уменьшение числа типов используемых устройств позволяет упростить вычисление Q за счет сокращения возможных состояний и алгоритмы управления реконфигурацией. Так, для СМВС, содержащей f типов устройств, общее число состояний $M = \prod_{i=1}^f (s_i + 1)$, что существенно меньше величины

$M = 2^{\sum_{i=1}^f s_i}$, где s_i — максимальное количество устройств типа i .

Будем полагать, что потоки отказов и восстановлений устройств являются независимыми, пуассоновскими с постоянными параметрами λ_i , μ_i . Тогда средние времена пребывания i -го устройства в работоспособном и неработоспособном состояниях соответственно равны $T'_i =$

$= 1/\lambda_i$, $T_i'' = 1/\mu_i$, $i = \overline{1, f}$, а вероятности p_i и $q_i = 1 - p_i$ пребывания устройства в исправном и неисправном состояниях $p_i = T_i' / (T_i' + T_i'')$, $q_i = T_i'' / (T_i' + T_i'')$, $i = \overline{1, f}$.

Каждое состояние s_j МПС характеризуется тем, что часть устройств находится в рабочем, а часть — в нерабочем состоянии. Вероятность того, что t_i из s_i устройств i -го типа находится в рабочем состоянии, легко определить по формуле биномиального распределения $p(t_i, s_i, p_i) = \binom{s_i}{t_i} p_i^{t_i} q_i^{s_i - t_i}$.

Тогда вероятность состояния s_j , в котором работают t_i устройств каждого i -го типа,

$$p(s_j) = p_j = \prod_{i=1}^f p(t_i, s_i, p_i) = \prod_{i=1}^f \binom{s_i}{t_i} p_i^{t_i} q_i^{s_i - t_i}.$$

Методика определения производительности w подобной системы для любых состояний была описана в § 7.5.

Таким образом, нам известны все составляющие выражения для Q и коэффициент жизнеспособности может быть рассчитан для каждой конкретной реализации.

Задачу синтеза структуры МПС повышенной жизнеспособности целесообразно разбить на два этапа. На первом этапе определяют минимальную конфигурацию системы без учета отказов устройств, достаточную для обработки в МПС в течение заданного времени T входного потока данных, на втором — дополнительное количество ММ, блоков ЗУи каналов связи для удовлетворения условиям задачи синтеза.

В качестве функции ограничения в задаче максимизации Q в пределах отпущенных средств используем функциональную зависимость $V = \alpha X + \beta Y + \gamma K$, где α , β , γ — соответственно аппаратные затраты ММ, блока ЗУ, канала связи. Приведем алгоритм решения задачи синтеза.

А л г о р и т м 7.2. На первом этапе, исходя из значений $N = N_{\max}$ в соответствии с результатами, полученными в § 7.6, решают оптимизационную задачу определения числа X микропроцессорных модулей, Y блоков ЗУ, K каналов связи по критерию максимальной производительности и определяют значение

$$\rho = \lambda E[T], \text{ где } \lambda = \sum_{i=1}^m \lambda_i.$$

Если $\rho < 1$, то уменьшая N на единицу, следует повторить весь процесс решения. Цикл прерывается на шаге, дающем значением $\rho \geq 1$. В случае $\rho \geq 1$ уже на первом шаге необходимо пересмотреть назначение программ обработки $(L_i^{(l)}, u_i^{(q)}) \in R_i$ [в множество R_i входит также программа $(L_i^{(0)}, u_i^{(0)})$, исключая заявку s_i из дальнейшей обработки].

После определения минимальной комплектации МПС проводят второй этап решения, аналогичный решению задачи оптимального резервирования. На этом этапе выбирают набор дополнительных устройств, включение которых в горячий резерв МПС максимизирует коэффициент жизнеспособности системы.

Переключение устройств из числа дополнительных в рабочую конфигурацию осуществляется с помощью специального устройства

реконфигурации и соответствующих программ операционной системы. Система связей должна допускать в этом случае соответствующее изменение адресации устройств в оперативном режиме.

Увеличения коэффициента жизнеспособности МПС можно достичь путем повышения производительности решения задач в МПС за счет включения в рабочую конфигурацию вычислительного ядра системы специализированных ММ (дискретных фильтров, спецпроцессоров быстрого преобразования Фурье и др.). Введение спецпроцессоров в состав МПС может существенно повысить производительность системы при меньших материальных затратах по сравнению с дополнительными универсальными ММ. В системах подобного типа с целью сохранения общей логической схемы обработки и упрощения устройств диспетчеризации, от которых требуется высокая пропускная способность и надежность, целесообразна параллельно-конвейерная организация вычислительного процесса.

При использовании в микропроцессорных модулях ППЗУ с электрической перезаписью возможна оперативная перестройка (специализация) ММ на эффективное решение отдельных задач. Если емкость ППЗУ недостаточна для хранения всех программ обработки информации, то целесообразно каждый ММ специализировать на какой-либо набор микрокоманд и организовать конвейерную обработку информации. Все это позволяет реализовать возможность построения МПС повышенной жизнеспособности.

§ 7.8. ЗАДАЧИ ПОСТРОЕНИЯ АДАПТИВНЫХ СИСТЕМ КОНТРОЛЯ РЭС

Проектирование автоматизированных систем контроля РЭС идет по различным направлениям. Развитие и широкое внедрение средств микропроцессорной техники открывает перед разработчиками аппаратуры АСК возможность построения адаптивных систем контроля. Под *адаптацией* понимают процесс накопления и использования информации в системе, направленный на достижение определенного, обычно оптимального в некотором смысле, состояния или поведения системы при изменяющихся внешних условиях. При адаптации могут изменяться параметры и структура системы, алгоритм функционирования, управляющие воздействия и т. п. Адаптация применяется в тех случаях, когда воздействующие на систему факторы являются полностью или частично неизвестными. В процессе адаптации система накапливает данные об этих факторах и определяет их характеристики.

Одним из основных элементов адаптивной системы является регулятор, который предназначен для реализации процесса компенсации влияния на систему дестабилизирующих факторов. Динамические свойства некоторых объектов полностью определяются внешними факторами, доступными прямому измерению (например, скорость объекта, напряжение).

Все адаптивные регуляторы делятся на два класса — самооптимизирующиеся и с эталонной моделью. Задачей *самооптимизирующихся регуляторов* является достижение наилучшего качества управления при заданном критерии оптимальности и наличии определенной информации об объекте и его сигналах (рис. 7.13). Обозначения на рисунке: Р — регулятор, ОК — объект контроля или управления. Процесс адаптации в системах с самооптимизирующимися регуляторами состоит из этапов идентификации объекта (ИО), расчета регулятора (РР), а также настройки регулятора или изменения его структуры.

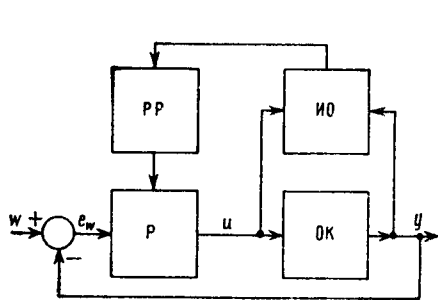


Рис. 7.13. Структура самооптимизирующего адаптивного регулятора

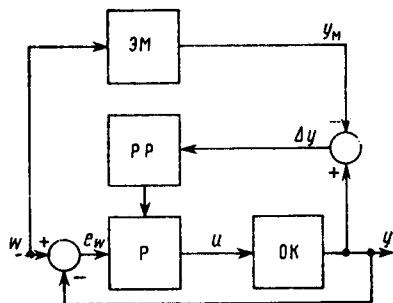


Рис. 7.14. Структура адаптивного регулятора с эталонной моделью

Задача *регуляторов с эталонной моделью* (рис. 7.14) состоит в получении такой реакции замкнутого контура управления на определенный входной сигнал, которая была бы максимально близкой к реакции на этот же сигнал заданной эталонной модели (ЭМ). Очевидно, данный принцип адаптации предполагает наличие некоторого измеряемого внешнего сигнала, причем адаптация производится лишь в те периоды, когда этот сигнал начинает изменяться. В таких системах процесс адаптации состоит из эталонов сравнения реакций замкнутого контура управления и эталонной модели, расчета регулятора и его настройки. В адаптивных системах контроля алгоритм функционирования системы может быть адаптирован либо к изменению параметров контролирующей аппаратуры, либо к изменению параметров контролируемых РЭС. В первом случае в состав аппаратуры контроля вводят контуры самонастройки и логические средства, с помощью которых в процессе функционирования определяется отклонение каких-либо параметров системы контроля от их номинального значения и производится коррекция параметров схмотехническими корректирующими средствами. Во втором случае в состав программно-аппаратных средств систем контроля вводят программные или аппаратные коммутаторы опроса контролируемых параметров, причем изменение закона функционирования этих коммутаторов коррелировано с изменением технического состояния контролируемых РЭС.

Адаптивные системы контроля содержат дополнительные датчики (блок датчиков внешних воздействий), учитывающие воздействие окружающей среды, а также дополнительные устройства памяти для хранения кодов значений границ зон неопределенности, годности, негодности состояния объекта контроля и зависимости погрешностей структурных элементов (преобразователей, генераторов стимулирующих сигналов) от факторов окружающей среды. Состав блока датчиков внешних воздействий и емкость дополнительной памяти могут изме-

няться в зависимости от требований к значению достоверности системы контроля. При проектировании адаптивных систем контроля требуется решать задачу определения необходимого количества элементов структуры системы, подвергаемых адаптации, т. е. тех преобразователей формы информации и генераторов стимулирующих сигналов, показания которых корректируются в зависимости от воздействия внешней среды на систему контроля.

Синтез структур адаптивных систем контроля базируется на модульном принципе построения аппаратуры, который предусматривает создание научно обоснованных и рационально ограниченных рядов унифицированных приборов и устройств с единой нормативной базой, включающей унификацию информационных сигналов, метрологических, надежностных, энергетических и других характеристик, необходимого математического обеспечения и конструктивного исполнения.

Устройства объединяются во взаимосвязанные, имеющие метрологическую, информационную, конструктивную и эксплуатационную совместимость агрегатные комплексы средств автоматизированного контроля.

Унификация и совместимость приборов в рамках модульного принципа построения автоматизированных систем контроля дают возможность резко сократить сроки проектирования, повысить надежность в процессе их эксплуатации. Модульный принцип позволяет изменять конфигурацию систем контроля путем варьирования набора унифицированных модулей, что обеспечивает адаптацию аппаратуры контроля к условиям ее эксплуатации.

При синтезе систем контроля необходимо учитывать ряд требований, предъявляемых к различным параметрам аппаратуры: точности и достоверности контроля, массе и габаритам аппаратуры, эксплуатационным и надежностным характеристикам, унификации и стандартизации блоков и узлов аппаратуры.

Отметим, что требуемые точность и быстродействие в современных системах обработки информации, в том числе реализованных на микропроцессорной элементной базе, обеспечиваются схемотехническими методами, вследствие чего значительно возрастают габариты и стоимость приборов. При этом следует учитывать неравномерность развития основных компонентов системы, заключающуюся в том, что средства цифровой обработки информации и управления процессом контроля развиваются более интенсивно по сравнению с датчиками, преобразователями, аналоговыми компонентами, устройствами регистрации, каналами связи.

Естественно считать, что перераспределение функций между элементами структурной схемы системы контроля должно регулярно производиться именно с учетом темпов такого развития. В силу этого появляется возможность возложения дополнительных функций на элементы цифровой обработки информации, входящие в состав аппаратуры контроля (на микроЭВМ, микропроцессорную систему и т. д.).

Одной из функций является компенсация погрешностей измерений, обусловленных воздействием дестабилизирующих факторов — температуры, влажности, давления и т.п. С этой целью для каждого элемента структурной схемы (преобразователя, генератора стимулирующих сигналов и др.) определяют зависимости погрешности от факторов окружающей среды с последующим использованием этих зависимостей для компенсации погрешностей путем соответствующей обработки информации на микроЭВМ или в МПС. При этом осуществляется централизованная программная коррекция, требующая для своей реализации структурной избыточности из-за введения дополнительного блока питания и датчиков дестабилизирующих воздействий.

Поэтому в ряде случаев введенная избыточность превышает выигрыш от улучшенного качества контроля и адаптивная система может оказаться экономически невыгодной. Кроме того, увеличивается время контроля за счет опроса датчиков и дополнительной программной обработки, т. е. время контроля может оказаться больше заданного. В связи с этим возможен комбинированный вариант системы контроля, включающей как приборы схмотехнической коррекции, так и приборы адаптивного типа со встроенными микропроцессорами. Задача структурного синтеза комбинированной системы контроля может быть сформулирована следующим образом: минимизировать значение целевой функции $V(X, t_i) \rightarrow \min$ при ограничениях

$$D(X) \geq D_{\text{тз}}(X);$$

$$T(X, t_i) \leq T_{\text{тз}}(X, t_i),$$

где V — объем аппаратуры, выраженный в условных элементах; D — достоверность контроля; T — время контроля; X — вектор контролируемых параметров. В качестве дополнительного варьируемого фактора используется быстроедействие t_i структурных элементов.

Точность решения задачи структурного синтеза зависит от многих факторов, в первую очередь от степени детализации при формировании функций $V(X, t_i)$, $D(X)$ и $T(X, t_i)$ и выбранного метода решения сформулированной задачи нелинейного программирования.

На этапе структурного синтеза решается также задача выбора способа реализации учета дестабилизирующих факторов. Идея учета дестабилизирующих факторов заключается в следующем. При калибровке измерительных приборов и преобразователей информации для каждой компоненты блока измерительных преобразователей снимается зависимость погрешности измерения от возмущающих воздействий. Эти зависимости заносятся в ПЗУ. В процессе контроля на основании показания датчиков $D_1, \dots, D_n$ проводится корректировка результатов измерения параметров путем считывания из ПЗУ соответствующих значений поправок. Скорректированные текущие значения контролируемых параметров поступают на регулирующее устройство Р, которое выдает управляющее воздействие на контролируемые РЭС. Достоинством такой АСК является достижение более высокой достоверно-

сти контроля, а недостатком — увеличение времени контроля за счет осуществляемой последовательно коррекции измеряемых значений контролируемых параметров.

Уменьшения времени контроля можно достичь за счет построения адаптивной АСК на базе приборов со встроенными микропроцессорами в составе каждого или некоторой части блоков измерительных преобразователей. Это позволяет реализовать корректировку значений контролируемых параметров автономно на уровне каждого прибора. Однако вводимая в такие структуры АСК аппаратная избыточность (дополнительные блоки памяти, датчики дестабилизирующих воздействий, встроенные микропроцессоры и др.) существенно увеличивает общую стоимость аппаратуры контроля.

Контрольные вопросы и задания

1. Объясните необходимость предварительной обработки информации в автоматизированных системах контроля РЭС.
2. Какими факторами объясняется широкое распространение магистрального интерфейса в МПС?
3. Какие средства аппаратной поддержки требуются для организации режима активной памяти?
4. Чем вызвана необходимость осуществления в МПС режимов активной и пассивной памяти?
5. В каких случаях на производительность МПС оказывают существенное влияние потери производительности из-за конфликтов с общей памятью и в каких — из-за конфликтов с общей шиной?
6. Объясните физическую сущность всех компонентов, входящих в функционал (7.19), характеризующий производительность.
7. Объясните физическую сущность термина «жизнеспособность» МПС.
8. Почему поиск оптимума функционала (7.19) следует проводить на целочисленной решетке значений переменных проектирования?
9. Что понимают под термином «адаптация» при построении адаптивных систем контроля?
10. Каким образом распределение программной информации по секциям общей памяти влияет на производительность МПС?
11. Определите параметр t для различных МПК БИС, например КР580, КМ1810, КМ1804, К588.
12. Перечислите факторы, влияющие на определение емкости буферной памяти, и объясните взаимосвязь между ними.
13. Объясните суть основных приемов, направленных на повышение производительности магистральных МПС.
14. Предложите структуру арбитра магистрали для разрешения конфликтных ситуаций.

МИКРОПРОЦЕССОРЫ И МИКРОЭВМ В СИСТЕМАХ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

§ 8.1. РОЛЬ МИКРОЭВМ И МПУ В СИСТЕМАХ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ

Функции микроЭВМ и МПУ в системах управления технологическими процессами. Микроминиатюризация и постоянное расширение номенклатуры РЭС, обилие разнообразного оборудования в технологии производства микроэлектронных РЭС (эпитаксиальное, диффузионное, вакуумное, плазмохимическое и др.) требуют четкого контроля работоспособности оборудования и управления всем технологическим процессом. На микроЭВМ и МПУ, работающих в составе технологического оборудования, возлагаются как традиционные функции управления, так и ряд дополнительных функций.

Диагностика и контроль — одна из основных функций систем управления технологическими процессами (ТП), предусматривающая самодиагностику системы управления (включая микропроцессорную часть), а также диагностику и контроль всего технологического оборудования.

Проверка и диагностика состояния отдельных систем оборудования (вакуумной, газовой, шлюзовой и др.) могут быть проведены путем укороченного рабочего цикла (в установках циклического действия) или проверки правильности функционирования всех систем на этапе подготовки их к рабочему циклу. При проектировании диагностики оборудования в ряде случаев необходимо предусматривать установку специальных датчиков, фиксирующих состояние систем. Некоторые параметры могут быть проверены алгоритмически, например скорость откачки рабочего объема определяют по заданному давлению и времени его достижения. Состояние механических систем также можно определить по времени исполнения команды.

Очень важной функцией является метрологическая диагностика и ее конечный результат — метрологическая аттестация оборудования.

Программирование — функция микроЭВМ и микропроцессорных систем, позволяющая оперативно изменять значения параметров и цикл технологического процесса, что крайне необходимо в условиях многономенклатурного производства.

Исполнение программы управления — основная функция системы управления, реализующая управление последовательностью технологического цикла и параметрами процесса. Следует отметить, что при

разработке конкретных систем управления технологическими процессами возможны различные сочетания технических средств: микропроцессорных контроллеров управления циклом, аналоговых регуляторов, устройств прямого цифрового управления и т. д.

Применение в системах управления микропроцессоров и микроЭВМ предоставляет возможность разработчику реализовать алгоритмы управления параметрами процесса, которые традиционными средствами управления реализовать невозможно или очень сложно. Можно строить системы с переменными параметрами настроек различных законов регулирования, реализовывать алгоритмы систем управления с переменной структурой, алгоритмы адаптивного управления при изменении параметров объекта управления в течение технологического цикла и т. п.

Важной функцией является *корректировка параметров* процесса по результатам межоперационного контроля и технологических условий. На микропроцессорную систему могут быть возложены и *специальные функции*: расчетные — для определения момента изменения или окончания процесса, профилирование температурных полей, многосвязное регулирование и др. Разнообразие таких функций велико и их количество непрерывно возрастает.

Отображение и документирование параметров процесса — функция, необходимая как для обработки технологических процессов, так и в производственных условиях для контроля правильности протекания процесса и его паспортизации.

Микропроцессорная система должна осуществлять *связь с ЭВМ* более высокого уровня иерархии для включения в состав локальной вычислительной сети управления ТП.

Задача управления технологическим процессом. Автоматизация технологических процессов производства РЭС вызвана рядом факторов, основными из которых являются: жесткие требования к ТП (особенно при производстве микроэлектронных РЭС), связанные с созданием в минимальном объеме твердого тела или на его поверхности максимального количества строго определенных областей с заданными геометрией, составом и свойствами; повышение рентабельности при снижении расходов дорогостоящих материалов; увеличение номенклатуры выпускаемых микроэлектронных РЭС и повышение их сложности.

Автоматизация технологических процессов базируется на моделях и методах оптимального управления. Главную роль при реализации методов оптимального управления играют микроЭВМ и МПУ, на которые возлагаются функции обработки информации и выдачи необходимых управляющих воздействий на исполнительные устройства управления технологическим процессом.

Рассмотрим структуру одноканальной системы автоматического управления технологическим процессом (рис. 8.1). Управляющее воздействие $u(t)$ формируется управляющим устройством на основе сравнения действительного состояния $x(t)$ технологического процесса с заданным $x_d(t)$. Обычно управляющее устройство в подобных системах

вырабатывает сигнал управления $u(t)$, являющийся функцией меры ошибки:

$$u(t) = u\{H[x_d(t), x(t)]\}, \quad (8.1)$$

где $H[...]$ — $x_d(t) - x(t)$ — мера ошибки, характеризующая разность между действительным и желаемым значениями параметра $x(t)$ технологического процесса.

Мгновенное значение сигнала управления $u(t)$ является функцией мгновенных значений выходного $x(t)$ и входного $x_d(t)$ сигналов. На рис. 8.2, а показан вид этих функций без прогноза характера их изменения. В системах оптимального управления учитываются рассогласования с учетом прогноза характера изменения параметра x (рис. 8.2, б). Чтобы

предсказанное управление было допустимым, необходимо учитывать интегральную меру ошибки J на интервале управления в будущем:

$$J = \int_t^{t+T} H[x_d(\tau), x(\tau)] d\tau, \quad (8.2)$$

причем очевидно, что критерий ошибки J следует минимизировать.

Во многих практических случаях применяется квадратичная мера ошибки вида

$$J = \int_t^{t+T} [x_d(\tau) - x(\tau)]^2 d\tau. \quad (8.3)$$

Отметим, что в оптимальных системах управления прогнозируется управляющий вход $u(t)$ в будущем на интервале $t \leq \tau \leq t+T$ (рис. 8.2) и управляющее устройство обрабатывает алгоритмы, направленные не на минимизацию критерия ошибки J . Алгоритмы функционирования управляющего устройства реализуют определенные законы управления, которые задаются с помощью дифференциальных уравнений во временной области. Законы управления описывают динамику технологических процессов. Существует множество способов описания динамики технологических процессов, из которых можно выделить два основных: 1) математическое описание процесса, основанное на физических законах, таких, как законы сохранения вещества, энергии, равновесия сил и т. п.; 2) математическое описание процесса, основанное на информации его экспериментального исследования.

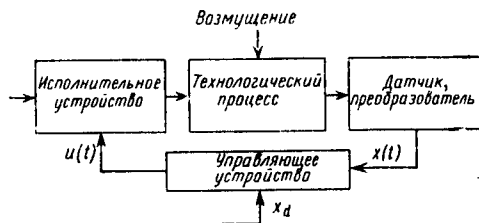


Рис. 8.1. Структура одноканальной системы автоматического управления технологическим процессом

Рассмотрим динамику технологического процесса. Сигналы управления $u(t)$ являются входными сигналами динамического процесса или независимыми переменными. Они генерируются системами, внешними по отношению к ТП, в частности, управляющим устройством, и влияют на его поведение. Физическое состояние ТП характеризуется

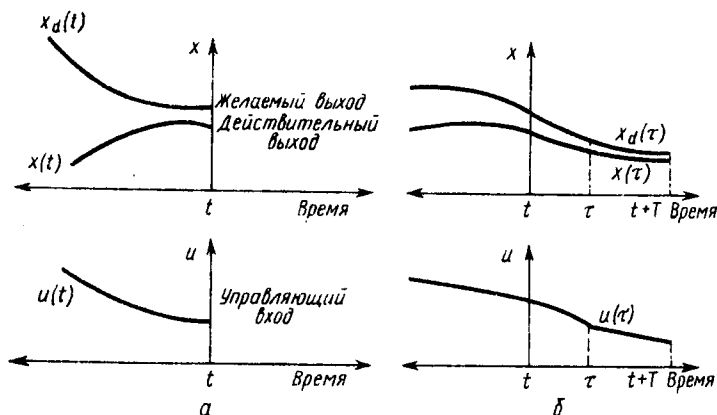


Рис. 8.2. Организация сигнала управления без прогноза (а); с прогнозом (б) характера их изменения

переменными состояния $x(t)$, отражающими внутреннее состояние ТП. Выбор сигналов состояния неоднозначен и, как правило, определяется возможностью их измерения. Сигналы с выхода датчиков, обеспечивающих текущий контроль ТП, называют наблюдаемыми переменными $y(t)$. Возмущения, действующие на ТП, обозначают $v(t)$. Возмущениями могут служить температура окружающей среды, влажность, вибрации и т. д. Переменные, влияющие на ТП и описывающие его, представлены на рис. 8.3.

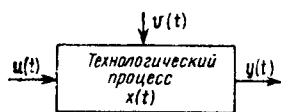


Рис. 8.3. Переменные, влияющие на технологический процесс и описывающие его

Динамика технологических процессов производства РЭС описывается в большинстве случаев обыкновенными дифференциальными уравнениями или дифференциальными уравнениями в частных производных. В любой момент времени t состояние ТП является функцией начального состояния $x(t_0)$ и вектора входного сигнала $u(t, t_0)$ и описывается уравнением

$$\dot{x}(t) = f[x(t), u(t), t], \quad (8.4)$$

или в скалярной форме

$$x_i(t) = f_i[x(t), u(t), t], \quad i = \overline{1, n}. \quad (8.5)$$

Эти уравнения обычно называют уравнениями состояния динамического процесса. Вектор выходного сигнала определяется соотношением

$$y(t) = g[x(t), u(t), t], \quad (8.6)$$

или в скалярной форме

$$y_j(t) = g_j[x(t), u(t), t]. \quad (8.7)$$

Эти выражения применимы в большинстве задач автоматического управления технологическими процессами производства РЭС.

Уравнение состояния обычно линеаризуют относительно его установившегося режима:

$$\dot{x}(t) = Ax(t) + Bu(t) + Cv(t),$$

где $x(t)$ — n -мерный вектор отклонения состояния; u — r -мерный вектор отклонения управления; v — s -мерный вектор изменения возмущений.

Уравнение выходного измеряемого сигнала может быть представлено в виде

$$y(t) = Dx(t) + w(t),$$

где $w(t)$ — m -мерный вектор вариации шумов измерения.

Элементы матрицы A характеризуют величины, обратно пропорциональные постоянным времени. Элементы матриц B и C пропорциональны коэффициентам усиления соответствующего контура, деленным на постоянные времени, а элементы матрицы D характеризуют передаточные свойства (коэффициенты усиления) измерительного прибора.

При описании динамики процесса в частотной области используют передаточные функции

$$W(p) = \frac{Y(p)}{U(p)} = \sum_{i=0}^{n-1} d_i p^i \bigg/ \sum_{i=0}^n a_i p^i,$$

где $Y(p)$ и $U(p)$ — преобразования Лапласа при нулевых начальных условиях функций $y(t)$, $u(t)$.

Типовые законы управления технологическими процессами. Широкое распространение в технике управления технологическими процессами получили типовые законы управления. Это объясняется тем, что типовые законы управления близки к оптимальным с точки зрения минимума среднеквадратической ошибки системы управления; кроме того, они просты по структуре, так как содержат элементарные операции типа дифференцирования и интегрирования.

Рассмотрим основные законы, широко используемые в системах прямого цифрового управления технологическими процессами.

Закон пропорционального управления (П-закон) записывается в виде

$$u(t) = W_{\text{П}}(p) x(t) = K_1 x(t), \quad W_{\text{П}}(p) = K_1,$$

или в дискретной форме

$$u[n] = K_1 x[n], \quad (8.8)$$

где $u[n] = u[nT]$ — значение выходного сигнала в дискретный момент времени $t_n = nT$; T — период квантования по времени; K_1 — коэффициент усиления регулятора (управляющего устройства согласно рис. 8.1).

Реализация П-закона позволяет осуществлять перемещение регулирующего органа в исполнительном устройстве пропорционально отклонению значения регулируемой величины.

Закон интегрального управления (И-закон) описывается формулами

$$\frac{du(t)}{dt} = K_2 x(t), \quad u(t) = K_2 \int_0^t x(t) dt,$$

или в дискретной форме

$$u[n] = K_2 \sum_{i=1}^n x[i], \quad (8.9)$$

где $K_2 = K_1 \Delta t / T_{\text{И}}$; Δt — время цикла регулирования; $T_{\text{И}}$ — постоянная времени интегрирования.

Передаточная функция имеет вид

$$W_{\text{И}}(p) = K_2 / p.$$

При реализации этого закона перемещение регулирующего органа осуществляется пропорционально интегралу от отклонения значения регулируемой величины, а скорость перемещения регулирующего органа в каждый момент времени пропорциональна отклонению значения регулируемой величины в тот же момент времени.

Закон пропорционально-интегрального управления (ПИ-закон, или изодромное регулирование) записывается таким образом:

$$u(t) = K_1 \left\{ x(t) + \frac{1}{T_{\text{И}}} \int_0^t x(t) dt \right\},$$

или в дискретной форме

$$u[n] = K_1 x[n] + K_2 \sum_{i=1}^n x[i]. \quad (8.10)$$

Передаточная функция имеет вид $W_{\text{ПИ}}(p) = K_1 \left(1 + \frac{1}{T_{\text{И}} p} \right)$.

Регулятор с таким законом функционирования осуществляет перемещение регулирующего органа пропорционально значению регулируемой величины и интегралу от его отклонения.

Закон пропорционально-интегрального управления с воздействием по производной (ПИД-закон) характеризуется передаточной функцией вида

$$W_{\text{пид}}(p) = K_1 \left(1 + \frac{1}{T_{\text{и}} p} + T_{\text{д}} p \right),$$

где $T_{\text{д}}$ — постоянная дифференцирования.

Уравнение непрерывного ПИД-регулятора записывается в виде

$$u(t) = K_1 \left\{ x(t) + \frac{1}{T_{\text{и}}} \int_0^t x(t) dt + T_{\text{д}} \frac{dx(t)}{dt} \right\},$$

или в дискретной форме

$$\begin{aligned} u[n] &= K_1 x[n] + K_2 \sum_{i=1}^n x[i] + K_3 (x[n] - x[n-1]) = \\ &= (K_1 + K_3) x[n] + K_2 \sum_{i=1}^n x[i] - K_3 x[n-1], \end{aligned} \quad (8.11)$$

где $K_3 = K_1 T_{\text{д}} / \Delta t$, $x[n-1]$ — значение переменной x в момент времени $t_{n-1} = (n-1) T$; коэффициенты K_1 , $T_{\text{и}}$, $T_{\text{д}}$ являются параметрами закона управления (параметрами регулятора) и определяются его настройкой.

Регулятор с таким алгоритмом работы осуществляет перемещение регулирующего органа в каждый момент времени пропорционально отклонению значения регулируемой величины, интегралу и производной отклонения.

Основной проблемой при реализации типовых законов управления является выбор параметров T , K_1 , K_2 и K_3 . Их значения, зависящие от параметров объекта, а также от требований к динамике и статике регулирования, рассчитывают методами теории автоматического управления.

В АСУ ТП используются разнообразные законы управления, по которым функционируют системы управления. Передаточные функции систем управления определяют по передаточным функциям входящих в их состав элементов и узлов, а также по виду их соединений. Элементами систем управления могут быть узлы, реализующие типовые законы управления: электронные и электромашинные усилители, электродвигатели, тахогенераторы и другие устройства. Сложные законы управления формируются из типовых с помощью алгебры передаточных функций. Алгебра передаточных функций — система правил, позволяющих определять передаточные функции системы в целом по передаточным функциям отдельных элементов, в которой можно выделить три характерных соединения: последовательное, параллельное и обратное.

Последовательным соединением элементов называется соединение, при котором выходная величина предшествующего элемента является входной величиной последующего (рис. 8.4).

Для последовательного соединения имеем

$$X_2(p) = W_1(p) X_1(p), \quad X_3(p) = W_2(p) X_2(p),$$

откуда

$$X_3(p) = W_1(p) W_2(p) X_1(p) = W(p) X_1(p)$$

и, значит,

$$W(p) = W_1(p) W_2(p).$$

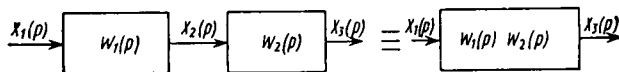


Рис. 8.4. Последовательное соединение элементов

В общем случае при последовательном соединении n элементов

$$W(p) = \prod_{i=1}^n W_i(p),$$

т. е. передаточная функция последовательного соединения элементов равна произведению передаточных функций отдельных элементов.

Параллельным соединением элементов называется соединение, при котором входная величина одна и та же для всех элементов, а их выходные величины суммируются (рис. 8.5).

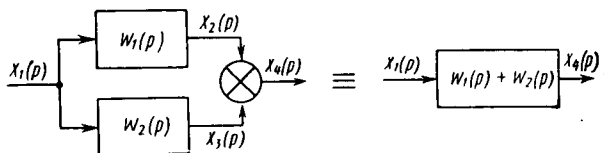


Рис. 8.5. Параллельное соединение элементов

Для параллельного соединения можно записать

$$X_2(p) = W_1(p) X_1(p), \quad X_3(p) = W_2(p) X_1(p), \quad X_4(p) = X_2(p) + X_3(p).$$

Тогда

$$X_4(p) = (W_1(p) + W_2(p)) X_1(p) = W(p) X_1(p)$$

и, значит,

$$W(p) = W_1(p) + W_2(p).$$

В общем случае при параллельном соединении n элементов

$$W(p) = \sum_{i=1}^n W_i(p),$$

т. е. передаточная функция параллельного соединения элементов равна сумме передаточных функций отдельных элементов.

Обратным соединением двух элементов или соединением обратной связи называется соединение, при котором выход каждого из элементов соединяется с входом другого элемента (рис. 8.6).

Для обратного соединения имеем

$$X_2(p) = W_1(p) X_1(p), \quad X_3(p) = W_2(p) X_2(p), \quad X_1(p) = X_0(p) \mp X_3(p).$$

Знак минус в выражении для $X_1(p)$ соответствует отрицательной, а знак плюс — положительной обратной связи.

Исходя из правила последовательного соединения элементов, следует, что

$$X_3(p) = W_1(p) W_2(p) X_1(p)$$

или

$$X_1(p) = \frac{1}{1 \pm W_1(p) W_2(p)} X_0(p)$$

и

$$X_2(p) = \frac{W_1(p)}{1 \pm W_1(p) W_2(p)} X_0(p) = W_{\text{OC}}(p) X_0(p).$$

Отсюда передаточная функция обратного соединения

$$W_{\text{OC}}(p) = \frac{W_1(p)}{1 \pm W_1(p) W_2(p)},$$

причем знак плюс в знаменателе означает отрицательную обратную связь, а знак минус — положительную.

Очевидно, что с помощью МПУ можно реализовать практически любую передаточную функцию системы. Алгоритм, реализующий такую передаточную функцию, будет существенно зависеть от вида объекта и требований к качеству процесса управления. Его разрабатывают индивидуально применительно к конкретной системе управления.

§ 8.2. РЕАЛИЗАЦИЯ ЗАКОНОВ УПРАВЛЕНИЯ НА МИКРОПРОЦЕССОРНОЙ ТЕХНИКЕ

Критерий качества системы управления. Целью автоматического управления технологическим процессом является достижение заданных параметров $x_d(t)$ технологического процесса посредством реализации некоторого закона управления и выдачи необходимых управляющих воздействий $u(t)$ на исполнительные органы, воздействующие на параметры ТП. Качество работы системы управления ТП можно, например, оценивать по квадратичной мере ошибки

$$H[x_d(t), x(t)] = [x_d(t) - x(t)]^2,$$

где $x(t)$ — фактическое состояние ТП.

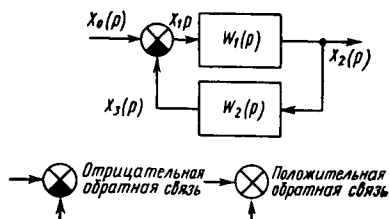


Рис. 8.6. Обратное соединение двух элементов

Квадрат разности $[x_d(t) - x(t)]^2$ характеризует меру отклонения реального ТП от его номинального (заданного) режима. Задача управления заключается в минимизации этого отклонения. В оптимальных системах управления ТП учитывают рассогласования между предсказанными значениями фактического выхода и желаемыми выходами в будущем. Для этой цели используют критерий качества, равный интегралу от квадрата меры ошибки на интервале управления в будущем:

$$J = \int_t^{t+T} [x_d(t) - x(t)]^2 dt. \quad (8.12)$$

В квадратичной мере ошибки обычно используются весовые коэффициенты q , придающие различные веса ошибке рассогласования, в частности

$$H = q_{11} [x_{1d}(t) - x_1(t)]^2 + q_{22} [x_{2d}(t) - x_2(t)]^2,$$

где q_{11} , q_{22} — веса рассогласования по координатам x_1 , x_2 .

Тогда выражение для критерия качества с учетом весовых коэффициентов приобретает вид

$$J = \int_0^{\infty} [x_d(t) - x(t)]^T \mathbf{Q}(t) [x_d(t) - x(t)] dt, \quad (8.13)$$

где $\mathbf{Q}(t)$ — диагональная матрица размерности $n \times n$ с положительными изменяющимися во времени элементами, учитывающая веса различных параметров ТП.

Выбор весовых коэффициентов меры ошибки представляет собой довольно трудную задачу из-за сложных взаимосвязей между параметрами ТП.

Практика проектирования систем управления ТП показывает, что квадратичные критерии качества обеспечивают достаточно простое решение задачи синтеза закона управления ТП, а также его техническую реализацию.

Проблемы синтеза оптимальных законов управления ТП. На любой технологический процесс производства РЭС, в частности микросистемных приборов, воздействуют случайные возмущения, например случайные примеси в исходном сырье, изменение характеристик оборудования, шумы измерения используемых датчиков и т. п. Кроме того, часто отдельные переменные состояния ТП не могут быть измерены непосредственно или измеряются с большой погрешностью. Случайные воздействия влияют и на процедуры измерения, и на различные операции технологического процесса. Поэтому для обеспечения оптимального управления технологическим процессом переменные состояния приходится оценивать по ансамблю измерений, каждое из которых является неточным. Для определения истинных значений переменных состояния необходимо производить фильтрацию измеренных параметров, причем она должна быть в определенном смысле оптимальной.

Структура системы оптимального управления представляет собой комбинацию оптимального фильтра и оптимального детерминированного регулятора, реализующих заданный закон управления по выбранному критерию оптимальности, например по минимуму квадратичного функционала:

$$J = \int_0^{\infty} \{ [x_d(t) - x(t)]^T Q(t) [x_d(t) - x(t)] + u^T(t) L(t) u(t) \} dt, \quad (8.14)$$

где $u(t)$ — r -мерный вектор управляющих входов; $L(t)$ — диагональная матрица размерности $r \times r$ весовых коэффициентов, зависящих от времени.

В системах управления, цель которых — поддержать действительный выход как можно ближе к постоянному желаемому выходу, элементы весовых матриц Q и L целесообразно выбирать постоянными.

Довольно удобным и эффективным является метод формирования системы управления в виде совокупности типовых элементарных звеньев, описываемых своими передаточными функциями $W_i(p)$. При формировании структурной схемы системы управления используют свойства последовательного и параллельного соединений элементарных звеньев. Напомним, что передаточная функция $W_c(p)$ системы, состоящей из последовательного соединения элементарных звеньев с передаточными функциями $W_i(p)$, $i = \overline{1, n}$, определяется в виде их произведения, т. е. $W_c(p) = \prod_{i=1}^n W_i(p)$, а передаточная функция системы, состоящей из n параллельно соединенных звеньев, — как сумма их передаточных функций, т. е. $W_c(p) = \sum_{i=1}^n W_i(p)$. При синтезе закона управления ТП необходимо определить способы соединения элементарных звеньев, а также значения коэффициентов и постоянных времени в их передаточных функциях.

Особенности микропроцессорного управления. Микропроцессорное управление завоевало прочные позиции в системах управления ТП РЭС (особенно при их микроэлектронном исполнении): в системах управления процессами окислирования, диффузии, зонной очистки, управления тепловыми процессами, а также различными технологическими установками. По сравнению с аналоговыми регуляторами микропроцессорные системы являются более надежными, компактными и обладают лучшими точностью и гибкостью.

Аппаратно-программная реализация цифровых регуляторов на микропроцессорной элементной базе позволяет получать унифицированные устройства, обладающие следующими свойствами: большим выбором законов управления; простотой сопряжения с ЭВМ верхнего уровня в иерархических системах управления; возможностью автоматического изменения режимов работы и установки коэффициентов в передаточных функциях программным путем или по командам ЭВМ

верхнего уровня; простотой и высокой точностью задания переменных коэффициентов настройки; повышенной стабильностью временных характеристик; высокой точностью обработки информации; малой стоимостью стандартного изделия серийного производства в сочетании с программной специализацией проектируемого устройства.

Технические характеристики микропроцессоров налагают ограничения на реализацию законов управления технологическими процессами, а именно: переменные состояния динамики ТП должны быть непосредственно связаны с легко измеряемыми и легко вычисляемыми свойствами управляемого ТП; параметры регулятора должны вычисляться с учетом информации о параметрах модели ТП без итераций с минимальной сложностью; непосредственное вычисление управляющей последовательности должно быть простым и быстрым, не содержащим трансцендентных функций и итеративных процедур.

Особое внимание следует уделять программному обеспечению микропроцессорных систем. При программировании задач управления нужно прежде всего учитывать условия реализации алгоритмов управления в реальном времени, так как преобразования входной информации и ее обработка должны выполняться в строго ограниченные сроки. Большая связность программ управления различных параметров технологического процесса обуславливает большое многообразие функциональных задач, связанных общими переменными. При этом возникает необходимость в синхронизации отдельных выполняемых программ или их частей с разнообразными внешними событиями.

При выборе МПК БИС для построения системы управления технологическими процессами особое внимание следует уделять таким параметрам, как производительность, многоуровневые прерывания, развитая система ввода — вывода, системное асинхронное управление, наличие аппаратурных средств поддержки, наличие канала прямого доступа к памяти, возможность автоматического перезапуска после сбоя, наличие системных средств защиты питания.

Подчеркнем, что с созданием новых МПК БИС и разнообразных аппаратурных средств поддержки технические характеристики микропроцессоров интенсивно совершенствуются и роль перечисленных требований к МП постоянно снижается.

В качестве примера рассмотрим организацию микропроцессорного управления технологическими потоками, широко используемую в микроэлектронной технологии производства РЭС.

§ 8.3. МИКРОПРОЦЕССОРНОЕ УПРАВЛЕНИЕ ТЕХНОЛОГИЧЕСКИМИ ПОТОКАМИ

В микроэлектронной технологии для нанесения вещества на поверхность подложки широко используется активная фаза потоков газовой и парогазовой смесей: диффузия в газе носителя, эпитаксиальное наращивание за счет осаждения атомов кремния из газовой фазы, разложение паров и т. п.

Рассмотрим организацию микропроцессорного управления технологическими потоками.

Для измерения количества расходуемой жидкости (газа) наибольшее распространение получили диафрагменные измерители. Их действие основано на законе Бернулли, связывающем перепад давления Δp с сопротивлением диафрагмы потоку z , удельным весом вещества потока γ и скоростью потока W :

$$\Delta p = z \frac{\gamma}{2g} W^2 \cdot 10^{-4},$$

где $g = 9,81 \text{ м/с}^2$ — ускорение свободного падения.

Измеряя перепад давлений на диафрагме и зная ее сопротивление потоку при фиксированном отверстии, можно определить скорость потока и, следовательно, его расход. Для расширения диапазона измерений применяют измерители с отверстием, изменяемым соответственно изменению скорости потока. В качестве такого измерителя служит клапанное устройство, проницаемость которого описывается коэффициентом K_v . Этот коэффициент зависит от положения заслонки и определяется из уравнения $f = K_v \sqrt{\Delta p / \gamma}$, где f — расход жидкости через клапан. Коэффициент K_v и, следовательно, размер отверстия измерителя соответствуют определенному перепаду давления Δp на клапане. При изменении перепада давления Δp необходимо производить соответствующую корректировку значения K_v с помощью управляющего воздействия $\Delta \alpha$, которое исполнительный механизм преобразует в положение клапана α . Расход жидкости является функцией перепада давления Δp и положения заслонки клапана α . Оценку потока находят из уравнения $f = K_v(\alpha) \sqrt{\Delta p / \gamma}$.

Техническая реализация системы управления технологическим потоком показана на рис. 8.7. Система включает в себя измерительный комплекс — датчики перепада давления и положения клапана; исполнительное устройство — электродвигатель; два АЦП, преобразующих непрерывные сигналы с выходов датчиков; микропроцессор, формирующий управляющую последовательность; пульт управления и дисплей. Для измерения положения заслонки клапана используется потенциометрический или емкостный датчик, связанный с осью заслонки клапана. Измерение расхода жидкости основано на измерении перепада давления на регулирующем клапане и осуществляется с помощью дифференциальных манометров. Положение заслонки

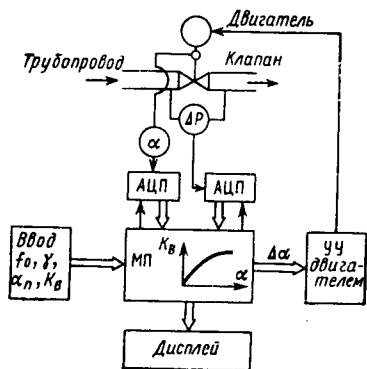


Рис. 8.7. Схема управления технологическим потоком

клапана управляется двигателем с помощью тиристорного устройства управления.

В запоминающем устройстве микропроцессора хранятся зависимость $K_b = \varphi(\alpha)$, программы регулирования, а также все необходимые константы и данные. В микропроцессоре производятся все вычисления, по окончании которых генерируется управляющее воздействие $\Delta\alpha$.

Замена параметров или запуск сервисных программ (градуирования, полного открытия или закрытия клапана, проверки параметров) обеспечивается программой прерываний. Ввод программ осуществляется с пульта управления.

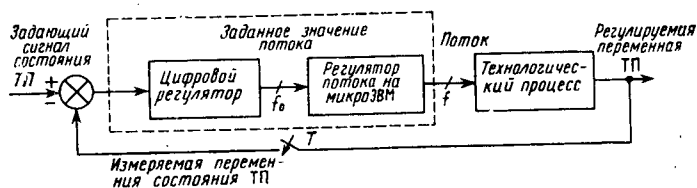


Рис. 8.8. Схема управления технологическим процессом на основе регулятора потока

Микропроцессорный регулятор потока может работать как автономно, так и в замкнутых контурах управления температурой, концентрацией, уровнем и т. п. (рис. 8.8). В замкнутом контуре задающий сигнал потока f_0 формируется цифровым регулятором и подается на микропроцессорный регулятор потока. Очевидно, микропроцессор, управляющий потоком, может быть использован также для реализации прямого цифрового управления (реализация ПИД и других алгоритмов) замкнутой системой. Нетрудно показать, что в подобных системах управления технологическими процессами использование микропроцессорного управления экономически оправдано.

§ 8.4. МИКРОПРОЦЕССОРНОЕ УПРАВЛЕНИЕ ТЕХНОЛОГИЧЕСКИМ ПРОЦЕССОМ НА ОСНОВЕ ТЕКУЩЕГО РЕГРЕССИОННОГО АНАЛИЗА

На любой технологический процесс оказывает влияние множество случайных факторов (неточность оборудования, примеси в сырье, неточности режущего инструмента и приспособлений, внутренние напряжения обрабатываемой детали, разброс параметров радиоэлектронных элементов при монтаже схем и т. п.). Поэтому параметры изготавливаемых изделий являются случайными величинами и при этом, как правило, не известны функциональные зависимости влияния действующих случайных факторов на изменение параметров изделий. Однако в подобных случаях очень важно учитывать характер взаимосвязи между случайными величинами. Для количественного выражения этой

взаимосвязи служат регрессия и корреляция. Остановимся более подробно на этих понятиях.

Пусть x и y — случайные величины, характеризующие параметры некоторого изделия, причем упорядоченная пара (x, y) характеризует параметры одного варианта изделия и может быть изображена точкой на плоскости. Полная совокупность вариантов изображается множеством точек на плоскости. Математические ожидания случайных величин x и y равны соответственно $\bar{x}$, $\bar{y}$, а среднеквадратические отклонения σ_x и σ_y характеризуют рассеивание величин x , y относительно их математических ожиданий.

Рассмотрим зависимость $\bar{y}(x)$, являющуюся условным математическим ожиданием $M(y|x)$. Используя выражение для условного математического ожидания и обозначая через $p(x, y)$ совместную вероятность данных значений x, y , находим

$$\bar{y}(x) = M(y|x) = \frac{\sum_y y p(x, y)}{\sum_y p(x, y)}. \quad (8.15)$$

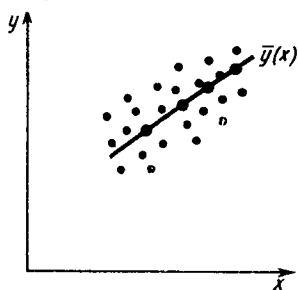


Рис. 8.9. Линия регрессии

Определяя $\bar{y}(x)$ при различных x , можно построить функцию, графически выражающую эту зависимость и называемую функцией регрессии y по x . Аналогично может быть получена зависимость $\bar{x}(y)$, называемая регрессией x по y .

На практике наиболее часто встречается случай линейной регрессии (рис. 8.9), уравнение которой записывается в виде

$$\bar{y}(x) = a + b(x - \bar{x}). \quad (8.16)$$

Коэффициенты a и b выбирают таким образом, чтобы получилась наибольшая концентрация точек (x, y) вблизи прямой $\bar{y}(x)$, что выражается условием

$$\Phi(a, b) = M\{[y - \bar{y}(x)]^2\} = \min. \quad (8.17)$$

Условие (8.17) с учетом (8.16) дает систему уравнений для определения коэффициентов a и b :

$$\begin{cases} \bar{y} - a = 0, \\ M[y(x - \bar{x})] - b\sigma_x^2 = 0. \end{cases} \quad (8.18)$$

Величина $\mu_{xy} = M[y(x - \bar{x})]$ называется ковариацией между x и y , которая служит мерой взаимной связи между случайными величинами x, y .

Из (8.18) находят значения a и b , определяющие линию регрессии: $a = \bar{y}$, $b = \mu_{xy}/\sigma_x^2$.

Ковариация μ_{xy} зависит от дисперсий самих случайных величин, поэтому для оценки взаимосвязи между случайными величинами более удобным оказывается использование коэффициента корреляции $r_{xy} = \mu_{xy} / \sigma_x \sigma_y$, который может изменяться от нуля для независимых случайных величин до единицы, если случайные величины связаны линейной функциональной зависимостью.

Для иллюстрации рассмотрим организацию управления технологическим процессом на основе текущего регрессионного анализа на при-

мере управления технологическим процессом производства магнитоуправляемых контактов (МК). Основная задача производства МК — получение изделий с заданными числом ампервитков срабатывания θ и зазором S между контактами.

Схема управления заваркой лепестков МК приведена на рис. 8.10. В ее состав входят пульта статистического контроля параметров θ и S , а также микроЭВМ для проведения статистического анализа и выработки на его основе управляющих воздействий на технологический процесс.

Технологический процесс установки зазора по заданным ампервиткам срабатывания состоит в следующем:

- 1) заваривается верхний пружинный контакт (ВК);
- 2) нижний пружинный контакт (НК) отводится исполнительным механизмом на максимальное расстояние;
- 3) в катушку K генератором тока возбуждения МК подается ток, значение которого обеспечивает заданное число ампервитков срабатывания МК и определяется микроЭВМ;
- 4) начинается движение контакта НК к контакту ВК.

Движение контактов происходит до момента их замыкания под воздействием поля катушки K . Напряженность поля пропорциональна протекающему в катушке току I . При замыкании регистратор срабатывает и останавливает исполнительный механизм;

- 5) заваривается нижний пружинный контакт НК, зазор становится равным S .

Исследования показывают, что величины θ и S после заварки не связаны жесткой функциональной зависимостью. Геометрическое место точек $\theta(S)$ зависит от многих факторов — качества исходного материала, режимов операции заварки и т. д.

С пультов статистического контроля данные экспериментов поступают в управляющую микроЭВМ, в которой непрерывно строятся регрессионные модели и вырабатываются управляющие воздействия на

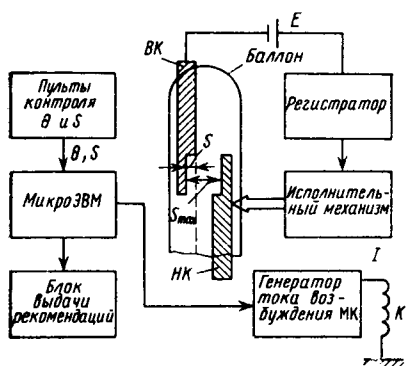


Рис. 8.10. Схема управления заваркой лепестков магнитоуправляемого контакта

технологический процесс (корректируется значение тока в катушке, выдаются различные рекомендации).

Математически процесс управления заваркой МК можно свести к следующей модели. Пусть θ_T и S_T — требуемые число ампервитков срабатывания и зазор между контактами. Для решения задачи оптимизации сформулируем целевую функцию вида

$$F(\theta, S) = \left(\frac{S_T - S}{S_T} \right)^2 + \left(\frac{\theta_T - \theta}{\theta_T} \right)^2,$$

которую следует минимизировать с учетом ограничения $\theta = kS + b$, определяемого уравнением регрессии.

Сформулированная задача является задачей квадратичного программирования, которую можно решить с помощью неопределенных множителей Лагранжа.

Функция Лагранжа имеет вид

$$F(\theta, S, \lambda) = \left(\frac{S_T - S}{S_T} \right)^2 + \left(\frac{\theta_T - \theta}{\theta_T} \right)^2 + \eta(\theta - kS - b).$$

Взяв частные производные, получим

$$\begin{aligned} \frac{\partial F}{\partial S} &= -2 \left(\frac{S_T - S}{S_T} \right) - k\lambda = 0, \\ \frac{\partial F}{\partial \theta} &= -2 \left(\frac{\theta_T - \theta}{\theta_T} \right) + \lambda = 0, \end{aligned} \tag{8.19}$$

$$\frac{\partial F}{\partial \lambda} = \theta - kS - b = 0.$$

Решением системы будут значения S^* и θ^* , при этом θ^* косвенно определяет ток I в катушке.

Пусть $\theta_T = 40 \pm 2$ А/м, $S_T = 10 \pm 2_{\text{мкм}}$, $\theta = 3S + 6$, число витков в катушке $w = 50$.

Решая систему уравнений (8.19), получаем $\theta^* = 40,44$ А/м, $S^* = 10,48$ мкм.

Оптимальные значения θ и S находятся в поле допуска. Необходимый ток в катушке $I = 0,8$ А. После установки тока $I = 0,8$ А строят новую регрессионную зависимость и рассчитывают новое значение I . В результате многократной коррекции тока I находят такое значение тока, при котором процент выхода МК, отвечающих требованиям технических условий, будет максимальным.

§ 8.5. ОРГАНИЗАЦИЯ РАСПРЕДЕЛЕННЫХ МИКРОПРОЦЕССОРНЫХ СИСТЕМ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ

Внедрение средств микропроцессорной техники в автоматизированные системы управления ТП привело к созданию надежных, гибких, простых в обращении и высокоэффективных систем управления. Эти показатели достигаются наличием сетевых структур, модульностью технических и программных средств, распределенностью функций контроля и управления, удобным человеко-машинным интерфейсом, автоматизацией проектирования АСУ ТП для конкретной реализации как технических, так и программных средств.

Главными особенностями микропроцессорных систем управления ТП являются децентрализация сбора и переработки информации, распределение вычислительных ресурсов с одновременным приближением их к рабочим местам пользователей. При этом требуется не только высокопроизводительная обработка данных, но и высокоскоростные системы связей, обеспечивающие эффективное резервирование ресурсов системы за счет быстрой транспортировки данных и программ. Указанным требованиям удовлетворяют распределенные микропроцессорные системы, построенные на базе локальных вычислительных сетей (ЛВС).

Организация ЛВС для управления ТП. В АСУ ТП применяются малые ЛВС с простым и дешевым сетевым контроллером, пригодные для включения в сеть как сложных, так и простых устройств, эффективные в случае обмена короткими сообщениями при жестких временных ограничениях, учитывающих срочность этих сообщений. Малые ЛВС (МЛВС) допускают организацию связей до 2 ... 3 км. Архитектура МЛВС приведена на рис. 8.11.

Выбор в МЛВС конкретной структуры связей и способа резервирования линий связи зависит от выбора протоколов управления. Скорость передачи данных в МЛВС должна быть не хуже 2 ... 10 М бит/с. Следует отметить, что такая скорость обмена в МЛВС сейчас вполне достижима, а применение волоконно-оптических линий связи сдвигает эти скорости в диапазон выше 100 М бит/с.

Важными характеристиками, определяющими возможности МЛВС, являются *способ управления* доступом абонентов к общему каналу и протокол информационной связи. Для АСУ ТП управление доступом имеет свои особенности. С целью увеличения надежности сети и уменьшения времени реакции на поступающие заявки управление должно быть *децентрализованным*, динамически учитывать срочность предоставления доступа абонентов к каналу, т. е. основываться на *динамических приоритетах*. Управление должно обеспечивать доступ абонента к каналу в течение гарантированного времени, т. е. должно быть *быстродействующим и детерминированным*.

Технические средства систем управления ТП. Микропроцессорные системы управления ТП обладают чертами систем обмена данными

ми и распределенных тесно связанных средств обработки данных, поэтому номенклатура их технических средств довольно разнообразна (рис. 8.12).

Станции контроля и управления. Станции контроля и управления, обеспечивая децентрализованное управление ТП, являются составляющей частью распределенной микропроцессорной системы и обеспечивают управление определенными технологическими процессами и агрегатами. В этих станциях отсутствуют устройства ввода — вывода,

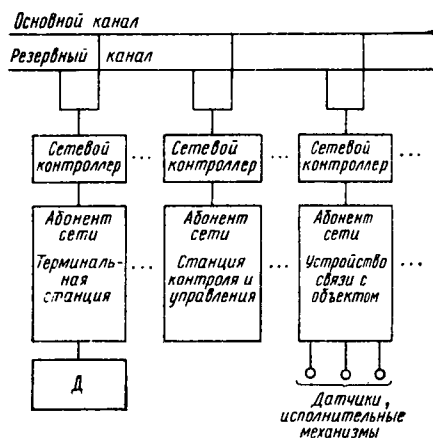


Рис. 8.11. Архитектура малой локальной вычислительной сети

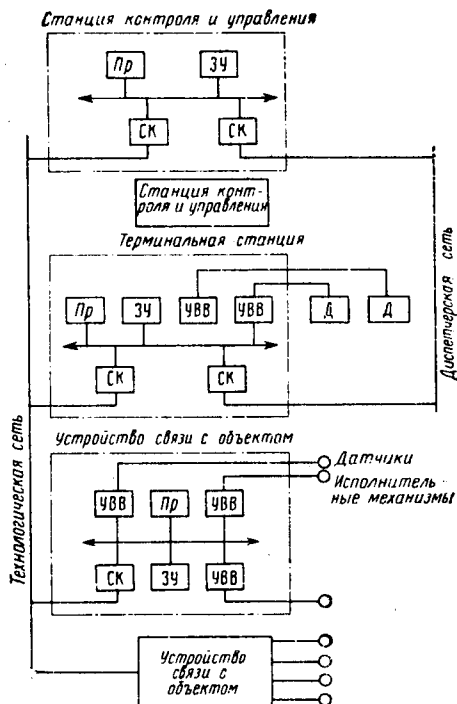


Рис. 8.12. Структура технических средств распределенной микропроцессорной системы управления технологическим процессом

что обеспечивает их максимальную унификацию. В качестве устройств обработки информации (процессоров Пр) применяются 8- или 16-разрядные микропроцессоры. Выбор разрядности определяется требованиями со стороны управляемого объекта.

Терминальные станции. Под терминальными станциями понимают технические средства, позволяющие осуществлять активный диалог человека-оператора с микропроцессорными управляющими комплексами и объектом. К ним относятся микропроцессоры, осуществляющие связь операторов производства с системами контроля и управления, а также различные средства пульта оператора. Терминальные станции должны обеспечивать ввод, вывод, отображение информации, изменение заданий цифровым регулятором, санкционированные пуск и останов отдельных программ, организацию печати, управление опера-

циями периодических процессов, изменение установок регуляторов, вывод графиков, мнемосхем и т. д. Для получения удобного человеко-машинного интерфейса терминальные станции снабжают простейшей клавиатурой, черно-белыми и цветными видеомониторами. На видеомониторах индицируется технологическая информация и текстовая поддержка, направляющая действия оператора при работе с клавиатурой, а также результаты диагностики элементов системы. На рис. 8.12 введены обозначения: УВВ — управление вводом—выводом; СК — сетевой контроллер; Д — дисплеи видеомониторов.

Средства связи с объектом. Характерной особенностью распределенных систем управления ТП является наличие полностью распределенных устройств связи с объектом (УСО), максимально приближенных к технологическому объекту автоматизации. С целью повышения надежности системы управления перечень УСО унифицируют, что позволяет разработчикам систем управления ТП выбирать и заказывать модули УСО на заводе-изготовителе в зависимости от информационной мощности и типов сигналов конкретных технологических процессов и агрегатов.

Средства дистанционного контроля и ручного управления. Наличие таких средств в системах управления ТП вызывается необходимостью ручного резервирования части функций автоматического контроля и управления. Для этого посты оператора — технолога оснащают регистраторами наиболее важных технологических параметров. Функции управления резервируются модулями ручного дистанционного управления аналоговыми и позиционными исполнительными механизмами. Эти средства контроля и управления размещают в непосредственной близости от агрегатов процесса с возможностью санкционированного доступа к ним с целью воздействия на объект управления.

Программное обеспечение распределенной системы управления ТП. Программное обеспечение (ПО) системы должно обладать: функциональной полнотой, достаточной для решения широкого круга задач управления заданным классом технологических объектов; высокой гибкостью, позволяющей простыми средствами настраивать его на условия конкретного применения; средствами настройки, рассчитанными на специалистов в области управления технологическими объектами; системными средствами для обеспечения работы системы управления в реальном времени; средствами динамической реконфигурации.

Для удовлетворения сформулированных требований в распределенных микропроцессорных системах реализуются принципы сетевого режима работы и модульности с учетом наличия средств генерации прикладного ПО.

Сетевой режим работы позволяет рассматривать все множество ресурсов распределенной системы как единую интегрированную систему, что упрощает перераспределение ресурсов в случае отказов некоторых из них, тем самым повышая надежность и живучесть системы.

Модульность позволяет свести создание программного обеспечения конкретной системы управления к выбору требуемых типовых за-

дач, настройки их на необходимые режимы работы и установлению связей между ними. Дополнение ПО новыми программными модулями необходимо для расширения функциональных возможностей системы, что в свою очередь является решающим фактором в обеспечении функциональной полноты.

Выбор типов процессоров в системах управления ТП. На распределенные микропроцессорные системы управления ТП возложены функции обработки информации о протекающем ТП и выдачи управляющих воздействий на ТП в реальном времени. В распределенных системах управления обработки информации производится минимум на двух иерархических уровнях. На нижнем уровне, максимально приближенном к технологическому оборудованию, производится сбор и предварительная обработка информации. Обработка информации на этом уровне дает возможность существенно снизить потоки информации, поступающей на втором уровне системы. Для преобразования информации на этом уровне служат либо микро-ЭВМ, либо специализированные процессоры. Характерно, что микроЭВМ и специализированные процессоры данного уровня работают с локальной памятью. Это связано с желанием уменьшить потери времени на борьбу за магистрали, общую память и на обмен информацией между процессорами и общей памятью. Другими словами, второй уровень иерархии системы управления ТП, как правило, представляет собой свободно связанную МПС, элементами которой являются специализированные процессоры и микроЭВМ.

Одной из основных проблем, возникающих при проектировании свободносвязанных МПС, является проблема их эффективной организации, заключающейся в определении оптимального числа и типов используемых специализированных процессоров и микроЭВМ, а также в распределении поступающих задач по процессорам.

В свободносвязанных МПС специализация может осуществляться либо аппаратным, либо программным путем. При специализации процессора на решение определенной задачи производится его проблемная ориентация на ускоренное выполнение определенных функций, что позволяет резко увеличить эффективную производительность решения данной задачи. В качестве примеров аппаратной специализации можно привести математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, в частности, матричных методов их выполнения, МП для обработки данных в различных областях применения (МП для реализации алгоритма БПФ и др.). Программная специализация достигается за счет дифференцированной загрузки локальной памяти процессоров программами решения определенных задач управления ТП.

Задача организации системы процессоров в свободносвязанной МПС заключается в следующем. Исходными данными для проектирования являются структура свободносвязанной МПС (рис. 8.13), множество $C = \{c_1, \dots, c_m\}$ задач обработки информации о состоянии ТП, а также множество $D = \{d_1, \dots, d_r\}$ типов специализированных про-

цессоров и микроЭВМ, параметры которых заранее известны. Пусть T_{ij} — время решения задачи $c_i \in C$ процессором $d_j \in D$ (в специализированных процессорах $d_k \in D$ для некоторых задач $c_j \in C$ возможно $T_{ik} = \infty$). В общем случае параметр T_{ij} может быть определен как м. о. времени решения задачи c_i процессором d_j при произвольном распределении времени решения задач с учетом м. о. времени, необходимого на обмен информацией между процессорами и локальной памятью. Предполагается, что процессор d_j решает задачу c_i без прерываний

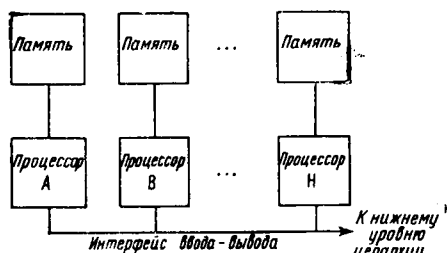


Рис. 8.13. Структура распределенной свободносвязанной МПС

каждого типа и такого отображения $C \rightarrow D_1$ (распределения задач по процессорам), чтобы синтезированная МПС характеризовалась наилучшим значением некоторого критерия оптимальности при выполнении всех ограничений на аппаратные параметры. При этом допускается комплексование МП как из однотипных, так и из разнотипных процессоров.

Сформулируем поставленную задачу в терминах математического программирования.

Поскольку класс рассматриваемых МПС предназначен для работы в составе аппаратуры оперативного управления технологическими процессами, в качестве критерия оптимальности для оценки эффективности работы МПС естественно выбрать критерий минимизации максимально возможного времени решения задач в МПС.

Обозначим через $d_j^{(q)}$ q -й процессор ($0 \leq q \leq \bar{y}_j$) типа $d_j \in D$, входящий в состав системы процессоров МПС, и введем переменные y_j — количество процессоров типа d_j , входящих в состав МПС, и

$$\omega_j^{(q)} = \begin{cases} 1, & \text{если задача } c_i \in C \text{ решается процессором } d_j^{(q)}, \\ 0 & \text{— в противном случае.} \end{cases}$$

Тогда формально задача выбора оптимального состава системы процессоров МПС и распределения задач по процессорам будет заключаться в минимизации целевой функции

$$F(\omega, T) = \max_{d_j^{(q)} \in D} \left(\sum_{c_i \in C_j^{(q)}} T_{ij} \omega_j^{(q)} \rightarrow \min \right) \quad (8.20)$$

при наличии системы ограничений на:

аппаратные параметры МПС (при условии, что они подчиняются свойству аддитивности)

$$\sum_{d_j \in D} \vartheta_{pj} y_j \leq V_p, \quad p = \overline{1, n}, \quad (8.21)$$

или

$$y_j = \left[\min_{\vartheta_{pj} \in \Theta_j} \left\{ \left(\frac{V_p}{\vartheta_{pj}} \right) \right\} \right], \quad j = \overline{1, r};$$

число решаемых задач (для специализированных процессоров)

$$\sum_{d_j \in D} n_j y_j \geq m; \quad (8.22)$$

корректность постановки задачи оптимизации

$$\begin{aligned} 1 \leq \sum_{d_i \in D} y_j \leq m; \quad \sum_{d_j^{(q)} \in D} w_{ij}^{(q)} = 1, \quad i = \overline{1, m}; \\ \sum_{c_i \in C} \sum_{d_j^{(q)} \in D} w_{ij}^{(q)} = m; \quad \sum_{c_i \in C} w_{ij}^{(q)} \leq n_j y_j, \quad j = \overline{1, r}; \end{aligned} \quad (8.23)$$

квазиоднородность системы процессоров МПС

$$\sum_{d_j \in D} \xi_j \leq L, \quad (8.24)$$

где

$$\xi_j = \begin{cases} 1, & \text{если } \sum_{c_i \in C} w_{ij}^{(q)} \geq 1, \\ 0, & \text{если } \sum_{c_i \in C} w_{ij}^{(q)} = 0 \end{cases}$$

(в случае построения однородных структур МПС $\sum_{d_j \in D} \xi_j = 1$);

необходимость решения всех поступающих задач (при наличии в структуре специализированных процессоров)

$$\sum_{c_i \in C} \sum_{d_j^{(k)} \in D} w_{ij}^{(q)} T_{ij} \leq K. \quad (8.25)$$

Здесь V_p — допустимое значение параметра ϑ_p ; L — максимально допустимое число типов процессоров в МПС; $K \gg T_{ij}$, $i = \overline{1, m}$; $j = \overline{1, r}$ — наперед заданное число; n_j — количество задач, которое может быть решено процессором d_j ; $C_j^{(q)}$ — подмножество задач, решаемых процессором $d_j^{(q)}$.

Решение задачи нелинейного целочисленного программирования (8.20) ... (8.25) представляет собой оптимальное решение задачи выбо-

ра числа и типов процессоров свободносвязанной МПС и распределения задач управления ТП по процессорам.

При наличии жестких временных ограничений на обработку отдельных параметров ТП в систему ограничений следует добавить дополнительные ограничения на допустимое время решения наиболее критичных задач.

§ 8.6. МИКРОПРОЦЕССОРНОЕ УПРАВЛЕНИЕ В РОБОТОТЕХНИКЕ

Робототехника — новое комплексное научно-техническое направление, которое включает создание и производство роботов, манипуляторов и автоматизированных технологических комплексов различного назначения. Робот состоит из механических рук — манипуляторов и устройства автоматического управления, в состав которого входят средства для обработки и хранения информации (мини- или микро-ЭВМ, микропроцессорные устройства и системы), а также чувствительные (сенсорные) устройства. Средства для обработки и хранения информации являются «мозгом» робота, а сенсорные устройства — «органами чувств». Функциональная схема робота представлена на рис. 8.14.

Одно из основных направлений развития роботов — совершенствование их систем управления на базе средств микропроцессорной техники. Аппаратурные и алгоритмические решения в организации систем управления определяют возможности роботов и их стоимость.

Методы и языки в программировании роботов. Существуют следующие методы программирования роботов: прямое обучение, роботориентированное программирование, задачно-ориентированное программирование.

Прямое обучение использовалось при программировании промышленных роботов первого поколения. Этот метод предполагает ручное перемещение робота во все требуемые положения и запись соответствующих им обобщенных координат сочленений. Выполнение программы заключается в перемещении сочленения робота в соответствии с заданной последовательностью положений. Роботы первого поколения работали по жестким программам подобно станкам с числовым программным управлением. Метод прямого обучения не требует использования ЭВМ. При этом невозможно использовать также датчики, что делает данный метод эффективным только в простых технологических процессах (окраска, точечная сварка и др.).

При *роботоориентированном программировании* производится опрос датчиков и в зависимости от обработки сенсорной информации определяется движение роботов. Преимущество этого метода заключается в возможности использования сенсорной информации от датчиков для коррекции движения роботов. Благодаря датчикам робот может функционировать в ситуации, когда данным во внешних объектах присуща некоторая степень неопределенности, что расширяет область применения роботов.

Описание программ функционирования роботов является робото-зависимым, поэтому языки программирования называют роботориентированными. К ним относятся языки MHL, WAVE, MINI, AL, VAL, AML, TEACH и ряд других.

Роботоориентированный язык можно разделить на две части: общую и специальную. Общая часть объединяет способы описания данных и операторы, аналогичные обычным языкам программирования. К ним относятся: описания данных и их типов; арифметические операции и выражения; оператор присваивания; операторы управления программой; процедуры и функции.

В специальную часть включены средства для управления роботом. Они обеспечивают программирование движения робота в декартовой и обобщенной системах координат, задание режимов движения и управление схватом, проверку состояния датчиков, моделирование рабочей обстановки. В качестве примера рассмотрим специальную часть языка управления роботом типа VAL.

Операторы управления роботом в языке VAL предназначены для непосредственного управления роботом. Их можно подразделить на следующие пять групп.

Операторы описания конфигурации робота предназначены для уточнения вида робота. С их помощью определяются «рука» робота (ПРАВАЯ, ЛЕВАЯ), положение «локтя» робота (ВЕРХНЕЕ, НИЖНЕЕ) и направление вращения сочленений робота (ПРЯМОЕ — по часовой стрелке, ОБРАТНОЕ).

Операторы управления схватом робота включают в свой состав шесть операторов для закрытия и открытия схвата робота на определенное расстояние. Оператор ОТКРЫТЬ {<число>} открывает схват робота на величину, задаваемую числом. Закрытие схвата на определенную величину осуществляется оператором ЗАКРЫТЬ {<число>}. Для мгновенного открытия и закрытия схвата робота на заданное расстояние в процессе его движения используются операторы: ОТНЕМЕДЛЕННО {<число>} и ЗАНЕМЕДЛЕННО {<число>}. Полное открытие схвата задается оператором РАЗЖАТЬ, для взятия объектов используется оператор СХВАТИТЬ.

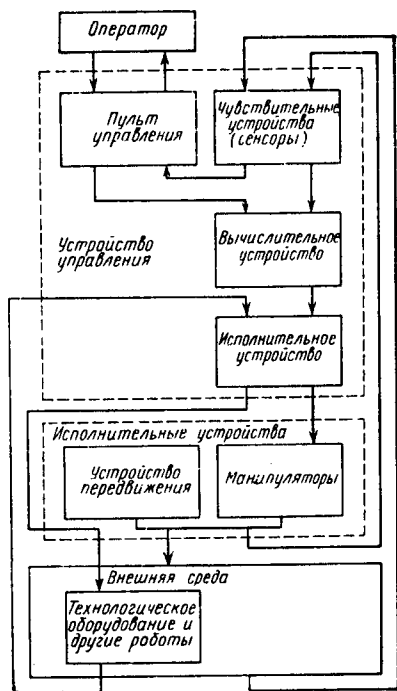


Рис. 8.14. Функциональная схема робота

Операторы задания режимов движения робота определяют скорость движения схвата и точность его расположения.

Операторы управления движением робота предназначены для перемещения схвата робота в заданную точку, координаты которой задаются как в обобщенной, так и в декартовой системах координат.

Вспомогательные операторы используются для задания некоторых дополнительных действий робота и выдачи оператору информации о состоянии робота.

При задачно-ориентированном программировании определяется не движение роботов, а желаемое расположение объектов. Данный метод позволяет абстрагироваться от особенностей конкретного робота, при этом пользователю не приходится определять траектории движения и положения, зависящие от геометрии и кинематики робота. Исходной информацией для задачно-ориентированного программирования является геометрическая модель рабочего пространства и робота. Задачно-ориентированное программирование вносит идеи и методы автоматизированного проектирования и искусственного интеллекта в робототехнику.

Концепции организации управления роботами. Современные системы управления роботами оснащаются средствами оцувствления, адаптации с использованием элементов искусственного интеллекта и автономных методов программирования. На системы управления роботами возложено большое число функций, в частности расчет, управление и контроль движений робота и технологического оборудования, обработка и опрос датчиков оцувствления и адаптации, синхронизация с работой оборудования производственного модуля, тестирование технических средств, связь с управлением верхнего уровня и т. п.

В настоящее время, несмотря на значительный прогресс в развитии элементной базы, снижение энергоемкости, повышение тактовой частоты микропроцессорных элементов, наблюдается потребность в дальнейшем повышении быстродействия вычислительных средств, встраиваемых в системы управления роботами. Поэтому наиболее эффективным способом повышения производительности является создание распределенных многопроцессорных систем управления роботами.

При управлении манипуляторами с простой кинематической схемой (декартовой или цилиндрической системой координат) и небольшим (до пяти) числом степеней подвижности управляющее устройство робота, как правило, строится на базе однопроцессорных микроЭВМ. Эти микроЭВМ выполняют арифметические операции над 16-разрядными словами с аппаратной реализацией умножения и деления, имеют развитые каналы ввода — вывода и систему обработки приоритетных прерываний. Широкий набор команд предусматривает различные режимы адресации. В таких микроЭВМ используются различные типы памяти: ПЗУ для хранения операционной системы, ОЗУ для отладочных работ, ОЗУ с сохранением информации при отключении питания для рабочих программ управления роботом.

При увеличении числа степеней подвижности, переходе к шарнирной структуре манипулятора, повышении точности и скорости движений для управления роботом требуется не одна, а несколько микроЭВМ. Повышение адаптивных возможностей промышленных роботов связано с необходимостью использования для управления информации от систем технического зрения, датчиков сил и моментов, дальномеров и других средств очувствления. Это также обуславливает необходимость применения многопроцессорных систем управления роботами.

В многопроцессорных системах управления центральный процессор, как правило, осуществляет общее управление системой, внешними связями, обеспечивает программирование и обучение робота. Кроме того, выделяется специальный процессор для вычислений различных тригонометрических и математических функций, преобразования координат и расчета траектории с заданными методами интерполяции. На нижнем уровне используются процессоры приводов, которые осуществляют управление движением каждого отдельного двигателя. На нижний уровень возлагаются также многопараметрические оптимизационные задачи динамического управления приводом, реализация адаптивного управления на исполнительном уровне, решение задач тестирования и контроля в реальном времени.

Для обработки сенсорной информации в систему управления включают один или несколько специализированных процессоров. Сенсорная система также имеет иерархическую структуру. На нижних уровнях она передает простейшие виды информации, на более высоких, где планируются действия робота, требуется более сложная информация описания отдельных объектов и внешней среды в целом. Так, например, система технического зрения должна передавать на нижний уровень лишь данные о расстоянии между объектами, а для верхнего уровня осуществлять идентификацию объекта, выделение поверхностей, ребер и т. п.

В общем случае для промышленных роботов выделяются интерфейсы: исполнительной системы робота, служащий для приема информации от датчиков обратной связи и выдачи управляющих сигналов на приводы отдельных степеней подвижности; «оператора», обеспечивающий взаимодействие человека с системой управления; ввода — вывода, предназначенный для связи с внешней производственной средой; средств адаптации и очувствления.

Структура промышленных систем управления роботами. В настоящее время в промышленности используются многопроцессорные системы управления роботами. На рис. 8.15 предстала обобщенная иерархическая структура управления промышленными роботами. В этой структуре взаимодействие ЭВМ реализуется по принципу «ведущий — ведомый». Ведущий процессор имеет высший приоритет и может приостановить работу ведомого в определенные моменты времени. Ведомые микроЭВМ реализуют задачи сбора и обработки информации, регулирования и управления степенями подвижности с заданными временными параметрами.

Центральный (ведущий) процессор выполняет задачи общего управления, связи, обработки данных, планирования движений, преобразования координат, интерполяции в декартовом пространстве, расшифровки команд, в то время как ведомые процессоры ответственны за выполнение задач управления движением отдельных степеней подвижности, учет динамических особенностей приводов, обработку

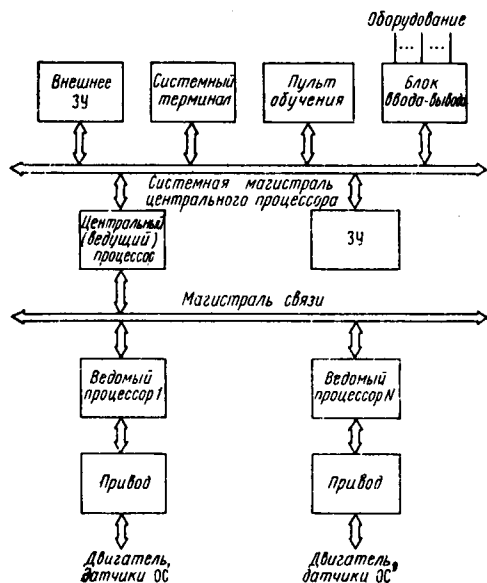


Рис. 8.15. Обобщенная иерархическая структура управления промышленными роботами

сигналов от датчиков обратной связи (ОС), реализацию режимов разгона — торможения и т. п.

Центральная системная магистраль служит для обмена данными между ведущим процессором и внешними функциональными узлами. Местные магистрали обеспечивают параллельную работу своих процессоров. Добавлением модулей в данную структуру можно обеспечить большую универсальность системы управления.

Иной подход к реализации многопроцессорной системы управления роботами иллюстрирует рис. 8.16. Система управления базируется на нескольких микроЭВМ, связанных общей магистралью. Связь дополнительных моду-

лей между собой обеспечивается через общую память. Ни одна из используемых микроЭВМ не связывается с другой непосредственно. Запись в любую ячейку памяти может производиться только одним вычислительным модулем, чтение данных из ячейки — всеми модулями. Работа такой системы управления производится в режиме разделения времени, т. е. каждому вычислительному модулю выделяется квант времени, в течение которого он читает определенный набор входных значений из соответствующих ячеек в общей памяти. В этом же кванте времени в память записываются полученные результаты. Если длительности кванта не хватает на выполнение указанных операций, их выполнение откладывается до следующего выделенного интервала времени. Подобные системы управления имеют простой протокол связи.

Передача данных через общую память означает, что добавление любого нового процессора требует лишь определения его области в общей памяти. Эта информация необходима только модулю, генерирующему данные (чтобы он «знал», куда должны быть записаны данные), и модулям, читающим данные (чтобы они «знали», где искать данные).

Никакие другие модули «не знают» и «не реагируют» на местонахождение новых данных. Таким образом, появляется возможность легко добавлять микроЭВМ, перемещать логические модули из одной микроЭВМ в другую, добавлять новые функции и вводить новые системы датчиков. Если магистраль имеет неиспользованные мощности, то конфигурация физической структуры системы может быть изменена, причем не потребуются изменения программного обеспечения в модулях, не участвующих непосредственно в изменении конфигурации.

Общая память всегда содержит легко получаемую карту текущего состояния системы, позволяет наблюдать процесс выполнения задачи.

Таким образом, при многопроцессорном управлении роботами можно выделить системы с *центральным управляющим процессором и с общей памятью*. Недостатком первого способа организации управления является то, что обмен информацией возможен лишь через центральный процессор или с его разрешения. При увеличении числа модулей и объемов информации обмена возникают очереди и конфликтные ситуации, ухудшающие характеристики режима реального времени. Недостаток второго способа — необходимость двойной передачи данных при обмене информацией между модулями через общую память. Однако при работе одного из процессоров в реальном времени такой способ передачи данных упрощает требования к синхронизации процессов другого процессора.

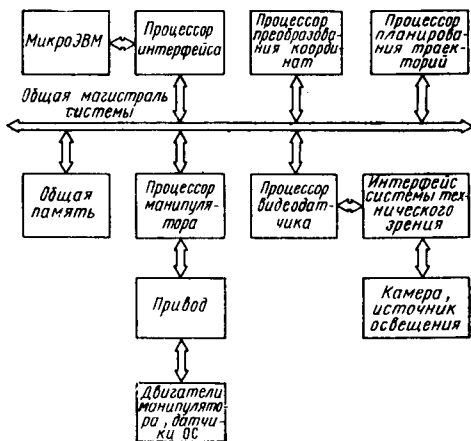


Рис. 8.16. Многопроцессорная система управления роботами с общей магистралью

Контрольные вопросы и задания

1. Как доказать адекватность математической модели реальному технологическому процессу?
2. Что характеризуют уравнения состояния динамического процесса?
3. Объясните физический смысл типовых законов управления технологическими процессами.
4. Как влияют технические характеристики микропроцессоров на реализацию законов управления технологическими процессами?
5. Каким образом с помощью текущего регрессивного анализа можно оценивать параметры технологического процесса?
6. Приведите примеры управления технологическим процессом на основе текущего регрессионного анализа.
7. Сформулируйте основные требования, предъявляемые к распределенным микропроцессорным системам управления технологическими процессами.

Глава 9

ПЕРСПЕКТИВЫ РАЗВИТИЯ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

§ 9.1. ТЕНДЕНЦИИ РАЗВИТИЯ АППАРАТНЫХ И ПРОГРАММНЫХ СРЕДСТВ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Развитие микропроцессорной техники привело к широкому внедрению цифровых вычислительных средств в системы автоматизации и управления, разработке большой номенклатуры разнообразных РЭС со встроенными микропроцессорами, появлению локальных вычислительных сетей и созданию качественно новой элементной базы. Разработаны мощные 16- и 32-разрядные микропроцессоры и разнообразные однокристальные микроЭВМ.

В настоящее время центр тяжести разработок сдвинулся в сферу создания СБИС (до 10^5 транзисторов на кристалле) и разработки программного обеспечения для ПЗУ и ППЗУ.

Следует отметить, что в области технологии производства СБИС микропроцессоров ведущие позиции завоевывает КМОП-технология. Разрабатываемые логические КМОП-вентили по рабочим характеристикам быстро догоняют лучшие образцы ИМС, изготовленных по ТТЛШ-технологии. По быстродействию и плотности упаковки КМОП БИС быстро догоняют n -канальные МОП-схемы. В области производства широкое распространение получили 16- и 32-разрядные МП и микроЭВМ. В области программного обеспечения укрепляются позиции языков высокого уровня, особенно языка АДА, а также осуществляется переход на микропрограммную реализацию программных средств.

Дальнейшее увеличение степени интеграции СБИС создало предпосылки для решения сложных задач обработки информации, которые условно можно разделить на два класса: задачи обработки с большим объемом вычислений и задачи обработки больших объемов нечисловой информации.

К задачам первого класса относятся распознавание сигналов и образов, синтез сигналов, опережающее реально-временное моделирование сложных систем. Наиболее распространенные области применения этих процессов: речевой ввод команд и информации в ЭВМ; машинное зрение роботов; помехозащищенная радиолокация и связь; ввод информации в ЭВМ в виде графиков и текстов, в том числе рукописных; синтез графических изображений для целей автоматизированного проектирования; различные виды диагностического оборудования; медицинская диагностика.

Задачи второго класса охватывают вопросы искусственного интеллекта, решаемые методами логической обработки больших объемов информации, представленной в символьной форме. Области их прило-

жения — диалоговый интерфейс «человек — машина», электронные каталоги и библиотеки, вычислительные экспертные системы, управление слабодетерминированными технологическими процессами производства изделий электронной техники, прогнозирование и распознавание образов.

Для решения этих задач универсальные микропроцессорные БИС классической архитектуры малоэффективны. Поэтому многообещающими являются тенденции создания проблемно-ориентированных СБИС, решающих задачи определенного класса. Из множества типов проблемно-ориентированных СБИС можно выделить СБИС для организации систем с параллельными вычислениями, сигнальные процессоры и логические процессоры для обработки нечисловой информации.

При разработке СБИС для *параллельной обработки* числовой информации прежде всего следует выделить классы решаемых задач. В частности, выявлен широкий класс задач, решение которых сводится к операциям вычисления произведений двух чисел и их последующего суммирования. К таким задачам относятся ДПФ, БПФ, вычисление свертки двух сигналов, цифровая фильтрация. Для аппаратной реализации параллельных алгоритмов решения подобных задач созданы параллельно-конвейерные структуры, реализующие разнообразные алгоритмы параллельной обработки информации.

Несмотря на то что параллельные системы могут решать любые вычислительные задачи, включая цифровую обработку аналоговых сигналов, к отдельной группе могут быть отнесены *цифровые сигнальные процессоры*. Можно выделить три направления развития аппаратурной архитектуры сигнальных процессоров.

Первое направление представляет собой развитие специализированных микропроцессоров с фиксированной структурой, выполняющих только одну специальную функцию, например БПФ входной последовательности данных. В структуре таких микропроцессоров необходимо предусматривать средства для организации последовательного или параллельного потока входных данных, а также интерпретации и хранения выходной последовательности. Главным элементом архитектуры специализированных МП является встроенный в кристалл умножитель.

Второе направление предполагает организацию процессоров по функциональному принципу. В этом случае сигнальный процессор может реализовать алгоритмы цифровой фильтрации, преобразования Фурье, усреднения, интегрирования, создания форматов выходных данных и т. д. Сигнальные процессоры подобной архитектуры строятся по модульному принципу. Модульный подход приводит к архитектуре, заключающейся в программируемом управлении набором аппаратно-программных функций, требуемых при обработке сигналов. Основной задачей организации управления в таких процессорах является генерирование потока команд и подачи их на различные модули, которые работают последовательно. Указанные сигнальные процессоры обеспечивают большую гибкость, чем устройства с жесткой структурой, но

одновременно требуют некоторой коммутации соединительных цепей и, возможно, изменений в памяти модулей, связанных с различными видами обработки информации.

Современное состояние технологии СБИС делает возможным организацию сигнальных процессоров описанных двух архитектур на одном кристалле, причем реализуема организация более сложных устройств в виде параллельного конвейера для многократного выполнения таких простых операций, как сложение, вычитание и комплексное умножение. Однако ограниченность числа выводов СБИС не позволяет полностью использовать ее возможности, при этом следует отметить, что скорости ввода и вывода информации в процессорах, реализованных по этим архитектурам, невелики.

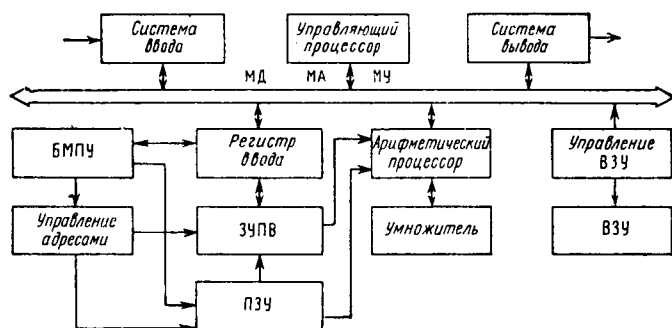


Рис. 9.1. Структура программируемого сигнального процессора

Третьим направлением является развитие архитектур *программируемых* сигнальных процессоров. Архитектурно они состоят из некоторого числа независимо управляемых элементов, каждый параллельного действия. Эти элементы включают в себя арифметическое устройство, устройство адресации и декодирования, канал управления вводом — выводом и управляющий процессор, функционирующий по программе, которая может храниться в ПЗУ или ППЗУ. Каждый элемент обладает локальной памятью, что позволяет избежать различного рода конфликтов. Структура типового программируемого сигнального процессора представлена на рис. 9.1. Основным элементом структуры является арифметический процессор, в котором с высокой точностью и большим быстродействием выполняются операции умножения-сложения в соответствии с алгоритмами обработки сигналов. Для высокоскоростного умножителя необходимо горизонтальное микропрограммное управление, что обеспечивает параллельное выполнение нескольких вычислительных операций. При решении многих прикладных задач обработки сигналов перспективной оказывается модульная организация, достигаемая за счет конвейеризации процесса вычислений, при которой определенное число этапов вычислений выполняется

последовательно для данного элемента данных, но все блоки работают параллельно. Это означает, что скорость обработки данных определяется временем обработки элемента данных только в одном блоке. Программируемые сигнальные процессоры могут быть реализованы в однокристальном исполнении.

Программное обеспечение программируемых сигнальных процессоров может состоять из двух уровней. На первом уровне программирование должно осуществляться на языке МП, предназначенном для разработчика системы и ориентированном на поток данных, на втором уровне — на языке обработки сигналов, предназначенном для системного программиста.

В группу микропроцессорных СБИС для обработки нечисловой информации входят ИМС, объединенные некоторыми общими принципами, такими, как ассоциативные методы обращения к памяти и широкое использование операции сравнения символьных строк, являющейся основной в задачах обработки нечисловой информации.

Особое внимание уделяется развитию архитектур персональных ЭВМ. Принцип открытой, магистральной архитектуры ПЭВМ, развитие которого по праву считается важным достижением разработчиков, в настоящее время вступает в конфронтацию с идеей моноблочного наполнения ПЭВМ. Происходит все более ускоряющаяся миниатюризация элементной базы и периферийных устройств. Использование КМОП-технологии, применение кристаллов ПЗУ с электрическим стиранием содержимого и прогресс в создании плоских индикаторных панелей позволяют создать моноблочные портативные ПЭВМ открытой архитектуры.

Возможности организации одновременного отображения различных аспектов обрабатываемой информации на нескольких дисплеях (полиэкранный диалог) эффективно могут быть использованы в любых профессиональных применениях ПЭВМ. Необходимость такой организации объясняется принципиально параллельной организацией профессиональной деятельности человека (работа с документом, поиск в архиве, табличные расчеты, построение чертежей и др.).

Перспективным направлением для существенного повышения производительности процессора ПЭВМ является развитие архитектуры с упрощенной системой команд — РИСК-архитектурой, описанной в § 5.5. Концептуальная простота этой архитектуры имеет важные следствия: применяются простые линейные модели адресации больших объемов памяти и расширяется номенклатура сопроцессоров (символьная и графическая обработка, арифметика).

Можно сделать заключение, что КМОП-технология с переходом к тактовой частоте 10 МГц и более, 32-разрядные магистрали с возможностью подключения сопроцессоров и архитектура с упрощенной системой команд являются перспективой развития ПЭВМ.

В связи с постоянным возрастанием удельного веса справочной информации, хранящейся в ПЭВМ, уже в ближайшие годы получат широкое применение оптические диски без возможности перезаписи дан-

ных. Формирование баз данных, тиражирование аудиовизуальных учебных пособий и т. п. становятся одним из важных направлений развития прикладной информатики и промышленного производства программных средств. Миниатюризованные накопители на винчестерских дисках по-прежнему сохраняют устойчивое положение благодаря уникальной комбинации надежности, плотности записи данных и высокого темпа обмена с основной памятью.

Адаптер канала связи с ПЭВМ должен предоставлять следующие виды информационного сервиса: прием широковещательных передач радио и телевидения, доступ к информационным сетям общего пользования, доступ к междугородной телефонной сети и др.

Для эффективного функционирования микроЭВМ в системах управления и контроля *программное обеспечение* микроЭВМ должно обладать такими свойствами: программируемой связью с объектом управления в рабочем, аварийном и исследовательских режимах; устойчивостью функционирования, достигаемой путем фильтрации и нейтрализации аномальных событий в системе; надежностью функционирования, обеспечиваемой контролем межмодульных интерфейсов при трансляции и контролем полноты отработки ситуаций.

§ 9.2. РАЗВИТИЕ СРЕДСТВ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ СБИС

Резкое повышение функциональной сложности СБИС вызвало необходимость разработки новых эффективных методов и средств их проектирования. В настоящее время для разработки топологии микросхем широко применяются частично автоматизированные методы проектирования с помощью графических интерактивных систем.

Увеличение степени интеграции СБИС делает акцент на разработку специализированных заказных и полузаказных БИС, выпускаемых малыми партиями. При этом стоимость и время разработок становятся неприемлемыми. Расширение номенклатуры микросхем с ростом их сложности и степени интеграции стимулирует создание систем автоматического проектирования СБИС. Термин «автоматическое проектирование» (automatic design) вместо «автоматизированное проектирование» (computer aided design) подразумевает процессы проектирования с резким ограничением использования интерактивных средств на большинстве стадий проектирования. Применение систем автоматического проектирования значительно уменьшает длительность цикла «проектирование — производство», резко сокращает стоимость проектирования и полностью устраняет ошибки в топологии.

Методология структурного автоматического проектирования основана на использовании *кремниевых компиляторов*. Кремниевый компилятор — пакет программ процесса автоматического синтеза сложных схемных конструкций и их моделей для анализа и верификации полученных технических решений, выполняемого на основе описания архитектуры микросхем на высоком уровне представления.

В основу организации кремниевых компиляторов положены следующие принципы:

- структурное иерархическое построение субсистем, узлов и элементов СБИС, а также комплекса программ проектирования;

- ориентация на ограниченный ряд типов архитектуры проектируемых СБИС и технологии производства;

- описание проектного задания на уровне функционирования СБИС;

- иерархическое описание топологии на языке высокого уровня;

- возможность параллельного и иерархического расширения;

- адаптируемость к конкретным конструктивно-технологическим ограничениям;

- широкое применение в проектируемых СБИС регулярных структур и субсистем.

Функции кремниевого компилятора и компилятора программ в ЭВМ в определенной степени аналогичны: компилятор в ЭВМ автоматически транслирует язык высокого уровня в машинный, кремниевый компилятор — описание функционирования СБИС на языке высокого уровня в язык низкого уровня, описывающий топологию. Основные структурные блоки кремниевого компилятора можно рассматривать как аналоги машинных конструкций в компиляторе программ, а саму кремниевую компиляцию — как процесс трансляции проектного описания в геометрическое.

Новое направление в автоматизированном проектировании, получившее название кремниевой компиляции и представляющее собой альтернативное решение проблемы рационального сочетания в САПР алгоритмических методов конструирования и опыта конструкторов-технологов, позволяет существенно упростить организацию процесса и сократить сроки создания специализированных СБИС.

ЗАКЛЮЧЕНИЕ

Интенсивное развитие технических средств микропроцессоров и микроЭВМ, а также системного и прикладного программного обеспечения приводит к быстрой смене поколений микропроцессорных РЭС. Как и в других отраслях науки и техники, знания, полученные студентами при изучении данного курса, требуют постоянного обновления.

Развитие микропроцессорной техники открывает новые и самые неожиданные применения МП и микроЭВМ в различных сферах человеческой деятельности, а радиоэлектронные средства представляют собой благодатную почву для их использования. Поэтому вполне вероятно появление в ближайшем времени качественно новых специалистов — радионинженеров, способных проектировать, конструировать и изготавливать разнообразные микропроцессорные радиоэлектронные средства.

СПИСОК ЛИТЕРАТУРЫ

1. Норенков И. П., Маничев В. Б. Системы автоматизированного проектирования электронной и вычислительной аппаратуры. — М.: Высшая школа, 1983. 272 с.
2. Пестряков В. Б., Кузенков В. Д. Радиотехнические системы, — М.: Радио и связь, 1985. 376 с.
3. Микропроцессоры. Архитектура и проектирование микроЭВМ. Организация вычислительных процессов /П. В. Нестеров, В. Ф. Шаньгин, В. Л. Горбунов и др.; Под ред. Л. Н. Преснухина. Кн.1. — М.: Высшая школа, 1986. 495 с.
4. Микропроцессоры. Средства сопряжения. Контролирующие и информационно-управляющие системы /В. Д. Вернер, Н. В. Воробьев, А. В. Горячев и др.; Под ред. Л. Н. Преснухина. Кн. 2. — М.: Высшая школа, 1986. 383 с.
5. Микропроцессоры. Средства отладки, лабораторный практикум и задачник /Н. В. Воробьев, В. Л. Горбунов, А. В. Горячев и др.; Под ред. Л. Н. Преснухина. — М.: Высшая школа, 1986. 351 с.
6. Григорьев В. Л. Программное обеспечение микропроцессорных систем. — М.: Энергоатомиздат, 1983. 208 с.
7. Григорьев В. Л. Программирование однокристалльных микропроцессоров. — М.: Энергоатомиздат, 1987. 288 с.
8. Проектирование цифровых систем на комплектах микропрограммируемых БИС /С. С. Булгаков, В. М. Мещеряков, В. В. Новоселов, Л. А. Шумилов; Под ред. В. Г. Колесников. — М.: Радио и связь, 1984. 240 с.
9. Мик Дж., Брик Дж. Проектирование микропроцессорных устройств с разрядно-модульной организацией: В 2 кн. — М.: Мир, 1984. 486 с.
10. Корячко В. П. Конструирование микропроцессорных систем контроля радиоэлектронной аппаратуры. — М.: Радио и связь, 1987. 160 с.
11. Мячев А. А., Иванов В. В. Интерфейсы вычислительных систем на базе мини- и микроЭВМ/Под ред. Б. Н. Наумова. — М.: Радио и связь, 1986. 248 с.
12. Гребенников Л. К., Летник Л. А. Программирование микропроцессорных систем на языке ПЛ/М. — М.: Финансы и статистика, 1986. 160 с.
13. Микропроцессоры: системы программирования и отладки /В. А. Мясников, М. Б. Игнатьев, А. А. Кочкин, Ю. Е. Шейнин; Под ред. В. А. Мясникова, М. Б. Игнатьева. — М.: Энергоатомиздат, 1985. 272 с.
14. Корячко В. П., Курейчик В. М., Норенков И. П. Теоретические основы САПР: — М.: Энергоатомиздат, 1987. 400 с.
15. Немчинов В. М. Применение микропроцессоров в измерительной аппаратуре для физического эксперимента. — М.: Изд. МИФИ, 1982, 70 с.
16. Мирский Г. Я. Микропроцессоры в измерительных приборах. — М.: Радио и связь, 1984. 160 с.
17. Литюк В. И. Автоматизированное проектирование цифровых фильтров на базе микропроцессорной техники. — Таганрог, Изд. ТРТИ, 1987, 80 с.
18. Клингман Э. Проектирование специализированных микропроцессорных систем. — М.: Мир, 1985, 360 с.
19. Микропроцессор K1815BF3 для цифровой обработки сигналов/ В. А. Бобков, А. И. Сухопаров, В. Ф. Гузик и др. Микропроцессорные средства и системы. 1986. № 2, с. 21—23.
20. Петровский А. А., Цырульников А. Н. Микропроцессорная реализация блока автоматики абонентской радиостанции. — Управляющие системы и машины, /1984, № 5, с. 31—36.
21. Кудрявцев Г. Г., Мамзелев И. А. Микропроцессорные средства в технике связи // Микропроцессорные средства и системы, 1984. № 3. с. 60—65.
22. Варламов И. В., Касаткин И. Л. Микропроцессоры в бытовой технике. — М.: Радио и связь, 1987. 80 с.
23. Пузырев В. А. Управление технологическими процессами производства микроэлектронных приборов. — М.: Радио и связь, 1984, 160 с.

24. Робототехника и гибкие автоматизированные производства: В 9 кн. Кн. 4. Вычислительная техника в робототехнических системах и гибких автоматизированных производствах/В. З. Рахманкулов, Ж. П. Ахромеев, В. В. Герасимов и др.; Под ред. И. М. Макарова. — М.: Высшая школа, 1986. 144 с.
25. Прангшвили И. В. Микропроцессоры и локальные сети микроЭВМ в распределенных системах управления. — М.: Энергоатомиздат, 1985. 72 с.
26. Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы. — М.: Радио и связь, 1981. 326 с.
27. Казаринов Ю. М., Номоконов В. Н., Филиппов Ф. В. Применение микропроцессоров и микроЭВМ в радиотехнических системах. — М.: Высшая школа, 1988. 207 с.
28. Лю Ю-Чжен, Гибсон Г. Микропроцессоры семейства 8086/8088. Архитектура, программирование и проектирование микрокомпьютерных систем. — М.: Радио и связь, 1987. 412 с.
29. Балашов Е. П., Григорьев В. Л., Петров Г. А. Микро- и мини-ЭВМ. — Л.: Энергоатомиздат, 1984. 376 с.
30. МикроЭВМ: В. 8 кн. Кн. 2. Персональные ЭВМ / В. С. Кокорин, А. А. Попов, А. А. Шишкевич; Под ред. Л. Н. Преснухина. — М.: Высшая школа, 1988. 159 с.
31. МикроЭВМ: В 8 кн. Кн. 5. Персонально-профессиональные ЭВМ / Г. П. Лопато, М. Е. Неменман, В. Я. Пыхтин, В. Н. Тикменов; Под ред. Л. Н. Преснухина. — М.: Высшая школа, 1988. 143 с.
32. Микропроцессорные системы автоматического управления / В. А. Бесекерский, Н. Б. Ефимов, С. И. Зиятдинов и др.; Под ред. В. А. Бесекерского. — Л.: Машиностроение, 1988. 365 с.
33. Микропроцессоры и микропроцессорные комплекты интегральных микросхем: Справочник. В 2 т./В. Б. Абрайтис, Н. Н. Аверьянов, А. И. Белоус и др.; Под ред. В. А. Шахнова. — М.: Радио и связь, 1988, Т. 1. 368 с.; т. 2, 368 с.
34. Хвощ С. Т., Варшинский Н. Н., Попов Е. А. Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник/Под ред. С. Т. Хвоща. — Л.: Машиностроение, 1987. 640 с.
35. Юэн Ч., Бичен К., Робинсон Дж. Микропроцессорные системы и их применение при обработке сигналов /Пер. с англ. — М.: Радио и связь, 1986, 296 с.

ОГЛАВЛЕНИЕ

Предисловие	3
Введение	5
Глава 1. Введение в проектирование микропроцессорных радиоэлектронных средств	6
§ 1.1. Тенденции развития РЭС и подходы к их проектированию	6
§ 1.2. Роль и место микропроцессоров в современных РЭС	9
§ 1.3. Формализация проектирования микропроцессорных РЭС	13
§ 1.4. Основные характеристики и классификация микропроцессоров	22
§ 1.5. Обоснование применения и выбор микропроцессорного комплекта для проектируемых РЭС	28
Глава 2. Архитектура микропроцессорных устройств	37
§ 2.1. Дуализм «программные средства — аппаратура» в микропроцессорной технике	37
§ 2.2. Типовые структуры микропроцессорных систем	39
§ 2.3. Интерфейс микропроцессоров	45
§ 2.4. Система команд микропроцессора	50
§ 2.5. Архитектура однокристалльного микропроцессора КМ1810ВМ86	62
§ 2.6. Организация управления микропроцессорами	74
§ 2.7. Архитектура секционного микропроцессора КМ1804ВС2	81
§ 2.8. Структура памяти МПУ	91
Глава 3. Организация программного обеспечения МПУ	99
§ 3.1. Структура и способы разработки программного обеспечения	99
§ 3.2. Язык ассемблера	109
§ 3.3. Структура данных	115
§ 3.4. Языки высокого уровня	120
§ 3.5. Принципы разработки рабочих программ для МПУ	143
§ 3.6. Микропрограммирование	148
§ 3.7. Метод формирования микропрограмм	154
§ 3.8. Средства разработки программного обеспечения МПУ	162
§ 3.9. Методы и средства автоматизации микропрограммирования	169

Глава 4. Отладка микропроцессорных устройств	179
§ 4.1. Необходимость отладки аппаратных и программных средств МПУ	179
§ 4.2. Методы тестирования МПУ	183
§ 4.3. Средства отладки аппаратуры МПУ	186
§ 4.4. Самотестирование	197
§ 4.5. Методы организации режима самотестирования	203
§ 4.6. Организация встроенной диагностики МПУ	209
§ 4.7. Комплексная отладка МПУ	213
 Глава 5. Организация микроЭВМ.	 220
§ 5.1. Архитектура микроЭВМ	220
§ 5.2. Особенности построения современных микроЭВМ	223
§ 5.3. Организация интерфейса в микроЭВМ	228
§ 5.4. Средства сопряжения микроЭВМ с объектами	234
§ 5.5. Персональные ЭВМ	246
§ 5.6. Микропроцессорные локальные вычислительные сети	256
 Глава 6. Микропроцессорные устройства в радиоэлектронных средствах	 263
§ 6.1. Обоснование применения МП в составе РЭС	263
§ 6.1. Определение нижней оценки быстродействия микропроцессора	264
§ 6.3. Оптимизация программ функционирования МПУ	270
§ 6.4. Оптимизация загрузки сверхоперативной памяти	275
§ 6.5. Микропроцессорные устройства и микроЭВМ в информационно-измерительной аппаратуре	287
§ 6.6. Микропроцессорные средства в технике связи	294
§ 6.7. Микропроцессоры в системах цифровой обработки сигналов	299
§ 6.8. Микропроцессорные устройства в бытовой радиоаппаратуре	314
 Глава 7. Микропроцессоры и микроЭВМ в системах контроля радиоэлектронных средств	 321
§ 7.1. Задачи автоматизированного контроля РЭС	321
§ 7.2. Организация внутрисистемного интерфейса в МПС	326
§ 7.3. Модель вычислительного ядра МПС	332
§ 7.4. Анализ конфликтных ситуаций в МПС	335
§ 7.5. Анализ производительности вычислительного ядра МПС	345
§ 7.6. Синтез структуры вычислительного ядра МПС	353
§ 7.7. Методика проектирования МПС повышенной жизнеспособности	357
§ 7.8. Задачи построения адаптивных систем контроля РЭС	361

Глава 8. Микропроцессоры и микроЭВМ в системах управления технологическими процессами радиоэлектронных средств	367
§ 8.1. Роль микроЭВМ и МПУ в системах управления технологическими процессами	367
§ 8.2. Реализация законов управления на микропроцессорной технике	375
§ 8.3. Микропроцессорное управление технологическими потоками	378
§ 8.4. Микропроцессорное управление технологическим процессом на основе текущего регрессионного анализа . . .	380
§ 8.5. Организация распределенных микропроцессорных систем управления технологическими процессами	384
§ 8.6. Микропроцессорное управление в робототехнике	390
 Глава 9. Перспективы развития микропроцессорной техники	 396
§ 9.1. Тенденции развития аппаратных и программных средств микропроцессорной техники	396
§ 9.2. Развитие средств автоматизации проектирования СБИС	400
Заключение	402
Список литературы	403

Учебное издание

Корячко Вячеслав Петрович

**МИКРОПРОЦЕССОРЫ И МИКРОЭВМ
В РАДИОЭЛЕКТРОННЫХ СРЕДСТВАХ**

Зав. редакцией В. И. Трефилов. Редактор Т. И. Артемова.
Оформление художника Ю. Д. Федичкина. Художественный редактор
Т. М. Скворцова. Младшие редакторы С. А. Пацева, И. С. Скирдова.
Технический редактор А. К. Нестерова. Корректор Р. К. Косинова

ИБ № 8558

Изд. № ЭР-513. Сдано в набор 06.03.90. Подп. в печать 11.09.90.
Формат 60×88^{1/8}. Бум. офс. № 2. Гарнитура литературная. Печать офсетная.
Объем 24,99 усл. печ. л. 24,99 усл. кр.-отг. 26,92 уч.-изд. л.
Тираж 30 000 экз. Зак. № 118. Цена 1 р. 20 к.

Издательство «Высшая школа», 101430, Москва, ГСП-4, Неглинная ул., д. 29/14.

Московская типография № 4 Государственного комитета СССР по печати.
129041, Москва, Б. Переяславская, 46.