МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение высшего профессионального образования
Северо-Западный государственный заочный технический университет

И. Г.Анкудинов

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

АРХИТЕКТУРА И ПРОЕКТИРОВАНИЕ

Учебное пособие

Санкт-Петербург
2003

Утверждено редакционно-издательским советом университета

УДК 681.31

Анкудинов И.Г. Микропроцессорные системы. Архитектура и проектирование: Учеб. пособие.– СПб.: СЗТУ, 2003. – 109 с.

Учебное пособие соответствует государственному образовательному стандарту высшего профессионального образования по направлению подготовки дипломированных специалистов 654600 – “Информатика и вычислительная техника” (Специальность 220100 – “Вычислительные машины, комплексы, системы и сети”).

В пособии даны классификация, краткая характеристика возможностей и применения микропроцессорных средств. Рассмотрены варианты построения архитектуры микропроцессорных (МПС) и мультимикропроцессорных систем, их основные конфигурации и области использования. Особое внимание уделено организации подсистем обработки, управления, памяти и ввода-вывода МПС.

Рассмотрены основные задачи проектирования МПС, включая вопросы организации и особенности проектирования МПС на основе однокристальных микроЭВМ и контроллеров. Приведены сведения о средствах разработки, тестирования и отладки МПС.

Пособие содержит также краткий обзор истории развития микропроцессорных средств, современного состояния и перспективных проектов МПС.

Учебное пособие предназначено для студентов пятого курса, изучающих дисциплину “Микропроцессорные системы”.

Рецензенты: В.В.Спиридонов, канд. техн. наук, доц. кафедры компьютерных технологий и программного обеспечения СЗТУ; Л.Я. Новосельцев, канд. техн. наук, доц. кафедры радиосистем СПб ГЭТУ (“ЛЭТИ”); В.П.Шеремет, канд. техн. наук, ст. научн. сотр. НПО «Аврора».

Предисловие

Изобретение микропроцессорной технологии в 70-е годы XX века и появление персональных компьютеров сыграло революционную роль в развитии информационной индустрии. На микропроцессорах и интегральных схемах создаются компьютеры, компьютерные сети и системы передачи данных. Микропроцессоры и микроЭВМ находят применение практически в любой сфере человеческой деятельности.

В первой главе настоящего учебного пособия даны основные определения, классификация и области применения микропроцессорных систем в персональных компьютерах, рабочих станциях, серверах и других устройствах. В этой же главе приведены требования, предъявляемые к современным микропроцессорным системами и история их развития.

Глава 2 посвящена архитектуре микропроцессорных систем, включая многомашинные и многопроцессорные структуры. В главе 3 рассмотрены основы организации ввода-вывода, прерываний и прямого доступа к памяти.

Главы 4 и 5 посвящены проектированию микропроцессорных систем, причем особое внимание уделено тестированию и отладке.

В заключении обрисованы перспективы развития микропроцессорных технологий.

Глава 1
ВИДЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ, ОСНОВНЫЕ ТРЕБОВАНИЯ И ИСТОРИЯ РАЗВИТИЯ

1.1. Основные определения

Микропроцессорная система (МПС) это микроЭВМ (микрокомпьютер – microcomputer) или вычислительный комплекс (ВК), построенный на основе микропроцессорного комплекта (МПК) больших (БИС) и/или сверхбольших (СБИС) интегральных микросхем [2, 4, 5, 6, 9, 10]. В состав МПК могут входить микропроцессорные и другие интегральные микросхемы различных схемотехнических типов, если они совместимы по архитектуре, электрическим параметрам и конструктивному исполнению.

МПС различаются областями применения, архитектурой и конструктивным исполнением.

Архитектуру МПС можно описать тремя составляющими:

- состав, характеристики и структурная организация (взаимосвязь) устройств МПС;
- принцип функционирования;
- набор машинных команд, или инструкций (машинный язык).

Современные МПС реализуют архитектуру, которая воплощает, как правило, следующие принципы:

1. Принцип хранимой в памяти программы.
2. Принцип адресного обращения устройств МПС друг к другу.
3. Принцип магистрально-модульной структуры.

Важной характеристикой МПС является число центральных процессоров. По этому признаку можно выделить следующие виды МПС:

- однопроцессорные системы;
- . мультипроцессорные системы;
- многомашинные системы (вычислительные комплексы).

Можно выделить два основных вида однопроцессорных систем:

- МикроЭВМ, ориентированные на более или менее четко определенные области применения, класс задач и/или пользователей.
- Специализированные микроЭВМ, или *микроконтроллеры (*МК).

Специализированные микроЭВМ предназначены для решения одной или ограниченного числа задач с максимально возможной эффективностью. В качестве примера назовем алгоритмы быстрого преобразования Фурье [12], широко используемые в задачах распознавания образов, сжатия цифровых данных и др.

Существует большое число видов микроЭВМ, ориентированных на те или иные функции в сетевых информационно-вычислительных системах: персональные микроЭВМ и рабочие станции; серверы, маршрутизаторы и коммутаторы.

В подкласс персональных микроЭВМ и рабочих станций можно включить также X-терминалы, представляющие из себя комбинацию бездисковой рабочей станции и стандартного ASCII-терминала и предназначенные для работы в многооконной системе.

Если персональные компьютеры ориентированы на пользователя, то *микроконтроллеры* – на объект управления. Устройствами ввода в микроконтроллеры являются преобразователи информации, а именно датчики, установленные на объекте управления. Датчики преобразуют неэлектрические величины в электрические сигналы. В состав микроконтроллеров обычно входят преобразователи аналоговых сигналов в цифровой код – аналого-цифровые преобразователи (АЦП).

Устройствами вывода микроконтроллеров являются исполнительные механизмы объектов, как правило, это – электронная система управления электрическими проводами. Для сопряжения выхода МК с системой привода в состав микроконтроллеров обычно входят также преобразователи цифрового кода в аналоговые сигналы – цифро-аналоговые преобразователи (ЦАП).

Микроконтроллеры работают, как правило, в реальном масштабе времени и выполняют ограниченный набор программ, повторяющихся во времени. В отличие от ПК микроконтроллеры не требуют больших вычислительных ресурсов (памяти команд и данных), причем алгоритмы преобразования в программах МК просты и сводятся к арифметическим и логическим операциям. Каждая команда микроконтроллера, как правило, это – программа, написанная на языке команд МП.

Простейший микроконтроллер состоит из одной БИС, на которой размещаются все устройства цифровой ЭВМ и преобразователи информации. Более сложные многоплатные конструкции микроконтроллеров, как правило, содержат типовой МП.

Мультипроцессорные системы характеризуются тем, что каждый процессор относительно независимо выполняет свою программу, причем

общая ОС оперативно распределяет нагрузку между процессорами. Взаимодействие процессоров осуществляется через общую ОП. ОС распределяет общие периферийные устройства между процессами. Таким образом, имеется возможность распараллеливать вычислительный процесс, а при отказе какого-либо процессора оперативно перераспределять работу между оставшимися процессорами. Благодаря такой организации достигается высокая производительность и отказоустойчивость. В качестве примера мультипроцессорных систем назовем транспьютеры.

Транспьютеры – это микропроцессоры, предназначенные для работы в мультипроцессорных системах с однотипными процессорами Особенностью транспьютеров является аппаратная поддержка вычислительных процессов и наличие быстрых коммуникационных каналов, каждый из которых способен одновременно передавать по одной магистрали данные в процессор, а по другой – данные из него. В составе команд транспьютера имеются команды управления процессами и команды поддержки инструкций языков высокого уровня. Системы программирования транспьютеров включают трансляторы с языков высокого уровня Паскаль, Си, Фортран.

Многомашинная МПС – это вычислительный комплекс, включающий несколько микроЭВМ, каждая из которых работает под управлением собственной ОС, а также имеющий в своем составе аппаратные и программные средства для организации взаимосвязи микроЭВМ. Системное ПО предоставляет прозрачный доступ программам и пользователям к ресурсам всех микроЭВМ в составе комплекса. Однако, поскольку взаимосвязь осуществляется через внешние устройства, эффективность взаимодействия микроЭВМ в составе комплекса ниже, чем в мультипроцессорных системах.

1.2. Персональные компьютеры и рабочие станции

Персональные компьютеры (ПК) и рабочие станции (настольные системы высокой производительности, ориентированные на профессиональных пользователей) появились в результате эволюции миникомпьютеров при переходе от элементной базы малой и средней степени интеграции на БИС и СБИС.

Персональные микроЭВМ

Персональная микроЭВМ (персональный компьютер – ПК) – это микроЭВМ, имеющая дружественный интерфейс и ориентированная на непрофессионала в области программирования и электронно-вычислительной техники. Современные персональные компьютеры имеют малое время ввода в эксплуатацию, достаточные вычислительные ресурсы (производительность памяти, трехуровневое ПО, набор периферийных устройств, включая мультимедиа), хорошие эксплуатационные параметры и развитую компьютерную инфраструктуру.

Низкая стоимость ПК, первоначально создававшихся на платформе Intel, программное обеспечение, ориентированное на конечного пользователя, "дружественный" интерфейс и проблемно-ориентированные средства для автоматизации разработки прикладных программ быстро завоевали прочные позиции на компьютерном рынке. С появлением новых поколений МПК Intel средний пользователь ПК получил возможность запускать одновременно несколько приложений таких, как текстовый процессор, электронные таблицы, базы данных и мультимедиа-приложения.

Рабочие станции

Рабочие станции, создававшиеся на платформе UNIX, были ориентированы на профессионалов в области научной, технической и инженерной деятельности. Их появление было связано с созданием 32-разрядных RISC-процессоров и микросхем памяти емкостью более 1 Мбит. Отличительные свойства рабочих станций – это сочетание высокого быстродействия с большим объемом оперативной и внешней памяти, высокопроизводительные внутренние магистрали, быстродействующие высококачественные графические подсистемы и разнообразные устройства ввода/вывода.

Развитие ПК и рабочих станций привело к сближению параметров и областей применения этих двух направлений, в результате чего появилось понятие "персональная рабочая станция".

С одной стороны, быстрый рост производительности ПК на базе новейших микропроцессоров Intel, развитие технологии локальных шин (VESA и PCI), устраняющей "узкие места" в архитектуре, а также резкое снижение

цен, сделали ПК весьма привлекательной альтернативой рабочим станциям, а также позволили использовать их в отдельных случаях в качестве серверов.

С другой стороны, производители рабочих станций на базе RISC-микропроцессоров стали создавать системы "начального уровня", превосходящие самые высокопроизводительные ПК по производительности и возможностям наращивания, но близкие к ним по стоимости. Такие поставщики UNIX-систем, как компании Sun Microsystems и Hewlett Packard, стали конкурировать с поставщиками ПК в области офисных приложений для коммерции и бизнеса. Чтобы не отстать, компания Intel, развивая архитектуру CISC, ускорила разработку семейства высокопроизводительных процессоров 486 и Pentium, в которых использовала локальную шину и ряд других технологических усовершенствований, а также многие подходы, применявшиеся ранее только в RISC-процессорах.

Одновременно происходил процесс разукрупнения (downsizing), заключающийся в том, что информационные системы на основе миникомпьютеров и рабочих станций стали вытесняться распределенными системами, в которых рабочие станции и, в меньшей степени, ПК могут использоваться не только как настольные системы, но и в качестве серверов.

X-терминалы

X-терминал – это комбинация бездисковой рабочей станции и стандартного ASCII-терминала. Они позволяют работать в многооконной системе и обеспечивают графические возможности. "Подчиненные" X-терминалы могут использовать мощную графическую рабочую станцию в качестве локального сервера.

X-терминалы отличаются от ПК и рабочих станций тем, что локальная вычислительная мощность X-терминала обычно используется только для обработки отображения результатов работы приложения, причем само приложение (называемое клиентом) выполняется удаленно на главном компьютере (сервере).

1.3. Серверы

МПС используются в качестве серверов в распределенной сетевой модели "клиент-сервер", в которой часть работы выполняет сервер, а часть – пользовательский компьютер, причем в общем случае клиентская и пользовательская части могут работать и на одном компьютере.

МПС, используемые в качестве серверов, можно классифицировать по виду ресурсов, которыми они владеют:

- файл-серверы (файловая система),
- серверы баз данных (база данных),
- принт-серверы (принтеры),
- вычислительные серверы и серверы приложений (процессоры или пакеты прикладных программ).

Требования к составу оборудования и программного обеспечения сервера, к его надежности и производительности сильно варьируются в зависимости от числа пользователей и характера решаемых ими задач. В качестве файл-серверов небольших рабочих групп (два-три десятка пользователей) можно использовать ПК и программное обеспечение Novell.

Для файл-сервера общего доступа, с которым одновременно работают более сотни, а то и несколько сотен человек, простой однопроцессорной платформы и программного обеспечения Novell уже недостаточно. То же самое можно сказать и о серверах баз данных крупных информационных систем, на которые ложится основная нагрузка по обработке запросов.

В этом случае используются суперсерверы – мощные многопроцессорные серверы с высокой общесистемной производительностью и отказоустойчивостью, с возможностями наращивания оперативной памяти до нескольких гигабайт, дискового пространства до сотен гигабайт, быстрыми интерфейсами дискового обмена (типа Fast SCSI-2, Fast&Wide SCSI-2 и Fiber Channel) и несколькими сетевыми интерфейсами.

В таких суперсерверах используют запатентованную высокоскоростную системную шину, которая связывает между собой несколько центральных процессоров RISC, либо Pentium и оперативную память, а также многоуровневую шинную архитектуру, включая множество стандартных шин ввода/вывода, размещенных в одном корпусе. Система суперсервера поддерживает режим симметричной многопроцессорной обработки и позволяет распределять задания по нескольким центральным процессорам, а также режим асимметричной многопроцессорной обработки, позволяющий выделять процессоры для выполнения конкретных задач.

Суперсерверы используют сетевые протоколы TCP/IP и NFS, операционную систему UNIX или Windows NT, которые обеспечивают многопотоковую многопроцессорную и многозадачную обработку. Они обеспечивают эффективное системное администрирование, возможности наращивания дискового пространства и вычислительной мощности, а также

средства обеспечения надежности хранения данных и защиты от несанкционированного доступа.

1.4. Мейнфреймы и кластерные архитектуры

К современным МПС, ориентированным на применение в корпоративных информационных системах, предъявляются высокие требования в отношении следующих показателей: отношение стоимость/производительность; надежность и отказоустойчивость; масштабируемость; совместимость и мобильность программного обеспечения.

Мейнфреймы, или большие универсальные ЭВМ, – это мощные и сверхмощные вычислительные системы общего назначения, обеспечивающие непрерывный круглосуточный режим эксплуатации.

Система IBM/360, выпущенная в 1964 году компанией IBM, – эталон архитектуры мейнфрейма. Машины ряда ЕС ЭВМ, которые в течение многих лет выпускались в СССР, воспроизводили именно эту архитектуру.

С точки зрения архитектуры мейнфрейм – это многопроцессорная система, содержащая один или несколько центральных и периферийных процессоров с общей памятью, связанных между собой высокоскоростными магистралями передачи данных.

При этом основная вычислительная нагрузка ложится на центральные процессоры, каждый из которых может иметь векторный сопроцессор для достижения суперкомпьютерной производительности за счет ускорения операций. Периферийные процессоры (по терминологии IBM – селекторные, блок-мультиплексные, мультиплексные каналы и процессоры телеобработки) обеспечивают работу мейнфрейма с широкой номенклатурой периферийных устройств.

Если первые мейнфреймы устанавливались в специально оборудованных помещениях, имели большие габариты, не могли работать без двухконтурной водяной системы охлаждения и кондиционирования, то с развитием элементно-конструкторской базы появились мейнфреймы, построенные по блочно-модульному принципу, для которых достаточно принудительной воздушной вентиляции и не требуются специальные помещения и кондиционеры.

По мере того, как принципы открытости и совместимости компьютерных и телекоммуникационных систем завоевывали всеобщее признание, поставщики мейнфреймов также стали расширять возможности своих систем в этом направлении, хотя первоначально мейнфреймы ориентировались на

патентованные операционные системы, централизованные вычисления и имели ограниченные возможности для объединения в единую систему оборудования различных фирм-поставщиков.

Главный недостаток мейнфреймов – относительно низкое соотношение производительность/стоимость. Тенденция "разукрупнения" (downsizing) вычислительных систем заключается в замене мейнфреймов на компьютеры менее дорогих классов: миникомпьютеры и многопроцессорные системы, включая распределенные системы – компьютерные сети. Определенные сложности надежной реализации распределенной архитектуры клиент-сервер являются причиной того, что для наиболее ответственных приложений продолжают использовать мейнфреймы.

Для критически важных приложений таких, как управление базами данных, обработка транзакций и телекоммуникации, применяются кластерные вычислительные системы высокой производительности, обеспечивающие продолжительное безотказное функционирование (High Availability Systems – системы высокой готовности). Кластерная система – это объединение машин, которое воспринимается операционной системой, системным программным обеспечением, прикладными программами и пользователями как единое целое. Концепция кластерных систем (VAX-кластеры) была предложена компанией DEC, позднее появились UNIX-кластеры.

Кластерные системы имеют параллельную масштабируемую архитектуру и поэтому обеспечивают заданный уровень производительности. Продолжительное безотказное функционирование системы зависит от трех факторов: надежности, готовности и удобства обслуживания. Кластерные системы обеспечивают достаточно высокую готовность при относительно низких затратах.

Повышение надежности кластерных систем достигается общими для любой аппаратуры методами и средствами, а именно путем снижения интенсивности отказов и сбоев за счет применения БИС и СБИС, снижения уровня помех, облегченных режимов работы совершенствования методов сборки аппаратуры.

Чтобы обеспечить высокую готовность любой системы, необходимо минимизировать время плановых и неплановых простоев системы. Для повышения готовности важную роль играют такие *эксплуатационные характеристики* системы, как удобство ее обслуживания, ремонтопригодность и контролепригодность.

Кластеризованные системы способны обеспечить *малое время неплановых простоев и высокую готовность* за счет того, что они имеют средства контроля, аппаратурную и программную избыточность, на основе которых реализуются различные варианты отказоустойчивых архитектур, осуществляющих коррекцию ошибок и автоматическое восстановление вычислительного процесса после появления неисправности. При отказе одного процессора они быстро перераспределяют работу на другие процессоры внутри кластера.

Применение кластеризации повышает эффективность систем параллельных баз данных. Высокая производительность обеспечивается благодаря тому, что кластеризация позволяет распределить задания по множеству процессорных ресурсов и тем самым распараллелить их выполнение. Этим достигается более высокая скорость обработки транзакций, поддержка большего числа одновременно работающих пользователей и ускорение выполнения сложных запросов. Производительность системы легко масштабируется в любое время за счет добавления процессоров, объемов оперативной и дисковой памяти, а также новых узлов.

Высокая готовность обеспечивается за счет того, что в случае отказа одного из узлов кластеризованной системы, оставшиеся работоспособные узлы могут взять на себя задания, выполнявшиеся на отказавшем узле, не останавливая общий процесс работы с базой данных. Это возможно благодаря тому, что в каждом узле системы имеется логический образ базы данных.

Рассмотрим три типа архитектур, поддерживающих параллельные базы данных.

Наиболее типичный случай – это *архитектура с общими (разделяемыми) дисками (Shared Disk Architecture),* поддерживает единую базу данных при работе с несколькими компьютерами, образующими узел кластерной системы. Каждый узел работает под управлением своей копии операционной системы. Все узлы разделяют доступ к общим дискам, на которых располагается единая база данных. Для повышения производительности таких систем можно увеличить число процессоров, устройств оперативной и внешней памяти, либо увеличить число узлов.

Архитектура без разделения ресурсов (Shared Nothing Architecture) так же, как и архитектура с общими дисками, поддерживает единый образ базы данных при работе с несколькими компьютерами, образующими узел и работающими под управлением своей копии операционной системы. Отличие таких систем заключается в том, что разделяется только общий коммуникационный канал между узлами системы, причем каждый узел имеет

собственную оперативную память и собственные диски, которые не разделяются между отдельными узлами системы. Для повышения производительности таких систем достаточно увеличить число процессоров, объемы оперативной и внешней памяти в каждом узле, либо число узлов.

Симметричная многопроцессорная архитектура с общей памятью (Shared Memory SMP Architecture) поддерживает единую базу данных, работающую на многопроцессорном сервере под управлением одной операционной системы. Для повышения производительности таких систем достаточно увеличить число процессоров, устройств оперативной и внешней памяти.

1.5. Требования, предъявляемые к современным микропроцессорным системам

Основные критерии оценки МПС:

- Отношение стоимость/производительность.
- Надежность и отказоустойчивость.
- Масштабируемость.
- Совместимость и мобильность программного обеспечения.

Отношение стоимость/производительность. По этому показателю можно выделить два крайних полюса:

- категория больших универсальных ЭВМ (мейнфреймов) и супер-компьютеров, которые стоят очень дорого;
- категория ЭВМ массового применения, к которой в первую очередь следует отнести персональные компьютеры, для которых характерной тенденцией является минимизация отношения стоимость/производительность.

Для достижения высокой производительности и надежности больших ЭВМ приходится игнорировать стоимостные характеристики. К этой категории относятся, например, суперкомпьютеры фирмы Cray Research и высокопроизводительные мейнфреймы компании IBM.

Между этими двумя крайними полюсами находится класс миникомпьютеров и рабочих станций, для которых характерной тенденцией является поиск компромисса между стоимостью и производительностью.

Мера производительности МПС. Для сравнения различных МПС между собой используются стандартные методики измерения производительности. Следует отличать производительность микропроцессоров от производительности МПС, хотя производительность МПС существенно

зависит от производительности микропроцессоров. Для измерения производительности вычислительных систем применяют тестовые программы (benchmark).

До усложнения архитектуры микропроцессоров (RISC-ядро, встроенная кэш-память, технология внутреннего умножения тактовой частоты) основной мерой производительности компьютеров и, в частности, микропроцессоров считалась их тактовая частота. Этот параметр микропроцессоров остаётся важным показателем их производительности, но уже не является определяющим.

В 1992 году фирма Intel предложила индекс iCOMP (Intel Comparative Microprocessor Performance) для относительной оценки производительности микропроцессоров. Индекс iCOMP отражает относительную производительность данного устройства по сравнению с процессором 486SX-25, производительность которого принимается за 100. При вычислении индекса iCOMP предполагается следующий процентный состав операций:

16-разрядные целые числа – 67%,

16-разрядные действительные числа – 3%,

32-разрядные целые числа – 25%),

32-разрядные числа с плавающей запятой –5%.

В специализированных МПС при оценке производительности следует исходить из процентного состава операций в реальных задачах.

Надежность и отказоустойчивость МПС. *Надежность* – это важнейшая характеристика МПС. Понятие надежности включает не только аппаратные средства, но и программное обеспечение. Главной целью повышения надежности МПС является целостность хранимых и обрабатываемых в них данных. Искажение данных может произойти в результате отказов и сбоев.

Для повышения надежности необходимо использовать в МПС электронные схемы с малой интенсивностью отказов (в частности компоненты с высокой и сверхвысокой степенью интеграции), соблюдать правильный тепловой режим их работы, а также снижать уровень помех и совершенствовать методы сборки аппаратуры.

Под о*тказоустойчивостью* понимают свойство МПС, основанное на избыточном аппаратном и программном обеспечении, которое обеспечивает ей возможность продолжения действий, заданных программой, после возникновения отказов или сбоев. Наиболее распространенный способ введения избыточности аппаратного обеспечения – это параллельные, многопроцессорные и многомашинные МПС. Концепции распараллеливания

вычислений и отказоустойчивости МПС естественным образом связаны между собой, поскольку имеющиеся ресурсы избыточности в параллельных МПС могут гибко использоваться как для повышения производительности, так и для повышения надежности. Для продолжения работы после возникновения отказа структура параллельной МПС должна быть приспособлена к автоматической реконфигурации.

Масштабируемость МПС. Масштабируемость означает возможность наращивания таких ресурсов МПС, как числа и мощности процессоров, объемов оперативной и внешней памяти и других. Масштабируемость МПС достигается за счет архитектуры, конструкторских решений, а также средств программного обеспечения. Действительно масштабируемая вычислительная система должна быть сбалансирована по всем параметрам.

Добавление единицы каждого ресурса в действительно масштабируемой сбалансированной системе должно давать прогнозируемое увеличение производительности и пропускной способности МПС при приемлемых затратах. Например, добавление *n* процессоров к системе в идеальном случае должно приводить к линейному росту производительности. Однако в этом случае возрастает трафик между процессорами и основной памятью, а также между памятью и устройствами ввода/вывода. Программное обеспечение МПС должно минимизировать трафик межпроцессорного обмена, который может препятствовать линейному росту производительности системы. При недостаточной пропускной способности шин могут возникать потери производительности. Реальное увеличение производительности существенно зависит от динамики поведения прикладных задач.

Совместимость и мобильность программного обеспечения. Характерная тенденция рынка информационных технологий состоит в том, что поставщики компьютерного оборудования ориентируются прежде всего на прикладное программное обеспечение, интересующее конечного пользователя, поскольку именно программное обеспечение позволяет решать его задачи, а не выбор той или иной аппаратной платформы.

Программная совместимость моделей вычислительных систем означает использование такой архитектуры, которая с точки зрения пользователей была бы одинаковой для всех моделей, независимо от года выпуска, цены и производительности. Программная совместимость позволяет сохранить наработанное программное обеспечение пользователя при переходе на новые, более производительные модели. Особенно ярко эта тенденция проявляется в развитии компьютерных сетей от однородных сетей программно совместимых

компьютеров к неоднородным сетям и мощным распределенным вычислительным системам. При этом неоднородная сеть не только выполняет функции обмена информацией, но и становится средством интеграции ресурсов отдельных серверов и рабочих станций таким образом, что каждый элемент лучше всего соответствует требованиям конкретной прикладной задачи.

В условиях жесткой конкуренции на рынке аппаратных платформ и программного обеспечения сложилась *концепция открытых систем*, включающая набор стандартов на различные компоненты вычислительной среды. Современные открытые неоднородные распределенные вычислительные системы должны обеспечивать следующие требования:

- возможность гибкого изменения состава аппаратного и программного обеспечения в соответствии с меняющимися требованиями решаемых задач;
- мобильность программного обеспечения, т.е. возможность запуска одних и тех же программных систем на различных аппаратных платформах;
- возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть.

Показательным примером воплощения концепции открытых систем является модель OSE (Open System Environment), предложенная комитетом IEEE POSIX, и рекомендуемые для федеральных учреждений США спецификации в области информационных технологий "Application Portability Profile (APP). The U.S. Government's Open System Environment Profile OSE/1 Version 2.0", выпущенные Национальным институтjv стандартов и технологии США. Этот документ определяет требования к мобильности системного и прикладного программного обеспечения.

1.6. История развития микропроцессорных средств

Разработкой и производством микропроцессоров занимаются многие фирмы, например Intel, AMD, Cyrix, IBM, DEC, Motorola, Texas Instruments и другие. Особая роль в появлении и развитии микропроцессорной техники принадлежит фирме Intel. Поэтому рассмотрим историю развития микропроцессорных средств на примере фирмы Intel.

Семейство процессоров x88/x86. Первый микропроцессор - i4004 - был изготовлен в 1971 году фирмой Intel (INTegrated Electronics). Максимальная тактовая частота i4004 составляла всего 750 кГц, тем не менее с тех пор он

прочно удерживает лидирующее положение на данном сегменте рынка. Этот четырехразрядный микропроцессор предназначался для калькуляторов и, хотя умещался на одном кристалле (чипе) размером не более 1/6 на 1/8 дюйма, оказался более мощным, чем лучший в мире компьютер того времени - ENIAC - компьютер американского правительства: i4004 обрабатывал 60000 инструкций в секунду, в сравнении с 5000 инструкций ENIAC, который занимал площадь в 3000 квадратных футов и весил 30 тонн.

Микропроцессор i4004 и ряд следующих проектов фирмы Intel по созданию однокристальных микропроцессоров открыли эру персональных компьютеров и путь к созданию искусственных интеллектуальных систем. В 1972 году компания Intel выпустила свой очередной микропроцессор – i8008, мощность которого возросла вдвое по сравнению с его предшественником. Дон Ланкастер (Don Lancaster) применил процессор i8008 в разработке терминала ввода-вывода прототипа персонального компьютера.

Подлинный успех корпорации Intel принес в 1974 году 8-разрядный микропроцессор i8080, выполненный по n-канальной МОП-технологии, с тактовой частотой не выше 2 МГц. Между прочим, Бил Гейтс написал свой первый интерпретатор Бейсика для компьютера «Альтаир», построенного на основе i8080. Классическая архитектура i8080 оказала огромное влияние на дальнейшее развитие однокристальных микропроцессоров. В нашей стране его аналог - микропроцессор КР580ИК80. С микропроцессором 8080 также связано появление стека внешней памяти, что позволило использовать программы любой вложенности.

Несмотря на заслуженный успех i8080, в 1978 году фирма Intel выпустила первый 16-битный микропроцессор i8086, который стал настоящим промышленным стандартом для персональных компьютеров и "прародителем" целого семейства, которое называют семейством 80x86 или x86. В нашей стране аналог i8086 – микропроцессор КМ1810ВМ86.

В июне 1979 года появился микропроцессор i8088, архитектурно повторяющий i8086. Микропроцессор i8088 имел 16-битные внутренние регистры, но только 8-битную внешнюю шину данных. Чип i8088 содержал около 29 тысяч транзисторов и, благодаря 20 адресным линиям, позволял физически адресовать область памяти в 1 Мбайт. Первоначально i8088 работал на частоте 4,77 МГц и имел быстродействие 0,33 MIPS (Million Instructions Per Second – миллион инструкций в секунду), однако впоследствии были разработаны его клоны, рассчитанные на более высокую тактовую частоту (например, 8 МГц).

Как раз в это время корпорация IBM образовала новое подразделение по разработке и производству персональных компьютеров, которое и приобрело крупную партию i8088 для первого выпуска знаменитых компьютеров IBM PC. Следует отметить, что в компьютерах IBM PC на основе i8088 использовалось всего лишь 640 Кбайт из мегабайтного адресного пространства i8088.

Опираясь на архитектуру i8086, достижения в архитектуре микрокомпьютеров и больших компьютеров и учитывая запросы рынка, в феврале 1982 года фирма Intel выпустила свой новый микропроцессор - i80286, содержащий около 130 тысяч транзисторов, известный также под наименованием 286. При тактовой частоте 8 МГц достигалась производительность 1,2 MIPS. Наряду с увеличением производительности этот микропроцессор может работать в двух режимах: *в режиме реального адреса* он эмулирует микропроцессор 8086, а в *защищенном режиме* виртуального адреса (Protected Virtual Address Mode) или P-режиме.

Таким образом, i80286 был первым процессором Intel, способным выполнять любые программы, написанные для его предшественников, и с тех пор программная совместимость стала отличительным признаком семейства микропроцессоров Intel. Впервые на уровне микросхем были реализованы многозадачность и управление виртуальной памятью. Защищённый режим работы позволял, например, таким операционным системам, как Windows 3.0 и OS/2, работать с оперативной памятью свыше 1 Мбайта. Благодаря новой 16-разрядной системной шине данных, которая была впервые использована в IBM PC/AT286, i80286 мог обмениваться с периферийными устройствами 2-байтными сообщениями. 24-адресные шины нового микропроцессора позволяли в защищённом режиме обращаться уже к 16 Мбайтам памяти.

Семейство микропроцессоров i386. В октябре 1985 года фирма Intel представила первый 32-разрядный микропроцессор i80386, который впоследствии получил название i80386DX. Новый чип содержал примерно 275 тысяч транзисторов и на тактовой частоте 16 МГц достигал быстродействия около 6 MIPS. Полностью 32-разрядная архитектура (32-разрядные регистры и 32-разрядная внешняя шина данных) была дополнена расширенным устройством управления памятью MMU (Memory Management Unit), которое включало блок сегментации (Segmentation Unit) и блок управления страницами (Paging Unit). Это позволяло легко переставлять сегменты из одного места памяти в другое (свопинг) и освобождать драгоценные килобайты памяти. 32 адресные линии микропроцессора позволяли физически адресовать 4 Гбайта памяти.

Режим виртуальной машины позволяет i80386DX переключаться в выполнении программ, управляемых различными операционными системами, например UNIX и MS-DOS. Микропроцессор i80386DX сохраняет с программным обеспечением, написанным для его предшественников, i 8086 и i 80286

В июне 1988 года появился микропроцессор i80386SX – более дешёвая альтернатива 32-разрядному i80386DX. Внутренние регистры i80386DX и i80386SX полностью идентичны, однако в отличие от i80386DX новый микропроцессор имел 16-разрядную шину внешних данных и 24-разрядную адресную (адресуемое пространство - 16 Мбайт). Программное обеспечение, написанное для i80386DX, корректно работало и на i80386SX. Новый микропроцессор хорошо вписывался в стандарт, поэтому уже к концу 1988 года объем выпуска микропроцессора i80386SX существенно превосходил все рекордные показатели для i80386DX.

В 1990 году для удовлетворения потребностей рынка портативных компьютеров фирмой Intel был разработан микропроцессор i80386SL, поскольку микросхемы i80386DX/SX не полностью отвечали требованиям разработчиков компьютеров типа *лэптоп* (laptop) и *ноутбук* (notebook). Чип i80386SL содержал около 855 тысяч транзисторов и использовал базовую архитектуру микропроцессора i80386SX, дополненную ещё несколькими вспомогательными контроллерами. В МПК для построения портативных компьютеров входил периферийный контроллер i82360SL, который впервые использовал новое прерывание (System Management Interrupt =SMI), которое использовалось, в частности, для обработки событий, связанных с управлением потребляемой мощностью. МПК, включавший также математический сопроцессор i80386SL, позволял создать 32-разрядный компьютер, умещающийся на ладони.

Семейство микропроцессоров i486. Семейство i486 разделяется на три класса: DX, SX и LX, причем все процессоры i486 имеют 32-разрядную архитектуру и внутреннюю кэш-память, допускают адресацию физической памяти размером 64 Тбайт, а также предусматривают несколько новых возможностей для построения мультипроцессорных систем, например команды для поддержки механизма семафоров памяти и аппаратно-реализованное управление для обеспечения согласованности между несколькими модулями кэш-памяти. Модели SX отличаются тем, что не имеют встроенного сопроцессора.

В 1989 году фирма Intel впервые представила микропроцессор i486DX, который содержит более миллиона транзисторов на одном кристалле и

полностью совместим с процессорами ряда x86. МПК i486 включает в себя следующие микросхемы: i80486 - быстродействующий 32- разрядный процессор; i82596CA - 32- разрядный сопроцессор LAN; i82320 - контроллер магистрали Micro Channel (MCA); i82350 - контроллер магистрали EISA и другие.

Микросхема i486DX впервые объединила на одном кристалле такие устройства, как центральный процессор, математический сопроцессор и кэш-память 8 КВ со сквозной записью. Встроенный математический сопроцессор существенно облегчает и ускоряет математические вычисления, а встроенная кэш-память ускоряет выполнение программ за счёт промежуточного хранения часто используемых команд и данных. Использование конвейерной архитектуры, характерной для RISC-процессоров, позволило увеличить производительность в четыре раза по сравнению с обычными 32-разрядными системами за счет уменьшения количества тактов для выполнения каждой команды. На тактовой частоте 25 МГц микропроцессор i486DX показал производительность 16,5 MIPS.

Микропроцессор i486DX был ориентирован на применение в сетевых серверах и рабочих станциях.

В 1991 г. фирма Intel выпустила новый микропроцессор i486SX, отличающийся от оригинального i486DX тем, что математический сопроцессор у него заблокирован. Исключение затрат на тестирование сопроцессора позволила фирме Intel существенно снизить цены на новый микропроцессор. Это было оправдано тем, что статистика показала, что математический сопроцессор используют только 30% пользователей.

i486SX послужил основой для создания мощных настольных компьютеров. Например, компьютеры на базе i486SX с тактовой частотой 20 МГц работают быстрее (примерно на 40%) компьютеров, основанных на i80386DX с тактовой частотой 33 МГц.

В 1992 г. для использования в портативных компьютерах фирма Intel выпустила микросхему i486SL, сочетающую качества семейств Intel: i486DX и i80386SL. Этот самый производительный процессор серии SL, разработанный фирмой Intel, имеет пониженное напряжение питания (3,3 В) и эффективное управление энергоснабжением (как в i80386SL). 16-разрядная шина высокоскоростного периферийного интерфейса PI поддерживает быстрый интерфейс графического дисплея и устройств хранения информации на основе флэш-памяти. С конца 1993 года фирма Intel начала выпускать новую серию микропроцессоров 486SL Enhanced с напряжением питания 3,3 В и развитой технологией энергосбережения, которая заменила 5-вольтовые серии SX, DX,

DX2 и OverDrive-процессоры (основное различие между процессорами серии DX2 и OverDrive Intel состоит в том, что первые монтируются на системных платах ещё при сборке компьютеров, а вторые должны устанавливаться самими пользователями).

Технология умножения частоты. Второе поколение микропроцессоров i486 связано с появлением в начале 1992 года микропроцессоров серии DX2. Модели серии DX2 используют новую технологию, при которой скорость работы внутренних блоков микропроцессора (арифметико-логического устройства, математического сопроцессора, устройства управления памятью, кэш-памяти) в два раза выше скорости остальной части системы, в то время как остальные элементы системной платы (системная и внешняя кэш-память, вспомогательные микросхемы) работают с обычной скоростью. Например, процессор 486DX2-66 устанавливается на 33-мегагерцовую системную плату, что позволяет повысить быстродействие почти в два раза, так как эффективность кэширования внутренней кэш-памяти составляет около 90 процентов. Тем самым появилась возможность объединения высокой производительности микропроцессора с внутренней тактовой частотой 50/66 МГц и эффективной по стоимости 25/33-мегагерцовой системной платой.

Микропроцессоры 486DX4-75, 486DX4-83 и 486DX4-100 семейства DX4, имеющие кодовое название P24C, имеют кэш-память 16 Кбайт и предназначены для установки на системные платы, работающие на тактовой частоте 25 и 33 МГц. Напряжение питания этих процессоров составляет 3,3 В, а количество транзисторов на кристалле - 1,6 миллиона. По производительности они занимают нишу между DX2-66 и Pentium-60/66.

Отметим, что повышение тактовой частоты процессоров сопровождается существенным увеличением потребляемой мощности. Технология умножения частоты (не только в два, но и, например, в полтора, два с половиной или три раза) широко используется во всех современных процессорах.

Pentium. Процессор Pentium – это, по-видимому, главное достижение фирмы Intel. Разработка процессора Pentium началась в 1989 году, а в марте 1993 года фирма Intel объявила о начале промышленных поставок 66- и 60-мегагерцовых версий процессора Pentium (известного ранее как 586 или P5). Достоинством процессора Pentium является полная совместимость систем, построенных на базе этого микропроцессора, со 100 миллионами персональных компьютеров, использующих микропроцессоры i8088, i80286, i80386 и i486.

Если для 486-х процессоров использовалась CMOS-технология, то для для производства Pentium фирма Intel применила 0,8-микронную BiCMOS-

технологию. Микросхема процессора имеет более 3 миллионов транзисторов, 32-разрядную адресную и 64-разрядную внешнюю шину данных, что обеспечивает обмен данными с системной платой со скоростью до 528 Мбайт/с.

Суперскалярная архитектура процессора Pentium имеет два пятиступенчатых конвейера, работающих независимо и обрабатывающих две команды за один такт синхронизации.

Наиболее интересное новшество, используемое в Pentium, это небольшая кэш-память Branch Target Buffer - BTB (буфер меток перехода), которая позволяет динамически предсказывать переходы в исполняемых программах. Другое новшество заключается в том, что Pentium имеет два раздельных 8-Кбайтных кэша: один для команд и один для данных. Pentium с тактовой частотой 66 МГц обеспечивает производительность 112 MIPS (миллионов операций в секунду).

Процессор Pentium имеет усовершенствованный встроенный блок с восьмиступенчатой конвейеризацией для выполнения операций с плавающей запятой, что позволяет выполнять такие операции за один такт. Высокое быстродействие процессора Pentium позволило реализовать мультимедийный приложения, но в настоящее время эти микросхемы сняты с производства.

Pentium Pro. В 1995 году фирма Intel начала производить микропроцессор нового поколения - Pentium Pro (ранее именовался P6), использующий технологию Dynamic Execution, для которой характерны многократное предсказание ветвлений, анализ потоков данных и эмуляция выполнения инструкций. Отличительная особенность архитектуры Pentium Pro – масштабируемость, т.е. возможность объединения большого числа процессоров.

Корпус микросхемы (PGA) имеет 387 выводов, содержит отдельный кристалл кэш-памяти второго уровня (256- или 512-Кбайт, соответственно 15,5 или 31 миллион транзисторов) и кристалл процессора (5,5 миллионов транзисторов), на котором расположен 16-Кбайтный кэш. При напряжении питания около 3 В процессор (вместе с кэш-памятью второго уровня) потребляет около 14 Вт.

На сегодняшний день в семейство Pentium Pro входят микропроцессоры с тактовыми частотами 200, 180, 166 и 150 МГц. Микросхема Pentium Pro 150 выпускается согласно технологическим нормам 0,6 мкм, а процессоры с более высокой тактовой частотой используют уже технологические нормы 0,35 мкм.

Показатель производительности для Pentium Pro 200 по тесту SPECint92 равен 366, т.е. выше производительности RISC-архитектур.

Pentium II Xeon. В 1998 году корпорация Intel представила миру самый мощный процессор архитектуры x86, а именно Pentium II Xeon (на английском произносится как «Зеон», хотя по нормам русского и греческого языков можно произносить как «Ксеон»).

Тактовая частота достигает 450 МГц, а емкость кэш-памяти второго уровня 2 Мбайт, причем она работает на тактовой частоте процессорного ядра. Увеличение ёмкости емкость кэш-памяти повышает пропускную способность системы благодаря мгновенному доступу процессоров к часто используемым данным и инструкциям. Например, увеличение ёмкости кэша с 512 Кбайт до 1 Мбайт приводит к 20% росту общей производительности системы.

В связи с таким увеличением ёмкости кэш-памяти увеличиваются теплоотдача, размеры и вес процессорного блока и поэтому он устанавливается в разъём новой конструкции Slot 2. Возникает также необходимость контроля температурного режима блока.

Большая емкость кэш-памяти второго уровня позволяет получить почти пропорциональный рост производительности системы по мере установки дополнительных процессоров с мегабайтным кэшем. Архитектура Xeon, допускающая 36-разрядную адресацию физической памяти, позволяет процессору получать доступ к системной памяти ёмкостью до 64 Гбайт. Однако новый механизм постраничного обмена PSE-36 (Page Size Extension-36) в настоящее время поддерживают только операционные системы Windows NT, SCO UnixWare и Sun Solaris, а для остальных − требуется обновить драйвер блока управления памятью.

Выпускаются два варианта микросхемных комплектов Intel 450NX PCIset, оптимизированных для Pentium II Xeon, имеющих одинаковую структуру ядра, но отличающихся производительностью и ценой: Basic для серверных систем и Full для систем среднего уровня.

Комплект Basic PCIset поддерживает до двух разъёмов 32-разрядной шины PCI, один разъем для 64-разрядной PCI и до 4 Гбайт системной памяти типа EDO. Комплект Full PCIset имеет более совершенные характеристики: он поддерживает до четырёх слотов типа EDO. Эти комплекты имеют одинаковую 100-мегагерцовую системную шину и поддерживают многопроцессорные конфигурации (до четырёх Xeon).

Появляется возможность согласовать мощность процессора и производительность подсистемы ввода-вывода: 64-разрядная шина PCI способна существенным образом повысить общую производительность системы при использовании оптоволоконной технологии обмена данными с

дисковыми массивами и высокопроизводительных сетевых магистралей на основе ATM, Gigabit Ethernet и других.

Для рабочих и графических станций разработан комплект Intel 440GX AGPset на базе известного микросхемного набора 440BX. Комплект Intel 440GX дает возможность обращения к памяти ёмкостью до 2 Гбайт, что в два раза больше, чем у 440BX. Производительность видеоподсистемы рабочих и графических станций возрастает, поскольку 440GX управляет работой порта AGP в режиме удвоения частоты(2x) с использованием технологии “двойной накачки” - данные передаются как по переднему, так и по заднему фронтам тактовых импульсов (у обычной AGP - только по переднему), при этом полоса пропускания достигает значения 533 Мбайт/с, а физические параметры интерфейса AGP остаются прежними.

Фирма Intel разрабатывает также многопроцессорные системы. Сначала создавались системы с четырьмя устройствами на одной плате, затем симметричные мультипроцессорные системы, поддерживающие до восьми микропроцессоров Xeon. Предусматривается возможность кластерных решений. Например, в процессорной шине набора Intel 450NX PCIset предусмотрен разъём кластерного соединения на основе стандартных четырёхпроцессорных узлов. При использовании архитектуры распределённой памяти, как правило, не требуется «переписывать» прикладные программы.

Рассмотренные выше микропроцессоры фирмы Intel реализуют CISC-архитектуру. Некоторые фирмы (например, AMD, Cyrix, Texas Instruments, IBM) выпускают процессоры, которые совместимы с рассмотренными выше процессорами Intel, однако имеют свои характерные особенности. Назовем также серию MC680XX CISC-микропроцессоров фирмы Motorola.

Как Intel, так и другие фирмы выпускают гораздо более мощные процессоры, относящиеся как к CISC-, так и к RISC-архитектуре. Микропроцессоры с RISC-архитектурой (80860, 80960, 80870, Power PC) выпускаются или выпускались многими фирмами. В качестве примера, назовем также проект Alpha AXP фирмы Digital Equipment, первоначально ориентированный на обработку 64-разрядных приложений в среде Unix, а позднее дополненный средствами поддержки операционной системы Microsoft Windows NT.

Компании IBM, Motorola и Apple совместно реализовали проект создания семейства RISC-процессоров POWER (Performance Optimised With Enchanced RISC) для широкого профиля применений в качестве серверов и ПК типа Macintosh. К микропроцессорам с RISC-архитектурой относятся также изделия ARM (Acorn RISC Machine) фирмы Acorn.

Если первый микропроцессор, а именно i4004, выпущенный в 1971 году, имел около 2300 транзисторов и работал на тактовой частоте 750 кГц, то

современные микропроцессоры могут иметь более 100 миллионов транзисторов и тактовую частоту в десятки гигагерц. Соответственно возросли технические характеристики и функциональные возможности МПС.

Глава 2
АРХИТЕКТУРА МИКРОПРОЦЕССОРНЫХ СИСТЕМ

2.1. Унифицированный системный интерфейс

Внутримашинный системный интерфейс – это система связи и сопряжения всех узлов и блоков микроЭВМ в виде набора электрических линий (проводов), схем сопряжения и алгоритмов (протоколов) передачи и преобразования сигналов [2, 4, 5, 6, 9, 15]. Можно выделить два варианта построения внутримашинного системного интерфейса.

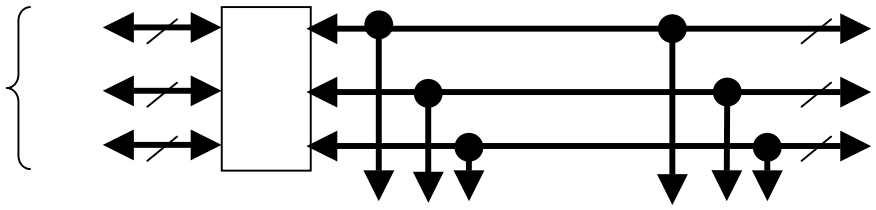
Многосвязный интерфейс: каждый блок микроЭВМ связан с другими блоками своими отдельными электрическими линиями (применяется в простейших микроЭВМ).

Односвязный интерфейс: все блоки микроЭВМ связаны друг с другом через общую системную шину.

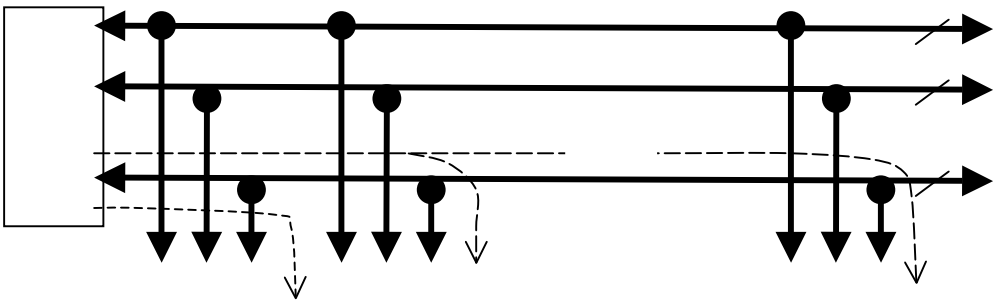
Хранимая в памяти программа, адресное обращение устройств друг к другу и магистрально-модульная структура являются характерными признаками архитектуры микроЭВМ. Центральный процессор (ЦП) микроЭВМ имеет набор выводов для подачи электропитания и соединения с оперативным запоминающим устройством и контроллерами внешних устройств. Выделяют три логические группы выводов (контактов) БИС ЦПЭ и соответственно шин:

- Шина адреса (ША, address bus = AB) служит для передачи адреса устройства, затребованного ЦПЭ для проведения обмена.)
- Двунаправленная шина данных (ШД, data bus=DB) используется для обмена командами и данными (операндами) между блоками МПС.
- Шина управления (ШУ, control bus=CB) служит для передачи сигналов, управляющих организацией обмена и работой микропроцессора.

Эти три шины и шина электропитания образуют унифицированный системный интерфейс, называемый *системной шиной* микроЭВМ.

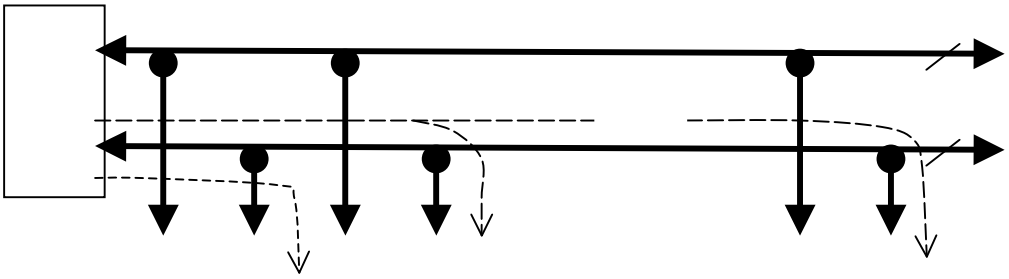


На рис.2.1 представлена структура микрокомпьютера с раздельными двунаправленными шинами, в которой используются отдельные шины адреса данных и управления для связи с памятью и внешними устройствами. Структура с раздельными шинами при реализации в виде одного кристалла требует слишком большого числа выводов БИС. Первый шаг к сокращению числа выводов БИС заключается в совместном использовании ША, ШД и ШУ внешними устройствами и памятью, что приводит к структуре на рис. 2.2.

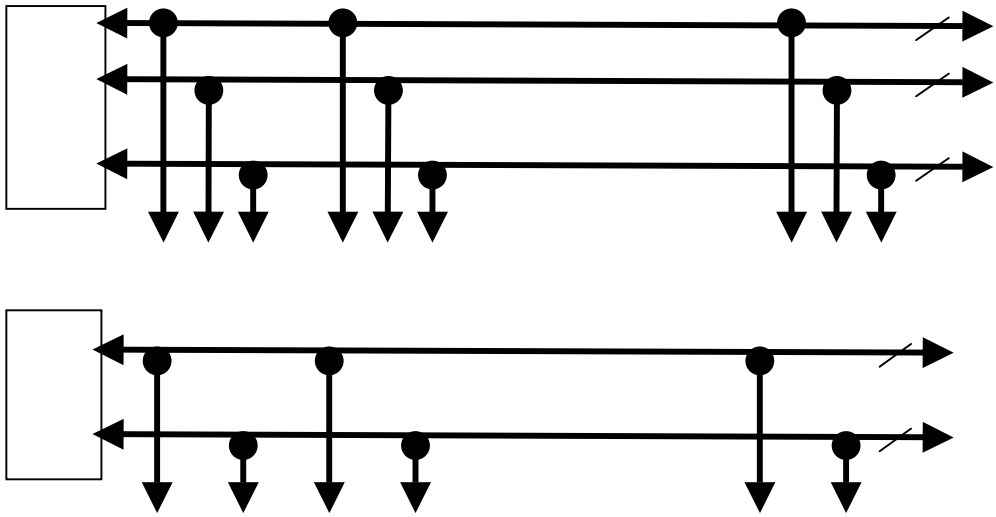


Однако при этом теряется возможность одновременного обмена данными между МП, памятью и каким-либо ВУ. Кроме того, приходится вводить управляющие сигналы R/W (READ/WRITE) для обмена с памятью и I/O (INPUT/OUTPUT) – для обмена с ВУ.

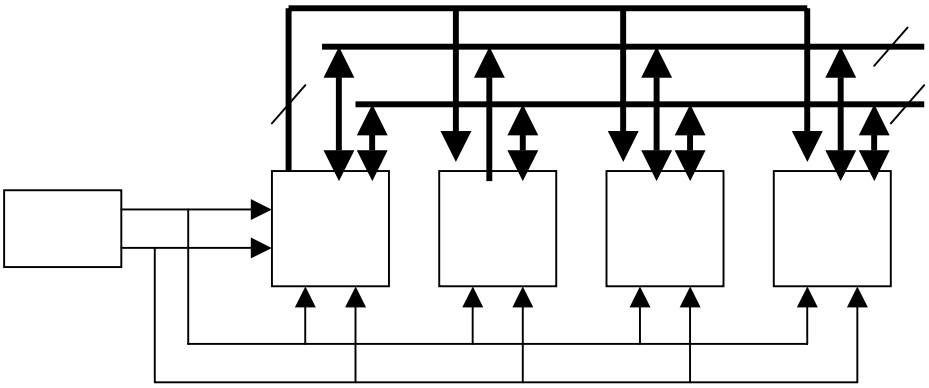
Для дальнейшего сокращения числа выводов СБИС можно совместить (мультиплексировать) ША и ШД (рис.2.3).



Дальнейшее упрощение структуры достигается за счет исключения команд ввода-вывода из системы команд микрокомпьютера. В этом случае для ввода и вывода (обмена данными с ВУ) используют любые команды чтения и записи в память, причем часть адресного пространства памяти отводится для адресов регистров контроллеров ВУ. Такая структура называется *структурой с общей шиной*. На рис.2.4 представлены два варианта структуры с общей шиной: а) без мультиплексирования; б) с мультиплексированием ША и ШД.



На рис.2.5 представлена более подробная трехшиннная (т.е. без мультиплексирования ША и ШД) структура микрокомпьютера для варианта с общими шинами (рис.2.4, а).



Генератор тактовых импульсов (ГТИ, clock) синхронизирует работу всех блоков микроЭВМ. Центральный процессорный элемент (ЦПЭ, central processing unit=CPU) микроЭВМ, или микропроцессор — электронный прибор, работающий под управлением программы, выполненный обычно на одном полупроводниковом кристалле по технологии БИС или СБИС. Постоянное запоминающее устройство (ПЗУ, read only memory=ROM) служит для долговременного хранения на протяжении всего жизненного цикла микроЭВМ постоянно используемых программ и данных (например, программ и данных специализированных микроЭВМ). Оперативное запоминающее устройство (ОЗУ, память с произвольным доступом, random access memory=RAM) служит для временного хранения программ и данных. Устройства ввода-вывода (УВВ, input-output=I/O) служат для связи микроЭВМ с внешней средой:

- для ввода данных с клавиатуры, датчиков, линий передачи или из внешних ЗУ;
- для вывода данных на экран монитора, систему привода исполнительных механизмов, в линии передачи или на внешние ЗУ.

Для подключения внешних устройств к системной шине используются контроллеры (адаптеры) ВВ. Функция контроллеров – это согласование уровней электрических сигналов, а в случае необходимости также и преобразование форматов машинных данных при их пересылке из одного устройства в другое. В процессе ввода-вывода передается информация двух видов: управляющие данные (слова) и собственно данные, или данные-сообщения.

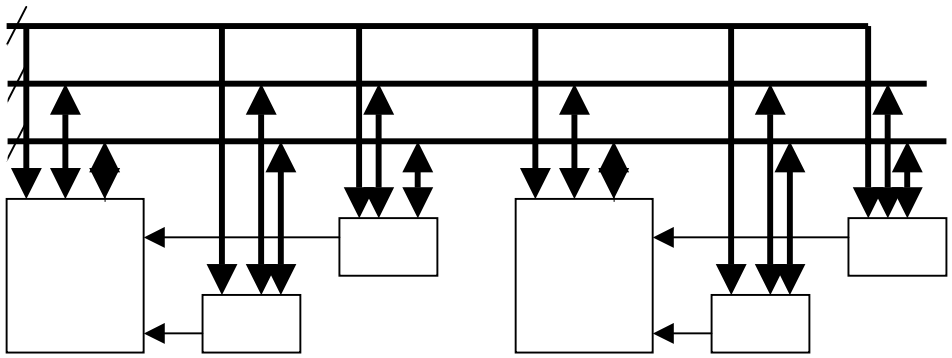
Следует различать четыре вида обмена УВВ микроЭВМ с внешней средой:

1. С устройствами, всегда готовыми к обмену (ПЗУ, датчики непрерывных физических величин).

2. С ожиданием сигнала готовности устройства.

3. С прерыванием выполняемой программы для обслуживания пересылок данных.

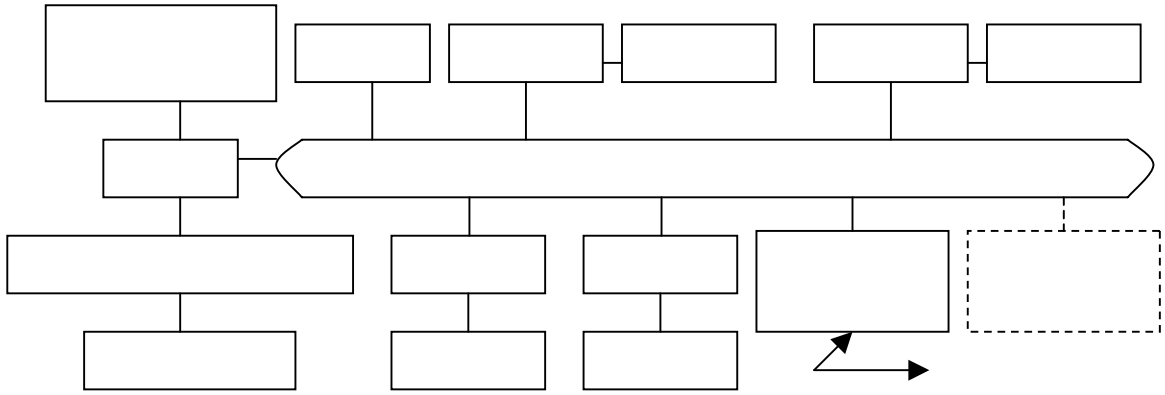
4. С прямым доступом в память (пересылки между такими периферийными устройствами, как НЖМД и ОЗУ без участия процессора).



Для организации процесса обмена используются контроллеры, управляющие системными связями, согласованием и буферизацией (временным хранением) данных. На рис.2.6 представлена схема подключения внешних устройств с контроллером прерываний и контроллером прямого доступа в память (ПДП).

Центральный процессор может быть выполнен в виде одной БИС (*однокристальный* ЦП) или в виде набора малоразрядных секций (многокристальный *секционный* ЦП). Однокристальные ЦП имеют фиксированную разрядность и набор машинных команд. Повышение разрядности вычислений возможно только программным путем. Секционные ЦП позволяют наращивать разрядность машинных слов и изменять состав машинных команд за счет микропрограммирования.

Односвязный системный интерфейс микроЭВМ может быть построен с использованием шин расширений и локальных шин.



Шина расширений – это шина общего назначения для подключения разнообразных внешних устройств: НЖМД и НГМД, мониторов, модемов, сетевых карт и т. д. На рис. 2.7, *а* приведен вариант организации односвязного интерфейса микроЭВМ только на основе шины расширения.

Локальная шина служит для обеспечения высокой пропускной способности при подключении небольшого числа устройств определенного класса, например основной памяти (ОЗУ и ПЗУ) и видеосистемы.

В табл. 2.1 приведены характеристики некоторых шин расширения и локальных шин. Кроме указанных в табл. 2.1 в качестве интерфейса только для внешних ЗУ широко используются также локальные шины IDE (Integrated Device Electronics), EIDE (Enhanced IDE) и SCSI (Small Computer Interface).

Шины PC/XT (Personal Computer eXtended Technology) и PC/AT (PC Advanced Technology) являются вариантами ранее широко распространенной системной шины Mulibus1. С появлением 32-разрядных МП шина ISA (Intdustry Standard Architecture) стала препятствием к увеличению быстродействия ПК. Для применения в быстродействующих сетевых серверах и рабочих станциях в 1989 г. была предложена шина EISA (Extended ISA), полностью совместимая с ISA .

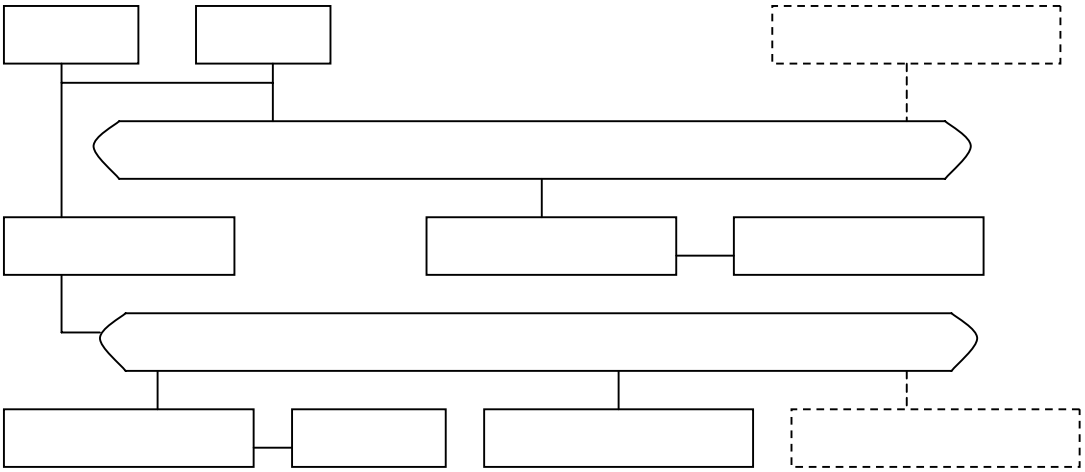
Шина MCA (Micro Channel Architecture), предложенная в 1987 г. фирмой IBM, предназначается только для машин PS/2, поэтому используется редко.

Локальные шины подключаются непосредственно к шине МП и работают на тактовой частоте МП.

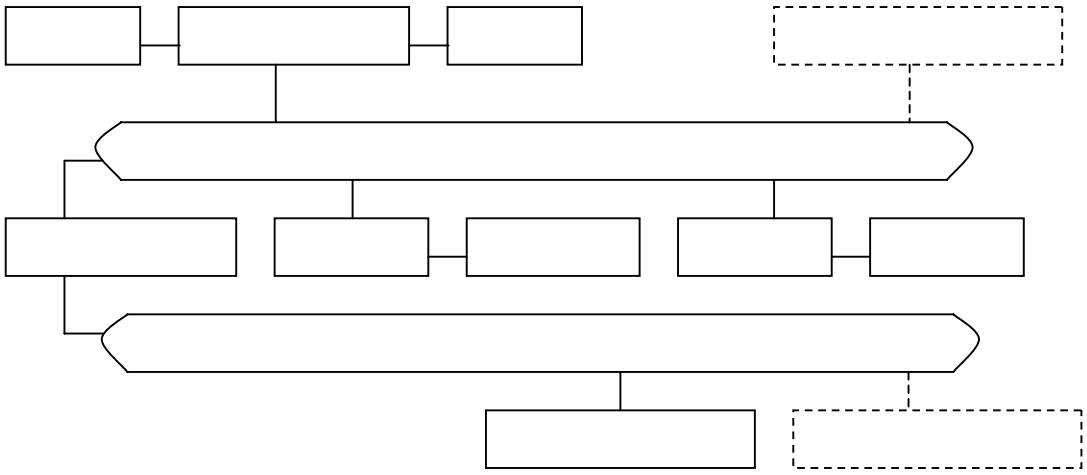
Таблица 2.1. **Типы шин расширения и локальных шин**

Параметр	PC/ XT bus	PC/ AT bus	ISA	EISA	MCA	VLB	PCI
Разрядность шины, бит							
данных	8	16	16	32	32; 64	32; 64	32; 64
адреса	20	24	24	32	32	32	32
Число линий							
для прерываний	4	7	15				
для прямого доступа	4	4	11				
Рабочая частота, МГц	4,77	8; 16	8	8–33	10–20	до 33	до 33
Пропускная способность, Мбайт/с							
теоретическая			4	33	76	132	132; 264
практическая			2	8	20	80	50; 100
Число подключаемых устройств			6	15	15	4	10
Тип МП	8086, 8088	80286	80386, 80486	80386, 80486	PS/2	80386, 80486	80486, Pentium Pro, Power PC

На рис. 2.8 приведена структура микроЭВМ с локальной шиной VLB и шиной расширения ISA/EISA. Шина VLB (VESA Local Bus) разработана в 1992 г. Ассоциацией стандартов видео оборудования (VESA – Video Electronics Standards Association).



На рис. 2.9 приведена структура микроЭВМ с локальной шиной PCI и шиной расширения ISA/EISA. Шина PCI (Peropheral Component Interconnect), разработанная в 1993 г. фирмой Intel, является более универсальной, чем шина VLB, и поэтому получила большее распространение.



2.2. Микропроцессоры

Во многом характеристики МПС зависят от того, как реализована функция микропроцессора, выполняющего обработку данных в соответствии с программой.

Микропроцессор (МП) – электронный прибор, выполненный на полупроводниковом кристалле по технологии БИС и СБИС и работающий под управлением программы [2, 3, 4, 5, 7, 10].

МП и микроЭВМ можно классифицировать по составу МПК и конструктивному исполнению:

- однокристальные;
- многокристальные;
- секционные (с разрядно-модульной организацией).

ЦП обрабатывает последовательность команд, задаваемую программой. Команда содержит сведения о выполняемой операции, операндах и следующей команде, которая должна быть выполнена. Выполнение команды состоит из следующих этапов: прием команды из ОЗУ или ПЗУ, ее расшифровка и выполнение, в ходе которого ЦП получает все необходимые данные.

Этапы выполнения команды:

- выборка (чтение) команды из ОЗУ или ПЗУ;
- декодирование (расшифровка) команды, т.е. выделение кода операции и адресов операндов;
- формирование исполнительного (физического) адреса каждого операнда, его чтение и пересылка в ЦП, если он вне ЦП;
- выполнение закодированной операции;
- сохранение или вывод результата операции.

Выделяют 3 логические группы выводов (контактов) в БИС МП и МПК:

- Шина адреса, которая служит для передачи адреса устройства, затребованного МП для проведения обмена.
- Двунаправленная шина данных, по которой МП принимает команды и операнды, выставляет результат.
- Шина управления для передачи сигналов, управляющих организацией обмена и самим микропроцессором.

Существуют два варианта построения набора команд микропроцессора: процессоры RISC-архитектуры и процессоры CISC-архитектуры.

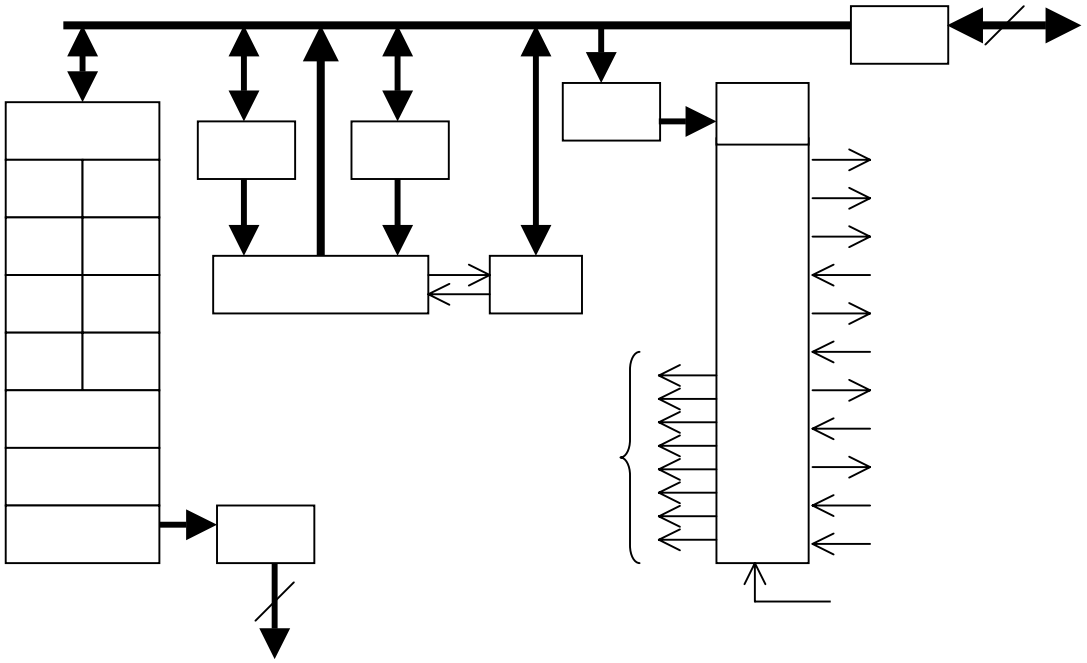
Микропроцессоры с архитектурой CISC (Complex Instruction Set Computers) имеют полную систему команд различной сложности (от простых, характерных для микропроцессоров первого поколения, до самых сложных, характерных для современных 32-разрядных микропроцессоров), реализованных на уровне машинного языка.

Архитектура RISC (Reduced Instruction Set Computers) использует сравнительно небольшой (сокращённый) набор команд, определённый в результате анализа частоты использования команд для основных областей применения CISC-процессоров.

Команды RISC-процессоров имеют простую структуру, реализуются полностью аппаратным способом с использованием эффективного конвейера при небольшом объеме оборудования, поэтому эти микропроцессоры работают в несколько раз быстрее, чем CISC-процессоры, имеющие ту же тактовую частоту.

Для учебных целей рассмотрим архитектуру микроЭВМ на базе микропроцессора КР580ИК80А. Система команд КР580ИК80А приведена в приложении. На рис. 2.10 представлена структура (функциональная схема) этого микропроцессора. В состав БИС КР580ИК80А входят:

- 8-разрядное арифметико-логическое устройство (ALU);
- регистр признаков (RS), фиксирующий признаки, вырабатываемые ALU в процессе выполнения команд;
- аккумулятор (ACC), или регистр А;
- буферный регистр (RG);
- регистр команд (IR) для хранения первого байта команды, содержащего код операции;
- дешифратор команд (DC);
- блок регистров для приема, хранения и выдачи данных в процессе выполнения команды, содержащий программный счетчик (PC), указатель стека (SP), регистр адреса (RGA), шесть регистров общего назначения (B,C,D,E,H,L) и вспомогательные регистры (W и Z);
- схема управления и синхронизации (CU), вырабатывающая последовательность управляющих сигналов для работы ALU и блока регистров;
- 16-разрядный буферный регистр адреса (BA);
- 8-разрядный буферный регистр данных (BD), двунаправленный мультиплексор (MUX) для обмена операндами и результатами операций и блоком регистров по внутренней шине данных (DB).



Программный счетчик PC хранит текущий адрес команды и автоматически увеличивается на 1, 2 или 3 в зависимости от формата выполняемой команды.

Указатель стека SP содержит текущий адрес вершины стека, который автоматически уменьшается на 2 при загрузке данных в стек и увеличивается на 2 при извлечении данных из стека. Для организации стека можно использовать любую область ОЗУ объемом до 64К.

Микропроцессор имеет внешнюю 16-разрядную шину адреса A(15-0) с тремя состояниями, 8-разрядную двунаправленную шину данных D(7-0) с тремя состояниями, два вывода (CKL1 и CLK2) для фаз синхронизации, а также четыре входных и шесть выходных выводов управления:

WR – сигнал “выдача” (напряжение L-уровня указывает на выдачу байта данных на шину D(7-0) для пересылки в ЗУ или УВВ);

DBIN – сигнал “прием” (напряжение H-уровня указывает на прием с шины D(7-0) байта данных, выданного ЗУ или УВВ);

INTE – сигнал “разрешение прерывания”;

INT – сигнал “запрос на прерывание”;

HLDA – сигнал “подтверждение захвата” (напряжение H-уровня указывает на перевод шин адреса и данных в третье состояние);

HOLD – сигнал “захват” (напряжение H-уровня указывает на запрос от другой системы на управление шинами системы);

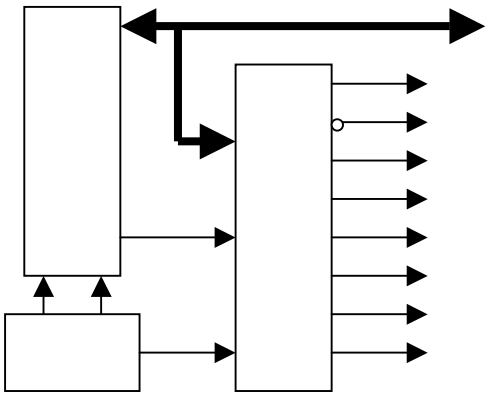
WAIT – сигнал “ожидание” (напряжение H-уровня указывает на состояние ожидания МП);

READY – сигнал “готовность” для синхронизации МП с ЗУ или УВВ (напряжение Н-уровня указывает на готовность данных на шине D(7-0) к вводу в МП или на готовность внешних устройств к приему данных);

SYNC – сигнал синхронизации (напряжение Н-уровня указывает на начало каждого машинного цикла);

RESET – сигнал сброса (устанавливает счетчик команд в нуль, сбрасывает триггеры разрешения прерывания и захвата шин).

Длительность машинного такта равна периоду следования тактовых импульсов CKL1 и CLK2, задаваемому генератором CLG, а длительность машинного цикла составляет от трех до пяти машинных тактов.



Операции, выполняемые МП в машинном цикле, характеризуются 8-разрядным словом (байтом) состояния процессора. Байт состояния выдается на шину данных в такте Т2 каждого машинного цикла. На рис.2.11 приведен пример фиксации слова состояния с помощью многорежимного буферного регистра К589ИР12.

Таблица 2.2. ***Сигналы состояния процессора***

Обозначение сигнала	Разряд ШД	Назначение сигнала
INTA	D0	Подтверждение прерывания; используется для стробирования приема команды RST в МП из схемы прерывания
WO	D1	Запись; L-уровень сигнала указывает на запись данных из ЗУ или ВУ; Н-уровень сигнала – запись в микропроцессор
STACK	D2	Стек; Н-уровень сигнала указывает, что на шине адреса установлено содержимое SP
HLTA	D3	Подтверждение останова; Н-уровень сигнала указывает на переход МП в состояние останова
OUT	D4	Вывод; Н-уровень сигнала указывает, что на шине адреса установлен код ВУ и можно вывести данные из МП по сигналу DBIN=1
M1	D5	Н-уровень сигнала указывает, что МП принимает первый байт команды
INP	D6	Ввод; Н-уровень сигнала указывает, что на шине адреса установлен код ВУ и можно ввести данные в МП по сигналу DBIN=1
MEMR	D7	Чтение; Н-уровень сигнала указывает, что осуществляется чтение содержимого ЗУ по адресу, установленному на шине адреса

В табл.2.2 приведены сигналы состояния процессора, а в табл.2.3 – разновидности машинных циклов.

Таблица 2.3. **Разновидности машинных циклов**

Машинный цикл	Обозначение сигнала слова состояния процессора							
	INTA	WO	STACK	HLTA	OUT	M1	INP	MEMR
Цикл М1 – выборка команды	0	1	0	0	0	1	0	1
Цикл чтения из ЗУ	0	1	0	0	0	0	0	1
Цикл записи в ЗУ	0	0	0	0	0	0	0	0
Цикл чтения из стека	0	1	1	0	0	0	0	1
Цикл записи в стек	0	0	1	0	0	0	0	0
Цикл ввода	0	1	0	0	0	0	1	0
Цикл вывода	0	0	0	0	1	0	0	0
Цикл прерывания	1	1	0	0	0	1	0	0
Цикл останова	0	1	0	1	0	0	0	0
Цикл прерывания при останове	1	1	0	1	0	1	0	0

2.3. Постоянные и оперативные запоминающие устройства микропроцессорных систем

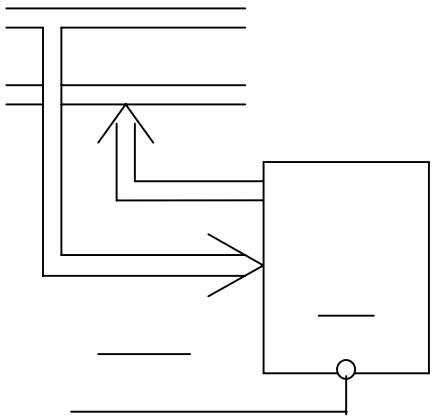
Для хранения данных в микропроцессорных системах используются полупроводниковые, магнитные и оптические запоминающие устройства. на основе внешних носителей. Основной объем внутренней памяти компьютера сосредоточен в интегральных микросхемах (ИМС) оперативных (ОЗУ) и постоянных запоминающих устройств (ПЗУ). ОЗУ может быть статическим и динамическим, а ПЗУ однократно или многократно программируемым. ПЗУ обеспечивают сохранение информации при отключении питания [2, 5, 15].

Кроме того, элементы памяти входят в состав других ИМС, – это внутренняя память центрального процессора, память контроллеров, видео- и других устройств различного назначения.

В качестве устройств используют в основном СБИС на основе кремния со структурой “металл-оксид-полупроводник” (МОП). Емкость памяти, степень

интеграции, электрические параметры при записи и хранении данных, быстродействие, помехоустойчивость, долговременная стабильность, устойчивость к внешним неблагоприятным воздействиям и другие характеристики ЗУ зависят от физических принципов работы, применяемых материалов и параметров технологических процессов изготовления ИМС ЗУ.

Важной характеристикой одной ИМС (одного кристалла) является организация памяти кристалла, обозначаемая как M x N, где M - число слов, N - разрядность слова. Емкость памяти ИМС обычно составляет 1024 бита, 4 Кбит, 16 Кбит, 64 Кбит, 1Мбит и т.п. Кристалл емкостью 16 Кбит может иметь различную организацию, например: 16К x 1, 4К x 4 и 2К x 8. При одинаковом времени обращения ИМС памяти с большей разрядностью имеет большую информационную емкость.



На рис.2.12 показано подключение одной микросхемы ПЗУ к модулю ЦП. Информация может быть считана из ПЗУ при задании адреса *A0 – AN* ячейки памяти и подаче сигнала $\overline{MEMR}$ на вход $\overline{CS}$ (Chip Select – выборка кристалла) с системного контроллера.

Для увеличения числа подключаемых микросхем ПЗУ необходимо использовать адресный дешифратор, на входы которого с адресной шины подаются старшие разряды адреса. Выходы дешифратора подключаются ко входам выбора микросхем $\overline{CS}$.

Полупроводниковые *оперативные запоминающие устройства* (ОЗУ) подразделяются на ОЗУ с произвольной выборкой, ЗУ с последовательным доступом и ассоциативные ЗУ. ОЗУ с произвольной выборкой подразделяются на статические (СОЗУ) и динамические оперативные запоминающие устройства (ДОЗУ). ЗУ с последовательным доступом подразделяются на регистры сдвига и приборы с зарядовой связью (ПЗС).

В основе большинства современных ОЗУ лежат комплиментарные МОП ИМС (КМОП), в которых используются пары МОП-транзисторов с разным типом канала: n-МОП и p-МОП. Эти ИМС отличаются малой потребляемой мощностью, поскольку как при низком, так и при высоком уровне сигнала на входе КМОП-инвертора один из транзисторов пары закрыт, поэтому

потребление энергии происходит только при переключении из "1" в "0" и обратно.

Наибольшая плотность упаковки достигнута в кристаллах динамической МОП-памяти. Следует учитывать, что СБИС динамической МОП-памяти в резервном режиме потребляют примерно в десять раз меньше энергии, чем в активном режиме.

В отличие от статических ЗУ, которые хранят информацию пока включено питание, в динамических ЗУ необходима постоянная регенерация информации, причем для хранения одного бита в динамических ОЗУ нужны всего 1-2 транзистора и накопительный конденсатор.

В микросхеме динамического ОЗУ есть один или несколько тактовых генераторов и логическая схема для восстановления информационного заряда, стекающего с конденсатора. Это несколько усложняет конструкцию ИМС.

Статические и динамические ОЗУ выполнены, как правило, в виде ЗУ с произвольной выборкой, поскольку они имеют ряд преимуществ перед ЗУ с последовательным доступом.

Основные характеристики полупроводниковых ЗУ

Перечислим основные характеристики и параметры ЗУ, которые необходимо учитывать при проектировании МПС:

1. *Емкость ЗУ*.
2. *Временные характеристики ЗУ*: время доступа и время восстановления. *Время доступа* – временной интервал, начинающийся в момент, когда ЦП выставляет адрес требуемой ячейки памяти на шину адреса и посылает приказ на чтение или запись данных по шине управления, и заканчивающийся в момент, когда устанавливается связь адресуемой ячейки с шиной данных. *Время восстановления* – это время, которое требуется, чтобы память пришла в исходное состояние после того, как ЦП прочитал данные с ШД и снял с ША – адрес, а с ШУ – сигнал "чтение" или "запись".
3. *Рассеиваемая мощность* (или потребляемая энергия) для активного режима, когда операции записи и считывания выполняются с номинальным быстродействием, и режима пассивного хранения.
4. *Условия эксплуатации: допустимая температура окружающей среды* отдельно для активной работы, для режима пассивного хранения данных и для нерабочего состояния с отключенным питанием; *допустимые механические воздействия и влажность*.

5. *Тип корпуса*.

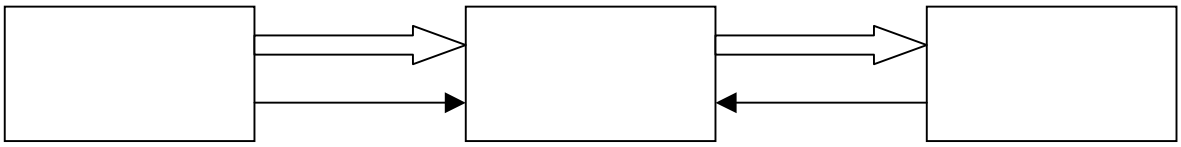
6. *Плотность упаковки* (определяется площадью запоминающего элемента и числом транзисторов в СБИС; зависит от используемой технологии).

7. *Удельная стоимость ЗУ* – это стоимость бита хранимой информации.

Буферная память

Буферная память используется для обмена данными между двумя подсистемами МПС в том случае, когда эти подсистемы имеют различное быстродействие, и в частности различную скорость передачи данных.

На рис.2.13 представлена схема обмена данными для подсистем А и Б, построенная на основе буферной памяти.



Данные от подсистемы А запоминаются в буферной памяти до того момента, когда подсистема Б готова к приему данных. К буферной памяти предъявляются следующие требования.

1. Буферная память должна работать по принципу FIFO (First In First Out, т.е. “первый вошел – первый вышел”) для того, чтобы сохранять порядок поступления данных от подсистемы А. Таким образом, в качестве буферной памяти типа FIFO целесообразно использовать ЗУ с последовательным доступом, которое выдает данные в порядке их поступления.

2. Буферная память должна быть "эластичной", т.е. записывать данные под управлением и с быстродействием подсистемы А, а считывать – под управлением и с быстродействием подсистемы Б.

3. В общем случае буферная память должна функционировать без синхронизации подсистем А и Б, т.е. допускать одновременное и независимое выполнение операций записи и считывания.

4. Емкость буферной памяти должна быть достаточной для хранения блоков данных, которые подсистема А формирует между считываниями их подсистемой Б.

Другие варианты организации обмена данными между подсистемами с разным быстродействием могут быть построены без использования буферной памяти типа FIFO, а именно на основе прерываний или канала прямого доступа к памяти.

В случае использования прерываний подсистема А формирует запросы на обслуживание по мере готовности данных к обмену, а подсистема Б обслуживает эти запросы. Однако обслуживание прерываний связано с непроизводительными потерями времени и в случае пакетного обмена данными производительность подсистемы Б существенно снижается.

Требуемую скорость обмена данными между подсистемами с разным быстродействием может обеспечить канал прямого доступа из подсистемы А к памяти подсистемы Б, однако для этого необходим довольно сложный контроллер ПДП.

Стековая память

Стековая память обеспечивает доступ к данным на основе механизма LIFO (Last In First Out, т.е. по принципу: "последним зашел – первым вышел"). Стековая память применяется при построении компиляторов и интерпретаторов для синтаксического анализа и вычисления арифметических выражений с использованием польской инверсной записи. Удобно применять стековую память в малых ЭВМ для реализации процедур вызова подпрограмм и при обработке прерываний.

Рассмотрим два варианта механизма доступа к стековой памяти: аппаратный и аппаратно-программный (внешний) стеки.

Аппаратный стек – это набор регистров, связи между которыми организованы таким образом, что при записи и считывании данных содержимое стека автоматически сдвигается. В большинстве МП аппаратный стек используется для хранения содержимого программного счетчика (стеком команд), причем его емкость варьируется от нескольких регистров до нескольких десятков регистров. Основное достоинство аппаратного стека – высокое быстродействие, а недостаток – ограниченная емкость.

Аппаратно-программный стек строится с использованием области памяти и специальных машинных команд. Для адресации аппаратно-программного стека используется специальный регистр, в который предварительно загружается указатель, определяющий адрес последней

занятой ячейки. Специальные команды CALL и RET используются для записи в стек и восстановления содержимого программного счетчика. В некоторых МП содержимое основных регистров запоминается в стеке автоматически при прерывании программ. Команды PUSH и POP используются для временного запоминания в стеке содержимого регистров и их восстановления. При выполнении команд PUSH и POP содержимое регистра указателя стека при записи (команда PUSH) автоматически уменьшается, а при считывании (команда POP) – увеличивается на 1.

2.4. Многомашинные и многопроцессорные системы

Вычислительные комплексы, системы и сети – это три основных направления реализации систем обработки дискретной информации.

Вычислительный комплекс (ВК) – это набор аппаратных средств в виде нескольких ЭВМ или микропроцессоров (МП) вместе с базовым ПО для реализации функций приема, хранения, обработки и выдачи информации.

Вычислительная система (ВС) – это система, ориентированная на решение задач определенного класса и состоящая из вычислительного комплекса, системного и прикладного ПО.

Вычислительная сеть – способ организации взаимосвязи нескольких ЭВМ.

Основное назначение ВК – повышение производительности и/или надежности вычислительных систем. Различают многомашинные (ММВК) и многопроцессорные ВК (МПВК).

Многомашинные ВК

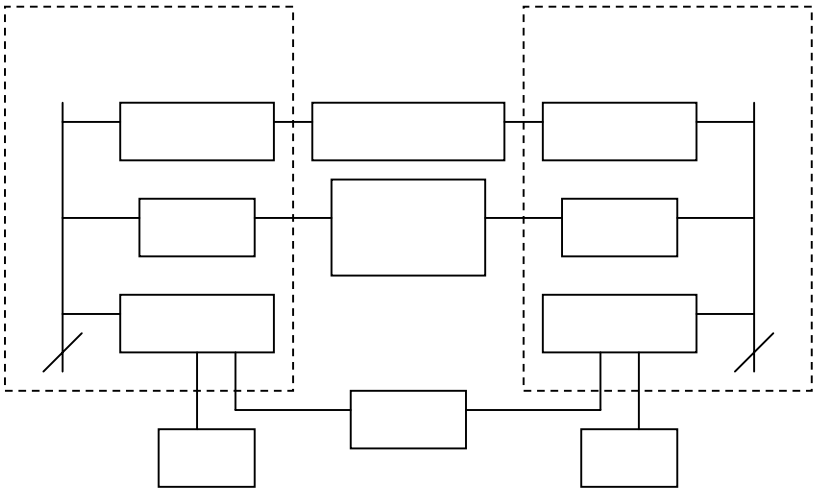
Многомашинный ВК функционирует как система со слабосвязанными элементами, то есть каждая ЭВМ управляет своей ОС. Обмен информацией между ЭВМ организуется в редких случаях с определенной целью. В зависимости от способа организации взаимодействия между ЭВМ различают ММВК с косвенным управлением, прямым управлением и сателлитные.

ММВК с косвенным управлением – это объединение нескольких микроЭВМ через общее внешнее ЗУ (ВЗУ). Таким образом, это слабосвязанный вычислительный комплекс, поскольку связь осуществляется

только через общую информацию. Такие комплексы применяются для повышения надежности ВС за счет холодного или горячего резервирования.

В случае *холодного резервирования* вычислительный процесс выполняется только на одной ЭВМ. При обнаружении отказа вычислительный процесс переносится на другую ЭВМ, которая до этого находилась в нерабочем (“холодном”) состоянии. Общее ВЗУ используется для передачи информации, необходимой для продолжения вычислений, с отказавшей машины на резервную.

В случае *горячего резервирования* две или более ЭВМ находятся в рабочем (“горячем”) состоянии и одновременно выполняют один и тот же вычислительный процесс, что позволяет в случае отказа или сбоя выполнить переключение на исправную резервную ЭВМ за минимально короткое время. Алгоритм горячего резервирования выполняет сравнение результатов работы этих ЭВМ в контрольных точках вычислительного процесса, что позволяет в случае несовпадения результатов определить факт отказа или сбоя.



ММВК с прямым управлением – это объединение нескольких микроЭВМ через общее ОЗУ (рис.2.14), причем осуществляется прямое управление совместным функционированием с помощью информационного обмена через общее ОЗУ или командное управление через контроллер процессоров всех ЭВМ комплекса. Наличие общего ОЗУ позволяет уменьшить время обнаружения неисправности по сравнению с управлением через общее ВЗУ, а использование контроллеров между процессорами – ускорить переключение на

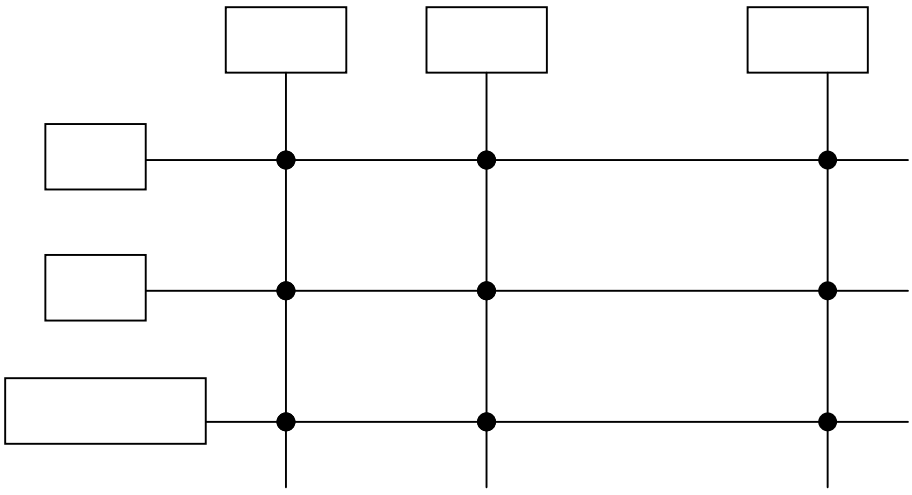
резервную ЭВМ, так как управление идет на уровне команд исполняемой программы.

Сателлитные ММВК используют один из рассмотренных выше способов объединения ЭВМ в иерархическую структуру, в которой одна ЭВМ является ведущей (host), а другие – вспомогательными и подчиненными ей, причем вспомогательные ЭВМ ведут простейшую предварительную обработку информации или управление обменом с внешними устройствами (ВУ) выводом через контроллеры ввода-вывода (КВВ).

Многопроцессорный ВК

Многопроцессорный ВК – это многопроцессорная машина, в которой все микропроцессоры управляются общей ОС комплекса, причем для каждого микропроцессора может быть предусмотрена отдельная копия ОС. По типу и организации связей между микропроцессорами различают МПВК с общей шиной, с коммутационной матрицей и с многовходовым ОЗУ.

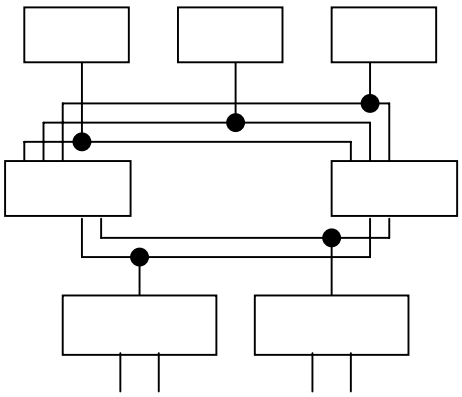
Рассмотрим *МПВК с общей шиной*. Один из недостатков такой структуры – низкое быстродействие, причиной которого являются конфликты между отдельными МП, использующими ресурс общей шины. Другой недостаток – низкая надежность, поскольку при выходе из строя какого-либо элемента нарушается работа всей шины. Для повышения надежности вводят вторую – резервную шину.



Структурная схема *МПВК с коммутационной матрицей* изображена на рис.2.15.

Коммутационная матрица (КМ) представляет из себя множество шин, на пересечении

которых располагаются БИС, выполняющие коммутационные функции (коммутационные узлы). По сравнению со структурой с общей шиной, структура с КМ позволяет снизить влияние конфликтов из-за использования шин, повысить надежность и быстродействие. Влияние конфликтов из-за использования ресурсов шин снижается, поскольку уменьшается вероятность одновременного обращения двух устройств к одной шине. Повышается надежность, так как для каждой пары устройств имеется свой узел связи. Повышение производительности достигается как за счет одновременной работы многих узлов КМ, так и за счет высокого быстродействия аппаратуры.



Однако в случае *МПВК с коммутационной матрицей* остается конфликт из-за использования ресурсов ОЗУ.

Структурная схема МПВК с многовходовым ОЗУ (Рис. 2.16) позволяет снизить влияние конфликтов из-за использования ресурсов ОЗУ и дополнительно организовать защиту информации.

Организация вычислительного процесса в МПВК

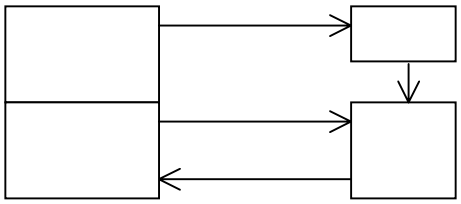
Независимо от архитектуры возможны три способа организации вычислительного процесса в МПВК:

1. С использованием выделенного ведущего процессора. Ведущий процессор распределяет задачи (программы) между остальными – ведомыми, организует передачу информации между процессами (программами), перераспределяет ресурсы и задания при выходе из строя аппаратуры.

2. С раздельным выполнением заданий в каждом процессоре. В этом случае в процессе выполнения задачи каждый процессор может выполнять как функции ведущего процессора, так и функции обработки. Распределение ресурсов и заданий выполняет пользователь.

4. Однородная обработка на основе очередей. Предварительного распределения заданий по ресурсам нет, но есть очередь заданий. Все процессоры одинаковы, каждый освободившийся процессор выбирает очередное задание.

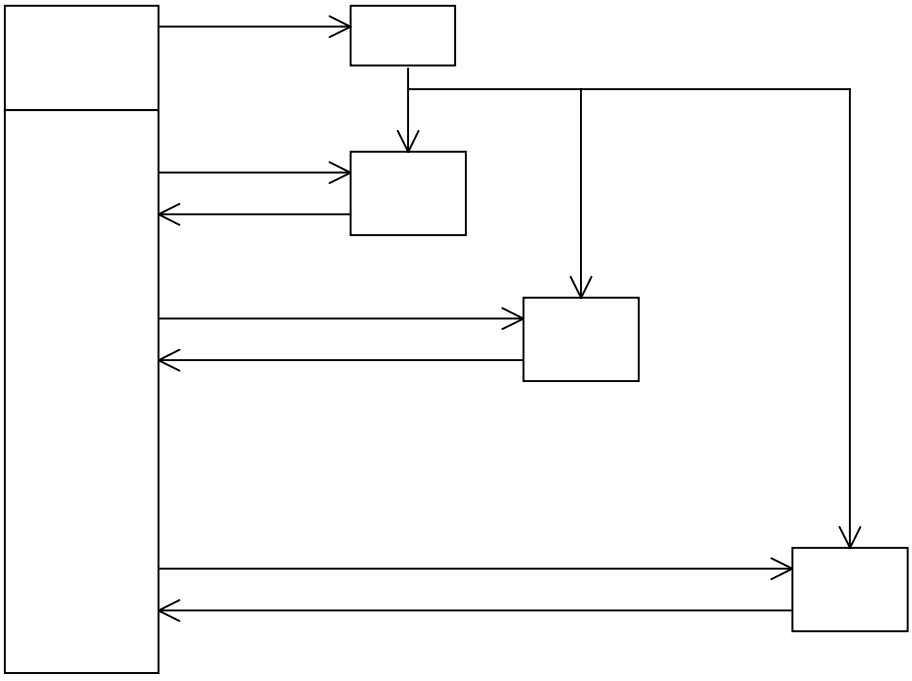
Классификация МПВС по кратности потоков команд и данных



Традиционной является классификация МПВС по числу потоков команд и числу потоков данных, обеспечивающих параллельное вычисление.

Системы с однократным потоком команд и однократным потоком данных (ОКОД) – это классическая структура (рис.2.17) однопроцессорной ЭВМ (Single Instruction Single Data – SISD).

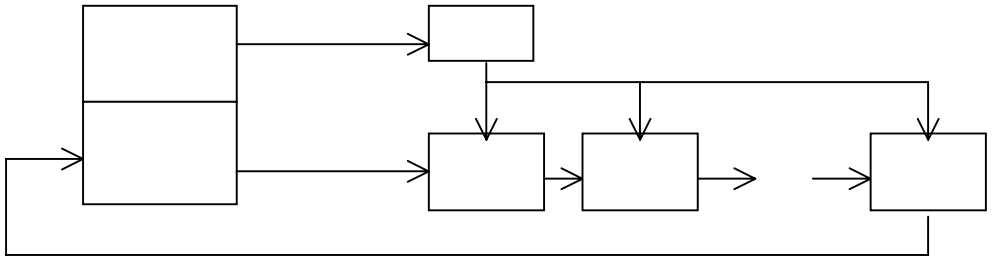
Системы с многократным потоком команд и однократным потоком данных (МКОД) – это магистральные (конвейерные) МПВС (Multiple Instruction Single Data – MISD). В этом случае АЛУ неоднородные. Такой способ называется конвейерным, потому что АЛУ одновременно выполняют разные операции над последовательным потоком



обрабатываемых данных (рис. 2.18).

Системы с однократным потоком команд и многократным потоком данных (ОКМД**)** **–** это векторные МПВС, в которых все процессоры

одновременно выполняют одну команду (рис.2.19) над различными данными (Single Instruction Multiple Data – SIMD).



Системы с многократным потоком команд и многократным потоком данных (МКМД) – это матричные МПВС, в которых микропроцессоры одновременно выполняют разные операции над несколькими потоками обрабатываемых данных (Multiple Instruction Multiple Data – MIMD). Построение систем, реализующих принцип МКМД в чистом виде – выполнение множества параллельных процессов над множеством данных, возможно только при использовании весьма быстрых программируемых аппаратных средств типа транспьютера.

В зависимости от способа реализации устройств обработки и взаимосвязи между процессорами можно говорить о многопроцессорных системах двух уровней:

- системы с малым (2-100) количеством процессоров;
- многопроцессорные системы ($100 – 2^{16}$).

Глава 3
ВВОД-ВЫВОД В МИКРОПРОЦЕССОРНЫХ СИСТЕМАХ

3.1. Принципы организации ввода-вывода в микропроцессорной системе

Архитектуру подсистемы ввода-вывода характеризуют режимы работы устройств ввода-вывода, форматы команд, скорость обмена, особенности прерываний и другие параметры [2, 4, 5, 9, 15]. Архитектура подсистемы ввода-вывода оказывает непосредственное влияние на эффективность всей системы и в большой степени определяет выбор серийной ЭВМ для конкретного применения. В операционных системах ЭВМ, как правило, имеется набор специальных подпрограмм, называемых драйверами ввода-вывода, которые скрывают от пользователя многие особенности и детали ВУ и интерфейсов ввода-вывода. Тем не менее разработка аппаратных средств и программного обеспечения ввода-вывода является наиболее сложным этапом проектирования управляющих ЭВМ.

Организация ввода-вывода подчиняется двум основным принципам, характеризующим действия, выполняемые на системной магистрали микроЭВМ.

1. При взаимодействии любых двух устройств микроЭВМ одно из них (чаще всего процессор) обязательно играет активную роль и является задатчиком, т.е. выполняет управляющую функцию, а второе оказывается управляемым, исполнителем.

2. В структуру интерфейса для взаимодействия любых двух устройств микроЭВМ заложен принцип квитирования (запроса-ответа): каждый управляющий сигнал, посланный задатчиком, должен быть подтвержден сигналом исполнителя. При отсутствии ответного сигнала от исполнителя в течение заданного интервала времени задатчик формирует сигнал ошибки обмена (тайм-аут) и прекращает операцию.

Режимы ввода-вывода в микроЭВМ можно классифицировать следующим образом:

1. Программно-управляемый ВВ (синхронный, асинхронный и ВВ по прерываниям);

2. Прямой доступ к памяти.

Программно-управляемый ввод-вывод

Программно-управляемый ввод-вывод (называемый также неформированным) характеризуется тем, что инициирование и управление ВВом осуществляется прикладной программой, выполняемой процессором, а внешние устройства играют сравнительно пассивную роль и сигнализируют только о своем состоянии, например о готовности к операциям ввода-вывода.

Наиболее просто осуществляется *синхронный (безусловный)* программно-управляемый ввод-вывод для внешних устройств, не требующих проверки готовности, таких, например, как индикатор на светодиодах. Для этого достаточно в соответствующем месте программы использовать соответствующие команды, например IN или OUT.

Однако большинство ВУ требуют проверки их готовности к обмену до выполнения операций ввода-вывода, т.е являются *асинхронными.* Состояние готовности/занятости таких устройств характеризуется либо отдельными флагами READY и BUSY, либо одним флагом готовности READY.

Для проверки флага готовности затрачивается одна или несколько команд процессора. Если флаг установлен, то инициируются собственно ввод или вывод одного или нескольких слов данных. Когда же флаг сброшен, процессор выполняет так называемый цикл ожидания готовности ВУ, т.е. цикл повторной проверки флага READY до тех пор, пока устройство не будет готово к операциям ввода-вывода.

Программно-управляемый асинхронный ВВ не требует дополнительных аппаратных средств и его просто реализовать, однако он приводит к непроизводительными потерями времени процессора в циклах ожидания готовности ВУ.

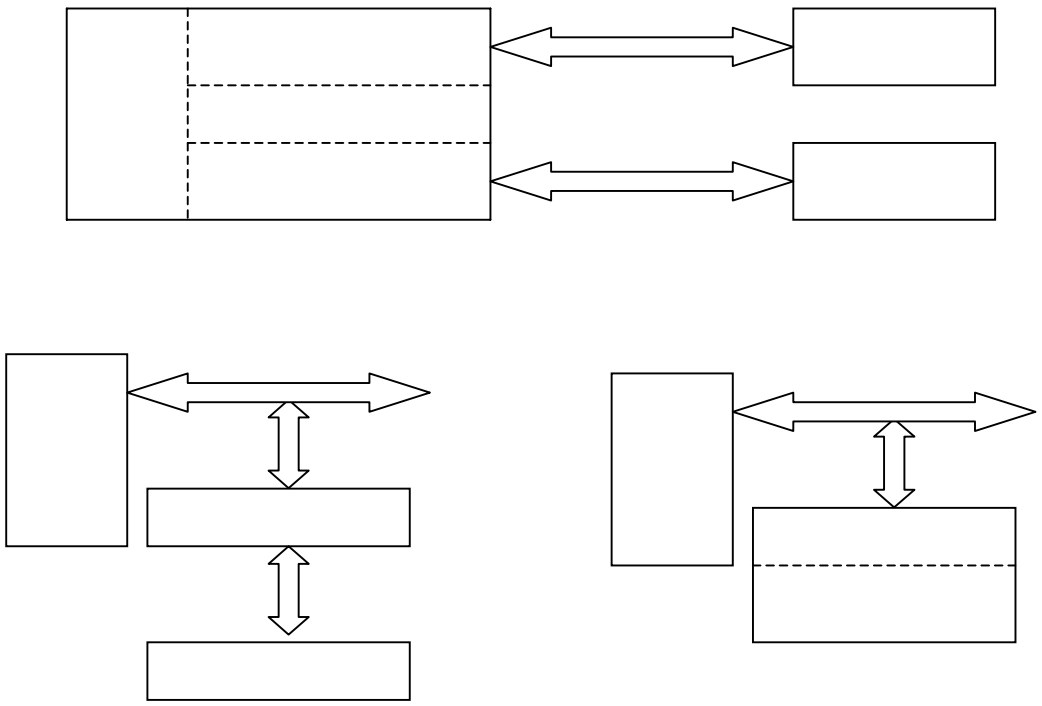
Ввод-вывод по прерываниям (называемый также *форсированным*) инициируется не процессором, а внешним устройством, генерирующим специальный сигнал прерывания, чтобы сообщить о своей готовности к передаче данных. Типичным примером являются сигналы прерывания, возникающие при нажатии клавиш или манипуляциях с мышью при работе на ПЭВМ. Реагируя на сигнал прерывания, процессор передает управление подпрограмме прерывания для обслуживания устройства, вызвавшего это прерывание. Действия подпрограммы прерывания определяются пользователем, а непосредственными операциями ввода-вывода в процессе обслуживания прерывания управляет процессор.

Прямой доступ к памяти

Прямой доступ к памяти используется для непосредственного обмена данными между основной памятью и быстродействующим ВУ, когда пропускной способности процессора недостаточно. В этом режиме действия процессора приостанавливаются, он отключается от системной шины и не участвует в передачах данных между памятью и ВУ.

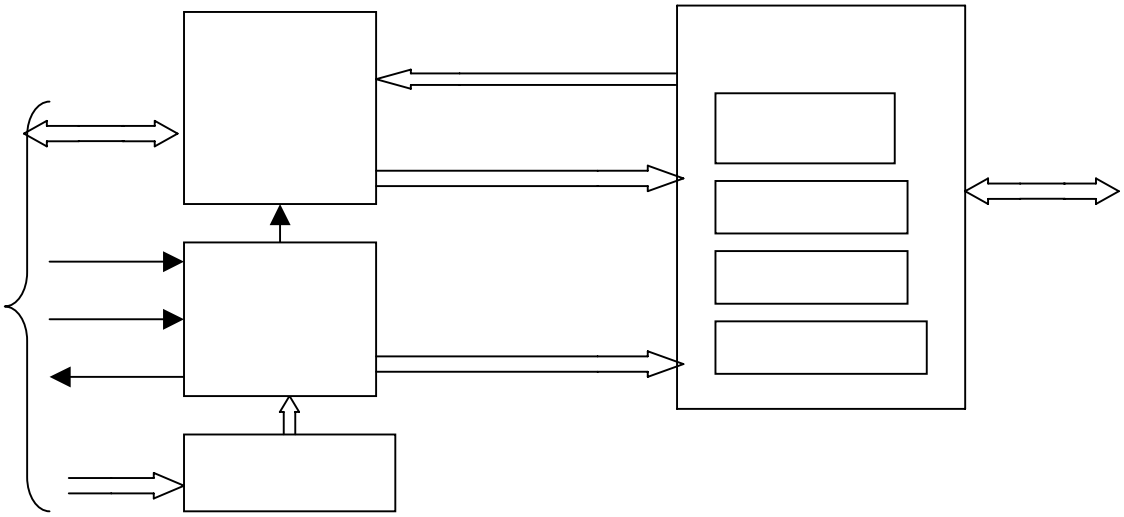
3.2. Контроллеры ввода-вывода

Для сопряжения микроЭВМ с ВУ используются *контроллеры*, или адаптеры. Возможны три варианта размещения контроллеров (рис. 3.1): а) вместе с процессором ; б) в виде отдельного устройства; в) непосредственно в ВУ.



В процессе ввода-вывода передаются данные двух видов: управляющие данные (слова) и собственно данные, или данные-сообщения. Управляющие данные от процессора (командные слова, или приказы) инициируют запуск

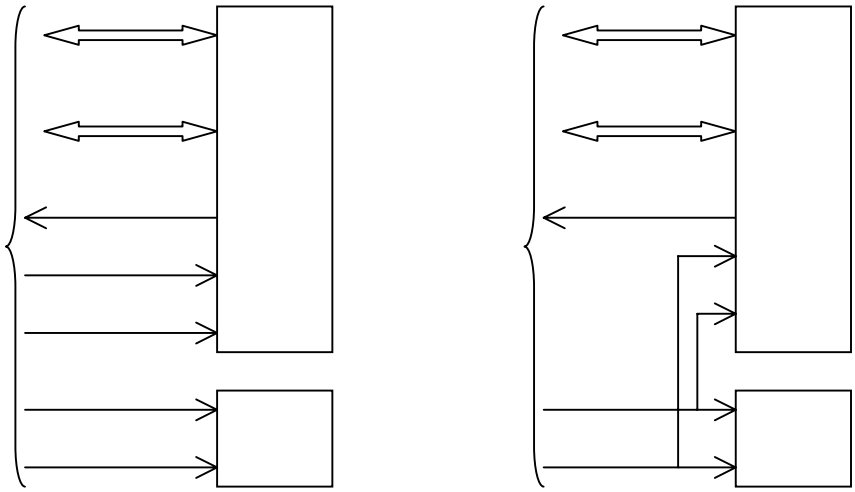
устройств, запрещение прерываний и другие действия, не связанные непосредственно с передачей данных. Управляющие данные от внешних устройств (словами состояния) несут информацию об определенных признаках состояния ВУ, например о готовности устройства к передаче данных, о возникновении ошибок при обмене и т.п. Для каждого признака состояния обычно выделяется один бит. Регистр, к которому процессор обращается в операциях ввода-вывода, образует порт ввода-вывода.



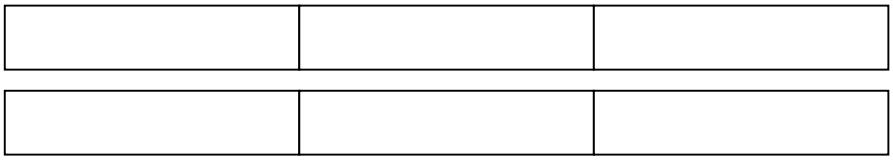
В общем случае контроллер внешнего устройства должен иметь четыре регистра ввода-вывода: регистр выходных данных (выходной порт), регистр входных данных (входной порт), регистр управления и регистр состояния (рис. 3.2). В конкретных случаях регистры состояния и управления могут быть объединены в один регистр, а для ввода и вывода использован двунаправленный порт. В любом случае каждый регистр контроллера ВУ должен иметь свой однозначный адрес, который расшифровывается дешифратором адреса.

Сопряжение регистров контроллера с шинами адреса и данных системного интерфейса обеспечивают соответственно приемники шины адреса и приемопередатчики шины данных. В МПС используются два варианта адресации регистров ввода-вывода ВУ и соответственно схем сопряжения контроллера ВУ с системным интерфейсом МПС (рис. 3.3):

- адресное пространство ВВ отделено от адресного пространства ОЗУ и ОП (использование специальных команд ВВ);
- общее адресное пространство ВВ, ОЗУ и ОП (адресация по аналогии с обращением к ОЗУ и ПЗУ).



На рис.3.4 представлен общий формат специальных команд ввода-вывода, которые позволяют адресовать регистры ввода-вывода ВУ отдельно и независимо от адресация памяти (ОЗУ и ПЗУ). Такая адресация ВВ используется, в частности, в МП КР580ИК80А.



Когда выполняется команда ввода (IN Port), содержимое адресуемого входного регистра Port передается во внутренний регистр Reg процессора, а при выполнении команды OUT Port содержимое регистра Reg передается в выходной порт Port. Кроме команд IN Port и OUT Port для организации ВВ могут потребоваться и другие команды, связанные с проверкой и модификацией содержимого регистров управления и состояния контроллера ВВ.

В случае адресации ВВ на основе специальных команд один и тот же адрес могут иметь порт ввода-вывода и ячейка памяти, поскольку адресное пространство портов ввода и вывода изолировано от адресного пространства памяти. Для разделения адресных пространств используются управляющие сигналы (рис. 3.3):

- “Чтение” (MEMRD) – считывание данных из памяти;

- “Запись” (MEMWR) – запись данных в память;
- “Ввод из ВУ” (IORD) – чтение порта ввода-вывода;
- “Вывод в ВУ” (IOWR) – запись в порт ввода-вывода).

В случае общего адресного пространства для памяти и ВВ признаком, дифференцирующим обращения к памяти и портам ввода-вывода, может быть старший бит адреса. Например, если адресное пространство памяти составляет 64 Кбайт, а для хранения программ и данных достаточно 32 Кбайт, то область адресов от 0 до 32 К-1 можно использовать для адресации ОЗУ и ПЗУ, а область от 32 К до 64 К-1 – для ввода-вывода.

При этом признаком, дифференцирующим обращения к памяти и портам ввода-вывода, может быть старший бит адреса. Таким образом, интерфейс с общими шинами (ввод-вывод с отображением на память) имеет организацию, при которой часть общего адресного пространства отводится для внешних устройств, регистры которых адресуются так же, как и ячейки памяти. В этом случае для адресации портов ввода-вывода (рис. 3.3) используются полные адресные сигналы: “Чтение” (READ), “Запись” (WRITE).

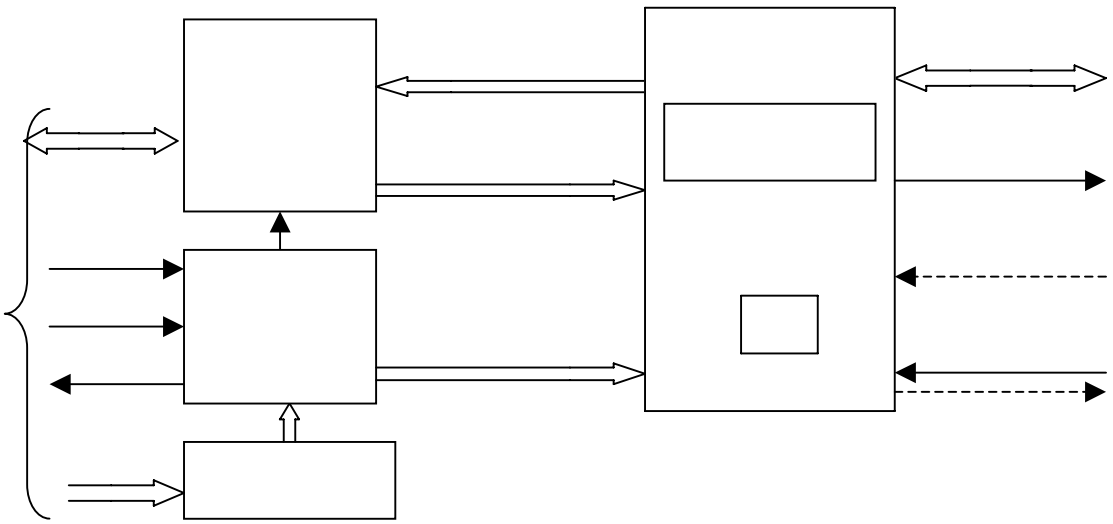
3.3. Способы и форматы передачи данных

Существуют два способа передачи данных по линиям интерфейса: параллельный и последовательный. Параллельная передача данных между контроллером и ВУ является по своей организации наиболее простым способом обмена.

Параллельная передача данных

Для организации параллельной передачи данных необходима шина данных, количество линий в которой равно числу одновременно передаваемых битов данных, причем в простейшем случае достаточно всего двух управляющих сигналов как при выводе , так и при вводе. Рабочее расстояние для шины *параллельной передачи данных* ограничивается, как правило, длиной 1-2 м, поскольку между отдельными параллельно расположенными проводниками существует электрическая емкость, что вызывает взаимные помехи между линиями шины. С увеличением длины шины помехи возрастают, и только за счет существенного удорожания шины или снижения скорости передачи длину шины можно увеличить до 10–20 м.

Простая функциональная схема контроллера на рис.3.5 для параллельного ВВ отличается от общей схемы на рис.3.2 тем, что в ней можно использовать один адресуемый регистр для входных и выходных данных (порт А1) и одноразрядный адресуемый регистр состояния и управления (регистр А2).



Рассмотрим как такой простой контроллер осуществляет вывод данных на ВУ. Регистр А2 служит для формирования в шине связи контроллера с ВУ управляющего сигнала "Выходные данные готовы" и приема из ВУ управляющего сигнала "Данные приняты". Очередной байт данных с шины данных системного интерфейса записывается в порт А1 и одновременно в регистр А2 записывается логическая единица. Тем самым формируется управляющий сигнал "Выходные данные готовы" в шине связи с ВУ.

ВУ, приняв байт данных, управляющим сигналом "Данные приняты" обнуляет регистр А2. При этом вырабатывается управляющий сигнал системного интерфейса "Готовность ВУ", а в соответствии с программой асинхронного обмена признак готовности ВУ к обмену передается в процессор по соответствующей линии шины данных системного интерфейса посредством стандартной операции ввода. Логика управления контроллера принимает управляющие сигналы системного интерфейса, выполняет адресную селекцию регистров контроллера и вырабатывает управляющий сигнал системного интерфейса "Готовность ВУ". Таким образом, алгоритм асинхронного параллельного вывода данных можно представить следующим образом:

1. Процессор микроЭВМ проверяет готовность ВУ к приему данных.

2. Если ВУ готово к приему данных (логический 0 в регистре А2), то данные передаются с шины данных системного интерфейса в регистр данных А1 контроллера и далее в ВУ. Иначе повторяется п. 1.

Для побайтного приема данных из ВУ простой контроллер параллельного интерфейса при взаимодействии с внешним устройством использует также только два управляющих сигнала (штриховые стрелки "Данные от ВУ готовы" и "Данные приняты" на рис.3.5) в соответствии с алгоритмом асинхронного ввода:

1. Процессор проверяет наличие данных в регистре данных контроллера А1.

2. Если данные готовы (логическая 1 в регистре А2), то они передаются из регистра данных А1 на шину данных системного интерфейса и далее в регистр процессора или ячейку памяти микроЭВМ. Иначе повторяется п. 1.

Схема параллельной передачи обеспечивает довольно высокую скорость обмена данными, поскольку для приема или передачи одного байта данных процессору необходимо выполнить всего несколько команд. Практически скорость обмена при параллельной передаче ограничивается только быстродействием ВУ.

Последовательная передача данных

Широкое распространение получил последовательный способ дистанционной передачи данных между ВУ и микроЭВМ, особенно в тех случаях, когда не требуется высокой скорости обмена. Такой способ также широко применяется для построения компьютерных сетей. Последовательные линии связи просты по своей организации, поскольку требуется всего два провода при симплексной и полудуплексной передаче и максимум четыре – при дуплексной. В современных микроЭВМ применяют, как правило, универсальные контроллеры для последовательного ввода-вывода, обеспечивающие как синхронный, так и асинхронный режим обмена данными с ВУ.

Для реализации последовательного режима обмена данными с внешними устройствами контроллер ВУ должен выполнять дополнительные функции по сравнению с контроллерами для параллельного обмена:

- при передаче в ВУ необходимо выполнять преобразование данных из параллельного формата, в котором они поступают в контроллер ВУ из системного интерфейса микроЭВМ, в последовательный и обратное преобразование при приеме данных из ВУ;
- поскольку возможны два режима последовательной передачи данных – синхронный и асинхронный, необходимо обеспечить способ обмена данными, соответствующий режиму работы внешнего устройства.

Синхронный последовательный интерфейс

При синхронной передаче каждый передаваемый бит сопровождается импульсом синхронизации, информирующим приемник о наличии на линии информационного бита. В этом случае передатчик и приемник должны быть соединены как минимум тремя проводами, два из которых используются для передачи бит данных и импульсов синхронизации, а один – как общий заземленный проводник.

Функциональная схема, изображенная на рис.3.2, в случае простого контроллера для синхронного обмена данными с ВУ по последовательной линии связи (последовательный интерфейс) должна иметь следующие регистры:

- сдвиговый регистр для преобразования данных из параллельного формата в последовательный при передаче в ВУ или для обратного преобразования при приеме данных из ВУ;
- счетчик импульсов (числа тактов преобразования);
- восьмиразрядный адресуемый буферный регистр контроллера A1 служит для временного хранения байта данных до его загрузки в сдвиговый регистр;
- одноразрядный адресуемый регистр состояния и управления контроллера A2.

Кроме того, для работы синхронного последовательного интерфейса требуется генератор тактовых импульсов.

Рассмотрим в качестве примера режим последовательной синхронной передачи. В буферный регистр контроллера A1 из системного интерфейса поступают данные в параллельном формате. Их преобразование в последовательный формат и передача на линию связи производятся в сдвиговом регистре с помощью генератора тактовых импульсов и трехразрядного счетчика импульсов.

Каждый тактовый импульс перемещает содержимое сдвигового регистра на один разряд вправо и посылает значение очередного разряда в линию связи "Данные". Одновременно со сдвигом по отдельной линии "Синхронизация" в ВУ передается тактовый импульс, благодаря чему каждый передаваемый по линии "Данные" бит сопровождается синхронизирующим сигналом по линии "Синхронизация". Тем самым обеспечивает однозначное восприятие данных на приемном конце последовательной линии связи.

В тот момент, когда содержимое счетчика становится равным 7, вырабатывается управляющий сигнал "Загрузка", обеспечивающий запись в сдвиговый регистр очередного байта из буферного регистра R1. Этот же управляющий сигнал устанавливает в "1" регистр состояния R2. Очередной тактовый импульс сбрасывает счетчик в "0", и начинается очередной цикл выдачи восьми битов информации из сдвигового регистра в линию связи.

Синхронная последовательная передача битов данных в линию связи должна производиться без какого-либо перерыва, и следующий байт данных должен быть загружен в буферный регистр R1 из системного интерфейса за время, не превышающее времени передачи восьми битов в последовательную линию связи. При записи байта данных в буферный регистр обнуляется регистр R2 контроллера. Нуль в регистре R2 указывает, что в линию связи передается байт данных из сдвигового регистра, а следующий передаваемый байт данных загружен в сдвиговый регистр.

Синхронная последовательная передача используется для пересылки массивов машинных слов. В начале передачи передатчик посылает в линию связи один или два символа синхронизации. Получив этот символ (символы), приемник переходит в режим приема и преобразования данных в параллельный формат.

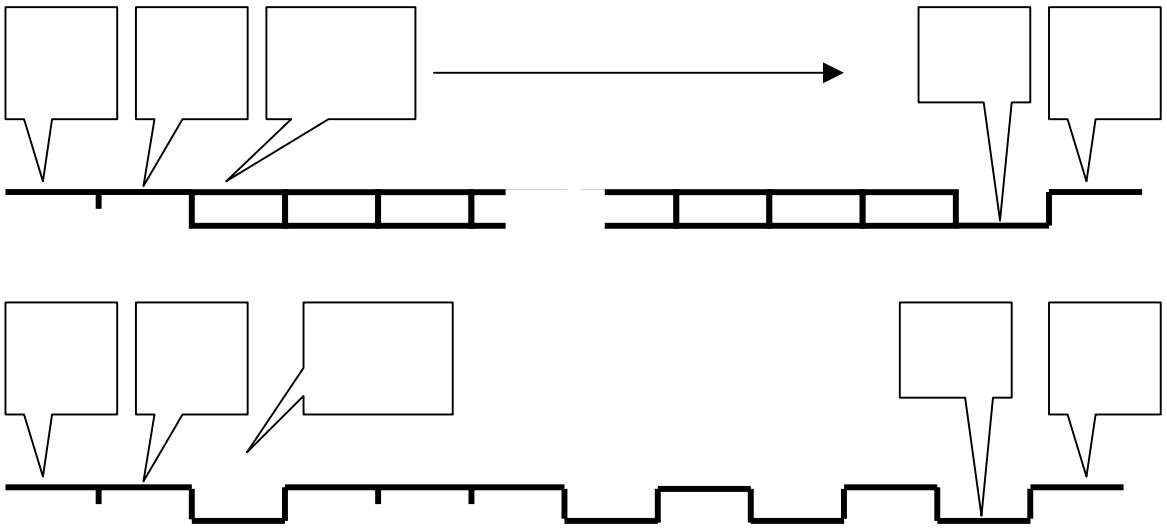
При большой удаленности микроЭВМ и ВУ (на несколько метров) необходимо защитить передаваемые сигналы от помех, поэтому приходится использовать экранированный кабель (коаксиальный или витую пару).

Асинхронный последовательный интерфейс

Асинхронная последовательная передача данных предполагает, что не используются синхронизирующие сигналы и, следовательно, у передатчика и приемника нет общего генератора синхроимпульсов. Однако передатчик и приемник имеют каждый свой генератор синхроимпульсов, которые должны быть согласованы по частоте и всем параметрам формата передачи.

Для того, чтобы приемник мог узнавать о моментах начала и завершения передачи последовательности бит данных, представляющих передаваемый символ, стандартный формат асинхронной последовательной передачи, используемый в ЭВМ и ВУ и называемый *кадром*, содержит 3-4 дополнительных бита кроме *n* пересылаемых бит информации. Обычно *n* равно 7 или 8 битам, причем дополнительные биты – это стартовый бит, один или два стоповых бита и, в некоторых случаях, бит контроля четности. Длительность передачи одного символа равна длительности кадра, т.е. промежутку времени от начала стартового бита до конца стопового бита (стоповых бит) соответствующего кадра.

Если нет пересылки данных (передатчик бездействует), на линии сохраняется уровень сигнала, соответствующий логической 1 (рис. 3.6,а).



Передатчик начинает пересылку символа посредством генерирования стартового бита, т.е. переводит линию в состояние логического 0 на время τ, точно равное времени передачи бита. После этого передатчик посылает последовательность всех битов передаваемого символа, начиная с самого младшего, за которой следуют дополнительный бит контроля четности и стоповый бит, переводящий линию в состояние логической 1 (рис. 3.6,б). Для того, чтобы наилучшим образом защитить принимаемый сигнал от воздействия помех, разброса частоты синхроимпульсов и искажений формы, приемник должен считывать принимаемый бит в середине его длительности.

Рассмотрим, каким образом приемник обнаруживает стартовый бит очередного кадра. Пока нет пересылки, на линии сохраняется сигнал 1. Если линия переходит в состояние логического нуля на время равное или

превышающее $\tau/2$ (половину интервала передачи бита), то приемник начинает вырабатывать сигналы считывания через интервалы, равные τ, т.е. выполняет считывание и сохранение принимаемых бит примерно на середине их передачи. Если же нет пересылки и линия переходит в состояние логического нуля на время меньшее $\tau/2$, то вероятнее всего это не появление стартового бита, а результат воздействия помехи, поэтому приемник остается в режиме обнаружения.

Завершая прием кадра, приемник считывает бит контроля четности и стоповый бит. Если на месте стопового бита обнаружен сигнал логического нуля, то это свидетельствует о том, что кадр принят неправильно ("ошибка кадра"). Если ошибки кадра не выявлены, выполняется проверка принятого символа на четность.

3.4. Организация прерываний в микроЭВМ

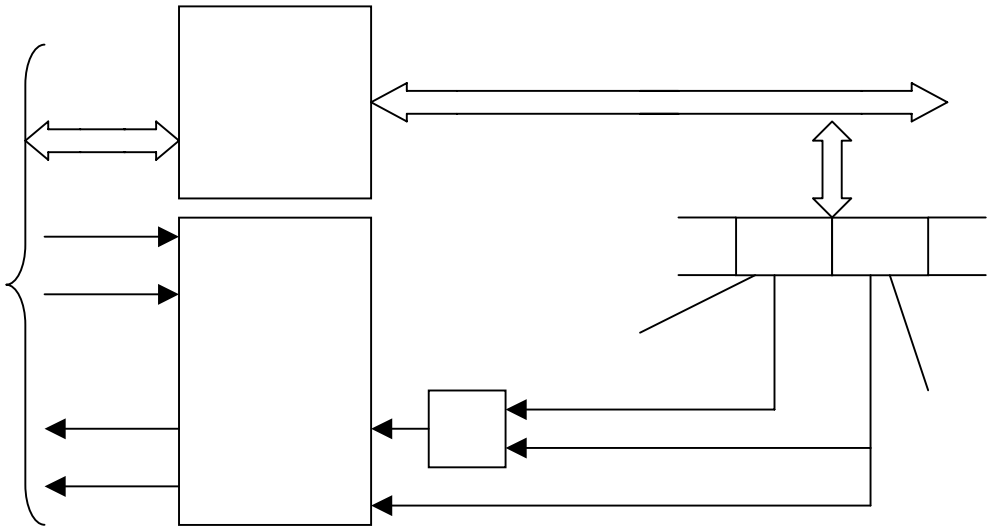
Организация обмена с прерыванием программы отличается от асинхронного программно-управляемого обмена тем, что переход к подпрограмме обработки прерываний, которая задает последовательность машинных операций по обмену данными, инициируется управляющим сигналом ("Запрос на прерывание" или "Требование прерывания"), поступающим из ВУ. Этот переход выполняется с помощью специальных аппаратных средств, задачей которых является приостановка выполнения процессором основной программы и передача управления подпрограмме обработки прерывания.

Таким образом, обмен данными с прерыванием программы может выполняться в реальном масштабе времени, а именно в произвольные моменты времени, определяемые только внешней по отношению к микроЭВМ средой и независящие от выполняемой основной программы. Поскольку исчезает необходимость в организации программных циклов ожидания готовности ВУ (особенно медленнодействующих), сокращаются связанные с обменом затраты процессорного времени.

Для выполнения подпрограммы обработки прерывания используются аппаратные ресурсы процессора. В момент прерывания в таких регистрах процессора, как счетчик команд, регистр состояния и другие, содержатся данные, связанные с выполнением основной программы. Их необходимо сохранить для последующего возврата в прерванную программу.

Сохранение содержимого счетчика команд и регистра состояния процессора, как правило, выполняется аппаратными средствами обработки прерывания, в то время как сохранение содержимого других регистров процессора, используемых подпрограммой обработки прерывания, производится непосредственно в этой подпрограмме. Если требуется сохранение большого объема данных и желательно минимизировать время реакции микроЭВМ на сигнал прерывания, то предпочтение может быть отдано аппаратной реализации операций сохранения данных основной программы.

Инициализация запросов ВУ на обслуживание (формирование сигналов прерываний) происходит в контроллерах ВУ. Наиболее простое решение заключается в том, что в качестве сигнала прерывания используется сигнал "Готовность ВУ", поступающий из контроллера ВУ в системный интерфейс микроЭВМ. Однако такое решение существенно усложняет организацию обмена данными в режиме прерывания с несколькими ВУ, поскольку процессор не имеет возможности управлять прерываниями, а именно разрешать или запрещать их для отдельных ВУ.



Проблему управления прерываниями можно решить следующим образом: регистр состояния и управления контроллера ВУ (рис. 3.7) дополняется разрядом "Разрешение прерывания". Это дает возможность, используя одну из линий шины данных системного интерфейса, осуществлять запись 1 или 0 в разряд "Разрешение прерывания" программным путем. Из схемы на рис.3.7 видно, что управляющий сигнал системного интерфейса

"Запрос на прерывание" формируется с помощью схемы совпадения только при наличии единиц в разрядах "Готовность ВУ" и "Разрешение прерывания" регистра состояния и управления контроллера.

Для решения проблемы управления прерываниями в микроЭВМ на системном уровне, в регистре состояния процессора выделяется разряд, значение которого, устанавливаемое программным путем, определяет разрешение или запрет прерывания от внешних устройств.

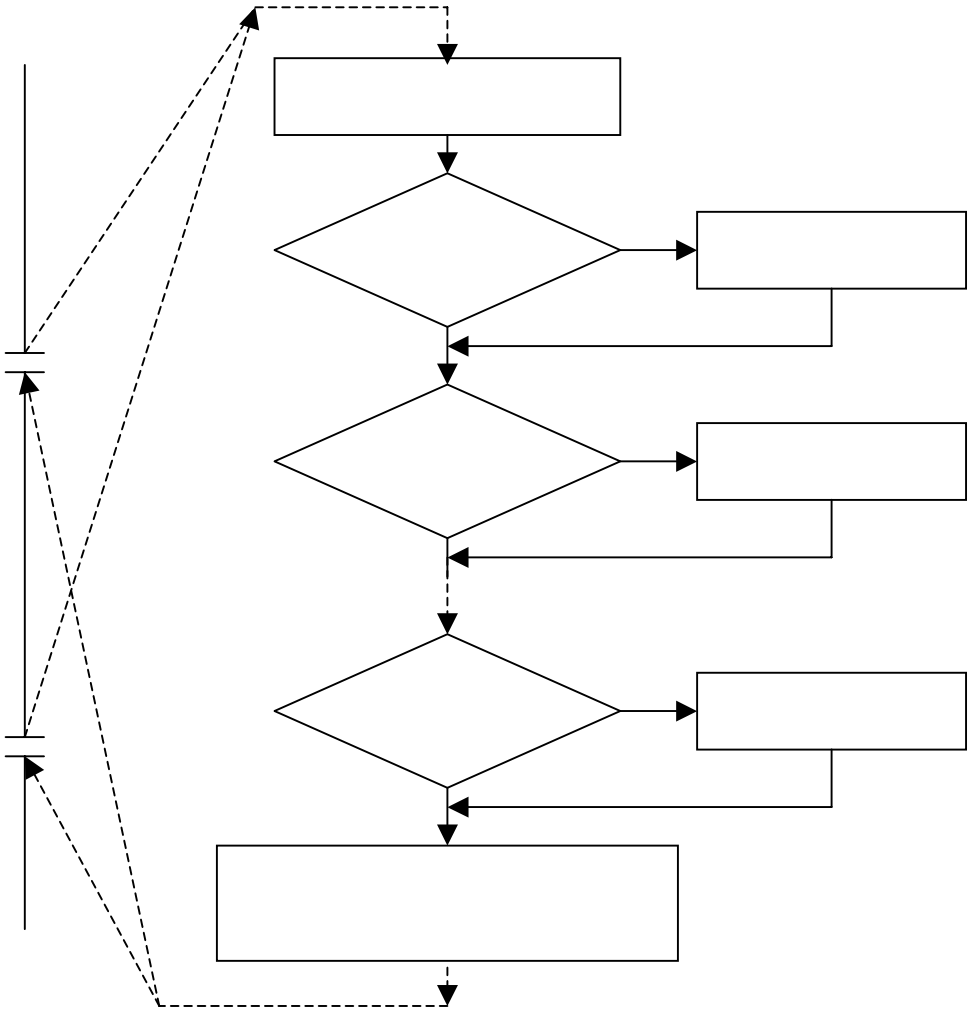
Если в микроЭВМ используется одноуровневая система прерываний, то сигналы "Запрос на прерывание" поступают от всех ВУ на один вход процессора. При этом возникает проблема определения ВУ, которое запросило обслуживание, и осуществление заданной очередности с учетом приоритета обслуживания в случае одновременного поступления сигналов прерывания от нескольких ВУ. Возможны два способа обслуживания ВУ, требующих обслуживания:

- на основе программного опроса разряда "Готовность ВУ" в регистрах состояния контроллеров всех ВУ;
- на основе использования векторов прерывания.

Организация прерываний с программным опросом готовности ВУ

На рис. 3.8 приведена структура подпрограммы опроса готовности для обслуживания прерываний от всех внешних устройств. В конце последнего машинного цикла выполнения очередной команды основной программы процессор проверяет наличие сигнала "Запрос на прерывание". Если прерывание разрешено и имеется сигнал прерывания, то процессор переключается на выполнение подпрограммы обработки прерываний.

Эта подпрограмма работает следующим образом. После сохранения содержимого регистров процессора, используемых в подпрограмме, выполняется последовательный опрос регистров состояния контроллеров всех ВУ, работающих в режиме прерывания. Если подпрограмма находит готовое к обмену ВУ, то сразу выполняются действия по его обслуживанию. Подпрограмма обработки прерывания завершает свою работу после опроса готовности всех ВУ и восстановления содержимого регистров процессора.



Чем раньше подпрограмма опрашивает готовность ВУ, тем меньше время реакции на соответствующий запрос. Следовательно, при организации обмена с программным опросом готовности внешних устройств их приоритет однозначно определяется порядком опроса ВУ, заложенным в подпрограмме обработки прерываний.

Основной недостаток организации обмена с программным опросом готовности внешних устройств – сложность выполнения жестких требований на время обработки сигналов прерывания. Это связано со сравнительно большим временем обслуживания тех устройств, которые опрашиваются последними.

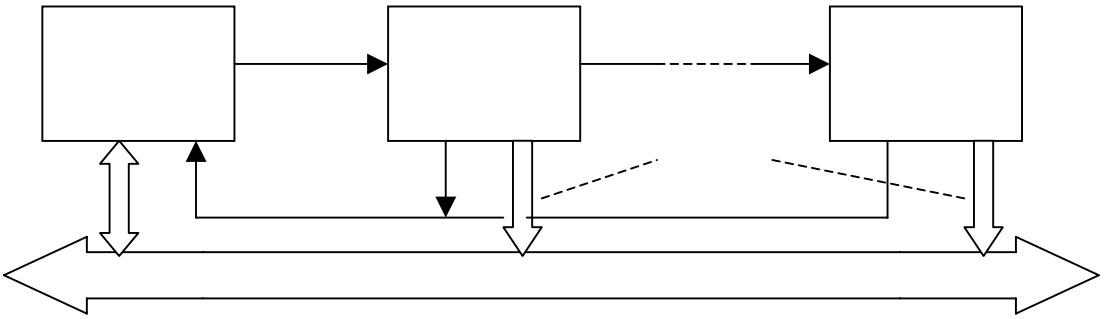
Организация обмена с использованием векторов прерываний

Организация обмена с ВУ на основе использования векторов прерываний позволяет устранить указанный выше недостаток способа с программным опросом готовности внешних устройств. Предполагается, что каждое ВУ должно иметь собственную подпрограмму обработки прерывания. Вектор прерывания – это адрес ячейки основной памяти, в которой хранится либо первая команда, либо адрес начала подпрограммы обслуживания прерывания данного ВУ.

Сущность обмена с использованием векторов прерываний заключается в том, что устройство, которому требуется обслуживание, инициирует посредством контроллера формирование и передачу в процессор вектора прерывания, что позволяет сразу переключаться на выполнение требуемой подпрограммы обработки прерывания. Существуют системы с формированием вектора прерываний в контроллерах ВУ и системы с формированием вектора прерываний в общем программируемом контроллере прерываний.

Формированием вектора прерываний в контроллерах ВУ

В этом случае вектор прерывания формируется контроллером ВУ, запросившим обслуживание. Для того, чтобы исключить одновременную выдачу векторов прерывания от нескольких ВУ, вектор прерывания выдается контроллером только по разрешению процессора (рис. 3.9).



Для реализации заданных приоритетов обслуживания прерываний организуется последовательный аппаратный опрос готовности ВУ. Это осуществляется следующим образом. Получив по общей линии системного

интерфейса "Запрос на прерывание", процессор формирует управляющий сигнал "Предоставление прерывания (вх.)", который поступает сначала в контроллер ВУ с наивысшим приоритетом. Если это устройство не требовало обслуживания, то его контроллер пропускает сигнал "Предоставление прерывания" на следующий контроллер, иначе дальнейшее распространение сигнала прекращается и контроллер выдает вектор прерывания на адресно-информационную шину.

Для того, чтобы пользователь мог устанавливать требуемые значения векторов прерывания для конкретных ВУ, регистр вектора прерывания обычно выполняется в виде набора перемычек или переключателей.

Поскольку аппаратный опрос готовности ВУ производится гораздо быстрее, нежели программный, системы с интерфейсным формированием вектора прерываний реагируют быстрее на запросы, чем системы с программным опросом готовности устройств. Как и в системах прерывания с программным опросом, если обслуживания запрашивают одновременно два или более ВУ, обслуживание менее приоритетных ВУ откладывается на время обслуживания более приоритетных.

Формирование вектора прерываний в программируемом контроллере

В этом случае контроллеры внешних устройств не имеют регистров для хранения векторов прерывания, а для выбора устройства, запросившего обслуживание и имеющего наивысший приоритет, используется общий для всех ВУ программируемый контроллер прерываний. Такой контроллер позволяет посредством соответствующих команд задавать различные варианты дисциплин многоуровневого обслуживания прерываний.

В качестве примера, можно назвать БИС программируемого контроллера прерываний (ПКП) КР580ВН59. Это устройство реализует до восьми уровней запросов на прерывания с возможностью программного маскирования и изменения порядка обслуживания (циклического сдвига приоритетов) прерываний. За счет каскадного включения БИС ПКП число уровней прерывания может быть расширено до 64. Векторная система с внеинтерфейсным вектором прерывания используется в IBM-совместимых персональных компьютерах.

3.5. Организация прямого доступа к памяти

В режиме прямого доступа к памяти (ПДП) обмен данными между ВУ и основной памятью микроЭВМ происходит без участия процессора в соответствии с микропрограммой, заложенной в электронные схемы ПДП.

Поскольку режим ПДП может обеспечить время обмена одним байтом данных между памятью и ВЗУ, равное циклу обращения к памяти, он позволяет использовать в микроЭВМ быстродействующие внешние запоминающие устройства такие, например, как накопители на жестких магнитных дисках.

В целях сокращения количества линий в шинах микроЭВМ контроллер ПДП подключается к памяти посредством шин адреса и данных системного интерфейса. Для решения проблемы совместного использования ресурса системного интерфейса процессором и контроллером ПДП используют два подхода: реализация обмена в режиме ПДП с "захватом цикла" и в режиме ПДП с блокировкой процессора.

Наиболее простой способ организации ПДП с "захватом цикла" заключается в использовании машинных циклов процессора, в которых он не обменивается данными с памятью. Применение этого способа организации ПДП не снижает производительности микроЭВМ, но при этом возникает необходимость выделения таких циклов, чтобы не произошло временного перекрытия обмена ПДП с операциями обмена, инициируемыми процессором. Возможности использования рассматриваемого способа ограничены, поскольку обмен возможен только одиночными байтами или словами.

Поэтому бóльшее распространение получил ПДП с блокировкой процессора (принудительным отключением от шин системного интерфейса). Для реализации этого режима необходимо дополнить системный интерфейс двумя линиями для передачи управляющих сигналов: "Требование прямого доступа к памяти" (ТПДП) и "Предоставление прямого доступа к памяти" (ППДП).

Контроллером ПДП формирует управляющий сигнал ТПДП. Получив этот сигнал и не дожидаясь завершения очередной команды, процессор приостанавливает работу, выдает на системный интерфейс управляющий сигнал ППДП и отключается от шин системного интерфейса. С этого момента все шины системного интерфейса управляются контроллером ПДП. Используя шины системного интерфейса, контроллер осуществляет обмен одним байтом или словом данных с памятью микроЭВМ и затем, сняв сигнал ТПДП, возвращает управление системным интерфейсом процессору. Как только

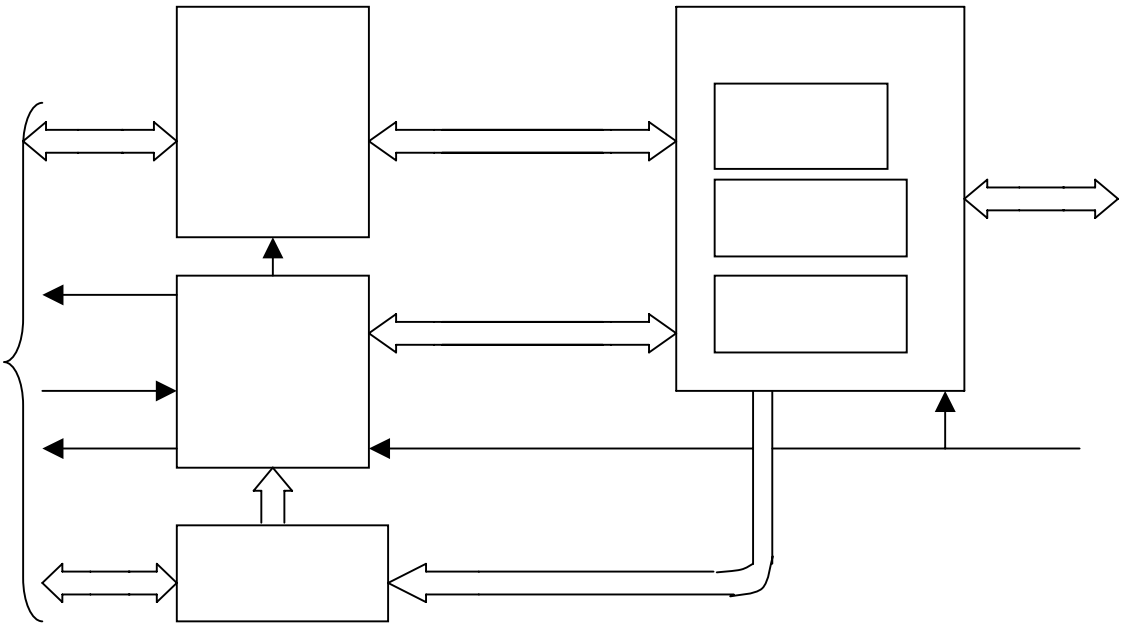
контроллер ПДП снова готов к обмену, захват цикла процессора и передача очередного байта повторяется и т.д.

Поскольку процессор простаивает в интервалы времени, когда интерфейс занят обменом в режиме ПДП, выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме прерываний.

Для организации режима ПДП необходимо выделить каждому ВУ область памяти, используемую при обмене, с указанием ее размера в байтах или словах, в зависимости от того, какими порциями данных ведется обмен. Контроллер ПДП должен иметь в своем составе регистр адреса и счетчик байт (слов).

При начальной загрузке программ в память в режиме ПДП содержимое регистра адреса и счетчика байт слов устанавливается перемычками или переключателями непосредственно на плате контроллера.

В остальных случаях перед началом обмена с ВУ процессор выполняет программу, которая обеспечивает запись начального адреса выделенной ВУ памяти и ее размера в указанные регистры контроллера ПДП.



Функциональная схема простого контроллера ПДП в режиме "Захват цикла" для ввода данных из ВУ в память микроЭВМ приведена на рис. 3.10. Перед началом очередного сеанса ввода процессор загружает, как уже было сказано выше, в счетчик байт – количество принимаемых байт данных, а в регистр адреса – начальный адрес области памяти для вводимых данных. Байты данных из ВУ поступают в регистр данных контроллера в постоянном темпе.

Каждый байт сопровождается управляющим сигналом из ВУ "Ввод данных", который обеспечивает запись байта данных в регистр данных контроллера, и при отличном от нуля состоянии счетчика байт обеспечивает формирование сигнала контроллера “Требование ПДП”. Получив ответный сигнал процессора “Предоставление ПДП”, контроллер выставляет на шины адреса и данных системного интерфейса содержимое своих регистров адреса и данных соответственно, а затем формирует управляющий сигнал "Вывод", обеспечивающий запись байта данных из регистра данных контроллера в память микроЭВМ. По каждому сигналу “Предоставление ПДП” из содержимого счетчика байт вычитается единица и изменяется содержимое регистра адреса. Как только содержимое счетчика становится равным нулю, контроллер прекращает формирование сигналов "Требование ПДП".

Любой сеанс обмена данными с ВУ в режиме ПДП инициируется программой, выполняемой процессором, и включает два следующих этапа.

1. Этап подготовки ВУ к очередному сеансу обмена. В режиме программно-управляемого обмена процессор опрашивает состояние ВУ и посылает в ВУ команды, обеспечивающие подготовку ВУ к обмену, загрузку регистров контроллера ПДП. Завершив подготовку к обмену в режиме ПДП, процессор переключается на выполнение другой программы.

2. Обмен данными в режиме ПДП начинается либо ВУ по инициативе (см. пример выше), либо процессора. В последнем случае используются регистры состояния и управления контроллера ПДП.

Глава 4
ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

4.1. Уровни абстрактного представления и этапы проектирования микропроцессорных систем

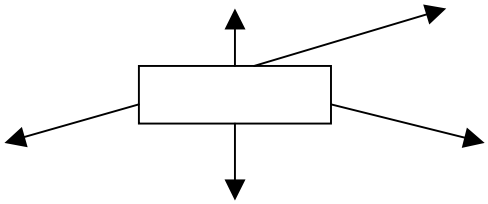
Проектирование МПС начинается с анализа и уточнения технического задания (ТЗ) на разработку МПС. В процессе разработки МПС происходит переход от одного уровня ее представления к другому, более детальному. На каждой стадии проектирования МПС описание полученных результатов (решений) характеризуется определенным уровнем абстракции. Основные уровни абстракции это:

1) концептуальный (уровень "черного ящика");

2) структурно-функциональный;
3) программный;
4) логический;
5) схемный;
6) конструкторско-технологический.

Уровням абстракции отвечают уровни структурного и параметрического описания разрабатываемой МПС. Каждый уровень несет в себе только информацию, относящуюся к данному уровню, и не содержит каких-либо сведений относительно более низких уровней.

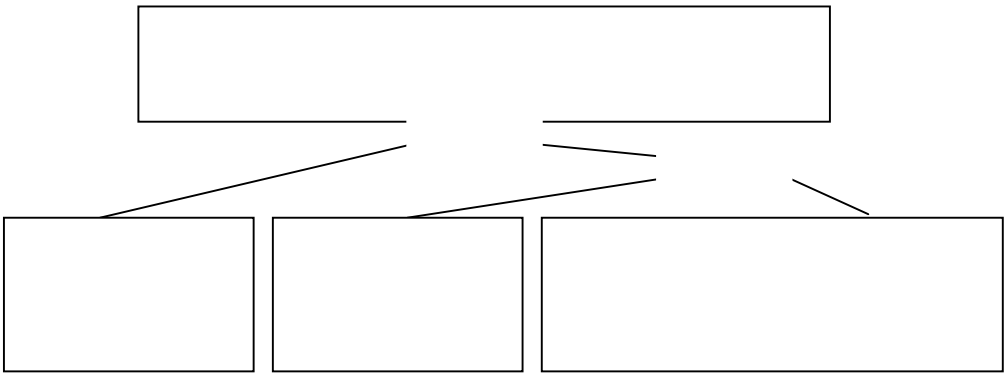
Процесс проектирования развертывается от неполноты описания к полноте описания, от абстракции к конкретике, от приближенного описания к точному, от альтернативных вариантов к оптимальному решению. Каждый уровень абстракции может иметь свои подуровни, соответствующие различной



степени детализации описания решений для соответствующего уровня.

Сочетание многовариантности (альтернативности) моделей и различных уровней их абстракции порождает все множество степеней свободы описания решений в задачах проектирования МПС (рис. 4.1).

При проектировании МПС часто используются уже имеющиеся апробированные решения для отдельных компонентов. Например, проектирование МПС, отличающейся от аналога только содержанием ПЗУ, будет состоять из разработки программ и изготовления ПЗУ. Таким образом, в общем случае разработка компонента *A* МПС заключается в создании ТЗ на разработку *A* с последующим выбором готового компонента *A* или разработкой структуры *S* компонента *A* и всех элементов структуры *S* (рис. 4.2).



Этапы и задачи проектирования МПС. Жизненный цикл МПС можно разделить на стадии проектирования, изготовления и эксплуатации, причем на каждой стадии можно выделить несколько фаз. Состав и содержание фаз, а также задачи проектирования зависят от назначения проектируемой МПС. Состав требований, например по надежности, габаритам и весу, к специализированным МПС может существенно отличаться от требований к МПС, ориентированным на широкий класс задач.

Приведем примерную последовательность этапов (горизонтальных уровней) создания МПС:

1. Системное проектирование и формализация требований к МПС.
2. Разработка структуры и архитектуры системы.
3. Разработка и изготовление аппаратных средств и программного обеспечения системы.
4. Комплексная отладка и приемосдаточные испытания.

Горизонтальные (иерархические) уровни отличаются степенью детализации описания объекта проектирования. На каждом горизонтальном уровне решаются группы задач, связанные с теми или иными сторонами (аспектами) объекта. Различают следующие аспекты:

- функциональный;
- алгоритмический;
- конструкторский;
- технологический.

Проектирование любой специализированной ЭВМ, в том числе на базе микропроцессорной техники, заключается в выборе оптимального

распределения функций между аппаратными и программными средствами, что требует от разработчика основательной подготовки в области схемотехники, структурной организации ЭВМ, принципов построения математического и программного обеспечения.

На этапе формализации требований составляются внешние спецификации, а именно, уточняются функции системы, формализуется техническое задание (ТЗ) на систему.

На этапе разработки структуры и архитектуры системы определяются функции отдельных устройств и программных средств, выбирается элементная база (микропроцессорный комплект) для реализации системы, временные параметры и порядок взаимодействия аппаратных и программных средств.

Разработка и изготовление аппаратных средств и программного обеспечения МПС выполняются, по-возможности, параллельно. Схемотехники проектируют структурные и принципиальные схемы, занимаются макетированием, изготовлением прототипа (опытного образца) и автономной отладкой. Разработчики математического обеспечения и программисты проектируют алгоритмы, тексты исходных программ, выполняют их трансляцию в объектные коды и автономную отладку.

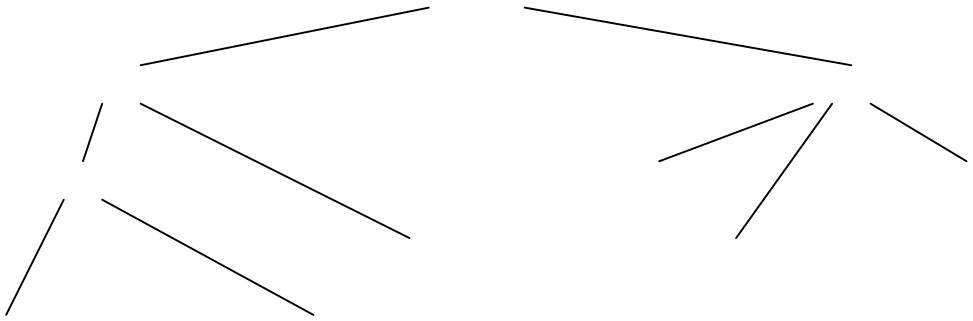
Комплексная отладка заключается в доводке рабочих программ и обеспечении правильного функционирования на реальной аппаратуре и в реальных условиях. Задача комплексной отладки состоит в выявлении ошибок проектирования и дефектов аппаратуры, которые могли быть не замечены на предшествующих стадиях разработки МПС.

На любой стадии жизненного цикла МПС имеется вероятность возникновения конструктивных или физических неисправностей, приводящих к появлению той или иной неисправности. В процессе функционирования МПС могут появляться ошибки, вызванные различными неисправностями: отказами аппаратуры, сбоями в работе и дефектами проектирования и изготовления. Поскольку качество проекта МПС существенно зависти от того, насколько система защищена от воздействия неисправностей, рассмотрим классификацию возможных неисправностей и их проявлений [1, 8, 11, 15].

Классификация неисправностей. Ошибка – это проявление неисправности, заключающееся в искажении обрабатываемых данных. Смысл термина "ошибка" зависит от уровня иерархической структуры МПС. Например, для дискретных устройств ошибка означает появление неверных

двоичных сигналов ("0" вместо "1" и "1" вместо "0"), в то время как для программы ошибка – это отклонение поведения программы от заданного, приводящее к выдаче неверных данных.

Неисправность может приводить или не приводить к ошибке в зависимости от состояния системы и/или наличия системы автоматического восстановления работоспособности. В то же время возникновение ошибки свидетельствует о наличии некоторой неисправности. Если одна и та же ошибка может быть вызвана множеством неисправностей, то одна неисправность может повлечь несколько ошибок. На рис.4.3 приведена классификация неисправностей, дефектов и ошибок.



Источники ошибок кроются в дефектах элементов, либо в неблагоприятном физическом воздействии окружающей среды.

Дефекты – отклонения физических параметров компонентов системы за допустимые пределы. Постоянные отказы – это дефекты, связанные с выработкой ресурса и проявляющиеся как внезапный или постепенный выход из строя компонентов. Для устранения постоянного отказа нужно обнаружить факт отказа, осуществить поиск отказавшего элемента и заменить его. Постоянные отказы характеризуются средним временем $T_{отк}$ наработки на отказ, а степень приспособленности МПС к обнаружению и устранению постоянных отказов, или ремонтопригодность, характеризуется средним временем $T_{рем}$ ремонта (восстановления).

Производственные дефекты – это неисправные комплектующие изделия, установленные в системе при изготовлении.

Физические неисправности, связанные с воздействием среды, – это перемежающиеся (кратковременные) отказы или сбои, приводящие к

искажению данных при передаче, обработке и хранении. Восстановление работоспособности МПС после сбоя заключается в восстановлении правильности данных, искаженных сбоем. Перемежающиеся отказы характеризуются средним временем $T_{сб}$ наработки на один сбой и кратностью l сбоев.

Причиной субъективных, или внесенных, неисправностей являются ошибки проектирования, неправильный монтаж элементов или неправильные действия оператора.

Проектные неисправности вызваны недостатками схем аппаратуры, программ и конструкций, а также средств разработки и эксплуатации – компиляторов, ассемблеров, программ автоматизации проектирования, инструкций по эксплуатации, процедур и средств контроля и т.п. Эти недостатки могут быть внесены в систему на различных стадиях разработки МПС при структурном проектировании, разработке алгоритмов, написании программ, трансляции в машинный код, детальном логическом и техническом проектировании, а также при последующих модификациях аппаратного и программного обеспечения.

Интерактивные неисправности возникают, если в процессе эксплуатации или технического обслуживания оператор вводит в систему данные, нарушающие правильное функционирование системы. Это может быть связано с неточным вводом данных или следствием непонимания инструкции для оператора.

Следует отметить, что субъективные неисправности отличаются от физических тем, что после обнаружения, локализации и коррекции больше не возникают.

4.2. Системное проектирование и формализация требований к микропроцессорным системам

В связи с тем, что особенности проектирования специализированных МПС связаны с выполнением структурного и системного этапов, именно эти этапы наиболее подробно рассмотрены в настоящем пособии.

В начальной стадии проектирования МПС можно описать только на концептуальном уровне (уровне "черного ящика"), на котором микропроцессорная система описывается внешними спецификациями (внешними характеристиками). Возможны различные варианты выбора состава внешних характеристик и численных показателей (параметров) для их оценки.

На системном уровне основные задачи проектирования связаны с выбором состава объекта проектирования на уровне укрупненных блоков – устройств, определения функций и алгоритмов взаимодействия устройств.

Особенность системного проектирования специализированных МПС на базе МПК заключается в том, что основной задачей этого этапа является анализ технического задания на проектирование и выбор МПК по основным критериям, в качестве которых обычно берут быстродействие и точность решения вычислительных задач.

В результате системного проектирования определяется один или несколько вариантов структуры МПС на уровне отдельных устройств (процессора, оперативного и постоянного ЗУ, устройств ввода-вывода и обмена информацией) и технические характеристики выбранных устройств.

На этапе системного проектирования следует использовать методы исследования операций, теории массового обслуживания и принятия решений в условиях многокритериальности.

Рассмотрим требования, а также показатели качества и основные характеристики МПС, предназначенные для решения задач контроля и управления: тактико-технические, конструктивно-технологические, эксплуатационные, экономические, надежностные.

Задачи, решаемые специализированными МПС, можно классифицировать по характеру алгоритмов:

- алгоритмы логико-программного управления;
- алгоритмы на основе численных методов.

Рассмотрим характеристики требований на решение задач. Потоки требований на решение задач можно классифицировать следующим образом:

- детерминированные потоки требований;
- случайные потоки требований.

Детерминированные потоки характеризуются тем, что требования поступают с заданным постоянным периодом $t_{ц.i}$, где i=1,...,N. Для каждого типа требований задается допустимое время обслуживания $t_{обсл.доп.i}$. Если значения $t_{ц.i}$ выражаются целыми числами, то суммарный график для N потоков требований характеризуется периодом T=НОК$\{t_{ц.1},...,t_{ц.N}\}$, где НОК – наименьшее общее кратное.

Случайные потоки требований характеризуются тем, что требования поступают в случайные моменты времени. Стационарный поток требований можно характеризовать функцией распределения случайных интервалов t между моментами поступления требований $\Phi(\tau)=P[t\leq\tau]$. Для анализа

характеристик обслуживания случайных потоков требований используется теория массового обслуживания и имитационное моделирование.

Показатели эффективности специализированных МПС

Временные и точностные характеристики задаются как допустимая длительность обслуживания требований на решение задач $t_{\text{обсл.}i.\text{доп}}$ и допустимая среднеквадратичная погрешность $\sigma_{i.\text{доп}}$ для решения задач каждого i-го типа.

Характеристики надежности включают интенсивность отказов (среднее число отказов в единицу времени) $\lambda_{\text{отк}}$, интенсивность сбоев (среднее число сбоев в единицу времени) $\lambda_{\text{сб}}$, коэффициент готовности $K_{\text{г}}=T_{\text{отк}}/(T_{\text{отк}}+T_{\text{рем}})$, где $T_{\text{рем}}$ – среднее время ремонта. Коэффициент готовности характеризует вероятность того, что ремонтируемая МПС находится в работоспособном состоянии.

Технико-экономические характеристики включают стоимость проектирования, установки и эксплуатации.

Перечислим *эксплуатационно-технические характеристики*:

- масса, габариты или объем ($см^3$);
- потребляемая мощность;
- диапазон температур, влажность;
- устойчивость к ударным и вибрационным перегрузкам;
- устойчивость к радиации.

Расчет разрядной сетки специализированной МПС

Расчет числа разрядов представления входных данных, операндов и выходных данных специализированной МПС выполняется из условия обеспечения заданной допустимой среднеквадратической погрешности $\sigma_{i.\text{доп}}$ решения задач каждого i-го типа.

Основные теоретические положения. Предполагается, что ЭВМ предназначена для вычислений системы функций

$$y_i = f_i(x_1, \ldots, x_n), \quad i = 1 \ldots m,$$

причем заданы пределы изменения для каждого аргумента:

$$a_j \le x_j \le b_j, \quad j = 1 \ldots n.$$

Цель расчета: определить тип АЦП (число разрядов $n_{\text{вх.}j}$, j=1…n, погрешность АЦП), разрядность операндов $n_{\text{оп.i,}}$ а также тип ЦАП (число разрядов $n_{\text{вых.}i}$,

i=1…m, погрешность ЦАП), при которых достигается минимум аппаратных затрат, при условии выполнения требований к погрешности вычислений

$$\sigma_{р.i} \le \sigma_{доп.i},\ i=1\ldots m,$$

где $\sigma_{р.i}$, $\sigma_{доп.i}$ – фактическая и допустимая погрешность вычисления функции f_i, i=1…m.

Следует также учитывать, что разрядность операндов $n_{оп.i}$ существенно влияет на время выполнения арифметических операций, если требуемая разрядность достигается программным путем.

Учитывая большое число источников погрешности вычислений и случайный характер величины погрешности, создаваемой каждым источником, можно считать, что функция распределения результирующей погрешности имеет вид близкий к нормальному, а для оценки этой погрешности использовать среднеквадратическое значение, обозначаемое как $\sigma_{р.i}$. Тогда максимальное значение $\Delta_{р.i}$. погрешности вычисления функции f_i с вероятностью 0,997 не превосходит $3\sigma_{р.i}$.

Среднеквадратическую погрешность вычисления f_i можно оценить по формуле

$$\sigma_{р.i}^2 = \sigma_{м.i}^2 + \sigma_{и.i}^2 + \sigma_{т.i}^2 + \sigma_{цап.i}^2 + \sigma_{дин.i}^2,\ i=1\ldots m,$$

где $\sigma_{м.i}$ – методическая погрешность, обусловленная приближенным характером алгоритма и численного метода, реализующего алгоритм вычисления f_i;

$\sigma_{и.i}$ – инструментальная погрешность, обусловленная машинными округлениями в процессе выполнения арифметических операций;

$\sigma_{т.i}$ – трансформированная погрешность, т.е. вклад в общую погрешность $\sigma_{р.i}^2$ за счет трансформации функциональной зависимостью $y_i = f_i(x_1, \ldots, x_n)$ погрешностей получения и представления исходных данных (погрешности датчиков и АЦП);

$\sigma_{цап.i}$ – погрешность цифро-аналогового преобразования результатов вычисления y_i;

$\sigma_{дин.i}$ – динамическая погрешность, обусловленная квантованием входных данных по времени и запаздыванием выдачи результатов на величину времени счета y_i.

Для приближенной оценки инструментальной погрешности используем формулу

$$\sigma_{и.i}^2 = \sigma_{окр.i}^2 + \sigma_{вых.i}^2,$$

где $\sigma_{окр.i}$ – погрешность округления; $\sigma_{вых.i}$ – погрешность представления данных на выходе цифровой части ЭВМ.

Погрешность симметричного округления оценивается по формуле

$$\sigma_{окр.i} = w_{оп.i}(l_{окр.i}/12)^{1/2},$$

а для округления усечением по формуле

$$\sigma_{окр.i} = w_{оп.i}(l_{окр.i}/3)^{1/2},$$

где

$l_{окр.i}$ – число округлений;

$w_{оп.i}$ – вес младшего разряда операндов в единицах выходной величины y_i, определяемый по формуле

$$w_{оп.i} = |y_i|_{\max}/(2^{n_{оп.i}} - 1),$$

причем разрядность $n_{оп.i}$ операндов определяется разрядностью арифметико-логического устройства микропроцессора и выбранным вариантом программной реализации арифметических операций.

Погрешность представления выходных данных $\sigma_{вых.i}$ зависит от разрядности ЦАП, т.е. разрядности выходных данных $n_{вых.i}$. Для округления усечением

$$\sigma_{вых.i} = \begin{cases} 0 & \text{при } n_{вых.i} \geq n_{оп.i}; \\ w_{вых.i}/\sqrt{3} & \text{при } n_{вых.i} < n_{оп.i}. \end{cases}$$

Для симметричного округления

$$\sigma_{выхi} = \begin{cases} 0 & \text{при } n_{вых.i} \geq n_{оп.i}; \\ w_{выхi}/\sqrt{12} & \text{при } n_{вых.i} < n_{оп.i}; \end{cases}$$

Здесь $w_{вых.i} = |y_i|_{\max}/(2^{n_{вых.i}} - 1)$.

Трансформированная погрешность оценивается по формуле

$$\sigma^2_{T.i} = \sum_{j=1}^{n} M^2[d_{ij}] \cdot \sigma^2_j,$$

где σ_j – погрешность получения и представления аргумента x_j;

$M[d_{ij}]$ – математическое ожидание значения частной производной $d_{ij} = \partial f_i / \partial x_j$ (если $f_i(x_1,\ldots,x_n)$ дифференцируемая).

Формула для вычисления $\sigma_{T.i}$ записана в предложении, что погрешности аргументов x_j, j=1…n, являются случайными независимыми величинами. Эти погрешности оцениваются формулой

$$\sigma_j^2 = \sigma_{дат.j}^2 + \sigma_{АЦП.j}^2 ,$$

где $\sigma_{дат.j}$ – погрешность датчика и нормирующего усилителя;

$\sigma_{АЦП.j}$ – погрешность аналого-цифрового преобразователя.

Погрешность АЦП

$$\sigma_j^2 = \sigma_{дат.j}^2 + \sigma_{АЦП.j}^2 ,$$

где $\sigma_{A.j}$ – инструментальная погрешность АЦП, связанная с нестабильностью параметров аналоговой части АЦП;

$\sigma_{вх.j}$ – погрешность квантования по уровню, которая зависит от числа разрядов $n_{вх.j}$ АЦП:

$$\sigma_{вх.j}^2 = w_{вх.j} / \sqrt{12} ,$$

где

$$w_{вх.j} = (b_j - a_j) / (2^{n_{вх.j}} - 1) .$$

Для оценки $M[d_{ij}]$ необходимо знать плотность вероятности $\varphi(x_1,\ldots,x_n)$ распределения набора входных аргументов $\boldsymbol{x}=(x_1,\ldots,x_n)$:

$$M\left[d_{ij}\right] = \int_{a_1}^{b_1} \ldots \int_{a_n}^{b_n} d_{ij}(x) \cdot \varphi(x) dx_1 \ldots dx_n .$$

Если случайные величины $x_1,\ldots,x_n$ статистически независимы и подчиняются равномерному закону распределения, то

$$\varphi(x_1,\ldots,x_n) = 1 / \prod_{j=1}^{n} (b_j - a_j)$$

и определение $M[d_{ij}]$ сводится к вычислению значения кратного интеграла

$$\int_{a_1}^{b_1} \ldots \int_{a_n}^{b_n} d_{ij}(x) \cdot \varphi(x) dx_1 \ldots dx_n .$$

В общем случае, когда вычисления по формуле (3) затруднительны, для нахождения $M[d_{ij}]$ следует использовать метод статистических испытаний (метод Монте-Карло). Для этого генерируется последовательность наборов входных величин $x^k = (x_1^k, ..., x_n^k)$, k=1…K, удовлетворяющая заданной плотности распределения $\varphi(x_1,\ldots,x_n)$. Оценка $M[d_{ij}]$ в этом случае вычисляется по формуле

$$M\left[d_{ij}\right] \approx (1/K)\sum_{k=1}^{K} d_{ij}(x_1^k, ..., x_n^k).$$

Расчет тактовой частоты микропроцессора

Время решения задачи определяется тактовой частотой процессора:

$$t_{\text{реш}.i} = (\Sigma_{j \in M}\, n_j\, q_j)/f,$$

где M – число типов операций процессора;

n_j – число операций j-го типа;

q_j – число тактов выполнения операции j-го типа;

f – тактовая частота микропроцессора.

Требуемая тактовая частота микропроцессора рассчитывается из условия обеспечения заданного времени обслуживания требований $t_{\text{обсл}.i} \le t_{\text{обсл}.i.\text{доп}}$. Поскольку $t_{\text{обсл}.i} = t_{\text{ож}.i} + t_{\text{реш}.i}$, где $t_{\text{ож}.i}$ – время ожидания решения задачи из-за занятости процессора решением других задач, получаем соотношение для расчета требуемой тактовой частоты для решения задач i-го типа

$$f \ge (\Sigma_{j \in M}\, n_j\, q_j)/(\, t_{\text{обсл}.i.\text{доп}} - t_{\text{ож}.i}).$$

4.3. Разработка архитектуры и структуры микропроцессорной системы

Структурно-функциональный уровень. Неформальные процедуры формирования функциональной структуры (ФС) проектируемой системы опираются на следующую информацию:

1) Варианты функционирования проектируемой системы (тактические ситуации использования).

2) Аналоги проектируемой системы или ее компонентов в виде раньше существовавших систем близкого назначения или их компонентов.

3) Аналоги проектируемой системы или ее компонентов в виде существующих систем близкого назначения или их компонентов.

4) Мыслимые (концептуальные) варианты построения системы или ее компонентов.

5) Жизненный цикл существовавших систем или этапы жизненного цикла существующих систем близкого назначения.

Для формализации множества альтернативных вариантов ФС системы в задачах структурного моделирования систем целесообразно использовать аппарат исчисления высказываний и логики предикатов, теории графов и формальных грамматик [13, 14].

Структурно-функциональный уровень описывает компоненты МПС (микропроцессоры, запоминающие устройства, устройства ввода/вывода, внешние запоминающие устройства, каналы связи), функции отдельных устройств, их взаимосвязь и информационные потоки.

В связи с тем, что особенности проектирования специализированных МПС связанны с выполнением структурного и системного этапов, именно эти этапы наиболее подробно рассмотрены в настоящем пособии.

Структурная схема МПС на уровне устройств отражает магистрально-модульный принцип построения систем на базе МПК и включает такие функциональные модули, как процессор, ПЗУ и ОЗУ, таймер и устройства ввода-вывода, а также магистрали (шины) для данных, адресации и управления.

Функциональная схема на уровне узлов и блоков раскрывает структуру отдельных устройств и детализирует связи между ними.

Для построения процессора используется, например, БИС микро-процессора, БИС генератора тактовых импульсов, а также БИС системного контроллера.

Для построения устройств ввода–вывода могут использоваться много-функциональные буферные регистры или БИС программируемого параллельного интерфейса.

Функцию таймера, обеспечивающего заданную периодичность решения задач, может обеспечить БИС программируемого таймера либо счетчик тактовых импульсов, емкость Q, которого рассчитывается по формуле $Q = T * f_r$, где T - заданный период, f_r - тактовая частота микроЭВМ.

Для ввода аналоговых сигналов может использоваться коммутатор аналоговых сигналов и один АЦП. Если коммутатор не используется, то число АЦП должно быть равно числу входных аналоговых сигналов.

Для вывода аналоговых сигналов может использоваться коммутатор двоичных кодов, один ЦАП и устройство запоминания аналоговых сигналов. Если коммутатор не используется, то число ЦАП равно числу выходных аналоговых сигналов.

4.4. Разработка аппаратных средств микропроцессорных систем

Разработка и изготовление аппаратных средств включает подэтапы:

- структурного проектирования аппаратных средств;
- логического проектирования аппаратных средств;
- конструкторско-технологического проектирования аппаратных средств.

На этапе структурного проектирования аппаратных средств определяется или уточняется функциональная структура микроЭВМ на уровне отдельных узлов (корпусов БИС; узлов, собираемых из компонентов средней и малой степени интеграции), а алгоритмическое описание – на уровне микропрограмм всех операций.

На этапе логического проектирования аппаратных средств решаются в основном задачи схемотехнического характера: логический синтез и электрический расчет узлов микроЭВМ на компонентах малой и средней степени интеграции, обеспечивающих согласование компонентов по нагрузочной способности, логическим уровнем и реализацию логических функций, не покрываемых компонентами стандартного МПК.

Последний, конструкторско-технологический этап связан с аспектами конструкции и технологии, хотя данные аспекты должны учитываться и на более ранних этапах проектирования.

Логический уровень можно разделить на *подуровень переключательных схем* и *подуровень микроопераций*.

Подуровень переключательных схем образуется логическими элементами и построенными на их основе операторами обработки данных. Переключательные схемы подразделяются на комбинационные (автоматы без памяти) и последовательностные (автоматы с памятью). Поведение системы на подуровне переключательных схем описывается автоматными моделями,

описывающими отображение входных последовательностей 1 и 0 в выходные последовательности.

Комбинационные схемы представляются формулами алгебры логики и таблицами истинности, в которых каждому входному набору значений сигналов ставится в соответствие набор значений сигналов на выходах комбинационной схемы [13, 14].

Последовательностные схемы могут описываться автоматными моделями Мили или Мура, в виде диаграмм или таблиц переходов/выходов, в которых определены взаимно однозначные соответствия между входами схемы, внутренними состояниями (комбинациями значений элементов памяти) и выходами [13, 14].

Подуровень *микроопераций* характеризуется более высокой степенью абстрагирования и представляет собой описание регистров и выполняемых ими *микроопераций,* включая передачу данных между ними. Он включает в себя две части: операционную и управляющую. Операционная часть состоит из регистров и связывающих их линий (шин). Управляющая часть вырабатывает последовательности управляющих сигналов, инициирующих выполнение микроопераций (суммирование и сдвиги кодов, пересылку данных между регистрами), в зависимости от значения сигналов, характеризующих состояние операционной части.

На схемном уровне поведение МПС описывается изменениями тока и напряжения в электрических схемах, состоящих из транзисторов, резисторов и конденсаторов. Этот уровень описания скорее характерен для гибридных компонентов МПС (аналого-цифровых и цифро-аналоговых преобразователей).

4.5. Разработка программного обеспечения микропроцессорных систем

Разработка алгоритмов и проектирование программного обеспечения МПС выполняются, как правило, параллельно с разработкой и изготовлением аппаратных средств. Разработчики математического обеспечения и программисты проектируют алгоритмы, тексты исходных программ, выполняют их трансляцию в объектные коды и автономную отладку.

Комплексная отладка заключается в доводке рабочих программ и обеспечении их правильного функционирования на реальной аппаратуре и в реальных условиях. Задача комплексной отладки состоит в выявлении ошибок

проектирования и дефектов аппаратуры, которые могли быть не замечены на предшествующих стадиях разработки МПС.

Программный уровень можно разделить *на подуровень команд процессора* и *подуровень языка МПС*. На программном уровне МПС рассматривается как последовательность операторов или команд, вызывающих то или иное действие над некоторым полем данных.

Надежность и качество ПО МПС

Надежность специализированных МПС зависит от надежности и качества всех компонентов, в том числе и программного обеспечения (ПО) [8, 11]. На всех этапах создания МПС должны выполняться оценки надежности и качества ПО. Практика показывает, что 100-процентная полнота тестирования для сложного ПО недостижима и, следовательно, какая-то доля дефектов остается в процессе всего жизненного цикла ПО [11]. Нарушения в работе ПО могут вызываться не только дефектами в программах, но и возможными сбоями и отказами аппаратуры, учитывая, что надежность контроля аппаратуры не является абсолютной и не абсолютна его полнота.

Возможны следующие виды дефектов:

- в структуре ПО и в распределении ресурсов;
- в спецификациях и исходных текстах;
- в тестах отладки и имитационных моделях;
- в программной документации;
- инструментальные, вторичные.

Стоимость исправления дефектов в ПО изменяется примерно квадратично в зависимости от этапов создания МПС. По сравнению со стадией ТЗ эта стоимость на этапе автономной отладки (АО) ПО увеличивается в 5 раз, на этапе комплексной отладки (КО) в 10 раз, а при эксплуатации МПС – в 100 раз [11]. Из этого следует, в какой большой мере надежность ПО и успех в его создании зависят от начальных этапов его проектирования.

В работе [11] приводятся формализованные методы оценки надежности и качества ПО, что обеспечивает возможность их нормирования. Комплексная оценка надежности МПС с учетом надежности ПО обеспечивается принятием дифференцированных нормативов аппаратной и программной надежности:

1. Качество ПО оценивается по отношению к качеству ПО-эталона. Относительное качество ПО характеризует степень соответствия его требованиям стандартов. Качественные показатели ПО закладываются в него на начальных этапах его проектирования, до начала программирования,

сопровождаясь при этом оценками его качества, позволяющими определить достаточность принятых показателей, отвечающих нормам качества.

2. За счет придания ПО свойств самоконтроля и самозащиты обеспечивается:

- дополнительный контроль системы, осуществляемый параллельно с программно-аппаратным контролем аппаратуры системы;
- существенно уменьшаются вероятности неконтролируемых аппаратных и программных отказов и сбоев системы, неопределенные ситуации в ней переводятся в разряд контролируемых;
- повышается надежность системы за счет использования дополнительного контроля в управлении (автоматическом и ручном).

Надежность ПО определяется вероятностью возникновения ошибок, связанных с дефектами ПО.

При оценке качества ПО учитываются показатели, способствующие устранению последствий неисправностей в аппаратуре и программах, а также ошибок оператора. Оценки качества ПО получаются достаточно полными, если определяется как мера возможных нарушений, так и эффективность средств противодействия им.

Для оценки качества ПО используются показатели защищенности от ошибок (уровень самоконтроля и защиты ПО от аппаратных и программных дефектов и ошибок).

Обеспечение надежности и качества ПО является доминирующей, ключевой задачей технологии программирования.

В процессе отладки и испытаний ПО осуществляется его верификация, т.е. подтверждение его корректности и отработанности, которая включает:

- оценку и обеспечение нормативного времени комплексной отладки (КО) и испытаний ПО в составе системы управления на стенде или объекте;
- оценку критерия достаточности отработки, тестирования ПО на стенде (перед поставкой на объект);
- проверку при испытаниях на стенде контролируемых показателей ПО, определяющих его надежность и качество;
- оценку и обеспечение требуемого времени сопровождения ПО при эксплуатации [11].

Надежность ПО встроенных систем соизмерима с надежностью их аппаратуры при условии, что автономная отладка и тестирование ПО при испытаниях в составе системы были достаточно полными. Надежность ПО может оказаться недостаточной, если:

• ПО недостаточно тестировалось и испытывалось из-за дефицита времени;

• недостаточным является качество ПО, характеризуемое показателями защищенности от ошибок.

При выдерживании нормативного времени отладки ПО обеспечивается достаточно полная его отработка. Свойства защищенности от ошибок закладываются в ПО на ранних стадиях проектирования, а в дальнейшем контролируются и обеспечиваются на всех последующих этапах создания МПС.

Надежность и качество ПО в значительной мере определяют безопасность системы управления на основе МПС, которая трактуется как свойство МПС выдавать достоверную информацию и обеспечивать защиту от недостоверной информации. При наличии средств аппаратной защиты система будет безопасной, если за счет реализации в ПО защитных функций будет обеспечено:

• безопасное поведение ПО, т.е. исключение его опасных отказов;

• качество ПО, отвечающее норме качества.

Согласно требованиям стандартов, в том числе международных, характеристика надежности ПО должна быть комплексной и должна включать основные показатели качества такие, как устойчивость, стабильность в работе и восстанавливаемость. Практикой установлено, что создание "лишь работающей" программы (в лабораторных условиях) недостаточно, необходима разработка надежных программ, отвечающих условиям их применения на объектах. Стиль *надежного программирования* предполагает дополнение ПО функциями самодиагностики, самозащиты и адаптации к программным и аппаратным ошибкам, обеспечение восстановления ПО при нарушениях в его работе.

Создание надежного ПО базируется на методах надежного программирования, получивших широкое применение в мировой практике. Повышение надежности ПО обеспечивается по двум направлениям:

1) повышением полноты и надежности алгоритмов;

2) за счет приемов и методов собственно программирования.

Большинство вышеприведенных свойств, характеризующих качество ПО, таких, как контроль входов/выходов, обходы ошибок и др., изначально должны обеспечиваться в алгоритмах. Без достаточно полного и безошибочного алгоритма не может быть надежной программы.

Алгоритмы должны отвечать требованиям:

• должны быть корректными, функционально полными;

- не должны приводить к тупиковым состояниям, зацикливаниям, переполнениям памяти и другим неразрешимым ситуациям;
- должны иметь избыточность, под которой понимается обеспечение ими защиты и адаптации к ошибкам вычислений, в частности к ошибкам по параметрам.

Защитные свойства, реализуемые способом обхода ошибок, должны быть отработаны алгоритмически. Так, при защите выходов от ошибки типа "несрабатывание", равнозначные избыточные (корректирующие) параметры или их группы должны объединяться по дизъюнкции. Наоборот, при предпочтительности защиты от "ложного" они должны объединяться по конъюнкции. Выполнение одновременно двух стратегий требует намного большей избыточности и может оказаться нецелесообразным из-за чрезмерного усложнения алгоритмов.

Надежность программы регулирования будет высокой, если косвенно будет проконтролирована хотя бы приемлемость выходов, например по корреляции входов/выходов. Соответственно в алгоритме должен быть описан способ такого контроля.

Должны быть отработаны в алгоритмах и другие решения по надежности, такие как защита от неопределенностей, проверки на непротиворечивость входов и выходов, контроль входов/выходов по корреляции, контроль исполнения команд по обратной связи и т.п. Для обеспечения полноты и корректности сложных логических алгоритмов следует стремиться к более широкому использованию формализованных методов, в частности методов автоматного управления. Данный подход позволяет:

- использовать для анализа и синтеза алгоритмов типовые модели (например, графы переходов и выходов);
- произвести декомпозицию сложных моделей управления (процессов, режимов) на простейшие с любой степенью разбиения (способом последовательной и параллельной декомпозиции);
- осуществить композиционный синтез алгоритмов, т. е. из частных описаний синтезировать алгоритмы всей системы;
- самодиагностировать процесс с выдачей ряда диагностических сообщений.

Возможность самодиагностики и коррекции управления – вот те достоинства и отличия, притом только в алгоритмической части, характерные для систем с микропроцессорами, не считая многих других возможностей, относящихся к реализации вышеуказанных способов повышения надежности и качества ПО, следовательно, и надежности систем.

Показатели надежности ПО МПС

Для описания надежности ПО МПС можно использовать комплекс показателей [11]:

- объемы оригинального ПО;
- продолжительность КО и испытаний ПО на стенде, объекте;
- наработка ПО на отказ;
- эксплуатационные показатели – ВБР (вероятность безотказной работы) и/или коэффициент готовности ПО (в зависимости от временных режимов каналов системы управления);
- время сопровождения ПО.

Надежность систем управления с ЭВМ равна произведению надежности аппаратной части системы и надежности ПО системы. Данное положение отвечает закону умножения вероятностей, поскольку исходы (сбои, отказы) в аппаратуре и ПО происходят независимо.

Принятием дифференцированных нормативов предусматривается ужесточение норм аппаратной и программной надежности, причем в такой достаточной мере, чтобы их произведение было не меньше общей, интегрированной нормы надежности систем управления. Например, если задана ВБР всей системы управления 0,995, произведение аппаратной и программной надежности 0,996 * 0,9993 = 0,9953 обеспечивает требуемую надежность системы с учетом надежности ПО.

При наличии дифференцированных норм надежности, выполнение отдельной комплексной оценки надежности МПС представляется необязательным, т.е. достаточным является представление раздельных расчетов надежности ПО и аппаратуры, сложившееся традиционно. Если расчеты удовлетворяют дифференцированным нормам, то они заведомо удовлетворяют и нормам на всю систему.

При реализации в ПО функций самоконтроля, обеспечивающих его качество, вероятность неконтролируемого отказа системы можно представить как произведение

$$P_{н}(t) = P(t)^* P_{к}(t)^* P_{п}(t),$$

где $P(t)$ – вероятность отказа аппаратуры СУ;

$P_{к}(t)$ – вероятность отказа программно-аппаратного контроля систем;

$P_{п}(t)$ – вероятность отказа самоконтроля ПО.

Вероятность $P_{п}(t)$ входит сомножителем в уравнение (23), что в значительной мере уменьшает результирующую вероятность $P_{н}(t)$ неконтролируемого отказа СУ, аналогичное влияние оказывает также интенсивность $\lambda_{п}$.

Расчетами установлено, что повышение качества ПО, реализация в нем функций самоконтроля и адаптация к аппаратным и программным ошибкам дают следующие результаты:

- значительно повышается надежность контроля за работой системы за заданный период работы - по ВБР с 0,9989 до 0,9999; по ВОС (вероятности отсутствия сбоя) с 0,9899 до 0,9998;
- вероятности неконтролируемого отказа и неконтролируемого сбоя системы (вместе с ПО и тоже за заданный период) уменьшаются, как минимум, на 2 порядка и выходят на уровень невероятного события (10^{-5} по существующей градации вероятностей).

При этом неконтролируемые ситуации в системе переводятся в разряд контролируемых, благодаря чему обеспечивается существенное повышение надежности системы (за счет использования дополнительного контроля в управлении).

Глава 5
ТЕСТИРОВАНИЕ И ОТЛАДКА МИКРОПРОЦЕССОРНЫХ СИСТЕМ

5.1. Принципы тестирования и отладки

Отладка – процесс обнаружения ошибок и определение источников их появления по результатам тестирования при проектировании МПС. Отладка МПС заключается в том, что на каждой стадии и фазе жизненного цикла МПС выполняются процедуры тестового контроля и диагностики, направленные на обнаружение и локализацию неисправностей [1, 8, 11, 15]. Процедуры тестового контроля и диагностики заключаются в проведении экспериментов с МПС как с "черным ящиком". Для установления правильности функционирования МПС или ее отдельных компонентов на уровне "черного ящика" с полностью неизвестной внутренней структурой необходимо провести испытания (эксперименты), в ходе которых на входы "черного ящика" подаются все возможные комбинации входных воздействий и проверяется корректность ответных реакций на его выходах. Тестовые входные воздействия

и ответные реакции определяются, исходя из спецификаций на устройства, а также структурных схем устройств.

Однако полное, исчерпывающее тестирование практически осуществимо только для простых компонентов. Для достаточно сложных реальных систем исчерпывающее тестирование неосуществимо. Остаточные дефекты проявляются при эксплуатации как ошибки проектирования и, следовательно, на любой стадии жизненного цикла достаточно сложных систем нельзя утверждать об отсутствии неисправностей. Эксперименты с "черным ящиком" предполагают использование той или иной гипотетической модели неисправностей, встречающихся на практике, и тестовых наборов, которые могли бы обеспечить удовлетворительное выявление моделируемых неисправностей.

Требуется правильный выбор соотношения между степенью общности модели, стоимостью и степенью сложности формирования и прогона тестов, построенных для моделируемых неисправностей.

Чем конкретнее *модель неисправностей системы*, тем легче создать для нее набор тестов, но тем выше вероятность того, что некоторые неисправности останутся незамеченными. Если же модель неисправностей излишне общая, то число необходимых тестовых наборов и/или времени вычислений, требуемого для формирования тестов, выражаются экспоненциальной зависимостью от сложности тестируемой системы и, следовательно, такая модель пригодна только для несложных систем.

Обнаружение ошибок и диагностика неисправностей

Дефект может быть обнаружен только в том случае, если созданы условия для возникновения из-за него неисправности и распространения ее результата до выхода испытуемого объекта. Последнее необходимо для того, чтобы сделать последствия неисправности наблюдаемыми. Метод испытаний как раз и предполагает генерацию тестов, создающих условия, при которых моделируемые неисправности проявляются в виде обнаруживаемых ошибок.

Если тестирование используется в процессе эксплуатации, то при обнаружении ошибки необходимо произвести локализацию неисправности с целью ее устранения путем ремонта или замены отказавшего компонента. Диагностика неисправности – процесс определения причины появления ошибки и ее локализации по результатам тестирования.

Разрешающая способность теста – это точность, с которой тест локализует неисправности. Требуемая разрешающая способность зависит от конкретных целей испытаний.

Эффективность отладки зависит от *контролепригодности системы, то есть от* того, какие в нее заложены свойства, делающие ее удобной для выполнения операций отладки, а также какие для этого предусмотрены средства.

Контролепригодность системы определяется ее свойствами управляемости, наблюдаемости и предсказуемости. Управляемость – свойство системы, позволяющее управлять ее поведением, например остановить функционирование системы в определенном состоянии, и затем снова ее запустить. *Наблюдаемость* – свойство системы, позволяющее проследить за поведением системы, сменой ее внутренних состояний. *Предсказуемость* – свойство системы, позволяющее установить систему в состояние, из которого все последующие состояния могут быть предсказаны.

Функции средств отладки. Сроки и качество отладки проектируемой МПС зависят от средств отладки. Средства отладки должны выполнять следующие функции:

1) управлять поведением системы и/или ее модели на различных уровнях абстрактного представления;

2) собирать данные о поведении системы и/или ее модели, обрабатывать и представлять их на различных уровнях абстракции;

3) преобразовывать систему для улучшения ее контролепригодности;

4) моделировать поведение внешней среды проектируемой системы.

Анализ источников ошибок

Рассмотрим источники ошибок на первых трех этапах проектирования.

На этапе формализации требований к системе источниками ошибок могут быть логическая несогласованность требований, упущения, неточности алгоритма.

На этапе разработки структуры и архитектуры системы ошибок могут быть упущения функций, несогласованность протоколов взаимодействия аппаратуры и программ, неверный выбор микропроцессорных наборов, неточности алгоритмов, неверная интерпретация технических требований, упущение некоторых информационных потоков.

На этапе разработки и изготовления аппаратных средств и программного обеспечения системы источниками ошибок могут возникать при разработке аппаратуры, при изготовлении прототипа и при разработке программных средств. При разработке аппаратуры возможны упущения некоторых функций, неверная интерпретация технических требований, недостатки в схемах синхронизации. При изготовлении прототипа возможны дефекты комплектующих изделий, а также ошибки монтажа и сборки. Наконец, при разработке программных средств возможны упущения некоторых функций технического задания, неточности алгоритмов и кодирования.

Таким образом, возможно большое число причин и источников субъективных и физических неисправностей, которые необходимо локализовать и устранить. Сложность задач обнаружения ошибок и локализации неисправностей связана также с тем, что различные неисправности могут проявляться одинаковым образом.

Достаточно хорошо отработаны методы и средства обнаружения ошибок и локализации физических неисправностей, которые широко используются для проверки работоспособности и диагностики неисправностей при проектировании, производстве и эксплуатации дискретных систем.

Однако отсутствуют модели субъективных неисправностей, что затрудняет задачу построения эффективных методов и средств для выявления таких неисправностей. Как следует из перечня источников ошибок, субъективные неисправности могут быть внесены уже на этапе разработки спецификации системы. Это означает, что субъективные неисправности могут оставаться даже после самых тщательных испытаний системы на соответствие ее внешним спецификациям.

Поскольку процесс проектирования МПС имеет итерационный характер, неисправности, обнаруженные на этапе приемосдаточных испытаний, могут привести к коррекции спецификаций, а следовательно, вернуть проектировщиков к началу проектирования всей системы. Поэтому необходимо контролировать корректность проекта на каждом этапе разработки, чтобы была возможность как можно раньше обнаруживать неисправности.

Проверка правильности проекта. Перечислим основные методы контроля правильности проектирования:

- *верификация* – формальные методы доказательства корректности проекта;
- моделирование;

• тестирование.

Контроль корректности проекта на каждом этапе проектирования осуществляется путем тестирования на основе модели объекта проектирования соответствующего уровня абстрактного представления.

Контроль корректности особо необходим на этапе формализации требований, поскольку формализация и моделирование целей проектирования во многих случаях затруднительна.

Для анализа функциональной спецификации привлекается коллектив экспертов или, если есть возможность, разрабатывается соответствующая модель для проверки соответствия проекта целям проектирования.

После отработки функциональной спецификации приступают к созданию *функциональных тестовых программ*, с помощью которых устанавливается правильность функционирования системы в соответствии с требуемой функциональной спецификацией.

Возможны два подхода к построению тестов:

• тесты, целиком основанные на функциональной спецификации и дающие возможность проверки любой реализации системы;

• тесты строятся применительно к конкретной реализации системы.

Преимущество подхода к построению тестов для функциональной проверки независимо от реализации состоит в том, что автоматизация трудоемкого процесса по составлению тестовых программ позволяет сгенерировать их сразу после формирования требований к системе. Это сокращает продолжительность конструирования и отладки за счет получения тестовых программ на этапе конструирования, а также позволяет проектировщику изменять спецификации, не заботясь о переписывании всех тестовых программ заново. Однако такой подход не имеет практического значения из-за высокой степени общности, приводящей к чрезмерному объему перебора при построении тестов.

На практике распространен подход, при котором тесты строятся применительно к конкретной реализации системы. Такой подход оправдан следующими соображениями:

• разработке тестов обычно присваивают более низкий приоритет по сравнению с остальным проектом, поэтому тестовые программы появляются к моменту создания прототипа системы;

• практически нецелесообразно запускать детально подготовленные тесты на имитаторе, так как это требует больших затрат на разработку и прогон программ моделирования.

5.2. Тестирование и автономная отладка аппаратных средств

Тестирование дискретных автоматов

Для контроля и диагностики автоматов применяют специальные тесты [14]. *Контролем* устройства называют обнаружение неисправности в работе устройства. Процесс контроля и диагностики можно совместить с работой устройства или же выполнить во время простоя устройства. В последнем случае на комбинационный автомат подают *контролирующий тест* – набор входных воздействий, реакция схемы на которые позволяет обнаружить неисправное функционирование в случае контроля или же локализовать отказ – в случае диагностики.

Троичное моделирование. **В качестве примера рассмотрим тестирование моделей комбинационных автоматов на стадии проектирования. Задержки сигналов, возникающие в реальных логических элементах, вызывают задержку выходных сигналов комбинационного автомата, а также являются причиной таких явлений, *как статический* и *динамический риск сбоя*.**

Возможность появления ложного сигнала $\overline{f}(A)$ на выходе комбинационного автомата в результате мгновенной замены набора входных сигналов A набором B, причем $f(A)=f(B)$, называется *статическим риском сбоя*.

Возможность многократных изменений сигнала на выходе комбинационного автомата в результате мгновенной замены набора входных сигналов A набором B, причем $f(B)= \overline{f}(A)$, называется *динамическим риском сбоя*. Динамический риск является следствием статического риска сбоя, поэтому задача выявления статического риска является актуальной.

Для обнаружения статического риска сбоя *на стадии проектирования* применяют *троичное моделирование* работы комбинационного автомата. Для представления неопределенного, промежуточного (между 0 и 1) значения двоичного сигнала, получающегося во время переходного процесса в схеме, берут число ½. Таким образом, от двоичного алфавита мы переходим к *троичному алфавиту*. Логические операции И, ИЛИ, НЕ при этом модифицируются следующим образом:

$$a \wedge b = \min(a, b);$$
$$a \vee b = \max(a, b)$$

$\bar{a} = a - 1$.

Пусть входной набор $A=(a_1,...,a_n)$ переходит в набор $B=(b_1,...,b_n)$. Сформируем переходный набор $A/B=(a_1/b_1,...,a_n/b_n)$, где $a_i / b_i = a_i$, если $a_i = b_i$, $a_i / b_i = ½$, если $a_i = \bar{b_i}$. Тогда если выходной сигнал комбинационного автомата $f(A)=f(B)$, *а* $f(A/B)= ½$, то имеет место статистический риск сбоя при переходе от набора A к набору B.

Приводим алгоритм, выявляющий все троичные наборы сигналов, дающие $f=½$ на выходе схемы.

1. A:=(0,….,0).
2. Если A:=(1,….,1), то конец вычислений.
3. A:=A+½.
4. Если набор A не содержит координаты ½, то перейти к п.2.
5. Вычислить f(A).
6. Если $f(A)\neq ½$, то перейти к п.3.
7. Выдать значение A и перейти к п.2.

Выполнение п.3 приведенного выше алгоритма осуществляется следующей процедурой:

1. i:=1.
2. S:=a_i+½.
3. Если S<1+½, то a_i:=S и конец работы процедуры.
4. a_i:=0.
5. i:=1+1.
6. Если $i<n$, то перейти к п. 2.
7. Конец работы процедуры.

При программировании п.5 необходимо все элементы комбинационного автомата разбить на уровни срабатывания: сначала вычисляются выходные сигналы первого уровня, затем второго и т.д. до элементов последнего уровня.

Тестирование комбинационных автоматов. В исправном состоянии комбинационный автомат реализует переключательную функцию f, определенную на множестве входных воздействий X. Пусть G – множество возможных неисправностей автомата. *Функцией неисправности* $g\in G$ автомата называется функция f_g , выполненная автоматом при наличии неисправности g. Функция неисправности f_g отличается от функции f исправного устройства.

Подмножество входных воздействий $T_К\subset X$ называется *контролирующим тестом автомата*, если для каждой неисправности $g\in G$, для которой $f \neq f_g$, найдется воздействие $e\in T_К$, для которого $f(e) \neq f_g(e)$.

Подмножество $T_Д\subset X$ называется *диагностическим тестом автомата*, если для каждой пары неисправностей $g,k\in G$, для которой $f_g \neq f_k$, найдется

воздействие $e \in T_Д$, для которого $f_g(e) \neq f_k(e)$. Тест T называется *полным*, если он локализует неисправности любой кратности, в противном случае тест называется *неполным*. Тест T называется *неизбыточным*, если удаление из него любого воздействия e дает подмножество $T \setminus \{e\}$, не являющееся тестом.

Существует два подхода к построению тестов.

1. Для каждой неисправности $g \in G$ строится множество воздействий $\{e\}_g$, проверяющих эту неисправность.

2. Случайным образом генерируется множество воздействий S. Для каждого воздействия $e \in S$, где $S \subseteq X$, определяется множество неисправностей $\{g\}_e$, проверяемых воздействием e. Если S не является тестом, то в S добавляются новые воздействия до тех пор, пока S не станет тестом.

Практическая реализация первого подхода для автоматов с большим числом элементов связана с решением систем булевых уравнений большой размерности и поэтому сложна.

Применение второго подхода связано с анализом работы автомата при наличии некоторой неисправности g, если на его вход поступает некоторое воздействие.

Тестирование автоматов с памятью. В основе тестирования (контроля и диагностики) дискретных абстрактных автоматов с памятью лежит *теория экспериментов с автоматами* [14]. Эксперимент над автоматами заключается в следующем:

- на вход автомата подается последовательность входных символов;
- фиксируется реакция автомата (последовательность выходных символов);
- в результате анализа поведения автомата делается заключение о том, в каком состоянии находится автомат и каков закон его функционирования.

При этом предполагается, что исследователю известна таблица переходов/выходов исправного автомата и все варианты возможных таблиц, характеризующих неисправности автомата.

Пусть автомат A при любой возможной неисправности $i=1,\ldots,k$ превращается в один из автоматов $\{A_1,A_2,\ldots,A_k\}=\{A_i|\ i=1,\ldots,k\}$. Тогда задача контроля данного автомата X, который должен работать как автомат A, заключается в выяснении соотношения $X=A$. Если $X \neq A$,то $X \in \{A_i|\ i=1,\ldots,k\}$. Задача диагностики заключается в определении i, при котором $X=A_i$ [ ].

Отладка аппаратных средств

Отладка прототипа проектируемой системы начинается с отладки аппаратных средств и отладки программ [15].

Отладка аппаратных средств предполагает тестирование отдельных устройств МПС (процессора, ОЗУ, контроллеров, блока питания, генератора тактовых импульсов) путем подачи тестовых входных воздействий и наблюдения ответных реакций. Это позволяет проверить реальную аппаратуру прототипа, спецификации, структурные схемы и отладить тесты. После завершения отладки отдельных устройств проверяется их взаимодействие.

Чтобы проконтролировать выполнение программы в процессоре, сначала необходимо убедиться, что сигналы, управляющие взаимодействием процессора с другими устройствами, выдаются в соответствующем порядке. Затем анализируют сигналы шин адреса (ША), данных (ШД) и управления (ШУ). Поскольку ША и ШД синхронные, начинают с проверки последовательности данных на этих шинах.

Поскольку линии ШУ работают асинхронно, необходимо просматривать сигналы нескольких линий на одном временном интервале. При этом необходимо также наблюдать за сигналами управления при возникновении определенного события, а также отслеживать кратковременные ложные узкие импульсы, вызванные перекрестными помехами или шумами.

После проверки работоспособности ША, ШД и ШУ, приступают к дальнейшей проверке аппаратуры при различных режимах адресации процессора и кодах выбираемых данных. Тестовая программа для проверки выполнения процессором инструкций размещается в ОЗУ или ППЗУ. При этом проверяется временная диаграмма сигналов и прохождения данных в системе относительно строб-сигналов.

Для автономной отладки МПС широко используются осциллографы, вольтметры, амперметры, частотомеры, генераторы импульсов, позволяющие отлаживать аппаратуру на схемном уровне. Для автономной отладки МПС на более высоком уровне, применяют логические анализаторы, генераторы слов, пульты, комплексы диагностирования. Эти инструментальные средства можно классифицировать следующим образом:

- приборы для измерения напряжения и тока;
- генераторы сигналов и импульсов заданной формы;
- генераторы нескольких одновременных последовательностей сигналов в соответствии с заданной временной диаграммой или алгоритмом

функционирования аппаратуры, представленным на языке спецификаций высокого уровня;

• приборы для сбора значений сигналов многих линий в течение одного и того же промежутка времени, который определяется задаваемыми, программируемыми событиями – комбинацией или последовательностью сигналов на линиях;

• приборы для обработки и представления собранной информации в виде временных диаграмм, таблиц логических состояний (таблиц истинности), либо на языке высокого уровня, например языке регистровых передач.

5.3. Тестирование и автономная отладка программных средств

Создание и весь последующий жизненный цикл надежного программного обеспечения для современных информационно-вычислительных систем – многоэтапный и трудоемкий процесс, который упрощенно можно охарактеризовать как *перевод требований технического* задания сначала в точные спецификации и, наконец, в текст программы [8, 11, 15].

Сложность программного продукта как объекта проектирования – основная причина ошибок перевода и, следовательно, ненадежности программного обеспечения. Для снижения сложности проекта используют принципы теории систем: независимость компонентов системы и иерархическая структура проекта программного обеспечения. Разработаны принципы и технология модульного проектирования, объектно-ориентированный подход.

Распространенный подход к обеспечению надежности проектируемого программного обеспечения – это *тестирование*. Цель тестирования – выявление ошибок, вкравшихся в программу на разных стадиях проектирования. При таком подходе при написании программ акцент делается на их *тестируемость*, т.е. на создание программ, которые удобно тестировать, а безошибочность и корректность программы в значительной степени зависят от творческих способностей и интуиции разработчика.

В отличие от интуитивного подхода, который мы охарактеризовали выше, рассматриваемый далее подход, верификация (доказательство правильности) алгоритмов и программ, трактует программирование как точную математическую науку.

Для описания алгоритмов используются различные методы, отличающиеся степенью детализации и формализации. Теоретическое

описание обычно дается в повествовательно-формальном изложении, цель которого – обосновать без лишних подробностей процедуру, предлагаемую в качестве алгоритма. Для наглядного представления структуры алгоритмов широко применяются графические средства: графы, блок-схемы, сети. Формальное и полное описание алгоритмов осуществляется на специально разработанных алгоритмических языках (BASIC, FORTRAN, PASCAL и др.); оно содержит всю необходимую для реализации алгоритма информацию, но не связано непосредственно со специфическими особенностями вычислительных машин.

Логика доказательства правильности алгоритмов и программ

Цель данного раздела – познакомить читателя с принципами *верификации* (*доказательства правильности*) алгоритмов и программ [13]. В основе верификации программ (алгоритмов) – анализ действия программ (алгоритмов) над данными. Для каждого исходного состояния данных X, для которого выполнение завершается, результирующее состояние данных Y является определенным. Это значение Y единственно для данного X, поэтому множество всех упорядоченных пар (X,Y) определяет функцию, которую будем называть *программной функцией*.

Мы будем использовать *символьные вычисления*, чтобы получить аналитическое выражение программной функции для исследуемой программы. Программную функцию f для *простой* программы P обозначим через $[P]$ и определим выражением

$$[P]=\{(X,Y) \mid Y=f(X)\},$$

где X - состояние поля данных до выполнения P,

Y - состояние поля данных после выполнения P,

$\{(X,Y) \mid Y=f(X)\}$ - множество пар (X,Y) таких, что $Y=f(X)$.

Тестирование и отладка программных средств

Разработка и автономная отладка программных средств МПС может производиться на больших ЭВМ, миниЭВМ или микроЭВМ, система команд которых не совпадает с системой команд используемого микропроцессора. Кроме того, при отладке программ может отсутствовать внешняя среда микропроцессорной системы, ее также необходимо моделировать. Отладка программ МПС проводится, как правило, на тех же ЭВМ, на которых велась

разработка программ, и на том же языке программирования, на котором написаны отлаживаемые программы.

Если отладка программ начинается до создания аппаратуры МПС, то в системное программное обеспечение инструментальной ЭВМ должны входить программы (интерпретаторы или эмуляторы), моделирующие функции отсутствующих аппаратных средств.

Для проверки корректности программ, т.е. их соответствия внешним спецификациям, выполняется тестирование: программы запускаются с различными исходными данными, а результаты прогонов сравниваются с эталонными значениями.

Отладка программ состоит из этапов:

- планирование отладки;
- составление тестов и задания на отладку;
- прогон тестов и выдача результатов;
- анализ результатов, обнаружение ошибок и локализация неисправностей.

Перечислим функции, которые должны выполнять средства отладки программ:

- управление исполнением программы (запуск, изменение порядка, останов и т. д.);
- сбор информации о ходе выполнения программы;
- обеспечение диалога между программистом и ЭВМ на уровне языка программирования;
- моделирование работы отсутствующих аппаратных средств проектируемой МПС.

Рассмотрим два основных способа начального тестирования программ: пошаговый режим и трассировку программ. Эти способы применяются, как правило, к отдельным участкам программ.

Пошаговый режим предполагает, что программа выполняется по одной команде за один раз, а программист анализирует содержимое памяти и регистров и сопоставляет результаты с ожидаемыми значениями.

Трудоемкость пошагового режима уменьшается, если имеются средства отладки, автоматически показывающие содержимое регистров процессора и ячеек памяти, используемых в последней команде, и несколько следующих команд. Пошаговый режим является весьма эффективным средством предварительного тестирования, поскольку позволяет обнаруживать неисправности, прежде чем они существенно исказят программу и данные.

Пошаговый режим позволяет изменять содержимое регистров и ячеек памяти и, следовательно, проверять работу программы в разных условиях. Тем самым этот режим отладки позволяет программисту разработчику постоянно упреждать действия программы и оперативно обнаруживать ошибку.

Однако для осуществления пошагового режима с автоматическим показом результатов требуется терминал, позволяющий после каждого шага оперативно обновлять большой объем информации, выводимой на экран.

Трассировка программы осуществляется по шагам в соответствии с заданиями, содержащимися в операторах. При этом регистрируется последовательность операторов, реализуемых на каждом шаге процесса и, следовательно, получается *трасса или маршрут исполнения программы*, который для конкретной программы зависит только от значений исходных данных. Программа-отладчик выполняет автоматически команду за командой и выводит содержимое регистров процессора на терминал после каждого шага. Отслеживание программы продолжается до тех пор, пока не будет остановлено извне.

Существуют отладчики, которые выводят на терминал не только операторы программы, но и команды в дизассемблерной форме. Однако при этом способе содержимое памяти не выводится на терминал и разработчик должен сам делать выводы об изменениях в ней.

Результаты трассировки выводятся на экран терминала или на принтер. Программист, анализируя эти данные, может обнаружить ошибки. Для трассировки программ можно использовать более медленный терминал.

Недостаток трассировки заключается в том, что она, во-первых, не дает возможности изменять содержимое памяти и регистров, а во-вторых, может послужить причиной того, что программа разрушит себя или свои данные прежде, чем отслеживание будет остановлено.

Метод контрольных точек используют после проверки отдельных участков программы в пошаговом режиме или с помощью трассировки. В контрольных точках, установленных программистом, прерывается исполнение программы и управление передается программе-отладчику. Это позволяет по желанию выполнить избранные участки программы и проанализировать результаты.

Программист устанавливает контрольные точки, как правило, для конкретной команды, но в некоторых системах имеется возможность прерывания программы при чтении определенных ячеек памяти или при записи в них.

Проверка всех возможных маршрутов исполнения программы для разных исходных переменных возможна только для очень простых программ небольшого объема, причем при малых диапазонах изменения исходных данных. Поэтому при планировании отладки программ применяют *критерии полноты тестирования*. Критерии характеризуют глубину контроля программ и объем проверок. Назначение требуемой полноты проверки зависит от структурной сложности отлаживаемой программы и наличия ресурсов для тестирования.

Отладка позволяет выявить и устранить основную часть неисправностей программы, однако гарантировать отсутствие дефектов после отладки сложных программ нельзя.

5.4. Комплексная отладка микропроцессорных систем

Перечислим пять основных приемов комплексной отладки МПС [11, 15]:

1) пошаговое отслеживание поведения системы;

2) отслеживание поведения системы в реальном времени;

3) чтение (изменение) содержимого ячеек памяти или регистров системы;

4) останов функционирования системы при возникновении определенного события;

5) временное согласование программ.

Поскольку специализированные МПС – это, как правило, системы реального времени, корректность их функционирования зависит от времени выполнения отдельных программ и скорости работы аппаратуры. Такие системы считаются отлаженными только в том случае, когда рабочие программы правильно функционируют на реальной аппаратуре и в реальных условиях. Таким образом, средства комплексной отладки должны обладать дополнительным свойством по сравнению со средствами автономной отладки, а именно обеспечивать управление поведением МПС и сбор данных о ее поведении в реальном времени.

Тенденция развития средств отладки МПС заключается в объединении функций нескольких приборов в одном комплексе, в создании универсальных средств, пригодных как для автономной отладки аппаратуры и программ, так и для комплексной отладки всей системы. Такие комплексы позволяют вести разработку и отладку поэтапно с нарастанием сложности таким образом, что новые, не отлаженные компоненты аппаратуры и программ присоединяются к

уже проверенной части системы. Средства отладки, используемые на последних этапах, не должны вносить дополнительные задержки или нагрузки, т.е. влиять на правильность функционирования системы.

Если программы отлаживаются с использованием ОЗУ, а затем переносятся в микросхемы ПЗУ, то они должна быть заново протестированы в составе микропроцессора.

Для увеличения полноты контроля работоспособности проектируемой МПС следует увеличивать число тестируемых логических маршрутов обработки информации за счет расширения диапазона возможных сочетаний входных переменных. Для этого детерминированное тестирование можно сочетать со статистическим тестированием, при котором исходные данные изменяются в соответствии со статистическими законами работы реальных источников информации.

Комплексная отладка завершается приемосдаточными испытаниями, показывающими соответствие спроектированной МПС техническому заданию.

Заключение

Темпы развития микропроцессорной техники за последние 30 лет стремительны: мы являемся свидетелями безудержной гонки в развитии как аппаратных, так и программных средств МПС. В последнее десятилетие наблюдается появление новых областей использования информационных технологий таких, как виртуальные производства и электронная коммерция, которые трудно было предвидеть заранее. Особенно быстро развиваются сетевые интеллектуальные технологии.

Будущее микропроцессорной техники связано в первую очередь с расширением областей использования МПС. Появилась возможность использования микропроцессорных систем на всех уровнях производства, включая встроенные системы мониторинга и управления технологическим оборудованием и процессами на основе средств микропроцессорной техники. Расширяется производство высокотехнологичных изделий, насыщенных электроникой, точной механикой и оптикой (мехатроника).

Будущее микропроцессорной техники связано также с двумя новыми направлениями – нанотехнологиями и квантовыми вычислительными системами. Для этого ведутся теоретические исследования по использованию в вычислительной технике свойств молекул и субатомных частиц: основой для хранения и переработки данных должны стать не электрические цепи, как сейчас, а положение отдельных атомов или направление вращения (спины) электронов. Если "квантовые" компьютеры будут созданы, то они превзойдут современные машины по многим параметрам.

Приложение

Система команд микропроцессора КР580ИК80А

Таблица П1. **Команды пересылки**

Мнемокод команды	Операция	Формат команды	Число тактов
	Пересылка		
MOV r1, r2	(r1)←(r2)	01DDDSSS	5
MOV r, M	(r)←[(H)(L)]	01DDD110	7
MOV M, r	(r)←[(H)(L)]	01110SSS	7
MVI r, ⟨B2⟩	(r)←⟨B2⟩	00DDD110	7
MVI M, ⟨B2⟩	[(H)(L)]←⟨B2⟩	01100110	10
	Загрузка		
LXI rp, ⟨B2⟩ ⟨B3⟩	(rh)←⟨B2⟩, (rl)←⟨B2⟩	00RP0001	10
LDA	(A) ←[⟨B3⟩⟨B2⟩]	00111010	13
LHLD	(L)←[⟨B3⟩⟨B2⟩], (H)←[⟨B3⟩⟨B2+1⟩]	00101010	16
LDAX rp	(A) ←[(rp)] для rp∈{B,D}	00RP1010	7
	Запись		
STA	[⟨B3⟩⟨B2⟩]← (A)	00110010	13
SHLD	[⟨B3⟩⟨B2⟩]←(L), [⟨B3⟩⟨B2+1⟩]←(H)	00100010	16
STAX rp	[(rp)] ←(A) для rp∈{B,D}	00RP0010	7
	Обмен		
XCHG	(H)↔(D), (L)↔(E)	11101011	4

Таблица П2. **Арифметические, инкрементные и декрементные команды**

Мнемокод команды	Операция	Формат	Число тактов	Устанавливаемые флаги
	Сложение			
ADD r	(A)←(A)+(r)	10000SSS	4	Все
ADD M	(A)←(A)+[(H)(L)]	10000110	7	Все
ADI ⟨B2⟩	(A)←(A)+⟨B2⟩	11000110	7	Все
	Сложение с переносом			
ADC r	(A)←(A)+(r)+(CY)	10001SSS	4	Все
ADC M	(A)←(A)+[(H)(L)]+(CY)	10001110	7	Все
ACI ⟨B2⟩	(A)←(A)+⟨B2⟩+(CY)	11001110	7	Все
	Вычитание			
SUB r	(A)←(A)–(r)	10010SSS	4	Все
SUB M	(A)←(A)–[(H)(L)]	10010110	7	Все
SUI ⟨B2⟩	(A)←(A)–⟨B2⟩	11010110	7	Все

Окончание табл. П2

Мнемокод команды	Операция	Формат	Число тактов	Устанавливаемые флаги
	Вычитание с заемом			
SBB r	(A)←(A)–(r)–(CY)	10011SSS	4	Все
SBB M	(A)←(A)–[(H)(L)]–(CY)	10011110	7	Все
SBI ⟨B2⟩	(A)←(A)–⟨B2⟩–(CY)	11011110	7	Все
	Инкремент			
INR r	(r)←(r)+1	000DDD100	5	Z, S, P, CY
INR M	[(H)(L)]←[(H)(L)]+1	00110100	5	Z, S, P, CY
INX rp	(rh)(rl)←(rh)(rl)+1	00RP0011	5	–
	Декремент			
DCR r	(r)←(r)–1	00DDD101	5	Z, S, P, CY
DCR M	[(H)(L)]←[(H)(L)]–1	00110101	10	Z, S, P, CY
DCX rp	(rh)(rl)←(rh)(rl)–1	00RP1011	5	–
	Сложение пары регистров			
DAD rp	(H)(L)←(H)(L)+(rh)(rl)	00RP1001	10	CY
	Логические команды			
DAA	Десятичная коррекция*	00100111	4	Все
RLC	Циклический сдвиг A влево	00000111	4	CY
RRC	Циклический сдвиг A вправо	00001111	4	CY
RAL	Цикл. сдвиг A влево с переносом	00010111	4	CY
RAR	Цикл. сдвиг A вправо с переносом	00011111	4	CY
CMA	Инвертирование A	00101111	4	–

STC CMC	(CY) ←1 Инвертирование (CY)	00110111 00111111	4 4	CY=1 CY

**Примечание*. 8-разрядное число в аккумуляторе преобразуется по правилу: 1) если содержимое четырех младших разрядов больше 9 или если (CY)=1, то к аккумулятору добавляется 6; 2) если содержимое четырех старших разрядов стало после этого больше 9 или если (CY)=1, то число 6 добавляется и к содержимому четырех старших разрядов аккумулятора

Таблица П3. **Команды управления, ввода-вывода и работы со стеком**

Мнемокод команды	Операция	Формат	Число тактов
JMP ⟨B2⟩⟨B3⟩	Безусловный переход: (PC)←⟨B3⟩⟨B2⟩	11000011	10
PCHL	Загрузка PC: $(PC)_H$←(H), $(PC)_L$←(L)	11101110	5
Jcc ⟨B2⟩⟨B3⟩	Условный переход: (PC)←⟨B3⟩⟨B2⟩	11CCC010	10
CALL ⟨B2⟩⟨B3⟩	Безусловный вызов подпрограммы: [(SP)-1]← $(PC)_H$, [(SP)-2]← $(PC)_L$, (SP)←(SP)-2, (PC)←⟨B3⟩⟨B2⟩	11001101	17

Окончание табл. П3

Мнемокод команды	Операция	Формат	Число тактов
Ccc ⟨B2⟩⟨B3⟩	Условный вызов подпрограммы: [(SP)-1]← $(PC)_H$, [(SP)-2]← $(PC)_L$, (SP)←(SP)-2, (PC)←⟨B3⟩⟨B2⟩	11CCC100	11/17
RET	Безусловный возврат из подпрограммы: $(PC)_L$←[(SP)], $(PC)_H$←[(SP)+1], (SP) ←(SP)+2	11001001	10
Rcc	Условный возврат из подпрограммы: $(PC)_L$←[(SP)], $(PC)_H$←[(SP)+1], (SP) ←(SP)+2	11CCC000	5/11
RST n	Вызов подпрограммы прерываний: [(SP)-1]← $(PC)_H$, [(SP)-2]← $(PC)_L$, (SP)←(SP)-2, (PC)←8·(AAA)	11AAA111	11
PUSH rp	Проталкивание в стек содержимого пары rp: [(SP)-1]← (rh), [(SP)-2]← (rl), (SP)←(SP)-2	11RP0101	11
POP rp	Выталкивание из стека содержимого пары rp: (rl)←[(SP)], (rh)←[(SP)+1], (SP)←(SP)+2	11RP0001	10
PUSH PSW	Ввод PSW в стек	11110101	11
POP PSW	Выдача данных их стека в аккумулятор и регистр прерывания (устанавливаются все флаги)	11110001	10

XTHL	Обмен между вершиной стека и парой HL: (L)↔[(SP), (H)↔[(SP)+1]	11100011	18
SPHL	Загрузка указателя стека: (SP)←(H)(L)	11111001	5
IN port	Ввод из порта: (A) ← port	11011011	10
OUT port	Вывод в порт: port←(A)	11010011	10
EI	Разрешение прерывания	11111011	4
DI	Запрет прерывания	11110011	4
HLT	Останов	01110110	7
NOP	Пустая операция	00000000	4

Система команд МП КР580ИК80А содержит команды пересылки, арифметические, логические, управления, вызова подпрограмм и ввода-вывода. Для описания команд в табл.П1, П2 и П3 использованы следующие условные обозначения:

DDD и SSS – код одного из рабочих регистров (DDD – для регистра-приемника и SSS – для регистра-источника), указанного в команде, в соответствии с табл. П4;

M – ячейка памяти, адрес которой указан в регистровой паре HL;

rp – регистровая пара (B, D, H и SP);

RP – код регистровой пары в формате команды в соответствии с табл. П5;

rh – регистр пары, в котором содержатся старшие разряды;

rl – регистр пары, в котором содержатся младшие разряды;

B2 B3 – второй и третий байты команды соответственно;

[( )( )] – содержимое ячейки памяти, адрес которой указан в скобках.

Признаки результата (флаги S, Z, P, C и CY) устанавливаются только при выполнении большинства арифметических и логических операций, а также команды POP PSW. В табл.П6 приведены обозначения флагов и коды условий.

Таблица П4. **Коды регистров**

Регистр ЦП	A	B	C	D	E	H	L	M
Код регистра	111	000	001	010	011	100	101	110

Таблица П5. **Коды регистровых пар**

Регистровая пара	BC	DE	HL	SP
RP	00	01	10	11

Таблица П6. **Коды условий и значения флагов**

Условие cc	Код условия в формате команды ccc		Условие cc	Код условия в формате команды ccc
cc=NZ – ненулевой результат текущей операции (Z)=0	000		cc=PO – нечетное число единиц результата (P)=0	100
cc=Z – нулевой результат текущей операции (Z)=1	001		cc=PE – четное число единиц результата (P)=1	101
cc=NC –отсутствие переноса (CY)=0	010		cc=P – результат положительный (S)=0	110

cc=C – наличие переноса (CY)=1	011		cc=M – результат отрицательный (S)=1	111

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Майерс Г., Надежность программного обеспечения. - М.: Мир, 1980. – 360 с.

2. Хвощ С.Т., Варлинский Н.Н., Попов Е.А. Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник. – Л.: Машиностроение, 1987. – 640 с.

3. Применение интегральных микросхем в электронной вычислительной технике: Справочник. – М.: Радио и связь, 1987. – 384 с.

4. Угрюмов Е.П. Проектирование элементов и узлов ЭВМ. – М.: Высш. школа, 1987. – 318 с.

5. Майоров С.А., Кириллов В.В., Приблуда А.А. Введение в микроЭВМ. – Л.: Машиностроение, 1988. – 304 с.

6. Микропроцессорные системы автоматического управления / В.А. Бесекерский, Н.Б. Ефимов, С.И. Зиатдинов и др. – Л.: Машиностроение, 1988. – 386 с.

7. Гуртовцев А.Л., Гузыненко С.В. Программы для микропроцессоров: Справочное пособие. – Минск: Вышейшая школа, 1989. – 352 с.

8. Штрик А.А. и др. Структурное проектирование надежных программ встроенных ЭВМ/ А.А.Штрик, Л.Г. Осовецкий, И.Г.Мессих. - Л.: Машиностроение, 1989. – 296 с.

9. Корячко В.П. – Микропроцессорные системы и микроЭВМ в радиоэлектронных средствах. – М.: Высш. школа, 1990. – 407 с.

10. Микропроцессоры: Методические указания к выполнению лабораторных работ. – Л.: СЗПИ, 1990. – 22 с.

11. Шеремет В.П., Петров Б.Г., Надежность и качество программного обеспечения судовых СУТС. - Л.: Аврора, 1992.– 120 с.

12. Анкудинов Г.И., . Анкудинов И.Г., Хамидуллин Р.Р. Численные методы. Линейные преобразования, приближение функций.– СПб.: СЗТУ, 2000.– 72 с.

13. Анкудинов Г.И., . Анкудинов И.Г., Петухов О.А. Математическая логика и теория алгоритмов.– СПб.: СЗТУ, 2002.– 104 с.

14. Анкудинов Г.И., . Анкудинов И.Г., Хамидуллин Р.Р. Теория автоматов.– СПб.: СЗТУ, 2002.– 112 с.

15. Ершова Н.Ю, Ивашенков О.Н., Курсков С.Ю. Микропроцессоры. – http://docs.lab127.karelia.ru/hardware/microcpu.

Основные сокращения

АЦП –аналого-цифровой преобразователь
БИС – большая интегральная схема
ВВ – ввод-вывод
ВК – вычислительный комплекс
ВУ – внешнее устройство
ЗУ – запоминающее устройство
ИМС – интегральная микросхема
МКОД – системы с многократным потоком команд и однократным потоком данных
МП – микропроцессор
МПВК – многопроцессорный ВК
МПС – микропроцессорная система
ОЗУ – оперативное ЗУ
ОКМД – однократный поток команд и многократный поток данных
ОКОД – однократный поток команд и однократный поток данных
ОП – оперативная память
ОС – операционная система
ОШ – общая шина
ПЗУ – постоянное ЗУ
ПК – персональный компьютер
СБИС – сверхбольшая интегральная схема
УВВ – устройство ВВ
ЦАП – цифро-аналоговый преобразователь
ЦП – центральный процессор
ША – шина адреса
ШД – шина данных
ШУ – шина управления
CISC – Complex Instruction Set Computers (полная система команд)
FIFO – (первым вошел – первым обслужен)
LIFO – Last In First Out (последним вошел – первым обслужен)
RISC – Reduced Instruction Set Computers (сокращенная система команд)– First In First Out

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

СОДЕРЖАНИЕ

Анкудинов Иван Георгиевич

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

АРХИТЕКТУРА И ПРОЕКТИРОВАНИЕ

Учебное пособие

Редактор И.Н.Садчикова

Сводный темплан 2003 г.
Лицензия ЛР №020308 от 14.02.97

Подписано в печать **Формат 60 x 84 1/16**

Б. кн.-журн. П.л. 6,812 **Б.л. 3,406 РТП РИО СЗТУ.**

Тираж 100 **Заказ**

Северо-Западный государственный заочный технический университет

РИО СЗТУ, член Издательско-полиграфической ассоциации вузов Санкт-Петербурга

191186, Санкт-Петербург, ул.Миллионная, 5