

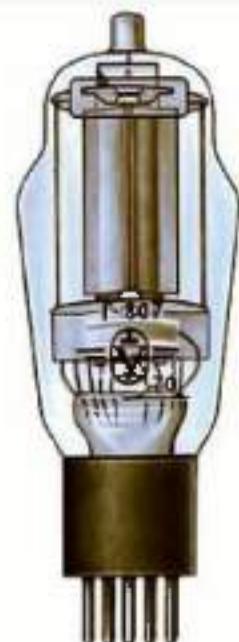
# **Администрирование и программирование микропроцессорной техники**

## **Электронно-вычислительные машины**

В 1970 году Марсиан Эдвард Хофф из фирмы Intel сконструировал интегральную схему, аналогичную по своим функциям центральному процессору большой ЭВМ - первый микропроцессор Intel-4004, который уже в 1971 году был выпущен в продажу.

15 ноября 1971 г. можно считать началом новой эры в электронике. В этот день компания приступила к поставкам первого в мире микропроцессора Intel 4004.

## 1.1. Первое поколение ЭВМ



Появление электронно-вакуумной лампы позволило ученым претворить в жизнь идею создания вычислительной машины. Она появилась в 1946 году в США для решения задач и получила название ЭНИАК (ENIAC - Electronic Numerical Integrator and Calculator, в переводе "электронный численный интегратор и калькулятор").

Характерные черты ЭВМ первого поколения.

Элементная база: электронно-вакуумные лампы, резисторы, конденсаторы. Соединение элементов - навесной монтаж проводами.

Габариты: ЭВМ выполнена в виде громоздких шкафов и занимает специальный машинный зал.

Быстродействие: 10-20 тыс. оп/с.

Эксплуатация слишком сложна из-за частого выхода из строя. Существует опасность перегрева ЭВМ.

Программирование: трудоёмкий процесс в машинных кодах. При этом необходимо знать все команды машины, их двоичное представление, а также различные структуры ЭВМ.

## 1.2. Второе поколение



Второе поколение пришлось на период от конца 50-х до конца 60-х годов. Был изобретён транзистор, который пришёл на смену электронным лампам. Это позволило изменить элементную базу ЭВМ.

Характерные черты ЭВМ второго поколения

Элементная база: полупроводниковые элементы.  
Соединение элементов - печатные платы и навесной монтаж.

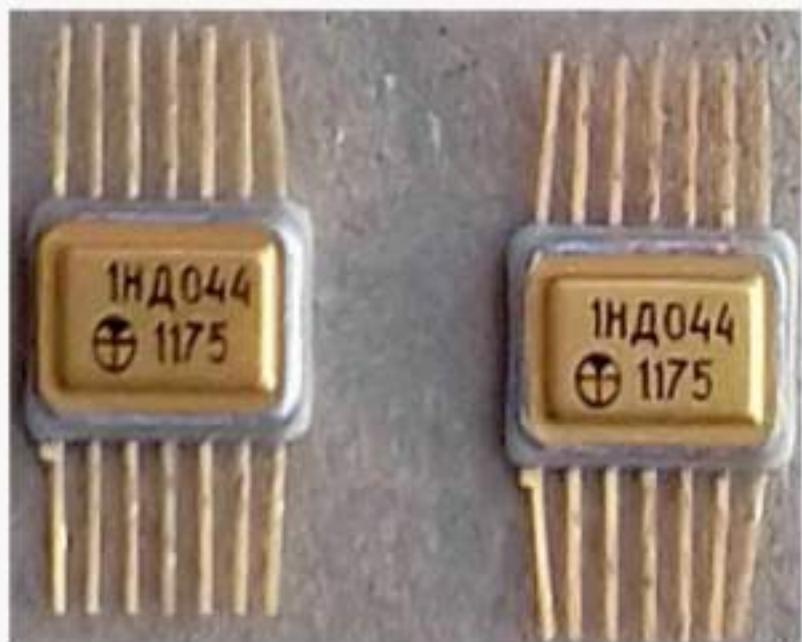
Габариты: ЭВМ выполнена в виде однотипных стоек, чуть выше человеческого роста. Для их размещения требуется специально оборудованный машинный зал.

Производительность: до 1 млн. оп/с.

Эксплуатация: упростилась. Появились вычислительные центры с большим штатом обслуживающего персонала.

Программирование: существенно изменилось, так как велось преимущественно на алгоритмических языках. Программисты уже не работали в зале, а отдавали свои программы на перфокартах или магнитных лентах специально обученным операторам.

### 1.3. Третье поколение



Этот период длился с конца 60-х до конца 70-х годов. Появление интегральных схем ознаменовало новый этап в развитии вычислительной техники.

Первой ЭВМ, выполненной на интегральных схемах, была IBM-360 фирмы IBM.

Характерные черты ЭВМ третьего поколения

Элементная база: интегральные схемы, которые вставляются в специальные гнёзда на печатной плате.

Габариты: для их размещения также требуется машинный зал.

Производительность: сотни тысяч - миллионы оп/с.

Эксплуатация: более оперативно производится ремонт стандартных неисправностей, но из-за большой сложности системной организации требуется штат высококвалифицированных специалистов. Незаменимую роль играет системный программист. Появились дисплеи и графопостроители.

Программирование: во многих вычислительных центрах появились дисплейные залы, где каждый программист в определенное время мог подсоединиться к ЭВМ в режиме разделения времени. Как и прежде, основным оставался режим пакетной обработки задач.

## 1.4. Четвёртое поколение



Этот период оказался самым длинным - от конца 70-х годов по настоящее время. Новые технологии создания интегральных схем позволили разработать ЭВМ четвертого поколения на больших интегральных схемах (БИС), степень интеграции которых составляет десятки и сотни тысяч элементов на одном кристалле.

Характерные черты ЭВМ четвертого поколения

Элементная база: микропроцессоры. Первый микропроцессор был создан фирмой Intel в 1971 году. На одном кристалле удалось сформировать минимальный по составу аппаратуры процессор, содержащий 2250 транзисторов.

Габариты: персональные компьютеры, размещающиеся на столе пользователя.

Производительность: десятки - сотни миллионов оп/с.

Эксплуатация: совместимость программного обеспечения снизу вверх и принцип открытой архитектуры, предусматривающий возможность дополнения имеющихся аппаратных средств без смены старых или их модификации без замены всего компьютера.

## **История развития микропроцессоров. Внутренняя организация микропроцессора.**

Микропроцессор (МП) – это программируемое электронное цифровое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки, выполненное на одной или нескольких интегральных схемах с высокой степенью интеграции электронных элементов.

### **История развития микропроцессора**

В 1970 году Маршиан Эдвард Хофф из фирмы Intel сконструировал интегральную схему, аналогичную по своим функциям центральному процессору большой ЭВМ - первый микропроцессор Intel-4004, который уже в 1971 году был выпущен в продажу.

15 ноября 1971 г. можно считать началом новой эры в электронике. В этот день компания приступила к поставкам первого в мире микропроцессора Intel 4004.

Кристалл представлял собой 4-разрядный процессор с классической архитектурой ЭВМ гарвардского типа и изготавливается по передовой р-канальной МОП технологии с проектными нормами 10 мкм. Электрическая схема прибора насчитывала 2300 транзисторов. МП работал на тактовой частоте 750 кГц при длительности цикла команд 10,8 мкс. В систему его команд входило всего 46 инструкций.

1 апреля 1972 г. фирма Intel начала поставки первого в отрасли 8-разрядного прибора i8008. Кристалл изготавливается по р-канальной МОП-технологии с проектными нормами 10 мкм и содержал 3500 транзисторов. Процессор работал на частоте 500 кГц при длительности машинного цикла 20 мкс (10 периодов задающего генератора).

В отличие от своих предшественников МП имел архитектуру ЭВМ принстонского типа, а в качестве памяти допускал применение комбинации ПЗУ и ОЗУ.

Система команд насчитывала 65 инструкций. МП мог адресовать память объемом 16 Кбайт. Его производительность по сравнению с четырехразрядными МП возросла в 2,3 раза.

Возможности р-канальной технологии для создания сложных высокопроизводительных МП были почти исчерпаны, поэтому развитие вычислительной техники перешло на п-канальную МОП технологию.

1 апреля 1974 был выпущен МП Intel 8080. Благодаря использованию технологии п-МОП с проектными нормами 6 мкм, на кристалле удалось разместить 6 тыс. транзисторов (рис. 1.1).



Рис. 1.1 Микропроцессор Intel 8080

## 2.2 Классификация микропроцессоров

Для описания МП как функциональных устройств необходимо охарактеризовать формат обрабатываемых данных и команд, количество, тип и гибкость команд, методы адресации данных, число внутренних регистров общего назначения и регистров результата, возможности организации и адресации стека, параметры виртуальной памяти и информационную смкость прямо адресуемой памяти. Большое значение имеют средства построения системы прерываний программ, построения эффективных систем ввода — вывода данных и развитого интерфейса.

По назначению различают универсальные и специализированные микропроцессоры.

Универсальные МП предназначены для решения широкого круга задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. В системе команд МП заложена алгоритмическая универсальность, означающая, что выполняемый машиной состав команд позволяет получить преобразование информации в соответствии с любым заданным алгоритмом.

К универсальным МП относятся и секционные микропроцессоры, поскольку для них система команд может быть оптимизирована в каждом частном проекте создания секционного микропроцессора.

Эта группа МП наиболее многочисленна, в нее входят такие комплекты как K580, Z80, Intel 80x86, K582, K587, K1804, K1810 и др.

*Специализированные МП* предназначены для решения определенного класса задач, а иногда только для решения одной конкретной задачи. Их существенными особенностями являются простота управления, компактность аппаратурных средств, низкая стоимость и малая мощность потребления.

Специализированные МП имеют ориентацию на ускоренное выполнение определенных функций, что позволяет резко увеличить эффективную производительность при решении только определенных задач.

Среди специализированных микропроцессоров можно выделить различные микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций; математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например матричных методов их выполнения; МП для обработки данных в различных областях применений и т. д.

По виду обрабатываемых входных сигналов различают цифровые и аналоговые микропроцессоры.

Сами МП являются цифровыми устройствами обработки информации. Однако в ряде случаев они могут иметь встроенные аналого-цифровые и цифро-аналоговые преобразователи. Поэтому входные аналоговые сигналы передаются в МП через преобразователь в цифровой форме, обрабатываются и после обратного преобразования в аналоговую форму поступают на выход.

С архитектурной точки зрения такие микропроцессоры представляют собой аналоговые функциональные преобразователи сигналов. Они выполняют функции любой аналоговой схемы (например, производят генерацию колебаний, модуляцию, смещение, фильтрацию, кодирование и декодирование сигналов в реальном масштабе времени и т. д., заменяя сложные схемы, состоящие из операционных усилителей, катушек индуктивности, конденсаторов и т.д.).

Обычно в составе однокристальных аналоговых МП имеется несколько каналов аналого-цифрового и цифро-аналогового преобразования. В аналоговом микропроцессоре разрядность обрабатываемых данных достигает 24 бит и более. Большое значение уделяется увеличению скорости выполнения арифметических операций.

Отличительная черта аналоговых МП - это способность к переработке большого объема числовых данных, т. е. к выполнению операций сложения и умножения с большой скоростью, при необходимости даже за счет отказа от операций прерываний и переходов.

Аналоговый сигнал, преобразованный в цифровую форму, обрабатывается в реальном масштабе времени и передается на выход обычно в аналоговой форме через цифро-аналоговый преобразователь. При этом согласно теореме Котельникова частота квантования аналогового сигнала должна вдвое превышать верхнюю частоту сигнала.

По количество выполняемых программ различают одно- и многопрограммные микропроцессоры.

В однопрограммных МП выполняется только одна программа. Переход к выполнению другой программы происходит после завершения текущей программы.

В много- или мультипрограммных МП одновременно выполняется несколько (обычно несколько десятков) программ. Организация мультипрограммной работы микропроцессорных управляющих систем, например, позволяет осуществить контроль за состоянием и управлением большим числом источников или приемников информации.

По числу БИС в микропроцессорном комплекте различают однокристальные, многокристальные и многокристальные секционные МП.

Реализовать принципиальную схему обычного процессора в виде одной или нескольких БИС практически невозможно из-за специфических особенностей БИС (ограниченность количества элементов, сложность выполнения разветвленных связей, сравнительно небольшое число выводов корпуса).

Поэтому необходимо изменять структуру процессора так, чтобы полная принципиальная схема или ее части имели количество элементов и связей, совместимое с возможностями БИС. При этом МП приобретают *внутреннюю магистральную структуру*, т. е. в них к единой внутренней информационной магистрали подключаются все основные функциональные блоки (арифметико-логический, рабочих регистров, стека, прерываний, интерфейса, управления и синхронизации и др.).

Для обоснования классификации МП по числу БИС надо распределить все аппаратные блоки процессора между основными тремя функциональными частями: *операционной, управляющей и интерфейсной*.

Интерфейс процессора содержит несколько десятковшин информационных магистралей данных, адресов и управления.

Однокристальные МП получаются при реализации всех аппаратных средств процессора в виде одной БИС или СБИС. По мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса параметры однокристальных МП улучшаются.

Однако их возможности ограничены аппаратными ресурсами кристалла и корпуса. Поэтому более широко распространены многокристальные, а также многокристальные секционные МП.

Для получения многокристального МП необходимо провести разбиение его логической структуры на функционально законченные части и реализовать их в виде БИС (СБИС). Функциональная законченность БИС многокристального МП означает, что его части выполняют заранее определенные функции и могут работать автономно.

На рис. 1.2, а показано функциональное разбиение структуры МП при создании трехкристального микропроцессора (пунктирные линии), содержащие БИС операционного (ОП), БИС управляющего (УП) и БИС интерфейсного (ИП) процессоров.

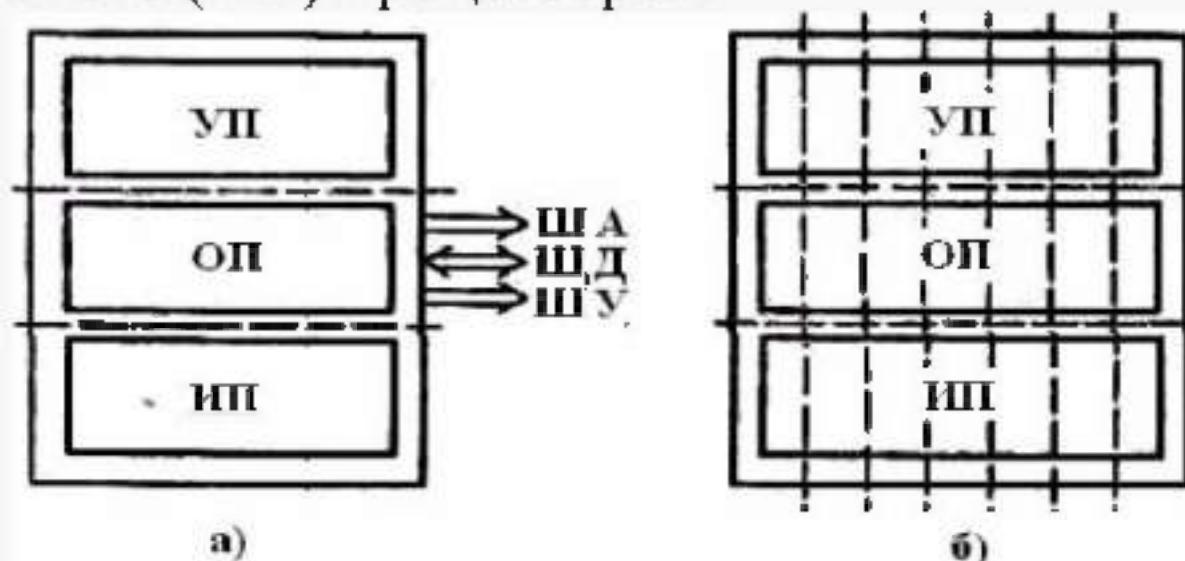


Рисунок 1.2 – Функциональное разбиение структуры МП

Операционный процессор ОП служит для обработки данных, управляющий процессор УП выполняет функции выборки, декодирования и вычисления адресов operandов и также генерирует последовательности микрокоманд.

Автономность работы и большое быстродействие БИС УП позволяет выбирать команды из памяти с большей скоростью, чем скорость их исполнения БИС ОП. При этом в УП образуется очередь еще не исполненных команд, а также заранее подготавливаются те данные, которые потребуются ОП в следующих циклах работы.

Такая опережающая выборка команд экономит время ОП на ожидание операндов, необходимых для выполнения команд программ. Интерфейсный процессор ИП позволяет подключить память и периферийные средства к МП; по существу, является сложным контроллером для устройств ввода — вывода информации. БИС ИП выполняет также функции канала прямого доступа к памяти.

Выбираемые из памяти команды распознаются и выполняются каждой частью МП автономно, и поэтому может быть обеспечен режим одновременной работы всех БИС МП, т. е. конвейерный поточный режим исполнения последовательности команд программы (выполнение последовательности с небольшим временным сдвигом). Такой режим работы значительно повышает его производительность.

Многокристальные секционные МП получаются в том случае, когда в виде БИС реализуются части (секции) логической структуры процессора при функциональном разбиении ее вертикальными плоскостями (рис. 6.1, б). Для построения многоразрядных МП при параллельном включении секций БИС МП в них добавляются средства «стыковки».

Для создания высокопроизводительных многоразрядных МП требуется столь много аппаратурных средств, не реализуемых в доступных БИС, что может возникнуть необходимость еще в функциональном разбиении структуры МП горизонтальными плоскостями. В результате рассмотренного функционального разделения структуры МП на функционально и конструктивно законченные части создаются условия реализации каждой из них в виде БИС. Все они образуют комплект секционных БИС МП.

Таким образом, микропроцессорная секция — это БИС, предназначенная для обработки нескольких разрядов данных или выполнения определенных управляющих операций.

Секционность БИС МП определяет возможность «наращивания» разрядности обрабатываемых данных или усложнения устройств управления микропроцессором при «параллельном» включении большего числа БИС.

С момента создания однокристальные МП развились от простых специализированных 4-разрядных до 32-разрядных процессоров. Трехкристальные МП имеют разрядность до 32 бит и параметры, сравнимые с параметрами старших моделей рядов мини-ЭВМ и средних ЭВМ общего применения.

Многокристальные секционные МП имеют разрядность от 2-4 до 8-32 бит и позволяют создавать разнообразные высокопроизводительные процессоры ЭВМ.

Однокристальные и трехкристальные БИС МП, как правило, изготавливают на основе микроэлектронных технологий униполярных полупроводниковых приборов, а многокристальные секционные БИС МП — на основе технологии биполярных полупроводниковых приборов.

Использование многокристальных микропроцессорных высокоскоростных биполярных БИС, имеющих функциональную законченность при малой физической разрядности обрабатываемых данных и монтируемых в корпус с большим числом выводов, позволяет организовать разветвление связи в процессоре, а также осуществить конвейерные принципы обработки информации для повышения его производительности.

По структурному признаку различают МП с фиксированной разрядностью и МП с наращиваемой разрядностью (секционные МП).

МП с фиксированной разрядностью имеют строго определенную разрядность обрабатываемых слов, величина которой определяется разрядностью МП.

МП с наращиваемой разрядностью позволяют с использованием секций увеличивать число разрядов МПС до требуемой величины, что, как правило, используется при построении миниЭВМ и больших ЭВМ вычислительного типа.

По виду алгоритма работы управляющего устройства МП подразделяют на два вида:

- МП с жестким алгоритмом управления, реализуемые схемно (МП с фиксированным набором команд),
- МП с алгоритмом управления, реализуемые программным путем в виде последовательности микроопераций (МП с микропрограммным управлением).

Здесь система команд определена не жестко, а зависит от микропрограммы, записанной в ПЗУ, входящей в состав устройства управления. Использование микропрограммного управления дает возможность получить необходимый набор команд, например, для воспроизведения (эмulation) набора команд другого МП.

По разрядности обрабатываемой информации МП могут быть 4, 8, 12, 16, 24, 32, 64, 128 - разрядными. На практике наибольшее распространение имеют 32, 64, 128 разрядные МП (Pentium, Celeron, AMD).

По характеру временной организации работы МП делятся на синхронные и асинхронные.

В синхронных МП начало и конец выполнения каждой операции задаются устройством управления, то есть фаза начала и конца выполнения команды строго привязана к временной оси.

В асинхронных МП начало выполнения следующей операции начинается сразу же после окончания выполнения предыдущей операции.

По количество одновременно выполняемых программ различают одно- и многопрограммные МП.

В однопрограммных МП на текущий момент времени выполняется только одна программа. Переход к выполнению другой программы происходит либо по завершению этой программы, либо по специальной команде условного или безусловного перехода, либо по прерыванию.

В многопрограммных МП одновременно может выполняться несколько программ, то есть обеспечивается мультипрограммный режим работы системы.

По виду технологии изготовления разрабатываются и выпускаются БИС МП:

- по униполярной технологии - р - канальные (р - МОП), п - канальные (п - МОП) и комплементарные (КМОП) БИС;
- по биполярной технологии - БИС на базе транзисторно-транзисторной логики (ТТЛ), в том числе и с диодами Шотки (ТТЛШ);
- по эмиттерно-связанной логике (ЭСЛ);
- по интегральной инжекционной логике ( $I^2L$ ).

Вид технологии изготовления БИС во многом определяет степень интеграции микросхем, быстродействие, энергопотребление, помехозащищенность и стоимость МП.

По комплексу этих признаков можно отдать предпочтение МП, выполненным по n - МОП и КМОП технологиям, обеспечивающих высокую плотность компоновки, высокое быстродействие и относительно малую стоимость.

ЭСЛ и ТТЛШ технологии обеспечивают МП самое высокое быстродействие, но микропроцессорные БИС (МП БИС) при этом отличаются самой низкой плотностью компоновки и высоким энергопотреблением.

МП на основе И<sup>2</sup>Л технологии обладают усредненными характеристиками. По плотности компоновки они уступают n - МОП, по быстродействию - ЭСЛ и ТТЛШ, а по стоимости - n - МОП и p - МОП МП. Вместе с тем, p - МОП технология обеспечивает МП наиболее низкую стоимость, но его быстродействие при этом является также наиболее низким.

## Технологии изготовления микропроцессоров

Микропроцессоры создаются на основе кремниевых подложек, которые предварительно вырезаются из слитка чистого кремния. Кремний является полупроводником - в разных условиях он может вести себя и как проводник электрического тока, и как изолятор.

Сначала на подложке под воздействием высокой температуры и кислорода формируется первый слой диоксида кремния. Этот процесс очень похож на возникновение ржавчины на железе, погруженном в воду. Разница заключается в том, что слой диоксида кремния формируется на подложке гораздо быстрее и не виден невооруженным глазом.

Затем подложка покрывается фотослоем. Фотослой под воздействием ультрафиолетового света становится растворимым. Ультрафиолетовое излучение отличается от видимого света меньшей длиной волны. Такое излучение обычно применяется для засвечивания определенного рисунка на верхнем слое кристалла - нанесение схемы на кристалл во многом подобно фотографическому процессу.

В процессе фотолитографии ультрафиолетовое излучение, проходя сквозь маску (которая выполняет функцию шаблона), формирует на подложке рисунок схемы. Засвеченные участки фотослоя становятся растворимыми. Для засветки каждого из слоев микропроцессора применяется своя маска.

Засвеченные участки фотослоя полностью удаляются с помощью растворителя. При этом открывается соответствующая часть слоя диоксида кремния.

Диоксид кремния, не защищенный незасвеченной частью фотослоя, вытравливается химическими препаратами. После этого удаляется оставшаяся часть фотослоя. Таким образом, на кремниевой подложке остается рисунок, выполненный диоксидом кремния.

Чтобы отделить готовый слой от нового, на полученном рисунке схемы выращивается дополнительный тонкий слой диоксида кремния. После этого наносится слой поликристаллического кремния и еще один фотослой. Далее процесс повторяется для второго и следующих слоев.

С помощью процесса ионной имплантации, области кремниевой подложки, обработанные ультрафиолетом, бомбардируются ионами различных примесей. Ионы проникают в подложку, обеспечивая необходимую электрическую проводимость этих областей.

Наложение новых слоев с последующим вытравливанием схемы осуществляется несколько раз, при этом для межслойных соединений в слоях оставляются "окна".

Эти "окна" заполняются атомами металла. После процесса нанесения фотослоя, засветки и вытравливания на кристалле остаются металлические полоски - проводящие области.

Для нанесения проводящих слоев (шин) внутри процессора используются алюминий и медь. Для электрического соединения кремниевой микросхемы с корпусом используется золото.

Таким образом, в современных процессорах устанавливаются связи между примерно 20 слоями, формирующими сложную трехмерную схему. Точное количество слоев может меняться в зависимости от типа процессора.

Производственный цикл состоит из более чем 250 стадий. В результате, на кремниевой пластине формируются сотни идентичных процессоров.

После окончания цикла формирования процессоров все они тщательно тестируются. Затем из пластины-подложки с помощью специального устройства вырезаются конкретные, уже прошедшие проверку кристаллы.

Каждый микропроцессор встраивается в защитный корпус, который также обеспечивает электрическое соединение кристалла микропроцессора с внешними устройствами. Тип корпуса зависит от типа и предполагаемого применения микропроцессора.

В настоящее время лидерами рынка микропроцессоров являются несколько всемирно известных компаний, таких как Intel, AMD (Advanced Micro Devices), HP (Hewlett-Packard Company), Samsung, Sony и др.

## Структура микропроцессорной системы

Микропроцессор – это интегральная схема, смонтированная на небольшой кремниевой пластине.

Микропроцессор как функциональное устройство обеспечивает эффективное автоматическое выполнение операций обработки цифровой информации в соответствии с заданным алгоритмом.

Для решения широкого круга задач в различных областях применений микропроцессор должен обладать алгоритмически полной системой команд (операций).

Минимальная алгоритмически полная система команд процессора состоит из одной или нескольких универсальных команд.

Однако использование процессоров с минимальными по числу операций системами команд ведет к неэкономичному использованию информационных емкостей памяти и значительным затратам времени на выполнение «длинных» программ.

Поэтому обычно в МП встраиваются аппаратные средства, позволяющие реализовать многие десятки и сотни команд. Такие развитые системы команд дают возможность обеспечить компактную запись алгоритмов и соответственно эффективные программы.

Аппаратная реализация сложных команд дает возможность увеличить быстродействие микропроцессора, но требует значительных аппаратных ресурсов кристалла интегральной схемы МП.

Программная реализация сложных команд позволяет обеспечивать программирование сложных задач, изменять количество и особенности исполнения сложных команд.

Скорость исполнения программных команд ниже скорости исполнения аппаратно-реализованных команд.

В микропроцессорной технике используются принципы организации работы и особенности архитектуры классических компьютерных систем.

Любой алгоритм преобразования данных должен быть описан в форме программы - конечной последовательности элементарных операций.

Центральный элемент такой системы - микропроцессор, который выполняет операции преобразования данных, производит ввод и вывод всех необходимых данных, в том числе и кодов самой программы, и управляет взаимодействием всех элементов системы.

Набор операций (выполняемых команд) микропроцессора должен обладать функциональной полнотой, т. е. обеспечивать выполнение необходимого множества операций преобразования данных, управления, обмена данных.

Так как преобразуемые данные и коды программы могут быть представлены в достаточно близких форматах, как правило, строгого разграничения между ними не делают.

Такой подход расширяет возможности программной обработки: данные могут использоваться как элементы кодов программы, а элементы кодов программы можно преобразовывать в процессе работы.

Для хранения данных и кодов программы микропроцессорная система содержит запоминающее устройство.

Стандартная организация микропроцессорной системы предполагает доступность для микропроцессора всех данных в запоминающем устройстве.

В микропроцессорной системе предполагается постоянная готовность запоминающего устройства к обмену данными, для обмена данными микропроцессор должен передать определенный код адреса данных и соответствующие сигналы управления.

Процедуры ввода-вывода данных также должны выполняться стандартными для микропроцессора алгоритмами. Эти алгоритмы, эффективные для управления микропроцессорной системы, могут не соответствовать особенностям работы различных периферийных устройств.

Поэтому микропроцессорная система обычно содержит устройства ввода-вывода, обеспечивающие согласование алгоритмов управления обменом данными.

Компонент микропроцессора – топология линий связи для передачи сигналов между микропроцессором и другими устройствами.

Топология должна быть такой, чтобы изменение структуры микропроцессорной системы не требовало изменения структуры используемых линий связи и формируемых сигналов.

Этому требованию удовлетворяет магистрально-модульный (шинный) принцип организации взаимодействия.

При шинной организации все устройства подключаются параллельно к используемым линиям связи.

Линии связи можно условно объединить в три группы – шины.

По *шине данных (ШД)* передаются требуемые данные, *шина адреса (ША)* служит для выбора устройств, участвующих в обмене данными, а *шина управления (ШУ)* необходима для передачи управляющих сигналов.

Шинная организация максимально универсальна: подключение дополнительных устройств абсолютно не затрагивает уже действующую структуру микропроцессорной системы.

Такая организация накладывает и ограничения на обмен данными.

1. управление обменом данными должно производиться только одним устройством, иначе могут возникнуть конфликты между управляющими устройствами, приводящие к некорректной работе шины.
2. обязательно требуется адресация устройств, параллельное подключение к шине требует выполнения процедур выбора только одного из нескольких устройств для обмена данными.
3. одновременный обмен данными с несколькими устройствами невозможен, если шины в текущий интервал времени уже заняты, передача каких-либо других данных невозможна.

Управляющим устройством системы, в том числе и для шины, обычно является микропроцессор, при нормальной работе он обменивается данными с другими устройствами системы поочередно, адресация используется не только для устройств, но и для данных.

Типичная структура микропроцессорной системы, построенной в соответствии с указанными принципами, приведена на рис. 1.3.

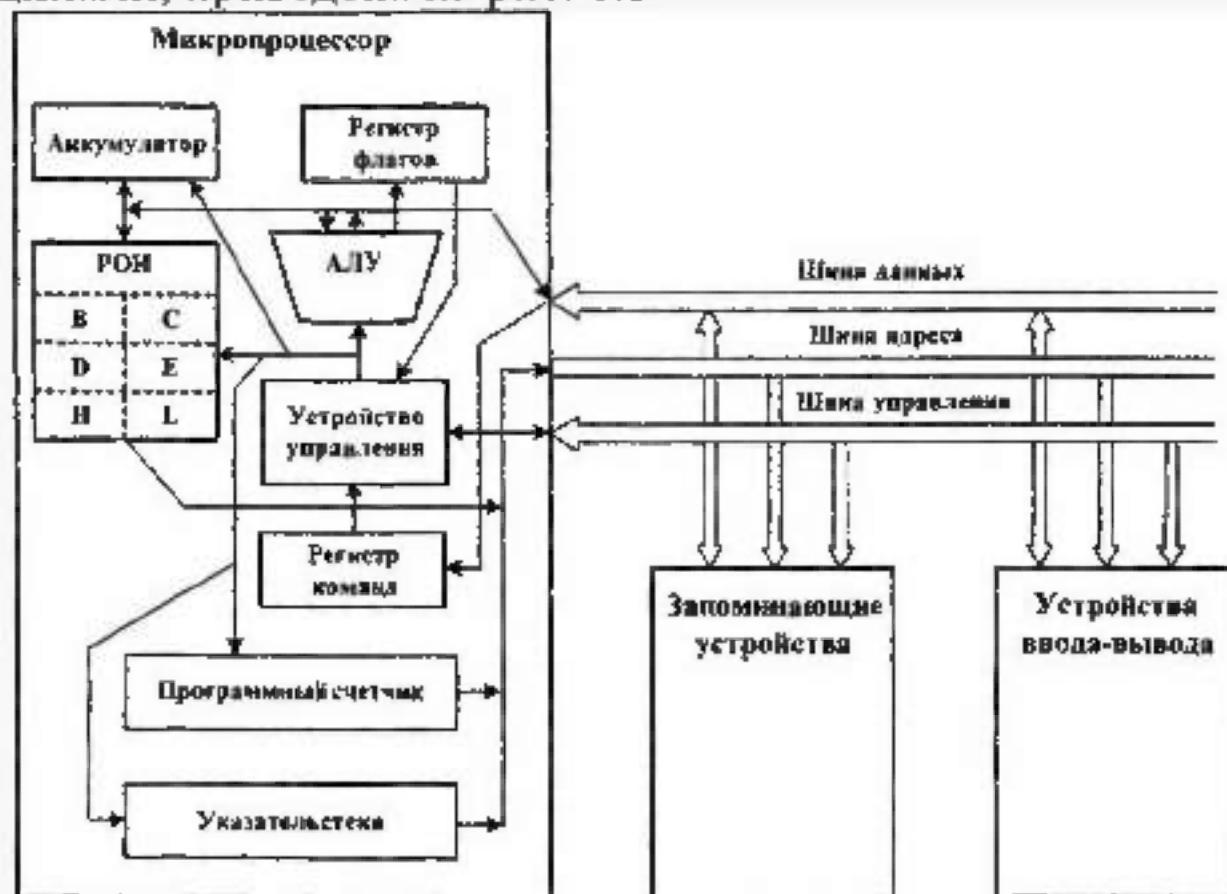


Рис. 1.3. Структурная схема микропроцессорной системы

В структуре устройства ввода-вывода (периферийные устройства) представлены в обобщенном виде, хотя в действительности они могут состоять из разнородных устройств с различными свойствами и реализуемыми функциями.

Данные в параллельном формате в виде отдельных байт (8 бит) могут передаваться из микропроцессора или в микропроцессор по ШД.

Сигналы шины адреса практически всегда формируются микропроцессором и являются адресом (16 бит) байта данных, который микропроцессором передается или принимается.

Сигналы шины управления в основном формируются устройством управления микропроцессора, хотя отдельные сигналы этой шины могут поступать и от других устройств, выполняя, например, функции запросов или подтверждений.

Микропроцессор должен содержать элементы, необходимые для выполнения требуемого набора команд (операций), и представляет собой средство выполнения этих команд,

Без программы (последовательности команд), реализующих заданный алгоритм работы микропроцессорной системы, никакие операции невозможны.

Любые действия в микропроцессоре начинаются с чтения из запоминающего устройства (памяти) кода очередной команды и его ввода в регистр команд.

Поэтому любую микропроцессорную систему необходимо рассматривать как программно-аппаратный комплекс.

Аппаратные (hardware) и программные (software) средства тесно связаны и могут работать только в едином комплексе.

Код команды, поступивший в регистр команд микропроцессора, сигналами устройства управления определяет необходимые функции, как внутренних элементов микропроцессора, так и других устройств микропроцессорной системы через шину управления.

Операции преобразования данных выполняются арифметико-логическим устройством – АЛУ (рис. 1.3).

Набор этих операций обычно стандартный и включает сложение, вычитание, инкремент (увеличение переменной на единицу), декремент (уменьшение переменной на единицу), логические операции И, ИЛИ, исключающее ИЛИ, инверсия и т. п.

В рассматриваемой системе основной формат данных АЛУ - байт (8 бит), АЛУ предназначено только для преобразования и не содержит элементов для хранения данных.

Для хранения входных переменных и результатов преобразования микропроцессор содержит регистры: регистр-аккумулятор и 6 регистров блока регистров общего назначения (РОН).

Одна из входных переменных (операндов) всегда размещается в аккумуляторе, а полученный результат преобразования также всегда направляется в аккумулятор. Второй операнд может храниться в одном из регистров РОН.

В коде команды преобразования обычно указывается регистр, из которого поступает второй операнд, если аккумулятор единственный, его не указывают в коде команды.

Организация микропроцессорной системы, кроме операций преобразования данных, требует выполнения операций пересылки данных, с помощью которых обеспечивается доступ к преобразуемым данным.

Операции пересылки должны производить обмен данных между регистрами микропроцессора, между регистрами микропроцессора и памятью или устройствами ввода-вывода.

Эти операции так же, как и операции преобразования, задаются командами программы с указанием кода операции и адресов приемника и источника для пересылаемых данных.

В рассматриваемой микропроцессорной системе используется память с единым адресным пространством и для данных, и для кодов программы.

Адреса данных содержатся в кодах программы, а адреса кодов программы определяются специальным регистром - *программным счетчиком*.

Разрядность программного счетчика соответствует формату адреса (2 байта), его содержимое определяет адрес ячейки памяти, в которой хранится код очередной команды.

Последовательность выполняемых команд обычно формируется в виде линейной последовательности кодов программы и размещается в ячейках памяти с последовательно нарастающими адресами.

Стандартный цикл работы микропроцессорной системы для очередной команды выполняется следующим образом:

1. По сигналам устройства управления микропроцессора производится считывание и ввод в регистр команд кода очередной команды по адресу, указанному в программном счетчике.
2. Содержимое программного счетчика автоматически инкрементируется (увеличивается на единицу) для определения следующего адреса хранения кодов программы.
3. Если код команды поступил в микропроцессор полностью, команда выполняется; если требуется чтение недостающих элементов кода, повторяется считывание с автоматическим инкрементом адреса в *программном счетчике* (повторение п. п. 1 и 2).
4. Когда очередная команда программы микропроцессорной системы выполнена, в программном счетчике уже содержится адрес следующей команды и начинается следующий рабочий цикл (см. п. 1).

## Архитектурные решения микропроцессоров

Анализируя адресные пространства программ и данных, определяют:

- МП с архитектурой фон Неймана (память программ и память данных находятся в едином пространстве и нет никаких признаков, указывающих на тип информации в ячейке памяти)
- МП с архитектурой Гарвардской лаборатории (память программ и память данных разделены, имеют свои адресные пространства и способы доступа к ним).

На рис. 1.4 показана традиционная структура вычислительной системы, соответствующая "фон-неймановской" вычислительной машине. Американский математик Д. фон Нейман (1903-1957) предложил концепцию вычислительной машины (и в частности, хранимой в памяти программы). Одним из основных моментов этой концепции является то, что система обладает единой памятью, в которой хранятся и команды программы и данные.

Система содержит одну шину данных (ШД), по которой передаются и команды программы и данные и шину адреса. Следовательно, в такой системе требуется три цикла для выборки команды и двух сомножителей, когда реализуется операция умножения (как основная), передаваемых по шине данных.

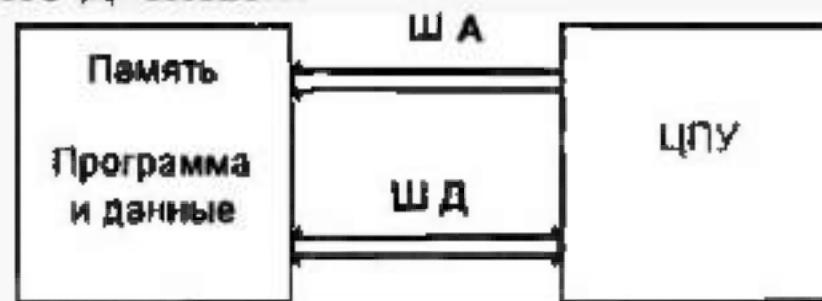


Рис. 1.4 Архитектура фон Неймана

Такая организация работы позволяет производить загрузку и выгрузку управляющих программ в произвольное место памяти, которая в этой структуре не разделяется на память программ и память данных. Любой участок памяти может служить как памятью программ, так и памятью данных. Причём в разные моменты времени одна и та же область памяти может использоваться и как память программ и как память данных.

Для того, чтобы программа могла работать в произвольной области памяти, её необходимо модернизировать перед загрузкой, то есть работать с нею как с обычными данными. Эта особенность архитектуры позволяет наиболее гибко управлять работой микропроцессорной системы. С другой стороны, это создаёт принципиальную возможность искажения управляющей программы, что понижает надёжность работы аппаратуры.

Гарвардская архитектура вычислительной системы приведена на рис. 1.5. Подобная архитектура названа по работе, выполненной в 40-х годах XX века в университете Гарварда под руководством Г. Айкена (1900-1973). В соответствии с этой концепцией для хранения программы (команд) и данных используются различные устройства памяти. Соответственно в системе имеется два комплекта шин для этих устройств: шина адреса памяти программ (ШАПП), шина данных памяти программ для работы с памятью программ (ПП) и шина адреса памяти данных (ШАПД), шина данных памяти данных (ШДПД) для работы с памятью данных (ПД) (рис. 1.5).

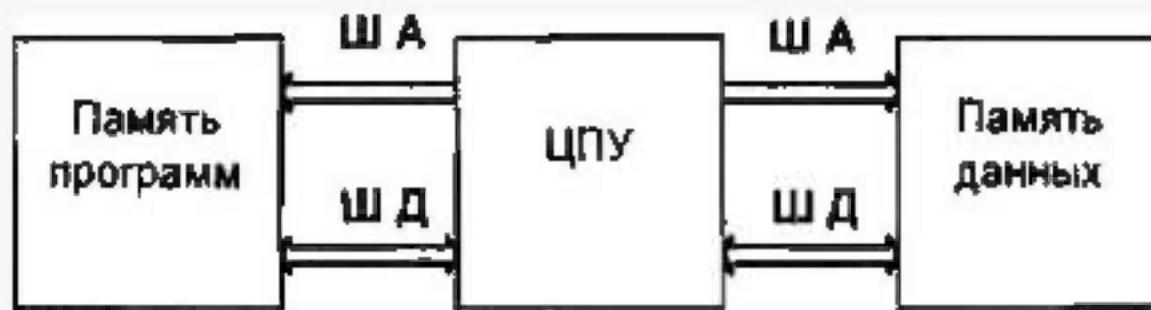


Рис. 1.5. Гарвардская архитектура

В системе с гарвардской архитектурой можно одновременно производить операции обращения к различным устройствам памяти, т. е. синхронно выбирать команду из памяти программ по шине данных памяти программ и сомножитель из памяти данных по шине данных памяти данных и как результат можно отдельно выбирать лишь команды. Также достоинством гарвардской архитектуры является предсказание выборки команд, снижающее частоту загрузок.

Соответственно при этом для выполнения операции умножения требуется два цикла работы процессора. Реально за счет различных дополнительных мер почти всегда время операции умножения сводится к одному циклу.

Следует иметь в виду, что несколько комплектов шин для одновременной выборки данных и команд из памяти данных и памяти программ используются только внутри кристалла микропроцессора при работе с внутренней памятью процессора. Для обращения к внешней памяти во всех процессорах применяется один комплект внешних шин — ВША (внешняя шина адресации) и ВШД (внешняя шина данных).

Так модифицированная гарвардская архитектура использует общую шину данных и шину адреса для всех внешних данных, а внутри процессора использовать шину данных, шину команд и две шины адреса.

Как правило, это определяется ограничениями, накладываемыми технологией на количество выводов интегральной схемы. Поэтому разработчики микропроцессорных систем стремятся использовать только внутреннюю память, и процессоры выпускаются с большой внутренней памятью как память программ, так память данных.

При использовании внешней памяти, особенно для хранения программы и данных, увеличивается время, затрачиваемое на выполнение операций.

## Классификация архитектурных решений

Основные типы архитектурных решений, с учетом способов адресации памяти.

1. Регистровая архитектура определяется наличием достаточно большого регистрового файла внутри МП. Команды получают возможность обратиться к операндам, расположенным в одной из двух запоминающих сред: оперативной памяти или регистрах. Размер регистра обычно фиксирован и совпадает с размером слова, физически реализованного в оперативной памяти. К любому регистру можно обратиться непосредственно, поскольку регистры представлены в виде массива запоминающих элементов - регистрового файла. Типичным является выполнение арифметических операций только в регистре, при этом команда содержит два операнда (оба операнда в регистре или один операнд в регистре, а второй в оперативной памяти).

К данному типу архитектуры относится микропроцессор Z80 (фирма Zilog). В нем помимо расширенной системы команд, одного номинала питания и способности исполнять программы, написанные для intel 8080.

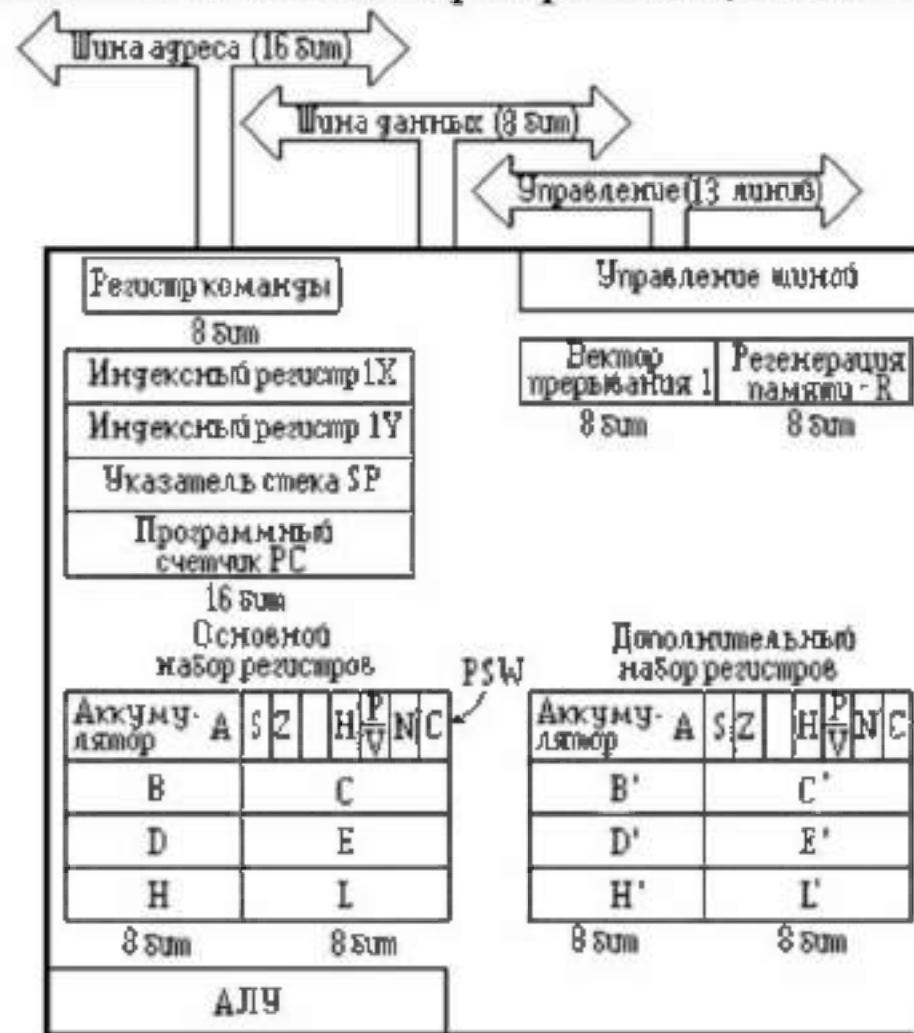


Рис. 1.6. Микропроцессор Z80 фирмы Zilog

В дополнение к основному набору РОН, в кристалле был реализован второй комплект аналогичных регистров. Это значительно упрощало работу при вызове подпрограмм или процедур обслуживания прерываний, поскольку программист мог использовать для них альтернативный набор регистров, избегая сохранения в стеке содержимого РОНов для основной программы с помощью операций PUSH (поместить операнд в стек).

Кроме того, в систему команд был включен ряд специальных инструкций, ориентированных на обработку отдельных битов, а для поддержки регенерации динамической памяти в схему процессора введены соответствующие аппаратные средства.

МП фирмы Motorola имел ряд существенных преимуществ. Прежде всего, кристалл MC6800 требовал для работы одного номинала питания, а система команд оказалась весьма прозрачной для программиста.

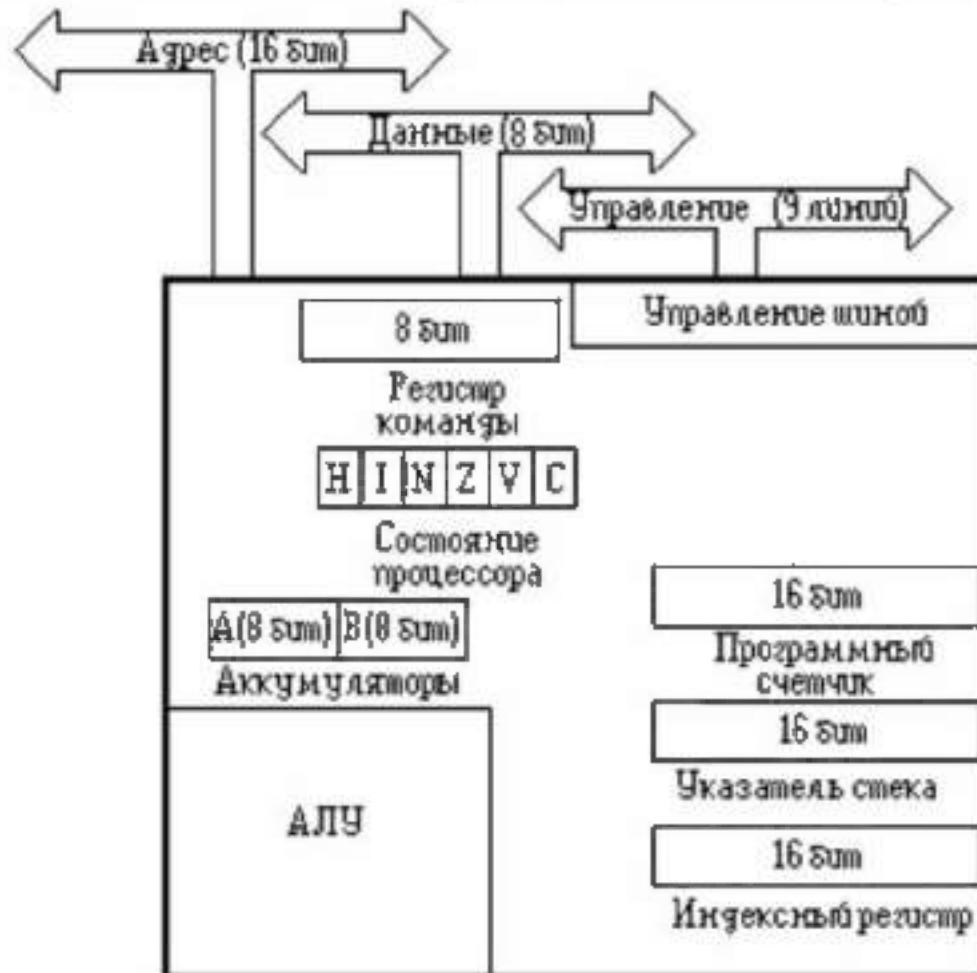


Рис 1.7. Микропроцессор MC6800 фирмы Motorola

Микропроцессор МС 6800 содержал два аккумулятора, и результат операции АЛУ мог быть помещен в любой из них. Но самым ценным качеством структуры МС 6800 было автоматическое сохранение в стеке содержимого всех регистров процессора при обработке прерываний (Z80 требовалось для этого несколько команд PUSH). Процедура восстановления РОН из стека тоже выполнялась аппаратно.

2. Стековая архитектура дает возможность создать поле памяти с упорядоченной последовательностью записи и выборки информации.

В общем случае команды неявно адресуются к элементу стека, расположенному на его вершине, или к двум верхним элементам стека.

3. Архитектура МП, ориентированная на оперативную память (типа "память-память"), обеспечивает высокую скорость работы и большую информационную емкость рабочих регистров и стека при их организации в оперативной памяти.

Архитектура этого типа не предполагает явного определения аккумулятора, регистров общего назначения или стека; все операнды команд адресуются к области основной памяти.

## Устройство управления

Принцип управления. Организация устройства управления. Устройство управления микропроцессора. Особенности программного и микропрограммного управления.

### Принцип управления

Принцип микропрограммного управления был предложен М. Уилксом в 1951 г и с того времени практически не претерпел изменений – любое электронное устройство может быть представлено в виде пары:

1. устройство управления
2. операционное устройство

На долю управляющего устройства приходится полный контроль работы операционного устройства (сюда входит анализ текущего состояния устройства, контроль параметров, принятие решений и т.д.).

На долю операционного устройства приходится вся "основная работа": непосредственно осуществление полученных команд и преобразование контролируемых величин (в том числе и своего состояния) в удобный для управляющего устройства вид.

## Организация устройств управления

Самым старым, но актуальным, является организация управляющего устройства в виде конечного автомата, т. е. в виде устройства обладающего памятью, и результат работы которого зависит от его текущего состояния.

Схематически пару управляющее устройство — операционное устройство можно представить следующим образом — рис. 5.1.

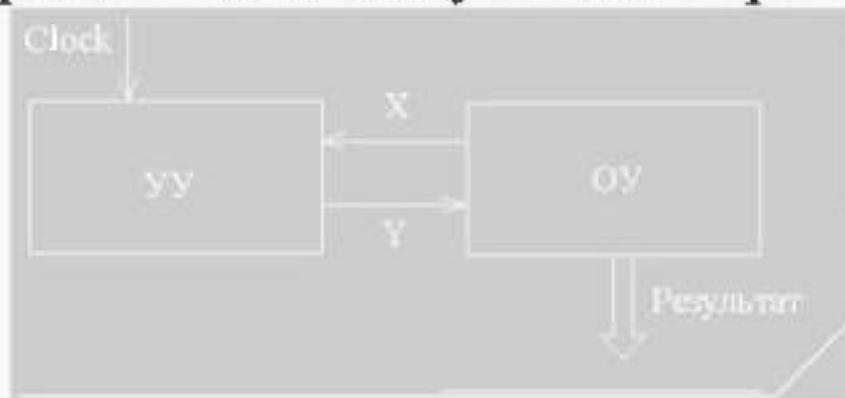


Рис. 5.1. Пара управляющее устройство — операционное устройство

УУ (управляющее устройство) получает сигналы о текущем состоянии — вектор  $X$ , и выдает управляющие сигналы — вектор  $Y$ , переход из одного состояния в другое происходит по тактовым импульсам —  $Clock$ .

Операционный автомат контролирует поведение чего-то, а может быть только себя с выдачей результатов работы, на основе полученных сигналов.

На текущий момент можно выделить три группы управляющих автоматов:

- автоматы с жесткой логикой, например Мили (Mealy) и Мура (Moog);
- автоматы с программируемой логикой (на сегодняшний день изжили себя как класс и могут рассматриваться лишь в историческом контексте): автоматы с принудительной адресацией, естественной адресацией и соответственно комбинированной;
- композиционные устройства управления — объединяют преимущества двух предыдущих типов устройств.

## Устройство управления микропроцессора

Другие устройства управления — микропроцессоры и микроконтроллеры. Микропроцессоры используются в стационарных решениях и там где требуется большая вычислительная мощность, микроконтроллеры — используются в портативных маломощных (как по питанию так и по вычислительной мощности) решениях; причем обладают дополнительным специфическим набором функций (в зависимости от специализации).

Типичную структурную схему устройства, использующего вышеназванное решение можно увидеть на рис. 5.2.

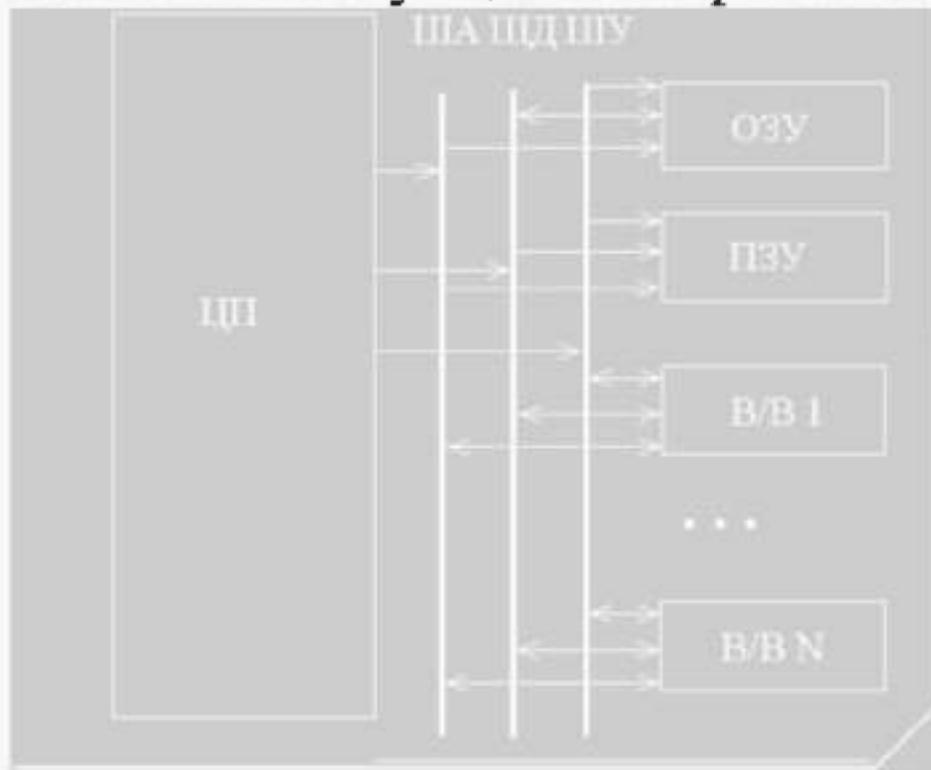


Рис. 5.2. Типичная схема микропроцессорного устройства

ЦП — собственно процессорный элемент; ША, ШД, ШУ — шины адреса, данных, управления соответственно; ОЗУ — текущая память (обычно используется для хранения результатов работы); ПЗУ — содержит набор команд определяющих работу ЦП; В/В 1 ... В/В N — устройства ввода вывода.

Для микроконтроллера, эта схема будет несколько отличаться: так например память находится на одном кристалле с ЦП (а так же все прочие непоказанные устройства — контроллер памяти, портов ввода/вывода, прерываний и т.д.) устройства подключенные к портам ввода/вывода не могут самостоятельно управлять шиной и т.п.

Программирование поведения микропроцессора (МП) /микроконтроллера (МК) выполняется записью необходимой информации в ПЗУ со стартового адреса. Собственно в этом и заключается еще одно отличие МК и МП — поведение МК практически полностью определяется содержимым его ПЗУ (в большинстве своем они в состоянии выполнять команды только из адресного пространства ПЗУ); МП же могут изменять свое поведение загружая различные программы в ОЗУ, что делает их более гибкими.

Однако это не является недостатком МК, т. к. они по определению являются компактными устройствами и не могут иметь большой объем памяти.

К положительным сторонам МК и МП можно отнести сравнительную простоту разработки решений на их базе — наличие большого числа стандартных схем сопровождения, простоту программирования, наличия специализированных программных продуктов для них, стандартность большинства решений (для проектирования большого числа проектов управляющая часть не понесет никаких изменений).

К недостаткам можно отнести то, что даже при разработке минимальной схемы необходимо использовать весь необходимый минимум микросхем (различные контроллеры и т.д.).

Еще одна система управления — управление с помощью ЭВМ.

Коды операции команд программы, воспринимаемые управляющей частью микропроцессора, расшифрованные и преобразованные в ней, дают информацию о том, какие операции надо выполнить, где в памяти расположены данные, куда надо направить результат и где расположена следующая за выполняемой командой.

Управляющее устройство имеет достаточно средств для того, чтобы после восприятия и интерпретации информации, получаемой в команде, обеспечить переключение (срабатывание) всех требуемых функциональных частей машины, а также для того, чтобы подвести к ним данные и воспринять полученные результаты.

Именно срабатывание, т. е. изменение состояния двоичных логических элементов на противоположное, позволяет посредством коммутации вентилей выполнять элементарные логические и арифметические действия, а также передавать требуемые операнды в функциональные части микроЭВМ.

Такт - минимальный рабочий интервал, в течение которого совершается одно элементарное действие; цикл - интервал времени, в течение которого выполняется одна машинная операция

Устройство управления в строгой последовательности в рамках тактовых и цикловых временных интервалов работы микропроцессора осуществляет:

- выборку команды;
- интерпретацию ее с целью анализа формата, служебных признаков и вычисления адреса операнда (операндов);
- установление номенклатуры и временной последовательности всех функциональных управляющих сигналов;
- генерацию управляющих импульсов и передачу их на управляющие шины функциональных частей микроЭВМ и вентили между ними;
- анализ результата операции и изменение своего состояния так, чтобы определить месторасположение (адрес) следующей команды.

## Особенности программного и микроуправления

В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов: программный и микропрограммный.

Выполнение операций в машине сводится к элементарным преобразованиям информации (передача информации между узлами в блоках, сдвиг информации в узлах, логические поразрядные операции, проверка условий и т.д.) в логических элементах, узлах и блоках под воздействием функциональных управляющих сигналов блоков (устройств) управления.

Элементарные преобразования, неразложимые на более простые, выполняются в течение одного такта сигналов синхронизации и называются микрооперациями.

В аппаратных (схемных) устройствах управления каждой операции соответствует свой набор логических схем, вырабатывающих определенные функциональные сигналы для выполнения микроопераций в определенные моменты времени.

При аппаратных построении устройства управления реализация микроопераций достигается за счет однажды соединенных между собой логических схем, поэтому ЭВМ с аппаратным устройством управления называют ЭВМ с жесткой логикой управления.

Это понятие относится к фиксации системы команд в структуре связей ЭВМ и означает практическую невозможность каких-либо изменений в системе команд ЭВМ после ее изготовления.

При микропрограммной реализации устройства управления в состав последнего вводится запоминающее устройство (ЗУ), каждый разряд выходного кода которого определяет появление определенного функционального сигнала управления.

Каждой микрооперации ставится в соответствие свой информационный код - микрокоманда. Набор микрокоманд и последовательность их реализации обеспечивают выполнение любой сложной операции. Набор микроопераций называют микропрограммами.

При микропрограммной реализации используется способ управления операциями путем:

- последовательного считывания и интерпретации микрокоманд из ЗУ (наиболее часто в виде микропрограммного ЗУ используют быстродействующие программируемые логические матрицы),
- использования кодов микрокоманд для генерации функциональных управляющих сигналов,

При этом способ управления называют микропрограммным, а микроЭВМ с таким способом управления - микропрограммными или с хранимой (гибкой) логикой управления.

К микропрограммам предъявляют требования функциональной полноты и минимальности. Первое требование необходимо для обеспечения возможности разработки микропрограмм любых машинных операций, а второе связано с желанием уменьшить объем используемого оборудования.

В целом, принцип микропрограммного управления (ПМУ) включает следующие позиции:

- 1) любая операция, реализуемая устройством, является последовательностью элементарных действий - микроопераций;
- 2) для управления порядком следования микроопераций используются логические условия;
- 3) процесс выполнения операций в устройстве описывается в форме алгоритма, представляемого в терминах микроопераций и логических условий, называемого микропрограммой;
- 4) микропрограмма используется как форма представления функции устройства, на основе которой определяются структура и порядок функционирования устройства во времени.

## Режимы работы микропроцессора

Режимы работы микропроцессора. Защищенный режим. Реальный режим. Режим системного управления. Переключение между режимами.

Впервые о различных режимах работы процессоров стали говорить с появлением процессора 80286. Это был первый представитель данного семейства процессоров, в котором были реализованы многозадачность и защищенная архитектура. Чтобы обеспечить совместимость с предыдущими представителями этого семейства (8086/88, 80186/188) в процессоре 80286 было реализовано два режима функционирования:

- режим эмуляции 8086 (режим реального адреса)
- защищенный режим, в котором используются все возможности процессора.

В последующих поколениях процессоров этого семейства защищенный режим становится основным режимом работы.

В новых поколениях процессоров Intel появился еще один режим работы - режим системного управления.

## Защищенный режим (Protected Mode)

Основным режимом работы микропроцессора является защищенный режим. Ключевыми особенностями защищенного режима являются: виртуальное адресное пространство, защита и многозадачность.

В защищенном режиме программа оперирует адресами, которые могут относиться к физически отсутствующим ячейкам памяти, поэтому такое адресное пространство называется виртуальным.

Размер виртуального адресного пространства программы может превышать емкость физической памяти и достигать 64 Тбайт.

Для адресации виртуального адресного пространства используется сегментированная модель, в которой адрес состоит из двух элементов: селектора сегмента и смещения внутри сегмента. С каждым сегментом связана особая структура, хранящая информацию о нем - дескриптор.

Встроенные средства переключения задач обеспечивают многозадачность в защищенном режиме. Среда задачи состоит из содержимого регистров МП и всего кода с данными в пространстве памяти.

Микропроцессор способен быстро переключаться из одной среды выполнения в другую, имитируя параллельную работу нескольких задач.

Для некоторых задач может эмулироваться управление памятью как у процессора 8086. Такое состояние задачи называется виртуальным режимом 8086 (Virtual 8086 Mode).

## Реальный режим (Real Mode)

В реальном режиме микропроцессор работает очень быстро с возможностью использования 32-битных расширений. Механизм адресации, размеры памяти и обработка прерываний (с их последовательными ограничениями) МП Intel386 в реальном режиме полностью совпадают с аналогичными функциями МП 8086. В отличие от 8086 микропроцессоры серии 286+ в определенных ситуациях генерируют исключения, например, при превышении предела сегмента, который для всех сегментов в реальном режиме - OFFFFh.

Имеются две фиксированные области в памяти, которые резервируются в режиме реальной адресации:

- область инициализации системы
- область таблицы прерываний

## Режим системного управления (System Management Mode)

Режим системного управления предназначен для выполнения некоторых действий с возможностью их полной изоляции от прикладного программного обеспечения и даже операционной системы. Переход в этот режим возможен только аппаратно.

При входе в режим SMM процессор сохраняет свой контекст в SMRAM (специальная часть системного ОЗУ) и передает управление процедуре, называемой обработчиком System Management Interrupt.

В режиме системного управления не предусмотрена работа с прерываниями. При возврате из SMM процессор восстанавливает свой контекст из SMRAM.

Особенности режима системного управления позволяют использовать его для реализации системы управления энергосбережением или функций безопасности и контроля доступа.

## Принципы формирования адресного пространства

Среда выполнения задачи. Архитектурные решения микропроцессоров.

### Среда выполнения задачи

Любая запущенная задача оперирует определенным набором ресурсов для исполнения инструкций, сохранения данных и состояния задачи. Этот набор ресурсов называется средой выполнения. Функционирование некоторых элементов среды выполнения микропроцессора зависит от режима работы микропроцессора.

К среде выполнения задачи относят:

- адресное пространство (размер адресного пространства и способы его адресации зависят от режима работы);
- регистры (регистры общего назначения, сегментные регистры, регистры флагов, указатель команды);
- регистры сопроцессора (регистры данных, регистр управления, регистр статуса, указатель команды, указатель операнда);
- стек (для реализации вложенных подпрограмм и передачи параметров между ними).

Кроме перечисленных ресурсов, среда выполнения микропроцессора содержит так называемые системные ресурсы, предназначенные для поддержки операционных систем и системного программного обеспечения:

- пространство портов ввода-вывода (обеспечивает взаимодействие с периферийными устройствами);
- управляющие регистры (определяют режим работы процессора и состояние задачи);
- регистры управления памятью и дескрипторные таблицы (используются в защищенном режиме);
- регистры отладки;
- прочие регистры (MTRRs, MSRs и др.)

## Система адресации

Режимы адресации. Способы адресации. Режимы адресации operandов.

Возможности микропроцессоров по адресации.

### Режимы адресации

Для взаимодействия с различными модулями должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т. е. однозначная комбинация бит.

Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда - отдельные адресные пространства регистров устройств ввода/вывода и внутренней памяти.

Кроме того, память хранит как данные, так и команды. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти - это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес.

## Способы адресации

Все способы адресации памяти можно разделить на:

1. прямой, когда исполнительный адрес берется непосредственно из команды или вычисляется с использованием значения, указанного в команде, и содержимого какого-либо регистра (прямая адресация, регистровая, базовая, индексная и т.д.);
2. косвенный, который предполагает, что в команде содержится значение косвенного адреса, т.е. адреса ячейки памяти, в которой находится окончательный исполнительный адрес (косвенная адресация).

В каждом МП реализованы только некоторые режимы адресации, использование которых определяется архитектурой МП.

Инструкция микропроцессора может содержать следующие поля:

префикс	КОП	Mod R/M	SIB	смещение	непосредственный операнд
0/1 байт	1/2 байта	0/1 байт	0/1 байт	0/1/2/4 байта	0/1/2/4 байта

**Префикс** - необязательная часть инструкции, позволяет изменить некоторые особенности ее выполнения. В команде может быть использовано сразу несколько префиксов разного типа. Типы префиксов:

- командные префиксы (префиксы повторения);
- префикс блокировки шины LOCK;
- префиксы размера;
- префиксы замены сегмента.

Байт "**Mod R/M**" определяет режим адресации, а также иногда дополнительный код операции. Необходимость байта "Mod R/M" зависит от типа инструкции.

Байт **SIB** (Scale-Index-Base) определяет способ адресации при обращении к памяти в 32-битном режиме. Необходимость байта SIB зависит от режима адресации, задаваемого полем "Mod R/M".

Кроме того, инструкция может содержать непосредственный операнд и/или смещение операнда в сегменте данных.

На размер инструкции накладывается ограничение в 15 байт. Инструкция большего размера может получиться при некорректном использовании большого количества префиксов.

## Режимы адресации operandов

Если для инструкции микропроцессора требуются operandы, то они могут задаваться следующими способами:

- непосредственно в коде инструкции (только operand-источник);
- в одном из регистров;
- через порт ввода-вывода;
- в памяти.

Непосредственный режим адресации подразумевает включение операнда-источника в код инструкции. Операнд может быть 8-битным или 16-битным, если значение эффективного размера операнда - 16. Операнд может быть 8-битным или 32-битным, если значение эффективного размера операнда - 32. Обычно непосредственные операнды используются в арифметических инструкциях.

Регистровый режим адресации определят операнд-источник или операнд-приемник в одном из следующих регистров:

- регистры общего назначения
- сегментные регистры
- регистр флагов
- управляющие регистры
- регистры отладки
- машинно-зависимые регистры
- регистры сопrocessора

Адресация через порт ввода-вывода подразумевает получение операнда или сохранение операнда через пространство портов ввода-вывода. Адрес порта ввода-вывода либо непосредственно включается в код инструкции, либо берется из специального регистра.

Очень распространенный способ адресации операнда - адресация через память. Таким образом, может быть указан операнд-источник или операнд-приемник. Процессор не позволяет одновременно задавать оба операнда через память (за исключением некоторых цепочек команд).

Для получения операнда из памяти процессору необходимо знать селектор сегмента и смещение в сегменте. В некоторых командах селектор может быть указан непосредственно в коде инструкции. В других случаях процессор может явно или неявно использовать значение одного из сегментных регистров. Под неявным использованием сегментных регистров подразумевается то, что в зависимости от предназначения операнда процессор использует определенный сегментный регистр для обращения к памяти.

## Возможности микропроцессоров по адресации

Одной из важнейших архитектурных характеристик МП является перечень возможных способов обращения к памяти или видов адресации. Возможности МП по адресации существенны с двух точек зрения.

Во-первых, большой объем памяти требует большой длины адреса, так как  $n$ -разрядный адрес позволяет обращаться к памяти емкостью  $2^n$  слов. Типовые 8-разрядные слова МП дают возможность непосредственно обращаться только к 256 ячейкам памяти, что явно недостаточно.

Если учесть, что обращение к памяти является наиболее часто встречающейся операцией, то очевидно, что эффективность использования МП во многом определяется способами адресации к памяти большого объема при малой разрядности МП.

Во-вторых, для удобства программирования желательно иметь простую систему формирования адресов данных при работе с массивами, таблицами и указателями.

Способы решения:

Если адресное поле в команде является ограниченным и недостаточным для непосредственного обращения к любой ячейке памяти, то память в таких случаях разбивают на страницы, где страницей считается  $2^n$  ячеек памяти.

Для согласования адресного поля команды малой разрядности с памятью большого объема (для решения “страничной” проблемы) в МП применяются различные виды адресации:

**1. Прямая адресация к текущей странице.** При такой адресации программный счетчик разбивается на два поля; старшие разряды указывают номер страницы, а младшие - адрес ячейки на странице. В адресном поле команды размещается адрес ячейки на странице, а адрес страницы должен быть установлен каким-то другим способом, например с помощью специальной команды.

## **2. Прямая адресация с использованием страничного регистра.**

В МП должен быть предусмотрен программно доступный страничный регистр, загружаемый специальной командой. Этот регистр добавляет к адресному полю команды несколько разрядов, необходимых для адресации ко всей памяти.

## **3. Прямая адресация с использованием двойных слов.** Для увеличения длины адресного поля команды под адрес отводится дополнительное слово (при необходимости может использоваться больше).

## **4. Адресация относительно программного счетчика.** Адресное поле команды рассматривается как целое со знаком, которое складывается с содержимым программного счетчика для формирования исполнительного адреса. Такой способ относительной адресации создает плавающую страницу и упрощает перемещение программ в памяти.

## 5. Адресация относительно индексного регистра.

Исполнительный адрес образуется суммированием содержимого индексного регистра и адресного поля команды, рассматриваемого как целое со знаком. Индексный регистр загружается специальными командами.

**6. Косвенная адресация.** При косвенной адресации в адресном поле команды указывается адрес на текущей странице, по которому хранится исполнительный адрес. В поле команды при этом требуется дополнительный разряд – признак косвенной адресации. Исполнительный адрес может храниться не в ячейке памяти, а в регистре общего назначения. В этом случае косвенная адресация называется регистровой.

## Устройство управления

Принцип управления. Организация устройства управления. Устройство управления микропроцессора. Особенности программного и микропрограммного управления.

### Принцип управления

Принцип микропрограммного управления был предложен М. Уилксом в 1951 г и с того времени практически не претерпел изменений – любое электронное устройство может быть представлено в виде пары:

1. устройство управления
2. операционное устройство

На долю управляющего устройства приходится полный контроль работы операционного устройства (сюда входит анализ текущего состояния устройства, контроль параметров, принятие решений и т.д.).

На долю операционного устройства приходится вся "основная работа": непосредственно осуществление полученных команд и преобразование контролируемых величин (в том числе и своего состояния) в удобный для управляющего устройства вид.

## Организация устройств управления

Самым старым, но актуальным, является организация управляющего устройства в виде конечного автомата, т. е. в виде устройства обладающего памятью, и результат работы которого зависит от его текущего состояния.

Схематически пару управляющее устройство — операционное устройство можно представить следующим образом — рис. 5.1.

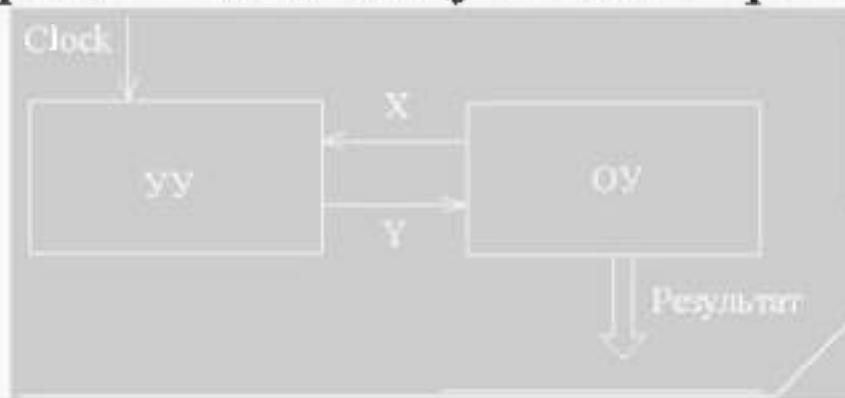


Рис. 5.1. Пара управляющее устройство — операционное устройство

УУ (управляющее устройство) получает сигналы о текущем состоянии — вектор X, и выдает управляющие сигналы — вектор Y, переход из одного состояния в другое происходит по тактовым импульсам — Clock

Операционный автомат контролирует поведение чего-то, а может быть только себя с выдачей результатов работы, на основе полученных сигналов.

На текущий момент можно выделить три группы управляющих автоматов:

- автоматы с жесткой логикой, например Мили (Mealy) и Мура (Moog);
- автоматы с программируемой логикой (на сегодняшний день изжили себя как класс и могут рассматриваться лишь в историческом контексте): автоматы с принудительной адресацией, естественной адресацией и соответственно комбинированной;
- композиционные устройства управления — объединяют преимущества двух предыдущих типов устройств.

## Устройство управления микропроцессора

Другие устройства управления — микропроцессоры и микроконтроллеры. Микропроцессоры используются в стационарных решениях и там где требуется большая вычислительная мощность, микроконтроллеры — используются в портативных маломощных (как по питанию так и по вычислительной мощности) решениях; причем обладают дополнительным специфическим набором функций (в зависимости от специализации).

Типичную структурную схему устройства, использующего вышеназванное решение можно увидеть на рис. 5.2.

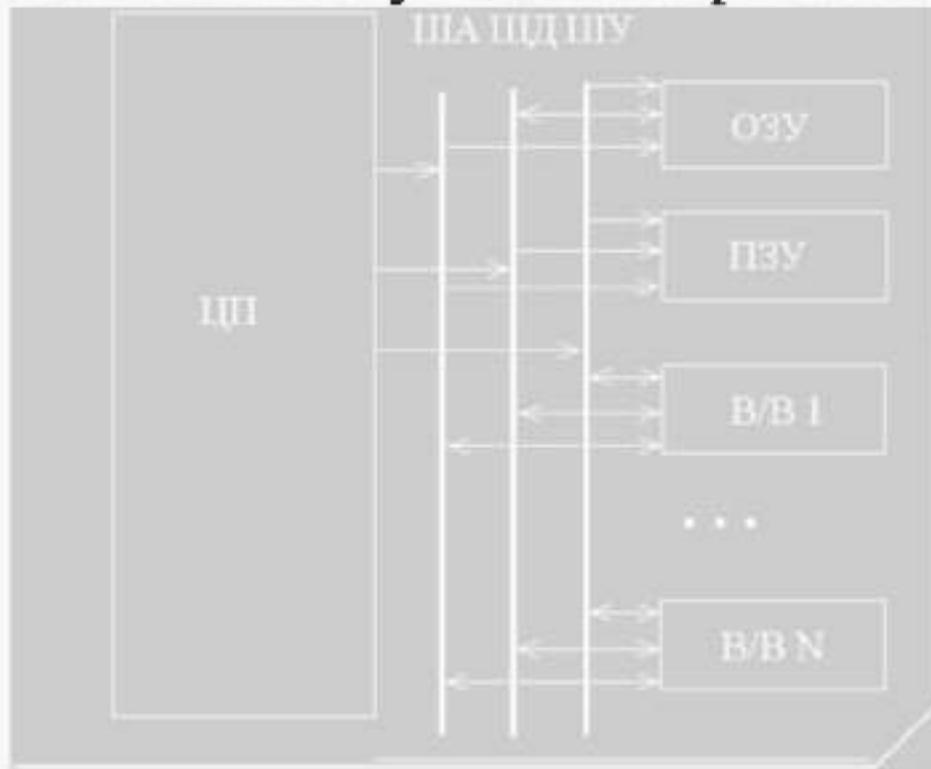


Рис. 5.2. Типичная схема микропроцессорного устройства

ЦП — собственно процессорный элемент; ША, ШД, ШУ — шины адреса, данных, управления соответственно; ОЗУ — текущая память (обычно используется для хранения результатов работы); ПЗУ — содержит набор команд определяющих работу ЦП; В/В 1 ... В/В N — устройства вывода.

Для микроконтроллера, эта схема будет несколько отличаться: так например память находится на одном кристалле с ЦП (а так же все прочие непоказанные устройства — контроллер памяти, портов ввода/вывода, прерываний и т.д.) устройства подключенные к портам ввода/вывода не могут самостоятельно управлять шиной и т.п.

Программирование поведения микропроцессора (МП) /микроконтроллера (МК) выполняется записью необходимой информации в ПЗУ со стартового адреса. Собственно в этом и заключается еще одно отличие МК и МП — поведение МК практически полностью определяется содержимым его ПЗУ (в большинстве своем они в состоянии выполнять команды только из адресного пространства ПЗУ); МП же могут изменять свое поведение загружая различные программы в ОЗУ, что делает их более гибкими.

Однако это не является недостатком МК, т. к. они по определению являются компактными устройствами и не могут иметь большой объем памяти.

К положительным сторонам МК и МП можно отнести сравнительную простоту разработки решений на их базе — наличие большого числа стандартных схем сопровождения, простоту программирования, наличия специализированных программных продуктов для них, стандартность большинства решений (для проектирования большого числа проектов управляющая часть не понесет никаких изменений).

К недостаткам можно отнести то, что даже при разработке минимальной схемы необходимо использовать весь необходимый минимум микросхем (различные контроллеры и т.д.).

Еще одна система управления — управление с помощью ЭВМ.

Коды операции команд программы, воспринимаемые управляющей частью микропроцессора, расшифрованные и преобразованные в ней, дают информацию о том, какие операции надо выполнить, где в памяти расположены данные, куда надо направить результат и где расположена следующая за выполняемой командой.

Управляющее устройство имеет достаточно средств для того, чтобы после восприятия и интерпретации информации, получаемой в команде, обеспечить переключение (срабатывание) всех требуемых функциональных частей машины, а также для того, чтобы подвести к ним данные и воспринять полученные результаты.

Именно срабатывание, т. е. изменение состояния двоичных логических элементов на противоположное, позволяет посредством коммутации вентилей выполнять элементарные логические и арифметические действия, а также передавать требуемые операнды в функциональные части микроЭВМ.

Такт - минимальный рабочий интервал, в течение которого совершается одно элементарное действие; цикл - интервал времени, в течение которого выполняется одна машинная операция

Устройство управления в строгой последовательности в рамках тактовых и цикловых временных интервалов работы микропроцессора осуществляет:

- выборку команды;
- интерпретацию ее с целью анализа формата, служебных признаков и вычисления адреса операнда (операндов);
- установление номенклатуры и временной последовательности всех функциональных управляющих сигналов;
- генерацию управляющих импульсов и передачу их на управляющие шины функциональных частей микроЭВМ и вентили между ними;
- анализ результата операции и изменение своего состояния так, чтобы определить месторасположение (адрес) следующей команды.

## Особенности программного и микроуправления

В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов: программный и микропрограммный.

Выполнение операций в машине сводится к элементарным преобразованиям информации (передача информации между узлами в блоках, сдвиг информации в узлах, логические поразрядные операции, проверка условий и т.д.) в логических элементах, узлах и блоках под воздействием функциональных управляющих сигналов блоков (устройств) управления.

Элементарные преобразования, неразложимые на более простые, выполняются в течение одного такта сигналов синхронизации и называются микрооперациями.

В аппаратных (схемных) устройствах управления каждой операции соответствует свой набор логических схем, вырабатывающих определенные функциональные сигналы для выполнения микроопераций в определенные моменты времени.

При аппаратных построении устройства управления реализация микроопераций достигается за счет однажды соединенных между собой логических схем, поэтому ЭВМ с аппаратным устройством управления называют ЭВМ с жесткой логикой управления.

Это понятие относится к фиксации системы команд в структуре связей ЭВМ и означает практическую невозможность каких-либо изменений в системе команд ЭВМ после ее изготовления.

При микропрограммной реализации устройства управления в состав последнего вводится запоминающее устройство (ЗУ), каждый разряд выходного кода которого определяет появление определенного функционального сигнала управления.

Каждой микрооперации ставится в соответствие свой информационный код - микрокоманда. Набор микрокоманд и последовательность их реализации обеспечивают выполнение любой сложной операции. Набор микроопераций называют микропрограммами.

При микропрограммной реализации используется способ управления операциями путем:

- последовательного считывания и интерпретации микрокоманд из ЗУ (наиболее часто в виде микропрограммного ЗУ используют быстродействующие программируемые логические матрицы),
- использования кодов микрокоманд для генерации функциональных управляющих сигналов,

При этом способ управления называют микропрограммным, а микроЭВМ с таким способом управления - микропрограммными или с хранимой (гибкой) логикой управления.

К микропрограммам предъявляют требования функциональной полноты и минимальности. Первое требование необходимо для обеспечения возможности разработки микропрограмм любых машинных операций, а второе связано с желанием уменьшить объем используемого оборудования.

В целом, принцип микропрограммного управления (ПМУ) включает следующие позиции:

- 1) любая операция, реализуемая устройством, является последовательностью элементарных действий - микроопераций;
- 2) для управления порядком следования микроопераций используются логические условия;
- 3) процесс выполнения операций в устройстве описывается в форме алгоритма, представляемого в терминах микроопераций и логических условий, называемого микропрограммой;
- 4) микропрограмма используется как форма представления функции устройства, на основе которой определяются структура и порядок функционирования устройства во времени.

## Режимы работы микропроцессора

Режимы работы микропроцессора. Защищенный режим. Реальный режим. Режим системного управления. Переключение между режимами.

Впервые о различных режимах работы процессоров стали говорить с появлением процессора 80286. Это был первый представитель данного семейства процессоров, в котором были реализованы многозадачность и защищенная архитектура. Чтобы обеспечить совместимость с предыдущими представителями этого семейства (8086/88, 80186/188) в процессоре 80286 было реализовано два режима функционирования:

- режим эмуляции 8086 (режим реального адреса)
- защищенный режим, в котором используются все возможности процессора.

В последующих поколениях процессоров этого семейства защищенный режим становится основным режимом работы.

В новых поколениях процессоров Intel появился еще один режим работы - режим системного управления.

## Защищенный режим (Protected Mode)

Основным режимом работы микропроцессора является защищенный режим. Ключевыми особенностями защищенного режима являются: виртуальное адресное пространство, защита и многозадачность.

В защищенном режиме программа оперирует адресами, которые могут относиться к физически отсутствующим ячейкам памяти, поэтому такое адресное пространство называется виртуальным.

Размер виртуального адресного пространства программы может превышать емкость физической памяти и достигать 64 Тбайт.

Для адресации виртуального адресного пространства используется сегментированная модель, в которой адрес состоит из двух элементов: селектора сегмента и смещения внутри сегмента. С каждым сегментом связана особая структура, хранящая информацию о нем - дескриптор.

Встроенные средства переключения задач обеспечивают многозадачность в защищенном режиме. Среда задачи состоит из содержимого регистров МП и всего кода с данными в пространстве памяти.

Микропроцессор способен быстро переключаться из одной среды выполнения в другую, имитируя параллельную работу нескольких задач.

Для некоторых задач может эмулироваться управление памятью как у процессора 8086. Такое состояние задачи называется виртуальным режимом 8086 (Virtual 8086 Mode).

## Реальный режим (Real Mode)

В реальном режиме микропроцессор работает очень быстро с возможностью использования 32-битных расширений. Механизм адресации, размеры памяти и обработка прерываний (с их последовательными ограничениями) МП Intel386 в реальном режиме полностью совпадают с аналогичными функциями МП 8086. В отличие от 8086 микропроцессоры серии 286+ в определенных ситуациях генерируют исключения, например, при превышении предела сегмента, который для всех сегментов в реальном режиме - OFFFFh.

Имеются две фиксированные области в памяти, которые резервируются в режиме реальной адресации:

- область инициализации системы
- область таблицы прерываний

## Режим системного управления (System Management Mode)

Режим системного управления предназначен для выполнения некоторых действий с возможностью их полной изоляции от прикладного программного обеспечения и даже операционной системы. Переход в этот режим возможен только аппаратно.

При входе в режим SMM процессор сохраняет свой контекст в SMRAM (специальная часть системного ОЗУ) и передает управление процедуре, называемой обработчиком System Management Interrupt.

В режиме системного управления не предусмотрена работа с прерываниями. При возврате из SMM процессор восстанавливает свой контекст из SMRAM.

Особенности режима системного управления позволяют использовать его для реализации системы управления энергосбережением или функций безопасности и контроля доступа.

## Принципы формирования адресного пространства

Среда выполнения задачи. Архитектурные решения микропроцессоров.

### Среда выполнения задачи

Любая запущенная задача оперирует определенным набором ресурсов для исполнения инструкций, сохранения данных и состояния задачи. Этот набор ресурсов называется средой выполнения. Функционирование некоторых элементов среды выполнения микропроцессора зависит от режима работы микропроцессора.

К среде выполнения задачи относят:

- адресное пространство (размер адресного пространства и способы его адресации зависят от режима работы);
- регистры (регистры общего назначения, сегментные регистры, регистры флагов, указатель команды);
- регистры сопроцессора (регистры данных, регистр управления, регистр статуса, указатель команды, указатель операнда);
- стек (для реализации вложенных подпрограмм и передачи параметров между ними).

Кроме перечисленных ресурсов, среда выполнения микропроцессора содержит так называемые системные ресурсы, предназначенные для поддержки операционных систем и системного программного обеспечения:

- пространство портов ввода-вывода (обеспечивает взаимодействие с периферийными устройствами);
- управляющие регистры (определяют режим работы процессора и состояние задачи);
- регистры управления памятью и дескрипторные таблицы (используются в защищенном режиме);
- регистры отладки;
- прочие регистры (MTRRs, MSRs и др.)

## Система адресации

Режимы адресации. Способы адресации. Режимы адресации operandов.

Возможности микропроцессоров по адресации.

### Режимы адресации

Для взаимодействия с различными модулями должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т. е. однозначная комбинация бит.

Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда - отдельные адресные пространства регистров устройств ввода/вывода и внутренней памяти.

Кроме того, память хранит как данные, так и команды. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти - это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес.

## Способы адресации

Все способы адресации памяти можно разделить на:

1. прямой, когда исполнительный адрес берется непосредственно из команды или вычисляется с использованием значения, указанного в команде, и содержимого какого-либо регистра (прямая адресация, регистровая, базовая, индексная и т.д.);
2. косвенный, который предполагает, что в команде содержится значение косвенного адреса, т.е. адреса ячейки памяти, в которой находится окончательный исполнительный адрес (косвенная адресация).

В каждом МП реализованы только некоторые режимы адресации, использование которых определяется архитектурой МП.

Инструкция микропроцессора может содержать следующие поля:

префикс	КОП	Mod R/M	SIB	смещение	непосредственный операнд
0/1 байт	1/2 байта	0/1 байт	0/1 байт	0/1/2/4 байта	0/1/2/4 байта

**Префикс** - необязательная часть инструкции, позволяет изменить некоторые особенности ее выполнения. В команде может быть использовано сразу несколько префиксов разного типа. Типы префиксов:

- командные префиксы (префиксы повторения);
- префикс блокировки шины LOCK;
- префиксы размера;
- префиксы замены сегмента.

Байт "**Mod R/M**" определяет режим адресации, а также иногда дополнительный код операции. Необходимость байта "Mod R/M" зависит от типа инструкции.

Байт **SIB** (Scale-Index-Base) определяет способ адресации при обращении к памяти в 32-битном режиме. Необходимость байта SIB зависит от режима адресации, задаваемого полем "Mod R/M".

Кроме того, инструкция может содержать непосредственный операнд и/или смещение операнда в сегменте данных.

На размер инструкции накладывается ограничение в 15 байт. Инструкция большего размера может получиться при некорректном использовании большого количества префиксов.

## Режимы адресации operandов

Если для инструкции микропроцессора требуются operandы, то они могут задаваться следующими способами:

- непосредственно в коде инструкции (только operand-источник);
- в одном из регистров;
- через порт ввода-вывода;
- в памяти.

Непосредственный режим адресации подразумевает включение операнда-источника в код инструкции. Операнд может быть 8-битным или 16-битным, если значение эффективного размера операнда - 16. Операнд может быть 8-битным или 32-битным, если значение эффективного размера операнда - 32. Обычно непосредственные операнды используются в арифметических инструкциях.

Регистровый режим адресации определят операнд-источник или операнд-приемник в одном из следующих регистров:

- регистры общего назначения
- сегментные регистры
- регистр флагов
- управляющие регистры
- регистры отладки
- машинно-зависимые регистры
- регистры сопrocessора

Адресация через порт ввода-вывода подразумевает получение операнда или сохранение операнда через пространство портов ввода-вывода. Адрес порта ввода-вывода либо непосредственно включается в код инструкции, либо берется из специального регистра.

Очень распространенный способ адресации операнда - адресация через память. Таким образом, может быть указан операнд-источник или операнд-приемник. Процессор не позволяет одновременно задавать оба операнда через память (за исключением некоторых цепочек команд).

Для получения операнда из памяти процессору необходимо знать селектор сегмента и смещение в сегменте. В некоторых командах селектор может быть указан непосредственно в коде инструкции. В других случаях процессор может явно или неявно использовать значение одного из сегментных регистров. Под неявным использованием сегментных регистров подразумевается то, что в зависимости от предназначения операнда процессор использует определенный сегментный регистр для обращения к памяти.

## Возможности микропроцессоров по адресации

Одной из важнейших архитектурных характеристик МП является перечень возможных способов обращения к памяти или видов адресации. Возможности МП по адресации существенны с двух точек зрения.

Во-первых, большой объем памяти требует большой длины адреса, так как  $n$ -разрядный адрес позволяет обращаться к памяти емкостью  $2^n$  слов. Типовые 8-разрядные слова МП дают возможность непосредственно обращаться только к 256 ячейкам памяти, что явно недостаточно.

Если учесть, что обращение к памяти является наиболее часто встречающейся операцией, то очевидно, что эффективность использования МП во многом определяется способами адресации к памяти большого объема при малой разрядности МП.

Во-вторых, для удобства программирования желательно иметь простую систему формирования адресов данных при работе с массивами, таблицами и указателями.

Способы решения:

Если адресное поле в команде является ограниченным и недостаточным для непосредственного обращения к любой ячейке памяти, то память в таких случаях разбивают на страницы, где страницей считается  $2^n$  ячеек памяти.

Для согласования адресного поля команды малой разрядности с памятью большого объема (для решения “страничной” проблемы) в МП применяются различные виды адресации:

**1. Прямая адресация к текущей странице.** При такой адресации программный счетчик разбивается на два поля; старшие разряды указывают номер страницы, а младшие - адрес ячейки на странице. В адресном поле команды размещается адрес ячейки на странице, а адрес страницы должен быть установлен каким-то другим способом, например с помощью специальной команды.

## **2. Прямая адресация с использованием страницного регистра.**

В МП должен быть предусмотрен программно доступный страницный регистр, загружаемый специальной командой. Этот регистр добавляет к адресному полю команды несколько разрядов, необходимых для адресации ко всей памяти.

## **3. Прямая адресация с использованием двойных слов.** Для увеличения длины адресного поля команды под адрес отводится дополнительное слово (при необходимости может использоваться больше).

## **4. Адресация относительно программного счетчика.** Адресное поле команды рассматривается как целое со знаком, которое складывается с содержимым программного счетчика для формирования исполнительного адреса. Такой способ относительной адресации создает плавающую страницу и упрощает перемещение программ в памяти.

## 5. Адресация относительно индексного регистра.

Исполнительный адрес образуется суммированием содержимого индексного регистра и адресного поля команды, рассматриваемого как целое со знаком. Индексный регистр загружается специальными командами.

**6. Косвенная адресация.** При косвенной адресации в адресном поле команды указывается адрес на текущей странице, по которому хранится исполнительный адрес. В поле команды при этом требуется дополнительный разряд – признак косвенной адресации. Исполнительный адрес может храниться не в ячейке памяти, а в регистре общего назначения. В этом случае косвенная адресация называется регистровой.

## Память

Внутренняя память компьютера. Организация физической памяти. ОЗУ, ПЗУ и FLASH-память.  
Логическая структура основной памяти.

Внутренняя память представлена в виде отдельных интегральных микросхем (ИМС) собственно памяти и элементов, включенных в состав других ИМС, не выполняющих непосредственно функцию хранения программ и данных – это и внутренняя память центрального процессора, и видеопамять, и контроллеры различных устройств.

Для создания элементов запоминающих устройств, в основном, применяют СБИС со структурой МОП (металл-оксид-полупроводник).

Для функционирования системы необходимо наличие как оперативного запоминающего устройства (ОЗУ), так и постоянного запоминающего устройства (ПЗУ), обеспечивающего сохранение информации при выключении питания.

ОЗУ может быть статическим и динамическим, а ПЗУ однократно или многократно программируемым.

## Организация физической памяти.

Физическая память, к которой микропроцессоры имеют доступ по шине адреса, называется *оперативной памятью* (или оперативным запоминающим устройством – ОЗУ).

ОП организована как последовательность байтов. Каждому байту соответствует уникальный адрес (его номер), который называется *физическим адресом*.

Диапазон значений адресов зависит от разрядности шины адреса микропроцессора.

Механизм управления памятью полностью аппаратный, т.е. программа сама не может сформировать физический адрес памяти на адреснойшине.

## ОЗУ, ПЗУ и FLASH-память

Оперативное запоминающее устройство предназначено для хранения информации (программ и данных) и непосредственно участвующей в вычислительном процессе.

ОЗУ – энергозависимая память: при отключении напряжения питания информация, хранящаяся в ней, теряется. Основу ОЗУ составляют большие интегральные схемы, содержащие матрицы полупроводниковых запоминающих элементов (триггеров).

Запоминающие элементы расположены на пересечении вертикальных и горизонтальных шин матрицы; запись и считывание информации осуществляются подачей электрических импульсов по тем шинам матрицы, которые соединены с элементами, принадлежащими выбранной ячейке памяти.

Постоянное запоминающее устройство также строится на основе установленных на материнской плате модулей и используется для хранения неизменяемой информации: загрузочных программ операционной системы, программ тестирования устройств компьютера и некоторых драйверов базовой системы ввода-вывода. ПЗУ – энергонезависимое запоминающее устройство.

Из ПЗУ можно только считывать информацию, запись информации в ПЗУ выполняется особыми способами. Модули ПЗУ имеют емкость, как правило, не превышающую нескольких сот килобайт.

FLASH-память – полупостоянные перепрограммируемые запоминающие устройства. Модули или карты FLASH-памяти могут устанавливаться прямо в разъемы платы МК. FLASH-память – энергонезависимое запоминающее устройство.

Для перезаписи информации необходимо подать на специальный вход FLASH-памяти напряжение программирования (12В), что исключает возможность случайного стирания информации. Перепрограммирование FLASH-памяти может выполняться при наличии; специального контроллера либо с внешнего программатора, подключаемого к ПК.

FLASH-память может быть полезной как для создания весьма быстродействующих компактных, альтернативных запоминающих устройств - "твердотельных дисков", так и для замены ПЗУ, хранящего программы BIOS, позволяя обновлять и заменять эти программы на более новые версии.

## Логическая структура основной памяти.

Каждая ячейка памяти имеет свой уникальный (отличный от всех других) адрес. Основная память имеет для ОЗУ и ПЗУ единое адресное пространство.

Адресное пространство определяет максимально возможное количество непосредственно адресуемых ячеек основной памяти.

Адресное пространство зависит от разрядности адресных шин, ибо максимальное количество разных адресов определяется разнообразием двоичных чисел, которые можно отобразить в  $n$  разрядах, т. е. адресное пространство равно  $2^n$ , где  $n$  - разрядность адреса.

Основная память в соответствии с методами доступа и адресации делится на отдельные, иногда частично или полностью перекрывающие друг друга области, имеющие общепринятые названия.

Основная память компьютера делится на две логические области: непосредственно адресуемую память, занимающую первые 1024 Кбайта ячеек с адресами от 0 до 1024 Кбайт-1, расширенную память, доступ к ячейкам которой возможен при использовании специальных программ-драйверов.

Стандартной памятью (СМА - Conventional Memory Area) называется непосредственно адресуемая память в диапазоне от 0 до 640 Кбайт.

Непосредственно адресуемая память в диапазоне адресов от 640 до 1024 Кбайт называется верхней памятью (УМА - Upper Memory Area). Верхняя память зарезервирована для видсопамяти и постоянного запоминающего устройства. Однако обычно в ней остаются свободные участки - "окна", которые могут быть использованы в качестве оперативной памяти общего назначения.

Расширенная память - это память с адресами 1024 Кбайта и выше.

## Прерывания

Понятие прерывания. Классификация прерываний. Внешние прерывания. Система прерываний. Аппаратные и программные средства системы прерываний. Обработка прерываний в реальном режиме.

Прерывание означает временное прекращение основного процесса вычислений для выполнения некоторых запланированных или незапланированных действий, вызываемых работой аппаратуры или программы.

Таким образом, это процесс, временно переключающий микропроцессор на выполнение другой программы с последующим возвратом к прерванной программе.

Пример: Нажимая клавишу на клавиатуре, мы инициируем немедленный вызов программы, которая распознает клавишу, заносит ее код в буфер клавиатуры, из которого он считывается другой программой.

Т.е. на некоторое время микропроцессор прерывает выполнение текущей программы и переключается на программу обработки прерывания, так наз. обработчик прерывания. После того, как обработчик прерывания завершит свою работу, прерванная программа продолжит выполнение с точки, где было приостановлено ее выполнение.

Адрес программы-обработчика прерывания вычисляется по таблице векторов прерываний, содержащий всю необходимую информацию для перехода к обработчику, в том числе его начальный адрес. Механизм прерываний поддерживается на аппаратном уровне.

## Классификация прерываний.

В зависимости от источника, прерывания делятся на:

- аппаратные - возникают как реакция микропроцессора на физический сигнал от некоторого устройства (клавиатура, системные часы, клавиатура, жесткий диск и т.д.), по времени возникновения эти прерывания асинхронны, т.е. происходят в случайные моменты времени;
- программные - вызываются искусственно с помощью соответствующей команды из программы (int), предназначены для выполнения некоторых действий операционной системы, являются синхронными;
- исключения - являются реакцией микропроцессора на нестандартную ситуацию, возникшую внутри микропроцессора во время выполнения некоторой команды программы (например, деление на ноль).

## Общая классификация прерываний

**внешние** - вызываются внешними по отношению к микропроцессору событиями (по существу - это группа аппаратных прерываний) Вложенных прерываний нет!

**внутренние** - возникают внутри микропроцессора во время вычислительного процесса (по существу - это исключительные ситуации и программные прерывания).

## **Внешние прерывания.**

Внешние прерывания возникают по сигналу какого-нибудь внешнего устройства.

Внешние прерывания подразделяются на **немаскируемые** и **маскируемые**.

В связи с тем, что существуют два специальных внешних сигнала среди входных сигналов процессора, при помощи которых можно прервать выполнение текущей программы и тем самым переключить работу центрального процессора. Это сигналы NMI (по mask interrupt, немаскируемое прерывание) и INTR (interrupt request, запрос на прерывание).

**Маскируемые прерывания** генерируются контроллером прерываний по заявке определенных периферийных устройств. Контроллер прерываний поддерживает восемь уровней (линий) приоритета; к каждому уровню “привязано” одно периферийное устройство. Именно маскируемые прерывания часто называют аппаратными прерываниями.

**Немаскируемые прерывания** (говорят, что оно одно, т.к. подается на вывод микропроцессора NMI) инициируют источники, требующие безотлагательного вмешательства со стороны микропроцессора.

## Система прерываний

### Аппаратные и программные средства системы прерываний

Система прерываний - это совокупность программных и аппаратных средств, реализующих механизм прерываний.

К *аппаратным средствам* системы прерываний относятся:

- выводы микропроцессора - на них формируются сигналы, извещающие микропроцессор либо о том, что некоторое внешнее устройство «просит уделить ему внимание» (INTR), либо о том, что требуется безотлагательная обработка некоторого события или катастрофическая ошибка (NMI).

о INTR - вывод для входного сигнала запроса на прерывание,

о NMI - вывод для входного сигнала немаскируемого прерывания  
о INTA - вывод для выходного сигнала подтверждения получения  
сигнала прерывания микропроцессором

- программируемый контроллер прерываний 8259А (предназначен для фиксирования сигналов прерываний от восьми различных внешних устройств; он выполнен в виде микросхемы; обычно используют две последовательно соединенные микросхемы, поэтому кол-во возможных источников внешних прерываний до 15 плюс одно немаскируемое прерывание; именно он формирует номер вектора прерывания и выдает его шину данных);
- внешние устройства (таймер, клавиатура, магнитные диски и т.п.)

*К программным средствам системы прерываний реального режима относятся:*

Таблица векторов прерываний инициализируется при запуске системы, но в принципе может быть изменена и перемещена.

Каждый вектор имеет свой номер и называется номером прерывания.

- **два флага в регистре флагов flags/eflags:**

IF (Interrupt Flag) - флаг прерывания. Предназначен для маскирования (запрещения) аппаратных прерываний. Если IF=1, микропроцессор обрабатывает внешние прерывания, если = 0, то игнорирует;

TF(Trace Flag) - флаг трассировки. Если TF=1, то микропроцессор переходит в режим покомандной работы. В этом режиме в микропроцессоре генсирируется внутреннее прерывание с номером 1;

- **машинные команды микропроцессора: int, into (прерывание по переполнению), iret, cli, sti**

## Обработка прерывания в реальном режиме

Обработка прерывания в реальном режиме производится в три этапа:

- 1) Прекращение выполнения текущей программы;

Должно произойти так, чтобы потом вернуться и продолжить работу. Для этого необходимо сохранить содержимое регистров, так как они являются ресурсами, разделяемыми между программами.

Наиболее удобным местом хранения регистров является стек.

После сохранения регистров в стеке микропроцессор сбрасывает бит флага IF (т.е.=0)

Этим предотвращается возможность возникновения вложенных внешних прерываний и порча регистров исходной программы вследствие неконтролируемых действий со стороны программы - обработчика вложенного прерывания.

## 2) Переход к выполнению и выполнение программы обработки прерывания;

Здесь определяется источник прерывания и вызывается соответствующий обработчик прерывания.

Затем выполняется сама программа обработки прерывания. Она тоже может быть прервана поступлением запроса от более приоритетного источника. Все источники прерывания имеют приоритеты.

## 3) Возврат управления прерванной программе.

Необходимо привести стек в состояние, в котором он был сразу после передачи управления данной процедуре. Для этого программист должен указать необходимые действия по восстановлению регистров и очистке стека.

## Система команд микропроцессора

Оптимальная система команд. Формат команды. Классификация команд.  
Примеры кодирования команд

### Оптимальная система команд.

Проектирование системы команд оказывает влияние на структуру МП. Оптимальную систему команд иногда определяют как совокупность команд, которая удовлетворяет требованиям проблемно-ориентированных применений таким образом, что избыточность аппаратных и аппаратно-программных средств на реализацию редко используемых команд оказывается минимальной.

В различных программах частота появления команд различна. Систему команд следует выбирать таким образом, чтобы затраты на редко используемые команды были минимальными.

При наличии статистических данных можно разработать (выбрать) МП с эффективной системой команд. Одним из подходов к достижению данной цели является разработка команд длиной в одно слово и кодирование их таким образом, чтобы разряды таких коротких

команд использовать оптимально, что позволит сократить время реализации программы и ее длину.

Другим подходом к оптимизации системы команд является использование микроинструкций. В этом случае отдельные биты или группы бит команды используются для кодирования нескольких элементарных операций, которые выполняются в одном командном цикле. Эти элементарные операции не требуют обращения к памяти, а последовательность их реализации определяется аппаратной логикой.

Сокращение времени выполнения программ и емкости памяти достигается за счет увеличения сложности логики управления.

## Формат команды

Важной характеристикой команды является ее формат, определяющий структурные элементы команды, каждый из которых интерпретируется определенные образом при ее выполнении. Среди таких элементов (полей) команды выделяют следующие:

- код операции, определяющий выполняемое действие;
- адрес ячейки памяти, регистра процессора, внешнего устройства;
- режим адресации; операнд при использовании непосредственной адресации;
- код анализируемых признаков для команд условного перехода.

## Классификация команд

Классификация команд по основным признакам представлена на рисунке

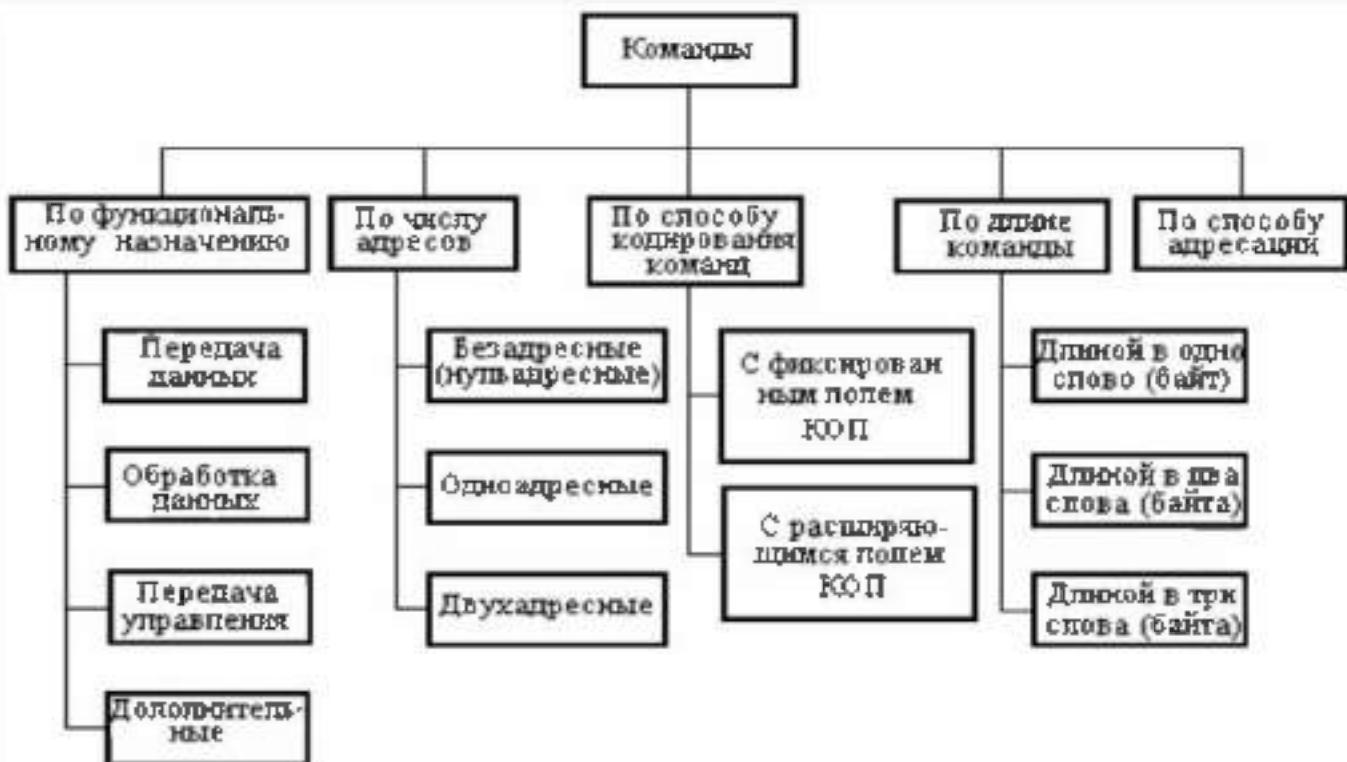


Рисунок. Классификация команд

Важнейшим структурным элементом формата любой команды является код операции (КОП), определяющей действие, которое должно быть выполнено.

Большое число КОП в процессоре очень важно, так как аппаратная реализация команд экономит память и время. Но при выборе ЭВМ необходимо концентрировать внимание на полноте операций с конкретными типами данных, а не только на числе команд, на доступных режимах адресации.

Число бит, отводимое под КОП, является функцией полного набора реализуемых команд.

В некоторых командах необходим только один operand и они называются однооперандными (или одноадресными) командами в отличие от двухoperandных (или двухадресных), в которых требуются два операнда. При наличии двух operandов командой обычно изменяется только один из них.

Так как информация берется только из одной ячейки, эту ячейку называются источником; ячейка, содержимое которой изменяется, называется приемником.

Ниже приведен формат двухадресной (двухоперандной) команды процессоров СМ.

а	15 11	10 6	5 0	6	15 11	10 0	
	КОП	Источник	Приемник		КОП	Приемник	

Формат команд процессоров СМ: а) двухадресная команда; б) одноадресная команда.

## Архитектуры системы команд (RISC, CISC, VLIW, MISC, EPIC)

### Архитектура CISC

Архитектура CISC (Complete Instruction Set Computer) - архитектура процессора с полным набором инструкций. К ним относится семейство процессоров 80x86. Процессоры с CISC архитектурой команд (80 x 86) имеют весьма сложную систему команд.

Для CISC-процессоров характерно:

- сравнительно небольшое число регистров общего назначения;
- большое количество машинных команд, некоторые из которых нагружены семантически аналогично операторам высокогоуровневых языков программирования и выполняются за много тиков;
- большое количество методов адресации;
- большое количество форматов команд различной разрядности;
- преобладание двухадресного формата команд;
- наличие команд обработки типа регистр-память.

В результате широкий набор команд усложняет декодирование инструкций, на что расходуются аппаратные ресурсы. Возрастает число тактов, необходимое для их выполнения.

Современные CISC-процессоры применяют ряд технологий ускорения выполнения кода, заимствованных у систем RISC (гибридная архитектура: CISC-процессор имеет RISC-ядро).

Типичными представителями являются процессоры на основе x86 команд (исключая современные Intel Pentium 4, AMD Athlon, которые являются гибридными).

## Архитектура RISC

RISC (Reduced Instruction Set Computing) — вычисления с сокращённым набором команд.

Это философия проектирования процессоров, которая во главу ставит следующий принцип: более компактные и простые инструкции выполняются быстрее. Простая архитектура позволяет как удешевить процессор, так и поднять тактовую частоту. Многие ранние RISC-процессоры даже не имели команд умножения и деления.

Первые RISC-процессоры были разработаны в начале 1980-х годов в Стэнфордском и Калифорнийском университетах США. Они выполняли небольшой (50 - 100) набор команд, тогда как обычные CISC (Complex Instruction Set computer) выполняли 100-200.

## Характерные особенности RISC-процессоров:

- Фиксированная длина машинных инструкций (например, 32 бита) и простой формат команды.
  - Ориентация системы на поддержку языка высокого уровня с помощью развитого компилятора;
  - Использование примитивного набора инструкций, который полностью реализуется аппаратными средствами. Одна инструкция выполняет только одну операцию с памятью — чтение или запись. Операции вида "прочитать-изменить-записать" отсутствуют.
  - Большое количество регистров общего назначения (32 и более).
  - Организация памяти и ввода-вывода, которая позволяет выполнять процессором большинство инструкций за один такт.

В настоящее время многие архитектуры процессоров являются RISC-подобными, к примеру, ARM, DEC Alpha, SPARC, AVR, MIPS, POWER и PowerPC. Наиболее широко используемые в настольных компьютерах процессоры архитектуры x86 ранее являлись CISC-процессорами, однако новые процессоры непосредственно перед

исполнением преобразуют CISC- инструкции процессоров x86 в более простой набор внутренних инструкций RISC.

Концепция RISC - архитектуры базируется на почти очевидной логической формуле: если «быстрые» технологии и параллельная обработка для всего списка команд недостижимы из-за высокого уровня затрат, то надо ускорять только часто выполняемые операции, а редко применяемыми и сложными следует пожертвовать ради повышения общей производительности.

Регистры - основное достоинство и главная проблема RISC. Все существующие RISC-процессоры базируются на единственном типе обработки данных в формате «регистр-регистр», а точнее, «регистр-регистр-регистр». Это позволяет без существенных затрат времени выбрать операнды из адресуемых оперативных регистров и записать в регистр результат операции.

Кроме того, трехоперандные операции дают компилятору большую гибкость по сравнению с типовыми двухместными операциями формата «регистр-память» архитектуры CISC. В сочетании с быстродействующей арифметикой RISC-операции типа «регистр-регистр» становятся очень мощным средством повышения производительности процессора. Проблема заключается в том, что в процессе выполнения задачи RISC-система неоднократно вынуждена обновлять содержимое регистров процессора, причем за минимальное время, чтобы не вызвать длительных простоев арифметического устройства (а это прямые потери производительности). Для CISC-систем подобной проблемы не существует, поскольку модификация регистров может происходить на фоне обработки команд формата «память - память».

Существует два подхода к решению проблемы модификации регистров в RISC - архитектуре: аппаратный, предложенный в проектах RISC университета в Беркли, и программный, разработанный специалистами IBM Станфордского университета. Принципиальная разница между ними заключается в том, что аппаратное решение основано на стремлении уменьшить время вызова процедур за счет установки дополнительного оборудования процессора, тогда как программное базируется на возможностях компилятора и является более экономичным с точки зрения аппаратуры центрального процессора.

Среди других особенностей RISC-архитектур, появившихся несколько позже, следует отметить идею суперскалярной или многоконвейерной обработки, внеочередное выполнение команд и появление "смешанных" или групповых команд для сокращения времени выполнения часто повторяющихся последовательностей.

## Архитектура VLIW

VLIW (Very long instruction word — «очень длинная машинная команда») — архитектура процессоров с несколькими вычислительными устройствами. Характеризуется тем, что одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно.

В суперскалярных процессорах также есть несколько вычислительных модулей, но задача распределения между ними работы решается аппаратно. Это сильно усложняет дизайн процессора, и может быть чревато ошибками. В процессорах VLIW задача распределения решается во время компиляции и в инструкциях явно указано, какое вычислительное устройство должно выполнять какую команду.

VLIW можно считать логическим продолжением идеологии RISC, расширяющей её на архитектуры с несколькими вычислительными модулями. Так же, как в RISC, в инструкции явно указывается, что именно должен делать каждый модуль процессора. Из-за этого длина инструкции может достигать 128 или даже 256 бит.

## Модельный пример

Рассмотрим работу модельного VLIW-процессора с двумя арифметикологическими устройствами (АЛУ). Пусть нам надо сложить четыре числа, находящиеся в регистрах R1, R2, R3 и R4. Тогда псевдокод может выглядеть так:

$R5=R1+R2$ ,  $R6=R3+R4$ ; каждое АЛУ складывает свою пару чисел

$R0=R5+R6$ , NOP; первое АЛУ находит сумму, второе пристаивает  
Преимущества и недостатки

Подход VLIW сильно упрощает архитектуру процессора, перекладывая задачу распределения вычислительных устройств на компилятор. Поскольку отсутствуют большие и сложные узлы, сильно снижается энергопотребление.

В то же время, код для VLIW обладает невысокой плотностью. Из-за большого количества пустых инструкций для пристаивающих устройств программы для VLIW-процессоров могут быть гораздо длиннее, чем аналогичные программы для традиционных архитектур.

Архитектура VLIW выглядит довольно экзотической и непривычной для программиста. Из-за сложных внутренних зависимостей кода, программирование на уровне машинных кодов для VLIW-архитектур практически невозможно вручную. Приходится полагаться на оптимизацию компилятора, который сам может содержать ошибки.

### *Реализации*

Первые VLIW-процессоры появились в конце 1980-х и были разработаны компанией Cydrome.

В чистом виде архитектуру VLIW имеют процессоры TriMedia фирмы Philips и семейство DSP C6000 фирмы Texas Instruments.

Микропроцессор Transmeta Crusoe содержит слой двоичной совместимости с архитектурой x86, который компилирует инструкции во внутренний формат процессора (Code morphing). Ядро Crusoe является VLIW-процессором.

Микропроцессор Intel Itanium, кроме традиционной, содержит 64-битную систему команд «с явным параллелизмом» (англ. Explicitly Parallel Instruction Computing, EPIC), которая позволяет использовать VLIW-ядро непосредственно.

## Архитектура MISC

MISC (Minimal Instruction Set Computer) — процессор, работающий с минимальным набором длинных команд. Увеличение разрядности процессоров привело к идее совмещении нескольких команд в одно большое слово. Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных.

Процессоры, образующие «компьютеры с минимальным набором команд» MISC, как и процессоры RISC, характеризуются небольшим числом чаще всего встречающихся команд. Вместе с этим, принцип «очень длинных слов команд» VLIW обеспечивает выполнение группы команд за один цикл работы процессора. Порядок выполнения команд распределяется таким образом, чтобы в максимальной степени загрузить маршруты, по которым проходят потоки данных.

Таким образом, архитектура MISC объединила вместе суперскалярную и VLIW концепции. Компоненты процессора просты и работают с высокими скоростями.

## Архитектура EPIC

EPIC (Explicitly Parallel Instruction Computing) – микропроцессорная архитектура с явным параллелизмом команд. В отличие от RISC, архитектура EPIC, как следует уже из ее названия, поддерживает параллелизм на уровне команд. EPIC представляет собой пример научно-исследовательской разработки, описывающей собой скорее принцип обработки информации, нежели архитектуру конкретного процессора. В архитектуре сочетаются многие технологические решения, довольно разнотипные, которые дают значительное повышение скорости обработки и решение некоторых проблем трансляции программ (рис. 4.1).

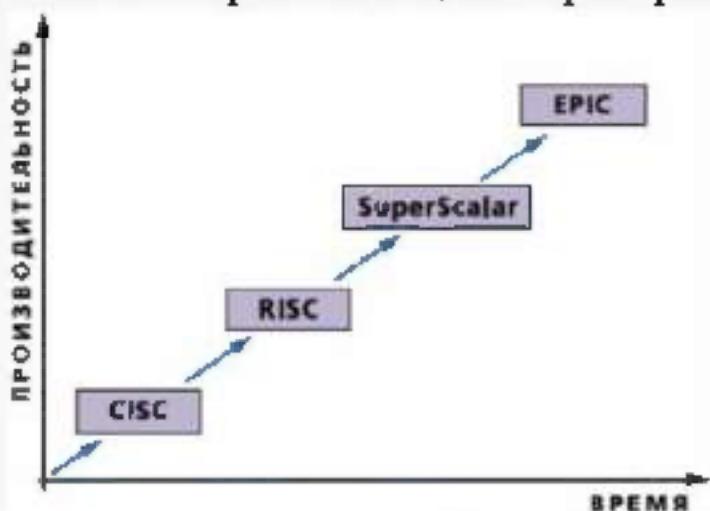


Рис. 4.1. Развитие технологий архитектуры команд

Обе архитектуры RISC и EPIC поддерживают выполнение нескольких команд в одном цикле за счет их параллельной обработки. При этом, в традиционных RISC-архитектурах задача распараллеливания вычислений решается процессором. При этом заметно усложняется структура чипа, и, как следствие, возникают препятствия для повышения скорости работы.

Основная идея, заложенная в архитектуру типа EPIC, состоит в явном распараллеливании кода еще до того, как он поступит на выполнение процессором. Если заложить параллелизм на этапе компиляции, можно избежать всех этих недостатков. При разработке программного обеспечения можно один раз потратить много времени и создать качественный машинный код. Кроме того, для возможной оптимизации компилятору доступна вся программа целиком, а не малый фрагмент в случае аппаратного распараллеливания.

Базовые принципы архитектуры EPIC были разработаны в университете Иллинойса, проект имел название Impact. В начале 1990-х годов были заложены теоретические основы самой архитектуры, затем начались работы в рамках создания инструментальных средств для процессора EPIC. Перечислим основные принципы новой архитектуры:

- Поддержка явно выделенного компилятором параллелизма. Формат команд имеет много общего с архитектурой с длинным командным словом — параллелизм также явно выделен. Однако, если длинное командное слово имеет обычно чётко заданную ширину, в процессоре EPIC существует некоторое количество образцов длинных команд, в которых с функциональными устройствами процессора явно сопоставлена операция. Кроме того, ряд образцов может сопоставлять инструкции из различных тиков выполнения, т. е. инструкция явно выполняется за 1-2 такта.

- Наличие большого регистрового файла. В архитектуре IPF (Itanium Processor Family) предусмотрено 128 регистров для целых чисел, 128 регистров для чисел с плавающей запятой и 64 регистра для предикатов.
- Наличие предикатных регистров, используемых для хранения результатов операций сравнения.

Предикатные регистры позволяют избавиться от известной проблемы с неразделяемым ресурсом регистра флагов большинства процессоров, поддерживающих скалярный параллелизм — программная конвейеризация циклов, содержащих условные переходы, практически невозможна.

Множественный предикатный флаговый регистр позволяет избавиться от паразитных связей по управлению из-за регистра флагов.

- Спекулятивная загрузка данных (позволяет вызывать операнды заранее, и даже если позднее окажется, что эти операнды не нужны), минимизирующая простой конвейера при загрузке данных из оперативной памяти.

- Поддержка предикатно выполняемых команд, которая позволяет:
  - 1) избежать излишних инструкций ветвления, если количество команд в ветвях условного оператора невелико;
  - 2) уменьшить нагрузку на устройство предсказания переходов.
- Аппаратная поддержка программной конвейеризации с помощью механизма переименования регистров (техника, используемая микропроцессорами, чтобы убрать зависимости между различными частями кода).
  - Стек регистров (и регистровые окна).
  - Поддержка компилятора для предсказания инструкций.
  - Специальная поддержка программной конвейеризации (метод составления расписания команд "по модулю") циклов, когда на каждой итерации имена регистров сдвигаются и каждая новая итерация оперирует уже с новым набором регистров.
  - Поддержка инструкций циклического выполнения команд без потерь времени на инструкции циклического выполнения.

Наиболее интересной в архитектуре EPIC является поддержка спекулятивного выполнения команд и загрузка данных. Большинство команд загрузки данных из памяти выполняется длительное время. Выполнение команды за 1-2 такта возможно только в том случае, если значение содержится в кэш-памяти первого уровня. Время несколько увеличивается, если значение находится в кэш-памяти второго уровня. В случае чтения же данных из микросхем динамической памяти даже при попадании на активную страницу происходит значительная задержка при загрузке, а при смене страницы динамической памяти задержка имеет астрономическую величину, причем конвейер быстро блокируется, так как команд, которые можно выполнить без нарушения зависимости по данным, обычно оказывается крайне мало.

При спекулятивном выполнении используется вынесение команд загрузки далеко вперед инструкций, использующих эти данные, в основном вверх, за инструкции условного перехода.

При достижении потоком команд места, где необходимо использовать загружаемые данные, вставляется инструкция, проверяющая, не произошло ли исключения в процессе загрузки (например, какая-то инструкция произвела запись по этому адресу), если исключение произошло, то вызывается специально написанный восстановительный код, который попросту перезагружает значение в регистр.

Все достоинства архитектуры EPIC воплощены в процессоре Intel® Itanium® 2.

## Конвейерный и суперскалярный подходы обработки данных

При обработке информации одним из главных критериев является скорость. Разработчики систем обработки информации стремятся к повышению производительности. Один из способов заставить процессоры работать быстрее — повышение их тактовой частоты, однако при этом существуют технологические ограничения (перегрев и ряд других причин принципиального характера). Поэтому большинство разработчиков для повышения производительности при данной тактовой частоте процессора используют параллелизм (выполнение двух или более операций одновременно).

Существует две основные формы параллелизма: параллелизм на уровне команд и параллелизм на уровне процессоров. В первом случае параллелизм реализуется за счет запуска большого количества команд каждую секунду. Во втором случае над одним заданием работают одновременно несколько процессоров.

## Конвейерный подход

Главным препятствием высокой скорости выполнения команд является необходимость их вызова из памяти. Для разрешения этой проблемы можно вызывать команды из памяти заранее и хранить в специальном наборе регистров. Набор регистров называется **буфером выборки с упреждением**. Таким образом, когда требовалась определенная команда, она вызывалась прямо из буфера, а обращения к памяти не происходило.

В действительности при выборке с упреждением команда обрабатывается за два шага: сначала происходит вызов команды, а затем — ее выполнение. Еще больше продвинула эту стратегию идея **конвейера**. При использовании конвейера команда обрабатывается уже не за два, а за большее количество шагов, каждый из которых реализуется определенным аппаратным компонентом, причем все эти компоненты могут работать параллельно.

На рис. 4.1, а изображен конвейер из пяти блоков, которые называются ступенями. Первая ступень (блок С1) вызывает команду из памяти и помещает ее в буфер, где она хранится до тех пор, пока не потребуется. Вторая ступень (блок С2) декодирует эту команду, определяя ее тип и тип ее операндов. Третья ступень (блок С3) определяет местонахождение операндов и вызывает их из регистров или из памяти.

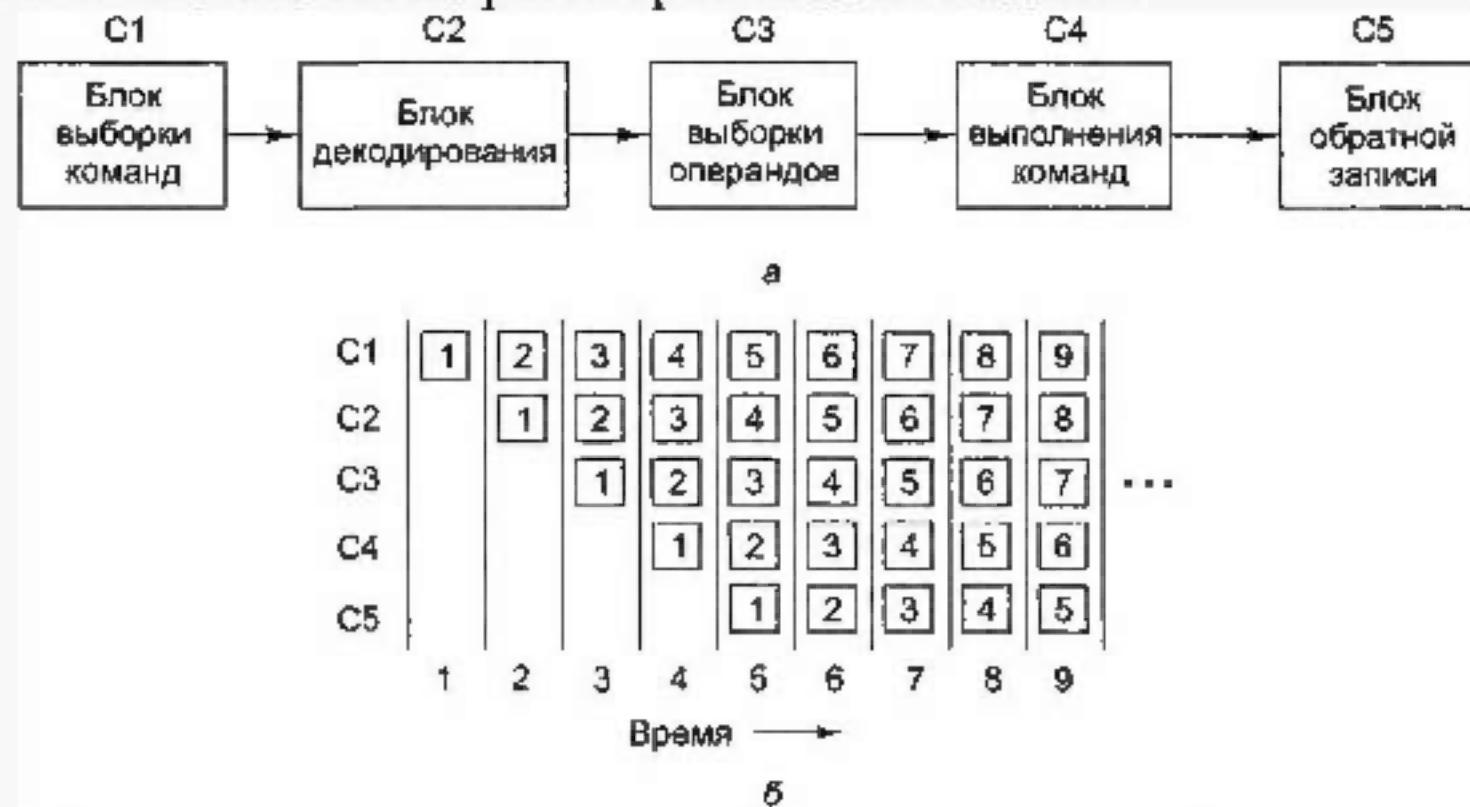


Рис. 4.2. Пятиступенчатый конвейер (а); состояние каждой ступени в зависимости от количества пройденных циклов (б). Показано 9 циклов

Четвертая ступень (блок С4) выполняет команду, обычно проводя операнды через шину данных. И наконец, блок С5 записывает результат обратно в нужный регистр.

На рис. 4.2, б мы видим, как действует конвейер во времени. Во время цикла 1 блок С1 обрабатывает команду 1, вызывая ее из памяти. Во время цикла 2 блок С2 декодирует команду 1, в то время как блок С1 вызывает из памяти команду 2. Во время цикла 3 блок С3 вызывает операнды для команды 1, блок С2 декодирует команду 2, а блок С1 вызывает команду 3. Во время цикла 4 блок С4 выполняет команду 1, С3 вызывает операнды для команды 2, С2 декодирует команду 3, а С1 вызывает команду 4. Наконец, во время цикла 5 блок С5 записывает результат выполнения команды 1 обратно в регистр, тогда как другие ступени конвейера обрабатывают следующие команды.

Предположим, что время цикла у этой машины — 2 нс. Тогда для того, чтобы одна команда прошла через весь конвейер, требуется 10 нс. На первый взгляд может показаться, что такой компьютер будет выполнять 100 млн команд в секунду, в действительности же скорость его работы гораздо выше. В течение каждого цикла (2 нс) завершается выполнение одной новой команды, поэтому машина выполняет не 100, а 500 млн команд в секунду!

Конвейеры позволяют добиться компромисса между **временем запаздывания** (время выполнения одной команды) и **пропускной способностью процессора** (количество команд, выполняемых процессором в секунду). Если время обращения составляет  $T$  нс, а конвейер имеет  $n$  ступеней, время запаздывания составит  $nT$  нс.

Поскольку одна команда выполняется за одно обращение, а за одну секунду таких обращений набирается  $10^9/T$ , количество команд в секунду также составляет  $10^9/T$ . Скажем, если  $T = 2$  нс, то каждую секунду выполняется 500 млн команд. Для того чтобы получить значение MIPS, нужно разделить скорость выполнения команд на 1 миллион; таким образом,  $(10^9/T)/10^6 = 1000/T$  MIPS (миллион операций в секунду).

## Суперскалярные архитектуры

Существуют процессоры не с одним, а с двумя конвейерами. Одна из возможных схем процессора с двумя конвейерами показана на рис. 4.5. В ее основе лежит конвейер, изображенный на рис. 4.2. Здесь общий блок выборки команд вызывает из памяти сразу по две команды и помещает каждую из них в один из конвейеров. Каждый конвейер содержит АЛУ для параллельных операций. Чтобы выполняться параллельно, две команды не должны конфликтовать из-за ресурсов (например, регистров), и ни одна из них не должна зависеть от результата выполнения другой. Как и в случае с одним конвейером, либо компилятор должен гарантировать отсутствие нештатных ситуаций (когда, например, аппаратура не обеспечивает проверку команд на несовместимость и при обработке таких команд выдает некорректный результат), либо за счет дополнительной аппаратуры конфликты должны выявляться и устраняться непосредственно в ходе выполнения команд.

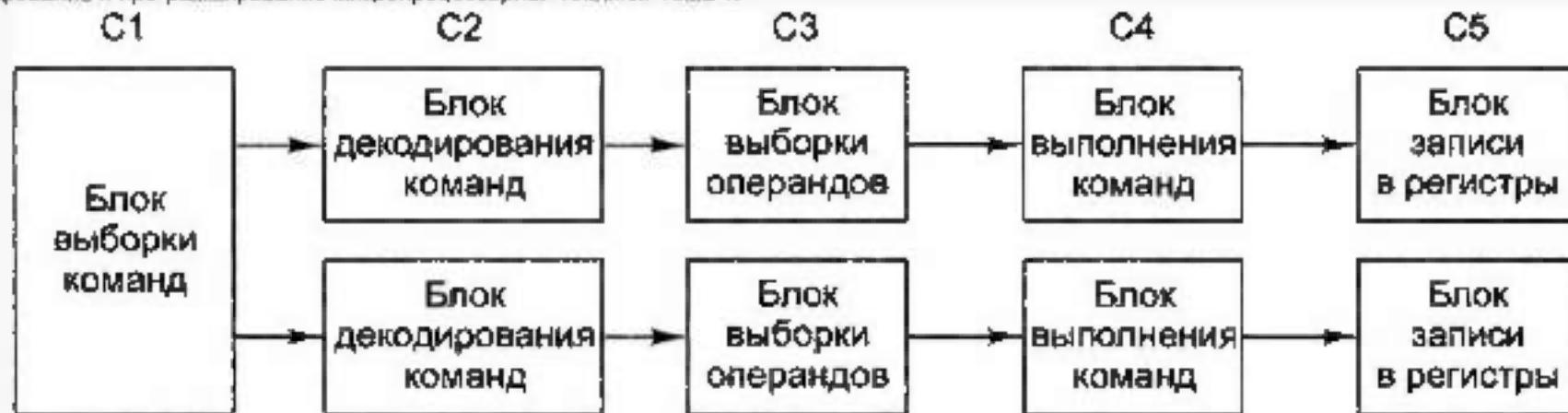


Рис. 4.3. Сдвоенный пятиступенчатый конвейер с общим блоком выборки команд

Конвейеры в процессорах появились в конце 80-х годов 20 века (только начиная с модели 486 компании Intel). Процессор 486 компании Intel имел один пятиступенчатый конвейер, а Pentium — два таких конвейера. Похожая схема изображена на рис. 1.6, но разделение функций между второй и третьей ступенями (они назывались декодер 1 и декодер 2) было немного другим. Главный конвейер (**u-конвейер**) мог выполнять произвольные команды. Второй конвейер (**v-конвейер**) мог выполнять только простые команды с целыми числами, а также одну простую команду с плавающей точкой (FXCH).

Переход к четырем конвейерам возможен, но требует громоздкого аппаратного обеспечения. Основная идея — один конвейер с большим количеством функциональных блоков, как показано на рис. 4.4. (Pentium II, к примеру, имеет сходную структуру). В 1987 году для обозначения этого подхода был введен термин **суперскалярная архитектура**.

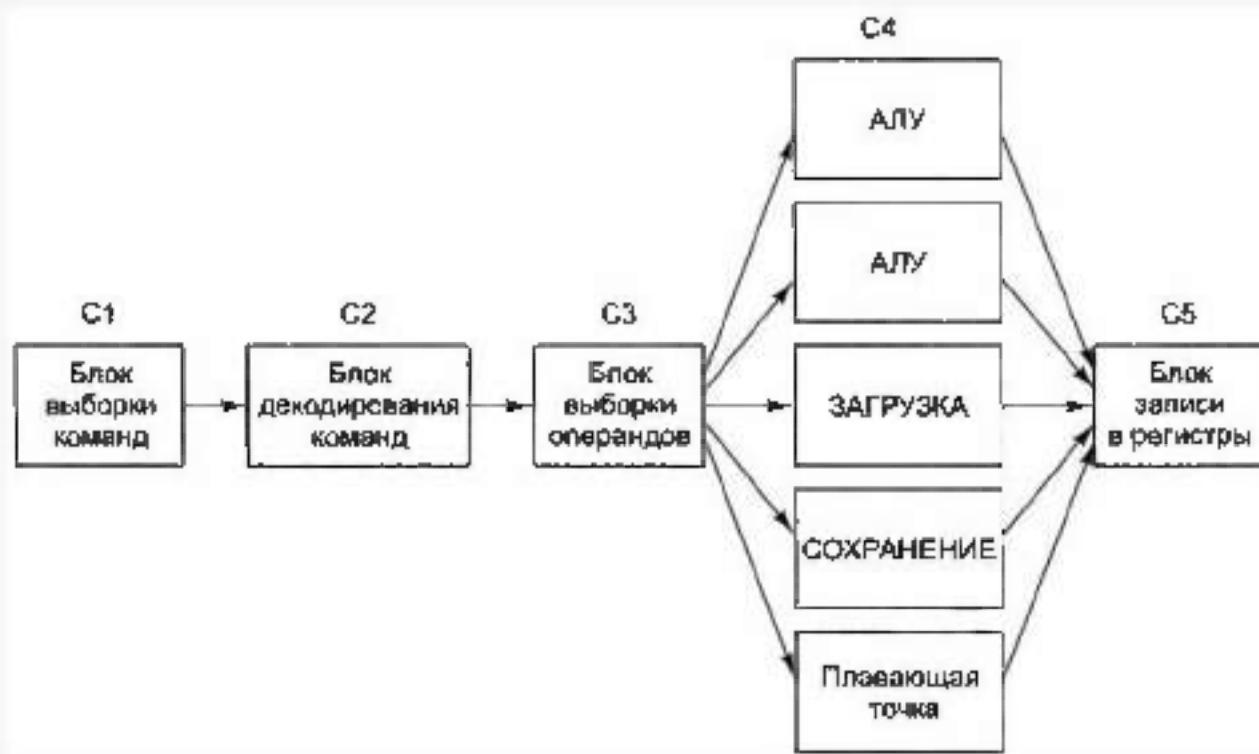


Рис. 4.4. Суперскалярный процессор с пятью функциональными блоками

\*\* Со временем значение понятия «суперскалярный» несколько изменилось. Теперь суперскалярными называют процессоры, способные запускать несколько команд (зачастую от четырех до шести) за один тактовый цикл. Естественно, чтобы передавать все эти команды, в суперскалярном процессоре должно быть несколько функциональных блоков. Поскольку в процессорах этого типа, как правило, предусматривается один конвейер, его устройство обычно соответствует рис. 4.4.

### 3 ПРОЦЕССОРЫ ARM

Обычно, в настольных компьютерах и ноутбуках применяются соответствующие мощные процессоры (чаще всего Intel или AMD). Такие ЦПУ спроектированы для обеспечения оптимальной производительности, при достаточном энергопотреблении и активном охлаждении.

Мобильные устройства требуют иного подхода. Чтобы оставаться портативными, их батареи должны иметь меньший размер, охлаждение зачастую может быть только пассивным, а энергопотребление не должно превышать довольно скромных значений.

Сложная архитектура настольных ЦПУ плохо подходит для мобильных устройств, из-за существенных отличий в аппаратных требованиях (нагрев и энергопотребление). В итоге мобильные устройства (в том числе смартфоны, планшеты, устройства IoT) были нежизнеспособны при использовании традиционной архитектуры.

Чтобы преодолеть эти трудности, производители решили заменить настольную ЦПУ архитектуру чем-то более подходящим для мобильного рынка. ARM процессоры стали идеальным выбором, потому что они используют упрощенные, менее энергозатратные методы обработки. Это также отражено и в имени ARM, которое расшифровывается как Advanced RISC Machine (усовершенствованная RISC-машина).

Первоначально, RISC представляла собой 32-битную архитектуру, но в 2011 году, ARM обеспечила поддержку 64-битных вычислений, в своих продуктах. Меньшая сложность RISC блоков также означает, что они состоят из меньшего числа транзисторов. Как правило, чем больше транзисторов - тем больше энергопотребление, выше сложность и стоимость производства. Поэтому процессоры ARM обычно стоят меньше, чем традиционные настольные ЦПУ.

## ARM7

Фирмой разработан целый ряд 32-разрядных RISC процессоров с различными возможностями и различной производительности а ее процессор ARM7, разработанный еще в 1994 году, используется до настоящего времени.

Сама фирма ARM определяет процессор ARM (в частности ARM7) как универсальное, с малым потреблением, ядро 32-разрядного RISC микропроцессора, предназначенное для использования в различных заказных и специальных ИС. Малые размеры RISC ядра позволяют успешно интегрировать его в большие заказные схемы, которые могут содержать RAM, ROM, DSP, дополнительную логику и другие элементы.

## Основные характеристики ядра ARM7

- 32-разрядный RISC процессор (32-разрядные шины данных и адреса) с производительностью 17 MIPS при тактовой частоте 25 МГц (пиковая производительность 25 MIPS)
- 32-разрядная адресация - линейное адресное пространство в 4 Гбайта - исключает потребность в сегментированной, разделенной на банки или оверлейной памяти
- Тридцать один 32-разрядный регистр общего назначения и шесть регистров состояния
- Регистры адресов, записи и конвейера
- Циклическое сдвиговое устройство и перемножитель
- Трехуровневый конвейер (выборка команды, ее декодирование и выполнение)
- Напряжение питания 3,3 и 5 В
- Малое потребление 0,6 мА/МГц, при изготовлении по CMOS технологии с топологическими нормами 0,8 мкм.
- Быстрый отклик на прерывания применений реального масштаба времени
- Поддержка систем виртуальной памяти
- Простая, но мощная система команд

Перевод ядра на технологию с уменьшенными топологическими нормами позволяет как повысить его производительность, так и еще больше снизить потребление.

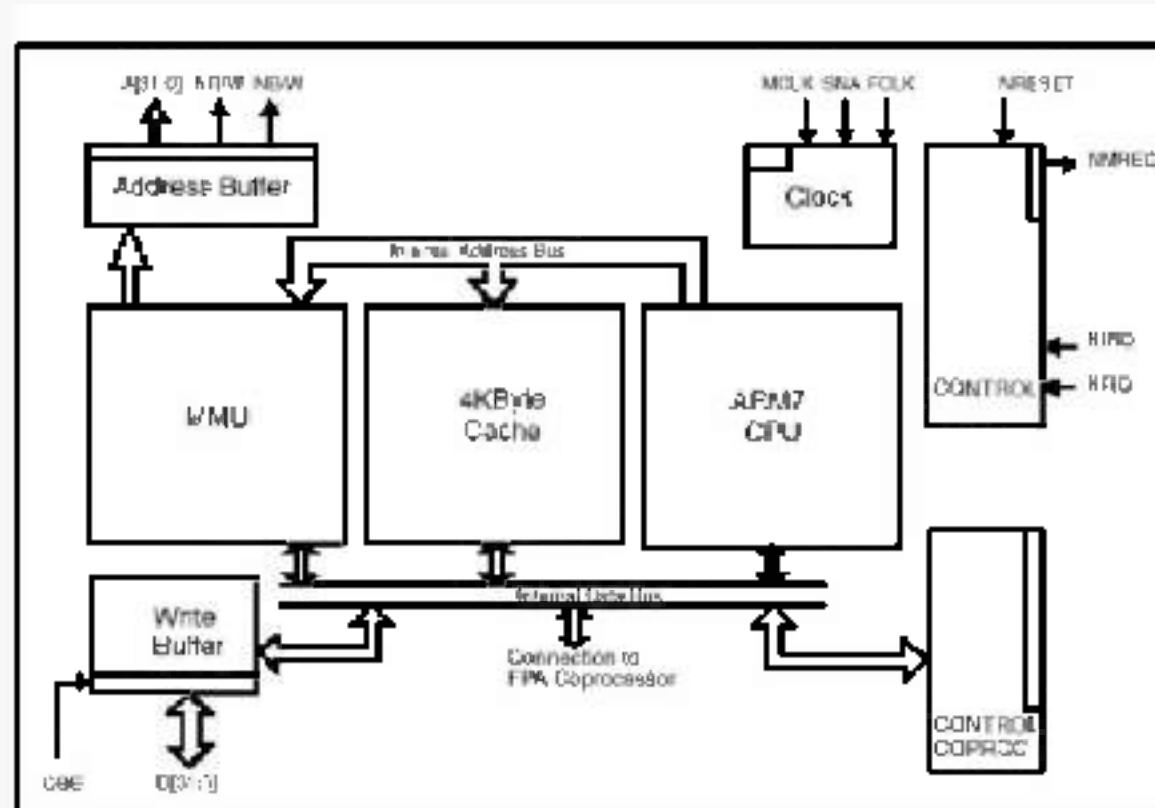


Рис. 1. Блок-схема ядра ARM7

Предоставляя, на лицензионной основе, ядро ARM7 фирма ARM на его основе разработала микроконтроллеры ARM7100, ARM7500 и ARM7500FE.

Микроконтроллер ARM7100 можно назвать микроконтроллером широкого применения, поскольку он ориентирован на: персональные информационные устройства (PDA) и организеры, смартфоны, карманные измерительные устройства и системы сбора.

Два других микроконтроллера ARM7500 и ARM7500FE являются однокристальными микрокомпьютерами, ориентированными на реализацию мультимедиа устройств, портативных и настольных компьютеров. Эти два микроконтроллера отличаются друг от друга наличием в приборе ARM7500FE ускорителя операций с плавающей точкой (FPA) и, соответственно, его более высокой производительностью. Они также реализованы по модульному принципу и объединяют ядро ARM7 с самодостаточными макроячейками видео, звука, FPA (ARM7500FE) и стандартных библиотечных ячеек периферии.

## **Подходы для решения проблемы большого размера кода**

### **Архитектура с расширенными возможностями**

Архитектура RISC фирмы ARM обеспечивает как малое потребление мощности и малый размер кристалла так и высокую производительность, необходимые во встраиваемых применениях. Фирма ARM расширила возможности этой архитектуры, с точки зрения решения проблемы размера кодов, разработав новую технологию - новую систему команд Thumb.

Существует несколько подходов, решающих проблему размера кода:

**Написание кода вручную на ассемблере.** Для получения минимального размера кода программист может писать коды вручную - на ассемблере.

**Использование улучшенного компилятора.** Технология компилирования может улучшить код, но опять таки меньшим размером кода будет при ручном кодировании на ассемблере.

**Использование сжатого кода.** Одним из вариантов может быть использование некоторой формы сжатого кода, который разворачивается во время выполнения. Однако, быстрая декомпрессия, которая не будет снижать производительность процессора при выполнении этого кода, достаточно сложна и требует использования дополнительных ресурсов системы.

## Концепция Thumb

Технология Thumb - дополнительное расширение к архитектуре ARM. Система команд Thumb содержит 36 команд, производных от стандартной 32-разрядной системы команд ARM, перекодированных в 16-разрядные коды.

Такой подход обеспечил очень высокую плотность кода, поскольку команды Thumb составляют половину ширины формата команд ARM.

В процессе выполнения эти новые 16-разрядные Thumb коды декомпрессируются процессором в соответствующие эквивалентные команды ARM, которые затем и выполняются ядром ARM обычным способом.

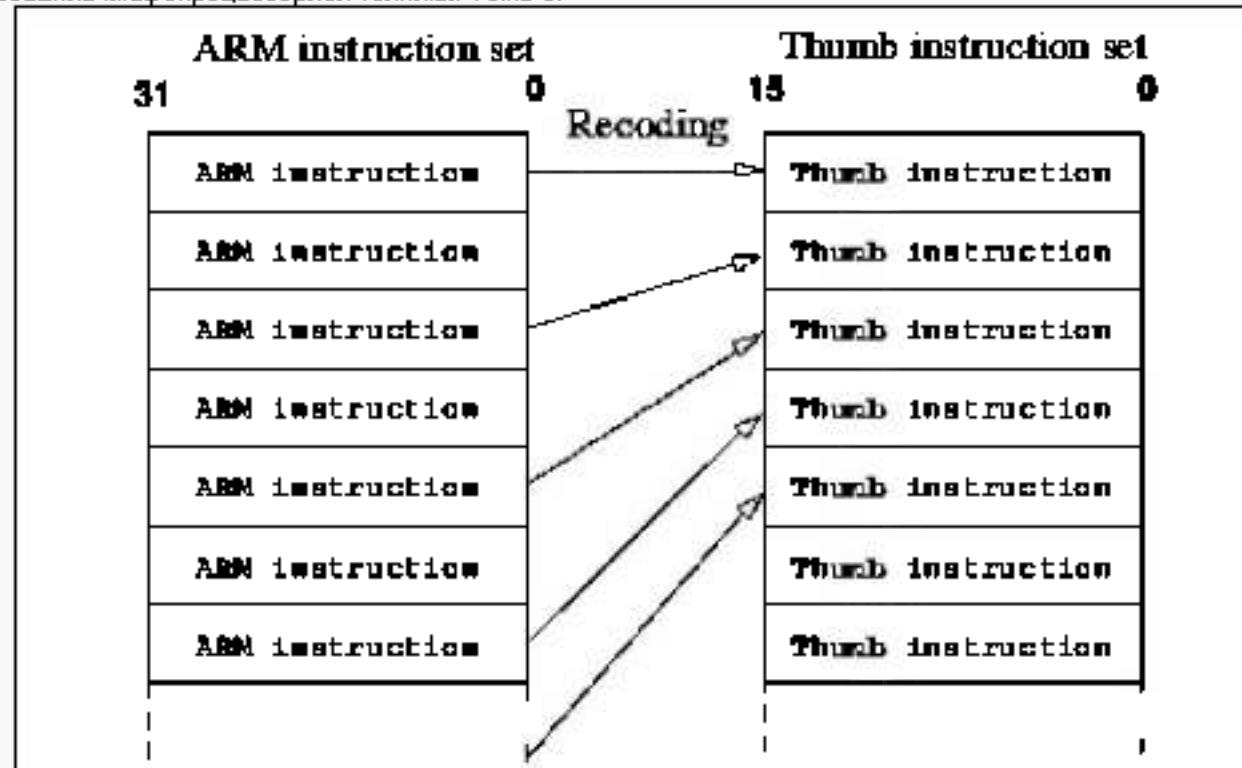


Рис. 2. Команды Thumb как кодированное подмножество системы команд ARM

Так как Thumb -ориентированные ядра способны выполнять и стандартную ARM систему команд и новые команды Thumb, разработчик, при переходе от подпрограммы к подпрограмме, может находить компромисс между размером кода и производительностью, подготавливая критичные к размеру подпрограммы в коде Thumb и критичные к производительности подпрограммы в кодах ARM.

Полученные к настоящему времени результаты показали улучшение плотности кода на 30%, по сравнению с кодом ARM, что позволяет считать Thumb -ориентированные процессоры лучшими по плотности кода в сравнении и с традиционными CISC процессорами.

**Поддержка полуслов.** Кроме введения новых Thumb команд, фирма ARM добавила к системам команд и ARM и Thumb поддержку формата полуслов (16-разрядных данных). В итоге можно отметить, что архитектура ARM полностью поддерживает 8, 16 и 32-разрядные данные.

## Примеры конфигурации системы

Данные конфигурации показывают  
thumb-ориентированное ядро в системе

### Пример 1

Достоинство этой системы в использовании внешней шины и памяти узкого формата, что соответствует требованиям недорогих встраиваемых применений.

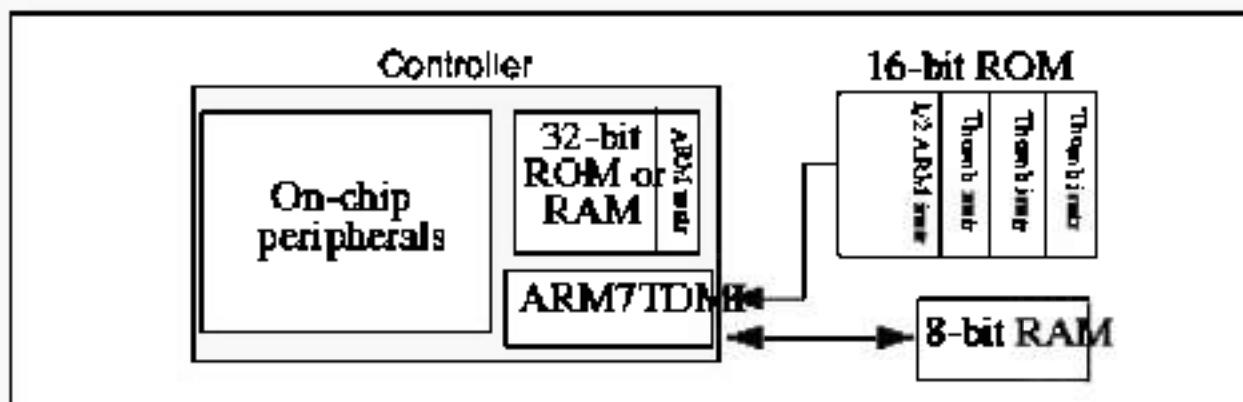


Рис. 3. Контроллер с 16-разрядной системой памяти

В контроллере интегрированы заданные заказчиком встроенные периферийные устройства и небольшие по объему быстрые 32-разрядные ROM или RAM, используемые для хранения критичного к быстродействию кода. Когда Thumb-ориентированное ядро переключается в состояние ARM для получения максимальной производительности, например, обработки прерывания, коды ARM выполняются из этой области быстродействующей памяти. Внешняя 16-разрядная ROM используется для хранения кодов и констант, а 8-разрядная RAM содержит сверхоперативные данные.

## Пример 2

Эта конфигурация показывает, как Thumb-ориентированное ядро может использоваться с медленной, недорогой 32-разрядной ROM.

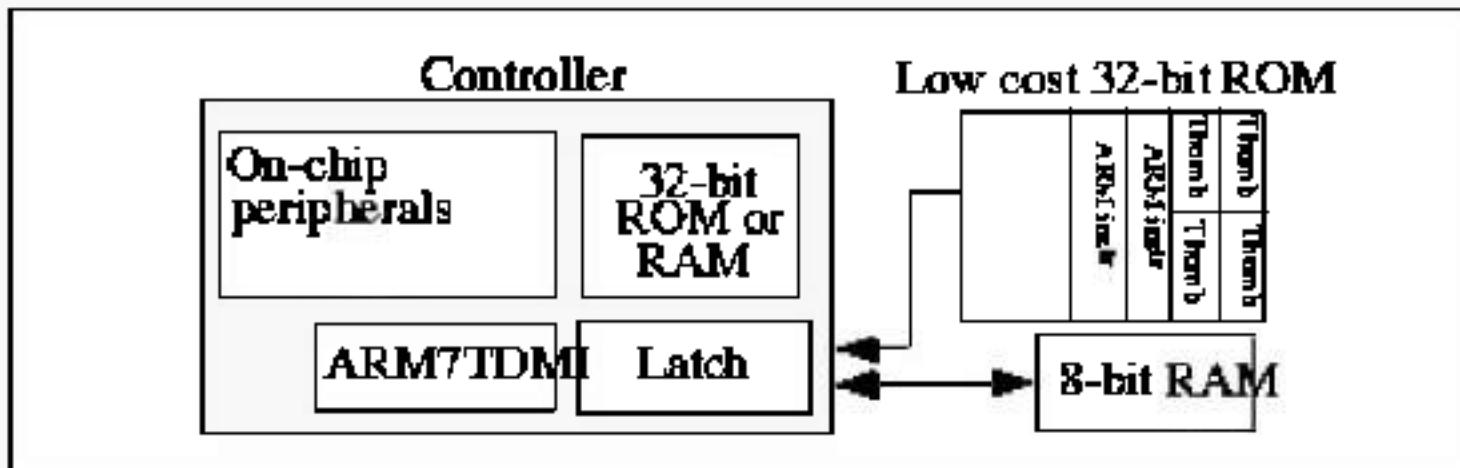


Рис. 4. Система с 32-разрядной ROM

В данной конфигурации в ROM сохраняется смесь подпрограмм 32-разрядного кода ARM, с одной командой на 32-разрядное слово, и подпрограмм кода Thumb с двумя командами на каждое слово. Каждая внешняя выборка выбирает или 32-разрядную ARM команду или две 16-разрядные команды Thumb.

Команды ARM поступают в основной конвейер обычным способом. Однако, в состоянии Thumb, в то время когда одна команда Thumb поступает в конвейер, другая сохраняется в 16-разрядной защелке (Latch), которая является буфером выборок команд с упреждением. При следующей выборке, эта сохраненная команда немедленно становится доступной ядру.

Моделирование показало, что в этой конфигурации с 200 нс ROM, технология Thumb превосходит по быстродействию стандартное ядро ARM на 10 - 20%, в зависимости от тактовой частоты процессора и кода.

Это связано с тем, что ядру ARM для выбора каждой команды из ROM необходимы состояния ожидания, в то время как Thumb-ориентированное ядро находится в состоянии ожидания только один раз на каждые две команды.

С быстродействующими ROM, синхронизированными с частотой процессора, состояния ожидания исключаются и, следовательно, ARM решение будет всегда превосходить по быстродействию Thumb-ориентированное ядро.

### • Пример 3

Данная конфигурация представляет последнее Thumb - ориентированное решение перед переходом к стандартному ARM ядру, обеспечивающему наивысшую производительность 32-разрядной системы.

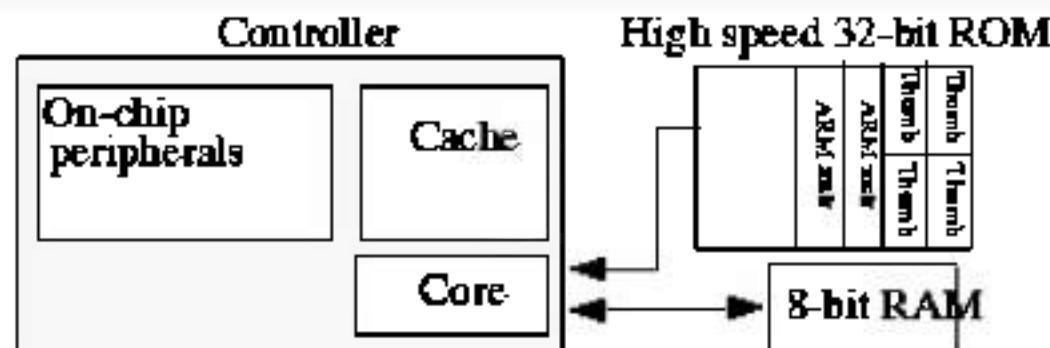


Рис. 5. Высокопроизводительная 32-разрядная система

При использовании быстродействующей ROM и встроенном кэш, эта система обеспечивает самую высокую производительность для Thumb-ориентированного ядра, поскольку 32-разрядные ARM команды могут выполняться непосредственно из быстродействующей памяти. Размер кода и стоимость системы, естественно, больше чем у недорогих 16-разрядных шины и системы памяти.

## Thumb-ориентированное ядро ARM7TDMI и его развитие

Впервые технология Thumb была встроена в ядро ARM7 еще в 1995 году. Адаптированное под технологию Thumb (Thumb-ориентированное) ядро получило типовое обозначение ARM7TDMI.

Конвейерная обработка реализована таким образом, что все компоненты систем памяти и обработки работают непрерывно. Обычно, в то время как одна команда выполняется, следующая команда декодируется и третья команда выбирается из памяти.

Процессор ARM7TDMI - 32-разрядный RISC процессор с 3-уровневым конвейером, сформированный вокруг банка из 37 32-разрядных регистров, в который входят 6 регистров состояния. Процессор оснащен встроенным умножителем 32x8 и 32-разрядным многорегистровым циклическим устройством сдвига. Пять независимых встроенных шин (PC шина, шина инкремента, шина ALU и A- и B-шины) обеспечивают, при выполнении команд высокую степень параллелизма.

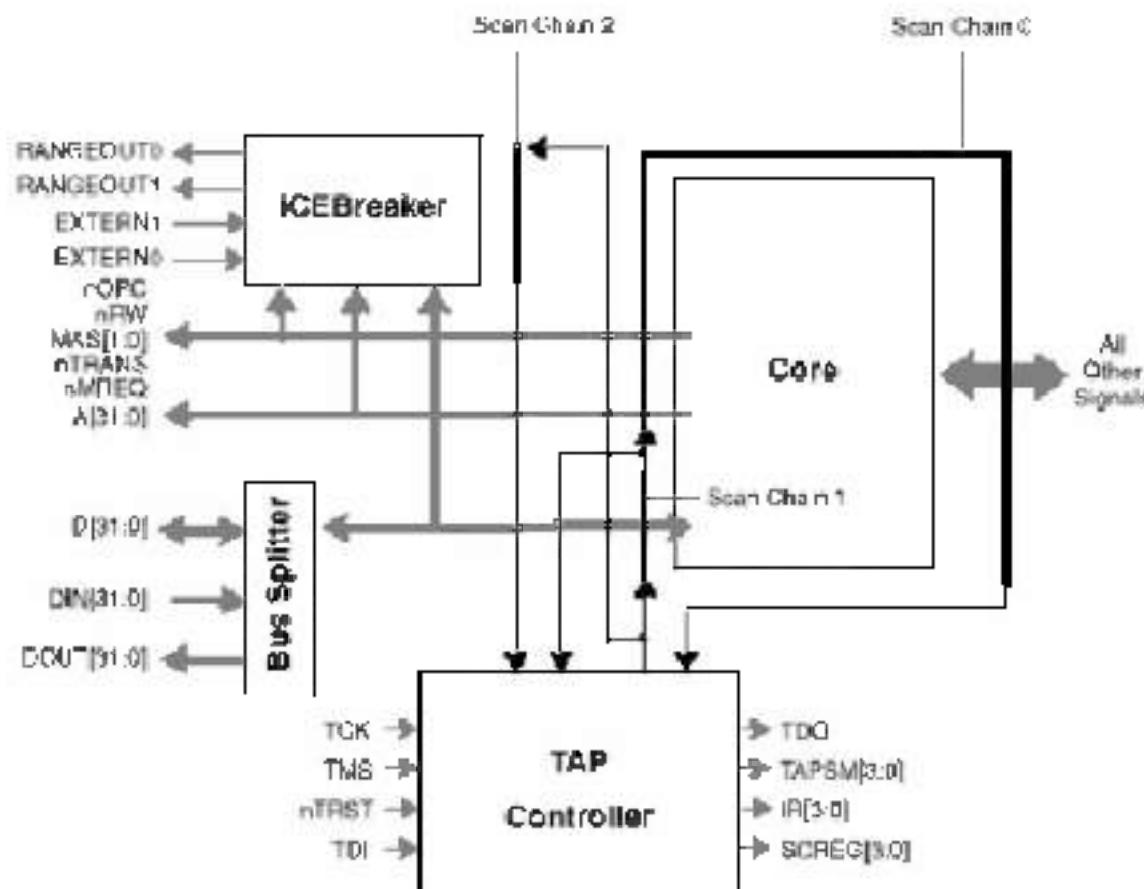


Рис. 6. Блок-схема ядра ARM7TDMI

Как видно из сравнения блок-схем процессоров ARM7 и ARM7TDMI в блоке конвейера процессора добавился декомпрессор команд Thumb.

Однако разработчики фирмы ARM этим не ограничились и, имея в виду встраивание ядра в приборы с большим уровнем интеграции, расширили ядро ARM7TDMI (см. Рис. 6) дополнительными аппаратными блоками, обеспечивающими возможность отладки глубоко встроенного ядра.

Ядро ARM7TDMI состоит из ядра собственно процессора и расширений отладки: контроллера сканирования ТАР (boundary scan) и внутрисхемного эмулятора (ICEBreaker).

Аппаратные расширения отладки ARM7TDMI, обеспечивают развернутые возможности отладки, облегчающие разработку пользовательского прикладного программного обеспечения, операционных систем, и самих аппаратных средств. Аппаратные расширения отладки позволяют останавливать ядро или при выборке заданной команды (в контрольной точке) или при обращении к данным (в точке просмотра), или асинхронно - по запросу отладки.

ARM активно использует технологию макроядер. При разработке этих макроядер фирма ARM ориентировалась на конкретные области применения, где особенности каждого макроядра позволяют реализовать дополнительные возможности без существенного прироста стоимости.

Добавление к макроядрам встроенного кэш позволяет минимизировать время доступа к внешней памяти и, сохранив максимальную производительность, позволяет использовать недорогие RAM.

Становится возможным использование системной шины и внешней памяти с быстродействием более низким, чем быстродействие процессора и, следовательно, уменьшить потребление.

Широкая полоса частот системной шины может быть также использована и для увеличения полной производительности системы - высвобожденную полосу частот могут использовать другие периферийные устройства, обеспечивая высокую пропускную способность данных в устройствах типа MPEG декодеров цифровых TV приставок.

**Макроядро ARM710T, ориентированное на персональные информационные устройства (PDA) и Internet применения, имеет возможность использования виртуальной памяти, обеспеченная MMU, позволяет безопасно использовать коды выгруженные из сети типа Internet или от независимого разработчика.** Такая возможность позволяет считать ядро процессора ARM710T идеальным для применения в PDA, интеллектуальных телефонах или Internet телевидении.

**Макроядро ARM720T, ориентированное на операционную систему WindowsCE, располагает всеми функциональными возможностями ядра ARM710T плюс специальная поддержка операционной системы WindowsCE.** Невысокая цена, высокая производительность и малое потребление процессора ARM720T делают его идеальным решением для перспективных приложений, использующих WindowsCE в PDA, карманных PC, TV и Internet приставках, интеллектуальных телефонах и автомобильных PC.

**Макроядро ARM740T, ориентированное на высокопроизводительные встраиваемые применения**, в отличие от других макроядер, оснащено кэш который может быть емкостью или 4 или 8 Кбайт и, кроме того, модулем защиты буфера записи и памяти (не полнофункциональным MMU). Макроядро ARM740T ориентировано на использование в мультимедиа и встраиваемых применениях типа цифровых TV приставок, Internet аппаратуры и сетевых устройств, в модемах и системах, для которых разрабатывается специальное ПО, не требующее управления виртуальной памятью, обеспечиваемой MMU.

Макроядра оснащены портами для присоединения встраиваемых сопроцессоров, что обеспечивает расширение функциональных возможностей макроядер ARM7XXT архитектурно непротиворечивым способом.

**Ядро ARM7TDMI-S<sup>TM</sup>**, являющееся последним дополнением семейства ARM7T, это Thumb-ориентированное синтезируемое 32-разрядное RISC ядро с изменяемой конфигурацией, высокой производительностью и малым потреблением.

Ядро ARM7TDMI-S программно совместимо с популярным ARM7TDMI ядром, так что программы для каждого ядра могут разрабатываться на одних и тех же средствах разработки программного обеспечения и основное отличие ядра ARM7TDMI-S от ARM7TDMI ядра заключается в том, что ядро ARM7TDMI-S является полностью синтезируемым - обеспечивающим простую интеграцию в современные технологии изготовления интегральных схем, специализированных для решения конкретной задачи (ASIC).

## **Встроенная системная шина AMBA**

При рассмотрении макроядер ARM710T, ARM720T и ARM740T была упомянута шина AMBA (Advanced Microcontroller Bus Architecture) - шина разработанная фирмой ARM для организации эффективного взаимодействия компонентов приборов, построенных на базе ядер фирмы.

Шина AMBA - стандартная встроенная шина, обеспечивающая быстрое модульное проектирование систем при упрощении многократного использования.

Блок-схема шины AMBA в прибора типа персонального информационного устройства (PDA), реализованного на основе ядра ARM, макроячеек библиотеки PrimeCell и шины AMBA, представлен на Рис. 7.

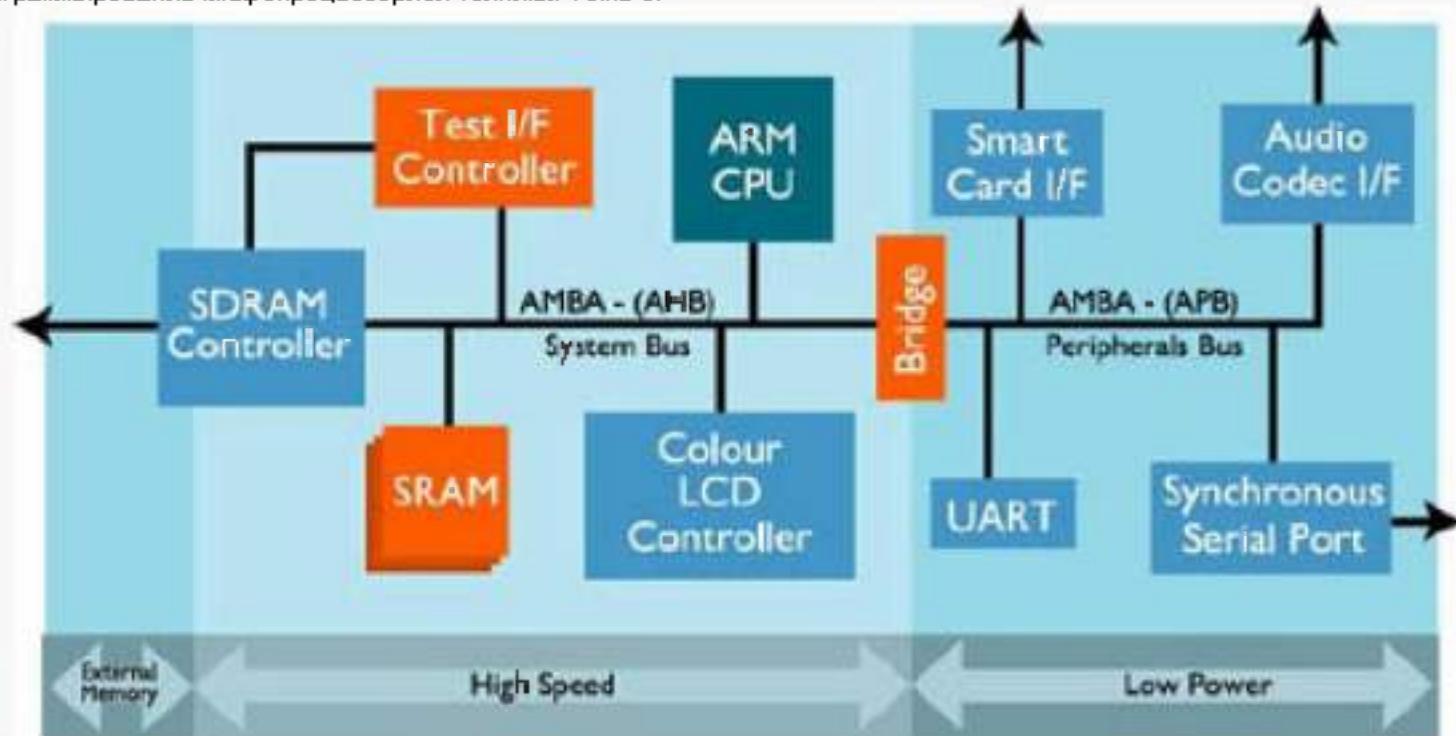


Рис. 7. Пример прибора класса "система-на-кристалле", использующего шину AMBA

Типовая шина AMBA содержит системную шину (в данном случае AHB - развитая высокопроизводительная шина) и шину периферии (APB).

*Системная шина* соединяет встраиваемые процессоры, такие как ARM ядра, с высокопроизводительной периферией, контроллерами DMA, встроенными памятью и интерфейсами. Это высокоскоростная, с широкой полосой пропускания шина, поддерживающая, для обеспечения максимальной производительности, управление с большим количеством ведущих устройств (Multi-master bus management).

*Шина периферии* - работает с упрощенным протоколом и разработана для организации интерфейса с периферийными устройствами общего назначения или дополнительными периферийными устройствами. С системной шиной она соединяется через мост (bridge), способствующий снижению потребления системы.

В спецификации шины AMBA определена методология тестирования, обеспечивающая быстрое тестирование модулей и кэш.

## Системная шина AMBA

Новая спецификация AMBA Rev 2.0 Specification определяет два типа системной шины:

- AHB - развитая высокопроизводительная шина
- ASB - развитая системная шина

### Развитая высокопроизводительная шина (Advanced High-performance Bus - AHB)

Шина AHB используется в высокопроизводительных системах класса "система-на-кристалле" и соответствует современным требованиям предъявляемым процессом синтеза приборов с уровнем интеграции система-на-кристалле.

Использует конвейерные и пакетные пересылки

- Конвейерная работа обеспечивает обращение к быстродействующей памяти или периферии без использования режимов ожидания, без холостых циклов шины.

- Пакетная работа позволяет оптимально использовать интерфейс с памятью за счет предоставления дополнительной информации о характере пересылаемых данных.

Использует поддержка разделения транзакций

- Обеспечивает максимальное использование полосы пропускания системной шины за счет высвобождения ее от медленных ведомых устройств на время завершения их внутренних операций.

Поддерживает возможность конфигурирования в широком формате (форматы от 32/64/128 до 1024 бит)

- Поддерживает применения с широкоформатной встроенной памятью с интенсивным обменом данными и широкой полосой частот.

## Развитая системная шина (Advanced System Bus - ASB)

Это оригинальная системная шина AMBA, разработанная на основе интерфейса ARM:

- Работает в режиме с множеством ведущих
- Обеспечивает конвейерные и пакетные пересылки

Примером использования шины ASB может служить блок-схема, представленная на Рис. 8.

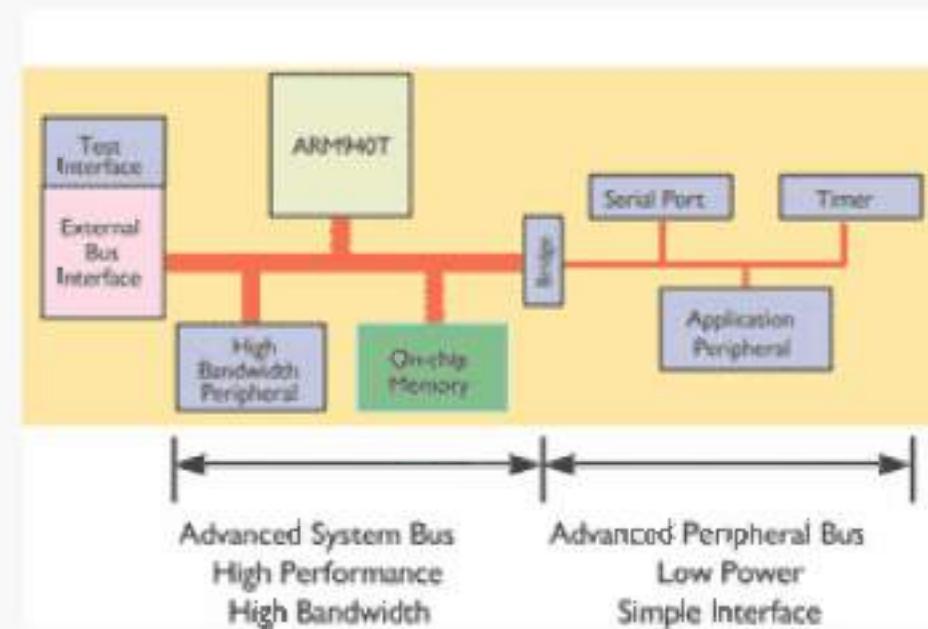


Рис. 8. Блок-схема прибора, реализованного на основе ядра ARM940T

## AMBA шина периферии

### Развитая шина периферии (Advanced Peripheral Bus - APB)

AMBA шина периферии предназначена для организации интерфейса с встроенными периферийными устройствами общего назначения, такими как таймеры, контроллеры прерываний, UART, порты I/O и т.п., и дополнительными периферийными устройствами. С основной системной шиной периферии соединяется мостом, обеспечивающим разгрузку системной шины и снижающим общее потребление системы.

В соответствии с новой спецификацией AMBA Rev 2.0 Specification шина отвечает современным требованиям последовательности синтеза приборов класса "система-на-кристалле".

#### 1. Простая шина

Бесконвейерная архитектура

Простое использование - все периферийные устройства обслуживаются как ведомые

Малое количество используемых вентилей

## 2. Малое потребление

Снижение загрузки основной системной шины за счет ее изоляции от периферии мостом

Сигналы на шине периферии активны только во время медленных пересылок периферии

### Семейство ARM9 Thumb

ASIC (*Applications Specific Integrated Circuit*) стали первыми приборами, которые в определенной степени учитывали особые требования, предъявляемые заказчиками, и которые не могли быть достаточно просто и экономично реализованы на стандартных приборах (универсальных - общего назначения).

Наиболее активно ASIC стали развиваться с введением технологии программных ядер и программных библиотек периферийных компонентов, которые, по мере их развития, позволили создавать на одном кристалле достаточное большое количество функций, чтобы можно было называть такие кристаллы "системами-на-кристалле". И если до появления ASIC функции, реализуемые электронной

аппаратурой, определялись, в основном, предлагаемыми полупроводниковыми отраслями стандартными приборами, то потом набор функций, возможности приборов, изготавливаемых в полупроводниковых отраслях определяли в первую очередь потребности пользователей т.е. рыночный спрос.

Семейство 32-разрядных ARM RISC процессоров ARM9TDMI, сохранившее основные преимущества ARM7TDMI, ставших промышленным стандартом, обеспечивает двукратное увеличение производительности, при изготовлении по эквивалентной технологии.

Производительность ARM9TDMI составляет 133 MIPS при 120 МГц и технологии CMOS с топологическими нормами 0,35 мкм. При топологии 0,25 мкм и 0,18 мкм рабочая частота составит выше 200 МГц и производительность выше 220 MIPS.

В семейство ARM9TDMI, в настоящее время, входит процессорное ядро ARM9TDMI, и оснащенные кэш процессорные макроядра ARM940T и ARM920T.

**Макроядро ARM940T** - законченная высокопроизводительная подсистема CPU, содержащая ядро ARM9TDMI (целочисленное RISC CPU), кэши команд и данных, емкостью по 4 Кбайт, буфер записи, блок защиты и интерфейс AMBA ASB шины.

Макроядро ARM940T предназначено для использования во встраиваемых управляющих применениях, не требующих поддержки виртуальной памяти.

Модуль защиты, обеспечивающий возможности конфигурирования и защиты памяти, необходимые для типичной RTOS, позволяет определять 8 областей памяти, каждая из которых располагает независимыми полномочиями разрешения и доступа к кэш и буферу записи.

Обычно макроядро ARM940T используется в тех применениях, в которых конечный пользователь никогда не добавляет программное обеспечение к системе - типа процессоров, встраиваемых в платы сетевого интерфейса, принтеры, TV и Internet приставки и в автомобильных применениях.

**Макроядро ARM920T**, также оснащенная кэш команд и данных, но емкостью по 16 Кбайт, используется высокопроизводительная подсистема CPU на основе ядра ARM9TDMI, с буфером записи и интерфейсом AMBA ASB шины, но вместо блока защиты макроядро оснащено полным MMU - блоком поддержки виртуальной памяти.

Макроядро ARM920T ориентировано на использование в "открытых" системах, для которых необходимо полное управление виртуальной памятью - к таким системам относятся персональные информационные устройства (PDA), сетевые компьютеры, смартфоны, X- и Windows-терминалы.

## Семейство ARM9E

Процессорное ядро ARM9E - это процессор ARM9TDMI<sup>TM</sup>, расширенный DSP возможностями и предназначенный для таких применений, в которых необходимо сочетание возможностей микроконтроллера и DSP.

Основными областями применения ядра могут быть контроллеры HDD, DVD и других устройств массовой памяти; контроллеры устройств распознавания и синтеза речи, средств кодирования и распространения речи по сетям и через Internet; устройства Dolby AC3 и MPEG MP3; персональные информационные устройства (PDA), торговые терминалы, аппаратные и в особенности программные модемы.

Ядра ARM946E и ARM966E являются макроядрами, реализованными на основе ядра ARM9E и предназначенными для интеграции в ASIC, ASSP и приборы класса SOC.

**Макроядро ARM946E**, в котором ядро ARM9E объединено с ассоциативным кэш, буфером записи и устройством защиты памяти, предназначено для встраиваемых применений, работающих с операционными системами реального времени.

Архитектура кэш дает возможность разработчикам изменять размер кэш в соответствии с требованиями применения.

**В макроядре ARM966E** ядро ARM9E объединено с буфером записи и жестко присоединенной статической памятью с произвольным доступом (SRAM), и это макроядро ориентировано на применения «действительно реального времени», в которых высокая производительность и малое потребление обеспечиваются без использования кэш.

Макроядра ARM946E и ARM966E сверху вниз совместимы на уровне кодов с процессорами семейств ARM7™ Thumb, ARM9™ Thumb.

## Семейство ARM10 Thumb

Всего через год после обнародования сведений о разработке процессоров семейства ARM9 Thumb были обнародованы (октябрь 1998 года) сведения о разработке процессоров нового еще более производительного семейства - семейства ARM10 Thumb, обеспечивающего производительность выше 400 MIPS.

Главной особенностью этого семейства нужно считать наличие векторного сопроцессора вычислений с плавающей точкой – подтверждение того, что в одном приборе внедрены управляющие целочисленные процессоры и процессоры цифровой обработки сигналов.

В основе процессоров семейства ARM10 Thumb целочисленное ядро ARM10TDMI, использующее ARM 32-разрядную RISC систему команд, сжатую 16-разрядную систему команд Thumb и расширенные Multi-ICE средства отладки программного обеспечения. Ядро процессора ARM10TDMI - первая реализация ARMV5T Архитектуры Системы команд (*Instruction Set Architecture - ISA*).

Ядро ARM10TDMI оснащено пятиуровневым конвейером, реализует параллельное выполнение команд, предсказание переходов и способно продолжать работу при неудачном обращении к кэш, обеспечивая высокую производительность в реальных применениях.

В настоящее время в семейство входит в ARM1020T - кэшированное макроядро процессора, сформированное на основе ядра ARM10TDMI и оснащенное встроенными кэш команд и данных емкостью по 32 Кбайт, виртуальная память (MMU), поддерживающим виртуальную память с подкачкой страниц по требованию, буфером записи, и широкополосным шинным интерфейсом AMBA, класса «система на кристалле».

Сопроцессор векторных вычислений с плавающей точкой VFP10 (Vector Floating Point - VFP) интегрируется на тот же кристалл что и процессор ARM1020T в тех применениях, для которых он необходим.

В конвейере вычислений с плавающей точкой VFP10 используется два конвейера - пятиуровневый конвейер команд загрузки и памяти, а также семиуровневый конвейер арифметических команд. Эти конвейеры совместно используют два первых уровня, что ограничивает количество выполняемых операций одной командой, но из-за векторного характера архитектуры VFP параллельно могут быть выполнены команды векторной арифметики и векторной загрузки или команда сохранения.

## Микропроцессоры семейства StrongARM

### Микропроцессор SA-110

StrongARM высокопроизводительные микропроцессоры, которые появились в результате сотрудничества фирмы ARM с Digital Equipment Corporation.

Первым членом семейства StrongARM стал ориентированный на встраиваемые применения микропроцессор SA-110 (StrongARM-110), выпущенный еще в сентябре 1996 года.

RISC 32-разрядная Гарвардская архитектура ARM с пятиуровневым конвейером в сочетании с низковольтной CMOS технологией и большим опытом разработки высокопроизводительных процессоров фирмы Digital позволили разработать высокопроизводительный микропроцессор с исключительным соотношением производительность/потребление.

Версия с тактовой частотой 160 МГц обеспечивает производительность 2.1 MIPS, потребляя всего 450 мВт, что соответствует соотношению производительность/потребление (MIPS/Bt) выше 400.

## Микропроцессор SA-1100

Всего через год после появления микропроцессора SA-110 фирма Digital Equipment Corporation расширила семейство StrongARM, выпустив микропроцессор SA-1100. Ориентированный на дальнейшее расширение возможностей и технических характеристик карманных PC, суб-ноутбуков, смартфонов, высокопроизводительных встраиваемых применений, микропроцессор StrongARM SA-1100, обеспечил в три-пять раз большую производительность, чем конкурирующие процессоры.

Высокая производительность микропроцессора SA-1100 обеспечивается высокой тактовой частотой истроенными кэш большого объема, включающими кэш команд емкостью 16 Кбайт (Icache), кэш обратной записи данных (Dcache) и миникэш, обеспечивающими дополнительную гибкость при управлении перемещением данных.

Используются встроенные буферы, модуль быстродействующего умножителя с функцией процессора цифрового сигнала (DSP).



процессоров на базе архитектуры Cortex-M3 даже в самых простых приложениях.

В основе процессора на базе архитектуры Cortex-M3 лежит ядро, выполненное по Гарвардской технологии с 3-ступенчатым конвейером, что обеспечивает такие привлекательные возможности, как предсказание переходов, однотактное умножение и аппаратно реализованное деление, что обеспечивает производительность порядка 1.25 MIPS/МГц (в тесте Dhrystone).

Несмотря на то, что ядро Cortex-M3 разрабатывалось как недорогое ядро, оно остается 32-битным микропроцессорным ядром и, в связи с этим, поддерживает два режима работы: потоковый режим (Thread) и режим обработчика (Handler). Благодаря этому появляется возможность разработки более интеллектуального программного обеспечения и поддержки операционных систем реального времени (ОСРВ).

В ядро Cortex также входит 24-битный автоматически перезагружаемый таймер, предназначенный для генерации

периодических прерываний и используемый ядром ОСРВ. Если у микропроцессоров ARM7 и ARM9 имеется два набора инструкций (32-битный ARM и 16-битный Thumb), то у семейства Cortex предусмотрена поддержка набора инструкций ARM Thumb-2. Этот набор представляет собой смесь 16- и 32-битных инструкций, которые позволяют добиться производительности 32-битного набора инструкций ARM и плотности кода, свойственной 16-битному набору инструкций Thumb.

Thumb-2 - обширный набор инструкций, ориентированный на компиляторы языков C/C++. Это означает, что программа для Cortex-микроконтроллера может быть полностью написана на языке программирования C/C++.

Семейство STM32 состоит из двух групп. Группа Performance Line работает на тактовых частотах до 72 МГц и оснащена полным набором устройств ввода-вывода (УВВ, англ. I/O), а группа Access Line работает на частотах до 36 МГц и интегрирует ограниченный набор устройств ввода-вывода.

Иногда фирма ARM ссылается на свои процессоры по наименованию версии архитектуры. В этих обозначениях версия архитектуры процессора Cortex-M3 определяется как ARMV7M. Иными словами, процессор Cortex-M3 выполнен по архитектуре ARMV7M и поддерживает исполнение инструкций Thumb-2.

### **Микропроцессорное ядро Cortex**

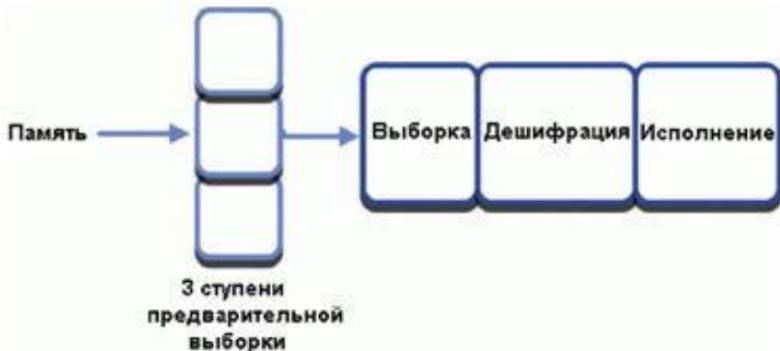
Основой процессора Cortex является 32-битное микропроцессорное ядро, имеющее RISC-систему команд. Данное ядро использует упрощенную модель программирования ARM7/9, но при этом более обширный набор инструкций с хорошей поддержкой целочисленной арифметики, улучшенными битовыми операциями и более строгими реально-временными характеристиками.

### **Конвейер**

Микропроцессорное ядро Cortex способно выполнять большинство инструкций за один цикл, что достигается с помощью

трехступенчатого конвейера: выборка-десиффрация-выполнение. Cortex поддерживает предсказание переходов для минимизации количества перезагрузок конвейера. Во время выполнения одной инструкции следующая инструкция десиффрируется, а третья инструкция считывается из памяти. Этот механизм отлично работает с линейным кодом, но если требуется выполнить переход, то, прежде чем продолжить выполнение кода программы, потребуется очистка и перезагрузка конвейера.

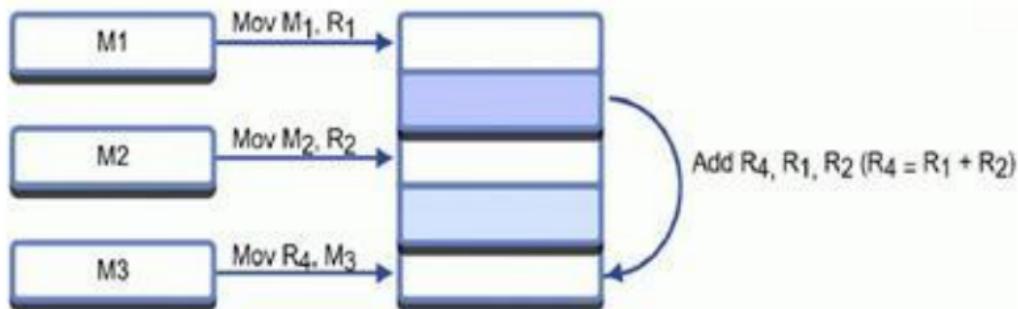
Трехступенчатый конвейер микропроцессора Cortex оснащен логикой предсказания переходов. Это означает, что при достижении инструкции условного перехода выполняется упреждающая выборка. В результате оба назначения инструкции условного перехода будут доступны для исполнения и не произойдет снижения производительности. Хуже обстоят дела с инструкциями косвенного перехода, т. к. в этом случае упреждающую выборку выполнить нельзя и перезагрузка конвейера может оказаться неизбежной.



Трехступенчатый конвейер

## Модель программирования

ЦПУ Cortex является RISC-процессором, который выполнен по архитектуре чтения/ записи. Для выполнения операций обработки данных вначале необходимо поместить операнды из памяти в центральный регистровый файл, затем выполнить требуемую операцию над данными в регистрах и, наконец, перезаписать результат обратно в память.



Cortex-M3 выполнен по архитектуре чтения/записи. Все данные, перед выполнением инструкции их обработки, необходимо поместить в центральный регистровый файл. Следовательно, вся активность программы фокусируется вокруг регистрационного файла ЦПУ. Данный регистровый файл образуют шестнадцать 32-битных регистров.

### **Регистровый файл**

Микропроцессорное ядро Cortex-M3 содержит регистровый файл, состоящий из шестнадцати 32-битных регистров (R0-R15). У регистров R13-R15 имеются особые функции.



Рис. 9. Регистр статуса программы: ICI - возобновляемая прерыванием инструкция;  
IT - поле 'if then'; ISR - процедура обработки прерывания

R13 выступает в роли указателя стека (SP). Данный регистр является банковским, что делает возможной работу Cortex в двух режимах работы, в каждом из которых используется свое собственное пространство стека. Данная возможность обычно используется операционными системами реального времени, которые могут выполнять свой системный код в защищенном режиме. У двух стеков Cortex имеются собственные наименования: основной стек и стек процесса.

R14 - регистр связи (LR). Он используется для хранения адреса возврата из подпрограммы. Благодаря этому регистру Cortex быстро переходит к подпрограмме и выходит из нее. Если же в программе

используется несколько уровней вложений подпрограмм, то компилятор будет автоматически сохранять R14 в стек.

R15 - счетчик программы (PC). Он является частью центрального регистрового файла, его чтение и обработка могут выполняться аналогично любым другим регистрам.

### Регистр статуса программы

Помимо регистрового файла, имеется отдельный регистр XPSR, который называется регистром статуса программы. Он не входит в основной регистровый файл, а доступ к нему возможен с помощью двух специальных инструкций. В XPSR хранятся значения полей, влияющих на исполнение инструкций Cortex. Регистр статуса программы содержит поля статуса, от которых зависит исполнение инструкций. Данный регистр разделен еще на три поля: статуса прикладной программы, исполнения программы и прерываний. Регистр статуса программы содержит поля, от которых зависит исполнение инструкций. Данный регистр разделен на три поля:

статуса прикладной программы, исполнения программы и прерываний (рис. 9). Биты регистра XPSR разделены на три группы, к каждой из которых возможен доступ по собственному наименованию.

Верхние пять бит (флаги кода условия) именуются полем статуса прикладной программы. Первые четыре флага кода условия N, Z, C, V (индикация отрицательного (N) или нулевого (Z) результата, переноса (C) и переполнения (V)) устанавливаются и сбрасываются по итогам выполнения инструкции обработки данных. Пятый бит Q используется при выполнении математических инструкций с насыщением алгоритмов цифровой обработки сигналов для индикации достижения переменной своего максимального или минимального значения. Так же как и 32-битные инструкции ARM, некоторые инструкции Thumb-2 выполняются только при условии совпадения кода условия инструкции и состояния флагов регистра статуса прикладной программы.

Если коды условия инструкции не совпадают, то инструкция проходит по конвейеру как NOP (нет операции). Этим гарантируется

равномерность прохождения инструкций по конвейеру и минимизируется число перезагрузок конвейера. У Cortex данный способ расширен полем статуса исполнения программы, который связан с битами регистра XPSR. Это поле состоит из трех полей: поле 'if then' (IT), поле возобновляемой прерыванием инструкции и поле инструкции Thumb. Набор инструкций Thumb-2 реализует эффективный метод выполнения компактных блоков инструкций типа 'if then'. Если проверяемое условие истинно, записью значения в поле IT можно сигнализировать микропроцессору о необходимости выполнения до четырех следующих инструкций. Если же проверяемое условие - ложное, то данные инструкции пройдут по конвейеру как NOP.

Несмотря на то, что большинство инструкций Thumb-2 выполняются за один цикл, некоторые инструкции (например, инструкции чтения/записи) требуют для выполнения несколько циклов. Чтобы точно знать время отклика микропроцессорного ядра Cortex на прерывания, данные инструкции должны быть прерываемыми. В

случае преждевременного прекращения исполнения инструкции в поле возобновляемых прерываниями инструкций запоминается номер следующего регистра, подлежащего обработке инструкцией многократного чтения или записи. Таким образом, сразу после завершения процедуры обработки прерывания выполнение инструкции многократного чтения/записи может быть восстановлено. Последнее поле Thumb предусмотрено для совместимости с предшествующими версиями микропроцессорного ядра ARM. Данное поле сигнализирует, что в настоящий момент микропроцессорное ядро выполняет инструкцию ARM или Thumb. У микропроцессорного ядра Cortex-M3 данный бит всегда равен единице. Наконец, в поле статуса прерывания хранится информация о любых приостановленных запросах прерывания.

### **Режимы работы микропроцессора**

Процессор Cortex разрабатывался как быстродействующее, простое в использовании и с малым числом логических элементов

микроконтроллерное ядро, в нем была учтена поддержка для работы операционных систем реального времени (ОСРВ). Процессор Cortex поддерживает два режима работы: режим Thread (или потоковый режим) и режим Handler (или режим обработчика).

Микропроцессор запускается в режиме Thread при непрерывном, фоновом выполнении инструкций и переключается в режим Handler при обработке исключительных ситуаций. Кроме того, микропроцессор Cortex может выполнять код программы в привилегированном или непривилегированном режиме. В привилегированном режиме ЦПУ имеет доступ ко всему набору инструкций, а в непривилегированном некоторые инструкции отключаются (например, инструкции MRS и MSR, осуществляющие доступ к регистру XPSR и его битовым группам).

## Системный интерфейс

Процессор Cortex-M3 выполнен по гарвардской архитектуре, которая подразумевает использование раздельных шин данных (D-bus)

и инструкций (I-bus). Обе эти шины могут осуществлять доступ к инструкциям и данным в диапазоне адресов 0x00000000-0x1 FFFFFFF. Также имеется дополнительная системная шина (System), которая предоставляет доступ к области системного управления по адресам 0x20000000-0xDFFFFFFF и 0xE0100000-0xFFFFFFFF.

У встроенной отладочной системы процессора Cortex имеется еще одна дополнительная шинная структура, которая называется локальной шиной УВВ (Private Peripheral Bus, PPB). Системная шина и шина данных подключаются к внешнему микроконтроллеру через набор высокоскоростных шин, называемых матрицей шин. Она образует несколько параллельных соединений между шинами Cortex и другими внешними шинными мастерами, как, например, каналы ПДП к встроенным ресурсам, статическое ОЗУ и УВВ. Если два шинных мастера (например, микропроцессорное ядро Cortex и канал ПДП) предпринимают попытку доступа к одному и тому же УВВ, то вступит в действие внутренний арбитр, который

разрешит конфликт, предоставив доступ к шине тому, кто имеет наивысший приоритет. Однако, благодаря тесной взаимосвязи блоков ПДП с микропроцессорным ядром Cortex, необходимость арбитража во многих случаях исключается.

## Организация памяти Cortex-M3

Процессор Cortex-M3 является стандартизованным микроконтроллерным ядром, и поэтому его карта памяти четко расписана (рис. 10). Несмотря на использование нескольких внутренних шин, адресное пространство является линейным и имеет размер 4 Гб.

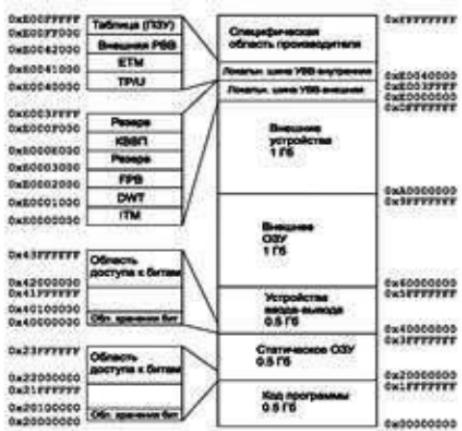


Рис. 10. Организация памяти микропроцессорного ядра Cortex-M3

Первый гигабайт памяти разделен равномерно между областью кода программы и областью статического ОЗУ. Пространство кода программы оптимизировано для работы с шиной I-Code. Аналогично пространство статического ОЗУ доступно через шину D-code. Несмотря на то, что в области статического ОЗУ поддерживается загрузка и исполнение инструкций, их выборка осуществляется через системную шину, что требует дополнительного состояния ожидания. Таким образом, выполнение кода программы из статического ОЗУ

будет более медленным, чем из встроенной флэш-памяти, расположенной в области кода программы.

Следующие полигигабайта памяти - область встроенных устройств ввода-вывода (УВВ). В этой области находятся все предоставляемые пользователю производителем микроконтроллера УВВ. Первый мегабайт в каждой области памяти (статическое ОЗУ и УВВ) является битноадресуемым. Для этого используется метод bit banding. Таким образом, все данные, хранящиеся в этих областях, могут обрабатываться как по слово, так и побитно.

Следующие два гигабайта адресного пространства выделены для внешних статического ОЗУ и устройства ввода-вывода (УВВ). Последние полигигабайта зарезервированы для системных ресурсов и будущих расширений процессора Cortex. Все регистры процессора Cortex расположены по фиксированным адресам во всех Cortex-микроконтроллерах. Благодаря этому облегчается портирование программ. Для процессора Cortex-M3 определена фиксированная карта памяти размером 4 Гб, в которой выделены конкретные области для

хранения кода программы, статического ОЗУ, устройств ввода-вывода, внешней памяти и устройств, а также системных регистров. Карта памяти одинакова для всех Cortex-микроконтроллеров.

## **Мультиядерные ARM**

Процессоры с ядрами ARM на данный момент являются достаточно популярными для мобильных устройств и встраиваемых систем различного применения. Самые массовые из них — это мобильные телефоны, смартфоны, коммуникаторы, мультимедийные устройства. Архитектура обладает такими привлекательными свойствами, как удобная и эффективная система команд, мощная поддержка при разработке аппаратной базы и программного обеспечения, высокая энергоэффективность. Относительно недавно было заявлено о разработке мультиядерной архитектуры на базе ARM, что открывает данным процессорам путь на рынок высокопроизводительных приложений. В частности, одно из возможных применений —

ноутбуки, сравнимые, а то и превосходящие их аналоги на архитектуре x86.

На данный момент компанией ARM представлены нескольких мультиядерных архитектур: ARM11 MPCore , Cortex-R5 MPCore MPCore, Cortex-R7 MPCore, Cortex-A7 MPCore, Cortex-A9 MPCore и Cortex-A15 MPCore, Cortex-A17 MPCore. Каждая из этих архитектур может масштабироваться четырех нескольких ядер. Код, написанный для одиночных процессоров, также может исполняться и на мультиядерных.

Целевая область применения процессоров MPCore лежит в области мобильных приложений с высокими требованиями по производительности совместно с ограниченными энергетическими ресурсами. Благодаря масштабируемой пиковой производительности данный процессор может достаточно легко справляться с требованиями современных высокопроизводительных встраиваемых приложений при сохранении инвестиций в программное обеспечение в условиях развивающегося рынка.

## Общая характеристика мультиядерных ARM-процессоров

Процессоры MPCore поддерживает полностью когерентный кэш данных, существенно упрощая как симметричный, так и асимметричный мультипроцессинг, собственно, как и любую другую мультипроцессорную технологию.

Производительность приложений увеличивается благодаря возможности разделения ядрами данных кэша, возможности распределения и балансирования вычислительной нагрузки между процессорами, портирования многозадачных приложений, а также масштабируемости приложений за счет эффективной загрузки процессора многопоточными приложениями, характерными для современного программного обеспечения. Возможность передачи данных между кэшами процессоров позволяет процессорам эффективно разделять данные без необходимости доступа в память. Оптимизированный кэш первого уровня существенно ускоряет операции с данными при сохранении достаточно низкого энергопотребления. Аппаратная реализация индексации и тэгирования

данных кэша снимают временные издержки при устранении наложения адресов или необходимости очистки кэша в ходе смены контекста в операционной системе.

Кэш данных используется как при операциях чтения, так и при записи данных совместно с адаптируемым буфером записи, который позволяет существенно снизить количество обращений к основной памяти и может формировать запросы на массированную передачу данных из нескольких запросов к памяти. Уникальная система кэш-памяти ускоряет выделение пространства кэша, в результате чего оно выполняется всего за один цикл.

Процессоры MPCore позволяют производителю использовать одни и те же ядра с различными конфигурациями для продуктов с различными свойствами и требованиями.

К настоящему времени лицензия на выпуск процессоров ARM MPCore приобретена более чем 15 компаниями, включая Broadcom, NEC Electronics, NVIDIA, Renesas Technology, Toshiba and Sarnoff Corporation, и воплощена в большом количестве приложений и

устройств, представленных на современном рынке. Технология существенно расширяет спектр приложений, предлагая более эффективные модели операций.

Все мультиядерные решения от ARM базируются на шинной архитектуре AMBA 3 AXI, дающей возможность подключать к процессорам не только память и периферийные устройства, но и другие процессоры.

Шинный интерфейс процессоров MPCore и масштабируемость позволяют настраивать производительность системы, оптимизировать ее энергопотребление и снижать общую стоимость решения и риск морального старения при переходе к следующему поколению цифровых устройств.

Интеграция с существующими системными компонентами также снижает риски, связанные, например, с поддержкой операционных систем и продуктов на базе данных процессоров. Работает стандартная для ARM-архитектур модель программирования с поддержкой существующих операционных систем и приложений. Доступны

совместимые с Linux 2.6 SMP операционные системы и инструменты разработки.

Занимаемая процессорами площадь на кристалле, диапазон рабочих частот и потребляемая мощность зависят от использованного при реализации технологического процесса, библиотек компонентов и оптимизации.

Несмотря на различия в ядрах и некоторые различия в построении мультиядерных вариантов процессоров, есть ряд технологий, поддерживаемый ими всеми.

Технологии ускорения выполнения Java-приложений — **Jazelle DBX** и **Jazelle RCT** для оптимизации процесса адаптивной компиляции "на лету" (Just In Time (ЛТ) and Dynamic Adaptive Compilation (DAC)), а также для уменьшения расхода памяти — максимум в три раза.

Технология **TrustZone** предназначена для обеспечения безопасности транзакций, управления цифровыми сертификатами, создания базы для проверки и защиты прав (Digital Rights Management (DRM)).

# ARM11 MPCore

Синтезируемый процессор ARM11 MPCore поддерживает микроархитектуру ARM11 и может содержать от одного до четырех процессоров (рис. 11), достигая производительности до 2600 DMIPS; имеет расширенную полосу пропускания памяти порядка 1,3 Гб/с для одиночного процессора.

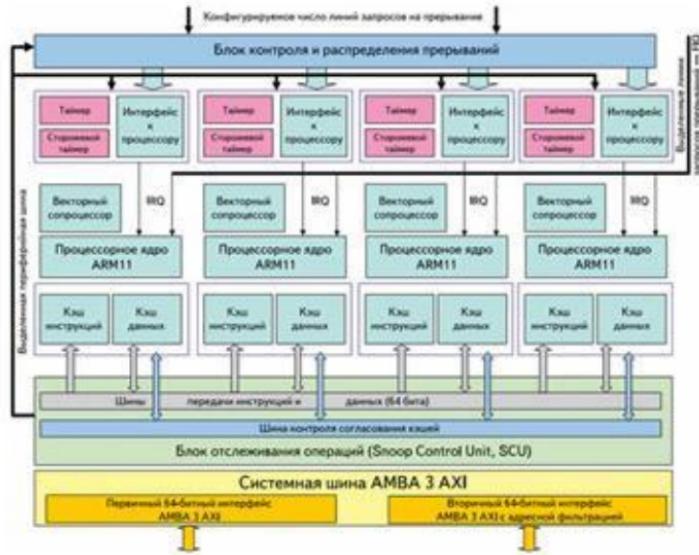


Рис. 11. Структура процессора ARM11 MPCore

Процессоры ARM11 MPCore поддерживают архитектуру ARMv6 с поддержкой Thumb, расширений цифровой обработки сигналов, SIMD мультимедийной обработки данных и ARM Jazelle Java.

У процессора высокопроизводительная подсистема памяти. Каждый процессор имеет свои независимые кэши данных и инструкций с поддержкой согласования данных. Размер кэшей инструкций и данных можно независимо изменять в пределах от 16 Кб до 64 Кб индивидуально для каждого ядра.

Поддерживается 64-битный интерфейс AMBA AXI с одиночной или двойной 64-битной шинной системой AMBA 3 AXI. Системная 64-битная шина AMBA 3 AXI упрощает обмен данными в системе при достаточно большой полосе пропускания и простой системе тактирования.

Векторный сопроцессор (Vector Floating Point coprocessors) работает с числами в формате с плавающей точкой.

Блок контроля и распределения прерываний является программируемым — возможно сконфигурировать до 255 независимых источников аппаратных прерываний

Добавлена система управления энергопотреблением: мультипроцессор имеет возможность отключать неиспользуемые ресурсы и процессоры (Adaptive Shutdown), что в итоге дает динамическое энергопотребление порядка 0,49 мВт/МГц — экономится до 85% энергии.

ARM11 MPCore позволяет разработчикам систем на кристалле рассматривать отдельный процессор как одиночный процессор, упрощая тем самым процесс разработки и уменьшая время выхода продукта на рынок.

### **Серия Cortex-A MPCore**

Процессоры семейства Cortex-A MPCore — Cortex-A5 MPCore и Cortex-A9 MPCore — помимо небольшой занимаемой площади и

энергоэффективности обладают богатым арсеналом возможностей и функциональностью архитектуры ARMv7, что в итоге дает высокую производительность и низкое энергопотребление, как на специфических прикладных приложениях, так и для устройств общего плана. Поддерживается также технология Thumb-2, обеспечивающая высокую производительность при одновременном уменьшении размера кода на 30%.

Процессоры, входящие в состав мультипроцессоров Cortex-A MPCore, имеют блок операций с плавающей точкой, способный выполнять операции с одинарной и двойной точностью. Он обладает примерно вдвое большей производительностью, чем предыдущие версии ARM FPU (Floating Point Unit).

Архитектура ARM Cortex-A5/A9 имеет мультимедийное 128-битное SIMD (одиночный поток команд, множественный поток данных) расширение архитектуры — NEON, предназначенное для поддержки мультимедийных операций и функций цифровой обработки сигналов (например, для ускорения работы таких алгоритмов, как H.264 или

MP3). Также данный модуль расширяет систему команд набором инструкций ARM **NEON** Advanced SIMD, впервые представленным с процессором Cortex-A8.

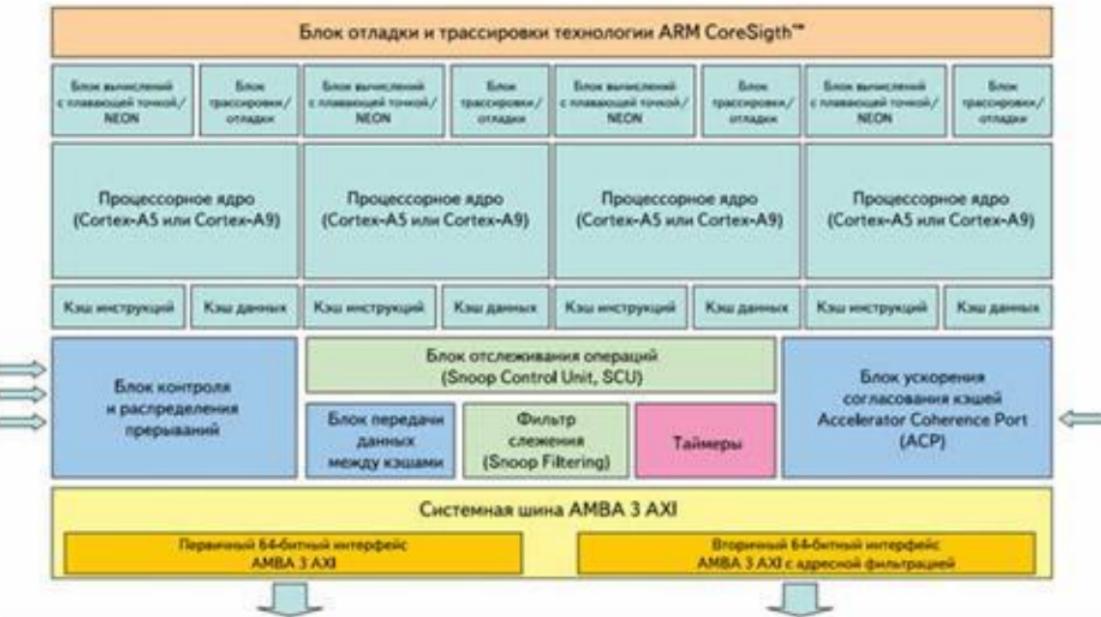


Рис. 12. Структура процессоров Cortex-A MPCore

## ARM Cortex-A9 MPCore

Процессор Cortex-A9 MPCore имеет возможность оптимизации производительности приложений и по скорости выполнения, и по потребляемой мощности.

Основные возможности включают в себя следующее:

Энергоэффективный суперскалярный конвейер производительностью более 2,0 DMIPS/МГц.

Оптимизированный по производительности и потребляемой мощности кэш первого уровня совмещает минимальное время задержки и минимальное энергопотребление. Добавлен контроллер кэша второго уровня, позволяющий осуществлять доступ с малыми временами задержки и высокой пропускной способностью к кэш памяти размером до 2 Мб.

Мультипроцессор Cortex-A9 MPCore иллюстрирует практически линейную масштабируемость производительности на различных тестах.

Процессоры ARM Cortex-A9 — и одиночный вариант (рис. 13), и мультипроцессор — поддерживают ряд специфических расширений ARM-архитектуры, в том числе: DSP, SIMD, Jazelle , TrustZone, Intelligent Energy Manager (IEM TM). В дополнение к этому ARM разработала ряд поддерживающих технологий для сокращения времени разработки и сокращения времени выхода продукта на рынок.

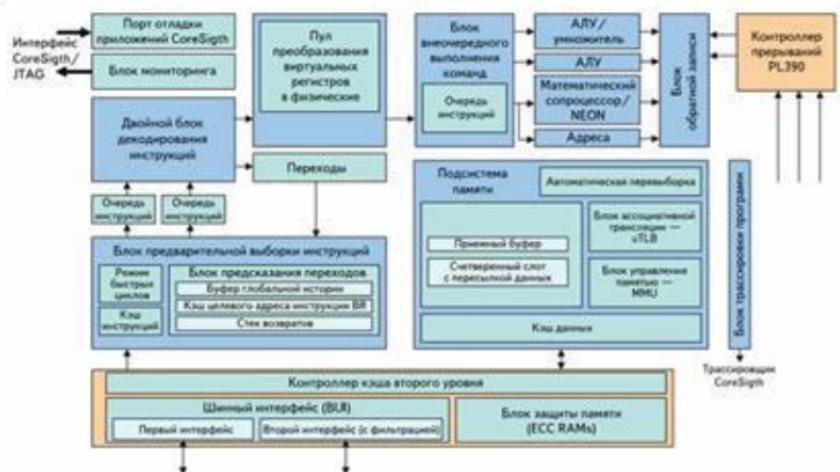


Рис. 13. Структура процессора ARM Cortex-A9