Министерство сельского хозяйства Российской Федерации
Трубчевский аграрный колледж -
филиал федерального государственного бюджетного образовательного
учреждения высшего образования
«Брянский государственный аграрный университет»

Живодеров А. Н.

Основы архитектуры, устройство и функционирование вычислительных систем

КУРС ЛЕКЦИЙ

для специальности 09.02.04 Информационные системы (по отраслям)

Брянская область
2020

УДК 004.2 (042)
ББК 32.973.26-02
Ж 67

Живодеров, А. Н. **Основы архитектуры, устройство и функционирование вычислительных систем:** курс лекций для обучающихся по специальности 09.02.04 Информационные системы (по отраслям) / А. Н. Живодеров. – Брянск: Изд-во Брянский ГАУ, 2020. – 83 с.

Курс лекций дисциплины ОП.01. Основы архитектуры, устройство и функционирование вычислительных систем для специальности 09.02.04 Информационные системы (по отраслям) содержат краткий теоретический курс, отражающий требования ФГОС СПО специальности. В курсе лекций предусмотрены вопросы для самопроверки.

Курс лекций предназначен для обучающихся и преподавателей специальностей 09.02.04 Информационные системы (по отраслям).

Составитель:

Живодеров А.Н. – преподаватель первой категории Трубчевского филиала ФГБОУ ВО Брянский ГАУ

Рецензент:

Сидоренко Л.М. – к.э.н., методист Трубчевского филиала ФГБОУ ВО Брянский ГАУ, высшая категория.

Рекомендации одобрены к печати методическим советом филиала, протокол № 2 от 29 ноября 2019 г.

СОДЕРЖАНИЕ

ЛЕКЦИЯ 1. СТАНОВЛЕНИЕ И ЭВОЛЮЦИЯ ЦИФРОВОЙ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Вычислительная машина — это комплекс технических и программных средств, предназначенный для автоматизации подготовки и решения задач пользователей.

Вычислительная система это совокупность взаимосвязанных и взаимодействующих процессоров или вычислительных машин, периферийного оборудования и программного обеспечения, предназначенную для подготовки и решения задач пользователей.

Под архитектурой вычислительной машины обычно понимается логическое построение ВМ, то есть то, какой машина представляется программисту. Из рассмотрения выпадают вопросы физического построения вычислительных средств: состав устройств, число регистров процессора, емкость памяти, наличие специального блока для обработки вещественных чисел, тактовая частота центрального процессора и т.д. Этот круг вопросов принято определять понятием организация вычислительной машины.

УРОВНИ ДЕТАЛИЗАЦИИ СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНОЙ МАШИНЫ

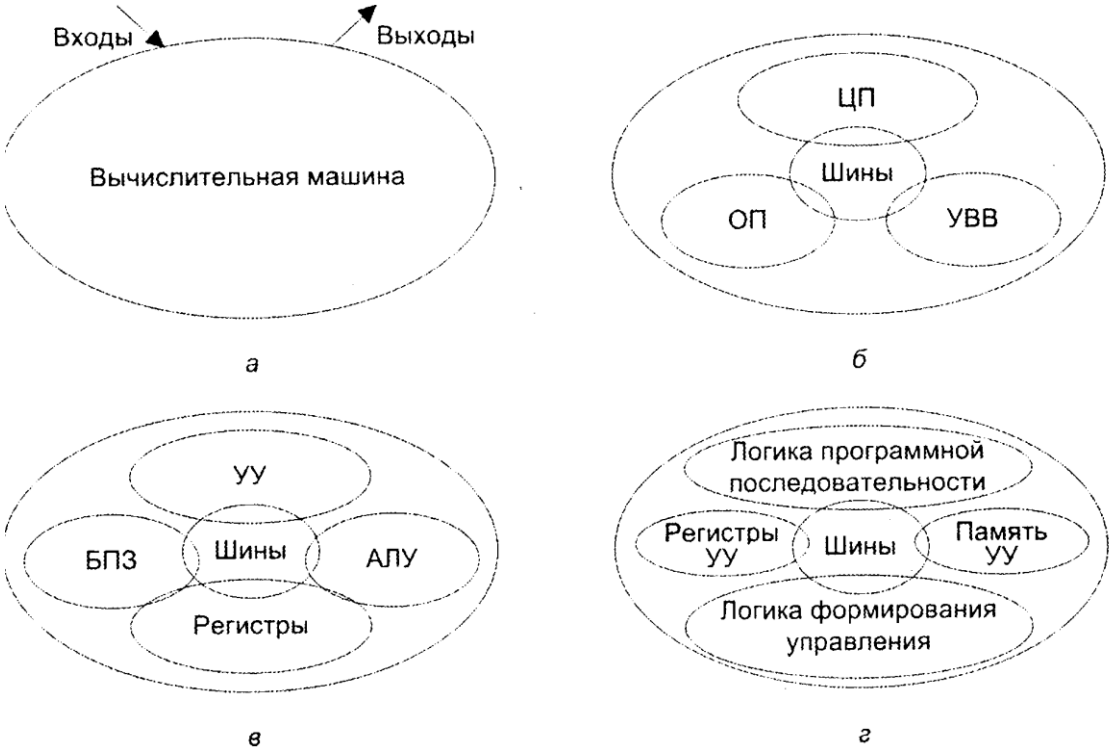


Рис. 1.1. Уровни детализации вычислительной машины:
а — уровень «черного ящика»;
б — уровень общей архитектуры;
в — уровень архитектуры центрального процессора;
г — уровень архитектуры устройства управления

На первом уровне вычислительная машина рассматривается как устройство, способное хранить и обрабатывать информацию, а также обмениваться данными с внешним миром (см. рис. 1.1, а). ВМ представляется «черным ящиком», который может быть подключен к коммуникационной сети и к которому, в свою очередь, могут подсоединяться периферийные устройства.

Уровень общей архитектуры (см. рис. 1.1,6) предполагает представление ВМ в виде четырех составляющих: центрального процессора (ЦП), основной памяти (ОП), устройства ввода/вывода (УВВ) и системы шин.

На третьем уровне детализируется каждое из устройств второго уровня. Для примера взят центральный процессор (см. рис. 1.1, в). В простейшем варианте в нем можно выделить:

арифметико-логическое устройство (АЛУ), обеспечивающее обработку целых чисел; блок обработки чисел в формате с плавающей запятой (БПЗ); регистры процессора, использующиеся для краткосрочного хранения команд, данных и адресов; устройство управления (УУ), обеспечивающее совместное функционирование устройств ВМ; внутренние шины.

На четвертом уровне детализируются элементы третьего уровня. Так, на рис. 1.1, г раскрыта структура устройства управления. УУ представлено в виде четырех составляющих: логики программной последовательности — электронных схем, обеспечивающих выполнение команд программы в последовательности, предписываемой программой; регистров и дешифраторов устройства управления; управляющей памяти; логики формирования управления, генерирующей все необходимые управляющие сигналы.

Применительно к параллельным и распределенным многопроцессорным и многомашинным вычислительным системам зачастую вводят понятие «метауровня».

ЭВОЛЮЦИЯ СРЕДСТВ АВТОМАТИЗАЦИИ ВЫЧИСЛЕНИЙ

Попытки облегчить, а в идеале автоматизировать процесс вычислений имеют давнюю историю, насчитывающую более 5000 лет. С развитием науки и технологий средства автоматизации вычислений непрерывно совершенствовались. Современное состояние вычислительной техники (ВТ) являет собой результат многолетней эволюции. В последнее время вопросы развития ВТ стали предметом особо пристального внимания ученых, свидетельством чего служит активно развивающаяся новая область знаний, получившая название «Теория эволюции компьютеров» (Computer evolution theory). Создатели теории обратили внимание на сходство закономерностей эволюции вычислительной техники и эволюции в биологии. В основу новой науки положены следующие постулаты:

- самозарождение «живых» вычислительных систем из «неживых» элементов (в биологии это явление известно как *абиогенез);*
- поступательное продвижение по древу эволюции — от однопроцессорных вычислительных машин к многопроцессорным вычислительным системам;
- прогресс в технологии вычислительных систем как следствие полезных мутаций и вариаций;
- отмирание устаревших технологий в результате естественного отбора;
- закон Мура – плотность транзисторов на кремниевой подложке удваивается каждые 18-24 месяца, соответственно в два раза растет их производительность и в два раза падает их рыночная стоимость.

В традиционной трактовке эволюцию вычислительной техники представляют как последовательную смену поколений ВТ.

НУЛЕВОЕ ПОКОЛЕНИЕ (1492-1945)

Для полноты картины упомянем два события, произошедшие до нашей эры: первые счеты — абак, изобретенные в древнем Вавилоне за 3000 лет до н. э., и их более «современный» вариант с косточками на проволоке, появившийся в Китае примерно за 500 лет так же до н. э.

«Механическая» эра (нулевое поколение) в эволюции ВТ связана с механическими, а позже — электромеханическими вычислительными устройствами. Основным элементом механических устройств было зубчатое колесо. Начиная с XX века роль базового элемента переходит к электромеханическому реле.

ПЕРВОЕ ПОКОЛЕНИЕ (1937-1953)

На роль первой в истории электронной вычислительной машины в разные периоды претендовало несколько разработок. Общим у них было использование схем на базе электронно-вакуумных ламп вместо электромеханических реле. Предполагалось, что электронные ключи будут значительно надежнее, поскольку в них отсутствуют движущиеся части,

однако технология того времени была настолько несовершенной, что по надежности электронные лампы оказались ненамного лучше, чем реле. Однако у электронных компонентов имелось одно важное преимущество: выполненные на них ключи могли переключаться примерно в тысячу раз быстрее своих электромеханических аналогов.

ВТОРОЕ ПОКОЛЕНИЕ(1954-1962)

Второе поколение характеризуется рядом достижений в элементной базе, структуре и программном обеспечении. Принято считать, что поводом для выделения нового поколения ВМ стали технологические изменения, и, главным образом, переход от электронных ламп к полупроводниковым диодам и транзисторам со временем переключения порядка 0,3 мс.

ТРЕТЬЕ ПОКОЛЕНИЕ(1963-1972)

Третье поколение ознаменовалось резким увеличением вычислительной мощности ВМ, ставшим следствием больших успехов в области архитектуры, технологии и программного обеспечения. Основные технологические достижения связаны с переходом от дискретных полупроводниковых элементов к интегральным микросхемам и началом применения полупроводниковых запоминающих устройств, начинающих вытеснять ЗУ на магнитных сердечниках. Существенные изменения произошли и в архитектуре ВМ. Это, прежде всего, микропрограммирование как эффективная техника построения устройств управления сложных процессоров, а также наступление эры конвейеризации и параллельной обработки. В области программного обеспечения определяющими вехами стали первые операционные системы и реализация режима разделения времени.

ЧЕТВЕРТОЕ ПОКОЛЕНИЕ (1972-1984)

Отсчет четвертого поколения обычно ведут с перехода на интегральные микросхемы большой (large-scale integration, LSI) и сверхбольшой (very large-scale integration, VLSI) степени интеграции. К первым относят схемы, содержащие около 1000 транзисторов на кристалле, в то время как число транзисторов на одном кристалле VLSI имеет порядок 100 000. При таких уровнях интеграции стало возможным уместить в одну микросхему не только центральный процессор, но и вычислительную машину (ЦП, основную память и систему ввода/вывода).

ПЯТОЕ ПОКОЛЕНИЕ(1984-1990)

Главным поводом для выделения вычислительных систем второй половины 80-х годов в самостоятельное поколение стало стремительное развитие ВС с сотнями процессоров, ставшее побудительным мотивом для прогресса в области параллельных вычислений. Ранее параллелизм вычислений выражался лишь в виде конвейеризации, векторной обработки и распределения работы между небольшим числом процессоров. Вычислительные системы пятого поколения обеспечивают такое распределение задач по множеству процессоров, при котором каждый из процессоров может выполнять задачу отдельного пользователя.

ШЕСТОЕ ПОКОЛЕНИЕ (1990-)

На ранних стадиях эволюции вычислительных средств смена поколений ассоциировалась с революционными технологическими прорывами. Каждое из первых четырех поколений имело четко выраженные отличительные признаки и вполне определенные хронологические рамки. Последующее деление на поколения уже не столь очевидно и может быть понятно лишь при ретроспективном взгляде на развитие вычислительной техники. Пятое и шестое поколения в эволюции ВТ — это отражение нового качества, возникшего в результате последовательного накопления частных достижений, главным образом в архитектуре вычислительных.

КОНЦЕПЦИЯ МАШИНЫ С ХРАНИМОЙ В ПАМЯТИ ПРОГРАММОЙ

Алгоритм — одно из фундаментальных понятий математики и вычислительной техники. Международная организация стандартов (ISO) формулирует понятие алгоритм как «конечный набор предписаний, определяющий решение задачи посредством конечного количества операций» (ISO 2382/1-84).

Основными свойствами алгоритма являются: дискретность, определенность, массовость и результативность.

Дискретность выражается в том, что алгоритм описывает действия над дискретной информацией (например, числовой или символьной), причем сами эти действия также дискретны.

Свойство определенности> означает, что в алгоритме указано все, что должно быть сделано, причем ни одно из действий не должно трактоваться двояко.

Массовость алгоритма подразумевает его применимость к множеству значений исходных данных, а не только к каким-то уникальным значениям.

Наконец, результативность алгоритма состоит в возможности получения результата за конечное число шагов.

Рассмотренные свойства алгоритмов предопределяют возможность их реализации на ВМ, при этом процесс, порождаемый алгоритмом, называют вычислительным процессом.

В основе архитектуры современных ВМ лежит представление алгоритма решения задачи в виде программы последовательных вычислений. Согласно стандарту ISO 2382/1-84, программа для ВМ — это «упорядоченная последовательность команд, подлежащая обработке».

ВМ, где определенным образом закодированные команды программы хранятся в памяти, известна под названием вычислительной машины с хранимой в памяти программой. Идея принадлежит создателям вычислителя ENIAC Эккерту, Моч-ли и фон Нейману.

Сущность фон-неймановской концепции вычислительной машины можно свести к четырем принципам:

- двоичного кодирования;
- программного управления;
- однородности памяти;
- адресности.

ПРИНЦИП ДВОИЧНОГО КОДИРОВАНИЯ

Согласно этому принципу, вся информация, как данные, так и команды, кодируются двоичными цифрами 0 и 1. Каждый тип информации представляется двоичной последовательностью и имеет свой формат. Последовательность битов в формате, имеющая определенный смысл, называется полем. В числовой информации обычно выделяют поле знака и поле значащих разрядов. В формате команды можно выделить два поля:

- поле кода операции;
- поле адресов (адресную часть).

Код операции представляет собой указание, какая операция должна быть выполнена, и задается с помощью двоичной комбинации.

Вид адресной части и число составляющих ее адресов зависят от типа команды: в командах преобразования данных АЧ содержит адреса объектов обработки (операндов) и результата; в командах изменения порядка вычислений — адрес следующей команды программы; в командах ввода/вывода — номер устройства ввода/ вывода. Адресная часть также представляется двоичной последовательностью. Таким образом, команда в вычислительной машине имеет вид (длина кода операции + длина поля адресов)-разрядной двоичной комбинации.

ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ

Все вычисления, предусмотренные алгоритмом решения задачи, должны быть представлены в виде программы, состоящей из последовательности управляющих слов — команд. Каждая команда предписывает некоторую операцию из набора операций, реализуемых вычислительной машиной. Команды программы хранятся в последовательных ячейках памяти вычислительной машины и выполняются в естественной последовательности, то есть в порядке их положения в программе. При необходимости, с помощью специальных команд, эта последовательность может быть изменена. Решение об изменении порядка выполнения команд программы принимается либо на основании анализа результатов предшествующих вычислений, либо безусловно.

ПРИНЦИП ОДНОРОДНОСТИ ПАМЯТИ

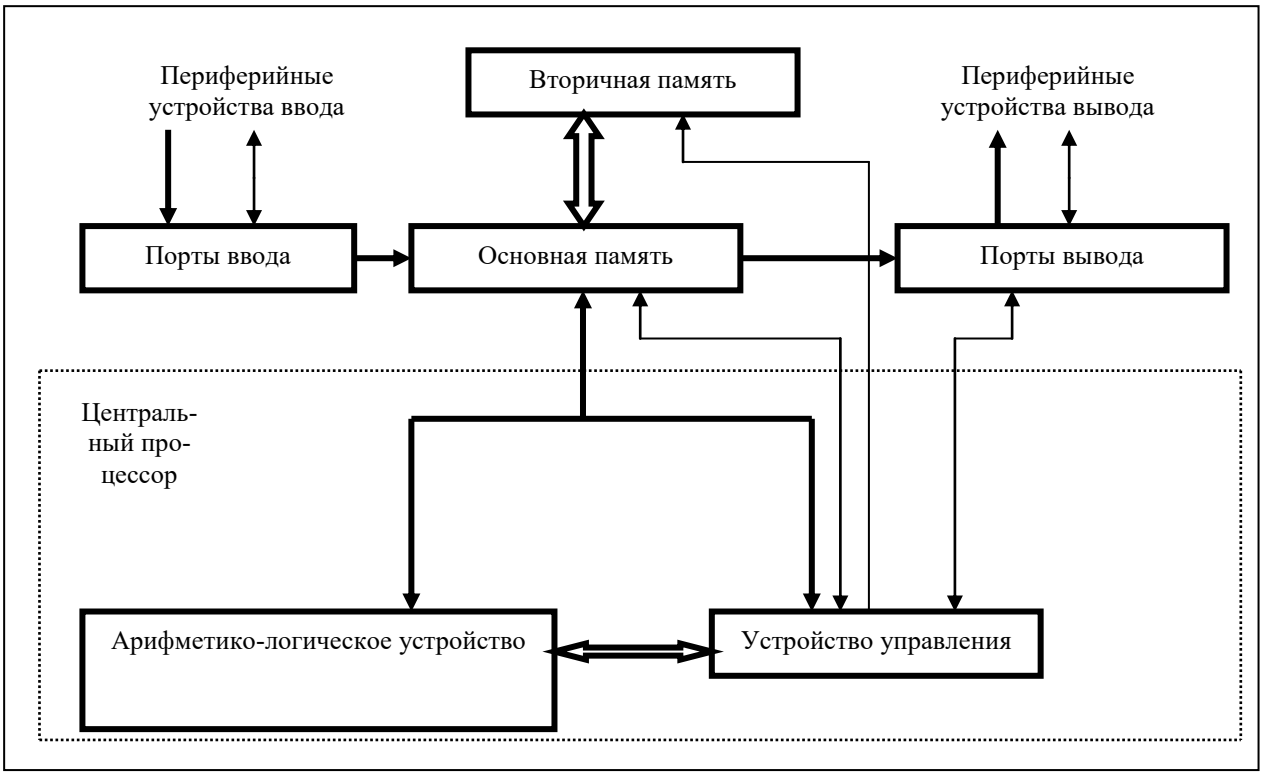
Команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы. Распознать их можно только по способу использования. Это позволяет производить над командами те же операции, что и над числами, и, соответственно, открывает ряд возможностей. Так, циклически изменяя адресную часть команды, можно обеспечить обращение к последовательным элементам массива данных. Более полезным является другое следствие принципа однородности, когда команды одной программы могут быть получены как результат исполнения другой программы.

ПРИНЦИП АДРЕСНОСТИ

Структурно основная память состоит из пронумерованных ячеек, причем процессору в произвольный момент доступна любая ячейка. Двоичные коды команд и данных разделяются на единицы информации, называемые словами, и хранятся в ячейках памяти, а для доступа к ним используются номера соответствующих ячеек — адреса.

ФОН-НЕЙМАНОВСКАЯ АРХИТЕКТУРА

Большинство современных ВМ по своей структуре отвечают принципу программного управления. Типичная фон-неймановская ВМ содержит: память, устройство управления, арифметико-логическое устройство и устройство ввода/вывода.



В любой ВМ имеются средства для ввода программ и данных к ним. Информация поступает из подсоединенных к ЭВМ периферийных устройств ввода. Результаты вычислений выводятся на периферийные устройства вывода. Связь и взаимодействие вычислительной машины и периферийных устройств обеспечивают порты ввода и порты вывода. Термином порт обозначают аппаратуру сопряжения периферийного устройства с ВМ и управления им. Совокупность портов ввода и вывода называют устройством ввода/вывода (УВВ) или модулем ввода/вывода ВМ (МВБ).

Введенная информация сначала запоминается в основной памяти, а затем переносится во вторичную память, для длительного хранения. Чтобы программа могла выполняться, команды и данные должны располагаться в основной памяти (ОП), организованной таким образом, что каждое двоичное слово хранится в отдельной ячейке, идентифицируемой адресом, причем соседние ячейки памяти имеют следующие по порядку адреса. Доступ к любым ячейкам запоминающего устройства (ЗУ) основной памяти может производиться в произвольной последовательности. Такой вид памяти известен как память с произвольным доступом. ОП современных ВМ в основном состоит из полупроводниковых оперативных запоминающих устройств (ОЗУ), обеспечивающих как считывание, так и запись информации. Для таких ЗУ характерна энергозависимость — хранимая информация теряется при отключении электропитания. Если необходимо, чтобы часть основной памяти была энергонезависимой, в состав ОП включают постоянные запоминающие устройства (ПЗУ), также обеспечивающие произвольный доступ. Хранящаяся в ПЗУ информация может только считываться (но не записываться).

Размер ячейки основной памяти обычно принимается равным 8 двоичным разрядам — байту. Для храпения больших чисел используются 2,4 или 8 байтов, размещаемых в ячейках с последовательными адресами. В этом случае за адрес числа часто принимается адрес его младшего байта. Так, при хранении 32-разрядного числа в ячейках с адресами 200, 201, 202 и 203 адресом числа будет 200.

Для долговременного хранения больших программ и массивов данных в ВМ обычно имеется дополнительная память, известная как вторичная. Вторичная память энергонезависима и чаще всего реализуется на базе магнитных дисков. Информация в ней хранится в виде специальных программно поддерживаемых объектов — файлов (согласно стандарту ISO, файл — это «идентифицированная совокупность экземпляров полностью описанного в конкретной программе типа данных, находящихся вне программы во внешней памяти и доступных программе посредством специальных операций»).

Устройство управления (УУ) — важнейшая часть ВМ, организующая автоматическое выполнение программ (путем реализации функций управления) и обеспечивающая функционирование ВМ как единой системы. Для пояснения функций УУ ВМ следует рассматривать как совокупность элементов, между которыми происходит пересылка информации, в ходе которой эта информация может подвергаться определенным видам обработки. Пересылка информации между любыми элементами ВМ инициируется своим сигналом управления (СУ), то есть управление вычислительным процессом сводится к выдаче нужного набора СУ в нужной временной последовательности. Основной функцией УУ является формирование управляющих сигналов, отвечающих за извлечение команд из памяти в порядке, определяемом программой, и последующее исполнение этих команд. Кроме того, УУ формирует СУ для синхронизации и координации внутренних и внешних устройств ВМ.

Еще одной неотъемлемой частью ВМ является арифметико-логическое устройство (АЛУ). АЛУ обеспечивает арифметическую и логическую обработку двух входных переменных, в результате которой формируется выходная переменная. Функции АЛУ обычно сводятся к простым арифметическим и логическим операциям, а также операциям сдвига. Помимо результата операции АЛУ формирует ряд признаков результата (флагов), характеризующих полученный результат и события, произошедшие в процессе его получения (равенство нулю, знак, четность, перенос, переполнение и т.д.). Флаги могут анализироваться в УУ с целью принятия решения о дальнейшей последовательности выполнения команд программы.

УУ и АЛУ тесно взаимосвязаны и их обычно рассматривают как единое устройство, известное как центральный процессор (ЦП) или просто процессор. Помимо УУ и АЛУ в процессор входит также набор регистров общего назначения (РОН), служащих для промежуточного хранения информации в процессе ее обработки.

ТИПЫ СТРУКТУР ВЫЧИСЛИТЕЛЬНЫХ МАШИН И СИСТЕМ

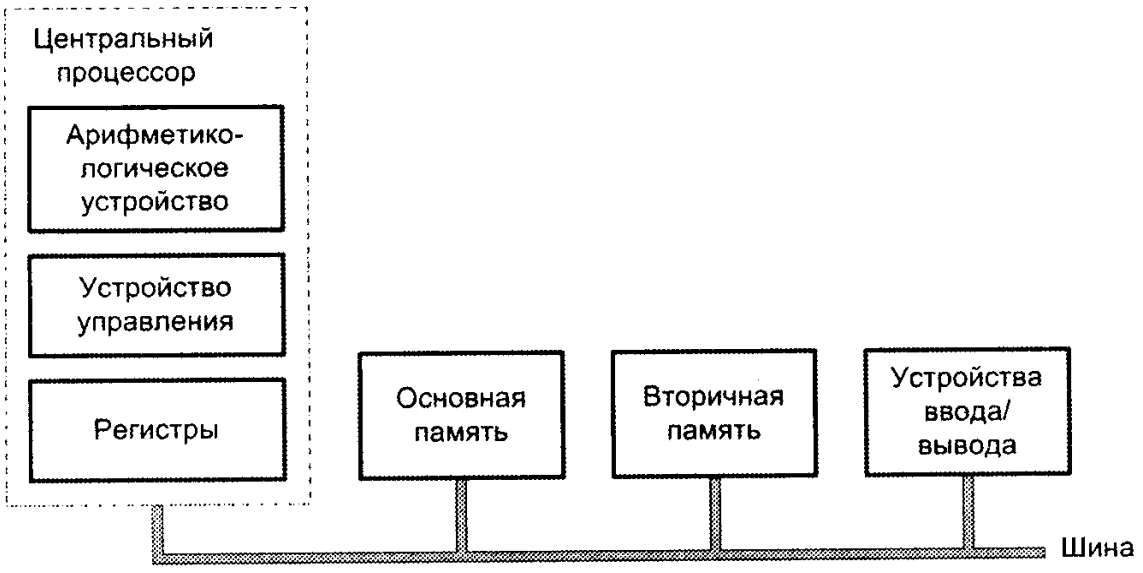
Достоинства и недостатки архитектуры вычислительных машин и систем изначально зависят от способа соединения компонентов. При самом общем подходе можно говорить о двух основных типах структур вычислительных машин и двух типах структур вычислительных систем.

СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

В настоящее время примерно одинаковое распространение получили два способа построения вычислительных машин: с непосредственными связями и на основе шины.

Типичным представителем первого способа может служить классическая фон-неймановская ВМ. В ней между взаимодействующими устройствами (процессор, память, устройство ввода/вывода) имеются непосредственные связи. Особенности связей (число линий в шинах, пропускная способность и т. п.) определяются видом информации, характером и интенсивностью обмена. Достоинством архитектуры с непосредственными связями можно считать возможность развязки «узких мест» путем улучшения структуры и характеристик только определенных связей, что экономически может быть наиболее выгодным решением. У фон-неймановских ВМ таким «узким местом» является канал пересылки данных между ЦП и памятью, и «развязать» его достаточно непросто. Кроме того, ВМ с непосредственными связями плохо поддаются реконфигурации.

В варианте с общей шиной все устройства вычислительной машины подключены к магистральной шине, служащей единственным трактом для потоков команд, данных и управления. Наличие общей шины существенно упрощает реализацию ВМ, позволяет легко менять состав и конфигурацию машины. Благодаря этим свойствам шинная архитектура получила широкое распространение в мини-и микроЭВМ. Вместе с тем, именно с шиной связан и основной недостаток архитектуры: в каждый момент передавать информацию по шине может только одно устройство. Основную нагрузку на шину создают обмены между процессором и памятью, связанные с извлечением из памяти команд и данных и записью в память результатов вычислений. На операции ввода/вывода остается лишь часть пропускной способности шины. Практика показывает, что даже при достаточно быстрой шине для 90% приложений этих остаточных ресурсов обычно не хватает, особенно в случае ввода или вывода больших массивов данных.



В целом следует признать, что при сохранении фон-неймановской концепции последовательного выполнения команд программы шинная архитектура в чистом ее виде оказывается недостаточно эффективной. Более распространена архитектура с иерархией шин, где помимо магистральной шины имеется еще несколько дополнительных шин. Они могут обеспечивать непосредственную связь между устройствами с наиболее интенсивным обменом, например процессором и кэш-памятью. Другой вариант использования дополнительных шин — объединение однотипных устройств ввода/вывода с последующим выходом с дополнительной шины на магистральную. Все эти меры позволяют снизить нагрузку на общую шину и более эффективно расходовать ее пропускную способность.

СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Понятие «вычислительная система» предполагает наличие множества процессоров или законченных вычислительных машин, при объединении которых используется один из двух подходов.

В вычислительных системах с общей памятью имеется общая основная память, совместно используемая всеми процессорами системы. Связь процессоров с памятью обеспечивается с помощью коммуникационной сети, чаще всего вырождающейся в общую шину. Таким образом, структура ВС с общей памятью аналогична рассмотренной выше архитектуре с общей шиной, в силу чего ей свойственны те же недостатки. Применительно к вычислительным системам данная схема имеет дополнительное достоинство: обмен информацией между процессорами не связан с дополнительными операциями и обеспечивается за счет доступа к общим областям памяти.

Альтернативный вариант организации — распределенная система, где общая память вообще отсутствует, а каждый процессор обладает собственной локальной памятью. Часто такие системы объединяют отдельные ВМ. Обмен информацией между составляющими системы обеспечивается с помощью коммуникационной сети посредством обмена сообщениями.

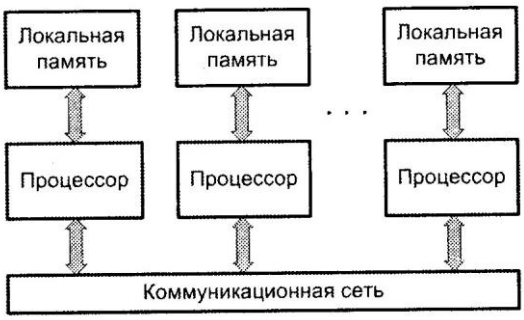


Рис 1.6. Структура распределенной вычислительной системы

Рис 1.5. Структура вычислительной системы с общей памятью

Подобное построение ВС снимает ограничения, свойственные для общей шины, но приводит к дополнительным издержкам на пересылку сообщений между процессорами или машинами.

ВОПРОСЫ САМОКОНТРОЛЯ

1. УРОВНИ ДЕТАЛИЗАЦИИ СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНОЙ МАШИНЫ
2. ЭВОЛЮЦИЯ СРЕДСТВ АВТОМАТИЗАЦИИ ВЫЧИСЛЕНИЙ
3. НУЛЕВОЕ ПОКОЛЕНИЕ (1492-1945)
4. ПЕРВОЕ ПОКОЛЕНИЕ (1937-1953)
5. ВТОРОЕ ПОКОЛЕНИЕ(1954-1962)
6. ТРЕТЬЕ ПОКОЛЕНИЕ(1963-1972)
7. ЧЕТВЕРТОЕ ПОКОЛЕНИЕ (1972-1984)
8. ПЯТОЕ ПОКОЛЕНИЕ(1984-1990)
9. ШЕСТОЕ ПОКОЛЕНИЕ (1990-)
10. КОНЦЕПЦИЯ МАШИНЫ С ХРАНИМОЙ В ПАМЯТИ ПРОГРАММОЙ
11. ПРИНЦИП ДВОИЧНОГО КОДИРОВАНИЯ
12. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ
13. ПРИНЦИП ОДНОРОДНОСТИ ПАМЯТИ
14. ПРИНЦИП АДРЕСНОСТИ
15. ФОН-НЕЙМАНОВСКАЯ АРХИТЕКТУРА
16. ТИПЫ СТРУКТУР ВЫЧИСЛИТЕЛЬНЫХ МАШИН И СИСТЕМ
17. СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ МАШИН
18. СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

ЛЕКЦИЯ 2. КЛАССИФИКАЦИЯ АРХИТЕКТУР СИСТЕМЫ КОМАНД

Системой команд вычислительной машины называют полный перечень команд, которые способна выполнять данная ВМ. В свою очередь, под архитектурой системы команд (АСК) принято определять те средства вычислительной машины, которые видны и доступны программисту. АСК можно рассматривать как линию согласования нужд разработчиков программного обеспечения с возможностями создателей аппаратуры вычислительной машины.

В истории развития вычислительной техники как в зеркале отражаются изменения, происходившие во взглядах разработчиков на перспективность той или иной архитектуры системы команд. Сложившуюся на настоящий момент ситуацию в области АСК иллюстрирует рис. 2.3.

Среди мотивов, чаще всего предопределяющих переход к новому типу АСК, остановимся на двух наиболее существенных. Первый — это состав операций, выполняемых вычислительной машиной, и их сложность. Второй — место хранения операндов, что влияет на количество и длину адресов, указываемых в адресной части команд обработки данных. Именно эти моменты взяты в качестве критериев излагаемых ниже вариантов классификации архитектур системы команд.

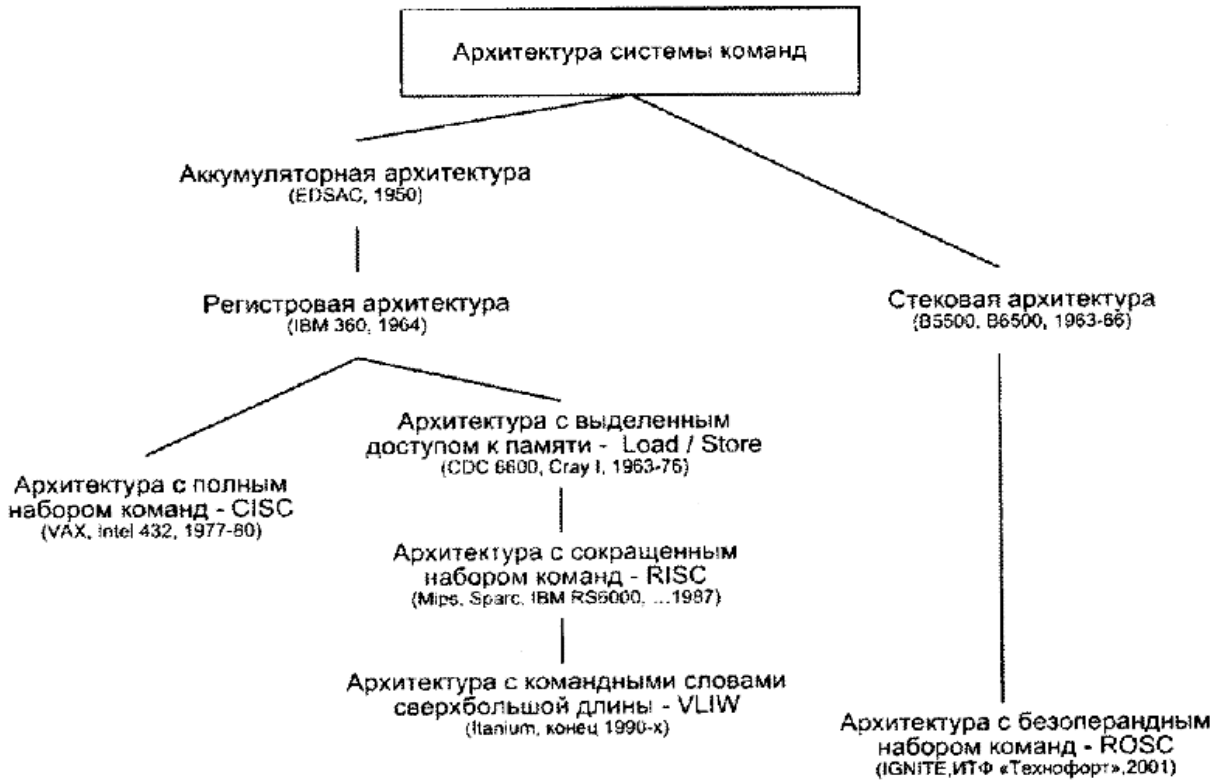


Рис. Хронология развития архитектур системы команд

КЛАССИФИКАЦИЯ ПО СОСТАВУ И СЛОЖНОСТИ КОМАНД

Современная технология программирования ориентирована на языки высокого уровня (ЯВУ), главная цель которых — облегчить процесс программирования. Переход к ЯВУ, однако, породил серьезную проблему: сложные операторы, характерные для ЯВУ, существенно отличаются от простых машинных операций, реализуемых в большинстве вычислительных машин. Проблема получила название *семантического разрыва,* а ее следствием становится недостаточно эффективное выполнение программ на ВМ. Пытаясь преодолеть семантический разрыв, разработчики вычислительных машин в настоящее время выбирают один из трех подходов и, соответственно, один из трех типов АСК:

- архитектуру с полным набором команд: CISC (Complex Instruction Set Computer);
- архитектуру с сокращенным набором команд: RISC (Reduced Instruction Set Computer);

- архитектуру с командными словами сверхбольшой длины: VLIW (Very Long Instruction Word).

В вычислительных машинах типа CISC проблема семантического разрыва решается за счет расширения системы команд, дополнения ее сложными командами, семантически аналогичными операторам ЯВУ. Основоположником CISC-архитектуры считается компания IBM, которая начала применять данный подход с семейства машин IBM 360 и продолжает его в своих мощных современных универсальных ВМ, таких как IBM ES/9000. Аналогичный подход характерен и для компании Intel в ее микропроцессорах серии 8086 и Pentium.

Для CISC-архитектуры типичны:

- наличие в процессоре сравнительно небольшого числа регистров общего назначения;
- большое количество машинных команд, некоторые из них аппаратно реализуют сложные операторы ЯВУ;
- разнообразие способов адресации операндов;
- множество форматов команд различной разрядности;
- наличие команд, где обработка совмещается с обращением к памяти.

К типу CISC можно отнести практически все ВМ, выпускавшиеся до середины 1980-х годов, и значительную часть производящихся в настоящее время. Рассмотренный способ решения проблемы семантического разрыва вместе с тем ведет к усложнению аппаратуры ВМ, главным образом устройства управления, что, в свою очередь, негативно сказывается на производительности ВМ в целом. Это заставило более внимательно проанализировать программы, получаемые после компиляции с ЯВУ. Был предпринят комплекс исследований, в результате которых обнаружилось, что доля дополнительных команд, эквивалентных операторам ЯВУ, в общем объеме программ не превышает 10-20%, а для некоторых наиболее сложных команд даже 0,2%. В то же время объем аппаратных средств, требуемых для реализации дополнительных команд, возрастает весьма существенно. Так, емкость микропрограммной памяти при поддержании сложных команд может увеличиваться на 60%. Детальный анализ результатов упомянутых исследований привел к серьезному пересмотру традиционных решений, следствием чего стало появление *RISC-архитектуры.* Термин RISC впервые был использован Д. Паттерсоном и Д. Дитцелем в 1980 году. Идея заключается в ограничении списка команд ВМ наиболее часто используемыми простейшими командами, оперирующими данными, размещенными только в регистрах процессорах. Обращение к памяти допускается лишь с помощью специальных команд чтения и записи. Резко уменьшено количество форматов команд и способов указания адресов операндов. Сокращение числа форматов команд и их простота, использование ограниченного количества способов адресации, отделение операций обработки данных от операций обращения к памяти позволяет существенно упростить аппаратные средства ВМ и повысить их быстродействие. RISC-архитектура разрабатывалась таким образом, чтобы уменьшить $T_{ВИЧ}$ за счет сокращения CPI и I/. Как следствие, реализация сложных команд за счет последовательности из простых, но быстрых RISC-команд оказывается не менее эффективной, чем аппаратный вариант сложных команд в CISC-архитектуре. Отметим, что в последних микропроцессорах фирмы Intel и AMD широко используются идеи, свойственные RISC-архитектуре, так что многие различия между CISC и RISC постепенно стираются.

Помимо CISC- и RISC-архитектур в общей классификации был упомянут еще один тип АСК — архитектура с командными словами сверхбольшой длины (VLIW). Концепция VLIW базируется на RISC-архитектуре, где несколько простых RISC-команд объединяются в одну сверхдлинную команду и выполняются параллельно. В плане АСК архитектура VLIW сравнительно мало отличается от RISC. Появился лишь дополнительный уровень параллелизма вычислений, в силу чего архитектуру VLIW логичнее адресовать не к вычислительным машинам, а к вычислительным системам. Таблица 2.1 позволяет оценить наиболее существенные различия в архитектурах типа CISC, RISC и VLIW.

Таблица 2.1. Сравнительная оценка CISC-, RISC- и VLIW-архитектур

Характеристика	**CISC**	**RISC**	**VLIW**
Длина команды	Варьируется	Единая	Единая
Расположение нолей и команде	Варьируется	Неизменное	Неизменное
Количество регистров	Несколько (часто специализированных)	Много регистров общего назначения	Много регистров общего назначения
Доступ к памяти	Может выполняться как часть команд различных типов	Выполняется только специальными командами	Выполняется только специальными командами

КЛАССИФИКАЦИЯ ПО МЕСТУ ХРАНЕНИЯ ОПЕРАНДОВ

Количество команд и их сложность, безусловно, являются важнейшими факторами, однако не меньшую роль при выборе АСК играет ответ на вопрос о том, где могут храниться операнды и каким образом к ним осуществляется доступ. С этих позиций различают следующие виды архитектур системы команд:

- стековую;
- аккумуляторную;
- регистровую;
- с выделенным доступом к памяти.

Выбор той или иной архитектуры влияет на принципиальные моменты: сколько адресов будет содержать адресная часть команд, какова будет длина этих адресов, насколько просто будет происходить доступ к операндам и какой, в конечном итоге, будет общая длина команд.

Стековая архитектура

Стеком называется память, по своей структурной организации отличная от основной памяти ВМ. Принципы построения стековой памяти детально рассматриваются позже, здесь же выделим только те аспекты, которые требуются для пояснения особенностей АСК на базе стека. Стек образует множество логически взаимосвязанных ячеек (рис. 2.4), взаимодействующих по принципу «последним вошел, первым вышел» (L1FO, Last In First Out).

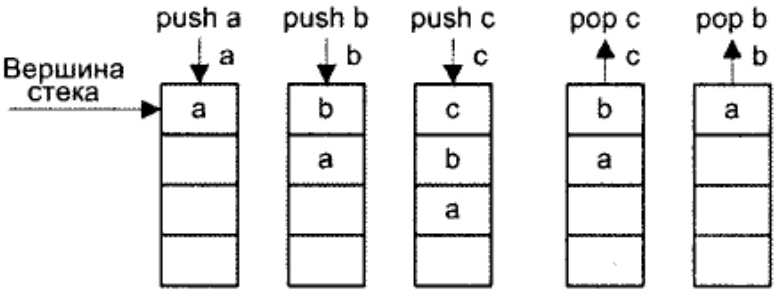


Рис. 2.4. Принцип действия стековой памяти

Верхнюю ячейку называют *вершиной стека*. Для работы со стеком предусмотрены две операции: *push* (проталкивание данных в стек) и *pop* (вытаякивание данных из стека). Запись возможна только в верхнюю ячейку стека, при этом вся хранящаяся в стеке информация предварительно проталкивается на одну позицию вниз. Чтение допустимо также только из вершины стека. Извлеченная информация удаляется из стека, а оставшееся его содержимое продвигается вверх. В вычислительных машинах, где реализована АСК на базе стека (их обычно называют стековыми), операнды перед обработкой помещаются в две верхних ячейки стековой памяти.

Аккумуляторная архитектура

Архитектура на базе аккумулятора исторически возникла одной из первых. В ней для хранения одного из операндов арифметической или логической операции в процессоре име-

ется выделенный регистр — *аккумулятор.* В этот же регистр заносится и результат операции. Поскольку адрес одного из операндов предопределен, в командах обработки достаточно явно указать местоположение только второго операнда. Изначально оба операнда хранятся в основной памяти, и до выполнения операции один из них нужно загрузить в аккумулятор. После выполнения команды обработки результат находится в аккумуляторе и, если он не является операндом для последующей команды, его требуется сохранить в ячейке памяти.

Регистровая архитектура

В машинах данного типа процессор включает в себя массив регистров (регистровый файл), известных как регистры общего назначения (РОН). Эти регистры, в каком-то смысле, можно рассматривать как явно управляемый кэш для хранения недавно использовавшихся данных. Размер регистров обычно фиксирован и совпадает с размером машинного слова. К любому регистру можно обратиться, указав его номер. Количество РОН в архитектурах типа CISC обычно невелико (от 8 до 32), и для представления номера конкретного регистра необходимо не более пяти разрядов, благодаря чему в адресной части команд обработки допустимо одновременно указать номера двух, а зачастую и трех регистров (двух регистров операндов и регистра результата). RISC-архитектура предполагает использование существенно большего числа РОН (до нескольких сотен), однако типичная для таких ВМ длина команды (обычно 32 разряда) позволяет определить в команде до трех регистров. Регистровая архитектура допускает расположение операндов в одной из двух запоминающих сред: основной памяти или регистрах. С учетом возможного размещения операндов в рамках регистровых АСК выделяют три подвида команд обработки: регистр-регистр; регистр-память; память-память.

В варианте «регистр-регистр» операнды могут находиться только в регистрах. В них же засылается и результат. Подтип «регистр-память» предполагает, что од и из операндов размещается в регистре, а второй в основной памяти. Результат обычно замещает один из операндов. В командах типа «память-память» оба операнда хранятся в основной памяти. Результат заносится в память. Каждому из вариантов свойственны свои достоинства и недостатки (табл. 2.4).

В выражениях вида *(m, n),* приведенных в первом столбце таблицы, *m* означает количество операндов, хранящихся в основной памяти, а n — общее число операндов в команде арифметической или логической обработки.

Вариант «регистр-регистр» является основным в вычислительных машинах типа RISC. Команды типа «регистр-память» характерны для CISC-машин. Наконец, вариант «память-память» считается неэффективным, хотя и остается в наиболее сложных моделях машин класса CISC.

Вариант	**Достоинства**	**Недостатки**
Регистр-регистр (0,3)	Простота реализации, фиксированная длина команд, простая модель формирования объектного кода при компиляции программ, возможность выполнения всех команд за одинаковое количество тактов	Большая длина объектного кода, из-за фиксированной длины команд часть разрядов в коротких командах не используется
Регистр-память (1.2)	Данные могут быть доступны без загрузки в регистры процессора, простота кодирования команд, объектный код получается достаточно компактным	Потеря одного из операндов при записи результата, длинное поле адреса памяти в коде команды сокращает место под помер регистра, что ограничивает общее число РОН. СР1 зависит от места размещения операнда
Память-память (3,3)	Компактность объектного кода, малая потребность в регистрах для хранения промежуточных данных	Разнообразие форматов команд и времени их исполнения, низкое быстродействие из-за обращения к памяти

Архитектура с выделенным доступом к памяти

В архитектуре с выделенным доступом к памяти обращение к основной памяти возможно только с помощью двух специальных команд: *load* и *store.* В английской транскрип-

ции данную архитектуру называют Load/Store architecture. Команда *load* (загрузка) обеспечивает считывание значения из основной памяти и занесение его в регистр процессора (в команде обычно указывается адрес ячейки памяти и номер регистра). Пересылка информации в противоположном направлении производится командой *store* (сохранение). Операнды во всех командах обработки информации могут находиться только в регистрах процессора (чаще всего в регистрах общего назначения). Результат операции также заносится в регистр. В архитектуре отсутствуют команды обработки, допускающие прямое обращение к основной памяти. Допускается наличие в АСК ограниченного числа команд, где операнд является частью кода команды.

АСК с выделенным доступом к памяти характерна для всех вычислительных машин с RISC-архитектурой. Команды в таких ВМ, как правило, имеют длину 32 бита и трехадресный формат. В качестве примеров вычислительных машин с выделенным доступом к памяти можно отметить HP PA-RISC, IBM RS/6000, Sun SPARC, MIPS R4000, DEC Alpha и т.д. К достоинствам АСК следует отнести простоту декодирования и исполнения команды.

СИСТЕМЫ СЧИСЛЕНИЯ

Понимание порядка представления чисел в двоичной, десятичной и шестнадцатеричной системах счисления является одним из необходимых условий успешного программирования. Система счисления – это совокупность правил записи чисел. Системы счисления подразделяются на позиционные и непозиционные. Непозиционные системы счисления появились раньше позиционных. Они характеризуются тем, что в них символы, обозначающие то или иное число (то есть цифры), не меняют своего значения в зависимости от местоположения в записи этого числа. Классическим примером такой системы счисления является римская. В ней для записи чисел используются буквы латинского языка. При этом буква I означает единицу, V – пять, X – десять, L – пятьдесят, C – сто, D – пятьсот, M – тысячу. Для получения количественного эквивалента числа в римской системе счисления необходимо просто просуммировать количественные эквиваленты входящих в него цифр. Исключение из этого правила составляет случай, когда младшая цифра находится перед старшей, - в этом случае нужно не складывать, а вычитать число вхождений этой цифры. Например: DLXXVII=500+50+10+10+5+1+1=577 или CDXXIX=500-100+10+10-1+10=429.

В позиционной системе счисления количество символов в наборе равно основанию системы счисления. Место каждой цифры в числе называется позицией. Номер позиции символа (за вычетом единицы) называется разрядом. Каждой цифре соответствует определенный количественный эквивалент. Введем обозначение – запись $A_{(p)}$ будет означать количественный эквивалент числа A, состоящего из n цифр $a_{(k)}$ (где k=0,…,n-1) в системе счисления с основанием p. Это число можно представить в виде последовательности цифр:

$A_{(p)}=an_{-1}an_{-2}…a_1a_0$. При этом, конечно, всегда выполняется неравенство $a_{(k)}<p$.

В общем случае количественный эквивалент некоторого положительного числа A в позиционной системе счисления можно представить выражением:

$$A_{(p)}=a_{n-1}*p^{n-1}+ a_{n-2}*p^{n-2}+…+ a_1*p^1+ a_0*p^0, \quad (1)$$

где p – основание системы счисления (некоторое целое положительное число), a – цифра данной системы счисления, n – номер старшего разряда числа.

Для получения количественного эквивалента числа в некоторой позиционной системе счисления необходимо сложить произведения количественных значений цифр на степени основания, показатели которых равны номерам разрядов (обратите внимание на то, что нумерация разрядов начинается с нуля).

Двоичная система счисления

Набор цифр для двоичной системы счисления – {0,1}, основание степени (p) – 2. Количественный эквивалент некоторого целого n-значного двоичного числа вычисляется согласно формуле (1):

$A_{(2)}=a_{n-1}*2^{n-1}+ a_{n-2}*2^{n-2}+\ldots+ a_1*2^1+ a_0*2^0$. (2)

Наличие этой системы обусловлено тем, что компьютер построен на логических схемах, имеющих в своем элементарном виде только два состояния – включено и выключено. Производить счет в двоичной системе просто для компьютера, но сложно для человека.

Рассмотрим двоичное число 10100111.

Вычислим десятичный эквивалент этого двоичного числа. Согласно формуле (2), это будет величина, равная следующей сумме:

$1*2^7+0*2^6+1*2^5+0*2^4+0*2^3+1*2^2+1*2^1+1*2^0=167$

Сложение и вычитание двоичных чисел выполняется так же, как и в других позиционных системах счисления, например десятичной. Точно так же выполняется заем (перенос) единицы из младшего разряда в старший разряд.

Шестнадцатеричная система счисления.

Шестнадцатеричная система счисления имеет набор цифр {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} и основание степени (р) – 16.

Количественный эквивалент некоторого целого n-значного шестнадцатеричного числа f45ed23c равен:

$15*16^7+4*16^6+5*16^5+14*16^4+13*16^3+2*16^2+3*16^1+12*16^0$.

Приведем соответствие двоичных чисел и их десятичных и шестнадцатеричных эквивалентов.

Десятичное число	Двоичная тетрада	Шестнадцатеричное число
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Поначалу запомнить эти соотношения сложно, поэтому полезно иметь под руками некоторую справочную информацию. Приведенная таблица содержит представления десятичных чисел из диапазона 0-16 в двоичной и шестнадцатеричной системах счисления. Ее удобно использовать для взаимного преобразования чисел в рассмотренных трех системах счисления. Шестнадцатеричная система счисления при вычислениях несколько сложнее, чем двоичная, в частности, в том, что касается правил переносов в старшие разряды. Главное здесь запомнить следующее равенство – $(1+F=10)_{16}$.

Перевод чисел из одной системы счисления в другую

Одного знания о существовании разных систем счисления мало. Для того, чтобы в полной мере использовать их в своей практической работе при программировании, необходимо научиться выполнять взаимное преобразование чисел между тремя системами счисления.

Перевод в десятичную систему счисления

Перевод в десятичную систему счисления является самым простым. Обычно его производят с помощью так называемого алгоритма замещения, суть которого заключается в следующем: сначала в десятичную систему счисления переводится основание степени p, а затем – цифры исходного числа. Результаты подставляются в формулу (1). Полученная сумма и будет искомым результатом.

Перевод в двоичную систему счисления

Перевод из десятичной системы счисления

Перевод числа в двоичную систему счисления из десятичной выполняется по следующему алгоритму:

1. Разделить десятичное число A на 2. Запомнить частное q и остаток a.

2. Если в результате шага 1 частное q не равно 0, то принять его за новое делимое и отметить остаток a, который будет очередной значащей цифрой, и вернуться к шагу 1, на котором в качестве нового делимого участвует полученное на шаге 2 частное.

3. Если в результате шага 1 частное q равно 0, алгоритм прекращается. Выписать остатки в порядке, обратном их получению. Получится двоичный эквивалент исходного числа.

Переведем в двоичную систему счисления число 247.

1 шаг.

Делим 247 на 2. Результат 123 остаток 1.

2 шаг.

Делим 123 на 2. Результат 61 остаток 1.

3 шаг.

Делим 61 на 2. Результат 30 остаток 1.

4 шаг.

Делим 30 на 2. Результат 15 остаток 0.

5 шаг.

Делим 15 на 2. Результат 7 остаток 1.

6 шаг.

Делим 7 на 2. Результат 3 остаток 1.

7 шаг.

Делим 3 на 2. Результат 1 остаток 1.

8 шаг.

Делим 1 на 2. Результат 0, есть остаток. (1)

Получаем следующее двоичное число: 11110111.

Перевод из шестнадцатеричной системы счисления.

Перевод из шестнадцатеричной системы счисления заключается в последовательной замене шестнадцатеричных цифр соответствующими двоичными тетрадами согласно таблице перевода. Например, шестнадцатеричному числу e4d5 соответствует двоичное число 1110 0100 1101 0101.

Перевод в шестнадцатеричную систему счисления.

Перевод из десятичной системы счисления.

Общая идея алгоритма перевода из десятичной системы счисления в шестнадцатеричную аналогична рассмотренной ранее в алгоритме перевода в двоичную систему счисления из десятичной.

1. Разделить десятичное число A на 16. Запомнить частное q и остаток a.

2. Если в результате шага 1 частное q не равно 0, то принять его за новое делимое, записать остаток и вернуться к шагу 1.

3. Если частное q равно 0, прекратить работу алгоритма. Выписать остатки в порядке, об-

ратном их получению. Получится шестнадцатеричный эквивалент исходного десятичного числа.

Переведем число 247.

1 шаг.

Делим 247 на 16. Результат 15 остаток 7.

2 шаг.

15 на 16 не делится. Результат 0 остаток 15.

Получаем число F7.

Перевод из двоичной системы счисления.

Идея алгоритма состоит в том, что двоичное число разбивается на тетрады, начиная с младшего разряда. Далее каждая тетрада приводится к к соответствующему шестнадцатеричному числу согласно таблице перевода.

1111 0111=F7

ВОПРОСЫ САМОКОНТРОЛЯ

1. КЛАССИФИКАЦИЯ ПО СОСТАВУ И СЛОЖНОСТИ КОМАНД
2. КЛАССИФИКАЦИЯ ПО МЕСТУ ХРАНЕНИЯ ОПЕРАНДОВ
3. СИСТЕМЫ СЧИСЛЕНИЯ

ЛЕКЦИЯ 3. ОРГАНИЗАЦИЯ ШИН

Совокупность трактов, объединяющих между собой основные устройства ВМ (центральный процессор, память и модули ввода/вывода), образует *структуру взаимосвязей* вычислительной машины.

Структура взаимосвязей должна обеспечивать обмен информацией между:

- центральным процессором и памятью;
- центральным процессором и модулями ввода/вывода;
- памятью и модулями ввода/вывода.

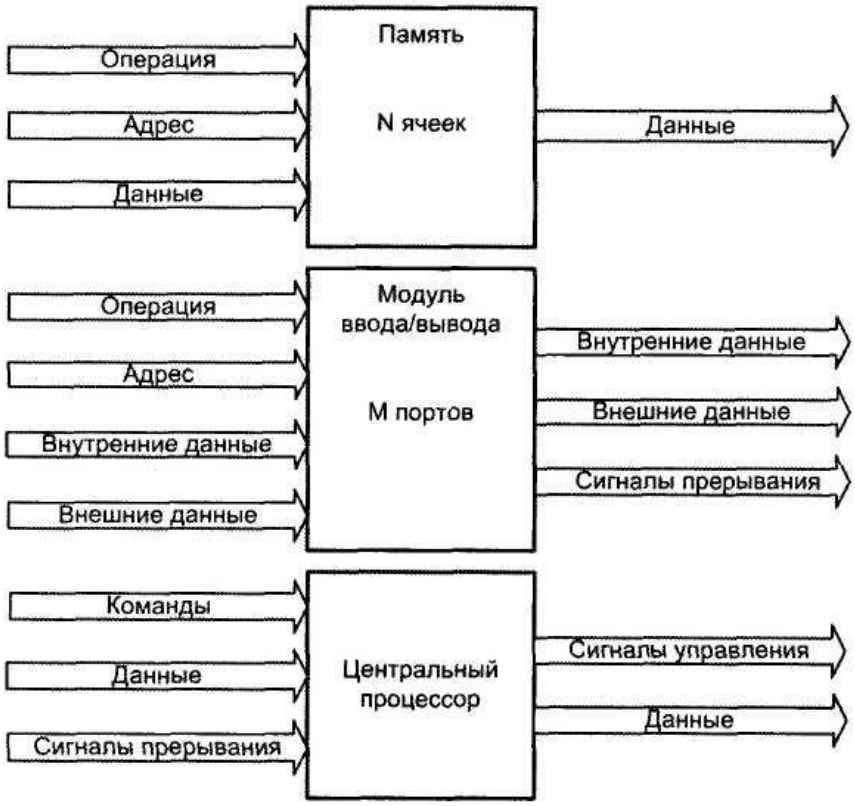


Рис. 4.1. Информационные потоки в вычислительной машине

С развитием вычислительной техники менялась и структура взаимосвязей устройств ВМ (рис. 4.2). На начальной стадии преобладали непосредственные связи между взаимодействующими устройствами ВМ. С появлением мини-ЭВМ, и особенно первых микроЭВМ, все более популярной становится схема с одной общей шиной. Последовавший за этим быстрый рост производительности практически всех устройств ВМ привел к неспособности единственной шины справиться с возросшим трафиком, и ей на смену приходят структуры взаимосвязей на базе нескольких шин. Дальнейшие перспективы повышения производительности вычислений связаны не столько с однопроцессорными машинами, сколько с многопроцессорными вычислительными системами. Способы взаимосвязей в таких системах значительно разнообразнее, и их рассмотрению посвящен один из разделов учебника. Возвращаясь к вычислительным машинам, более внимательно рассмотрим вопросы, связанные с организацией взаимосвязей на базе шин.

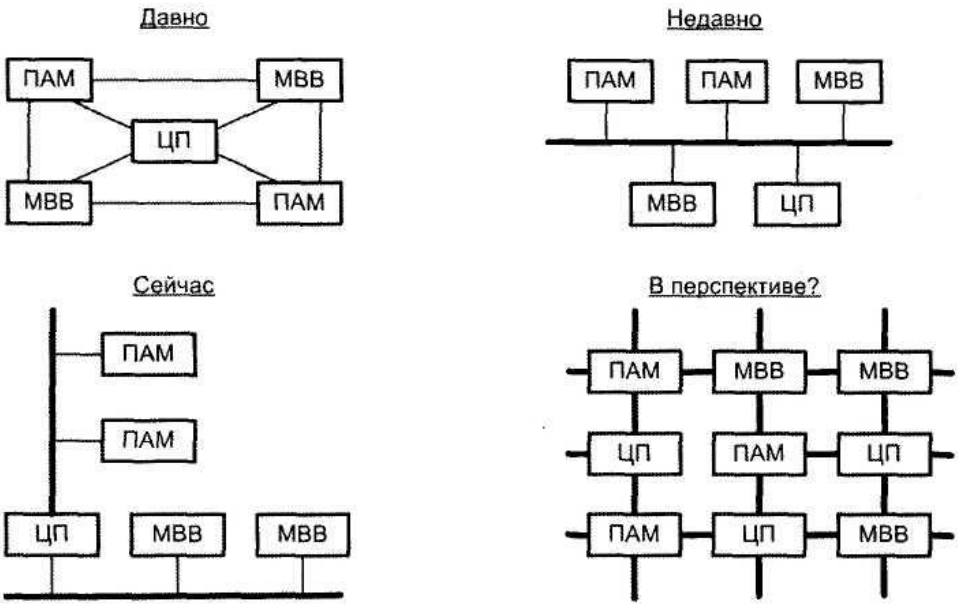


Рис. 4.2. Эволюция структур взаимосвязей (ЦП — центральный процессор, ПАМ — модуль основной памяти, МВВ — модуль ввода/вывода)

Взаимосвязь частей ВМ и ее «общение» с внешним миром обеспечиваются системой шин. Большинство машин содержат несколько различных шин, каждая из которых оптимизирована под определенный вид коммуникаций. Часть шин скрыта внутри интегральных микросхем или доступна только в пределах печатной платы. Некоторые шины имеют доступные извне точки, с тем чтобы к ним легко можно было подключить дополнительные устройства, причем большинство таких шин не просто доступны, но и отвечают определенным стандартам, что позволяет подсоединять к шине устройства различных производителей.

Чтобы охарактеризовать конкретную шину, нужно описать (рис. 4.3):

- совокупность сигнальных линий;
- физические, механические характеристики и электрические характеристики шины используемые сигналы арбитража, состояния, управления и синхронизации;
- правила взаимодействия подключенных к шине устройств (протокол шины).

Рис. 4.3. Параметры, характеризующие шину

Шину образует набор коммуникационных линий, каждая из которых способна передавать сигналы, представляющие двоичные цифры 1 и 0. По линии может пересылаться раз-

вернутая во времени последовательность таких сигналов. При совместном использовании несколько линий могут обеспечить одновременную (параллельную) передачу двоичных чисел. Физически линии шины реализуются в виде отдельных проводников, как полоски проводящего материала на монтажной плате либо как алюминиевые или медные проводящие дорожки на кристалле микросхемы.

Операции на шине называют *транзакциями.* Основные виды транзакций — *транзакции чтения* и *транзакции записи.* Если в обмене участвует устройство ввода/вывода, можно говорить о *транзакциях ввода* и *вывода,* по сути эквивалентных транзакциям чтения и записи соответственно. Шинная транзакция включает в себя две части: посылку адреса и прием (или посылку) данных.

Когда два устройства обмениваются информацией по шине, одно из них должно инициировать обмен и управлять им. Такого рода устройства называют *ведущими* (bus master). В компьютерной терминологии «ведущий» — это любое устройство, способное взять на себя владение шиной и управлять пересылкой данных. Ведущий не обязательно использует данные сам. Он, например, может захватить управление шиной в интересах другого устройства. Устройства, не обладающие возможностями инициирования транзакции, носят название *ведомых* (bus slave). В принципе к шине может быть подключено несколько потенциальных ведущих, но в любой момент времени активным может быть только один из них: если несколько устройств передают информацию одновременно, их сигналы перекрываются и искажаются. Для предотвращения одновременной активности нескольких ведущих в любой шине предусматривается процедура допуска к управлению шиной только одного из претендентов (арбитраж). В то же время некоторые шины допускают широковещательный режим записи, когда информация одного ведущего передается сразу нескольким ведомым (здесь арбитраж не требуется). Сигнал, направленный одним устройством, доступен всем остальным устройствам, подключенным к шине.

Английский эквивалент термина «шина» — «bus» — восходит к латинскому слову *omnibus,* означающему «для всего». Этим стремятся подчеркнуть, что шина ведет себя как магистраль, способная обеспечить всевозможные виды трафика.

ТИПЫ ШИН

Важным критерием, определяющим характеристики шины, может служить ее целевое назначение. По этому критерию можно выделить:

- шины «процессор-память»;
- шины ввода/вывода;
- системные шины.

ШИНА «ПРОЦЕССОР-ПАМЯТЬ»

Шина «процессор-память» обеспечивает непосредственную связь между центральным процессором (ЦП) вычислительной машины и основной памятью (ОП). В современных микропроцессорах такую шину часто называют *шиной переднего плана* и обозначают аббревиатурой FSB (Front-Side Bus). Интенсивный трафик между процессором и памятью требует, чтобы полоса пропускания шины, то есть количество информации, проходящей по шине в единицу времени, была наибольшей. Роль этой шины иногда выполняет системная шина (см. ниже), однако в плане эффективности значительно выгоднее, если обмен между ЦП и ОП ведется по отдельной шине. К рассматриваемому виду можно отнести также шину, связывающую процессор с кэш-памятью второго уровня, известную как *шина заднего плана* — BSB (Back-Side Bus). BSB позволяет вести обмен с большей скоростью, чем FSB, и полностью реализовать возможности более скоростной кэш-памяти.

Поскольку в фон-неймановских машинах именно обмен между процессором и памятью во многом определяет быстродействие ВМ, разработчики уделяют связи ЦП с памятью особое внимание. Для обеспечения максимальной пропускной способности шины «процессор-память» всегда проектируются с учетом особенностей организации системы памяти, а длина шины делается по возможности минимальной

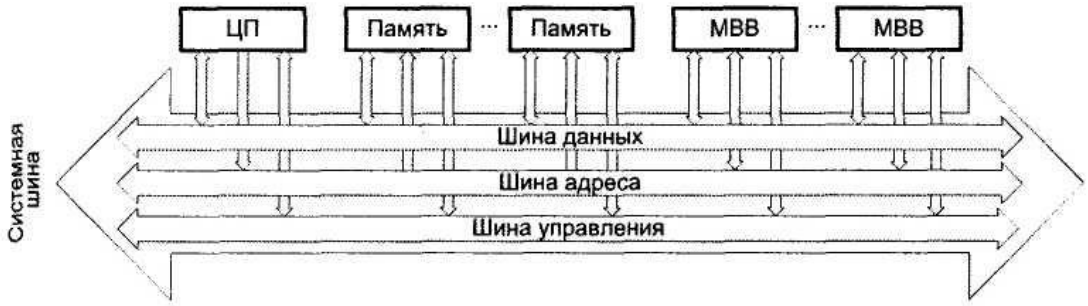
ШИНА ВВОДА/ВЫВОДА

Шина ввода/вывода служит для соединения процессора (памяти) с устройствами ввода/вывода (УВВ). Учитывая разнообразие таких устройств, шины ввода/вывода унифицируются и стандартизируются. Связи с большинством УВВ (но не с видеосистемами) не требуют от шины высокой пропускной способности. При проектировании шин ввода/вывода в учет берутся стоимость конструктивна и соединительных разъемов. Такие шины содержат меньше линий по сравнению с вариантом «процессор-память», но длина линий может быть весьма большой. Типичными примерами подобных шин могут служить шины PCI и SCSI.

СИСТЕМНАЯ ШИНА

С целью снижения стоимости некоторые ВМ имеют общую шину для памяти и устройств ввода/вывода. Такая шина часто называется системной. *Системная шина* служит для физического и логического объединения всех устройств ВМ. Поскольку основные устройства машины, как правило, размещаются на общей монтажной плате, системную шину часто называют объединительной шиной (backplane bus), хотя эти термины нельзя считать строго эквивалентными.

Системная шина в состоянии содержать несколько сотен линий. Совокупность линий шины можно подразделить на три функциональные группы (рис. 4.4): шину данных, шину адреса и шину управления. К последней обычно относят также линии для подачи питающего напряжения на подключаемые к системной шине модули.



Функционирование системной шины можно описать следующим образом. Если один из модулей хочет передать данные в другой, он должен выполнить два действия: получить в свое распоряжение шину и передать по ней данные. Если какой-то модуль хочет получить данные от другого модуля, он должен получить доступ к шине и с помощью соответствующих линий управления и адреса передать в другой модуль запрос. Далее он должен ожидать, пока модуль, получивший запрос, пошлет данные.

Физически системная шина представляет собой совокупность параллельных электрических проводников. Этими проводниками служат металлические полоски на печатной плате. Шина подводится ко всем модулям, и каждый из них подсоединяется ко всем или некоторым ее линиям. Если ВМ конструктивно выполнена на нескольких платах, то все линии шины выводятся на разъемы, которые затем объединяются проводниками на общем шасси.

Среди стандартизированных системных шин универсальных ВМ наиболее известны шины Unibus, Fastbus, Futurebus, VME, NuBus, Multibus-H. Персональные компьютеры, как правило, строятся на основе системной шины в стандартах ISA, EISA или MCA.

ИЕРАРХИЯ ШИН

Если к шине подключено большое число устройств, ее пропускная способность падает, поскольку слишком частая передача прав управления шиной от одного устройства к другому приводит к ощутимым задержкам. По этой причине во многих ВМ предпочтение отдается использованию нескольких шин, образующих определенную иерархию. Сначала рассмотрим ВМ с одной шиной.

ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С ОДНОЙ ШИНОЙ

В структурах взаимосвязей с одной шиной имеется одна системная шина, обеспечивающая обмен информацией между процессором и памятью, а также между УВВ, с одной стороны, и процессором либо памятью — с другой (рис. 4.5).

Рис. 4.5. Структура взаимосвязей с одной шиной

Для такого подхода характерны простота и низкая стоимость. Однако одношинная организация не в состоянии обеспечить высокие интенсивность и скорость транзакций, причем «узким местом» становится именно шина.

ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С ДВУМЯ ВИДАМИ ШИН

Хотя контроллеры устройств ввода/вывода (УВВ) могут быть подсоединены непосредственно к системной шине, больший эффект достигается применением одной или нескольких шин ввода/вывода (рис. 4.6). УВВ подключаются к шинам ввода/вывода, которые берут на себя основной трафик, не связанный с выходом на процессор или память. *Адаптеры шин* обеспечивают буферизацию данных при их пересылке между системной шиной и контроллерами УВВ. Это позволяет ВМ поддерживать работу множества устройств ввода/вывода и одновременно «развязать» обмен информацией по тракту процессор-память и обмен информацией с УВВ.

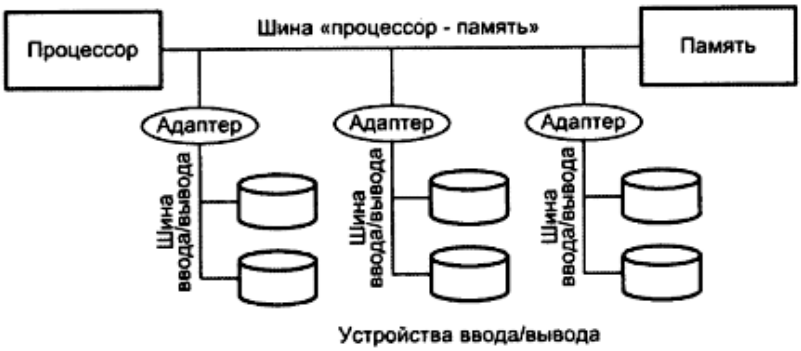


Рис. 4.6. Структура взаимосвязей с двумя видами шин

Подобная схема существенно снижает нагрузку на скоростную шину «процессор-память» и способствует повышению общей производительности ВМ. В качестве примера можно привести вычислительную машину Apple Macintosh II, где роль шины «процессор-память» играет шина NuBus. Кроме процессора и памяти к ней подключаются некоторые УВВ. Прочие устройства ввода/вывода подключаются к шине SCSI Bus.

ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С ТРЕМЯ ВИДАМИ ШИН

Для подключения быстродействующих периферийных устройств в систему шин может быть добавлена высокоскоростная шина расширения (рис. 4.7).

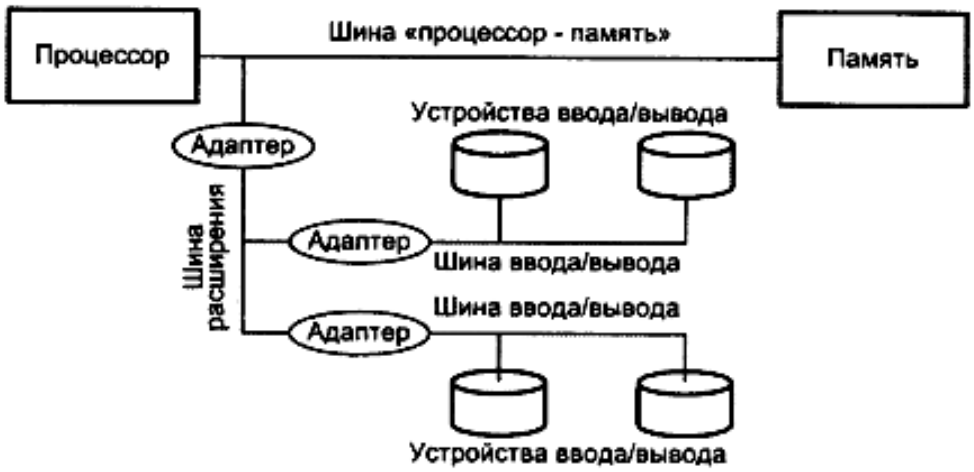


Рис. 4.7. Структура взаимосвязей с тремя видами шин

Шины ввода/вывода подключаются к шине расширения, а уже с нее через адаптер к шине «процессор-память». Схема еще более снижает нагрузку на шину «процессор-память». Такую организацию шин называют *архитектурой с «пристройкой»* (mezzanine architecture).

РАСПРЕДЕЛЕНИЕ ЛИНИЙ ШИНЫ

Любая транзакция на шине начинается с выставления ведущим устройством адресной информации. Адрес позволяет выбрать ведомое устройство и установить соединение между ним и ведущим. Для передачи адреса используется часть сигнальных линий шины, совокупность которых часто называют *шиной адреса* (ША).

На ША могут выдаваться адреса ячеек памяти, номера регистров ЦП, адреса портов ввода/вывода и т. п. Многообразие видов адресов предполагает наличие дополнительной информации, уточняющей вид, используемый в данной транзакции. Такая информация может косвенно содержаться в самом адресе, но чаще передается по специальным управляющим линиям шины.

Разнообразной может быть и структура адреса. Так, в адресе может конкретизироваться лишь определенная часть ведомого, например, старшие биты адреса могут указывать на один из модулей основной памяти, в то время как младшие биты определяют ячейку внутри этого модуля.

В некоторых шинах предусмотрены адреса специального вида, обеспечивающие одновременный выбор определенной группы ведомых либо всех ведомых сразу (broadcast). Такая возможность обычно практикуется в транзакциях записи (от ведущего к ведомым), однако существует также специальный вид транзакции чтения (одновременно от нескольких ведомых общему ведущему). Английское название такой транзакции чтения *broadcall* можно перевести как - *«широковещательный опрос».* Информация, возвращаемая ведущему, представляет собой результат побитового логического сложения данных, поступивших от всех адресуемых ведомых.

Число сигнальных линий, выделенных для передачи адреса *(ширина шины адреса),* определяет максимально возможный размер адресного пространства. Это одна из базовых характеристик шины, поскольку от нее зависит потенциальная емкость адресуемой памяти и число обслуживаемых портов ввода/вывода.

Совокупность линий, служащих для пересылки данных между модулями системы, называют *шиной данных* (ШД). Важнейшие характеристики шины данных — ширина и пропускная способность.

Ширина шины данных определяется количеством битов информации, которое может быть передано по шине за одну транзакцию *(цикл шины).* Цикл шины следует отличать от периода тактовых импульсов — одна транзакция на шине может занимать несколько такто-

вых периодов. В середине 1970-х годов типовая ширина шины данных составляла 8 бит. В наше время это обычно 32,64 или 128 бит. В любом случае ширину шины данных выбирают кратной целому числу байтов, причем это число, как правило, представляет собой целую степень числа 2.

Элемент данных, задействующий всю ширину ШД, принято называть *словом,* хотя в архитектуре некоторых ВМ понятие «слово» трактуется подругому, то есть слово может иметь разрядность, не совпадающую с шириной ШД.

В большинстве шин используются адреса, позволяющие указать отдельный байт слова. Это свойство оказывается полезным, когда желательно изменить в памяти лишь часть полного слова.

При передаче по ШД части слова пересылка обычно производится по тем же сигнальным линиям, что и в случае пересылки полного слова, однако в ряде шин «урезанное» слово передается по младшим линиям ШД. Последний вариант может оказаться более удобным при последующем расширении шины данных, поскольку в этом случае сохраняется преемственность со «старой» шиной. Ширина шины данных существенно влияет на производительность ВМ. Так, если шина данных имеет ширину вдвое меньшую чем длина команды, ЦП в течение каждого цикла команды вынужден осуществлять доступ к памяти дважды.

Пропускная способность шины характеризуется количеством единиц информации (байтов), которые допускается передать по шине за единицу времени (секунду), а определяется физическим построением шины и природой подключаемых к ней устройств. Очевидно, что чем шире шина, тем выше ее пропускная способность.

Последовательность событий, происходящих на шине данных в процессе одной транзакции, иллюстрирует рис. 4.9. Пусть устройство А на одном конце шины передает данные устройству В на другом ее конце.

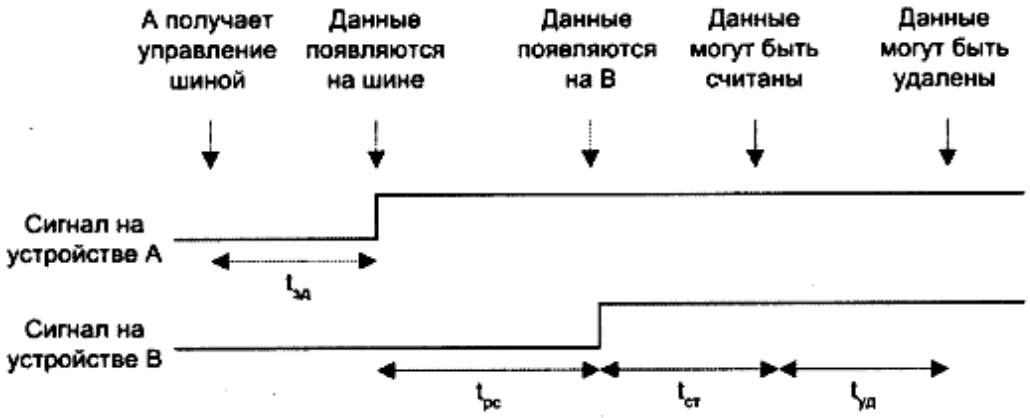


Рис. 4.9. Временная диаграмма пересылки данных

Сначала устройство А выставляет данные на шину. Здесь $t_п$ — это задержка между моментом выставления данных устройством А и моментом их появления на шине. Этот интервал времени может составлять от 1 до 4 не. Как уже отмечалось, скорость распространения данных по шине реально не в состоянии превысить 70% от скорости света. Единственный способ уменьшения задержки распространения $t_{рс}$ — сокращение длины шины. Когда сигнал достигает устройства, он должен быть «захвачен». Захват данных устройством В может быть произведен только по прошествии некоторого времени стабилизации. Время стабилизации $t_{ст}$ — это время, в течение которого данные на входе устройства В должны стабилизироваться с тем, чтобы их можно было однозначно распознать. Необходимо также упомянуть и время удержания $t_{уд}$ — интервал, в течение которого информация должна оставаться на шине данных после того, как они были зафиксированы устройством В. Общее время передачи данных по шине $t_п$ определяется выражением $t_п = t_{зд} + t_{рс}.+ t_{ст} + t_{уд}$. Если подставить

типовые значения этих параметров, получим 4 + 1,5 + 2 + 0 = 7,5 не, что соответствует частоте шины 109/7,5 = 133,3 МГц.

На практике передача данных осуществляется с задержкой на инициализацию транзакции (£„). Учитывая эту задержку, максимальную скорость передачи можно определить как

$$\frac{1}{t_{\text{п}} + t_{\text{и}}}.$$

Некоторые шины содержат дополнительные линии, используемые для обнаружения ошибок, возникших в процессе передачи. Выделение по одной дополнительной линии на каждый отдельный байт данных позволяет контролировать любой байт по паритету, причем и в случае пересылки по ШД лишь части слова. Возможен и иной вариант контроля ошибок. В этом случае упомянутые дополнительные линии используются совместно. По ним передается корректирующий код, благодаря которому ошибка может быть не только обнаружена, но и откорректирована. Такой метод удобен лишь при пересылке по шине полных слов.

Если адрес и данные в шине передаются по независимым (выделенным) сигнальным линиям, то ширина ША и ШД обычно выбирается независимо. Наиболее частые комбинации: 16-8, 16-16, 20-8, 20-16, 24-32 и 32-32. Во многих шинах адрес и данные пересылаются по одним и тем же линиям, но в разных тактах цикла шины. Этот прием называется *временным мультиплексированием* и будет рассмотрен позже. Здесь же отметим, что в случае мультиплексирования ширина ША и ширина ШД должны быть взаимоувязаны.

Применение раздельных шин адреса и данных позволяет повысить эффективность использования шины, особенно в транзакциях записи, поскольку адрес ячейки памяти и записываемые данные могут передаваться одновременно.

Помимо трактов пересылки адреса и данных, неотъемлемым атрибутом любой шины являются линии, по которым передается управляющая информации и информация о состоянии участвующих в транзакции устройств. Совокупность таких линий принято называть *шиной управления* (ШУ), хотя такое название представляется не совсем точным. Сигнальные линии, входящие в ШУ, можно условно разделить на несколько групп.

Первую группу образуют линии, по которым пересылаются *сигналы управления транзакциями,* то есть сигналы, определяющие:

- тип выполняемой транзакции (чтение или запись);
- количество байтов, передаваемых по шине данных, и, если пересылается часть слова, то какие байты;
- какой тип адреса выдан на шину адреса;
- какой протокол передачи должен быть применен.

На перечисленные цели обычно отводится от двух до восьми сигнальных линий.

Ко второй группе отнесем линии передачи *информации состояния (статуса).* В эту группу входят от одной до четырех линий, по которым ведомое устройство может информировать ведущего о своем состоянии или передать код возникшей ошибки.

Третья группа — *линии арбитража.* Вопросы арбитража рассматриваются несколько позже. Пока отметим лишь, что арбитраж необходим для выбора одного из нескольких ведущих, одновременно претендующих на доступ к шине. Число линий арбитража в разных шинах варьируется от 3 до 11.

Четвертую группу образуют *линии прерывания.* По этим линиям передаются запросы на обслуживание, посылаемые от ведомых устройств к ведущему. Под собственно запросы обычно отводятся одна или две линии, однако при одновременном возникновении запросов от нескольких ведомых возникает проблема арбитража, для чего могут понадобиться дополнительные линии, если только с этой целью не используются линии третьей группы.

Пятая группа — линии для организации *последовательных локальных сетей.* Наличие от 1 до 4 таких линий стало общепринятой практикой в современных шинах. Обусловлено это тем, что последовательная передача данных протекает значительно медленнее, чем

параллельная, и сети значительно выгоднее строить, не загружая быстрые линии основных шин адреса и данных. Кроме того, шины этой группы могут быть использованы как полноценный, хотя и медленный, избыточный тракт для замены ША и ШД в случае их отказа. Иногда шины пятой группы назначаются для реализации специальных функций, таких, например, как обработка прерываний или сортировка приоритетов задач.

В некоторых ШУ имеется **шестая группа** сигнальных линий — от 4 до 5 *линий позиционного кода,* подсоединяемых к специальным выводам разъема. С помощью перемычек на этих выводах можно задать уникальный позиционный код разъема на материнской плате или вставленной в этот разъем дочерней платы. Такой код может быть использован для индивидуальной инициализации каждой отдельной платы при включении или перезапуске системы.

Наконец, в каждой шине обязательно присутствуют линии, которые в нашей классификации входят в **седьмую группу,** которая по сути является одной из важнейших. Это группа линий *тактирования и синхронизации.* При проектировании шины таким линиям уделяется особое внимание. В состав группы, в зависимости от протокола шины (синхронный или асинхронный), входят от двух до шести линий.

В довершение необходимо упомянуть линии для подвода питающего напряжения и линии заземления.

Таблица 4.1. Распределение линий 32-разрядной шины в 64-контактном разъеме

Линии	Типовое число выводов	Комментарии
Адрес	16–32	Могут быть объединены в мультиплексируемую шину
Данные	8–32	
Арбитраж	3–11	
Управление	2–8	
Состояние	1–4	
Тактирование и синхронизация	2–6	
Локальная сеть	1–4	
Позиционный код	4–5	
Питание	2–20	
«Земля»	2–20	

Большое количество линий в шине предполагает использование разъемов со значительным числом контактов. В некоторых шинах разъемы имеют сотни контактов, где предусмотрены подключение вспомогательных шин специального назначения, свободные линии для локального обмена между дочерними платами, множественные параллельно расположенные контакты для «размножения» питания и «земли». Значительно чаще число контактов разъема ограничивают. В табл. 4.1 показано возможное распределение линий 32-разрядной шины в 64-контактном разъеме.

АРБИТРАЖ ШИН

В реальных системах на роль ведущего вправе одновременно претендовать сразу несколько из подключенных к шине устройств, однако управлять шиной в каждый момент времени может только одно из них. Чтобы исключить конфликты, шина должна предусматривать определенные механизмы арбитража запросов и правила предоставления шины одному из запросивших устройств. Решение обычно принимается на основе приоритетов претендентов.

СХЕМЫ ПРИОРИТЕТОВ

В реальных системах на роль ведущего вправе одновременно претендовать сразу несколько из подключенных к шине устройств, однако управлять шиной в каждый момент вре-

мени может только одно из них. Чтобы исключить конфликты, шина должна предусматривать определенные механизмы арбитража запросов и правила предоставления шины одному из запросивших устройств. Решение обычно принимается на основе приоритетов претендентов.

Каждому потенциальному ведущему присваивается определенный уровень приоритета, который может оставаться неизменным *(статический* или *фиксированный приоритет)* либо изменяться по какому-либо алгоритму *(динамический приоритет).*

Основной недостаток статических приоритетов в том, что устройства, имеющие высокий приоритет, в состоянии полностью блокировать доступ к шине устройств с низким уровнем приоритета. Системы с динамическими приоритетами дают шанс каждому из запросивших устройств рано или поздно получить право на управление шиной, то есть в таких системах реализуется принцип равнодоступности.

Наибольшее распространение получили следующие алгоритмы динамического изменения приоритетов:

- простая циклическая смена приоритетов;
- циклическая смена приоритетов с учетом последнего запроса;
- смена приоритетов по случайному закону;
- схема равных приоритетов;
- алгоритм наиболее давнего использования.

В алгоритме *простой циклической смены приоритетов* после каждого цикла арбитража все приоритеты понижаются на один уровень, при этом устройство, имевшее ранее низший уровень приоритета, получает наивысший приоритет.

В схеме *циклической смены приоритетов с учетом последнего запроса* все возможные запросы упорядочиваются в виде циклического списка. После обработки очередного запроса обслуженному ведущему назначается низший уровень приоритета. Следующее в списке устройство получает наивысший приоритет, а остальным устройствам приоритеты назначаются в убывающем порядке, согласно их следованию в циклическом списке.

В обеих схемах циклической смены приоритетов каждому ведущему обеспечивается шанс получить шину в свое распоряжение, однако большее распространение получил второй алгоритм.

При *смене приоритетов по случайному закону* после очередного цикла арбитража с помощью генератора псевдослучайных чисел каждому ведущему присваивается случайное значение уровня приоритета.

В *схеме равных приоритетов* при поступлении к арбитру нескольких запросов каждый из них имеет равные шансы на обслуживание. Возможный конфликт разрешается арбитром. Такая схема принята в асинхронных системах.

В *алгоритме наиболее давнего использования* (LRU, Least Recently Used) после каждого цикла арбитража наивысший приоритет присваивается ведущему, который дольше чем другие не использовал шину.

Помимо рассмотренных существует несколько алгоритмов смены приоритетов, которые не являются чисто динамическими, поскольку смена приоритетов происходит не после каждого цикла арбитража. К таким алгоритмам относятся:

- алгоритм очереди (первым пришел — первым обслужен);
- алгоритм фиксированного кванта времени.

В *алгоритме очереди* запросы обслуживаются в порядке очереди, образовавшейся к моменту начала цикла арбитража. Сначала обслуживается первый запрос в очереди, то есть запрос, поступивший раньше остальных. Аппаратурная реализация алгоритма связана с определенными сложностями, поэтому используется он редко.

В *алгоритме фиксированного кванта времени* каждому ведущему для захвата шины в течение цикла арбитража выделяется определенный квант времени. Если ведущий в этот момент не нуждается в шине, выделенный ему квант остается не использованным. Такой метод наиболее подходит для шин с синхронным протоколом.

СХЕМЫ АРБИТРАЖА

Арбитраж запросов на управление шиной может быть организован по централизованной или децентрализованной схеме. Выбор конкретной схемы зависит от требований к производительности и стоимостных ограничений.

Централизованный арбитраж

При *централизованном арбитраже* в системе имеется специальное устройство — *центральный арбитр,* — ответственное за предоставление доступа к шине только одному из запросивших ведущих. Это устройство, называемое иногда *центральным контроллером шины,* может быть самостоятельным модулем или частью ЦП. Наличие на шине только одного арбитра означает, что в централизованной схеме имеется единственная точка отказа. В зависимости от того, каким образом ведущие устройства подключены к центральному арбитру, возможные схемы централизованного арбитража можно подразделить на параллельные и последовательные.

В параллельном варианте центральный арбитр связан с каждым потенциальным ведущим индивидуальными двухпроводными трактами. Поскольку запросы к центральному арбитру могут поступать независимо и параллельно, данный вид арбитража называют *централизованным параллельным арбитражем* или *централизованным арбитражем независимых запросов.*

Идею централизованного параллельного арбитража на примере восьми ведущих устройств иллюстрирует рис. 4.11, а.

Здесь и далее под «текущим ведущим» будем понимать ведущее устройство, управляющее шиной в момент поступления нового запроса. Устройство, выставившее запрос на управление шиной, будем называть «запросившим ведущим». Сигналы запроса шины (ЗШ) поступают на вход центрального арбитра по индивидуальным линиям. Ведущему с номером *i,* который был выбран арбитром, также по индивидуальной линии возвращается сигнал предоставления шины ($ПШ_{i}$). Реально же занять шину новый ведущий сможет лишь после того, как текущий ведущий (пусть он имеет номеру) снимет сигнал занятия шины (ШЗ). Текущий ведущий должен сохранять сигналы ШЗ и 31Ц активными в течение всего времени, пока он использует шину. Получив запрос от ведущего, приоритет которого выше, чем у текущего ведущего, арбитр снимает сигнал ПШ^ на входе текущего ведущего и выдает сигнал предоставления шины ПШ, запросившему ведущему. В свою очередь, текущий ведущий, обнаружив, что центральный арбитр убрал с его входа сигнал niUj, снимает свои сигналы ШЗ и 3UJj, после чего запросивший ведущий может перенять управление шиной. Если в момент пропадания сигнала ПШ на шине происходит передача информации, текущий ведущий сначала завершает передачу и лишь после этого снимает свои сигналы.

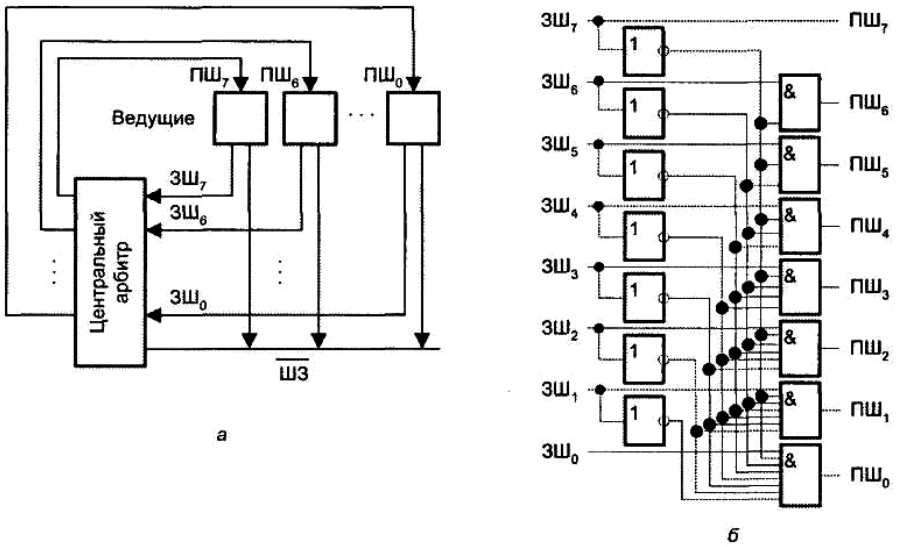


Рис. 4.11. Централизованный параллельный арбитраж: а — общая схема; б — возможная реализация

Логика выбора одного из запрашивающих ведущих обычно реализуется аппаратными средствами. В качестве примера рассмотрим реализацию системы централизованного параллельного арбитража для статических приоритетов (рис. 4.11, б). Пусть имеется восемь потенциальных ведущих 7-0, восемь сигналов запроса шины $ЗШ_7$-$ЗШ_0$ и восемь соответствующих им сигналов предоставления шины $ПШ_7$-$ПШ_0$. Положим, что приоритеты ведущих последовательно убывают с уменьшением их номера. Если текущим является ведущий 3, то шину у него могут перехватить ведущие с номерами от 4 до 7, а ведущие 0-2 этого сделать не могут. Ведущий 0 вправе использовать шину лишь тогда, когда она свободна, и должен освободить ее по запросу любого другого ведущего. Схема статических приоритетов может быть относительно просто реализована на основе логических выражений, которые применительно к рассматриваемому примеру имеют вид:

$$
\begin{aligned}
&ПШ_7 \quad ЗШ_7;\\
&ПШ_6 \quad \overline{ЗШ_7} \quad ЗШ_6;\\
&ПШ_5 \quad \overline{ЗШ_7} \quad \overline{ЗШ_6} \quad ЗШ_5;\\
&ПШ_4 \quad \overline{ЗШ_7} \quad \overline{ЗШ_6} \quad \overline{ЗШ_5} \quad ЗШ_4;\\
&ПШ_3 \quad \overline{ЗШ_7} \quad \overline{ЗШ_6} \quad \overline{ЗШ_5} \quad \overline{ЗШ_4} \quad ЗШ_3;\\
&ПШ_2 \quad \overline{ЗШ_7} \quad \overline{ЗШ_6} \quad \overline{ЗШ_5} \quad \overline{ЗШ_4} \quad \overline{ЗШ_3} \quad ЗШ_2;\\
&ПШ_1 \quad \overline{ЗШ_7} \quad \overline{ЗШ_6} \quad \overline{ЗШ_5} \quad \overline{ЗШ_4} \quad \overline{ЗШ_3} \quad \overline{ЗШ_2} \quad ЗШ_1;\\
&ПШ_0 \quad \overline{ЗШ_7} \quad \overline{ЗШ_6} \quad \overline{ЗШ_5} \quad \overline{ЗШ_4} \quad \overline{ЗШ_3} \quad \overline{ЗШ_2} \quad \overline{ЗШ_1} \quad ЗШ_0.
\end{aligned}
$$

Устройства арбитража, реализующие систему статических приоритетов, обычно выполняются в виде отдельных микросхем (например, SN74278 фирмы Texas Instruments), которые, с целью увеличения числа входов и выходов, могут объединяться по каскадной схеме, что, однако, ведет к увеличению времени арбитража.

При наличии большого числа источников запроса центральный арбитр может строиться по схеме двухуровневого параллельного арбитража. Все возможные запросы разбиваются на группы, и каждая группа анализируется своим арбитром первого уровня. Каждый арбитр первого уровня выбирает запрос, имеющий в данной группе наивысший приоритет. Арбитр второго уровня отдает предпочтение среди арбитров первого уровня, обнаруживших запросы на шину, тому, который имеет более высокий приоритет. Если количество возможных запросов очень велико, могут вводиться дополнительные уровни арбитража.

Схема централизованного параллельного арбитража очень гибка — вместо статических приоритетов допускается использовать любые варианты динамической смены приоритетов. Благодаря наличию прямых связей между центральным арбитром и ведущими схема обладает высоким быстродействием, однако именно непосредственные связи становятся причиной повышенной стоимости реализации. В параллельных схемах затруднено подключение дополнительных устройств. Обычно максимальное число ведущих при параллельном арбитраже не превышает восьми. У схемы есть еще один существенный недостаток — сигналы запроса и подтверждения присутствуют только на индивидуальных линиях и не появляются на общих линиях шины, что затрудняет диагностику.

Второй вид централизованного арбитража известен как *централизованный последовательный арбитраж.* В последовательных схемах для выделения запроса с наивысшим приоритетом используется один из сигналов, поочередно проходящий через цепочку ведущих, чем и объясняется другое название — *цепочечный* или *гирляндный арбитраж.* В дальнейшем будем полагать, что уровни приоритета ведущих устройств в цепочке понижаются слева направо.

В зависимости от того, какой из сигналов используется для целей арбитража, различа-

ют три основных типа схем цепочечного арбитража: с цепочкой для сигнала предоставления шины, с цепочкой для сигнала запроса шины (ЗШ) и с цепочкой для дополнительного сигнала разрешения (РШ). Наиболее распространена схема *цепочки для сигнала* ПШ (рис. 4.12).

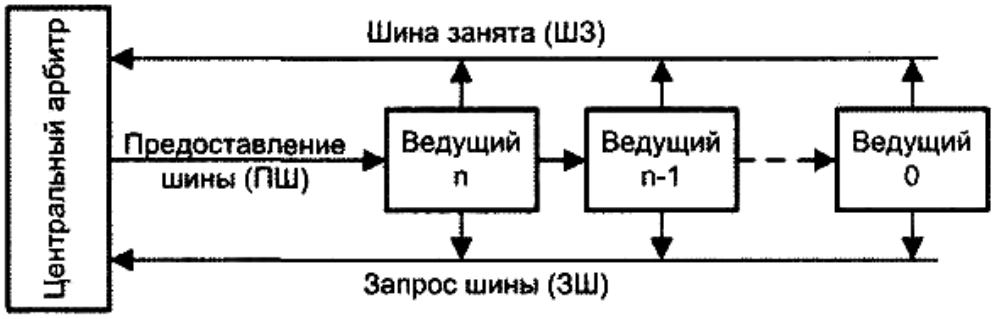


Рис. 4.12. Централизованный последовательный арбитраж с цепочкой для сигнала предоставления шины

Запросы от ведущих объединяются на линии запроса шины по схеме «монтажного ИЛИ». Аналогично организована и линия, сигнализирующая о том, что шина в данный момент занята одним из ведущих. Когда один или несколько ведущих выставляют запросы, эти запросы транслируются на вход центрального арбитра. Получив сигнал ЗШ, арбитр анализирует состояние линии занятия шины, и если шина свободна, формирует сигнал ПШ. Сигнал предоставления шины последовательно переходит по цепочке от одного ведущего к другому. Если устройство, на которое поступил сигнал ПШ, не запрашивало шину, оно просто пропускает сигнал дальше по цепочке. Когда П Ш достигнет самого левого из запросивших ведущих, последний блокирует дальнейшее распространение сигнала ПШ по цепочке и берет на себя управление шиной.

Еще раз отметим, что очередной ведущий не может приступить к управлению шиной до момента ее освобождения. Центральный арбитр не должен формировать сигнал ПШ вплоть до этого момента.

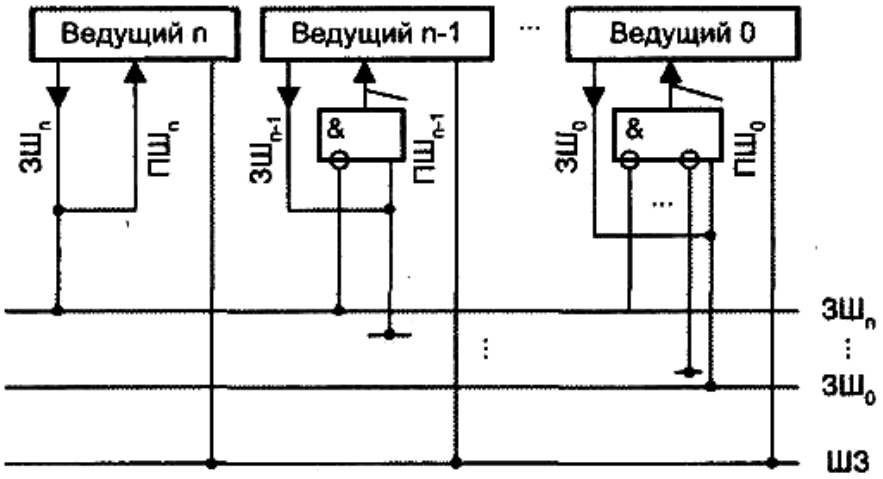
Цепочечная реализация предполагает статическое распределение приоритетов. Наивысший приоритет имеет ближайшее к арбитру ведущее устройство (устройство, на которое арбитр выдает сигнал ПШ). Далее приоритеты ведущих в цепочке последовательно понижаются.

Основное достоинство цепочечного арбитража заключается в простоте реализации и в малом количестве используемых линий. Последовательные схемы арбитража позволяют легко наращивать число устройств, подключаемых к шине.

Схеме тем не менее присущи существенные недостатки. Прежде всего, последовательное прохождение сигнала по цепочке замедляет арбитраж, причем время арбитража растет пропорционально длине цепочки. Статическое распределение приоритетов может привести к полному блокированию устройств с низким уровнем приоритета (расположенных в конце цепочки). Наконец, как и параллельный вариант, централизованный последовательный арбитраж не очень удобен в плане диагностики работы шины.

Децентрализованный арбитраж

При *децентрализованном* или *распределенном арбитраже* единый арбитр отсутствует. Вместо этого каждый ведущий содержит блок управления доступом к шине, и при совместном использовании шины такие блоки взаимодействуют друг с другом, разделяя между собой ответственность за доступ к шине. По сравнению с централизованной схемой децентрализованный арбитраж менее чувствителен к отказам претендующих на шину устройств.



Одна из возможных схем, которую можно условно назвать схемой децентрализованного параллельного арбитража, показана на рис. 4.13. Каждый ведущий имеет уникальный уровень приоритета и обладает собственным контроллером шины, способным формировать сигналы предоставления и занятия шины. Сигналы запроса от любого ведущего поступают на входы всех остальных ведущих. Логика арбитража реализуется в контроллере шины каждого ведущего. Под децентрализованный арбитраж может быть модифицирована также схема, приведенная на рис. 4.12. Подобный вариант, называемый кольцевой схемой, показан на рис. 4.14.

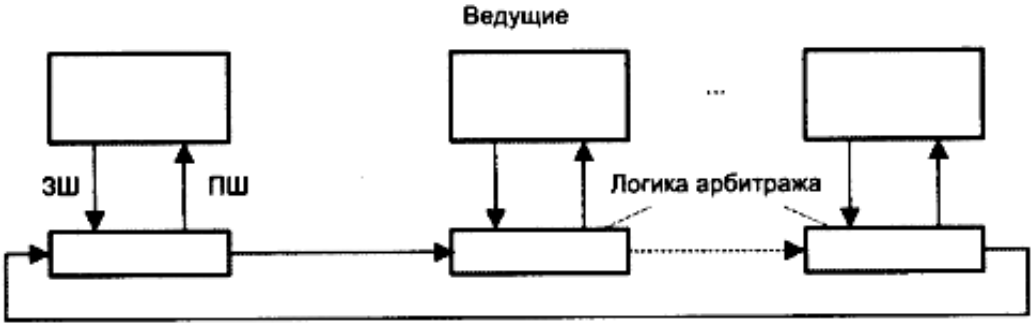


Рис. 4.14. Кольцевая схема

Здесь сигнал может возникать в различных точках цепочки, замкнутой в кольцо. Переход к новому ведущему сопровождается циклической сменой приоритетов. В следующем цикле арбитража текущий ведущий будет иметь самый низкий уровень приоритета. Соседний ведущий справа получает наивысший приоритет, а далее каждому устройству в кольце присваивается уровень приоритета на единицу меньше, чем у соседа слева. Иными словами, реализуется циклическая смена приоритетов с учетом последнего запроса.

Текущий ведущий, управляющий шиной, генерирует сигнал ПШ, который проходит через все ведущие устройства, не запросившие шину. Ведущий, сформировавший запрос и имеющий на входе активный сигнал ПШ, запрещает прохождение этого сигнала далее по цепочке, но не может взять на себя управление шиной до момента ее освобождения текущим ведущим. Когда текущий ведущий обнаруживает, что «потерял» сигнал ПШ на своем входе, он обязан при первой возможности освободить шину и снять сигнал занятия шины.

Для большинства шин все-таки более характерна другая организация децентрализованного арбитража. Такие схемы предполагают наличие в составе шины группы арбитражных линий, организованных по схеме «монтажного ИЛИ». Это позволяет любому ведущему видеть сигналы, выставленные остальными устройствами. Каждому ведущему присваивается уникальный номер, совпадающий с кодом уровня приоритета данного ведущего. Запрашивающие шину устройства выдают на арбитражные линии свой номер. Каждый из запросив-

ших ведущих, обнаружив на арбитражных линиях номер устройства с более высоким приоритетом, снимает с этих линий младшие биты своего номера. В конце концов на арбитражных линиях остается только номер устройства, обладающего наиболее высоким приоритетом. Победителем в процедуре арбитража становится ведущий, опознавший на арбитражных линиях свой номер. Подобная схема известна также как *распределенный арбитраж с самостоятельным выбором,* поскольку ведущий сам определяет, стал ли он победителем в арбитраже, то есть выбирает себя самостоятельно.

Идея подобного арбитража была предложена М. Таубом (Matthew Taub) в 1975 году. В алгоритме Тауба под арбитраж выделяются две группы сигнальных линий, доступные всем устройствам на шине. Устройства подключаются к этим линиям по схеме «монтажного ИЛИ». Первая группа служит для передачи сигналов синхронизации и управления. Вторую группу линий условно назовем шиной приоритета и обозначим В. В зависимости от принятого числа уровней приоритета эта группа может содержать от 4 до 7 линий. Каждому потенциальному ведущему назначается уникальный уровень приоритета. Приоритет Р представлен ^-разрядным двоичным кодом. Каждому разряду кода приоритета соответствует линия в шине В. Ведущие, претендующие на управление шиной, выдают на шину В свои коды приоритета Р. Дальнейшее поведение ведущих определяется следующим правилом: если *i-ii* разряд кода приоритета равен О ($P_i = 0$), а на *i*-й линии шины В в данный момент присутствует единица ($B_i = 1$), то ведущий обнуляет в выставляемом коде все младшие разряды, от 0-го до $_i$ - го. В результате такой процедуры на шине В остается код наивысшего из выставленных приоритетов. Устройство, распознавшее на шине свой код приоритета, считается выигравшим арбитраж. После завершения своей транзакции выигравшее устройство снимает с шины В свой код приоритета, при этом ситуация на линиях В меняется. Ведущие, претендовавшие на шину, восстанавливают ранее обнуленные разряды, и начинается новый цикл арбитража.

В целом схемы децентрализованного арбитража потенциально более надежны, поскольку отказ контроллера шины в одном из ведущих не нарушает работу с шиной на общем уровне. Тем не менее должны быть предусмотрены средства для обнаружения неисправных контроллеров, например на основе тайм-аута. Основной недостаток децентрализованных схем — в относительной сложности логики арбитража, которая должна быть реализована в аппаратуре каждого ведущего.

В некоторых ВМ применяют комбинированные последовательно-параллельные схемы арбитража, в какой-то мере сочетающие достоинства обоих методов. Здесь все ведущие разбиваются на группы. Арбитраж внутри группы ведется по последовательной схеме, а между группами — по параллельной.

Ограничение времени управления шиной

Вне зависимости от принятой модели арбитража должна быть также продумана стратегия ограничения времени контроля над шиной. Одним из вариантов может быть разрешение ведущему занимать шину в течение одного цикла шины, с предоставлением ему возможности конкуренции за шину в последующих циклах. Другим вариантом является принудительный захват контроля над шиной устройством с более высоким уровнем приоритета, при сохранении восприимчивости текущего ведущего к запросам на освобождение шины от устройств с меньшим уровнем приоритета.

Опросные схемы арбитража

В опросных методах запросы только фиксируются, и контроллер шины способен узнать о них, лишь опросив ведущих. Опрос может быть как централизованным — с одним контроллером, производящим опрос, так и децентрализованным — с несколькими контроллерами шины.

Данный механизм использует специальные линии опроса между контроллером (контроллерами) шины и ведущими — по одной линии для каждого ведущего. С целью уменьшения числа таких линий может формироваться номер запрашивающего ведущего, для чего вместо 2" достаточно *n* линий. Кроме того, используются также линии запроса шины и линия сигнала занятия шины.

Централизованный опрос

Централизованный опрос иллюстрирует рис. 4.15.

Контроллер шины последовательно опрашивает каждое ведущее устройство на предмет, находится ли оно в ожидании предоставления шины. Для этого контроллер выставляет на линии опроса адрес соответствующего ведущего. Если в момент выставления адреса ведущий ожидает разрешения на управление шиной, то он, распознав свой адрес, сигнализирует об этом, делая активной шину (ЗШ). Обнаружив сигнал, контроллер разрешает ведущему использовать шину. Последовательность опроса ведущих может быть организована в порядке убывания адресов, либо меняться в соответствии с алгоритмом динамического приоритета.

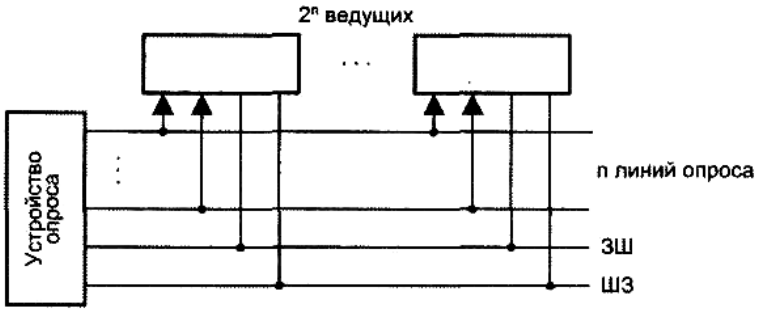


Рис. 4.15. Организация централизованного опроса ведущих

Децентрализованный опрос

Организация децентрализованного опроса показана на рис. 4.16.

Каждый ведущий содержит контроллер шины, состоящий из дешифратора адреса и генератора адреса. В начале опросной последовательности формируется адрес, который распознается контроллером. Если соответствующий ведущий ожидает доступа к шине, он вправе теперь ее занять. По завершении работы с шиной контроллер текущего ведущего генерирует адрес следующего ведущего, и процесс повторяется. При такой схеме обычно требуется применять систему с квитированием, использующую сигнал ЗШ, формируемый генератором адреса, и сигнал ПШ, генерируемый дешифратором адреса.

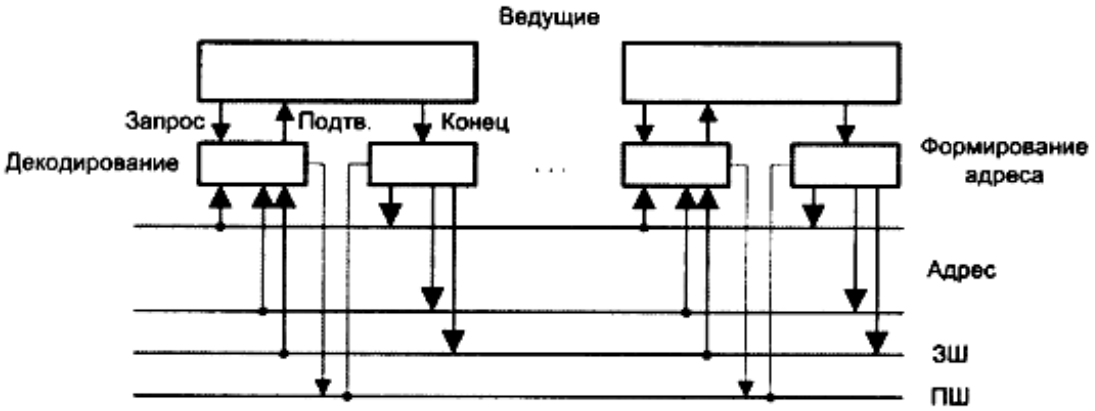


Рис. 4.16. Организация децентрализованного опроса ведущих

При децентрализованном опросе отказ в одной из точек приводит к отказу всей системы арбитража. Такая ситуация, впрочем, может быть предотвращена с помощью механизма тайм-аута: по истечении заданного времени функции отказавшего контроллера берет на себя следующий контроллер.

НАДЕЖНОСТЬ И ОТКАЗОУСТОЙЧИВОСТЬ

Надежность и отказоустойчивость — важнейшие аспекты проектирования шин. Основные надежды здесь обычно возлагают на корректирующие коды, которые позволяют обнаружить отказ одиночного элемента или шумовой выброс и автоматически парировать ошибку. Подобный подход, широко практикуемый в системах памяти, применительно к шинам порождает специфические проблемы.

В шинах отдельные функциональные группы сигналов (сигналы адреса, данных, управления, состояния и арбитража) предполагают независимые контроль и коррекцию. При наличии множества небольших групп сигналов метод корректирующих кодов становится малоэффективным из-за необходимости включения в шину большого числа контрольных линий. Кроме того, в шинах весьма вероятно одновременное возникновение ошибок сразу в нескольких сигналах. Для учета такой ситуации необходимо увеличивать разрядность корректирующего кода, то есть вводить в шину дополнительные сигнальные линии. Достаточно неясным остается вопрос защиты одиночных сигналов, в частности сигналов тактирования и синхронизации.

Вычисление корректирующих кодов и коррекция ошибок требуют дополнительного времени, что замедляет шину.

Усложнение аппаратуры, обусловленное использованием корректирующих кодов, ведет к снижению общей надежности шины, в результате чего суммарный выигрыш может оказаться меньше ожидаемого. В силу приведенных соображений становится ясным, почему проектировщики постоянно ищут альтернативные способы обеспечения надежности и отказоустойчивости шин.

Достаточно хорошие результаты дают так называемые «высокоуровневые» подходы. Здесь вместо отслеживания каждого цикла шины производятся контроль и коррекция более крупных единиц, например целых блоков данных или законченной программной операции.

При наличии в системе избыточных процессоров и шин возможен перекрестный контроль, причем программное обеспечение может производить изменения в конфигурации системы и предупреждать оператора о необходимости замены определенных блоков. Даже если шина имеет встроенные средства коррекции ошибок, желательно дополнять их некоторым дополнительным уровнем «разумности» для предотвращения такой постепенной деградации системы, компенсировать которую имеющийся механизм коррекции будет уже не в состоянии.

При разработке аппаратуры необходимо обязательно учитывать определенные требования, связанные с обеспечением отказоустойчивости. Так, если обнаружена ошибка, то для ее коррекции должна быть предусмотрена возможность повторной передачи данных. Это предполагает, что оригинальная передача не должна приводить к необратимым побочным эффектам. Например, если операция чтения с периферийного устройства вызывает стирание исходных данных или сбрасывает флаги состояния, успешное повторное чтение становится невозможным. Другой пример: работа с буферной памятью типа FIFO (First In First Out), работающей по принципу «первым прибыл, первым обслужен», где ошибочные данные внутри очереди недоступны и поэтому не могут быть откорректированы.

Чтобы учесть подобные ситуации, при разработке адресуемой памяти необходимо предусмотреть буферы, а очистка ячеек и сброс флагов должны быть не побочными эффектами, а выполняться только явно с помощью определенных команд. Память типа FIFO может быть снабжена адресуемыми буферами, предназначенными для хранения данных вплоть до завершения передачи.

ВОПРОСЫ САМОКОНТРОЛЯ

1. ТИПЫ ШИН
2. ШИНА «ПРОЦЕССОР-ПАМЯТЬ»
3. ШИНА ВВОДА/ВЫВОДА
4. СИСТЕМНАЯ ШИНА
5. ИЕРАРХИЯ ШИН
6. ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С ОДНОЙ ШИНОЙ
7. ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С ДВУМЯ ВИДАМИ ШИН
8. ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С ТРЕМЯ ВИДАМИ ШИН
9. РАСПРЕДЕЛЕНИЕ ЛИНИЙ ШИНЫ
10. АРБИТРАЖ ШИН
11. СХЕМЫ ПРИОРИТЕТОВ
12. СХЕМЫ АРБИТРАЖА
13. НАДЕЖНОСТЬ И ОТКАЗОУСТОЙЧИВОСТЬ

ЛЕКЦИЯ 4. ПАМЯТЬ

В любой ВМ, вне зависимости от ее архитектуры, программы и данные хранятся в памяти. Функции памяти обеспечиваются запоминающими устройствами (ЗУ), предназначенными для фиксации, хранения и выдачи информации в процессе работы ВМ. Процесс фиксации информации в ЗУ называется *записью,* процесс выдачи информации — *чтением* или *считыванием,* а совместно их определяют как *процессы обращения* к ЗУ.

ХАРАКТЕРИСТИКИ СИСТЕМ ПАМЯТИ

Перечень основных характеристик, которые необходимо учитывать, рассматривая конкретный вид ЗУ, включает в себя:

- место расположения;
- емкость;
- единицу пересылки;
- метод доступа;
- быстродействие;
- физический тип;
- физические особенности;
- стоимость.

По *месту расположения* ЗУ разделяют на процессорные, внутренние и внешние. Наиболее скоростные виды памяти (регистры, кэш-память первого уровня) обычно размещают на общем кристалле с центральным процессором, а регистры общего назначения вообще считаются частью ЦП. Вторую группу (внутреннюю память) образуют ЗУ, расположенные на системной плате. К внутренней памяти относят основную память, а также кэш-память второго и последующих уровней (кэш-память второго уровня может также размещаться на кристалле процессора). Медленные ЗУ большой емкости (магнитные и оптические диски, магнитные ленты) называют внешней памятью, поскольку к ядру ВМ они подключаются аналогично устройствам ввода/вывода.

Емкость ЗУ характеризуют числом битов либо байтов, которое может храниться в запоминающем устройстве. На практике применяются более крупные единицы, а для их обозначения к словам «бит» или «байт» добавляют приставки: кило, мега, гига, тера, пета, экза (kilo, mega, giga, tera, peta, exa). Стандартно эти приставки означают умножение основной единицы измерений на 10^3, 10^G, 10^9, 10^{12}, 10^{15}и 10^1 соответственно. В вычислительной технике, ориентированной на двоичную систему счисления, они соответствуют значениям достаточно близким к стандартным, но представляющим собой целую степень числа 2, то есть 2^{10}, 2^{20}, 2^{30}, 2^{40}, 2^{50}, 2^{60}. Во избежание разночтений, в последнее время ведущие международные организации по стандартизации, например IEEE (Institute of Electrical and Electronics Engineers), предлагают ввести новые обозначения, добавив к основной приставке слово binary (бинарный): kilobinary, megabinary, gigabinary, terabinary, petabinary, exabinary. В результате вместо термина «килобайт» предлагается термин «киби-байт», вместо «мегабайт» — «мебибайт» и т. д. Для обозначения новых единиц предлагаются сокращения: Ki, Mi, Gi, Ti, Pi и Ei.

Важной характеристикой ЗУ является *единица пересылки.* Для основной памяти (ОП) единица пересылки определяется шириной шины данных, то есть количеством битов, передаваемых по линиям шины параллельно. Обычно единица пересылки равна длине слова, но не обязательно. Применительно к внешней памяти данные часто передаются единицами, превышающими размер слова, и такие единицы называются *блоками.*

При оценке быстродействия необходимо учитывать применяемый в данном типе *ЗУ метод доступа* к данным. Различают четыре основных метода доступа:

Последовательный доступ. ЗУ с последовательным доступом ориентировано на хранение информации в виде последовательности блоков данных, называемых записями. Для доступа к нужному элементу (слову или байту) необходимо прочитать все предшествующие ему данные. Время доступа зависит от положения требуемой записи в последовательности записей на носителе информации и позиции элемента внутри данной записи. Примером мо-

жет служить ЗУ на магнитной ленте.

Прямой доступ. Каждая запись имеет уникальный адрес, отражающий ее физическое размещение на носителе информации. Обращение осуществляется как адресный доступ к началу записи, с последующим последовательным доступом к определенной единице информации внутри записи. В результате время доступа к определенной позиции является величиной переменной. Такой режим характерен для магнитных дисков.

Произвольный доступ. Каждая ячейка памяти имеет уникальный физический адрес. Обращение к любой ячейке занимает одно и то же время и может производиться в произвольной очередности. Примером могут служить запоминающие устройства основной памяти.

Ассоциативный доступ. Этот вид доступа позволяет выполнять поиск ячеек, содержащих такую информацию, в которой значение отдельных битов совпадает с состоянием одноименных битов в заданном образце. Сравнение осуществляется параллельно для всех ячеек памяти, независимо от ее емкости. По ассоциативному принципу построены некоторые блоки кэш-памяти.

Быстродействие ЗУ является одним из важнейших его показателей. Для количественной оценки быстродействия обычно используют три параметра:

Время доступа ($T_Д$). Для памяти с произвольным доступом оно соответствует интервалу времени от момента поступления адреса до момента, когда данные заносятся в память или становятся доступными. В ЗУ с подвижным носителем информации — это время, затрачиваемое на установку головки записи/считывания (или носителя) в нужную позицию.

Длительность цикла памяти или *период обращения* ($Г_ц$). Понятие применяется к памяти с произвольным доступом, для которой оно означает минимальное время между двумя последовательными обращениями к памяти. Период обращения включает в себя время доступа плюс некоторое дополнительное время. Дополнительное время может требоваться для затухания сигналов на линиях, а в некоторых типах ЗУ, где считывание информации приводит к ее разрушению, — для восстановления считанной информации.

Скорость передачи. Это скорость, с которой данные могут передаваться в память или из нее. Для памяти с произвольным доступом она равна $1/Г_ц$. Для других видов памяти скорость передачи определяется соотношением:

$$T_N = T_A + \frac{N}{R},$$

где T_N — среднее время считывания или записи N битов; T_A — среднее время доступа; R — скорость пересылки в битах в секунду.

Говоря о *физическом типе* запоминающего устройства, необходимо упомянуть три наиболее распространенных технологии ЗУ — это полупроводниковая память, память с магнитным носителем информации, используемая в магнитных дисках и лентах, и память с оптическим носителем — оптические диски.

В зависимости от примененной технологии следует учитывать и ряд *физических особенностей* ЗУ, например энергозависимость. В энергозависимой памяти информация может быть искажена или потеряна при отключении источника питания. В энергонезависимых ЗУ записанная информация сохраняется и при отключении питающего напряжения. Магнитная и оптическая память — энергонезависимы. Полупроводниковая память может быть как энергозависимой, так и нет, в зависимости от ее типа. Помимо энергозависимости нужно учитывать, приводит ли считывание информации к ее разрушению.

ОСНОВНАЯ ПАМЯТЬ

Основная память (ОП) представляет собой единственный вид памяти, к которой ЦП может обращаться непосредственно (исключение составляют лишь регистры центрального процессора). Информация, хранящаяся на внешних ЗУ, становится доступной процессору только после того, как будет переписана в основную память. Основную память образуют запоминающие устройства с произвольным доступом. Такие ЗУ образованы как массив ячеек,

а «произвольный доступ» означает, что обращение к любой ячейке занимает одно и то же время и может производиться в произвольной последовательности. Каждая ячейка содержит фиксированное число запоминающих элементов и имеет уникальный адрес, позволяющий различать ячейки при обращении к ним для выполнения операций записи и считывания. Основная память может включать в себя два типа устройств: *оперативные запоминающие устройства* (ОЗУ) и *постоянные запоминающие устройства* (ПЗУ).

Преимущественную долю основной памяти образует ОЗУ, называемое оперативным, потому что оно допускает как запись, так и считывание информации, причем обе операции выполняются однотипно, практически с одной и той же скоростью, и производятся с помощью электрических сигналов. В англоязычной литературе ОЗУ соответствует аббревиатура RAM — *Random Access Memory,* то есть «память с произвольным доступом», что не совсем корректно, поскольку памятью с произвольным доступом являются также ПЗУ и регистры процессора. Для большинства типов полупроводниковых ОЗУ характерна энергозависимость — даже при кратковременном прерывании питания хранимая информация теряется. Микросхема ОЗУ должна быть постоянно подключена к источнику питания и поэтому может использоваться только как временная память.

Вторую группу полупроводниковых ЗУ основной памяти образуют энергонезависимые микросхемы ПЗУ (ROM — *Read-Only Memory).* ПЗУ обеспечивает считывание информации, но не допускает ее изменения (в ряде случаев информация в ПЗУ может быть изменена, но этот процесс сильно отличается от считывания и требует значительно большего времени).

ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Большинство из применяемых в настоящее время типов микросхем оперативной памяти не в состоянии сохранять данные без внешнего источника энергии, то есть являются энергозависимыми (volatile memory). Широкое распространение таких устройств связано с рядом их достоинств по сравнению с энергонезависимыми типами ОЗУ (non-volatile memory): большей емкостью, низким энергопотреблением, более высоким быстродействием и невысокой себестоимостью хранения единицы информации.

Энергозависимые ОЗУ можно подразделить на две основные подгруппы: динамическую память (DRAM — Dynamic Random Access Memory) и статическую память (SRAM — Static Random Access Memory).

Статическая и динамическая оперативная память

В *статических ОЗУ* запоминающий элемент может хранить записанную информацию неограниченно долго (при наличии питающего напряжения). Запоминающий элемент *динамического ОЗУ* способен хранить информацию только в течение достаточно короткого промежутка времени, после которого информацию нужно восстанавливать заново, иначе она будет потеряна. Динамические ЗУ, как и статические, энергозависимы.

Статические оперативные запоминающие устройства

Напомним, что роль запоминающего элемента в статическом ОЗУ исполняет триггер. Статические ОЗУ на настоящий момент — наиболее быстрый, правда, и наиболее дорогостоящий вид оперативной памяти. Известно достаточно много различных вариантов реализации SRAM, отличающихся по технологии, способам организации и сфере применения (рис. 5.9). *Асинхронные статические ОЗУ.* Асинхронные статические ОЗУ применялись в кэш-памяти второго уровня в течение многих лет, еще с момента появления микропроцессора i80386. Для таких ИМС время доступа составляло 15-20 не (в лучшем случае — 12 не), что не позволяло кэш-памяти второго уровня работать в темпе процессора.

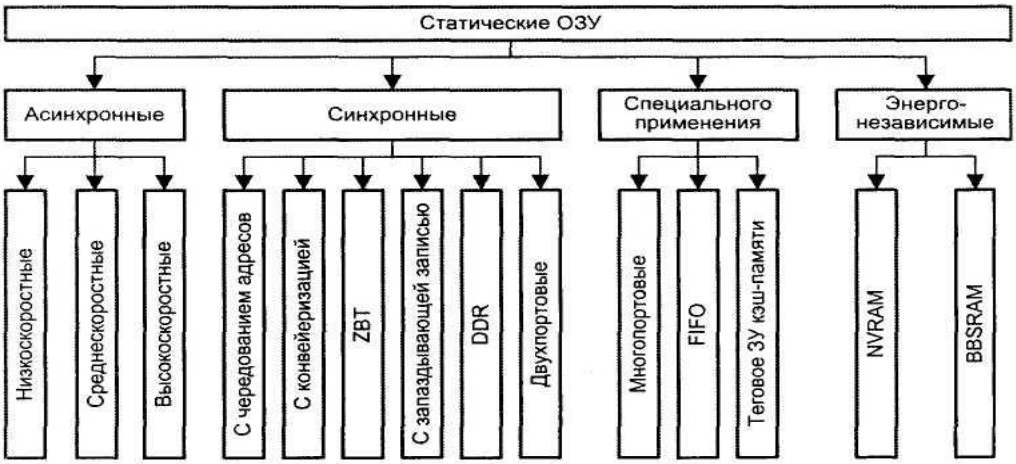


Рис. 5.9. Виды статических ОЗУ

Синхронные статические ОЗУ. В рамках данной группы статических ОЗУ выделяют ИМС типа SSRAM и более совершенные PB SRAM.

Последние модификации микропроцессоров Pentium, начиная с Pentium II, взамен SSRAM оснащаются статической оперативной памятью с пакетным конвейерным доступом (PB SRAM — Pipelined Burst SRAM). В этой разновидности SRAM реализована внутренняя конвейеризация, за счет которой скорость обмена пакетами данных возрастает примерно вдвое. Память данного типа хорошо работает при повышенных частотах системной шины. Время доступа к PB SRAM составляет от 4,5 до 8 нс, при этом формула 3-1-1-1 сохраняется даже при частоте системной шины 133 МГц.

Особенности записи в статических ОЗУ. Важным моментом, характеризующим SRAM, является технология записи. Известны два варианта записи: *стандартная* и *запаздывающая.* В стандартном режиме адрес и данные выставляются на соответствующие шины в одном и том же такте. В режиме запаздывающей записи данные для нее передаются в следующем такте после выбора адреса нужной ячейки, что напоминает режим конвейерного чтения, когда данные появляются на шине в следующем такте. Оба рассматриваемых варианта позволяют производить запись данных с частотой системной шины. Различия сказываются только при переключении между операциями чтения и записи.

Динамические оперативные запоминающие устройства

Динамической памяти в вычислительной машине значительно больше, чем статической, поскольку именно DRAM используется в качестве основной памяти ВМ. Как и SRAM, динамическая память состоит из ядра (массива ЗЭ) и интерфейсной логики (буферных регистров, усилителей чтения данных, схемы регенерации и др.).

В отличие от SRAM адрес ячейки DRAM передается в микросхему за два шага — вначале адрес столбца, а затем строки, что позволяет сократить количество выводов шины адреса примерно вдвое, уменьшить размеры корпуса и разместить на материнской плате большее количество микросхем. Это, разумеется, приводит к снижению быстродействия, так как для передачи адреса нужно вдвое больше времени. Для указания, какая именно часть адреса передается в определенный момент, служат два вспомогательных сигнала RAS и CAS. При обращении к ячейке памяти на шину адреса выставляется адрес строки. После стабилизации процессов на шине подается сигнал RAS и адрес записывается во внутренний регистр микросхемы памяти. Затем на шину адреса выставляется адрес столбца и выдается сигнал CAS. В зависимости от состояния линии WE производится чтение данных из ячейки или их запись в ячейку (перед записью данные должны быть помещены на шину данных). Интервал между установкой адреса и выдачей сигнала RAS (или CAS) оговаривается техническими характеристиками микросхемы, но обычно адрес выставляется в одном такте системной шины, а управляющий сигнал — в следующем. Таким образом, для чтения или записи одной

ячейки динамического ОЗУ требуется пять тактов, в которых происходит соответственно: выдача адреса строки, выдача сигнала RAS, выдача адреса столбца, выдача сигнала CAS, выполнение операции чтения/записи (в статической памяти процедура занимает лишь от двух до трех тактов).

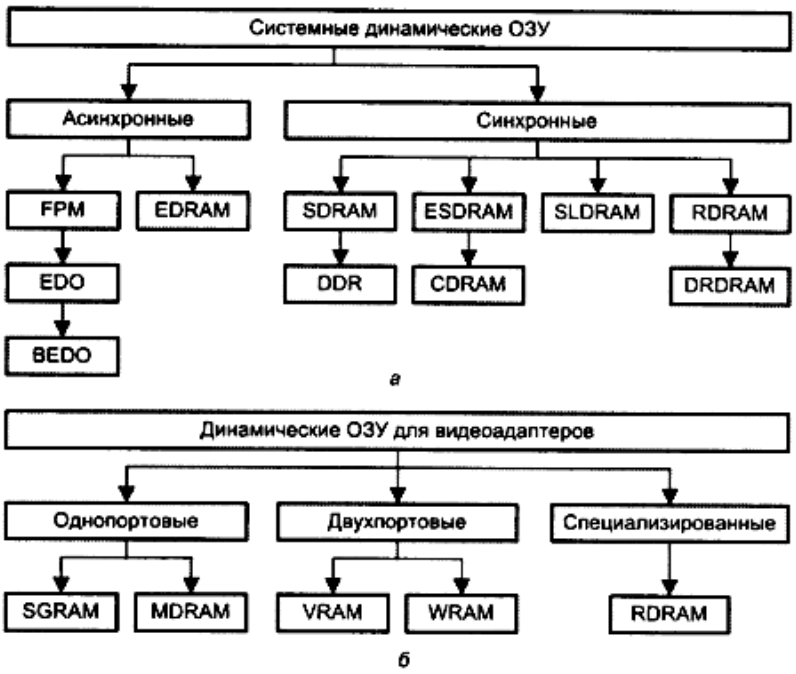


Рис. 5.10. Классификация динамических ОЗУ: а — микросхемы для основной памяти; *б* — микросхемы для видеоадаптеров

Следует также помнить о необходимости регенерации данных. Но наряду с естественным разрядом конденсатора ЗЭ со временем к потере заряда приводит также считывание данных из DRAM, поэтому после каждой операции чтения данные должны быть восстановлены. Это достигается за счет повторной записи тех же данных сразу после чтения. При считывании информации из одной ячейки фактически выдаются данные сразу всей выбранной строки, но используются только те, которые находятся в интересующем столбце, а все остальные игнорируются. Таким образом, операция чтения из одной ячейки приводит к разрушению данных всей строки, и их нужно восстанавливать. Регенерация данных после чтения выполняется автоматически интерфейсной логикой микросхемы, и происходит это сразу же после считывания строки. Теперь рассмотрим различные типы микросхем динамической памяти, начнем с системных DRAM, то есть микросхем, предназначенных для использования в качестве основной памяти. На начальном этапе это были микросхемы асинхронной памяти, работа которых не привязана жестко к тактовым импульсам системной шины.

Асинхронные динамические ОЗУ. Микросхемы асинхронных динамических ОЗУ управляются сигналами RAS и CAS, и их работа в принципе не связана непосредственно тактовыми импульсами шины. Асинхронной памяти свойственны дополнительные затраты времени на взаимодействие микросхем памяти и контроллера. Так, в асинхронной схеме сигнал RAS будет сформирован только после поступления в контроллер тактирующего импульса и будет воспринят микросхемой памяти через некоторое время. После этого память выдаст данные, но контроллер сможет их считать только по приходу следующего тактирующего импульса, так как он должен работать синхронно с остальными устройствами ВМ. Таким образом, на протяжении цикла чтения/записи происходят небольшие задержки из-за ожидания памятью контроллера и контроллером памяти.

Микросхемы DRAM. В первых микросхемах динамической памяти применялся наиболее простой способ обмена данными, часто называемый традиционным (conventional). Он

позволял считывать и записывать строку памяти только на каждый пятый такт (рис. 5.11, *а).* Этапы такой процедуры были описаны ранее. Традиционной DRAM соответствует формула 5-5-5-5. Микросхемы данного типа могли работать на частотах до 40 МГц и из-за своей медлительности (время доступа составляло около 120 не) просуществовали недолго.

Микросхемы FPMDRAM. Микросхемы динамического ОЗУ, реализующие режим FPM, также относятся к ранним типам DRAM. Сущность режима была показана ранее. Схема чтения для FPM DRAM (рис. 5.11, *6)* описывается формулой 5-3-3-3 (всего 14 тактов). Применение схемы быстрого страничного доступа позволило сократить время доступа до 60 не, что, с учетом возможности работать на более высоких частотах шины, привело к увеличению производительности памяти по сравнению с традиционной .DRAM приблизительно на 70%. Данный тип микросхем применялся в персональных компьютерах примерно до 1994 года.

Микросхемы EDO DRAM. Следующим этапом в развитии динамических ОЗУ стали ИМС с *гиперстраничным режимом доступа* (HPM, Hyper Page Mode), более известные как EDO (Extended Data Output — расширенное время удержания данных на выходе). Главная особенность технологии — увеличенное по сравнению с FPM DRAM время доступности данных на выходе микросхемы. В микросхемах FPM DRAM выходные данные остаются действительными только при активном сигнале CAS, из-за чего во втором и последующих доступах к строке нужно три такта: такт переключения CAS в активное состояние, такт считывания данных и такт переключения CAS в неактивное состояние. В EDO DRAM по активному (спадающему) фронту сигнала CAS данные запоминаются во внутреннем регистре, где хранятся еще некоторое время после того, как поступит следующий активный фронт сигнала. Это позволяет использовать хранимые данные, когда CAS уже переведен в неактивное состояние (рис. 5.11, б). Иными словами, временные параметры улучшаются за счет исключения циклов ожидания момента стабилизации данных на выходе микросхемы.

Схема чтения у EDO DRAM уже 5-2-2-2, что на 20% быстрее, чем у FPM. Время доступа составляет порядка 30-40 не. Следует отметить, что максимальная частота системной шины для микросхем EDO DRAM не должна была превышать 66 МГц.

Микросхемы BEDO DRAM. Технология EDO была усовершенствована компанией VIA Technologies. Новая модификация EDO известна как BEDO (Burst EDO — пакетная EDO). Новизна метода в том, что при первом обращении считывается вся строка микросхемы, в которую входят последовательные слова пакета. За последовательной пересылкой слов (переключением столбцов) автоматически следит внутренний счетчик микросхемы. Это исключает необходимость выдавать адреса для всех ячеек пакета, но требует поддержки со стороны внешней логики. Способ позволяет сократить время считывания второго и последующих слов еще на один такт (рис. 5.11, *г),* благодаря чему формула приобретает вид 5-1-1-1.

Микросхемы EDRAM. Более быстрая версия DRAM была разработана подразделением фирмы Ramtron — компанией Enhanced Memory Systems. Технология реализована в вариантах FPM, EDO и BEDO. У микросхемы более быстрое ядро и внутренняя кэш-память. Наличие последней — главная особенность технологии. В роли кэш-памяти выступает статическая память (SRAM) емкостью 2048 бит. Ядро EDRAM имеет 2048 столбцов, каждый из которых соединен с внутренней кэш-памятью. При обращении к какой-либо ячейке одновременно считывается целая строка (2048 бит). Считанная строка заносится в SRAM, причем перенос информации в кэш-память практически не сказывается на быстродействии, поскольку происходит за один такт. При дальнейших обращениях к ячейкам, относящимся к той же строке, данные берутся из более быстрой кэш-памяти. Следующее обращение к ядру происходит при доступе к ячейке, не расположенной в строке, хранимой в кэш-памяти микросхемы.

Технология наиболее эффективна при последовательном чтении, то есть когда среднее время доступа для микросхемы приближается к значениям, характерным для статической памяти (порядка 10 нс). Главная сложность состоит в несовместимости с контроллерами, используемыми при работе с другими видами DRAM.

Синхронные динамические ОЗУ. В синхронных DRAM обмен информацией синхронизируется внешними тактовыми сигналами и происходит в строго определенные моменты

времени, что позволяет взять все от пропускной способности шины «процессор-память» и избежать циклов ожидания. Адресная и управляющая информация фиксируются в ИМС памяти. После чего ответная реакция микросхемы произойдет через четко определенное число тактовых импульсов, и это время процессор может использовать для других действий, не связанных с обращением к памяти. В случае синхронной динамической памяти вместо продолжительности цикла доступа говорят о минимально допустимом периоде тактовой частоты, и речь уже идет о времени порядка 8-10 не.

Микросхемы SDRAM. Аббревиатура SDRAM (Synchronous DRAM — синхронная DRAM) используется для обозначения микросхем «обычных» синхронных динамических ОЗУ. Кардинальные отличия SDRAM от рассмотренных выше асинхронных динамических ОЗУ можно свести к четырем положениям:

синхронный метод передачи данных на шину;

конвейерный механизм пересылки пакета;

применение нескольких (двух или четырех) внутренних банков памяти;

передача части функций контроллера памяти логике самой микросхемы.

Синхронность памяти позволяет контроллеру памяти «знать» моменты готовности данных, за счет чего снижаются издержки циклов ожидания и поиска данных. Так как данные появляются на выходе ИМС одновременно с тактовыми импульсами, упрощается взаимодействие памяти с другими устройствами ВМ.

В отличие от BEDO конвейер позволяет передавать данные пакета по тактам, благодаря чему ОЗУ может работать бесперебойно на более высоких частотах, чем асинхронные ОЗУ. Преимущества конвейера особенно возрастают при передаче длинных пакетов, но не превышающих длину строки микросхемы.

Значительный эффект дает разбиение всей совокупности ячеек на независимые внутренние массивы (банки). Это позволяет совмещать доступ к ячейке одного банка с подготовкой к следующей операции в остальных банках (перезарядкой управляющих цепей и восстановлением информации). Возможность держать открытыми одновременно несколько строк памяти (из разных банков) также способствует повышению быстродействия памяти. При поочередном доступе к банкам частота обращения к каждому из них в отдельности уменьшается пропорционально числу банков и SDRAM может работать на более высоких частотах. Благодаря встроенному счетчику адресов SDRAM, как и BEDO DRAM, позволяет производить чтение и запись в пакетном режиме, причем в SDRAM длина пакета варьируется и в пакетном режиме есть возможность чтения целой строки памяти. ИМС может быть охарактеризована формулой 5-1-1-1. Несмотря на то, что формула для этого типа динамической памяти такая же, что и у BEDO, способность работать на более высоких частотах приводит к тому, что SDRAM с двумя банками при тактовой частоте шины 100 МГц по производительности может почти вдвое превосходить память типа BEDO.

Микросхемы DDR SDRAM. Важным этапом в дальнейшем развитии технологии SDRAM стала DDR SDRAM (Double Data Rate SDRAM - SDRAM с удвоенной скоростью передачи данных). В отличие от SDRAM новая модификация выдает данные в пакетном режиме по обоим фронтам импульса синхронизации, за счет чего пропускная способность возрастает вдвое. Существует несколько спецификаций DDR SDRAM, в зависимости от тактовой частоты системной шины: DDR266, DDR333, DDR400, DDR533. Так, пиковая пропускная способность микросхемы памяти спецификации DDR333 составляет 2,7 Гбайт/с, а для DDR400 — 3,2 Гбайт/с. DDR SDRAM в настоящее время является наиболее распространенным типом динамической памяти персональных ВМ.

Микросхемы RDRAM, DRDRAM. Наиболее очевидные способы повышения эффективности работы процессора с памятью — увеличение тактовой частоты шины либо ширины выборки (количества одновременно пересылаемых разрядов). К сожалению, попытки совмещения обоих вариантов наталкиваются на существенные технические трудности (с повышением частоты усугубляются проблемы электромагнитной совместимости, труднее становится обеспечить одновременность поступления потребителю всех параллельно пересылае-

мых битов информации). В большинстве синхронных DRAM (SDRAM, DDR) применяется широкая выборка (64 бита) при ограниченной частоте шины.

Принципиально отличный подход к построению DRAM был предложен компанией Rambus в 1997 году. В нем упор сделан на повышение тактовой частоты до 400 МГц при одновременном уменьшении ширины выборки до 16 бит. Новая память известна как RDRAM (Rambus Direct RAM). Существует несколько разновидностей этой технологии: Base, Concurrent и Direct. Во всех тактирование ведется по обоим фронтам синхросигналов (как в DDR), благодаря чему результирующая частота составляет соответственно 500-600, 600-700 и 800 МГц. Два первых варианта практически идентичны, а вот изменения в технологии Direct Rambus (DRDRAM) весьма значительны.

Сначала остановимся на принципиальных моментах технологии RDRAM, ориентируясь в основном на более современный вариант — DRDRAM. Главным отличием от других типов DRAM является оригинальная система обмена данными между ядром и контроллером памяти, в основе которой лежит так называемый «канал Rambus», применяющий асинхронный блочно-ориентированный протокол. На логическом уровне информация между контроллером и памятью передается пакетами.

Различают три вида пакетов: пакеты данных, пакеты строк и пакеты столбцов. Пакеты строк и столбцов служат для передачи от контроллера памяти команд управления соответственно линиями строк и столбцов массива запоминающих элементов. Эти команды заменяют обычную систему управления микросхемой с помощью сигналов RAS, CAS, WE и CS.

Микросхемы SLDRAM. Потенциальным конкурентом RDRAM на роль стандарта архитектуры памяти для будущих персональных ВМ выступает новый вид динамического ОЗУ, разработанный консорциумом производителей ВМ SyncLink Consortium и известный под аббревиатурой SLDRAM. В отличие от RDRAM, технология которой является собственностью компаний Rambus и Intel, данный стандарт — открытый. На системном уровне технологии очень похожи. Данные и команды от контроллера к памяти и обратно в SLDRAM передаются пакетами по 4 или 8 посылок. Команды, адрес и управляющие сигналы посылаются по однонаправленной 10-разрядной командной шине. Считываемые и записываемые данные передаются по двунаправленной 18-разрядной шине данных. Обе шины работают на одинаковой частоте. Пока что еще эта частота равна 200 МГц, что, благодаря технике DDR, эквивалентно 400 МГц. Следующие поколения SLDRAM должны работать на частотах 400 МГц и выше, то есть обеспечивать эффективную частоту более 800 МГц.

К одному контроллеру можно подключить до 8 микросхем памяти. Чтобы избежать запаздывания сигналов от микросхем, более удаленных от контроллера, временные характеристики для каждой микросхемы определяются и заносятся в ее управляющий регистр при включении питания.

Микросхемы ESDRAM. Это синхронная версия EDRAM, в которой используются те же приемы сокращения времени доступа. Операция записи в отличие от чтения происходит в обход кэш-памяти, что увеличивает производительность ESDRAM при возобновлении чтения из строки, уже находящейся в кэш-памяти. Благодаря наличию в микросхеме двух банков простои из-за подготовки к операциям чтения/записи сводятся к минимуму. Недостатки у рассматриваемой микросхемы те же, что и у EDRAM — усложнение контроллера, так как он должен учитывать возможность подготовки к чтению в кэш-память новой строки ядра. Кроме того, при произвольной последовательности адресов кэш-память задействуется неэффективно.

Микросхемы CDRAM. Данный тип ОЗУ разработан в корпорации Mitsubishi, и его можно рассматривать как пересмотренный вариант ESDRAM, свободный от некоторых ее несовершенств. Изменены емкость кэш-памяти и принцип размещения в ней данных. Емкость одного блока, помещаемого в кэш-память, уменьшена до 128 бит, таким образом, в 16-килобитовом кэше можно одновременно хранить копии из 128 участков памяти, что позволяет эффективнее использовать кэш-память. Замена первого помещенного в кэш участка памяти начинается только после заполнения последнего (128-го) блока. Изменению подверг-

лись и средства доступа. Так, в микросхеме используются раздельные адресные шины для статического кэша и динамического ядра. Перенос данных из динамического ядра в кэш-память совмещен с выдачей данных на шину, поэтому частые, но короткие пересылки не снижают производительности ИМС при считывании из памяти больших объемов информации и уравнивают CDRAM с ESDRAM, а при чтении по выборочным адресам CDRAM явно выигрывает. Необходимо, однако, отметить, что вышеперечисленные изменения привели к еще большему усложнению контроллера памяти.

ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Процедура программирования таких ПЗУ обычно предполагает два этапа: сначала производится стирание содержимого всех или части ячеек, а затем производится запись новой информации.

В этом классе постоянных запоминающих устройств выделяют несколько групп:

- EPROM (Erasable Programmable ROM — стираемые программируемые ПЗУ);
- EEPROM (Electrically Erasable Programmable ROM — электрически стираемые программируемые ПЗУ);
- флэш-память.

Микросхемы EPROM. В EPROM запись информации производится электрическими сигналами, так же как в PROM, однако перед операцией записи содержимое всех ячеек должно быть приведено к одинаковому состоянию (стерто) путем воздействия на микросхему ультрафиолетовым облучением[1]. Кристалл заключен в керамический корпус, имеющий небольшое кварцевое окно, через которое и производится облучение. Чтобы предотвратить случайное стирание информации, после облучения кварцевое окно заклеивают непрозрачной пленкой. Процесс стирания может выполняться многократно. Каждое стирание занимает порядка 20 мин.

Данные хранятся в виде зарядов плавающих затворов МОП-транзисторов, играющих роль конденсаторов с очень малой утечкой заряда. Заряженный ЗЭ соответствует логическому нулю, а разряженный — логической единице. Программирование микросхемы происходит с использованием технологии инжекции горячих электронов. Цикл программирования занимает нескольких сотен миллисекунд.

Микросхемы EEPROM. Более привлекательным вариантом многократно программируемой памяти является электрически стираемая программируемая постоянная память EEPROM. Стирание и запись информации в эту память производятся побайтово, причем стирание — не отдельный процесс, а лишь этап, происходящий автоматически при записи. Операция записи занимает существенно больше времени, чем считывание — несколько сотен микросекунд на байт. В микросхеме используется тот же принцип хранения информации, что и в EPROM. Программирование EPROM не требует специального программатора и реализуется средствами самой микросхемы.

Флэш-память. Относительно новый вид полупроводниковой памяти — это флэш-память (название flash можно перевести как «вспышка молнии», что подчеркивает относительно высокую скорость перепрограммирования). Впервые анонсированная в середине 80-х годов, флэш-память во многом похожа на EEPROM, но использует особую технологию построения запоминающих элементов. Аналогично EEPROM, во флэш-памяти стирание информации производится электрическими сигналами, но не побайтово, а по блокам или полностью. Здесь следует отметить, что существуют микросхемы флэш-памяти с разбивкой на очень мелкие блоки (страницы) и автоматическим постраничным стиранием, что сближает их по возможностям с EEPROM. Как и в случае с EEPROM, микросхемы флэш-памяти выпускаются в вариантах с последовательным и параллельным доступом.

По организации массива ЗЭ различают микросхемы типа:

Bulk Erase (тотальная очистка) — стирание допустимо только для всего массива ЗЭ;

Boot Lock — массив разделен на несколько блоков разного размера, содержимое которых может очищаться независимо. У одного из блоков есть аппаратные средства для защи-

ты от стирания;

Flash File — массив разделен на несколько равноправных блоков одинакового размера, содержимое которых может стираться независимо.

Полностью содержимое флэш-памяти может быть очищено за одну или несколько секунд, что значительно быстрее, чем у EEPROM. Программирование (запись) байта занимает время порядка 10 мкс, а время доступа при чтении составляет 35-200 нс.

Как и в EEPROM, используется только один транзистор на бит, благодаря чему достигается высокая плотность размещения информации на кристалле (на 30% выше чем у DRAM).

ЭНЕРГОНЕЗАВИСИМЫЕ ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Под понятие *энергонезависимое ОЗУ* (NVRAM — Non-Volatile RAM) подпадает несколько типов памяти. От перепрограммируемых постоянных ЗУ их отличает отсутствие этапа стирания, предваряющего запись новой информации, поэтому вместо термина «программирование» для них употребляют стандартный термин «запись».

Микросхемы BBSRAM. К рассматриваемой группе относятся обычные статические ОЗУ со встроенным литиевым аккумулятором и усиленной защитой от искажения информации в момент включения и отключения внешнего питания. Для их обозначения применяют аббревиатуру BBSRAM (Battery-Back SRAM).

Микросхемы NVRAM. Другой подход реализован в микросхеме, разработанной компанией Simtec. Особенность ее в том, что в одном корпусе объединены статическое ОЗУ и перепрограммируемая постоянная память типа EEPROM. При включении питания данные копируются из EEPROM в SRAM, а при выключении — автоматически перезаписываются из SRAM в EEPROM. Благодаря такому приему данный вид памяти можно считать энергонезависимым.

Микросхемы FRAM. FRAM (Ferroelectric RAM — ферроэлектрическая память) разработана компанией Ramtron и представляет собой еще один вариант энергонезависимой памяти. По быстродействию данное ЗУ несколько уступает динамическим ОЗУ и пока рассматривается лишь как альтернатива флэш-памяти. Причисление FRAM к оперативным ЗУ обусловлено отсутствием перед записью явно выраженного цикла стирания информации.

Запоминающий элемент FRAM похож на ЗЭ динамического ОЗУ, то есть состоит из конденсатора и транзистора. Отличие заключено в диэлектрических свойствах материала между обкладками конденсатора. В FRAM этот материал (несмотря на название, он не содержит железа и имеет химическую формулу $BaTiO_3$,) обладает большой диэлектрической постоянной и может быть поляризован с помощью электрического поля. Поляризация сохраняется вплоть до ее изменения противоположно направленным электрическим полем, что и обеспечивает энергонезависимость данного вида памяти. Данные считываются за счет воздействия на конденсатор электрического поля. Величина возникающего при этом тока зависит от того, изменяет ли приложенное поле направление поляризации на противоположное или нет, что может быть зафиксировано усилителями считывания. В процессе считывания содержимое ЗЭ разрушается и должно быть восстановлено путем повторной записи, то есть как и DRAM, данный тип ЗУ требует регенерации. Количество циклов перезаписи для FRAM обычно составляет 10 млрд.

Главное достоинство данной технологии в значительно более высокой скорости записи по сравнению с EEPROM. В то же время относительная простота ЗЭ позволяет добиться высокой плотности размещения элементов на кристалле, сопоставимой с DRAM. FRAM выпускаются в виде микросхем, полностью совместимых с последовательными и параллельными EEPROM. Примером может служить серия 24Cxx.

СПЕЦИАЛЬНЫЕ ТИПЫ ОПЕРАТИВНОЙ ПАМЯТИ

В ряде практических задач более выгодным оказывается использование специализи-

рованных архитектур ОЗУ, где стандартные функции (запись, хранение, считывание) сочетаются с некоторыми дополнительными возможностями или учитывают особенности применения памяти. Такие виды ОЗУ называют специализированными и к ним причисляют:

- память для видеоадаптеров;
- память с множественным доступом (много портовые ОЗУ);
- память типа очереди (ОЗУ типа FIFO).

Два последних типа относятся к статическим ОЗУ.

Оперативные запоминающие устройства для видеоадаптеров

Использование памяти в видеоадаптерах имеет свою специфику и для реализации дополнительных требований прибегают к несколько иным типам микросхем. Так, при создании динамичных изображений часто достаточно просто изменить расположение уже хранящейся в видеопамяти информации. Вместо того чтобы многократно пересылать по шине одни и те же данные, лишь несколько изменив их расположение, выгоднее заставить микросхему памяти переместить уже хранящиеся в ней данные из одной области ядра в другую. На ИМС памяти можно также возложить операции по изменению цвета точек изображения.

Кратко рассмотрим некоторые из типов ОЗУ, ориентированных на применение в качестве видеопамяти.

Микросхемы SGRAM. Аббревиатура SGRAM (Synchronous Graphic DRAM - синхронное графическое динамическое ОЗУ) обозначает специализированный вид синхронной памяти с повышенной внутренней скоростью передачи данных. SGRAM может самостоятельно выполнять некоторые операции над видеоданными, в частности блочную запись. Предусмотрены два режима такой записи. В первом — режиме блочной записи (Block Write) — можно изменять цвет сразу восьми элементов изображения (пикселов). Назначение второго режима — блочной записи с маскированием определенных битов (Masked Write или Write-per-Bit) — предотвратить изменение цвета для отдельных пикселов пересылаемого блока. Имеется также модификация данной микросхемы, известная как DDR SGRAM, отличие которой очевидно из приставки DDR. Использование обоих фронтов синхросигналов ведет к соответствующему повышению быстродействия ИМС.

Микросхемы VRAM. ОЗУ типа VRAM (Video RAM) отличается высокой производительностью и предназначено для мощных графических систем. При разработке ставилась задача обеспечить постоянный поток данных при обновлении изображения на экране. Для типовых значений разрешения и частоты обновления изображения интенсивность потока данных приближается к 200 Мбит/с. В таких условиях процессору трудно получить доступ к видеопамяти для чтения или записи. Чтобы разрешить эту проблему, в микросхеме сделаны существенные архитектурные изменения, позволяющие обособить обмен между процессором и ядром VRAM для чтения/записи информации и операции по выдаче информации на схему формирования видеосигнала (ЦАП — цифро-аналоговый преобразователь). Связь памяти с процессором обеспечивается параллельным портом, а с ЦАП — дополнительным последовательным портом. Кроме того, динамическое ядро DRAM дополнено памятью с последовательным доступом (SAM — Serial Access Memory) емкостью 4 Кбайт. Оба вида памяти связаны между собой широкой внутренней шиной. Выводимая на экран информация порциями по 4 Кбайт из ядра пересылается в SAM и уже оттуда, в последовательном коде (последовательный код формируется с помощью подключенных к SAM сдвиговых регистров), поступает на ЦАП. В момент перезаписи в SAM новой порции ядро VRAM полностью готово к обслуживанию запросов процессора. Наряду с режимами Block Write и Write-per-Bit микросхема реализует режим Flash Write, позволяющий очистить целую строку памяти. Имеется также возможность маскировать определенные ячейки, защищая их от записи.

Микросхемы WRAM. Данный вид микросхем, разработанный компанией Samsung, во многом похож на VRAM. Это также двух портовая память, допускающая одновременный доступ со стороны процессора и ЦАП, но по конструкции она несколько проще, чем VRAM. Имеющиеся в VRAM, но редко используемые функции исключены, а вместо них введены дополнительные функции, ускоряющие вывод на экран текста и заполнение одним цветом

больших площадей экрана. В WRAM применена более быстрая схема буферизации данных и увеличена разрядность внутренней шины. Ускорено также ядро микросхемы, за счет использования режима скоростного страничного режима (UFP — Ultra Fast Page), что обеспечивает время доступа порядка 15 нс. В среднем WRAM на 50% производительнее, чем VRAM, и на 20% дешевле. Применяется микросхема в мощных видеоадаптерах.

Микросхемы MDRAM. Микросхема типа MDRAM (Multibank DRAM — многоблочное динамическое ОЗУ) разработана компанией MoSys и ориентирована на графические карты. Память содержит множество независимых банков по 1К 32-разрядных слов каждый. Банки подключены к быстрой и широкой внутренней шине. Каждый банк может выполнять определенные операции независимо от других банков. Отказ любого из банков ведет лишь к сокращению суммарной емкости памяти и некоторому снижению показателей быстродействия. Благодаря блочному построению технология позволяет изготавливать микросхемы практически любой емкости, не обязательно кратной степени числа 2.

Микросхемы 3D-RAM. Этот тип памяти разработан совместно компаниями Mitsubishi и Sun Microsystems с ориентацией на трехмерные графические ускорители. Помимо массива запоминающих элементов, микросхема 3D-RAM (трехмерная RAM) содержит процессор (арифметико-логическое устройство) и кэш-память. Процессор позволяет выполнять некоторые операции с изображением прямо в памяти. Основные преобразования над пикселами реализуются за один такт, поскольку стандартная последовательность действий «считал, изменил, записал» сводится к одной операции — «изменить», выполняемой в момент записи. Процессор микросхемы позволяет за секунду выполнить около 400 млн операций по обработке данных и закрасить до 4 млн элементарных треугольников. Кэш-память обеспечивает более равномерную нагрузку на процессор при интенсивных вычислениях. Ядро 3D-RAM состоит из четырех банков общей емкостью 10 Мбит. Размер строк памяти выбран таким, чтобы в пределах одной и той же области памяти находилось как можно больше трехмерных объектов. Это дает возможность сэкономить время на переходы со строки на строку. По цене данный тип микросхем сравним с VRAM.

Многопортовые ОЗУ

В отличие от стандартного в n-портовом ОЗУ имеется *n* независимых наборов шин адреса, данных и управления, гарантирующих одновременный и независимый доступ к ОЗУ *n* устройствам. Данное свойство позволяет существенно упростить создание многопроцессорных и многомашинных вычислительных систем, где многопортовое ОЗУ выступает в роли общей или совместно используемой памяти. В рамках одной ВМ подобное ОЗУ может обеспечивать обмен информацией между ЦП и УВВ (например, контроллером магнитного диска) намного эффективней, чем прямой доступ к памяти. В настоящее время серийно выпускаются двух- и четырехпортовые микросхемы, среди которых наиболее распространены первые.

ОБНАРУЖЕНИЕ И ИСПРАВЛЕНИЕ ОШИБОК

При работе с полупроводниковой памятью не исключено возникновение различного рода отказов и сбоев. Причиной *отказов* могут быть производственные дефекты, повреждение микросхем или их физический износ. Проявляются отказы в том, что в отдельных разрядах одной или нескольких ячеек постоянно считывается 0 или 1, вне зависимости от реально записанной туда информации. *Сбой* — это случайное событие, выражающееся в неверном считывании или записи информации в отдельных разрядах одной или нескольких ячеек, не связанное с дефектами микросхемы. Сбои обычно обусловлены проблемами с источником питания или с воздействием альфа-частиц, возникающих в результате распада радиоактивных элементов, которые в небольших количествах присутствуют практически в любых материалах. Как отказы, так и сбои крайне нежелательны, поэтому в большинстве систем основной памяти содержатся схемы, служащие для обнаружения и исправления ошибок.

Вне зависимости от того, как именно реализуется контроль и исправление ошибок, в основе их всегда лежит введение избыточности. Это означает, что контролируемые разряды

дополняются контрольными разрядами, благодаря которым и возможно детектирование ошибок, а в ряде методов — их коррекция. Общую схему обнаружения и исправления ошибок иллюстрирует рис. 5.15.

На рисунке показано, каким образом осуществляются обнаружение и исправление ошибок. Перед записью М-разрядных данных в память производится их обработка, обозначенная на схеме функцией «f», в результате которой формируется добавочный К-разрядный код. В память заносятся как данные, так и этот вычисленный код, то есть *(М* + К)-разрядная информация. При чтении информации повторно формируется К-разрядный код, который сравнивается с аналогичным кодом, считанным из ячейки. Сравнение приводит к одному из трех результатов:

- **Не обнаружено ни одной ошибки.** Извлеченные из ячейки данные подаются на выход памяти.
- **Обнаружена ошибка, и она может быть исправлена.** Биты данных и добавочного кода подаются на схему коррекции. После исправления ошибки в М-разрядных данных они поступают на выход памяти.
- **Обнаружена ошибка, и она не может быть исправлена.** Выдается сообщение о неисправимой ошибке.

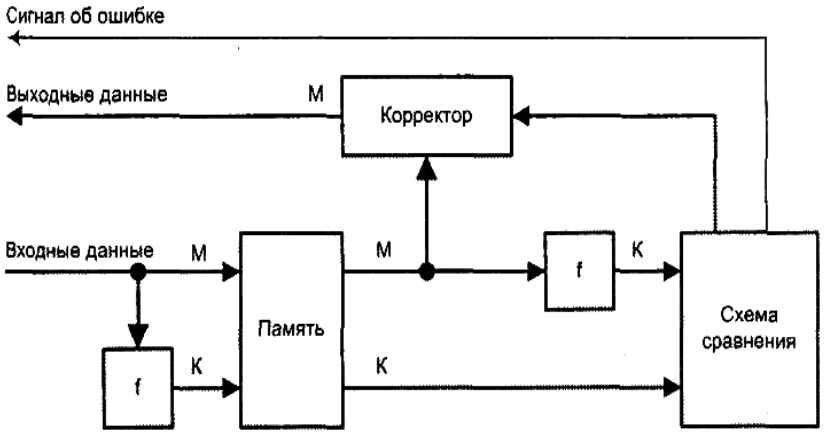


Рис. 5.15. Общая схема обнаружения и исправления ошибок [200]

Коды, используемые для подобных операций, называют *корректирующими кодами* или *кодами с исправлением ошибок..*

СТЕКОВАЯ ПАМЯТЬ

Стековая память обеспечивает такой режим работы, когда информация записывается и считывается по принципу *«последним записан — первым считан»* (LIFO -Last In First Out). Память с подобной организацией широко применяется для запоминания и восстановления содержимого регистров процессора (контекста) при обработке подпрограмм и прерываний.

АССОЦИАТИВНАЯ ПАМЯТЬ

В рассмотренных ранее видах запоминающих устройств доступ к информации требовал указания адреса ячейки. Зачастую значительно удобнее искать информацию не по адресу, а опираясь на какой-нибудь характерный признак, содержащийся в самой информации. Такой принцип лежит в основе ЗУ, известного как *ассоциативное запоминающее устройство* (АЗУ). В литературе встречаются и иные названия подобного ЗУ:

память, адресуемая по содержанию (content addressable memory);
память, адресуемая по данным (data addressable memory);
память с параллельным поиском (parallel search memory);
каталоговая память (catalog memory); информационное ЗУ (information storage);
тегированная память (tag memory).

Ассоциативное ЗУ — это устройство, способное хранить информацию, сравнивать ее с некоторым заданным образцом и указывать на их соответствие или несоответствие друг другу. Признак, по которому производится поиск информации, будем называть *ассоциативньш признаком,* а кодовую комбинацию, выступающую в роли образца для поиска, — *признаком поиска.* Ассоциативный признак может быть частью искомой информации или дополнительно придаваться ей. В последнем случае его принято называть *тегом* или *ярлыком.*

КЭШ-ПАМЯТЬ

Как уже отмечалось, в качестве элементной базы основной памяти в большинстве ВМ служат микросхемы динамических ОЗУ, на порядок уступающие по быстродействию центральному процессору. В результате процессор вынужден простаивать несколько тактовых периодов, пока информация из памяти установится на шине данных ВМ. Если ОП выполнить на быстрых микросхемах статической памяти, стоимость ВМ возрастет весьма существенно. Экономически приемлемо решение этой проблемы было предложено М. Уилксом в 1965 году в процессе разработки ВМ Atlas и заключается оно в использовании двухуровневой памяти, когда между ОП и процессором размещается небольшая, но быстродействующая буферная память. В процессе работы такой системы в буферную память копируются участки ОП, к которым производится обращение со стороны процессора. В общепринятой терминологии — производится *отображение* участков ОП на буферную память. Выигрыш достигается за счет ранее рассмотренного свойства локальности — если отобразить участок ОП в более быстродействующую буферную память и переадресовать на нее все обращения в пределах скопированного участка, можно добиться существенного повышения производительности ВМ. Уилкс называл рассматриваемую буферную память подчиненной (slave тегу). Позже распространение получил термин *кэш-память* (от английского слова cache — убежище, тайник), поскольку такая память обычно скрыта от программиста в том смысле, что он не может ее адресовать и может даже вообще не знать о ее существовании. Впервые кэш-системы появились в машинах модели 85 семейства IBM 360.

На эффективность применения кэш-памяти в иерархической системе памяти влияет целый ряд моментов. К наиболее существенным из них можно отнести:

- емкость кэш-памяти;
- размер строки;
- способ отображения основной памяти на кэш-память;
- алгоритм замещения информации в заполненной кэш-памяти;
- алгоритм согласования содержимого основной и кэш-памяти;
- число уровней кэш-памяти.

ЕМКОСТЬ КЭШ - ПАМЯТИ

Выбор емкости кэш-памяти — это всегда определенный компромисс. С одной стороны, кэш-память должна быть достаточно мала, чтобы ее стоимостные показатели были близки к величине, характерной для ОП. С другой — она должна быть достаточно большой, чтобы среднее время доступа в системе, состоящей из основной и кэш-памяти, определялось временем доступа к кэш-памяти. В пользу уменьшения размера кэш-памяти имеется больше мотивировок. Так, чем вместительнее кэш-память, тем больше логических схем должно участвовать в ее адресации. Как следствие, кэш-память повышенной емкости работают медленнее по сравнению с микросхемами меньшей емкости, даже если они выполнены по одной и той же технологии.

Реальная эффективность использования кэш-памяти зависит от характера решаемых задач, и невозможно заранее определить, какая ее емкость будет действительно оптимальной. Установлено, что для большинства задач близкой к оптимальной является кэш-память емкостью от 1 до 512 Кбайт.

ОДНОУРОВНЕВАЯ И МНОГОУРОВНЕВАЯ КЭШ – ПАМЯТЬ

Современные технологии позволяют разместить кэш-память и ЦП на общем кристал-

ле. Такая внутренняя кэш-память строится по технологии статического ОЗУ и является наиболее быстродействующей. Емкость ее обычно не превышает 64 Кбайт. Попытки увеличения емкости обычно приводят к снижению быстродействия, главным образом из-за усложнения схем управления и дешифрации адреса. Общую емкость кэш-памяти ВМ увеличивают за счет второй (внешней) кэш-памяти, расположенной между внутренней кэш-памятью и ОП. Такая система известна под названием двухуровневой, где внутренней кэш-памяти отводится роль первого уровня (L1), а внешней — второго уровня (L2). Емкость L2 обычно на порядок больше, чем у L1, а быстродействие и стоимость — несколько ниже. Память второго уровня также строится как статическое ОЗУ. Типичная емкость кэш-памяти второго уровня — 256 и 512 Кбайт, реже — 1 Мбайт, а реализуется она, как правило, в виде отдельной микросхемы, хотя в последнее время L2 часто размещают на одном кристалле с процессором, за счет чего сокращается длина связей и повышается быстродействие.

При доступе к памяти ЦП сначала обращается к кэш-памяти первого уровня. В случае промаха производится обращение к кэш-памяти второго уровня. Если информация отсутствует и в L2, выполняется обращение к ОП и соответствующий блок заносится сначала в L2, а затем и в L1. Благодаря такой процедуре часто запрашиваемая информация может быть быстро восстановлена из кэш-памяти второго уровня.

Для ускорения обмена информацией между ЦП и L2 между ними часто вводят специальную шину, так называемую *шину заднего плана,* в отличие от *шины переднего плана,* связывающей ЦП с основной памятью.

Количество уровней кэш-памяти не ограничивается двумя. В некоторых ВМ уже можно встретить кэш-память третьего уровня (L3) и ведутся активные дискуссии о введении также и кэш-памяти четвертого уровня (L4). Характер взаимодействия очередного уровня с предшествующим аналогичен описанному для L1 и L2. Таким образом, можно говорить об иерархии кэш-памяти. Каждый последующий уровень характеризуется большей емкостью, меньшей стоимостью, но и меньшим быстродействием, хотя оно все же выше, чем у ЗУ основной памяти.

ДИСКОВАЯ КЭШ-ПАМЯТЬ

Концепция кэш-памяти применима и к дисковым ЗУ. Принцип кэширования дисков во многом схож с принципом кэширования основной памяти, хотя способы доступа к диску и ОП существенно разнятся. Если время обращения к любой ячейке ОП одинаково, то для диска оно зависит от целого ряда факторов. Во-первых, нужно затратить некоторое время для установки головки считывания/записи на нужную дорожку. Во-вторых, поскольку при движении головка вибрирует, необходимо подождать, чтобы она успокоилась. В-третьих, искомый сектор может оказаться под головкой также лишь спустя некоторое время.

Дисковая кэш-память представляет собой память с произвольным доступом, «размещенную» между дисками и ОП. Емкость такой памяти обычно достаточно велика — от 8 Мбайт и более. Пересылка информации между дисками и основной памятью организуется контроллером дисковой кэш-памяти. Изготавливается дисковая кэш-память на базе таких же полупроводниковых запоминающих устройств, что и основная память, поэтому в ряде случаев с ней обращаются как с дополнительной основной памятью. С другой стороны, в ряде операционных систем, таких как UNIX, в качестве дискового кэша используется область основной памяти.

В дисковой кэш-памяти хранятся блоки информации, которые с большой вероятностью будут востребованы в ближайшем будущем. Принцип локальности, обеспечивающий эффективность обычной кэш-памяти, справедлив и для дисковой, приводя к сокращению времени ввода/вывода данных от величин 20-30 мс до значений порядка 2-5 мс, в зависимости объема передаваемой информации.

В качестве единицы пересылки может выступать сектор, несколько секторов, а также одна или несколько дорожек диска. Кроме того, иногда применяется пересылка информации, начиная с выбранного сектора на дорожке до ее конца. В случае пересылки секторов кэш-

память заполняется не только требуемым сектором, но секторами, непосредственно следующими за ним, так как известно, что в большинстве случаев взаимосвязанные данные хранятся в соседних секторах. Этот метод известен также как *опережающее чтение* (read ahead).

В дисковых кэшах обычно используется алгоритм сквозной записи. Специфика состоит в том, что далеко не всю информацию, перемещаемую между дисками и основной памятью, выгодно помещать в дисковый кэш. В ряде случаев определенные данные и команды целесообразно пересылать напрямую между ОП и диском. По этой причине в системах с дисковым кэшем предусматривают специальный динамический механизм, позволяющий переключать тракт пересылки информации: через кэш или минуя его.

Одна из привлекательных сторон дискового кэша в том, что связанные с ним преимущества могут быть получены без изменений в имеющемся аппаратном и программном обеспечении. Многие серийно выпускаемые дисковые кэши интегрированы в состав дисковых ЗУ.

Примечательно, что архитектура кэш-памяти современных магнитных дисков типа «винчестер» реализует полностью ассоциативное отображение.

ПОНЯТИЕ ВИРТУАЛЬНОЙ ПАМЯТИ

Для большинства типичных применений ВМ характерна ситуация, когда размещение всей программы в ОП невозможно из-за ее большого размера. В этом, однако, и нет принципиальной необходимости, поскольку в каждый момент времени «внимание» машины концентрируется на определенных сравнительно небольших участках программы. Таким образом, в ОП достаточно хранить только используемые в данный период части программ, а остальные части могут располагаться на внешних ЗУ (ВЗУ). Сложность подобного подхода в том, что процессы обращения к ОП и ВЗУ существенно различаются, и это усложняет задачу программиста. Выходом из такой ситуации было появление в 1959 году идеи *виртуализации памяти*, под которой понимается метод автоматического управления иерархической памятью, при котором программисту кажется, что он имеет дело с единой памятью большой емкости и высокого быстродействия. Эту память называют виртуальной (кажущейся) памятью. По своей сути виртуализация памяти представляет собой способ аппаратной и программной реализации концепции иерархической памяти.

В рамках идеи виртуализации памяти ОП рассматривается как линейное пространство N адресов, называемое *физическим пространством* памяти. Для задач, где требуется более чем *N* ячеек, предоставляется значительно большее пространство адресов (обычно равное общей емкости всех видов памяти), называемое *виртуальным пространством,* в общем случае не обязательно линейное. Адреса виртуального пространства называют *виртуальными,* а адреса физического пространства — *физическими.* Программа пишется в виртуальных адресах, но поскольку для ее выполнения нужно, чтобы обрабатываемые команды и данные находились в ОП, требуется, чтобы каждому виртуальному адресу соответствовал физический. Таким образом, в процессе вычислений необходимо, прежде всего, переписать из ВЗУ в ОП ту часть информации, на которую указывает виртуальный адрес (отобразить виртуальное пространство на физическое), после чего преобразовать виртуальный адрес в физический.

Среди систем виртуальной памяти можно выделить два класса: системы с фиксированным размером блоков (страничная организация) и системы с переменным размером блоков (сегментная организация). Оба варианта обычно совмещают (сегментно-страничная организация).

МАССИВЫ МАГНИТНЫХ ДИСКОВ С ИЗБЫТОЧНОСТЬЮ

Магнитные диски, будучи основой внешней памяти любой ВМ, одновременно остаются и одним из «узких мест» из-за сравнительно высокой стоимости, недостаточной производительности и отказоустойчивости. Характерно, что если в плане стоимости и надежности ситуация улучшается, то разрыв в производительности между МД и ядром ВМ постоянно растет. Так, при удвоении быстродействия процессоров примерно каждые два года для МД

такое удвоение было достигнуто лишь спустя десять лет. Ясно, что уже с самого начала использования подсистем памяти на базе МД не прекращаются попытки улучшить их характеристики. Одно из наиболее интересных и универсальных усовершенствований было предложено в 1987 году учеными университета Беркли (Калифорния) [179]. Проект известен под аббревиатурой RAID (Redundant Array of Independent (or Inexpensive) Disks) — *массив независимых (или недорогих) дисков с избыточностью.* В основе концепции RAID лежит переход от одного физического МД большой емкости к массиву недорогих, независимо и параллельно работающих физических дисковых ЗУ, рассматриваемых операционной системой как одно большое логическое дисковое запоминающее устройство. Такой подход позволяет повысить производительность дисковой памяти за счет возможности параллельного обслуживания запросов на считывание и запись, при условии, что данные находятся на разных дисках. Повышенная надежность достигается тем, что в массиве дисков хранится избыточная информация, позволяющая обнаружить и исправить возможные ошибки. На период, когда концепция RAID была впервые предложена, определенный выигрыш достигался и в плане стоимости. В настоящее время, с развитием технологии производства МД, утверждение об экономичности массивов RAID становится проблематичным, что, однако, вполне компенсируется их повышенными быстродействием и отказоустойчивостью.

Было предложено пять схем организации данных и способов введения избыточности, названные авторами уровнями RAID: RAID I, RAID 2,..., RAID 5. В настоящее время производители RAID-систем, объединившиеся в ассоциацию RAB (RAID Advisory Board), договорились о единой классификации RAID, включающей в себя шесть уровней (добавлен уровень RAID 0). Известны также еще несколько схем RAID, не включенных в эту классификацию, поскольку по сути они представляют собой различные комбинации стандартных уровней. Хотя ни одна из схем массива МД не может быть признана идеальной для всех случаев, каждая из них позволяет существенно улучшить какой-то из показателей (производительность, отказоустойчивость) либо добиться наиболее подходящего сочетания этих показателей. Для всех уровней RAID характерны три общих свойства:

RAID представляет собой набор физических дисковых ЗУ, управляемых операционной системой и рассматриваемых как один логический диск;

данные распределены по физическим дискам массива;

избыточное дисковое пространство используется для хранения дополнительной информации, гарантирующей восстановление данных в случае отказа диска.

Повышение производительности дисковой подсистемы

Повышение производительности дисковой подсистемы в RAID достигается с помощью приема, называемого *расслоением* или *расщеплением* (striping). В его основе лежит разбиение данных и дискового пространства на сегменты, так называемые *полосы* (strip — узкая полоса). Полосы распределяются по различным дискам массива, в соответствии с определенной системой. Это позволяет производить параллельное считывание или запись сразу нескольких полос, если они расположены на разных дисках. В идеальном случае производительность дисковой подсистемы может быть увеличена в число раз, равное количеству дисков в массиве. Размер (ширина) полосы выбирается исходя из особенностей каждого уровня RAID и может быть равен биту, байту, размеру физического сектора МД (обычно 512 байт) или размеру дорожки.

Чаще всего логически последовательные полосы распределяются по последовательным дискам массива. Так, в и-дисковом массиве *n* первых логических полос физически расположены как первые полосы на каждом из *n* дисков, следующие *n* полос — как вторые полосы на каждом физическом диске и т. д. Набор логически последовательных полос, одинаково расположенных на каждом ЗУ массива, называют *поясом* (stripe — широкая полоса).

Как уже упоминалось, минимальный объем информации, считываемый с МД или записываемый на него за один раз, равен размеру физического сектора диска. Это приводит к определенным проблемам при меньшей ширине полосы, которые в RAID обычно решаются за счет усложнения контроллера МД.

Повышение отказоустойчивости дисковой подсистемы

Одной из целей концепции RAID была возможность обнаружения и коррекции ошибок, возникающих при отказах дисков или в результате сбоев. Достигается это за счет избыточного дискового пространства, которое задействуется для хранения дополнительной информации, позволяющей восстановить искаженные или утерянные данные. В RAID предусмотрены три вида такой информации:

- дублирование;
- код Хэмминга;
- биты паритета.

Первый из вариантов заключается в дублировании всех данных, при условии, что экземпляры одних и тех же данных расположены на разных дисках массива. Это позволяет при отказе одного из дисков воспользоваться соответствующей информацией, хранящейся на исправных МД. В принципе распределение информации по дискам массива может быть произвольным, но для сокращения издержек, связанных с поиском копии, обычно применяется разбиение массива на пары МД, где в каждой паре дисков информация идентична и одинаково расположена. При таком дублировании для управления парой дисков может использоваться общий или раздельные контроллеры. Избыточность дискового массива здесь составляет 100%.

Второй способ формирования корректирующей информации основан на вычислении кода Хэмминга для каждой группы полос, одинаково расположенных на всех дисках массива (пояса). Корректирующие биты хранятся на специально выделенных для этой цели дополнительных дисках (по одному диску на каждый бит). Так, для массива из десяти МД требуются четыре таких дополнительных диска, и избыточность в данном случае близка к 30%.

В третьем случае вместо кода Хэмминга для каждого набора полос, расположенных в идентичной позиции на всех дисках массива, вычисляется контрольная полоса, состоящая из битов паритета. В ней значение отдельного бита формируется как сумма по модулю два для одноименных битов во всех контролируемых полосах. Для хранения полос паритета требуется только один дополнительный диск. В случае отказа какого-либо из дисков массива производится обращение к диску паритета, и данные восстанавливаются по битам паритета и данным от остальных дисков массива. Реконструкция данных достаточно проста.

RAID уровня О

RAID уровня 0, строго говоря, не является полноценным членом семейства RAID, поскольку данная схема не содержит избыточности и нацелена только на повышение производительности в ущерб надежности.

В основе RAID 0 лежит расслоение данных. Полосы распределены по всем дискам массива дисковых ЗУ по циклической схеме (рис. 5.39). Преимущество такого распределения в том, что если требуется записать или прочитать логически последовательные полосы, то несколько таких полос (вплоть до *n)* могут обрабатываться параллельно, за счет чего существенно снижается общее время ввода/вывода. Ширина полос в RAID 0 варьируется в зависимости от применения, но в любом случае она не менее размера физического сектора МД.

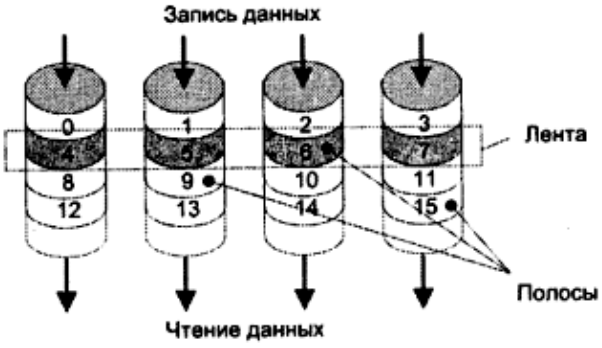


Рис. 5.39. RAID уровня 0

RAID 0 обеспечивает наиболее эффективное использование дискового пространства и максимальную производительность дисковой подсистемы при минимальных затратах и простоте реализации. Недостатком является незащищенность данных — отказ одного из дисков ведет к разрушению целостности данных во всем массиве. Тем не менее существует ряд приложений, где производительность и емкость дисковой системы намного важнее возможного снижения надежности. К таким можно отнести задачи, оперирующие большими файлами данных, в основном в режиме считывания информации (библиотеки изображений, большие таблицы и т. п.), и где загрузка информации в основную память должна производиться как можно быстрее. Учитывая отсутствие в RAID 0 средств по защите данных, желательно хранить дубликаты файлов на другом, более надежном носителе информации, например на магнитной ленте.

RAID уровня 1

В RAID 1 избыточность достигается с помощью дублирования данных. В принципе исходные данные и их копии могут размещаться по дисковому массиву произвольно, главное чтобы они находились на разных дисках. В плане быстродействия и простоты реализации выгоднее, когда данные и копии располагаются идентично на одинаковых дисках. Рисунок 5.40 показывает, что, как и в RAID 0, здесь имеет место разбиение данных на полосы. Однако в этом случае каждая логическая полоса отображается на два отдельных физических диска, так что каждый диск в массиве имеет так называемый «зеркальный» диск, содержащий идентичные данные: Для управления каждой парой дисков может быть использован общий контроллер, тогда данные сначала записываются на основной диск, а затем — на «зеркальный» («зеркалирование»). Более эффективно применение самостоятельных контроллеров для каждого диска, что позволяет производить одновременную запись на оба диска.

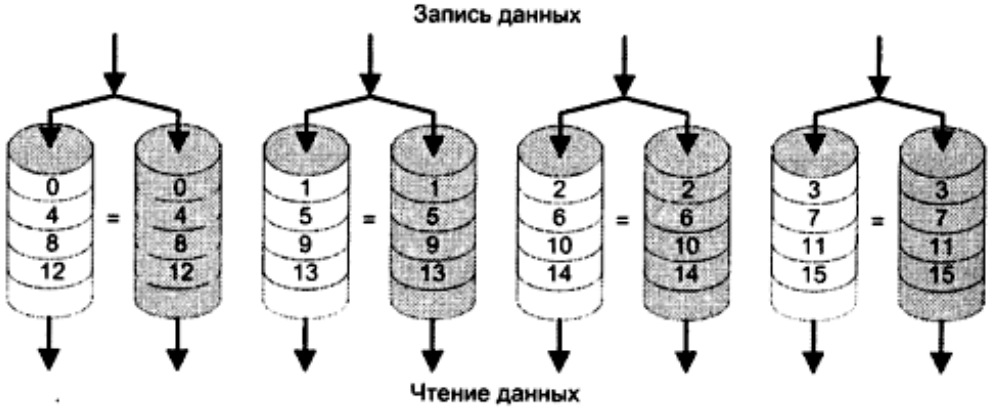


Рис. 5.40. RAID уровня 1

Запрос на чтение может быть обслужен тем из двух дисков, которому в данный момент требуется меньшее время поиска и меньшая задержка вращения. Запрос на запись требует, чтобы были обновлены обе соответствующие полосы, но это выполнимо и параллельно, причем задержка определяется тем диском, которому нужны большие время поиска и задержка вращения. В то же время у RAID 1 нет дополнительных затрат времени на вычисление вспомогательной корректирующей информации. Когда одно дисковое ЗУ отказывает, данные могут быть просто взяты со второго.

Принципиальный изъян RAID 1 — высокая стоимость: требуется вдвое больше физического дискового пространства. По этой причине использование RAID 1 обычно ограничивают хранением загрузочных разделов, системного программного обеспечения и данных, а также других особенно критичных файлов: RAID 1 обеспечивает резервное копирование всех данных, так что в случае отказа диска критическая информация доступна практически немедленно.

RAID уровня 2

В системах RAID 2 используется техника параллельного доступа, где в выполнении каждого запроса на В/ВЫВ одновременно участвуют все диски. Обычно шпиндели всех дисков синхронизированы так, что головки каждого ЗУ в каждый момент времени находятся в одинаковых позициях. Данные разбиваются на полосы длиной в 1 бит и распределены по дискам массива таким образом, что полное машинное слово представляется поясом, то есть число дисков равно длине машинного слова в битах. Для каждого слова вычисляется корректирующий код (обычно это код Хэмминга, способный корректировать одиночные и обнаруживать двойные ошибки), который, также побитово, хранится на дополнительных дисках (рис. 5.41). Например, для массива, ориентированного на 32-разрядные слова (32 основных диска) требуется семь дополнительных дисковых ЗУ (корректирующий код состоит из 7 разрядов).

При записи вычисляется корректирующий код, который заносится на отведенные для него диски. При каждом чтении производится доступ ко всем дискам массива, включая дополнительные. Считанные данные вместе с корректирующим кодом подаются на контроллер дискового массива, где происходит повторное вычисление корректирующего кода и его сравнение с хранившимся на избыточных дисках. Если присутствует одиночная ошибка, контроллер способен ее мгновенно распознать и исправить, так что время считывания не увеличивается.

RAID 2 позволяет достичь высокой скорости В/ВЫВ при работе с большими последовательными записями, но становится неэффективным при обслуживании записей небольшой длины. Основное преимущество RAID 2 состоит в высокой степени защиты информации, однако предлагаемый в этой схеме метод коррекции уже встроен в каждое из современных дисковых ЗУ.

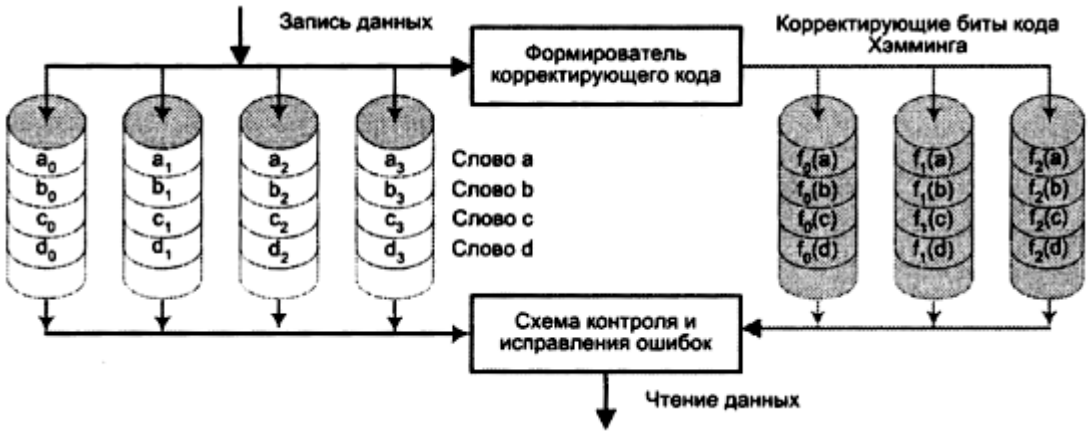


Рис. 5.41. RAID уровня 2

Корректирующие разряды вычисляются для каждого сектора диска и хранятся в соответствующем поле этих секторов. В таких условиях использование нескольких избыточных дисков представляется неэффективным, и массивы уровня RAID 2 в настоящее время не производятся.

RAID уровня 3

RAID 3 организован сходно с RAID2. Отличие в том, что RAID 3 требует только одного дополнительного диска — диска паритета, вне зависимости от того, насколько велик массив дисков (рис. 5.42). В RAID 3 используется параллельный доступ к данным, разбитым на полосы длиной в бит или байт. Все диски массива синхронизированы. Вместо кода Хэмминга для набора полос идентичной позиции на всех дисках массива (пояса) вычисляется полоса, состоящая из битов паритета. В случае отказа дискового ЗУ производится обращение к диску паритета, и данные восстанавливаются по битам паритета и данным от остальных дисков массива.

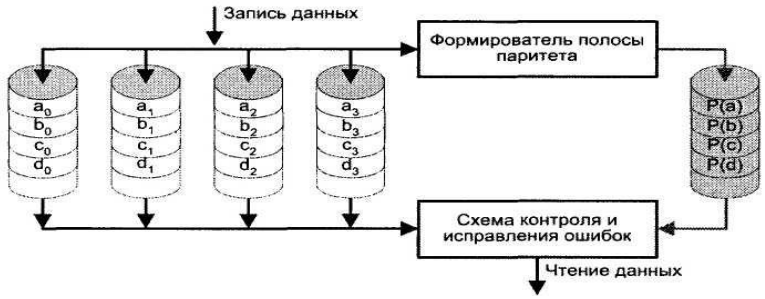


Рис. 5.42. RAID уровня 3

Так как данные разбиты на очень маленькие полосы, RAID 3 позволяет достигать очень высоких скоростей передачи данных. Каждый запрос на ввод/вывод приводит к параллельной передаче данных со всех дисков. Для приложений, связанных с большими пересылками данных, это обстоятельство очень существенно. С другой стороны, параллельное обслуживание одиночных запросов невозможно, и производительность дисковой подсистемы в этом случае падает.

Ввиду того что для хранения избыточной информации нужен всего один диск, причем независимо от их числа в массиве, именно уровню RAID 3 отдается предпочтение перед RAID 2.

RAID уровня 4

По своей идее и технике формирования избыточной информации RAID 4 идентичен RAID 3, только размер полос в RAID 4 значительно больше (обычно один-два физических блока на диске). Главное отличие состоит в том, что в RAID 4 используется техника независимого доступа, когда каждое ЗУ массива в состоянии функционировать независимо, так, что отдельные запросы на ввод/вывод могут удовлетворяться параллельно (рис. 5.43).

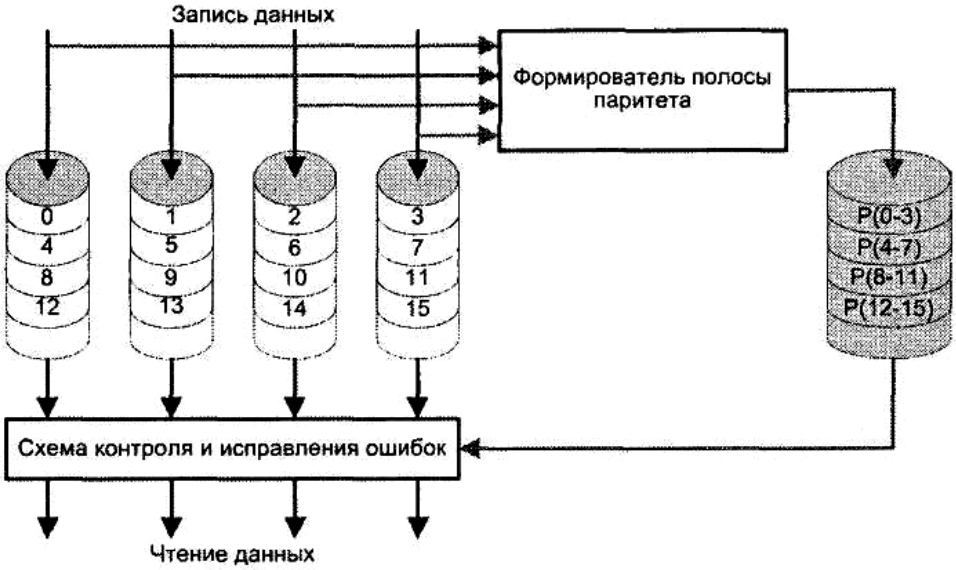


Рис. 5.43. RAID уровня 4

Для RAID 4 характерны издержки, обусловленные независимостью дисков. Если в RAID 3 запись производилась одновременно для всех полос одного пояса, в RAID 4 осуществляется запись полос в разные пояса. Это различие ощущается особенно при записи данных малого размера.

Каждый раз для выполнения записи программное обеспечение дискового массива должно обновить не только данные пользователя, но и соответствующие биты паритета.

Для вычисления новой полосы паритета программное обеспечение управления масси-

вом должно прочитать старую полосу пользователя и старую полосу паритета. Затем оно может заменить эти две полосы новой полосой данных и новой вычисленной полосой паритета. Таким образом, запись каждой полосы связана с двумя операциями чтения и двумя операциями записи.

В случае записи большого объема информации, охватывающего полосы на всех дисках, паритет вычисляется достаточно легко путем расчета, в котором участвуют только новые биты данных, то есть содержимое диска паритета может быть обновлено параллельно с дисками данных и не требует дополнительных операций чтения и записи.

Массивы RAID 4 наиболее подходят для приложений, требующих поддержки высокого темпа поступления запросов ввода/вывода, и уступает RAID 3 там, где приоритетен большой объем пересылок данных.

RAID уровня 5

RAID 5 имеет структуру, напоминающую RAID 4. Различие заключается в том, что RAID 5 не содержит отдельного диска для хранения полос паритета, а разносит их по всем дискам. Типичное распределение осуществляется по циклической схеме, как это показано на рис. 5.44. В я-дисковом массиве полоса паритета вычисляется для полос *n*— 1 дисков, расположенных в одном поясе, и хранится в том же поясе, но на диске, который не учитывался при вычислении паритета. При переходе от одного пояса к другому эта схема циклически повторяется.

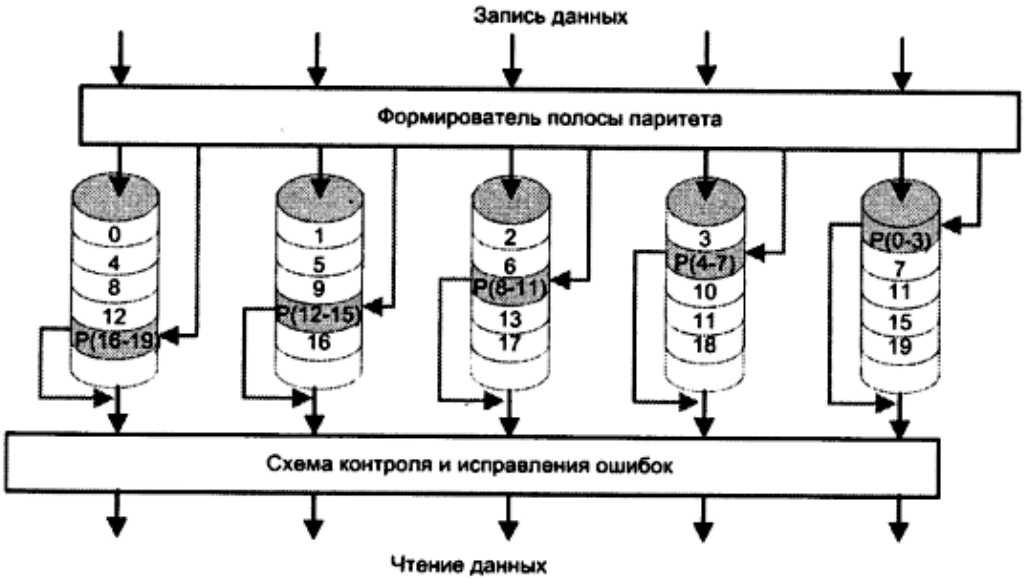


Рис. 5.44. RAID уровня 5

Распределение полос паритета по всем дискам предотвращает возникновение проблемы, упоминавшейся для RAID 4.

RAID уровня 6

RAID 6 очень похож на RAID 5. Данные также разбиваются на полосы размером в блок и распределяются по всем дискам массива. Аналогично, полосы паритета распределены по разным дискам. Доступ к полосам независимый и асинхронный. Различие состоит в том, что на каждом диске хранится не одна, а две полосы паритета. Первая из них, как и в RAID 5, содержит контрольную информацию для полос, расположенных на горизонтальном срезе массива (за исключением диска, где эта полоса паритета хранится). В дополнение формируется и записывается вторая полоса паритета, контролирующая все полосы какого-то одного диска массива (вертикальный срез массива), но только не того, где хранится полоса паритета. Сказанное иллюстрируется рис. 5.45.

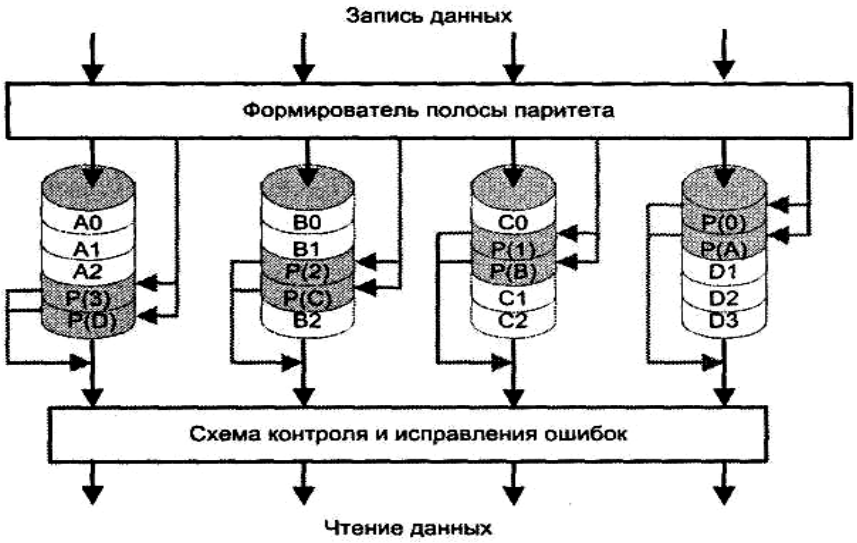


Рис. 5.45. RAID уровня 6

Такая схема массива позволяет восстановить информацию при отказе сразу двух дисков. С другой стороны, увеличивается время на вычисление и запись паритетной информации и требуется дополнительное дисковое пространство. Кроме того, реализация данной схемы связана с усложнением контроллера дискового массива. В силу этих причин схема среди выпускаемых RAID-систем встречается крайне редко.

Рис. 5.46. RAID уровня 7

RAID уровня 7

Схема RAID 7, запатентованная Storage Computer Corporation, объединяет массив асинхронно работающих дисков и кэш-память, управляемые встроенной в контроллер массива операционной системой реального времени (рис. 5.46). Данные разбиты на полосы размером в блок и распределены по дискам массива. Полосы паритета хранятся на специально выделенных для этой цели одном или нескольких дисках.

Схема некритична к виду решаемых задач и при работе с большими файлами не уступает по производительности RAID 3. Вместе с тем RAID 7 может так же эффективно, как и RAID 5, производить одновременно несколько операций чтения и записи для небольших объемов данных. Все это обеспечивается использованием кэш-памяти и собственной операционной системой.

RAID уровня 10

Данная схема совпадает с RAID 0, но в отличие от нее роль отдельных дисков выпол-

няют дисковые массивы, построенные по схеме RAID 1 (рис. 5.47).

Таким образом, в RAID 10 сочетаются расслоение и дублирование. Это позволяет добиться высокой производительности, характерной для RAID 0 при уровне отказоустойчивости RAID 1. Основной недостаток схемы — высокая стоимость ее реализации. Кроме того, необходимость синхронизации всех дисков приводит к усложнению контроллера.

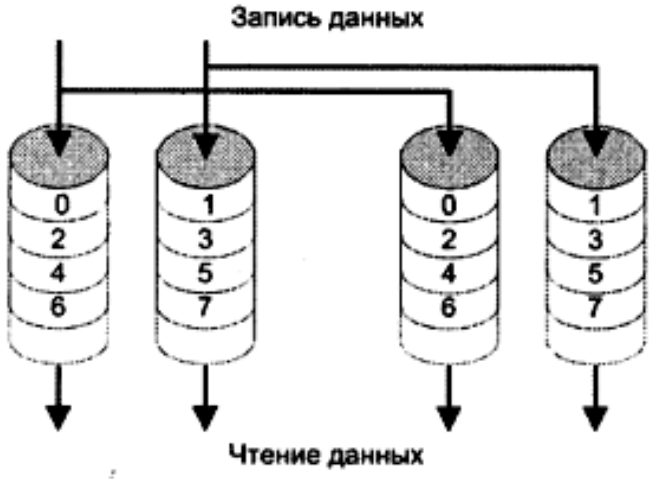


Рис. 5.47. RAID уровня 10

RAID уровня 53

В этом уровне сочетаются технологии RAID 0 и RAID 3, поэтому его правильнее было бы назвать RAID 30. В целом данная схема соответствует RAID 0, где роль отдельных дисков выполняют дисковые массивы, организованные по схеме RAID 3. Естественно, что в RAID 53 сочетаются все достоинства RAID 0 и RAID 3. Недостатки схемы такие же, что и у RAID 10.

Особенности реализации RAID-систем

Массивы RAID могут быть реализованы программно, аппаратно или как комбинация программных и аппаратных средств.

При программной реализации используются обычные дисковые контроллеры и стандартные команды ввода/вывода. Работа дисковых ЗУ в соответствии с алгоритмами различных уровней RAID обеспечивается программами операционной системы ВМ. Программный режим RAID предусмотрен, например, в Windows NT. Это дает возможность программного изменения уровня RAID, в зависимости от особенностей решаемой задачи. Хотя программный способ является наиболее дешевым, он не позволяет добиться высокого уровня производительности, характерного для аппаратурной реализации RAID.

Аппаратурная реализация RAID предполагает возложение всех или большей части функций по управлению массивом дисковых ЗУ на соответствующее оборудование, при этом возможны два подхода. Первый из них заключается в замене стандартных контроллеров дисковых ЗУ на специализированные, устанавливаемые на место стандартных. Базовая ВМ общается с контроллерами на уровне обычных команд ввода/вывода, а режим RAID обеспечивают контроллеры. Как и обычные, специализированные контроллеры/адаптеры ориентированы на определенный вид шины. Поскольку наиболее распространенной шиной для подключения дисковых ЗУ в настоящее время является шина SCSI, большинство производителей RAID-систем ориентируют свои изделия на протокол SCSI, определяемый стандартами ANSI X3.131 и ISO/IEC. При втором способе аппаратной реализации RAID-система выполняется как автономное устройство, объединяющее в одном корпусе массив дисков и контроллер. Контроллер содержит микропроцессор и работает под управлением собственной операционной системы, полиостью реализующей различные RAID-режимы. Такая подсисте-

ма подключается к шине базовой ВМ или к ее каналу ввода/вывода как обычное дисковое ЗУ. При аппаратной реализации RAID-систем обычно предусматривается возможность замены неисправных дисков без потери информации и без остановки работы. Кроме того, многие из таких систем позволяют разбивать отдельные диски на разделы, причем разные разделы дисков могут объединяться в соответствии с различными уровнями RAID.

МАГНИТНЫЕ ЛЕНТЫ

ЗУ на базе магнитных лент используются в основном для архивирования информации. Носителем служит тонкая полистироловая лента шириной от 0,38-2,54 см и толщиной около 0,025 мм, покрытая магнитным слоем. Лента наматывается на бобины различного диаметра. Данные записываются последовательно, байт за байтом, от начала ленты до ее конца. Время доступа к информации на магнитной ленте значительно больше, чем у ранее рассмотренных видов внешней памяти.

Обычно вдоль ленты располагается 9 дорожек, что позволяет записывать поперек ленты байт данных и бит паритета. Информация на ленте группируется в блоки — *записи.* Каждая запись отделяется от соседней *межблочным промежутком,* дающим возможность позиционирования головки считывания/записи на начало любого блока. Идентификация записи производится по полю заголовка, содержащемуся в каждой записи. Для указания начала и конца ленты используются физические маркеры в виде металлизированных полосок, наклеиваемых на магнитную ленту, или прозрачных участков на самой ленте. Известны также варианты маркирования начала и конца ленты путем записи на нее специальных кодов-индикаторов.

В универсальных ВМ обычно применяются бобинные устройства с вакуумными системами стабилизации скорости перемещения ленты. В них скорость перемещения ленты составляет около 300 см/с, плотность записи — 4 Кбайт/см, а скорость передачи информации — 320 Кбайт/с. Типовая бобина содержит 730 м магнитной ленты.

В ЗУ на базе картриджей используются кассеты с двумя катушками, аналогичные стандартным аудиокассетам. Типовая ширина ленты — 8 мм. Наиболее распространенной формой таких ЗУ является DAT (Digital Audio Tape). Данные на ленту заносятся по диагонали, как это принято в видеокассетах. По размеру такой картридж примерно вдвое меньше, чем обычная компакт-кассета, и имеет толщину 3,81 мм. Каждый картридж позволяет хранить несколько гигабайтов данных. Время доступа к данным невелико (среднее между временами доступа к дискетам и к жестким дискам). Скорость передачи информации выше, чем у дискет, но ниже, чем у жестких дисков.

Вторым видом ЗУ на базе картриджей является устройство стандарта DDS (Digital Data Storage). Этот стандарт был разработан в 1989 году для удовлетворения требований к резервному копированию информации с жестких дисков в мощных серверах и многопользовательских системах. В сущности, это вариант DAT, обеспечивающий хранение 2 Гбайт данных при длине ленты 90 м. В более позднем варианте стандарта DDS-DC (Digital Data Storage — Data Compression) за счет применения методов сжатия информации емкость ленты увеличена до 8 Гбайт. Наконец, третий вид ЗУ на базе картриджей также предназначен для резервного копирования содержимого жестких дисков, но при меньших объемах такой информации. Этот тип ЗУ отвечает стандарту QIC (Quarter Inch Cartridge tape) и более известен под названием *стример.* Известны стримеры, обеспечивающие хранение от 15 до 525 Мбайт информации. В зависимости от информационной емкости и фирмы-изготовителя изменяются и характеристики таких картриджей. Так, число. дорожек может варьироваться в диапазоне от 4 до 28, длина ленты — от 36 до 300 м и т. д.

ВОПРОСЫ САМОКОНТРОЛЯ

1. ХАРАКТЕРИСТИКИ СИСТЕМ ПАМЯТИ
2. ОСНОВНАЯ ПАМЯТЬ
3. ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА
4. ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА
5. ЭНЕРГОНЕЗАВИСИМЫЕ ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА
6. СПЕЦИАЛЬНЫЕ ТИПЫ ОПЕРАТИВНОЙ ПАМЯТИ
7. ОБНАРУЖЕНИЕ И ИСПРАВЛЕНИЕ ОШИБОК
8. СТЕКОВАЯ ПАМЯТЬ
9. АССОЦИАТИВНАЯ ПАМЯТЬ
10. КЭШ-ПАМЯТЬ
11. ЕМКОСТЬ КЭШ - ПАМЯТИ
12. ОДНОУРОВНЕВАЯ И МНОГОУРОВНЕВАЯ КЭШ – ПАМЯТЬ
13. ДИСКОВАЯ КЭШ-ПАМЯТЬ
14. ПОНЯТИЕ ВИРТУАЛЬНОЙ ПАМЯТИ
15. МАССИВЫ МАГНИТНЫХ ДИСКОВ С ИЗБЫТОЧНОСТЬЮ
16. МАГНИТНЫЕ ЛЕНТЫ

ЛЕКЦИЯ 5. СИСТЕМЫ ВВОДА/ВЫВОДА

Помимо центрального процессора (ЦП) и памяти, третьим ключевым элементом архитектуры ВМ является *система ввода/вывода* (СВВ). Система ввода/вывода призвана обеспечить обмен информацией между ядром ВМ и разнообразными внешними устройствами (ВУ). Технические и программные средства СВВ несут ответственность за физическое и логическое сопряжение *ядра вычислительной машины* и ВУ.

В процессе эволюции вычислительных машин системам ввода/вывода по сравнению с прочими элементами архитектуры уделялось несколько меньшее внимание. Косвенным подтверждением этого можно считать, например, то, что многие программы контроля производительности (бенчмарки) вообще не учитывают влияние операций ввода/вывода (В/ВЫВ) на эффективность ВМ. Следствием подобного отношения стал существенный разрыв в производительности процессора и памяти, с одной стороны, и скоростью ввода/вывода — с другой.

Технически система ввода/вывода в рамках ВМ реализуется комплексом *модулей ввода/вывода* (МВВ). Модуль ввода/вывода выполняет сопряжение ВУ с ядром ВМ и различные коммуникационные операции между ними. Две основные функции МВВ:

обеспечение интерфейса с ЦП и памятью *(«большой» интерфейс);*

обеспечение интерфейса с одним или несколькими периферийными устройствами *(«малый» интерфейс).*

Анализируя архитектуру известных ВМ, можно выделить три основных способа подключения СВВ к ядру процессора (рис. 8.1).

В варианте с раздельными шинами памяти и ввода/вывода (см. рис. 8.1, *а)* обмен информацией между ЦП и памятью физически отделен от ввода/вывода, поскольку обеспечивается полностью независимыми шинами. Это дает возможность осуществлять обращение к памяти одновременно с выполнением ввода/вывода. Кроме того, данный архитектурный вариант ВМ позволяет специализировать каждую из шин, учесть формат пересылаемых данных, особенности синхронизации обмена и т. п. В частности, шина ввода/вывода, с учетом характеристик реальных ВУ, может иметь меньшую пропускную способность, что позволяет снизить затраты на ее реализацию. Недостатком решения можно считать большое количество точек подключения к ЦП.

Рис. 8.1. Место системы ввода/вывода в архитектуре вычислительной машины:
а — раздельными шинами памяти и ввода/вывода;
б — с совместно используемыми линиями данных и адреса;
в — подключение на общих правах с процессором и памятью

Второй вариант — с совместно используемыми линиями данных и адреса (а рис. 8.1, *б).* Память и СВВ имеют общие для них линии адреса и линии данных разделяя их во времени. В то же время управление памятью и СВВ, а также синхронизация их взаимодействия с процессором осуществляются независимо по р дельным линиям управления. Это позволяет учесть особенности процедур обращения к памяти и к модулям ввода/вывода и добиться наибольшей эффективно доступа к ячейкам памяти и внешним устройствам.

Последний тип архитектуры ВМ предполагает подключение СВВ к системной шине на общих правах с процессором и памятью (см. рис. 8.1, *в*). Преимуществ недостатки такого подхода обсуждались при рассмотрении вопросов организации и (глава 4). Потенциально возможен также вариант подключения внешних устрой к системной шипе напрямую, без использования МВБ, но против него можно вынуть сразу несколько аргументов. Во-первых, в этом случае ЦП пришлось бы оснащать универсальными схемами для управления любым ВУ. При большом разнообразии внешних устройств, имеющих к тому же различные принципы действия, таз схемы оказываются чересчур сложными и избыточными. Во-вторых, пересылка данных при вводе и выводе происходит значительно медленнее, чем при обмене между ЦП и памятью, и было бы невыгодно задействовать для обмена информацией с ВУ высокоскоростную системную шину. И, наконец, в ВУ часто используются иные форматы данных и длина слова, чем в ВМ, к которым они подключены.

АДРЕСНОЕ ПРОСТРАНСТВО СИСТЕМЫ ВВОДА/ВЫВОДА

Как и обращение к памяти, операции ввода/вывода также предполагают наличие некоторой системы адресации, позволяющей выбрать один из модулей СВВ, а также одно из подключенных к нему внешних устройств. Адрес модуля и ВУ является составной частью соответствующей команды, в то время как расположение данных на внешнем устройстве определяется пересылаемой на ВУ информацией.

Адресное пространство ввода/вывода может быть совмещено с адресным пространством памяти или быть выделенным. При *совмещении адресного пространства* для адресации модулей ввода/вывода отводится определенная область адресов (рис. 8.2). Обычно все операции с модулем ввода/вывода осуществляются с использованием входящих в него внутренних регистров: управления, состояния, данных. Фактически процедура ввода/вывода сводится к записи информации в одни регистры МВВ и считыванию ее из других регистров. Это позволяет рассматривать регистры МВВ как ячейки основной памяти и работать с ними с помощью обычных команд обращения к памяти, при этом в системе команд ВМ вообще могут отсутствовать специальные команды ввода и вывода. Так, модификацию регистров МВВ можно производить непосредственно с помощью арифметических и логических команд. Адреса регистрам МВВ назначаются в области адресного пространства памяти, отведенной под систему ввода/вывода.

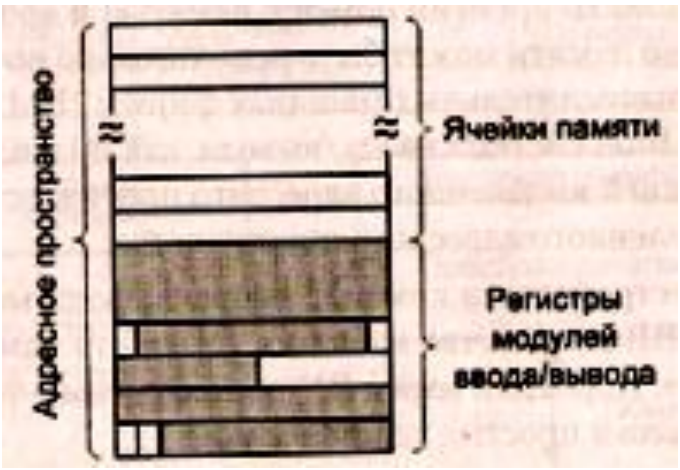


Рис. 8.2. Распределение совмещенного адресного пространства

Такой подход представляется вполне оправданным, если учесть, что ввод/вывод обычно составляет малую часть всех операций, выполняемых вычислительной машиной, чаще всего не более 1% от общего числа команд в программе.

Реализация концепции совмещенного адресного пространства в ВМ с кэш-памятью и виртуальной адресацией сопряжена с определенными проблемами. В частности, усложняется отображение виртуального адреса устройства ввода/вывода на физическое ВУ. Сложности также возникают и с кэшированием регистров МВВ.

Сформулируем преимущества и недостатки совмещенного адресного пространства.

Достоинства совмещенного адресного пространства:

- расширение набора команд для обращения к внешним устройствам, что позволяет сократить длину программы и повысить быстродействие;
- значительное увеличение количества подключаемых внешних устройств;
- возможность внепроцессорного обмена данными между внешними устройствами, если в системе команд есть команды пересылки между ячейками памяти; возможность обмена информацией не только с аккумулятором, но и с любым регистром центрального процессора.

Недостатки совмещенного адресного пространства:

- сокращение области адресного пространства памяти;
- усложнение декодирующих схем адресов в СВВ;
- трудности распознавания операций передачи информации при вводе/выводе среди других операций. Сложности в чтении и отладке программы, в которой простые команды вызывают выполнение сложных операций ввода/вывода;
- трудности при построении СВВ на простых модулях ввода/вывода: сигналя управления не смогут координировать сложную процедуру ввода/вывода. Поэтому МВВ часто должны генерировать дополнительные сигналы под управлением программы.

Совмещенное адресное пространство используется в вычислительных машинах MIPS и SPARC.

В случае *выделенного адресного пространства* для обращения к модулям ввода/вывода применяются специальные команды и отдельная система адресов. Это позволяет разделить шины для работы с памятью и шины ввода/вывода, что дает возможность совмещать во времени обмен с памятью и ввод/вывод. Кроме тог:., адресное пространство памяти может быть использовано по прямому назначению в полном объеме. В вычислительных машинах фирмы IBM и микро ЭВМ на базе процессоров фирмы Intel система ввода/вывода, как правило, организуется в соответствии с концепцией выделенного адресного пространства.

Достоинства выделенного адресного пространства:

- адрес внешнего устройства в команде ввода/вывода может быть коротким. В большинстве СВВ количество внешних устройств намного меньше количества ячеек памяти. Короткий адрес ВУ подразумевает такие же короткие команды ввода/вывода и простые дешифраторы;
- программы становятся более наглядными, так как операции ввода/вывода выполняются с помощью специальных команд;
- разработка СВВ может проводиться отдельно от разработки памяти, Недостатки выделенного адресного пространства: ввод/вывод производится только через аккумулятор центрального процессора. Для передачи информации от ВУ в РОН, если аккумулятор занят, требуется выполнение четырех команд (сохранение содержимого аккумулятора, ВЕ из ВУ, пересылка из аккумулятора в РОН, восстановление содержимого аккумулятора);
- перед обработкой содержимого ВУ это содержимое нужно переслать в ЦП.

ВНЕШНИЕ УСТРОЙСТВА

Связь ВМ с внешним миром осуществляется с помощью самых разнообразных, внешних устройств. Каждое ВУ подключается к МВВ посредством индивидуальной шины. Интерфейс, по которому организуется такое взаимодействие МВВ и ВУ, часто называют *малым.* Индивидуальная шина обеспечивает обмен данными и управляющими сигналами, а

также информацией о состоянии участников обмена. Внешнее устройство, подключенное к МВБ, обычно называют *периферийным устройством* (ПУ). Все множество ПУ можно свести к трем категориям [200]:

- для общения с пользователем;
- для общения с ВМ;
- для связи с удаленными устройствами.

Примерами первой группы служат видеотерминалы и принтеры. Ко второй группе причисляются внешние запоминающие устройства (магнитные и оптические диски, магнитные ленты и т.п.), датчики и исполнительные механизмы. Отметим двойственную роль внешних ЗУ, которые, с одной стороны, представляют собой часть памяти ВМ, а с другой — являются внешними устройствами. Наконец, устройства третьей категории позволяют ВМ обмениваться информацией с удаленными объектами, которые могут относиться к двум первым группам. В роли удаленных объектов могут выступать также другие ВМ.

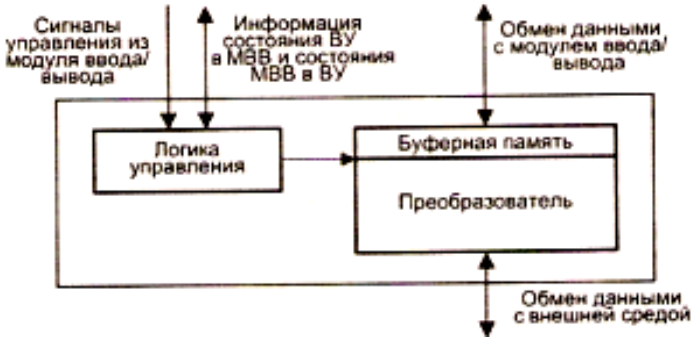


Рис. 8.3. Структура внешнего устройства

Обобщенная структура ВУ показана на рис. 8.3. Интерфейс с МВБ реализуется в виде сигналов управления, состояния и данных. *Данные* представлены совокупностью битов, которые должны быть переданы в модуль ввода/вывода или получены из него. *Сигналы управления* определяют функцию, которая должна быть выполнена внешним устройством. Это может быть стандартная для всех устройств функция — посылка данных в МВБ или получение данных из него, либо специфичная для данного типа ВУ функция, такая, например, как позиционирование головки магнитного диска или перемотка магнитной ленты. *Сигналы, состояния* характеризуют текущее состояние устройства, в частности включено ли ВУ и готово ли оно к передаче данных.

Логика управления — это схемы, координирующие работу ВУ в соответствии с направлением передачи данных. Задачей *преобразователя* является трансформация информационных сигналов, имеющих самую различную физическую природу, в электрические сигналы, а также" обратное преобразование. Обычно совместно с преобразователем используется *буферная память,* обеспечивающая временное хранение данных, пересылаемых между МВБ и ВУ.

МОДУЛИ ВВОДА/ВЫВОДА. ФУНКЦИИ МОДУЛЯ

Модуль ввода/вывода в составе вычислительной машины отвечает за управление одним или несколькими ВУ и за обмен данными между этими устройствами с одной стороны, и основной памятью или регистрами ЦП — с другой. Основные функции МВВ можно сформулировать следующим образом:

- локализация данных;
- управление и синхронизация;
- обмен информацией;
- буферизация данных;
- обнаружение ошибок.

Локализация данных

Под *локализацией данных* будем понимать возможность обращения к одному из ВУ, а также адресации данных на нем.

Адрес ВУ обычно содержится в адресной части команд ввода/вывода. Как уже отмечалось, в состав СВВ могут входить несколько модулей ввода/вывода. Каждому модулю назначается определенный диапазон адресов, независимо от того, является ли пространство адресов совмещенным или раздельным. Старшие разряды в адресах диапазона, выделенного модулю, обычно одинаковы и обеспечивают выбор одного из МВБ в рамках системы ввода/вывода. Младшие разряды адреса представляют собой уникальные адреса регистров данного модуля или подключенных к нему ВУ.

Одной из функций МВБ является проверка вхождения поступившего по шине адреса в выделенный данному модулю диапазон адресов. При положительном ответе модуль должен обеспечить дешифровку поступившего адреса и перенаправление информации к адресуемому объекту или от него.

Для простейших внешних устройств (клавиатура, принтер и т. п.) адрес ВУ однозначно определяет и расположение данных на этом устройстве. Для более сложных ВУ, таких как внешние запоминающие устройства, информация о местонахождении данных требует детализации. Так, для ЗУ на магнитной ленте необходимо указать номер записи, а для магнитного диска — номер цилиндра, номер сектора и т. п. Эта часть адресной информации передается в МВВ не по шине адреса, а в виде служебных сообщений, пересылаемых по шине данных. Обработка такой информации в модуле, естественно, сложнее, чем выбор нужного регистра или ВУ. В частности, она может требовать от МВВ организации процедуры поиска на носителе информации.

Управление и синхронизация

Функция управления и синхронизации заключается в том, что МВВ должен координировать перемещение данных между внутренними ресурсами ВМ и внешними устройствами. При разработке системы управления и синхронизации модуля ввода/вывода необходимо учитывать целый ряд факторов.

Прежде всего, нужно принимать во внимание, что ЦП может взаимодействовать одновременно с несколькими ВУ, причем быстродействие подключаемых к МВВ внешних устройств варьируется в очень широких пределах — от нескольких байтов в секунду в терминалах до десятков миллионов байтов в секунду при обмене с магнитными дисками. Если в системе используются шины, каждое взаимодействие между ЦП и МВВ включает в себя одну или несколько процедур арбитража.

В отличие от обмена с памятью процессы ввода/вывода и работа ЦП протекают не синхронно. Очередная порция информация может быть выдана на устройство вывода лишь тогда, когда это устройство готово их принять. Аналогично, ввод от устройства ввода допустим только в случае доступности информации на устройстве ввода. Несинхронный характер процессов ввода/вывода предполагает обмен сигналами, аналогичный процедуре «рукопожатия» (handshake), описанной в главе 4. Для двухпроводной системы синхронизации эта процедура состоит из четырех шагов, которые применительно к операции вывода можно описать следующим образом:

1. Центральный процессор с помощью сигнала ДД = 1 (данные достоверны) извещает о доступности данных, подлежащих выводу.

2. Приняв данные, устройство вывода сообщает процессору об их получении сигналом ДП = 1 (данные приняты).

3. Получив подтверждение, ЦП обнуляет сигнал ДД и снимает данные с шины, после чего может выставить на шину новые данные.

4. Обнаружив, что ДД = 0, устройство вывода, в свою очередь, устанавливает в нулевое состояние сигнал ДП, после чего оно готово для обработки принятых данных все время до получения очередного сигнала ДД = 1.

Описанную процедуру иллюстрирует рис. 8.4 (в скобках указаны номера шагов).

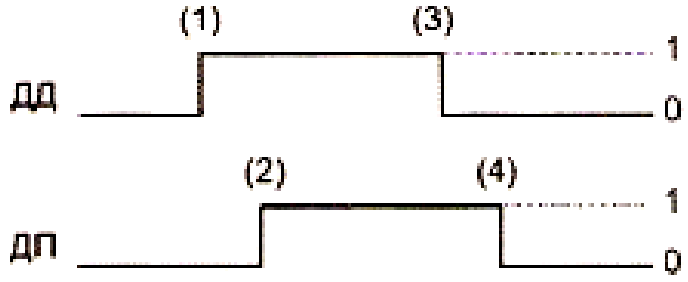


Рис. 8.4. Временная диаграмма процедуры «рукопожатия»

Таким образом, модуль ввода/вывода обязан снабдить центральный процессор информацией о собственной готовности к обмену, а также готовности подключенных к модулю ВУ. Помимо этого, процессор должен обладать оперативными сведениями и об иных происходящих в СВВ событиях.

Обмен информацией

Основной функцией МВВ является обеспечение обмена информацией. Со стороны «большого» интерфейса — это обмен с ЦП, а со стороны «малого» интерфейса — обмен с ВУ. В таком плане требования к МВВ непосредственно проистекают из типовой последовательности операций, выполняемых процессором при вводе/выводе.

1. Выбор требуемого внешнего устройства.
2. Определение состояния МВБ и ВУ.
3. Выдача указания модулю ввода/вывода на подключение нужного ВУ к процессору.
4. Получение от МВБ подтверждения о подключении затребованного ВУ к процессору.
5. Распознавание сигнала готовности устройства к передаче очередной порции информации.
6. Прием (передача) порции информации.
7. Циклическое повторение двух предшествующих пунктов до завершения передачи информации в полном объеме.
8. Логическое отсоединение ВУ от процессора.

С учетом описанной процедуры функция *обмена информацией с* ЦП включает в себя:

дешифровку команды: МВБ получает команды из ЦП в виде сигналов на шине управления;

пересылку данных между МВБ и ЦП по шине данных;

извещение о состоянии: из-за того, что БУ — медленные устройства, важно знать состояние модуля ввода/вывода. Так, в момент получения запроса на пересылку данных в центральный процессор МВВ может быть не готов выполнить эту пересылку, поскольку еще не завершил предыдущую команду. Этот факт должен быть сообщен процессору с помощью соответствующего сигнала. Возможны также сигналы, уведомляющие о возникших ошибках;

распознавание адреса; МВВ обязан распознавать адрес каждого ВУ, которым он управляет.

Наряду с обеспечением обмена с процессором МВВ должен выполнять функцию обмена информацией с ВУ. Такой обмен также включает в себя передачу данных, команд и информации о состоянии.

Буферизация

Важной задачей модуля ввода/вывода является буферизация данных, необходимость которой иллюстрирует табл. 8.1.

Таблица 8.1. Примеры устройств ввода/вывода, упорядоченные по режиму работы, субъекту и скорости передачи данных

Устройство	Режим работы	Партнер	Скорость передачи данных, Кбайт/с
Клавиатура	Ввод	Человек	0,01
Мышь	Ввод	Человек	0,02
Сканер	Ввод	Человек	200
Строчный принтер	Вывод	Человек	1
Лазерный принтер	Вывод	Человек	100
Графический дисплей	Вывод	Человек	30000
Локальная сеть	Ввод/вывод	ВМ	200
Гибкий диск	Память	ВМ	50
Оптический диск	Память	ВМ	500
Магнитный диск	Память	ВМ	2000

Несмотря на различия в скорости обмена информацией для разных ВУ, все они в этом плане значительно отстают от ЦП и памяти. Такое различие компенсируется за счет буферизации. При выводе информации на ВУ данные пересылаются из основной памяти в МВБ с большой скоростью. В модуле эти данные буферизируются и затем направляются в ВУ со скоростью, свойственной последнему. При вводе из ВУ данные буферизируются так, чтобы не заставлять память работать в режиме медленной передачи. Таким образом, МВБ должен обладать способностью работать как со скоростью памяти, так и со скоростью ПУ.

Обнаружение ошибок

Еще одной из важнейших функций МВБ является обнаружение ошибок, возникающих в процессе ввода/вывода. Центральный процессор следует оповещать о каждом случае обнаружения ошибки. Причинами возникновения последних бывают самые разнообразные факторы, которые в первом приближении можно свести к следующим группам:

- воздействие внешней среды;
- старение элементной базы;
- системное программное обеспечение;
- пользовательское программное обеспечение.

Из наиболее «активных» факторов окружения ВМ следует выделить:

- загрязнение и влагу;
- повышенную или пониженную температуру окружающей среды;
- электромагнитное облучение;
- скачки напряжения питания.

Степень влияния каждого из этих факторов зависит от типа и конструкции МВБ и ВУ. Так, к загрязнению наиболее чувствительны оптические и механические элементы ВУ, в то время как работа электронных компонентов СВВ в большей степени зависит от температуры внешней среды, электромагнитного воздействия и стабильности питающего напряжения.

Фактор старения характерен как для механических, так и для электронных элементов СВВ. В механических элементах он выражается в виде износа, следствием чего может быть неточное позиционирование головок считывания/записи на внешних запоминающих устройствах или неправильная подача бумаги в принтерах. Старение электронных элементов обычно выражается в изменении электрических параметров схем, приводящем к нарушению управления и синхронизации. Так, отклонения в параметрах электронных компонентов в состоянии вызвать недопустимый «перекос» сигналов, передаваемых между ЦП и МВВ или внутри МВВ.

Источником ошибок может стать и несовершенство системного программного обеспечения (ПО):

- непредвиденные последовательности команд или кодовые комбинации;
- некорректное распределение памяти;
- недостаточный размер буфера ввода/вывода;
- недостаточно продуманные и оттестированные комбинации системных модулей.

Среди ошибок, порождаемых пользовательским ПО, наиболее частыми являются:

- нарушение последовательности выполнения программы;
- некорректные процедуры.

Вероятность возникновения ошибки внутри процессора для современных ЦП оценивается величиной порядка 10^{-18}, в то время как для остальных составляющих ВМ она лежит в диапазоне $10^{-8} – 10^{-12}$.

МЕТОДЫ УПРАВЛЕНИЯ ВВОДОМ/ВЫВОДОМ

В ВМ находят применение три способа организации ввода/вывода (В/ВЫВ):

- программно управляемый ввод/вывод;
- ввод/вывод по прерываниям;
- прямой доступ к памяти.

При *программно управляемом вводе/выводе* все связанные с этим действия происходят по инициативе центрального процессора и под его полным контролем. ЦП 3 выполняет программу, которая обеспечивает прямое управление процессом ввода/вывода, включая проверку состояния устройства, выдачу команд ввода или вывода. Выдав в МВВ команду, центральный процессор должен ожидать завершения ее выполнения, и, поскольку ЦП работает быстрее, чем МВВ, это приводит к потере времени.

Ввод/вывод по прерываниям во многом совпадает с программно управляемым методом. Отличие состоит в том, что после выдачи команды ввода/вывода ЦП не должен циклически опрашивать МВВ для выяснения состояния устройства. Вместо этого процессор может продолжать выполнение других команд до тех пор, пока не получит запрос прерывания от МВВ, извещающий о завершении выполнения ранее выданной команды В/ВЫВ. Как и при программно управляемом В/ВЫВ, *ЦП* отвечает за извлечения данных из памяти (при выводе) и записи данных в память (при вводе).

Повышение как скорости В/ВЫВ, так и эффективности использования ЦП обеспечивает третий способ В/ВЫВ — *прямой 'доступ* к *памяти* (ПДП). В этом режиме основная память и модуль ввода/вывода обмениваются информацией напрямую, минуя процессор.

ПРОГРАММНО УПРАВЛЯЕМЫЙ ВВОД/ВЫВОД

Наиболее простым методом управления вводом/выводом является *программно управляемый ввод/вывод,* часто называемый также *вводом /выводом с опросом.* Здесь ввод/вывод происходит под полным контролем центрального процессора и реализуется специальной процедурой ввода/вывода. В этой процедуре ЦП с помощью команды ввода/вывода сообщает модулю ввода/вывода, а через него и внешнему устройству о предстоящей операции. Адрес модуля и ВУ, к которому производится обращение, указывается в адресной части команды ввода или вывода. Модуль исполняет затребованное действие, после чего устанавливает в единицу соответствующий бит в своем регистре состояния. Ничего другого, чтобы уведомить ЦП, модуль не предпринимает. Следовательно, для определения момента завершения операции или пересылки очередного элемента блока данных процессор должен периодически опрашивать и анализировать содержимое регистра состояния МВБ.

Иллюстрация процедуры программно управляемого ввода блока данных с устройства ввода приведена на рис. 8.6. Данные читаются пословно. Для каждого читаемого слова ЦП должен оставаться в цикле проверки, пока не определит, что слово находится в регистре данных МВБ, то есть доступно для считывания.

Процедура начинается с выдачи процессором команды ввода, в которой указан адрес конкретного МВБ и конкретного ВУ. Существуют четыре типа команд В/ВЫВ, которые может получить МВБ: управление, проверка, чтение и запись.

Команды управления используются для активизации ВУ и указания требуемой операции. Например, в устройство памяти на магнитной ленте может быть выдана команда перемотки или продвижения на одну запись. Для каждого типа ВУ характерны специфичные для него команды управления.

Команда проверки применяется для проверки различных ситуаций, возникающих в МВБ и ВУ в процессе ввода/вывода. С помощью таких команд ЦП способен выяснить, включено ли ВУ, готово ли оно к работе, завершена ли последняя операция ввода/вывода и не возникли ли в ходе ее выполнения какие-либо ошибки. Действие команды сводится к установке или сбросу соответствующих разрядов регистра состояния МВБ.

Команда чтения побуждает модуль получить элемент данных из ВУ и занести его в регистр данных (РД). ЦП может получить этот элемент данных, запросив МВБ поместить его на шину данных.

Команда записи заставляет модуль принять элемент данных (байт или слово) с шины данных и переслать его в РД с последующей передачей в ВУ.

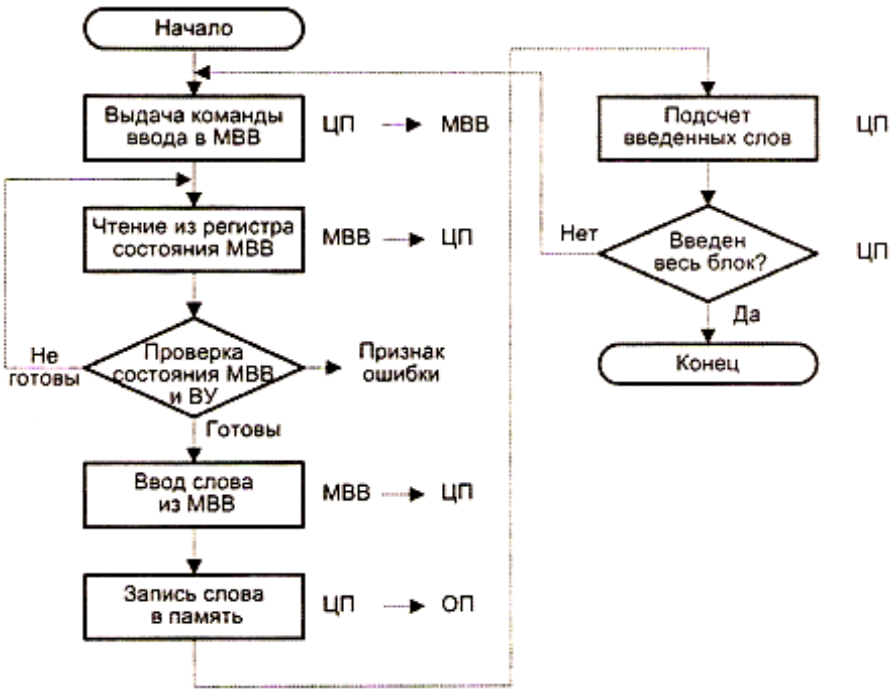


Рис. 8.6. Программно управляемый ввод данных

Если к МВБ подключено несколько ВУ, то в процедуре ввода/вывода нужно производить циклический опрос всех устройств, с которыми в данный момент up-. -изводятся операции В/ВЫВ.

Из блок-схемы (см. рис. 8.6) явно виден основной недостаток программно управляемого В/ВЫВ — неэффективное использование процессора из-за ожидания готовности очередной порции информации, в течение которого никаких иных полезных действий ЦП не выполняет. Кроме того, пересылка даже одного слова требует выполнения нескольких команд. ЦП должен тратить время на анализ битов состояния МВБ, запись в МВБ битов управления, чтение или запись данных со скоростью, определяемой внешним устройством. Все это также отрицательно сказывается на эффективности ввода/вывода.

Главным аргументом в пользу программно управляемого ввода/вывода является простота МВБ, поскольку основные функции по управлению В/ВЫВ берет на себя процессор. При одновременной работе с несколькими ВУ приоритет устройств легко изменить программными средствами (последовательностью опроса). Наконец, подключение к СВВ новых внешних устройств или отключение ранее подключенных также реализуется без особых сложностей.

ВВОД/ВЫВОД ПО ПРЕРЫВАНИЯМ

Как уже отмечалось, основным недостатком программно управляемого В/ВЫВ являются простои процессора в ожидании, пока модуль ввода/вывода выполнит очередную операцию. Альтернативой может быть вариант, когда ЦП выдает команду В/ВЫВ, а затем продолжает делать другую полезную работу. Когда ВУ готово к обмену данными, оно через МВВ извещает об этом процессор с помощью запроса на прерывание. ЦП осуществляет передачу очередного элемента данных, после чего возобновляет выполнение прерванной программы.

Обсудим процесс ввода блока данных е использованием В/ВЫВ по прерываниям (рис. 8.7). Оставим без внимания такие подробности, как сохранение и восстановления контекста, действия, выполняемые при завершении пересылки блока данных, а также в случае возникновения ошибок.

Процедура ввода блока данных по прерываниям реализуется следующим образом. ЦП выдает команду чтения, а затем продолжает выполнение других заданий, например другой программы. Получив команду, МВВ приступает к вводу элемента данных с ВУ. Когда считанное слово оказывается в регистре данных модуля, VIBB формирует на управляющей линии сигнал прерывания ЦП. Выставив запрос, МВВ помещает введенную информацию на шину данных, после чего он готов к следующей операции В/ВЫВ. ЦП в конце каждого цикла команды проверяет наличие запросов прерывания. Когда от МВВ приходит такой сигнал, ЦП сохраняет контекст текущей программы и обрабатывает прерывание. В рассматриваемом случае ЦП читает слово из модуля, записывает его в память и выдает модулю команду на считывание очередного слова. Далее ЦП восстанавливает контекст прерванной программы и возобновляет ее выполнение.

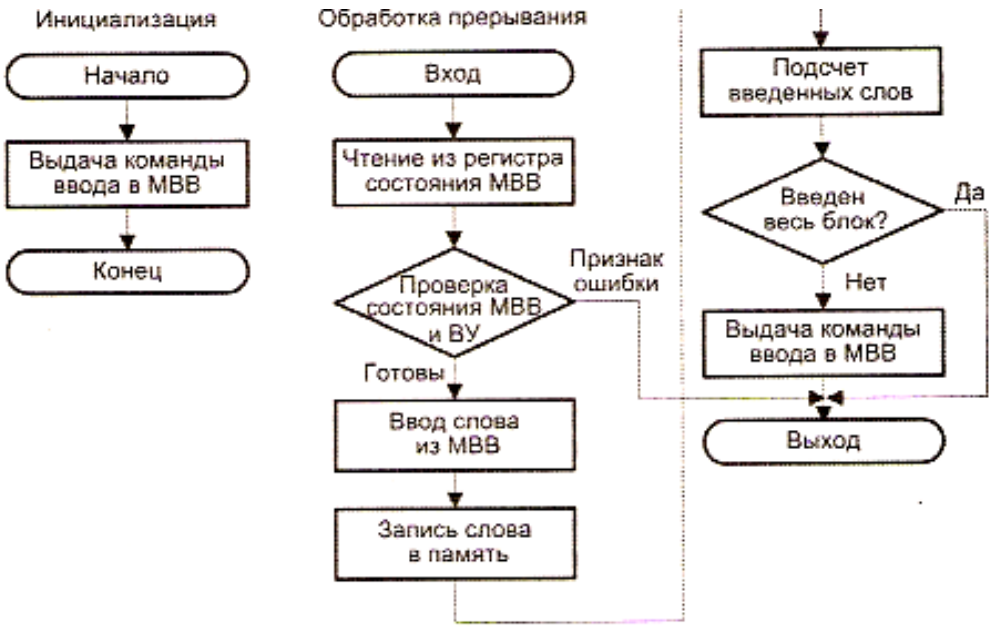


Рис. 8.7. Ввод данных по прерыванию

Этот метод эффективнее программно управляемого В/ВЫВ, поскольку устраняет ненужные ожидания, однако обработка прерывания занимает достаточно много времени ЦП. Кроме того, каждое слово, пересылаемое из памяти в модуль В/ВЫВ или в противоположном направлении, как и при программно управляемом В/ВЫВ, проходит через ЦП.

Реализация ввода/вывода по прерываниям

При реализации ввода/вывода по прерываниям необходимо дать ответы на два вопроса. Во-первых, определить, каким образом ЦП может выяснить, какой из МВВ и какое из подключенных к этому модулю внешних устройств выставили запрос. Во-вторых, при мно-

жественных прерываниях требуется решить, какое из них должно быть обслужено в первую очередь.

Сначала рассмотрим вопрос идентификации устройства. Здесь возможны три основных метода:

- множественные линии прерывания;
- программная идентификация;
- векторное прерывание.

Наиболее простой подход к решению проблемы определения источника запроса — применение *множественных линий прерывания* между ЦП и модулями ввода/вывода, хотя выделение слишком большого количества управляющих линий для этих целей нерационально. Более того, даже если присутствует несколько линий прерывания, желательно, чтобы каждая линия использовалась всеми МВБ. при этом для каждой линии действует один из двух остальных методов идентификации устройства.

При *программной идентификации,* обнаружив запрос прерывания, ЦП переходит к общей программе обработки прерывания, задачей которой является опрос всех МВБ с целью определения источника запроса. Для этого может быть выделена специальная командная линия опроса. ЦП помещает на адресную шину адрес опрашиваемого МВВ и формирует на этой линии сигнал опроса. Реакция модуля зависит от того, выставлял он запрос или нет. Возможен и иной вариант, когда каждый МВВ включает в себя адресуемый регистр состояния. Тогда ЦП считывает содержимое РС каждого модуля, после чего выясняет источник прерывания. Когда источник прерывания установлен, ЦП переходит к программе обработки прерывания, соответствующей этому источнику. Недостаток метода программной идентификации заключается в больших временных потерях.

Наиболее эффективную процедуру идентификации источника прерывания обеспечивают аппаратурные методы, в основе которых лежит идея *векторного прерывания.* В этом случае, получив подтверждение прерывания от процессора, выставившее запрос устройство выдает на шину данных специальное слово, называемое *вектором прерывания.* Слово содержит либо адрес МВБ, либо какой-нибудь другой уникальный идентификатор, который ЦП интерпретирует как указатель на соответствующую программу обработки прерывания. Такой подход устраняет необходимость в предварительных действиях с целью определения источника запроса прерывания. Реализуется он с помощью хранящейся в ОП *таблицы векторов прерывания* (рис. 8,8), где содержатся адреса программ обработки прерываний. Входом в таблицу служит вектор прерывания. Начальный адрес таблицы (база) обычно задается неявно, то есть под таблицу отводится вполне определенная область памяти.

Наиболее распространены два варианта векторной идентификации источника запроса прерывания: цепочечный опрос и арбитраж шины.

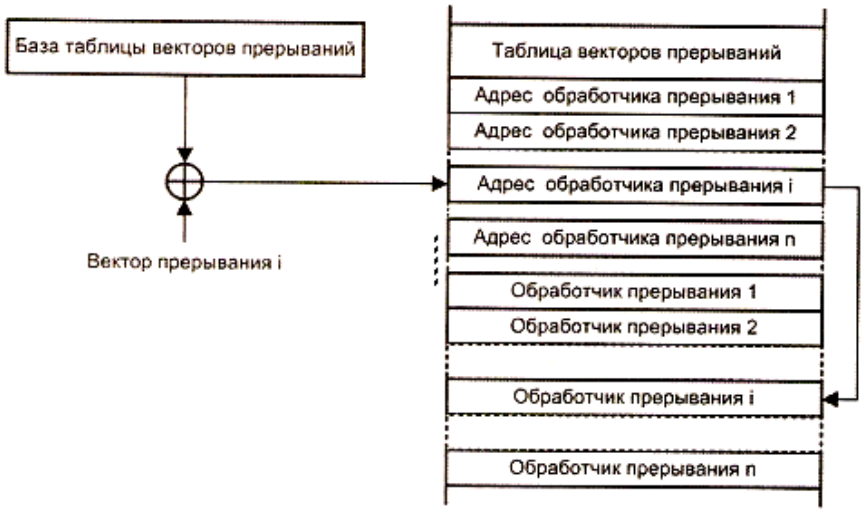


Рис. 8.8. Идентификация запроса с помощью вектора прерывания

При *цепочечном методе* для передачи запроса прерывания модули ввода/вывода совместно используют одну общую линию. Линия подтверждения прерывания последовательно проходит через все МВВ. Когда ЦП обнаруживает запрос прерывания, он посылает сигнал по линии подтверждения прерывания. Этот сигнал движется через цепочку модулей, пока не достигнет того, который выставил запрос. Запросивший модуль реагирует путем выдачи на шину данных своего вектора прерывания.

В варианте *арбитража шипы* МВБ, прежде чем выставить запрос на линии запроса прерывания, должен получить право на управление шиной. Таким образом, в каждый момент времени активизировать линию запроса прерывания может только один из модулей. Когда ЦП обнаруживает прерывание, он отвечает по линии подтверждения. После этого запросивший модуль помещает на шину данных свой вектор прерывания.

Перечисленные методы служат не только для идентификации запросившего МВВ, но и для назначения приоритетов, когда прерывание запрашивают несколько устройств. При множественных линиях запроса ЦП начинает с линии, имеющей наивысший приоритет. В варианте программной идентификации приоритет модулей определяется очередностью их проверки. Для цепочечного метода приоритет модулей определяется порядком их следования в цепочке.

ПРЯМОЙ ДОСТУП К ПАМЯТИ

Хотя ввод/вывод по прерываниям эффективнее программно управляемого, оба этих метода страдают двумя недостатками:

- темп передачи при вводе/выводе ограничен скоростью, с которой ЦП в состоянии опросить и обслужить устройство;
- ЦП вовлечен в управление передачей, для каждой пересылки он должен выполнить определенное количество команд.

Когда пересылаются большие объемы данных, требуется более эффективны;: способ ввода/вывода— *прямой доступ к памяти* (ПДП). ПДП предполагает наличие на системной шине дополнительного модуля — *контроллера прямого доступа к памяти* (КПДП), способного брать на себя функции ЦП по управлению системной шиной и обеспечивать прямую пересылку информации между ОП и ВУ. без участия центрального процессора. В сущности, КПДП — это и есть модуль ввода/вывода, реализующий режим прямого доступа к памяти.

Если ЦП желает прочитать или записать блок данных, он прежде всего должен поместить в КПДП (рис. 8.9) информацию, характеризующую предстоящее действие. Этот процесс называется инициализацией КПДП и включает в себя занесение в контроллер следующих четырех параметров:

вида запроса (чтение или запись);

адреса устройства в в од а/вы во да;

адреса начальной ячейки блока памяти, откуда будет извлекаться или куда будет вводиться информация;

количества слов, подлежащих чтению или записи.

Первый параметр определяет направление пересылки данных: из ОП в ВУ или наоборот. За исходную точку обычно принимается память, поэтому под чтением понимают считывание данных из ОП и выдачу их на устройство вывода, а под записью — прием данных из устройства ввода и запись в ОП. Вид запроса запоминается в схеме логики управления контроллера.

К КПДП обычно могут быть подключены несколько ВУ, а адрес УВВ конкретизирует, какое из них должно участвовать в предстоящем обмене данными. Этот адрес запоминается в логике управления КПДП.

Третий параметр — адрес начальной ячейки — хранится в регистре адреса (РА) контроллера. После передачи каждого слова содержимое РА автоматически увеличивается на единицу, то есть в нем формируется адрес следующей ячейки ОП.

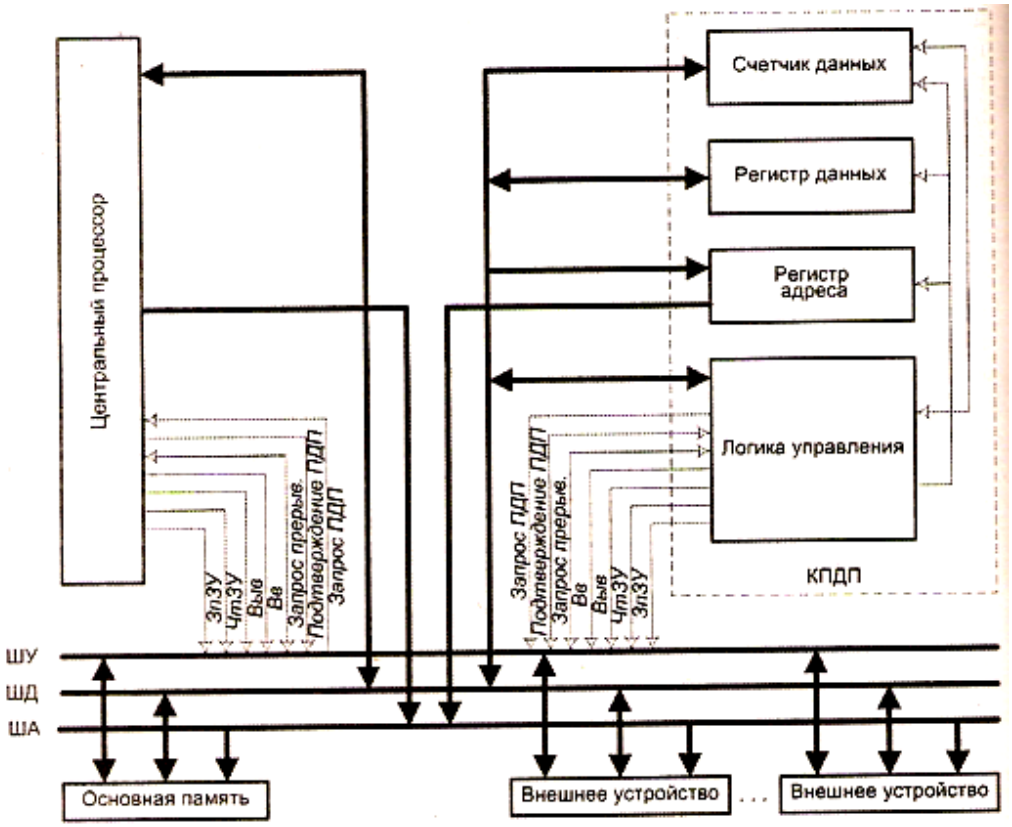


Рис. 8.9. Организация прямого доступа к памяти

Размер блока в словах заносится в счетчик данных (СД) контроллера. После передачи каждого слова содержимое СД автоматически уменьшается на единицу. Нулевое состояние СД свидетельствует о том, что пересылка блока данных завершена.

После инициализации процесс пересылки информации может быть начат в любой момент. Инициаторами обмена вправе выступать как ЦП, так и ВУ. Устройство, желающее начать В/ВЫВ, извещает об этом контроллер подачей соответствующего сигнала. Получив такой сигнал, КПДП выдает в ЦП сигнал «Запрос ПДП». В ответ ЦП освобождает шины адреса и данных, а также тс линии шины управления, по которым передаются сигналы, управляющие операциями на шине адреса (ША) и шине данных (ШД). К таким, прежде всего, относятся линии ЧтЗУ, ЗпЗУ, Выв, Вв и линия выдачи адреса на 111 А. Далее ЦП отвечает контроллеру сигналом «Подтверждение ПДП», который для последнего означает, что ему делегированы права на управление системной шиной и можно приступать к пересылке данных.

Процесс пересылки каждого слова блока состоит из двух этапов.

При выполнении операции чтения (ОП → ВУ) на первом этапе КПДП выставляет на шину адреса содержимое РА (адрес текущей ячейки ОП) и формирует сигнал ЧтЗУ. Считанное из ячейки ОП слово помещается на шину данных. На втором этапе КПДП выставляет на ША адрес устройства вывода и формирует сигнал Выв, который обеспечивает передачу слова с шины данных в ВУ.

При выполнении операции записи (ВУ -> ОП) КПДП сначала выдает на шину данных адрес устройства ввода и формирует сигнал Вв, по которому введенные данные поступают на шину данных. На втором этапе КПДП помещает на ША адрес ячейки ОП, куда должны быть занесены данные, и выдает сигнал ЗпЗУ. Этим сигналом информация с ШД записывается в ячейку ОП.

Как при чтении, так и при записи происходит буферизация пересылаемого слова в регистре данных (РД) контроллера. Это необходимо для компенсации различий в скорости работы ОП и ВУ, в силу чего сигналы Выв и Вв формируются контроллером лишь при получении от ВУ подтверждения о готовности. Буферизация сводится к тому, что после первого

этапа слово с ШД заносится в РД, а перед вторым — возвращается из РД на шину данных.

После пересылки каждого слова логика управления прибавляет единицу к содержимому РА (формирует адрес следующей ячейки ОП) и уменьшает на единицу содержимое СД (ведет подсчет переданных слов).

Когда пересылка завершена (при нулевом значении в СД), КПДП снимает сигнал «Запрос ПДП», в ответ на что ЦП снимает сигнал «Подтверждение ПДП» и вновь берет на себя управление системной шиной, то есть ЦП вовлечен в процесс ввода/вывода только в начале и конце передачи.

Эффективность ПДП зависит от того, каким образом реализовано распределение системной шины между ЦП и КПДП в процессе пересылки блока. Здесь может применяться один из трех режимов:

- блочная пересылка;
- пропуск цикла;
- прозрачный режим.

При *блочной пересылке* КПДП полностью захватывает системную шину с момента начала пересылки и до момента завершения передачи всего блока. На весь этот период ЦП не имеет доступа к шине.

В режиме *пропуска цикла* КПДП после передачи каждого слова на один цикл шины освобождает системную шину, предоставляя ее на это время процессору. Поскольку КПДП все равно должен ждать готовности ПУ, это позволяет ЦП эффективно распорядиться данным обстоятельством.

В *прозрачном режиме* КПДП имеет доступ к системной шине только в тех циклах, когда ЦП в ней не нуждается. Это обеспечивает наиболее эффективную работу процессора, но может существенно замедлять операцию пересылки блока данных. Здесь многое зависит от решаемой задачи, поскольку именно она определяет интенсивность использования шины процессором.

В отличие от обычного прерывания в пределах цикла команды имеется несколько точек, где КПДП вправе захватить шину (рис 8.10). Отметим, что это не прерывание: процессору не нужно запоминать контекст задачи.

Рис. 8.10. Точки возможного вмешательства в цикл команды при прямом доступе к памяти и при обычном прерывании

Механизм ПДП может быть реализован различными путями. Некоторые возможности показаны на рис. 8.11.

В первом примере (см. рис. 8.11, а) все ВУ совместно используют общую системную шину. КПДП работает как заменитель ЦП и обмен данными между памятью и ВУ через КПДП производит через программно управляемый ввод/вывод. Хотя этот вариант может быть достаточно дешевым, эффективность его невысока. Как и в случае программно управляемого ввода/вывода, осуществляемого процессором, каждая пересылка требует двух циклов шины.

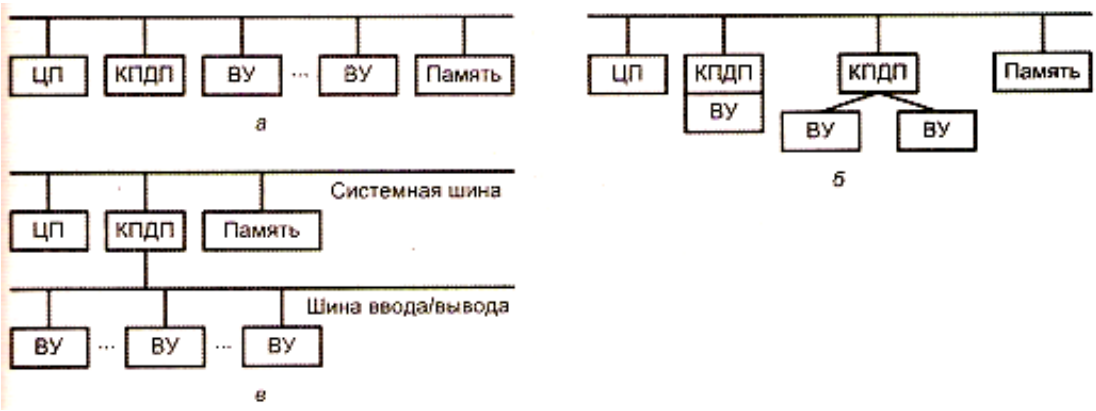


Рис. 8.11. Возможные конфигурации систем прямого доступа к памяти

Число необходимых циклов шины может быть уменьшено при объединении функций КПДП и ВУ. Как видно из рис. 8.11, *б,* это означает, что между КПДП и одним или несколькими ВУ есть другой тракт, не включающий системную шину. Логика ПДП может быть частью ВУ, либо это может быть отдельный КПДП, управляющий одним или несколькими внешними устройствами. Допустим и еще один шаг в том же направлении (см. рис. 8.11, в) — соединение КПДП с ВУ посредством шины ввода/вывода. Это уменьшает число интерфейсов В/ВЫВ в КПДП и делает такую конфигурацию легко расширяемой. В двух последних вариантах системная шина задействуется КПДП только для обмена данными е памятью. Обмен данными между КПДП и ВУ реализуется минуя системную шину.

КАНАЛЫ И ПРОЦЕССОРЫ ВВОДА/ВЫВОДА

По мере развития систем В/ВЫВ их функции усложняются. Главная цель такого усложнения — максимальное высвобождение ЦП от управления процессами ввода/вывода. Некоторые пути решения этой задачи уже были рассмотрены. Следующими шагами в преодолении проблемы могут быть:

1. Расширение возможностей МВБ и предоставление ему прав процессора со специализированным набором команд, ориентированных на операции ввода/ вывода. ЦП дает указание такому процессору В/ВЫВ выполнить хранящуюся в памяти ВМ программу ввода/вывода. Процессор В/ВЫВ извлекает и исполняет команды этой программы без участия центрального процессора и прерывает ЦП только после завершения всей программы ввода/вывода.

2. Рассмотренному в пункте 1 процессору ввода/вывода придается собственная локальная память, при этом возможно управление множеством устройств В/ ВЫВ с минимальным привлечением ЦП.

В первом случае МВБ называют *каналом ввода/вывода* (КВВ), а во втором — *процессором ввода/вывода.* В принципе различие между каналом и процессором ввода/вывода достаточно условно, поэтому в дальнейшем будем пользоваться термином «канал».

Концепция системы ввода/вывода с КВВ характерна для больших универсальных вычислительных машин (мэйнфреймов), где проблема эффективной организации В/ВЫВ и максимального высвобождения центрального процессора в пользу его основной функции стоит наиболее остро. СВВ с каналами ввода/вывода была предложена и реализована в ВМ семейства IBM 360 и получила дальнейшее развитие в семействах IBM 370 и IBM 390.

В ВМ с каналами ввода/вывода центральный процессор практически не участвует в непосредственном управлении внешними устройствами, делегируя эту задачу специализированному процессору, входящему в состав КВВ. Все функции ЦП сводятся к запуску и остановке операций в КВВ, а также проверке состояния канала и подключенных к нему ВУ. Для этих целей ЦП использует лишь несколько (от 4 до 7) команд ввода/вывода. Например, в IBM 360 таких команд четыре:

- «Начать ввод/вывод»;
- «Остановить ввод/вывод»;

- «Проверить ввод/вывод»;
- «Проверить канал».

КВВ реализует операции В/ВЫВ путем выполнения так называемой *канальной программы.* Канальные программы для каждого ВУ, с которым предполагается обмен информацией, описывают нужную последовательность операций ввода/ вывода и хранятся в основной памяти ВМ. Роль команд в канальных программах выполняют *управляющие слова канала* (УСК), структура которых отличается от структуры обычной машинной команды. Типовое УСК содержит:

- *код операции,* определяющий для КВВ и ВУ тип операции: «Записать» (вывод информации из ОП в ВУ), «Прочитать» (ввод информации из ВУ в ОП), «Управление» (перемещение головок НМД, магнитной ленты и т, п.);
- *указатели* — дополнительные предписания, задающие более сложную последовательность операций В/ВЫВ, например при вводе пропускать отдельные записи или наоборот — с помощью одной команды вводить «разбросанный» по ОП массив как единый;
- *адрес данных,* указывающий область памяти, используемую в операции ввода/вывода;
- *счетчик данных,* хранящий значение длины передаваемого блока данных.

Кроме того, в УСК может содержаться идентификатор ВУ и информация о его уровне приоритета, указания по действиям, которые должны быть произведены при возникновении ошибок и т.п.

Центральный процессор инициирует ввод/вывод путем инструктирования канала о необходимости выполнить канальную программу, находящуюся в ОП. и указания начального адреса этой программы в памяти ВМ. КВВ следует этим указаниям и управляет пересылкой данных. Отметим, что пересылка информации каналом ведется в режиме прямого доступа к памяти. ВУ взаимодействуют с каналом, получая от него приказы. Таким образом, в ВМ с КВВ управление вводом/выводом строится иерархическим образом. В операциях ввода/вывода участвуют три типа устройств:

- процессор (первый уровень управления);
- канал ввода/вывода (второй уровень);
- внешнее устройство (третий уровень).

Каждому типу устройств соответствует свой вид управляющей информации:

- процессору — команды ввода/вывода;
- каналу — управляющие слова канала;
- периферийному устройству — приказы.

Структура ВМ с канальной системой ввода/вывода показана на рис. 8.12.

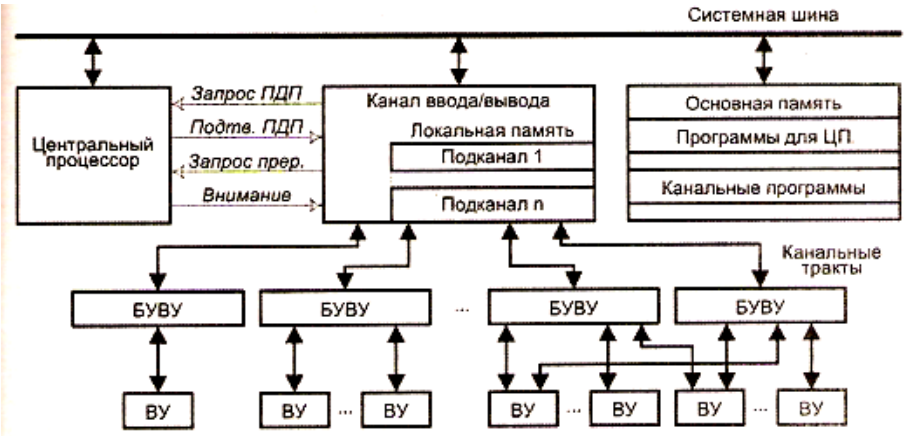


Рис. 8.12. ВМ с канальной системой ввода/вывода

Обмен информацией между КВВ и основной памятью осуществляется посредством системной шины ВМ. ВУ подключаются к каналу не непосредственно, а через блоки управ-

ления внешними устройствами (БУВУ). БУВУ принимает от канала приказы по управлению внешним устройством (чтение, запись, перемещение носителя или магнитной головки и т. п.) и преобразует их в сигналы управления, свойственные данному типу ВУ. Обычно один БУВУ может обслуживать несколько однотипных ВУ, но для подключения быстродействующих внешних устройств часто применяются индивидуальные блоки управления. В свою очередь, некоторые ВУ могут подключаться одновременно к нескольким БУВУ. Это позволяет воспользоваться свободным трактом другого БУВУ при занятости данного БУВУ обслуживанием одного из подключенных к нему ПУ. Физически БУВУ может быть самостоятельным устройством или интегрирован с ВУ или каналом.

Обмен информацией между БУВУ и КВВ обеспечивается так называемыми *канальными трактами.* Обычно каждое БУВУ связано с одним из канальных трактов, но возможно также подключение блока управления сразу к нескольким трактам, что дает возможность избежать нежелательных задержек при занятости одного из них.

Обмен информацией между ВУ и ОП, как уже упоминалось, реализуется в режиме прямого доступа к памяти, при этом для взаимодействия ЦП и канала задействованы сигналы «Запрос ПДП» и «Подтверждение ПДП».

Чтобы известить ЦП об окончании текущей канальной программы или об ошибках, возникших при ее выполнении, КВВ выдает в ЦП сигнал «Запрос прерывания». В свою очередь, ЦП может привлечь внимание канала сигналом «Внимание».

Способ организации взаимодействия ВУ с каналом определяется соотношением быстродействия ОП и ВУ. По этому признаку ВУ образуют две группы: быстродействующие (накопители па магнитных дисках (НМД), накопители на магнитных лентах (НМЛ)) со скоростью приема и выдачи информации около 1 Мбайт/с и медленнодействующие (дисплеи, печатающие устройства и др.) со скоростями порядка 1 Кбайт/с и менее. Быстродействие основной памяти обычно значительно выше. С учетом производительности ВУ в КВВ реализуются два режима работы: *мультиплексный* (режим разделения времени) и *монопольный.*

В *мультиплексном режилге* несколько внешних устройств разделяют канал во времени, при этом каждое из параллельно работающих с каналом ВУ связывается с КВВ на короткие промежутки времени только после того, как ВУ будет готово к приему или выдаче очередной порции информации (байта, группы байтов и т.д.). Такая схема принята в *мультиплексном канале ввода/вывода.* Если в течение сеанса связи пересылается один байт или несколько байтов, образующих одно машинное слово,'канал называется *байт-мультиплексным.* Канал, в котором в пределах сеанса связи пересылка данных выполняется поблочно, носит название *блок-мультиплексного.*

В *монопольном режиме* после установления связи между каналом и ВУ последнее монополизирует канал на все время до завершения инициированной процессором канальной программы и всех предусмотренных этой программой пересылок данных между ВУ и ОП. На все время выполнения канальной программы канал оказывается недоступным для других ВУ. Данную процедуру обеспечивает *селекторный канал ввода/вывода.* Отметим, что в блок-мультиплексном канале в рамках сеанса связи пересылка блока осуществляется в монопольном режиме.

ВОПРОСЫ САМОКОНТРОЛЯ

1. АДРЕСНОЕ ПРОСТРАНСТВО СИСТЕМЫ ВВОДА/ВЫВОДА
2. ВНЕШНИЕ УСТРОЙСТВА
3. МОДУЛИ ВВОДА/ВЫВОДА
4. ФУНКЦИИ МОДУЛЯ
5. МЕТОДЫ УПРАВЛЕНИЯ ВВОДОМ/ВЫВОДОМ
6. ПРОГРАММНО УПРАВЛЯЕМЫЙ ВВОД/ВЫВОД
7. ВВОД/ВЫВОД ПО ПРЕРЫВАНИЯМ
8. ПРЯМОЙ ДОСТУП К ПАМЯТИ
9. КАНАЛЫ И ПРОЦЕССОРЫ ВВОДА/ВЫВОДА

Литература

1. Архитектуры и топологии многопроцессорных вычислительных систем [Электронный ресурс] / А.В. Богданов [и др.]. — Электрон. текстовые данные. М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. 135 с. — 5-9556-0018-3. — Режим доступа: http://www.iprbookshop.ru/52189.html

2. Гуров В.В., Чуканов В.О. Архитектура и организация ЭВМ [Электронный ресурс].— Электрон. текстовые данные. М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. 183 с. 5-9556-0040-X. — Режим доступа: http://www.iprbookshop.ru/73706.html

3. Гуров В.В. Архитектура микропроцессоров [Электронный ресурс]. — Электрон. текстовые данные. М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. 115 с. — 978-5-9963-0267-3. — Режим доступа: http://www.iprbookshop.ru/56313.html

4. Гуров В.В., Чуканов В.О. Логические и арифметические основы и принципы работы ЭВМ [Электронный ресурс]. — Электрон. текстовые данные. М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. 166 с. — 5-9556-0040-X. — Режим доступа: http://www.iprbookshop.ru/73683.html

5. Лукьянов Б.В., Лукьянов П.Б. Архитектура предприятия [Электронный ресурс] : учебное пособие. - Электрон. текстовые данные. М. : Русайнс, 2015. 134 с. — 978-5-4365-0465-0. — Режим доступа: http://www.iprbookshop.ru/48872.html

6. Шаманов А.П. Системы счисления и представление чисел в ЭВМ [Электронный ресурс] : учебное пособие. — Электрон. текстовые данные. — Екатеринбург: Уральский федеральный университет, 2016. 52 с. — 978-5-7996-1719-6. — Режим доступа: http://www.iprbookshop.ru/66204.html

Учебное пособие

Живодеров А. Н.

Основы архитектуры, устройство и функционирование вычислительных систем

КУРС ЛЕКЦИЙ

для специальности 09.02.04 Информационные системы (по отраслям)

Редактор Лебедева Е.М.

Подписано к печати 17.02.2020 г. Формат 60x84 $^{1}/_{16}$.
Бумага офсетная. Усл. п. л. 4,82. Тираж 25 экз. Изд. № 6633.

Издательство Брянского государственного аграрного университета
243365 Брянская обл., Выгоничский район, с. Кокино, Брянский ГАУ