МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

В. А. КИРЬЯНЧИКОВ

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ. ИЕРАРХИЧЕСКАЯ СИСТЕМА ПАМЯТИ. ОРГАНИЗАЦИЯ ШИН. СИСТЕМА ВВОДА-ВЫВОДА

Учебное пособие

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»
2022

УДК 004.2(07+004.39(07))
ББК 3.973.2-02я7+3.973.2-04я7
К43

Кирьянчиков В. А.

К43 Организация ЭВМ и систем. Иерархическая система памяти. Организация шин. Система ввода-вывода: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2022. 94 с.

ISBN 978-5-7629-3047-5

Содержит основные сведения об иерархической организации системы памяти. Рассмотрены вопросы организации системы шин и построения системы ввода-вывода.

Предназначено для подготовки бакалавров по направлениям 09.03.04 «Программная инженерия» и 01.03.02 «Прикладная математика и информатика», также может быть полезно инженерно-техническим работникам этих областей знаний.

УДК 004.2(07+004.39(07))
ББК 3.973.2-02я7+3.973.2-04я7

Рецензенты: Институт кибербезопасности и защиты информации СПбПУ (д-р техн. наук, доц. Д. С. Лаврова); д-р техн. наук, проф. Е. В. Постников (АО «НИЦ СПб ЭТУ»).

Утверждено
редакционно-издательским советом университета
в качестве учебного пособия

ISBN 978-5-7629-3047-5

Оглавление

1. ИЕРАРХИЧЕСКАЯ СИСТЕМА ПАМЯТИ КОМПЬЮТЕРА

1.1. Иерархическая организация памяти

Для достижения компромисса между емкостью и быстродействием памяти, а также для снижения влияния разрыва в быстродействии процессора и памяти в современных компьютерах память реализуется в виде иерархической системы запоминающих устройств (ЗУ) (рис. 1.1) [1]. Более высокий уровень ЗУ характеризуется большим быстродействием, меньшей емкостью и большей стоимостью хранения 1 бит информации.

Иерархическая система памяти состоит из следующих уровней:

1. Сверхоперативная память (СОЗУ), которая реализуется на регистрах процессора, а также кеш-память 1-го уровня, размещаемая на одном кристалле с процессором и являющаяся быстрым видом памяти малой емкости.

2. Буферная память, создаваемая на основе кеш-памяти 2-го и 3-го уровней, вместе с кешем 1-го уровня служит для согласования скорости процессора и основной памяти. Образует с основной памятью систему буферизованной памяти. Для программиста эта память является прозрачной (скрытой), поэтому и называется кеш-памятью, она реализуется на биполярных элементах и размещается на одной плате с процессором. Для гарвардской архитектуры буферная память может разделяться на кеш команд и кеш данных.

3. Основная память (ОП) – блочно организованная память произвольного доступа, обеспечивающая хранение системных и пользовательских программ и данных, выполняемых в текущем интервале времени.

4. Дисковый кеш – буферная память, используемая для согласования быстродействия основной и дисковой памяти, реализуется на полупроводниковых элементах и может быть отдельным ЗУ, включаемым в состав магнитного дискового накопителя (МДН), или фрагментом ОП.

5. Твердотельная или магнитная дисковая память – так называемая внешняя память, являющаяся медленной (по сравнению с ОП) памятью большой емкости, служащей либо для хранения программ и данных, не используемых в настоящее время, либо для расширения доступного программисту адресного пространства путем организации совместно с ОП системы виртуальной памяти.

6. Архивная память – многотомные накопители информации большого объема, реализуемые на магнитных лентах или оптических дисках. Имеют самое низкое быстродействие и стоимость хранения и предназначены для долговременного хранения данных без разрушения.

Рис. 1.1

В целом следует отметить, что иерархическая организация памяти ведет к уменьшению общей стоимости хранения и доступа к информации при заданном уровне производительности.

1.2. Организация кеш-памяти

Использование кеш-памяти (cache memory) основано на свойстве *локальности* программ по обращению, имеющем три составляющие:

1) пространственную локальность программ, заключающуюся в том, что при выборке команд из памяти адрес очередной команды либо следует непосредственно за адресом предыдущей, либо находится вблизи него;

2) пространственную локальность данных, связанную с тем, что за счет широкого использования структурированных данных при обращении к данным, как правило, используются последовательные ячейки памяти;

3) временную локальность программ, заключающуюся в том, что в течение достаточно длительных интервалов времени исполняемая программа использует сравнительно небольшой диапазон адресов команд и/или небольшой диапазон адресов данных. Это обусловлено наличием в программе циклов и подпрограмм, а также необходимостью длительной обработки массивов данных.

Идея использования кеша как буферной памяти заключается в наличии двух видов памяти: быстрой памяти малой емкости M_1 (n_1, $t_{\text{обр1}}$) и медленной памяти большой емкости M_2 (n_2, $t_{\text{обр2}}$), параметры которых: число ячеек n_i и время обращения $t_{\text{обр}i}$ характеризуются неравенствами

$$n_1 << n_2 \text{ и } t_{обр1} << t_{обр2}.$$

Если данные имеются в быстрой кеш-памяти (эта ситуация называется cache hit – попадание в кеш), то они выбираются за время $t_{обр1}$, а если отсутствуют (эта ситуация называется cache miss – промах в кеше), то за время $t_{обр1} + t_{обр2}$ данные выбираются из основной памяти и одновременно подгружаются в кеш-память.

Если благодаря свойству локальности исполняемый фрагмент программы и обрабатываемый массив данных удается разместить в кеш-памяти, то вероятность $0 < \alpha << 1$ отсутствия команд или данных в кеш-памяти удается сделать достаточно малой – обычно $\alpha \approx 0.02 \ldots 0.05$. Тогда среднее время обращения к такой системе буферизованной памяти будет

$$M\{t_{обр}\} = (1 - \alpha) * t_{обр1} + \alpha * (t_{обр1} + t_{обр2}) = t_{обр1} + \alpha * t_{обр2} \approx t_{обр1}.$$

Поскольку емкость кеш-памяти значительно меньше емкости ОП, то эпизодически происходит обмен информацией между кешем и ОП, чтобы в кеше оказывалась актуальная на данное время информация. При этом обмен происходит *блоками* одинакового размера, которые в кеш-памяти принято называть *строками*. Когда процессор пытается выбрать слово из памяти, то сначала он ищет это слово в кеше, и если находит, то выбирает его из кеша, а при отсутствии слова в кеше оно выбирается из ОП. Одновременно из ОП в кеш-память пересылается блок данных, содержащий это слово [2].

Если длина строки кеша равна одному слову, то сказываются только преимущества временной локальности (например, при обработке циклов). Чтобы воспользоваться пространственной локальностью, в кеш-памяти используют строки, содержащие несколько последовательных слов. Преимущество строк с длиной, превышающей одно слово, заключается в том, что когда случается промах кеша и требуется прочитать слово данных из памяти, то в эту строку заодно загружаются и соседние слова. Таким образом, последующие обращения с большей вероятностью приведут к попаданию в кеш из-за пространственной локальности данных. Однако увеличившаяся длина строки означает, что кеш того же размера теперь будет иметь меньшее число самих строк. Это может привести к увеличению числа конфликтов и, соответственно, увеличить вероятность промахов кеша. Более того, потребуется больше времени на чтение данных в строку после промаха, так как из памяти необходимо будет прочитать не одно, а несколько слов. Время, требуемое для загрузки данных в строку кеша после промаха, называется *ценой промаха* (miss penalty). Если соседние слова данных в строке не будут использованы в дальнейшем, то уси-

лия на их загрузку будут потрачены зря. Тем не менее большинству реальных программ увеличение длины строки приносит пользу.

Так как число блоков ОП больше числа строк кеша, строка кеша не может быть выделена постоянно одному блоку ОП. Поэтому каждой строке кеша соответствует признак (*тег*), показывающий, копия какого блока ОП хранится в ней в данное время. В качестве тега обычно используется часть адреса ОП. При этом для разных типов (уровней) кеша применяются различные способы отображения ОП на кеш-память.

Основными параметрами, характеризующими кеш-память, являются: емкость C (в словах), количество строк или блоков M, длина (размер) строки L (в словах), число наборов S, объединяющих группу строк, и степень ассоциативности N (число строк кеша в наборе).

Применяемые способы отображения ОП на кеш-память классифицируются по числу строк в наборе кеша:

1. *Кеш прямого отображения* – каждый набор содержит только одну строку, так что кеш содержит $S = M$ наборов; таким образом, каждый из адресов основной памяти отображается в одну единственную строку кеша.

2. *Наборно-ассоциативный кеш* делится на наборы, каждый из которых состоит из N строк; при этом число наборов равно $S = M/N$, каждый адрес памяти по-прежнему отображается в один единственный набор, но данные могут оказаться в любой из N строк этого набора.

3. *Полностью ассоциативный кеш* имеет только один набор ($S = 1$), и данные могут оказаться в любой из M строк этого набора.

Рассмотрим различные способы отображения блоков ОП на кеш-память на следующем примере:

$C_{ОП}$ = 256 К слов = 2^{18} слов, $C_{кеш}$ = 2 К слов = 2^{11} слов, $C_{блока/строки}$ = = 16 слов (C – емкость); $M_{ОП} = N_{бл.\ ОП}$ = 256 К/16 = 16 К = 2^{14} блоков, $M_{кеш}$ = = 2 К/16 = 128 = 2^7 блоков (M – число блоков); $L_{адр.\ ОП}$ = 18 бит = 14 бит (адрес блока ОП) {7 бит (тег) + 7 бит (строка кеша)} + 4 бит (слово в блоке); $L_{адр.\ кеш}$ = = 11 бит = 7 бит (адрес строки кеша) + 4 бит (слово в блоке) (L – длина адреса в битах).

1.2.1. Прямое отображение блоков ОП на кеш-память

При *прямом отображении* адрес строки i кеш-памяти, на которую может быть отображен блок j ОП, однозначно определяется выражением [3]

$$i = j \bmod M_{кеш}.$$

В данном примере $i = j \bmod 128$, где $i = [0, 127]$, $j = [0, 16383]$ и на строку кеша с номером i отображается каждый 128-й блок ОП, начиная с блока i, что поясняет рис. 1.2. Здесь 14-битный адрес блока ОП разбивается на два поля: 7-битный тег (7 старших разрядов адреса) и 7-битный номер строки кеш-памяти, на которую может быть отображен этот блок ОП. При этом поле тега определяет номер блока в списке блоков ОП, закрепленных за данной строкой кеша, который сейчас адресуется. Когда блок ОП фактически заносится в соответствующую строку кеш-памяти, в поле тегов этой строки нужно записать тег именно этого блока, в качестве тега служат 7 старших разрядов адреса блока.

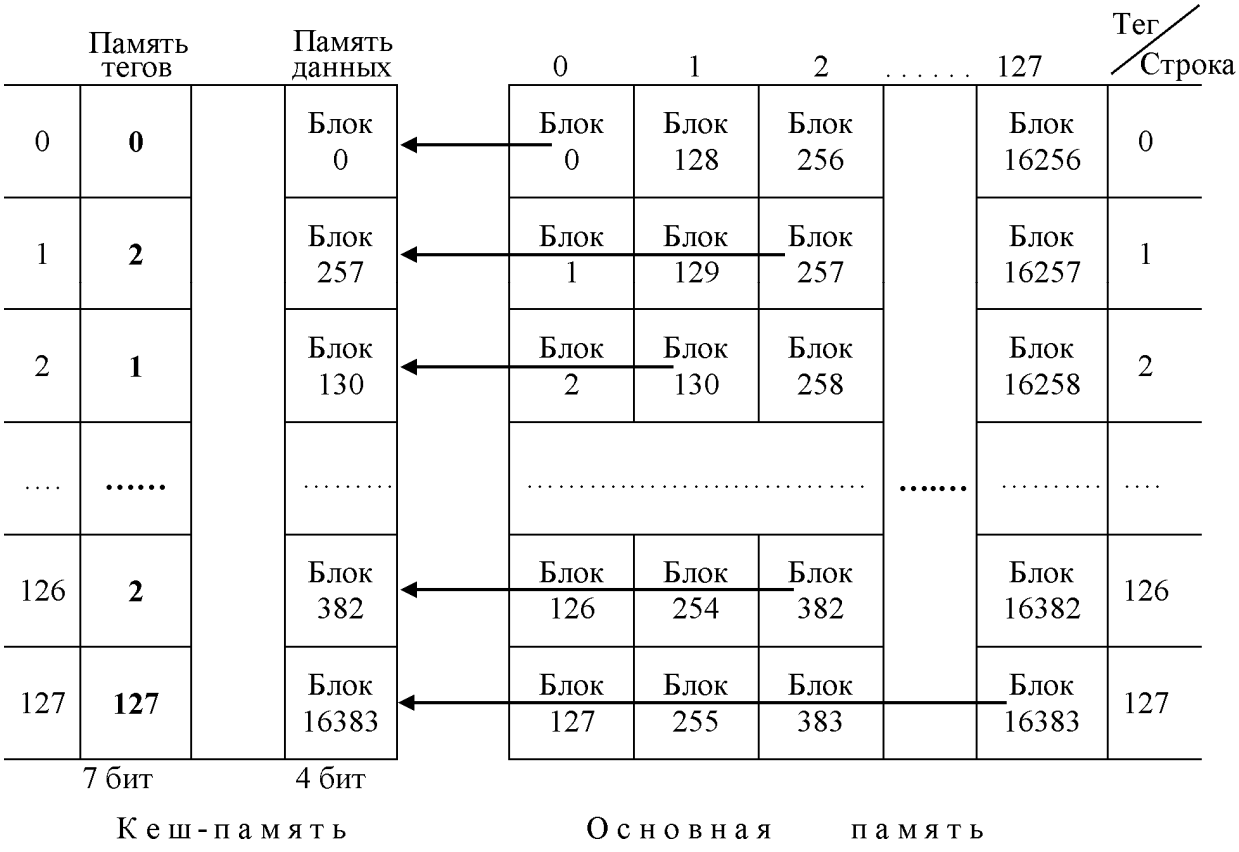

Рис. 1.2

При несомненной простоте прямого отображения его существенным недостатком является жесткое закрепление строки кеша за определенными блоками ОП. Поэтому при поочередном обращении программы к словам из двух блоков, отображаемых на одну и ту же строку кеша, постоянно будет происходить обновление этой строки, резко снижающее скорость доступа к памяти. Кеш с прямым отображением ввиду своей экономичности используется для построения кешей 2-го и 3-го уровней сравнительно большого объема.

1.2.2. Ассоциативное отображение блоков ОП на кеш-память

Такой способ отображения, иллюстрируемый рис. 1.3, позволяет выполнять загрузку любого блока ОП в любую строку кеш-памяти [4], [5]. В кеш-

памяти используется поле тегов, содержащее 2 тега: тег 1 (14 бит), соответствующий адресу блока ОП, и тег 2 (7 бит), определяющий адрес блока в кеше, а также поле данных, определяющее адрес слова в блоке. Контроллер кеш-памяти выделяет в адресе ОП два поля: поле тегов и поле слова. Поле тега совпадает с 14-битным адресом блока ОП. Для проверки наличия копии блока ОП в кеш-памяти логика управления контроллера кеша должна одновременно сравнить теги 1 всех строк кеша на совпадение с полем тега адреса ОП. Это обеспечивается использованием ассоциативной памяти для хранения тегов кеша. После нахождения строки кеша, тег 1 которой совпал с адресом искомого блока ОП, по тегу 2 определяется размещение искомого блока в кеше.

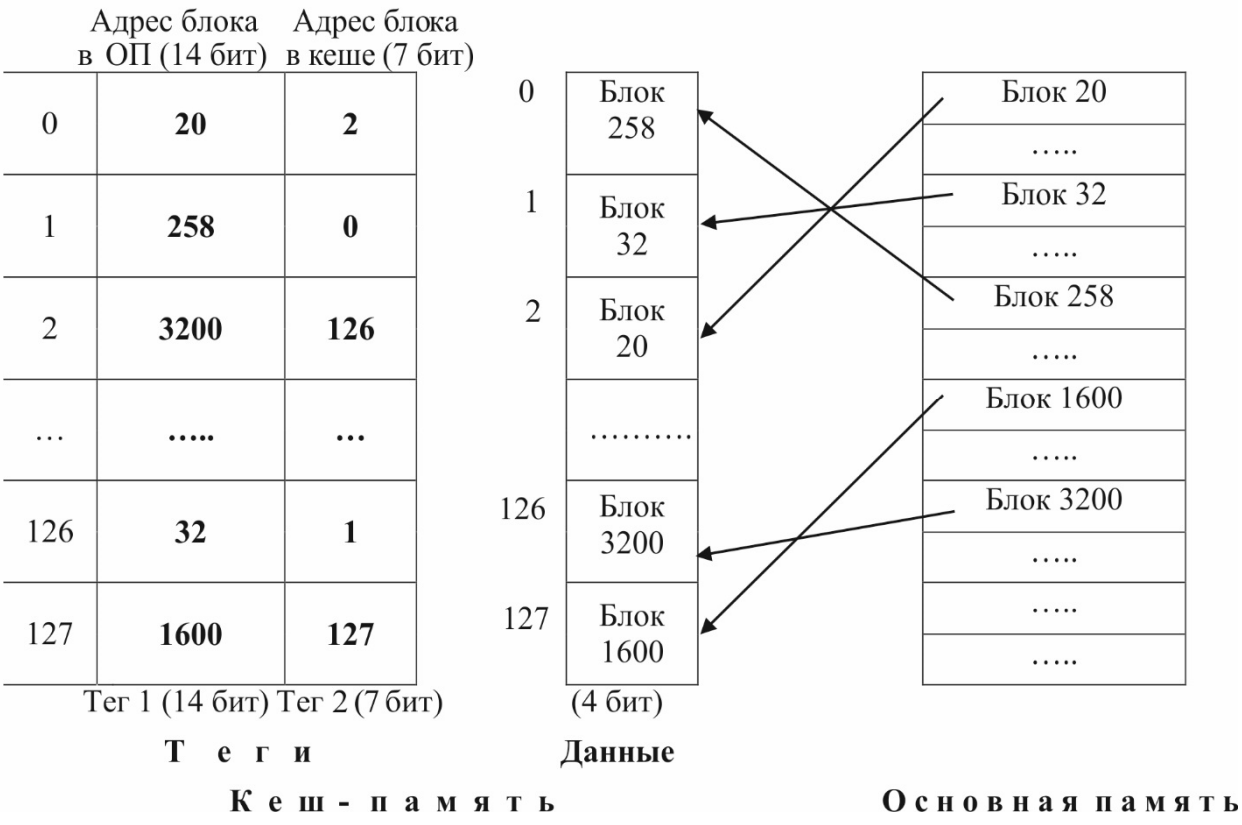

Рис. 1.3

Ассоциативное отображение обеспечивает гибкость при выборе строки кеш-памяти для размещения блока ОП, но требует использования дорогостоящей ассоциативной памяти, поэтому применяется только для построения кешей 1-го уровня (процессорных).

1.2.3. Наборно-ассоциативное отображение блоков ОП на кеш-память

Наборно-ассоциативное отображение, схема которого представлена на рис. 1.4, сочетает достоинства прямого и ассоциативного способов и позволяет каждому блоку ОП претендовать на одну из нескольких строк кеш-памяти,

объединенных в набор (множество). Можно считать, что в этом случае используется несколько параллельно и согласованно работающих каналов прямого отображения: каждому набору строк кеша соответствует жестко заданная группа блоков ОП, а в пределах набора контроллеру кеша по ассоциативному способу приходится выбирать, в какую строку набора помещать очередной блок данных из ОП.

В рассматриваемом примере кеш-память делится на 32 набора ($S = 32$) по 4 строки ($N = 4$) в каждом. Выбор набора i для размещения блока j ОП осуществляется по принципу прямого отображения:

$$i = j \bmod S.$$

14-битный адрес блока ОП разбивается на 9-битный тег, определяющий номер блока в списке блоков ОП, закрепленных за данным набором, и 5-битный номер набора, в который должен быть отображен этот блок ОП. Выбор строки набора (из 4 возможных) для размещения данных блока ОП осуществляется по ассоциативному признаку.

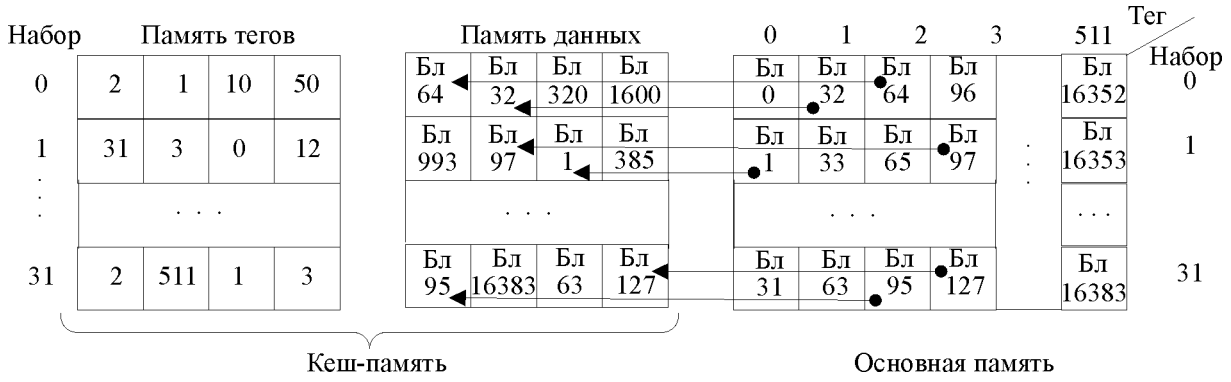

Рис. 1.4

Кеш с наборно-ассоциативным отображением ввиду относительно высокого быстродействия и не очень больших затрат может применяться для построения как первичных, так и вторичных кешей.

1.2.4. Проблемы при использовании кеш-памяти

1. *Замещение данных* в строке кеш-памяти на новый блок ОП при необходимости загрузки последнего в полностью занятую кеш-память. В кеш-памяти прямого отображения каждый адрес всегда отображается в одну и ту же строку одного и того же набора, поэтому когда нужно загрузить новые данные в набор, который уже содержит данные, строка в наборе просто замещается на новые данные.

В наборно-ассоциативной и полностью ассоциативной кеш-памяти нужно решить, какую именно из нескольких строк в наборе вытеснить. Учитывая

принцип временной локальности, наилучшим вариантом было бы заменить ту строку, которая дольше всего не использовалась, потому что маловероятно, что она будет использована снова. Именно поэтому большинство кешей используют стратегию замены долго неиспользуемых данных (least recently used – LRU), основанную на регистрации бита использования *U* в каждом наборе.

В наборно-ассоциативном кеше с двустрочными наборами *бит U* содержит номер той строки в наборе, которая дольше не использовалась. Каждый раз, когда происходит доступ к одной из строк набора, бит *U* устанавливается таким образом, чтобы указывать на другую строку. Для наборно-ассоциативных кешей с бóльшим количеством строк в наборе отслеживать самые редко используемые строки становится сложно. Для упрощения реализации строки часто делят на две группы, а бит использования указывает на ту группу, которая дольше не использовалась. При необходимости заместить строку вытесняется случайным образом выбранная строка из той группы, которая дольше не использовалась. Такая стратегия называется «псевдо-LRU» (pseudo-LRU) и на практике достаточно хорошо работает.

С целью упрощения реализации иногда используется стратегия замены FIFO (Fist-in, Fist-out) или обычная очередь – осуществляется замена в кеш-памяти блока ОП, который загружался раньше всех других; это более дешевая, но не столь эффективная, как LRU, стратегия замены блоков в кеше.

2. *Согласование данных* в кеше и ОП. Используются два основных способа согласования при записи данных в кеш:

1) Store through (Write through) – сквозная запись (одновременная запись данного в блок кеша и в блок основной памяти), при этом происходит трата ресурсов и времени на обновление ОП. Достоинство: надежный способ согласования данных;

2) Store In (Write back) – обратная (отложенная) запись: у каждой строки кеша есть бит изменения, устанавливаемый в 1 при записи в строку; измененные строки записываются обратно в ОП только тогда, когда они вытесняются из кеша. Достоинство: более высокая производительность. Недостаток: временное различие данных в ОП и в кеше может привести к конфликтам (проявляется в многопроцессорных системах с общей памятью).

Из-за того что время обращения к ОП велико, в современных системах обычно используют кеши с отложенной записью.

1.2.5. Сокращение частоты промахов

Процент промахов кеша можно сократить, изменяя его емкость, длину строки и/или ассоциативность [2]. Для этого необходимо разобраться в причи-

нах промахов. Промахи кеша делятся на *неизбежные* промахи (compulsory misses), промахи из-за недостаточной емкости (capacity misses) и промахи из-за конфликтов (conflict misses).

Первое обращение к строке кеша всегда приводит к неизбежному промаху, так как эту строку нужно прочесть из основной памяти хотя бы один раз независимо от архитектуры кеша. Промахи из-за недостаточной емкости происходят, когда кеш слишком мал для хранения всех одновременно используемых данных. Промахи из-за конфликтов случаются, если несколько адресов памяти отображаются на один и тот же набор кеша и выталкивают из него данные, которые все еще нужны.

Изменение параметров кеша может повлиять на число одного или нескольких типов промахов. Например, увеличение размера кеша может сократить промахи из-за недостатка емкости и промахи из-за конфликтов, но никак не повлияет на число неизбежных промахов. С другой стороны, увеличение длины строки может сократить число неизбежных промахов (благодаря локальности данных), но одновременно может увеличить частоту промахов из-за конфликтов, поскольку большее число адресов будет отображаться на один и тот же набор, увеличивая вероятность конфликтов. В целом организация кеш-памяти настолько сложна, что лучший способ оценивать ее производительность – это запускать тестовые программы, варьируя параметры кеша с получением графиков зависимости частоты промахов от размера кеша и степени ассоциативности для используемого набора тестовых программ.

1.2.6. Многоуровневые кеши

Чем больше размер кеша, тем больше вероятность, что интересующие нас данные в нем найдутся и у него будет меньше частота промахов. Но большой кеш обычно медленнее, чем маленький, поэтому в современных системах используются как минимум два уровня кеша. Кеш первого уровня ($L1$), реализуемый на основе SRAM, достаточно мал, чтобы обеспечить время доступа в один или два такта. Кеш второго уровня ($L2$) тоже делается на основе SRAM, но больше по размеру и поэтому медленнее, чем кеш $L1$. Сначала процессор ищет данные в кеше $L1$, а если происходит промах – то в кеше $L2$. Если и там происходит промах, то процессор обращается за данными к основной памяти. Многие современные системы используют еще больше уровней кеша в иерархии памяти, так как доступ к ОП чрезвычайно медленный, а необходимо согласовать быстродействие процессора и памяти.

1.3. NUMA-память

Одним из особых видов организации подсистемы памяти и ее взаимодействия с процессором является «память с неравномерным (или неоднородным) доступом» – NUMA (Non-Uniform Memory Access), ориентированная на повышение производительности в многопроцессорных системах [1].

Большинство типовых многопроцессорных систем реализовано в виде симметричной многопроцессорной архитектуры (Symmetric Multi Processing – SMP), в которой процессоры соединены с общей системной памятью симметрично и имеют к ней одинаковый однородный доступ, как показано на рис. 1.5 на примере двух CPU.

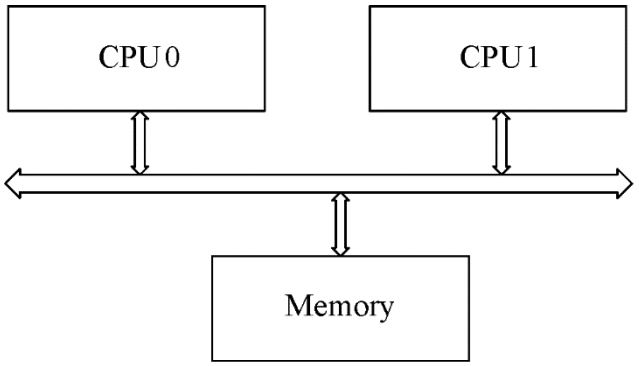

Рис. 1.5

С одной стороны, эта схема обеспечивает практически одинаковые задержки при доступе к памяти со стороны любого процессора, но с другой стороны, при росте числа процессоров общая системная шина становится узким местом всей системы, значительно ограничивая производительность приложений, интенсивно использующих память. Именно поэтому SMP-системы почти не масштабируются, два десятка процессоров для них – близко к пределу.

Альтернативой SMP является архитектура с NUMA-памятью, разделяющая систему на множественные узлы, имеющие доступ как к своей локальной памяти, так и к памяти других узлов (логично называемой «удаленной»). Задержки при обращении процессора к «своей» памяти оказываются невысоки (по сравнению с SMP-системой). В то же время доступ к «чужой» памяти, принадлежащей другому процессору, сопровождается более высокими задержками. Отсюда и название – «неоднородный доступ к памяти». Вместе с тем при правильной организации доступа к памяти (когда каждый процессор в основном оперирует данными, находящимися исключительно в «своей» памяти) такая схема будет выгодно отличаться от классического SMP-решения благодаря отсутствию ограничения по пропускной способности общей системной шины.

Двухсокетная NUMA-система Intel Xeon (а именно там дебютировала Intel NUMA) с контроллерами памяти, интегрированными в CPU, показана на рис. 1.6. Процессоры здесь связаны соединением «точка-точка» (Quick Path Intel) с высокой пропускной способностью и низкой задержкой передачи. На рисунке не показан кеш процессоров, но все уровни кеш-памяти, конечно же, здесь есть, а значит, есть и важная особенность NUMA, используемая в системах Intel, – поддержка когерентности кешей и разделяемой памяти (т. е. соответствие данных между кешами разных CPU), поэтому ее иногда называют ccNUMA – cache coherent NUMA. Это означает наличие специального аппаратного решения для согласования содержимого кешей и памяти, когда более чем один кеш хранит одну и ту же часть памяти. Такое соответствие кешей ухудшает общую производительность системы, когда несколько процессоров подряд запрашивают доступ к одному блоку памяти, но без него программировать систему с непредсказуемым текущим состоянием данных затруднительно. Для уменьшения влияния этого эффекта следует избегать ситуаций, когда несколько процессоров сразу работают с одним блоком памяти, к чему стремятся программные продукты, поддерживающие NUMA.

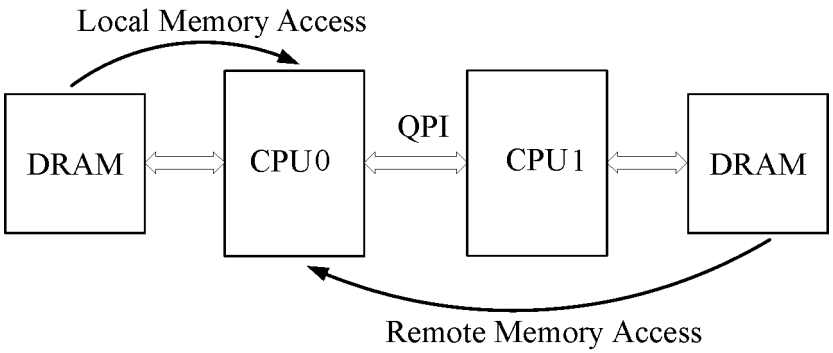

Рис. 1.6

Ключевым понятием использования архитектуры NUMA является «правильная организация доступа к памяти», которая должна поддерживаться как со стороны операционной системы (ОС) (хотя бы для того, чтобы сама система и приложения могли «увидеть» память всех процессоров, как единый блок памяти), так и со стороны приложений. Что касается приложений, здесь имеется в виду нежелательность возникновения ситуации, когда приложение размещает свои данные в области памяти одного процессора, после чего обращается к ним с другого процессора.

NUMA поддерживается следующими ОС: Windows XP 64-bit и Windows Vista – до 64 логических процессоров, Windows 7, Linux OS kernel: 2.6 и выше,

UNIX OS – Solaris. Если говорить о базах данных, то NUMA поддерживается Oracle8 – Oracle11, а также SQL Server 2005 и SQL Server 2008. Поддержка NUMA реализована и в Java SE 6u2, JVM 1.6, а также .NET runtime на вышеупомянутых версиях Windows.

В BIOS мультисокетных серверов с NUMA есть специальный пункт «*Разрешить/запретить NUMA*». В случае запрета NUMA в BIOS система не сообщит ОС и программному обеспечению (ПО) о том, что она NUMA, а значит, распределение памяти и планировка потоков будут «обычными», такими, как на симметричных многопроцессорных системах. Если BIOS разрешает NUMA, то операционная система сможет узнать о конфигурации NUMA узлов из System Resource Affinity Table (SRAT).

При выделении памяти при помощи malloc в Linux память только резервируется, а ее физическое выделение происходит при фактическом обращении к данной памяти. В этом случае память автоматически выделится на том узле, который ее и использует, что очень хорошо для NUMA. В Windows же malloc выделяет физическую память непосредственно на узле выделяющего память потока, и поэтому она может оказаться удаленной для других потоков, ее использующих. Но есть в Windows и дружественное к NUMA выделение памяти. Это VirtualAlloc, который может работать точно так же, как malloc в Linux.

Кроме того, стоит отдельно упомянуть *Affinity* – принудительную привязку потоков к конкретным процессорам, предотвращающую возможную переброску ОС потоков между процессорами, которая может вызвать потенциальный «отрыв» потоков от своей используемой локальной памяти. Для установки Affinity имеются соответствующие API как в Linux, так и в Windows.

Результаты анализа различных исследований позволяют заключить, что NUMA – более совершенная архитектура памяти по сравнению с традиционными SMP-решениями, способная во многих случаях обеспечить над ними преимущество по характеристикам подсистемы памяти.

1.4. Организация виртуальной памяти

Большинство современных вычислительных систем (ВС) в качестве нижнего уровня в иерархии памяти используют жесткие диски, представляющие собой магнитные или твердотельные ЗУ. По сравнению с идеальной памятью, которая должна быть быстрой, дешевой и большой, жесткий диск имеет большой объем и недорого стоит, однако очень медленно работает. Если существенная часть обращений к памяти осуществляется к жесткому диску, скорость работы ВС сильно снижается.

Виртуальная память – система основной и дисковой памяти, организуемая для расширения адресного пространства, доступного программам пользователей, при условии обеспечения достаточной скорости работы. При работе с виртуальной памятью различают:

1. ФАП (физическое адресное пространство) – совокупность адресов, соответствующих реально адресуемым физическим ячейкам памяти.

2. ЛАП (ВАП) (логическое или виртуальное адресное пространство) – это совокупность адресов, которая может использоваться для доступа к данным.

Очевидно, что емкость ВАП должна значительно превышать емкость ФАП.

Программы могут обращаться к данным в любом месте виртуальной памяти, поэтому они должны использовать *виртуальные адреса*, которые определяют расположение данных в виртуальной памяти.

Физическая память хранит последние запрошенные из виртуальной памяти блоки данных. Таким образом, физическая память выступает в роли кеша для виртуальной памяти, т. е. большинство обращений происходит к быстрой физической памяти (DRAM), и в то же время программа имеет доступ к большей по объему виртуальной памяти [2], [5].

Подсистемы виртуальной памяти используют другие термины для тех же самых принципов кеширования, которые были рассмотрены в 1.2. В табл. 1.1 приведено соответствие терминов кеша и виртуальной памяти.

Таблица 1.1

Кеш	Виртуальная память
Строка	Страница
Длина строки	Размер страницы
Смещение от начала строки	Смещение от начала страницы
Промах	Страничная ошибка
Тег	Номер виртуальной страницы

Для обеспечения доступа к ВАП из программ пользователей необходимо установить строгое соответствие между данными, размещаемыми в ОП и размещенными на диске, отражаемое в специальных таблицах соответствия адресов. Для сокращения длины таблицы соответствия адресов информация, хранящаяся в основной памяти и на диске, должна быть разбита на одинаковые (в обоих видах памяти) блоки, для которых и устанавливается соответствие.

Разбиение происходит двумя способами: 1) страничным; 2) сегментным.

При первом способе вся память (основная и дисковая) разбивается на одинаковые по размерам блоки, называемые страницами; разбиение происходит независимо от типа и характера размещаемых данных. Говорят, что страничное разбиение ориентируется на физическую память. При втором – основная

и дисковая память разбиваются на сегменты, как правило, соответствующие логически завершенным объектам задачи (программа, массив данных и т. д.); вследствие этого длина каждого сегмента может быть произвольной. Говорят, что сегментное разбиение имеет логическую ориентацию. Первый способ разбиения имеет более простые организацию и стратегию подгрузки и замещения страниц в основной памяти. Для второго способа достоинством является удобство организации защиты информации и коллективного доступа к данным.

Процесс преобразования виртуального адреса (ВА) в физический (ФА) называется трансляцией адреса. Рассмотрим трансляцию адреса для каждого из способов разбиения памяти. Трансляцию адреса для страничной организации виртуальной памяти можно пояснить с помощью схемы, представленной на рис. 1.7.

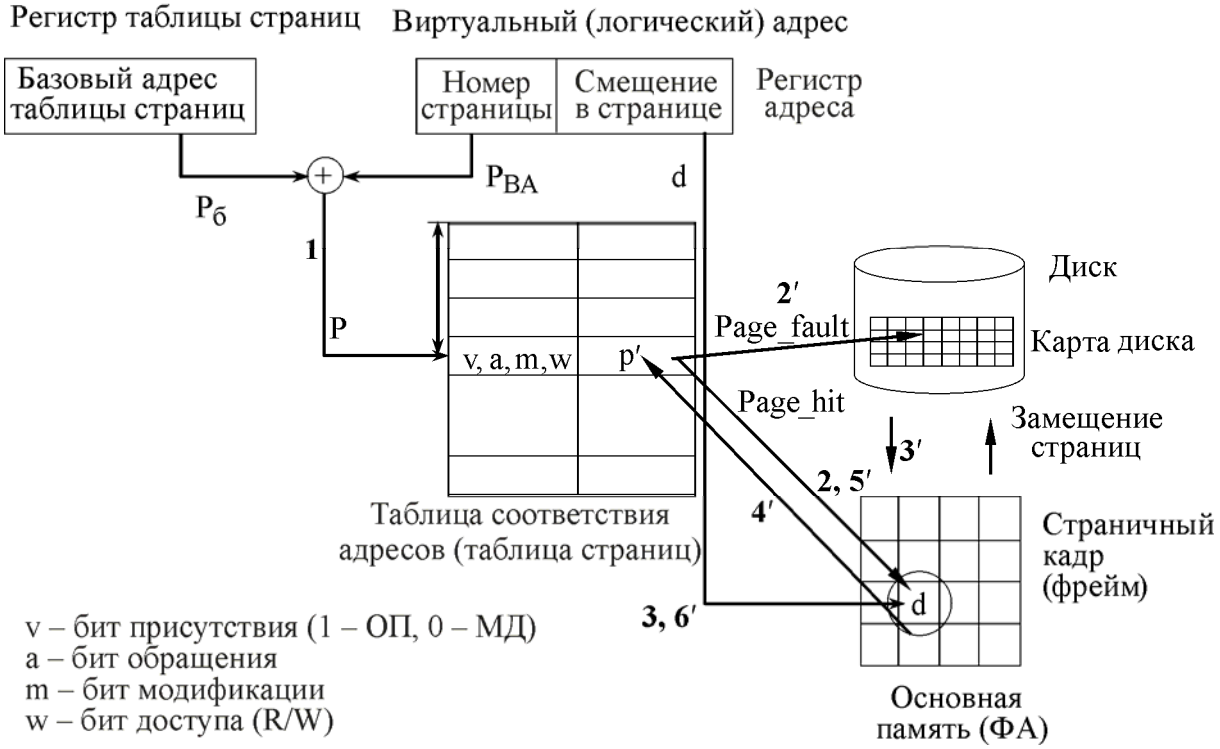

Рис. 1.7

Таблица страниц может храниться в любом месте физической памяти, ее расположение определяется операционной системой. Процессор использует выделенный регистр, называемый регистром таблицы страниц, для хранения ее базового адреса. Чтобы выполнить операцию загрузки или сохранения данных, процессор сначала транслирует виртуальный адрес в физический, а затем обращается к физической памяти, используя полученный физический адрес. Младшие биты виртуального адреса, определяющие смещение адреса внутри страницы, одинаковы для ВА и ФА, поэтому транслируется только номер стра-

ницы и сущность преобразования адресов: ВА p**.** d –> ФА p′**.** d, где операция «**.**» означает конкатенацию.

На рис. 1.7 цифрами обозначены шаги преобразования виртуального адреса в физический: 1 – по номеру виртуальной страницы и базовому адресу размещения таблицы страниц происходит обращение к строке таблицы страниц; 2 – если бит присутствия v строки показывает, что страница имеется в ОП (page hit), то по номеру p′ ее размещения в ОП происходит обращение к странице; 2′ – если бит присутствия v строки показывает, что страница отсутствует в ОП (page fault), то происходит обращение к карте диска, содержащей информацию о размещении виртуальных страниц на диске; 3 – используя смещение данного в странице непосредственно из ВА, получаем ФА данного в ОП, и на этом трансляция заканчивается; 3′ – выполняется подкачка требуемой виртуальной страницы в ОП, возможно, с замещением одной из физических страниц; 4′ – номер подгруженной страницы p′ засылается в используемую строку таблицы страниц, и бит v строки устанавливается в 1; 5′ – по номеру p′ размещения в ОП происходит обращение к странице; 6′ – используя смещение данного в странице непосредственно из ВА, получаем ФА данного в ОП, и на этом трансляция заканчивается.

При замещении страниц в основной памяти используются следующие стратегии:

1. FIFO – заменяется страница ОП, которая была загружена первой из имеющихся в ОП страниц.

2. LRU – заменяется страница ОП, к которой дольше всего не было обращений.

3. WS (work set) – рабочее множество: для конкретных интервалов времени создается совокупность страниц ОП, которые активно используются и не замещаются, остальные можно удалить из памяти.

Приведенная схема соответствует принципу прямого соответствия адресов, при котором для любого ВА есть свой ФА. При этом число строк таблицы страниц равно числу виртуальных страниц. Для сокращения времени доступа к таблице страниц используется принцип ассоциативного отображения страниц, реализуемый с помощью *буфера ассоциативной трансляции.* Поскольку таблица страниц хранится в физической памяти, каждая команда загрузки или сохранения требует два обращения к физической памяти. Соответственно, виртуальная память оказывала бы негативное влияние на производительность, если бы требовалось обращение к таблице страниц при выполнении каждой

команды загрузки или сохранения – это удваивало бы время выполнения этих команд. К счастью, обращения к таблице страниц имеют высокую временную локальность. Временная и пространственная локальность обращений к данным и большой размер страницы означают, что с большой вероятностью многие следующие друг за другом команды загрузки или сохранения обращаются к одной и той же странице. Поэтому если процессор запомнит последнюю запись таблицы страниц, которую он прочитал, то он сможет повторно использовать результат трансляции, не выполняя повторное чтение таблицы страниц. В целом процессор может хранить несколько последних записей, прочитанных из таблицы страниц в небольшой кеш-памяти, называемой буфером ассоциативной трансляции (Translation Lookaside Buffer – TLB). Процессор «заглядывает» в TLB в поисках информации, прежде чем обратиться к таблице страниц в физической памяти. В реальных программах большинство обращений находят в TLB нужную информацию (происходит попадание в TLB), что избавляет от затрат на повторное чтение таблицы страниц из физической памяти. TLB организован как полностью ассоциативный кеш и обычно хранит от 16 до 512 записей. Каждая запись в TLB хранит номер виртуальной страницы и соответствующий ей номер физической страницы, как это показано на рис. 1.8.

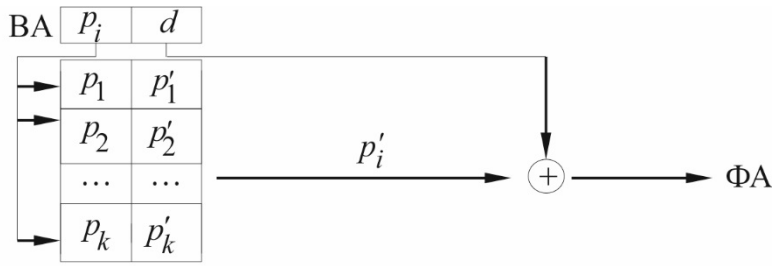

Рис. 1.8

Обращение к TLB происходит по номеру виртуальной страницы. Если происходит попадание в TLB, то возвращается соответствующий номер физической страницы. В противном случае процессор должен прочитать нужную запись из таблицы страниц в физической памяти. Буфер ассоциативной трансляции разрабатывают таким образом, чтобы он был маленьким и чтобы доступ к нему занимал около одного такта. Даже при этом доля попаданий в него обычно превышает 90 %. TLB уменьшает число обращений к памяти, требуемое для большинства команд загрузки и сохранения, с двух до одного.

Сегментная организация виртуальной памяти (ВП) условно поясняемая рис. 1.9, во многом аналогична станичной организации за исключением того, что длины сегментов могут существенно отличаться. Поэтому строка таблицы

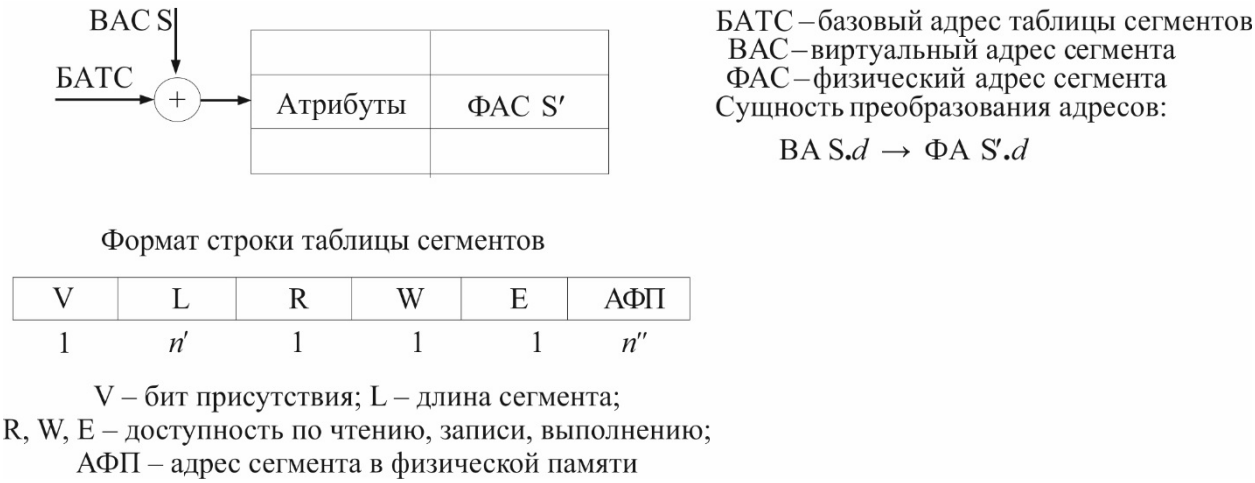

Рис. 1.9

сегментов должна содержать набор атрибутов, задающих как факт присутствия сегмента в ОП, так и его длину и способ доступа к сегменту.

Одной из существенных проблем виртуальной памяти является фрагментация памяти. Обычно различают внутреннюю и внешнюю фрагментацию. Внутренняя возникает от неполного использования страницы и образования участков памяти, доступ к которым невозможен или затруднен. Внешняя возникает только при сегментной организации ВП из-за того, что удаляемый

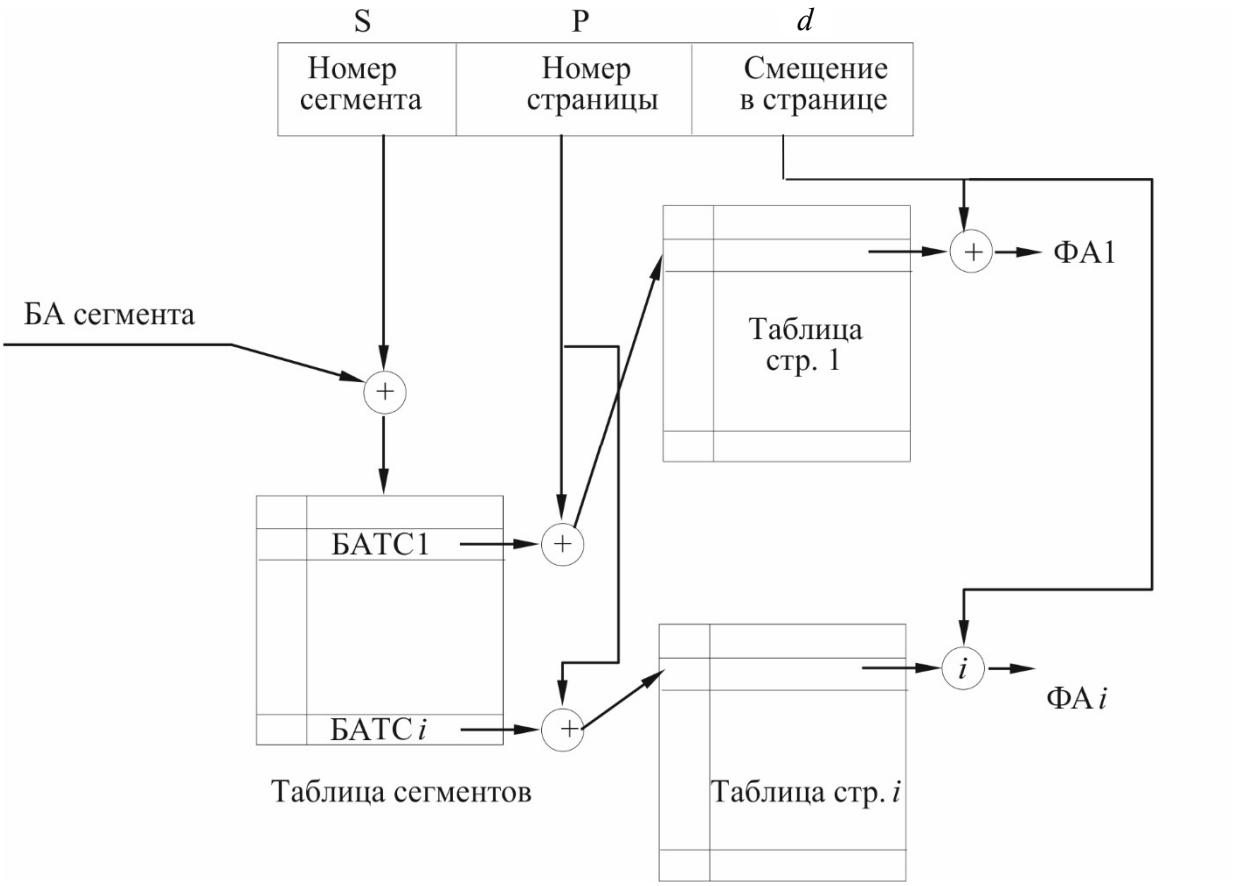

Рис. 1.10

и размещаемый на его место сегменты имеют разные размеры, в результате чего в физической памяти образуются «дырки», которые нельзя использовать. Для преодоления фрагментации ОС запускает процедуру, называемую «сборка мусора», для объединения неиспользуемых остатков страниц или «дырок» в сплошные блоки. Кроме того, для устранения внешней фрагментации и сокращения длины таблицы страниц применяется сегментно-страничная организация, поясняемая на рис. 1.10.

В данном случае используется двухуровневое разбиение: сегмент рассматривается не как целый блок, а разбивается на страницы, и именно части страниц будут теряться при фрагментации. Таблицу сегментов также рассматривают не как указатель на сплошное адресное пространство, а как совокупность указателей на различные, не обязательно смежные страницы. Для различных задач могут быть разные таблицы сегментов и разные связи с таблицами страниц. В результате на логическом уровне обеспечивается сегментная организация, а на физическом – страничная.

1.5. Организация виртуальной памяти в Intel 386 и более старших моделях

Процессоры могут работать в трех режимах:

1. Реальный режим – работает как быстрый процессор i8086, особенно с 32-битными данными, и имеет адресное пространство памяти до 4 Гбайт.

2. Защищенный режим – использует виртуальную организацию памяти, многозадачную работу, развитую систему защиты памяти.

3. Режим виртуальной машины i8086 – используется для совместимости с процессором i8086.

В защищенном режиме размер виртуальной памяти $С_{ВАП} = 2^{46}$ байт = 64 Тбайт. Это достигается благодаря разбиению ВАП на 16 К сегментов, максимальный размер каждого из которых равен 4 Гбайт.

При организации виртуальной памяти используется три вида адресов:

1. Логический адрес ЛА $\in$ ВАП.

2. Физический адрес ФА $\in$ ФАП.

3. Линейный адрес представляет собой объединение базового адреса сегмента и смещения в пределах сегмента.

Любое описание сегмента состоит из двух частей: *программно-доступной,* называемой *селектором сегмента* и размещаемой в одном из сегментных

регистров, и *скрытой,* называемой *дескриптором сегмента* и находящейся в одной из специальных структур в ОП, называемых таблицами дескрипторов [6]. Считается, что дескриптор после выборки размещается в скрытой части регистра сегментов, находящейся в памяти (рис. 1.11).

Рис. 1.11

Индекс задает смещение строки с описанием (дескриптором) данного сегмента в таблице дескрипторов. Поле TI определяет вид таблицы дескрипторов, на которую ссылается селектор (0 – глобальная – GDT, 1 – локальная – LDT и т. д.). Поле RPL (0–3) – запрашиваемый уровень привилегий (инициатор запроса – программа, которая хочет получить доступ к ресурсам). Содержание селекторов сегментов формируется операционной системой при загрузке задачи.

Структура дескриптора сегмента показана на рис. 1.12.

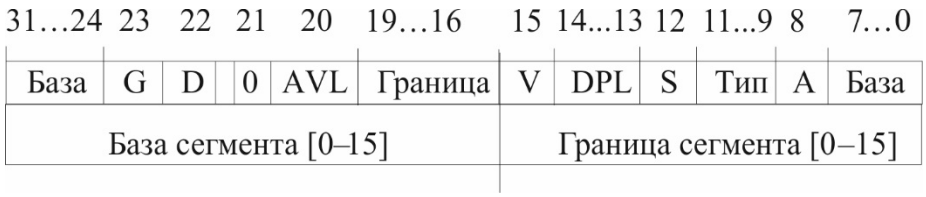

Рис. 1.12

Основные поля описания сегмента размещены в различных частях дескриптора: 32-битная база сегмента размещена в трех частях и определяет его место внутри 4-гигабайтного линейного адресного пространства, 20-битная граница сегмента размещена в двух частях и определяет его длину, остальные биты дескриптора образуют поле атрибутов.

Длина сегмента, в зависимости от бита дробности (G), задается либо в байтах, либо в страницах: при G = 0 – в байтах, а при G = 1 – в страницах (*C* = = 4 Кбайт) и объем будет $2^{20} * 2^{12} = 2^{32}$ (4 Гбайт).

Бит D задает тип данных, размещенных в сегменте: D = 0 – 16-битные данные; D = 1 – 32-битные данные; AVL – бит, предоставляемый в распоряжение пользователя (available); бит V – бит присутствия: 0 – на диске, 1 – в оперативной памяти; биты DPL – уровень привилегий дескриптора (Descriptor Privilege Level), обеспечивающий защиту доступа к сегменту.

При доступе к сегменту запрашиваемый уровень привилегий RPL или текущий уровень привилегий CPL (соответствует уровню привилегий дескриптора кода выполняемой сейчас программы) должен быть больше DPL (в логическом смысле; для сравнения их числовых значений отношение должно быть меньше).

Бит S – бит режима сегмента: пользовательский или системный (User / Supervisor). Биты типа говорят, является сегмент программой или данными, и определяют режим доступа: R – только чтение; W – только запись; E – бит расширения при загрузке сегмента (1 – для увеличения адресов, 0 – для уменьшения адресов); бит C – признак согласования сегментов (уровень привилегий, с которым обращаются к данному сегменту, всегда приравнивается к уровню данного сегмента – в моделях процессоров выше i386 он существует только для поддержки).

Биты	11	10	9
Данные	1	E	W
Код	0	C	R

Бит A – бит обращения к сегменту, устанавливается при обращении к сегменту и через некоторое время сбрасывается операционной системой (обеспечивает дисциплины замещения сегментов).

На рис. 1.13–1.15 представлены, соответственно, алгоритм преобразования виртуального адреса в физический и схемы выполнения сегментной и страничной трансляции адресов.

I этап. **Сегментная трансляция** – это преобразование логического адреса (селектор сегмента и смещение сегмента из команды программы) в линейный адрес. На основании селектора происходит выборка дескриптора из локальной или глобальной таблицы дескрипторов и запись его в скрытую часть сегментного регистра. В результате формируется линейный адрес сегмента (ЛАС) по правилу ЛАС = БАС + смещение.

II этап. **Страничная трансляция** – это преобразование линейного адреса в физический адрес. Если используется реальный режим или нет разбиения сегмента на страницы, то этот этап пропускается и ФА = ЛАС.

Алгоритм преобразования виртуального адреса в физический адрес

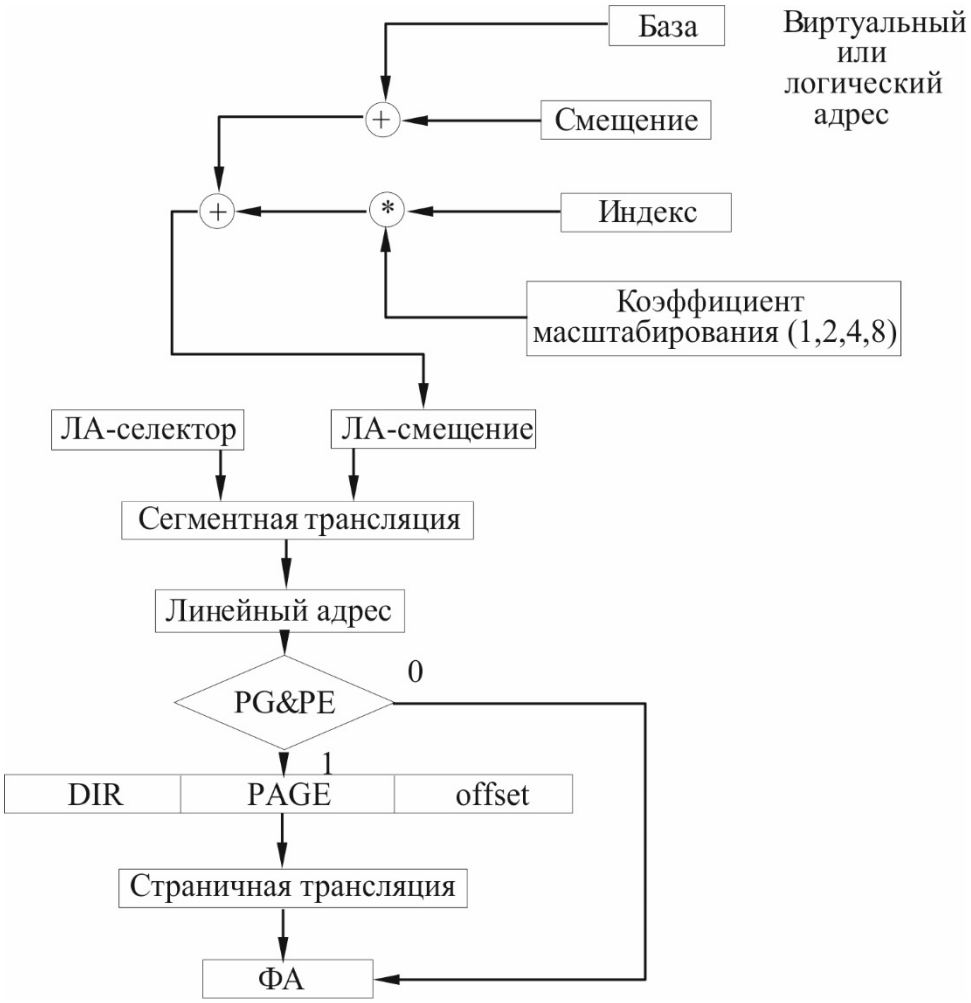

Рис. 1.13

Схема выполнения сегментной трансляции

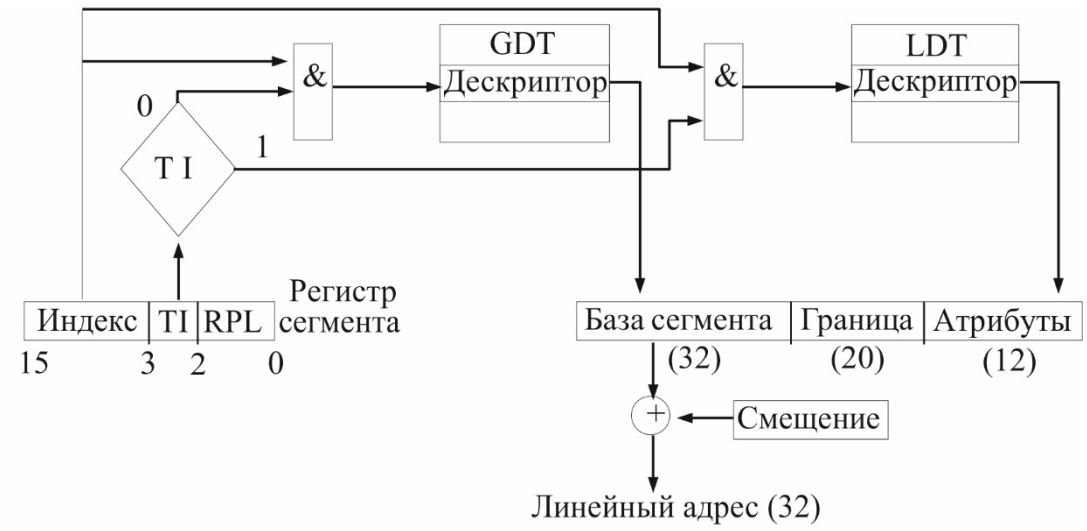

Рис. 1.14

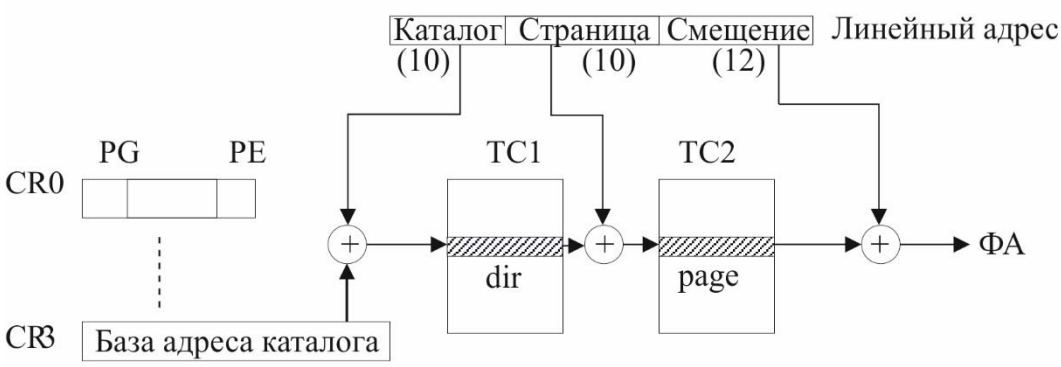

Рис. 1.15

На рис. 1.15 используются обозначения: CR*i* – *i*-й (*i* = 0…3) регистр управления; PG – бит признака страничной организации сегмента; PE (Protect Enable) – признак установки защищенного режима; (CR0.PG)&(CR0.PE) = = 1 – признак выполнения страничной трансляции; TC1 – таблица каталогов; TC2 – таблица страниц.

Размер ФАП = 1 К каталогов* 1 К страниц * 4 К смещение = 1 k × 1 k × × 4 k = 4 Gb.

Формат строки таблицы страниц:

31 12	11	10 7	6	5	4 3	2	1	0
Адрес страничного кадра	AVL	0	D	A	00	U/S	R/W	V

AVL – бит, предоставляемый в распоряжение пользователя; V – бит присутствия страницы; D – бит мусора; A – бит обращения к странице; U/S (User / Supervisor) – бит режима использования страницы: пользовательский или системный; R/W (Read/Write) – бит доступа (по чтению/записи).

1.6. Защита памяти в процессоре Intel 80386

Защита памяти обеспечивается только при CR0.PE = 1 (признак защищенного режима). Защита может осуществляться на уровне страниц и сегментов. Защита на уровне сегментов и страниц обеспечивается либо по атрибутам дескриптора сегмента, либо по атрибутам строки таблицы страниц. При защите на уровне сегментов влияет соотношение уровней привилегий:

• RPL – запрашиваемый уровень (при помещении селектора в сегментный регистр);

• DPL – уровень привилегий дескриптора, фиксируется при первом обращении к данным дескриптора и является минимальным уровнем, при котором данный сегмент доступен;

• CPL – текущий уровень привилегий, который соответствует уровню кодового сегмента текущей программы.

Виды защиты на уровне сегментов:

1. Контроль типа не зависят от уровня привилегий.
2. Контроль границ определяются дескриптором.
3. Ограничение адресуемой области памяти.
4. Ограничение точек входа в процедуру.
5. Ограничение набора команд.

(3–5: Зависят от RPL, DPL, CPL)

Контроль типа выполняется:

1. На этапе загрузки селектора в регистр сегмента (проверка типа сегмента, для которого осуществляется загрузка селектора).

2. На этапе обращения к сегменту.

Примеры. В CS может быть загружен только селектор дескриптора программного сегмента; в DS не может быть загружен селектор дескриптора, который недоступен для чтения; в SS не может быть загружен селектор дескриптора, который недоступен для записи.

При обращении нельзя:

1) записать в кодовый сегмент;

2) записать в сегмент данных, если бит записи равен нулю;

3) читать кодовый сегмент, если бит чтения равен нулю.

Контроль границ. Предотвращает выход за пределы сегмента.

Ограничение адресуемой области памяти. Доступ к памяти возможен при условии $\{RPL, CPL\} \le DPL$ (численно).

Ограничение точек входа в процедуру:

Если $CPL = DPL$, то возможен вход в процедуру по Call, Jump.

Если $CPL \ge DPL$ (численно), то возможен только доступ по Call через специальный дескриптор, называемый «вентиль вызова».

Ограничение набора команд. Все команды по уровню доступа на выполнение делятся на 3 группы:

1. Свободные команды (непривилегированные) выполняются в любом режиме.

2. Привилегированные команды выполняются на уровне ядра.

3. Чувствительные команды зависят от уровня привилегий, но могут выполняться в любом режиме супервизора (от 0 до 2). В основном это команды ввода-вывода.

Виды защиты на уровне страниц:

1. Контроль типа.

2. Ограничение адресуемой области памяти.

На уровне страниц защита зависит только от битов режима U/S и доступа R/W.

2. ОРГАНИЗАЦИЯ РАБОТЫ С ВНЕШНЕЙ ПАМЯТЬЮ

2.1. Типы, виды, свойства внешних накопителей информации

Выпускаемые накопители информации представляют собой гамму запоминающих устройств с различным принципом действия, физическими и эксплуатационными характеристиками. ЗУ принято делить на виды и категории в связи с их принципами функционирования, эксплуатационными, физическими и программными характеристиками. Так, например, по принципам функционирования различают следующие виды устройств [4], [7]:

– магнитные;

– электронные;

– оптические;

– смешанные – магнитооптические.

Каждый тип устройств организован на основе соответствующей технологии хранения/воспроизведения/записи цифровой информации. Сначала рассмотрим дисковые магнитные накопители – накопители на жестких и гибких магнитных дисках (ЖМД и ГМД).

2.2. Магнитные дисковые накопители

2.2.1. Состав и основные свойства магнитных ЗУ

Магнитные ЗУ состоят из собственно устройств чтения/записи информации и магнитного носителя, на который осуществляется запись и с которого считывается информация. Общая технология магнитных ЗУ состоит в намагничивании переменным магнитным полем участков носителя при записи информации и считывании информации, закодированной как области переменной намагниченности. Запись выполняется в цифровом коде. Дисковые устройства делят на два типа:

– гибкие (Floppy Disk);

– и жесткие (Hard Disk).

Основным свойством дисковых магнитных ЗУ является запись информации на концентрические замкнутые дорожки, равномерно распределенные по всему носителю с использованием физического и логического цифрового кодирования информации. Плоский дисковый носитель вращается в процессе чтения/записи, обеспечивая обслуживание всей концентрической дорожки, чтение и запись осуществляются при помощи магнитных головок чтения/записи, которые позиционируют по радиусу носителя с одной дорожки на другую.

Накопители на ЖМД (НЖМД) объединяют в одном корпусе носители и устройство чтения/записи, а также интерфейсную часть, называемую контроллером жесткого диска. Обычно рядом с корпусом носителей и головок располагаются схемы управления головками, дисками и интерфейсная часть и/или контроллер. В случае большего, чем один диск, числа носителей все дорожки, находящиеся одна под другой, называются цилиндром. Операции чтения/записи осуществляются подряд над всеми дорожками цилиндра, после чего головки перемещаются на новую позицию.

Диски вращаются постоянно, а скорость вращения носителей довольно высокая (от 4500 до 10 000 об/мин и выше), что обеспечивает высокую скорость чтения/записи. По величине диаметра носителя чаще других производятся диски 5.25, 3.14, 2.3 дюйма.

2.2.2. Основные физические и логические параметры ЖМД

Все накопители соответствуют стандартам, определяемым либо независимыми комитетами, либо самими производителями. Среди множества технических характеристик, отличающих одну модель от другой, можно выделить наиболее важные с точки зрения пользователей и производителей, которые используются при сравнении накопителей различных производителей и выборе устройства:

– диаметр дисков (disk diameter);
– число поверхностей (sides number);
– число цилиндров (cylinders number);
– число секторов (sectors count);
– число секторов на дорожке (sectors per track);
– частота вращения шпинделя (spindle speed);
– время перехода от одной дорожки к другой (track-to-track seek time);
– среднее время установки или поиска (average seek time);
– время ожидания (latency);
– время доступа (access time);
– среднее время доступа к данным (average access time);
– скорость передачи данных (data transfer rate);
– размер кеш-буфера контроллера (internal cash size);
– уровень шума (noise level);
– среднее время наработки на отказ (MTBF);
– физический и логический объем накопителей.

2.2.3. Контроллеры жестких дисков

Контроллер НЖМД физически расположен на плате электроники и предназначен для обеспечения операций преобразования и пересылки информации от головок чтения/записи к интерфейсу накопителя. Это сложнейшее устройство – микрокомпьютер, со своими процессором, ОЗУ и ПЗУ, схемами и системой ввода/вывода. В большинстве случаев производители размещают их в одном или двух микрочипах. Многие производители создают устройства, которые записывают различный объем информации на внутренние и внешние дорожки за счет размещения на них разного числа секторов. Это возможно благодаря аппаратному скрытию от программ и пользователя физических характеристик устройства на уровне его контроллера и/или интерфейса (устройства с интерфейсами IDE, EIDE и SCSI). Поэтому накопители, как правило, имеют различное физическое и логическое число цилиндров.

Режимы работы контроллеров НЖМД. Большинство современных накопителей поддерживают следующие режимы работы контроллеров:

– PIO;

– DMA2;

– Ultra DMA.

PIO (Programmed Input/Output – программный ввод-вывод), при котором все пересылки выполняет непосредственно центральный процессор. DMA (Direct Memory Access) – прямой доступ к памяти – режим взаимодействия контроллера накопителя и интерфейса ПК, при котором обмен данными осуществляется без участия ЦП. Режим DMA позволяет заметно разгрузить процессор по сравнению с режимом PIO. Все современные накопители могут работать в режиме DMA2, если это поддерживается ОС, а скорость обмена при этом может достигать16.6 Мбайт/с.

Накопители и системы с поддержкой режима Ultra DMA, при использовании соответствующего драйвера, могут передавать и принимать информацию со скоростью 33.3 Мбайт/с – для Ultra DMA-33, 66 Мбайт/с – для Ultra DMA-66 и 100 Мбайт/с – для Ultra DMA-100. В настоящее время современные модели винчестеров поддерживают стандарты Ultra DMA-66 и Ultra DMA-100.

2.2.4. Логическое хранение и кодирование информации

Для обеспечения оптимальной производительности, а также для улучшения программного интерфейса в НЖМД на основе физических структур – дорожек и секторов используется логическая структура хранения и доступа к

информации. Она называется файловой системой, а ее тип и характеристики зависят от используемой ОС. В настоящее время имеется много типов различных файловых систем, но они основывают свои логические структуры данных на нескольких первичных логических структурах. Рассмотрим их подробнее.

Master Boot Record (MBR). Первый сектор жесткого диска содержит хозяйственную загрузочную запись Master Boot Record (MBR), формат которой показан в табл. 2.1. MBR, в свою очередь, содержит загрузочную запись Boot Record (BR), выполняющуюся в процессе загрузки ОС. Загрузочная запись жестких дисков является объектом атаки компьютерных вирусов, заражающих MBR. За загрузчиком расположена таблица разделов Partition Table (PT), содержащая 4 записи – элементы логических разделов (Partitions). Завершается MBR специальной сигнатурой – последовательностью из 2 байт с шестнадцатеричными значениями 55H и AAH, указывающей на то, что данный раздел, после которого расположена сигнатура, является последним разделом в таблице.

Таблица 2.1

Название записи в MBR	Длина, байт
Загрузочная запись – Boot Record	446
Элемент таблицы разделов 1 – Partition 1	16
Элемент таблицы разделов 2 – Partition 2	16
Элемент таблицы разделов 3 – Partition 3	16
Элемент таблицы разделов 4 – Partition 4	16
Сигнатура окончания Partition Table	2

Каждый элемент таблицы разделов имеет формат, представленный в табл. 2.2, и содержит информацию о логическом разделе. Первым байтом в элементе раздела идет флаг активности раздела (0 – не активен, 128 (80H) – активен). Он служит для определения, является ли раздел системным и можно ли производить с него загрузку ОС при старте компьютера. Активным может быть только один раздел. Небольшие программы, называемые менеджерами загрузки (Boot Manager), могут располагаться в первых секторах диска. Они интерактивно запрашивают пользователя, с какого раздела выполнять загрузку, и, соответственно, корректируют флаги активности разделов. За флагом активности раздела следует байт номера головки, с которой начинается раздел. За ним следуют два байта, означающие, соответственно, номер сектора и номер цилиндра загрузочного сектора, где располагается первый сектор загрузчика ОС. Загрузчик ОС представляет собой маленькую программу, осуществляющую считывание в память начального кода ОС во время ее старта.

Затем следуют байты – кодовый идентификатор ОС, расположенной в разделе, и байт номера головки конца раздела, за которым идут два байта – номер сектора и номер цилиндра последнего сектора раздела.

Таблица 2.2

Название записи элемента Partition Table	Длина, байт
Флаг активности раздела	1
Номер головки начала раздела	1
Номер сектора и номер цилиндра загрузочного сектора раздела	2
Кодовый идентификатор операционной системы	1
Номер головки конца раздела	1
Номер сектора и номер цилиндра последнего сектора раздела	2
Младшее и старшее двухбайтовое слово относительного номера начального сектора	4
Младшее и старшее двухбайтовое слово размера раздела в секторах	4

Завершают элемент раздела младшее и старшее двухбайтовое слово относительного номера первого сектора раздела и размер раздела в секторах соответственно.

Для жестких дисков типичной является ситуация, когда имеется четыре записи в таблице разделов и, соответственно, четыре раздела. ОС MS-DOS использует только два из них, остальные резервируются для параллельного использования других ОС. Благодаря наличию такой структуры, как MBR, на одном физическом жестком носителе может располагаться несколько файловых систем различного типа разных операционных систем. Структуры MBR представляют собой важную информацию, повреждение которой приводит к частичной или полной потере доступа к данным логических устройств жесткого диска и к невозможности загрузки ОС с поврежденного носителя.

Логические разделы тоже имеют иерархическую структуру в зависимости от типа ОС и ее файловой системы.

Так, первый раздел жесткого диска в MS-DOS называется главным разделом (Primary Partition), а второй расширенным (Extended Partition). Главный раздел всегда должен присутствовать на диске, с него происходит загрузка MS-DOS. Расширенного раздела может не быть, он создается лишь в случае, когда необходимо получить более одного логического устройства на физическом диске. Логический раздел размещает в себе такие структуры файловой системы, как логические диски, устройства или тома (оформленные как подразделы), загрузчик операционной системы, таблицы распределения файлов, области пользовательских данных, в которых размещаются записи о каталогах и файлах и данные файлов. Число логических подразделов может быть более

четырех, а последний элемент каждого показывает, является ли он последним логическим подразделом раздела.

Основной единицей хранения информации в MS-DOS и других ОС с похожей логической структурой разделов является кластер (cluster) – группа секторов. В таких ОС для распределения минимального дискового пространства с точностью один байт выделяется целый кластер, содержащий много секторов и еще больше байтов (килобайты), что приводит к нерациональному использованию пространства ЖД для мелких файлов. Для доступа к каждому кластеру создается таблица соответствия номеров кластеров файлам логического раздела – таблица распределения файлов (File Allocation Table – FAT). Поэтому файловые системы такого типа называют FAT-системами. Это не самый оптимальный, но довольно быстрый способ организации информации на разделах, поэтому он «дожил» до наших дней с давних времен начала развития ПК, где использовался исключительно для накопителей на ГМД. Все остальные логические структуры – файлы или каталоги связаны локализацией с FAT.

Для других ОС, например UNIX (LINUX), использование разделов происходит иначе. Как правило, их может быть более четырех, все они равноправны и одинаково могут быть загрузочными, содержат собственные файловые системы на основе *i*-узлов. Такие файловые системы являются теговыми и не имеют таблиц распределения порций информации. Дисковое пространство распределяется посекторно, что дает максимально возможное использование пространства раздела, но несколько снижает производительность. Весь раздел разбивается на иерархически связанную цепочку узлов разного уровня, которым соответствует некоторое количество секторов. На основе узлов строится понятие файлов и каталогов, и в таких системах файлы и каталоги действительно не различаются, так как каталог является файлом, содержащим структуру узлов. Один раздел отводится для дискового свопа и имеет упрощенную структуру, так как никогда не содержит файлов и каталогов.

Все разделы могут содержать *загрузчик операционной системы*, который располагается, как правило, в первом секторе и занимает один сектор. В этом секторе располагаются структуры – записи, имеющие отношение лишь к конкретной ОС, и, следовательно, они могут отличаться для разных разделов и версий ОС. Многие специализированные программы (например, защиты данных, борьбы с вирусами и др.) могут изменять структуру или отдельные части загрузчика операционных систем. Загрузчик большинства персональных однопользовательских ОС является объектом воздействия вирусами, которые заражают загрузочные сектора жестких дисков.

2.2.5. Интерфейсы жестких дисков

Интерфейсом накопителей называется набор электроники, обеспечивающий обмен информацией между контроллером устройства (кеш-буфером) и компьютером. В настоящее время в настольных ПК чаще других используются две разновидности интерфейсов ATAPI (AT Attachment Packet Interface): Integrated Drive Electronics (IDE), Enhanced Integrated Drive Electronics (EIDE) и интерфейс SCSI (Small Computers System Interface).

Интерфейс IDE разрабатывался как недорогой и производительный интерфейс, предназначенный для подключения двух дисковых устройств. Отличительная особенность дисковых устройств, работающих с интерфейсом IDE, состоит в том, что собственно контроллер НЖМД располагается на плате самого накопителя вместе со встроенным внутренним кеш-буфером. Такая конструкция существенно упрощает устройство интерфейсной карты и дает возможность не только размещать ее на отдельной плате адаптера, вставляемой в разъем системной шины, но и интегрировать непосредственно на материнской плате компьютера. Интерфейс характеризуется простотой, высоким быстродействием, малыми размерами и относительной дешевизной.

Сегодня на смену интерфейсу IDE пришло его расширение фирмы Western Digital – Enhanced IDE (сокращенно EIDE). Сейчас это лучший вариант для подавляющего большинства настольных систем. Жесткие диски EIDE заметно дешевле аналогичных по емкости SCSI-дисков и в однопользовательских системах не уступают им по производительности, а большинство материнских плат имеют интегрированный двухканальный контроллер для подключения четырех устройств.

Можно выделить шесть основных отличий расширенного стандарта:

1) большая емкость дисков; если IDE не поддерживал диски свыше 528 Мбайт, то EIDE преодолевает это ограничение;

2) к нему подключается больше устройств – четыре вместо двух. Вместо одного канала контроллера для подключения двух IDE-устройств используются два: основной – на высокоскоростной локальной шине и вспомогательный;

3) появилась спецификация ATAPI, позволяющая подключать к этому интерфейсу не только жесткие диски, но и другие устройства – стриммеры и дисководы CD-ROM;

4) повысилась производительность: НЖМД с интерфейсом IDE имели максимальную скорость передачи данных на уровне 3 Мбайт/с, а диски EIDE поддерживают несколько новых режимов обмена, которые обеспечивают скорость передачи данных 11.1 и 16.6 Мбайт/с соответственно;

5) поддерживается режим прямого доступа к памяти: Mode 1 DMA (Direct Memory Access) или Mode 2 DMA и Ultra DMA, которые поддерживают обмен данными в монопольном режиме, при котором канал ввода-вывода в течение некоторого времени обслуживает только одно устройство;

6) расширена система команд управления устройством, передачи данных и диагностики, увеличен кеш-буфер обмена данными и существенно доработана механика.

Интеллектуальный многофункциональный интерфейс SCSI был разработан в конце 70-х гг. XX в. в качестве устройства сопряжения компьютера и интеллектуального контроллера дискового накопителя. Интерфейс SCSI является универсальным и определяет обмен данными между центральным процессором и несколькими внешними устройствами, имеющими свой контроллер. Помимо электрических и физических параметров определяются также команды, при помощи которых устройства, подключенные к шине, осуществляют связь между собой. Интерфейс SCSI поддерживает значительно более широкую гамму периферийных устройств и стандартизован ANSI (X3.131-1986).

Сегодня применяются в основном три стандарта:

– SCSI-2;

– Ultra SCSI;

– CSI-3.

В режиме Fast SCSI-2 скорость передачи данных доходит до 10 Мбайт/с в секунду при использовании 8-разрядной шины и до 20 Мбайт/с при 16-разрядной шине Fast Wide SCSI-2.

Появившийся позднее стандарт Ultra SCSI отличается еще большей производительностью – 20 Мбайт/с для 8-разрядной шины и 40 Мбайт/с для 16-разрядной.

В SCSI-3 увеличен набор команд, но быстродействие осталось на том же уровне.

Все применяющиеся сегодня стандарты совместимы с предыдущими версиями «сверху вниз», т. е. к адаптерам SCSI-2 и Ultra SCSI можно подключить старые SCSI-устройства. Интерфейсы SCSI-Wide, SCSI-2, SCSI-3 – стандарты модификации интерфейса SCSI разработаны комитетом ANSI. Общая концепция усовершенствований направлена на увеличение ширины шины до 32, с увеличением длины соединительного кабеля и максимальной скорости передачи данных с сохранением совместимости с SCSI. Это наиболее гибкий и стандартизованный тип интерфейса, применяющийся для подключения 7 и более периферийных устройств, снабженных контроллером интерфейса SCSI.

Интерфейс SCSI остается достаточно дорогим и самым высокопроизводительным из семейства интерфейсов периферийных устройств персональных компьютеров.

2.3. Электронные внешние ЗУ

2.3.1. Флеш-память

Флеш-память (flash memory) – разновидность полупроводниковой технологии электрически перепрограммируемой памяти (EEPROM). Является энергонезависимой памятью, характеризующейся большой емкостью, дешевизной, механической прочностью и скоростью работы. Основными недостатками данной технологии является ограниченный ресурс носителей и чувствительность к электростатическому разряду.

Флеш-память различается методом соединения ячеек в массив: NOR или NAND. Названия NOR и NAND произошли по ассоциации со схемами включения ячеек в массив в схемотехнике микросхем КМОП-логики.

Технология NOR использует классическую двумерную матрицу проводников, в которой на пересечении строк и столбцов установлено по одной ячейке. При этом проводник строк подключается к стоку транзистора, а столбцов – ко второму затвору. Исток подключается к общей для всех подложке.

Технология NAND – трехмерный массив. В основе та же самая матрица, что и в NOR, но вместо одного транзистора в каждом пересечении устанавливается столбец из последовательно включенных ячеек. В такой конструкции получается много затворных цепей в одном пересечении. Плотность компоновки можно резко увеличить (ведь к одной ячейке в столбце подходит только один проводник затвора), однако алгоритм доступа к ячейкам для чтения и записи заметно усложняется.

Технология NOR позволяет получить быстрый доступ индивидуально к каждой ячейке, однако площадь ячейки велика. Наоборот, NAND имеют малую площадь ячейки, но относительно длительный доступ сразу к большой группе ячеек. Соответственно, различаются области применения: NOR используется во встраиваемых системах как непосредственная память программ микропроцессоров и для хранения небольших вспомогательных данных. NAND чаще всего применяется для USB-флеш-памяти или SSD-накопителей.

Изменение заряда сопряжено с накоплением необратимых изменений в структуре и потому количество записей для ячейки флеш-памяти ограничено. Типичные количества циклов стирания-записи составляют от десятков и сотен тысяч до тысячи или менее, в зависимости от типа памяти и технологического

процесса. Гарантированный ресурс значительно более низок при хранении нескольких бит в ячейке. Одна из причин деградации – невозможность индивидуально контролировать заряд плавающего затвора в каждой ячейке. Дело в том, что запись и стирание выполняются над множеством ячеек одновременно – это неотъемлемое свойство технологии флеш-памяти. Постепенно заряд отдельных ячеек рассогласовывается и в некоторый момент выходит за допустимые границы, которые может скомпенсировать инжекцией автомат записи и воспринять устройство чтения.

Другая причина – взаимная диффузия атомов изолирующих и проводящих областей полупроводниковой структуры и электрические пробои изолятора при записи и стирании. Это приводит к уменьшению времени хранения заряда.

Все микросхемы флеш-памяти имеют ярко выраженную иерархическую структуру. Стирание, запись и чтение флеш-памяти всегда происходят относительно крупными блоками разного размера, при этом размер блока стирания всегда больше, чем блока записи, а размер блока записи не меньше, чем размер блока чтения. Это характерный отличительный признак флеш-памяти по сравнению с классической памятью EEPROM. Поэтому флеш-память разбивается на блоки, блоки состоят из секторов, секторы – из страниц. В зависимости от назначения конкретной микросхемы глубина иерархии и размер элементов могут меняться. Например, NAND-микросхема может иметь размер стираемого блока в сотни килобайт, размер страницы записи и чтения – 4 Кбайт. Для NOR-микросхем размер стираемого блока варьируется от единиц до сотен килобайт, размер сектора записи – до сотен байт, страницы чтения – единицы-десятки байт.

Скорость стирания у флеш-памяти варьируется от единиц до сотен миллисекунд в зависимости от размера стираемого блока. Скорость записи – десятки-сотни микросекунд. Обычно скорость чтения для NOR-микросхем нормируется в десятки наносекунд. Для NAND-микросхем скорость чтения составляет десятки микросекунд.

2.3.2. Твердотельная память SSD

Твердотельные накопители (ТТН) представляют собой устройства, хранящие данные в полупроводниковых микросхемах вместо вращающихся металлических дисков. В отличие от НЖМД у ТТН нет подвижных механических частей, а запись и считывание происходят по такому же принципу, как и у сменного USB-носителя информации. Причина их появления связана с тем, что скорость обработки данных в процессоре намного превышает скорость записи данных в НЖМД, которые стали «узким местом» в производительности

ВС. Твердотельные накопители за счет использования микросхем флеш-памяти обеспечивают намного большие скорости работы с данными по сравнению с жесткими дисками.

Основные достоинства твердотельных накопителей:

– малое время доступа к данным: от 100 до 1000 раз быстрее, чем у механических дисков;

– высокая скорость передачи, вплоть до нескольких гигабайт в секунду для произвольно расположенных данных;

– высокая надежность, ТТН дают уровень сохранности данных такой же, как другие полупроводниковые устройства;

– отсутствие шума и малый нагрев во время работы.

Основные недостатки ТТН:

– ограниченное по сравнению с НЖМД число циклов перезаписи информации;

– более высокая, чем у жестких дисков, цена, которая сильно зависит от доступной емкости из-за ограниченной плотности размещения ячеек памяти и ограничения размера кристалла в микросхеме;

– после внезапного выхода ТТН из строя снятие с него информации в отличие от классических НЖМД бывает сложно или вообще невозможно.

Существуют также гибридные жесткие диски (SSHD – solid-state hybrid drive), в которых память NAND используется совместно с магнитными пластинами. Подобное объединение позволяет воспользоваться частью преимуществ флеш-памяти (быстрый произвольный доступ) при сохранении небольшой стоимости хранения больших объемов данных. Так, технология Intel Smart Response позволяет совместно использовать SSD и HDD с целью кеширования часто используемых данных (файлов) на SSD.

2.4. Внешняя память на дисках CD и DVD

Устройство CD-диска. Стандартный компакт-диск (CD) состоит из трех слоев:

– основы;

– отражающего;

– защитного.

Основа выполнена из прозрачного поликарбоната, на котором методом прессования сформирован информационный рельеф. Поверх рельефа напыляется металлический отражающий слой (алюминий, золото, серебро, другие

металлы и сплавы). Отражающий слой покрывается сверху защитным слоем поликарбоната или нейтрального лака, чтобы вся металлическая поверхность была защищена от контакта с внешней средой.

Информационный рельеф диска представляет собой непрерывную спиральную дорожку, начинающуюся от центра и состоящую из последовательности углублений – питов (pits). Промежутки между питами носят название lands. Чередованием питов и промежутков различной длины на диске записывается закодированный цифровой сигнал: переход от промежутка к питу и наоборот обозначает единицу, а длина пита или промежутка – длину серии нулей. Расстояние между витками дорожки выбирается от 1.4 до 2 мкм, стандарт определяет расстояние в 1.6 мкм.

Способы записи и изготовления. Основной способ изготовления дисков – прессование с матрицы. Оригинал формируется с исходной цифровой мастер-ленты, содержащей уже подготовленный и закодированный цифровой сигнал, специальным высокоточным станком на стеклянном диске, покрытом слоем фоторезиста – материала, изменяющего свою растворимость под воздействием лазерного луча. При обработке записанного оригинала растворителем на стекле возникает требуемый рельеф, который методом гальванопластики переносится на никелевый оригинал (негатив), который может служить матрицей при мелкосерийном производстве либо основой для снятия позитивных копий, с которых, в свою очередь, снимаются негативы для массового тиражирования. Штамповка выполняется методом литья под давлением: с негативной матрицы прессуется поликарбонатная подложка с рельефом, сверху напыляется отражающий слой, который покрывается лаком.

Представление звукового сигнала. Исходный стереофонический звуковой сигнал подвергается оцифровке в 16-разрядные отсчеты (квантование) с частотой дискретизации 44.1 кГц. Общая длительность записи сигнала на компакт-диске – 74 мин. Записи состоят из фреймов по 588 отстчетов. В каждой секунде звукозаписи содержится по 75 фреймов.

Воспроизведение звука. При воспроизведении звуковой компакт-диск вращается с постоянной линейной скоростью относительно воспроизводящей головки (приблизительно 1.25 м/с). Система стабилизации скорости вращения обеспечивает скорость считанного цифрового потока равной 4.3218 Мбит/с. Угловая скорость диска при этом изменяется от 500 об/мин при чтении самых внутренних участков дорожки до 200 об/мин на самых внешних. Для считывания информации с диска используется полупроводниковый лазер с длиной

волны около 780 нм (инфракрасный диапазон). Луч лазера, проходя через фокусирующую линзу, падает на отражающий слой, отраженный луч попадает в фотоприемник, где происходит определение питов и промежутков.

CD-R и CD-RW. Система однократной (CD-Recordable – записываемый CD) и многократной (CD-Rewritable – перезаписываемый CD) записи компакт-дисков. Терминами CD-R и CD-RW обозначаются как устройства для записи, так и сами диски.

Для однократной записи используется так называемая болванка, представляющая собой компакт-диск, в котором отражающий слой выполнен преимущественно из золотой или серебряной пленки, а между ним и поликарбонатной основой расположен регистрирующий слой из органического материала, темнеющего при нагревании. В процессе записи лазерный луч нагревает выбранные точки слоя, которые темнеют и перестают пропускать свет к отражающему слою.

В перезаписываемых дисках используется промежуточный слой из органической пленки, изменяющей под воздействием луча свое фазовое состояние с аморфного на кристаллическое и обратно, в результате чего меняется прозрачность слоя. Существующие диски выдерживают от тысяч до десятков тысяч циклов перезаписи. Однако их отражающая способность существенно ниже штампованных и однократных CD, что затрудняет их считывание в обычных приводах. Для чтения CD-RW формально необходим привод с автоматической регулировкой усиления фотоприемника.

Запись дисков CD-R выполняется при помощи специальных программ – Easy CD, CD Creator, CD Publisher, Direct CD и т. п.

Приводы CD-ROM. Типовой привод состоит:

– из платы электроники;

– шпиндельного двигателя;

– системы оптической считывающей головки;

– системы загрузки диска.

Интерфейсы приводов CD-ROM. В настоящее время CD-ROM выпускаются только с интерфейсами SCSI и IDE. Диск подключается непосредственно к магистрали SCSI или IDE (ATA) с заданием номера устройства для SCSI или Master/Slave для IDE.

DVD. Первоначально сокращение DVD расшифровывалось как digital video disc – оптические диски с большой емкостью. Эти диски используются для хранения компьютерных программ и приложений, а также полнометражных

фильмов и высококачественного звука. Появившаяся позже расшифровка аббревиатуры DVD – digital versatile disc, т. е. универсальный цифровой диск, более логична.

Хотя DVD выглядят как обычные диски CD-ROM, они могут хранить в 26 раз больше данных. Однослойный, односторонний диск DVD может хранить 4.7 Гбайт данных, двухслойный, односторонний – до 8.5 Гбайт, двухсторонний – до 17 Гбайт.

Как и CD-ROM, диски DVD хранят данные за счет насечек, расположенных вдоль спиральных треков на отражающей металлической поверхности, покрытой пластиком. Используемый в устройствах чтения DVD-дисков лазер скользит вдоль треков по насечкам, а отраженный луч интерпретируется приемным устройством в виде единиц или нулей. Благодаря разработке более высокочастотного полупроводникового лазера с меньшей длиной волны стало возможным использовать насечки меньшего размера. Для записи видео и звука на DVD применяется компрессия данных, носящая название MPEG-2 или MPEG-4. Это увеличивает временную емкость записываемых данных.

3. ПРИНЦИПЫ ОРГАНИЗАЦИИ RAID-МАССИВОВ

Назначение. В переводе с английского RAID (Redundant Array of Independent Disks) означает «избыточный массив независимых дисков». Впервые термин RAID появился в 1987 г., когда исследователям из Калифорнийского университета в Беркли удалось создать массив из нескольких жестких дисков [3].

Предназначение RAID – создание на базе нескольких жестких дисков сравнительно небольшой емкости одного логического диска:

– с большой емкостью;

– увеличенной скоростью доступа;

– увеличенной надежностью хранения;

– возможностью восстановления данных при отказе части оборудования.

Именно эти обстоятельства сделали RAID-массивы столь востребованными бизнесом и военными. Впрочем, за объем, скорость и надежность пришлось платить повышением стоимости и сложности систем хранения данных. Со временем оборудование для построения RAID-массивов стало более доступным с появлением дешевых решений для дисков IDE/ATA и SATA.

Найти оптимальное решение одновременно по надежности, скорости, емкости и цене дисковой памяти непросто. Надо быть готовым к тому, что при-

дется купить не один, а несколько жестких дисков, и емкость как минимум одного из них не будет использоваться. Также потребуется специальная плата контроллера и соответствующее программное обеспечение.

3.1. Основные принципы построения RAID-массивов

В основе теории RAID лежат следующие принципы:

– массив (Array);

– зеркалирование (Mirroring) или дублирование;

– чередование полос (Striping);

– контроль четности (Parity).

Массивом называют несколько накопителей, которые централизованно настраиваются, форматируются и управляются. Логический массив – это уже более высокий уровень представления, на котором не учитываются физические характеристики системы. Соответственно, логические диски могут по количеству не совпадать с физическими. Для операционной системы вообще весь массив является одним большим логическим диском.

Зеркалирование – технология, позволяющая повысить надежность системы. В RAID-массиве с зеркалированием все данные одновременно пишутся не на один, а на два жестких диска, т. е. создается «зеркало» данных. При выходе из строя одного из дисков вся информация остается сохраненной на втором.

За такую стопроцентную защиту приходится дорого платить: считайте, что один винчестер у вас работает просто так, не увеличивая доступную емкость. При этом нет никакого выигрыша в производительности. Столь дорогое решение используется только во внешних RAID-массивах, предназначенных для ответственных приложений.

Чередование полос – отличная возможность повысить быстродействие системы. Очевидно, если чтение и запись вести параллельно на нескольких жестких дисках, можно получить выигрыш в скорости. Как это делается? Записываемый файл разбивается на части определенного размера (полосы – strip) и одновременно размещается на всех имеющихся накопителях в последовательном порядке. В таком фрагментированном виде файл и хранится. Считывание и запись соседних полос выполняются параллельно с разных дисков.

Размер «кусочка» может быть минимальным – 1 байт, но чаще используют более крупное дробление – по 512 байт (размер сектора).

Контроль четности является альтернативным решением, соединяющим в себе достоинства зеркалирования (высокая надежность) и чередования (высокая скорость работы).

Используется тот же принцип, что и в контроле четности оперативной памяти. Если имеется *i* блоков данных и на их основе вычисляется еще один дополнительный *экстраблок*, из получившихся (*i* + 1) блоков всегда можно восстановить информацию даже при повреждении одного из них. Соответственно, для создания нормального RAID-массива в этом случае требуется (*i* + 1) жесткий диск.

Распределение блоков по дискам точно такое же, как при чередовании. Экстраблок может записываться на отдельный накопитель либо раскидываться по дискам.

Каждый бит экстраблока равен результату выполнения логической операции XOR над соответствующими битами всех *i* блоков. XOR – удивительный оператор, при его повторном наложении можно получить первоначальный результат, т. е. (A XOR B) XOR B = A. Это правило распространяется на любое количество операндов.

Плюсы четности очевидны. За счет использования чередования повышается скорость работы. Повышение надежности за счет зеркалирования здесь делается так, что при этом «нерабочий» объем массива заметно уменьшается, он одинаков при любом количестве дисков и составляет емкость одного диска, т. е. при 5 дисках в массиве пропадает всего 20 % емкости.

Но у четности есть весомый минус. Для формирования экстраблоков требуются вычисления! Их надо делать на лету, причем с миллионами битов! Если это дело поручить центральному процессору, получим очень медленную систему. Необходимо использовать довольно дорогие платы с RAID-контроллерами, которые «берут все вычисления на себя». В случае выхода из строя одного из дисков процесс восстановления будет не столь быстрым, как при зеркалировании.

3.2. Одиночные уровни RAID

Обычно в классификациях систем уровни связаны с иерархией. В RAID-уровнях такой связи нет. RAID 4 не есть улучшенная модификация RAID 3, а RAID 5 не лучше RAID 1 – просто они разные. При этом есть как простые (single), так и составные (multiple) RAID-массивы. Составные являются сочетанием двух простых. Сначала рассмотрим 7 простых уровней.

RAID 0 – простейший массив, использующий чередование полос без четности. Вся входящая информация разбивается на блоки фиксированной длины (например, 16 Кбайт) и раскидывается на все имеющиеся диски (рис. 3.1).

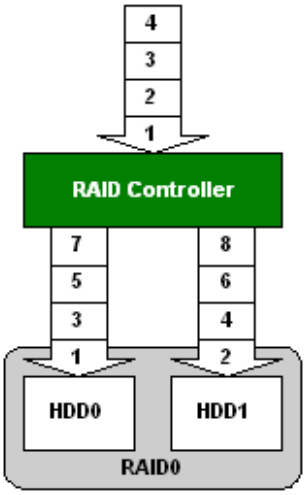

Рис. 3.1

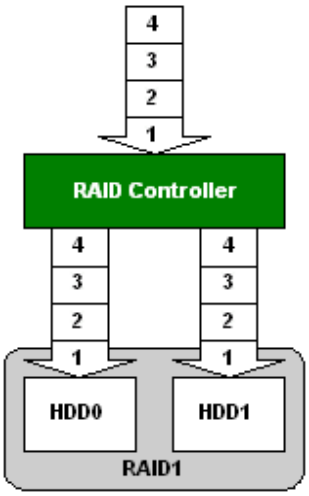

Рис. 3.2

При наличии двух-четырех дисков RAID 0 дает ощутимый выигрыш в скорости передачи данных, но совершенно не обеспечивает надежность. Для его построения подойдет любой дешевый RAID-контроллер. Подходит для тех, кому нужно выжать максимум производительности из файловой системы при минимальных затратах.

RAID 1 – уровень, являющийся обычным зеркалированием (рис. 3.2). На два жестких диска пишутся две одинаковые копии данных. При этом можно использовать дешевый RAID-контроллер или его программную реализацию. RAID 1 позволяет надежно защитить данные и обеспечить работу системы даже при отказе одного из дисков. Выигрыша в скорости при использовании RAID 1 нет.

RAID 2 – уровень, который сейчас не используется (рис. 3.3). В нем предполагалось использовать две технологии – побитовое чередование и код Хэмминга для восстановления ошибок. При этом часть дисков используется для

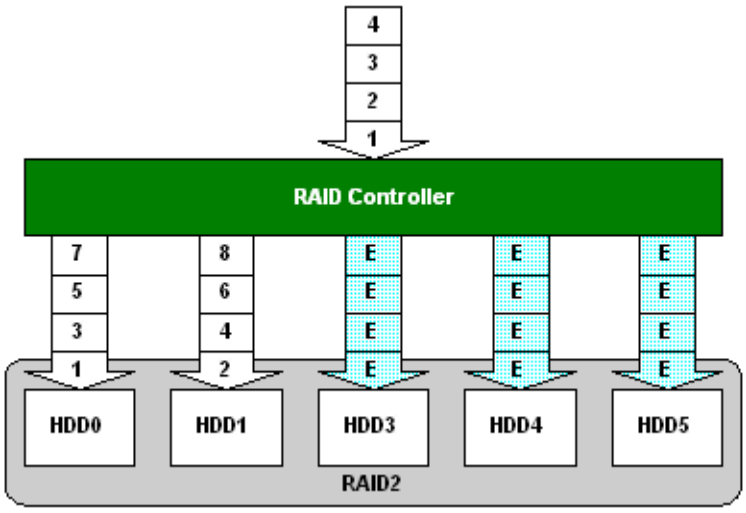

Рис. 3.3

хранения данных с чередованием, остальные – для хранения вычисленных контрольных сумм. Реализация RAID 2 требует увеличения числа дисков и специальных дорогих контроллеров, поэтому применения не нашла.

RAID 3 – уровень, использующий чередование полос и выделенный диск для контроля четности (рис. 3.4). Блоки данных обычно имеют длину не более 512 байт. Информация распределяется на несколько дисков, а вычисленные значения контроля по четности сохраняются на отдельном диске. Все скоростные преимущества чередования снижаются необходимостью записывать контрольную сумму на выделенный диск. К достоинствам следует отнести возможность работы массива при отказе одного из дисков.

RAID 4 – уровень, отличающийся от RAID 3 только размером блока данных при чередовании (не менее 2 физических блоков диска) (рис. 3.5). Это улучшает работу массива при случайном чтении, но запись довольно медленная. Диск с контрольными суммами является ярко выраженным «узким местом» в системе. RAID 4 является компромиссным вариантом между RAID 3 и RAID 5, не нашел своего места на рынке и редко используется.

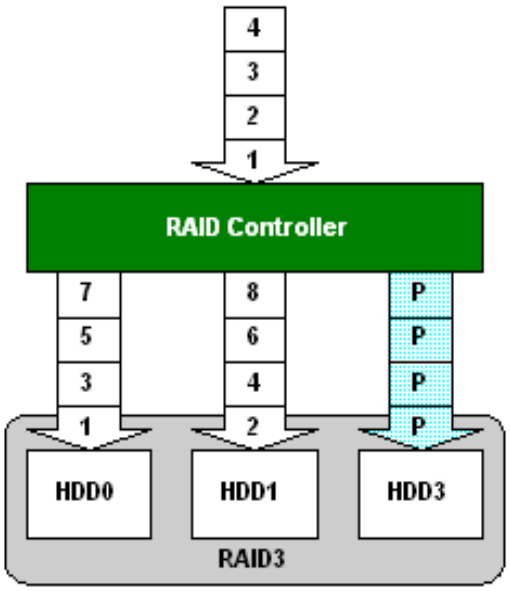

Рис. 3.4

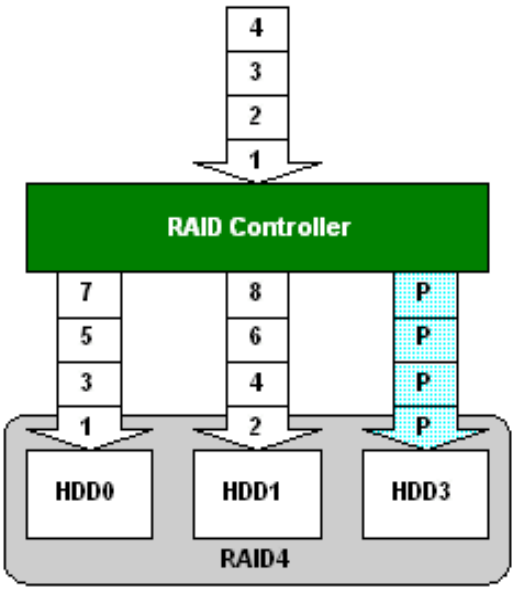

Рис. 3.5

RAID 5 – наиболее распространенный в системах хранения данных уровень (рис. 3.6). Он характеризуется применением чередования полос и контроля четности. В отличие от RAID 3 контрольные суммы не хранятся на одном диске, а распределяются по всем, что позволяет значительно повысить скорость записи. Главный принцип распределения экстраблоков – они не должны располагаться на том же диске, который учитывался при вычислении паритета.

Надежность и скорость работы такой системы оказываются очень высокими. При восстановлении информации всю работу на себя берет RAID-контроллер, так что операция проходит довольно быстро.

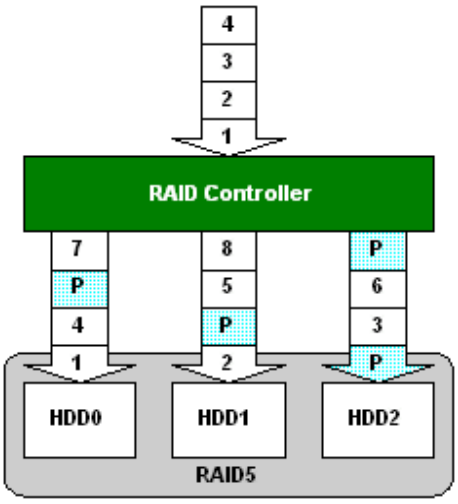

Рис. 3.6

RAID 6. Для некоторых особо критичных приложений требуется повышенная надежность. Например, чтобы при выходе из строя даже двух дисков массив сохранил данные и остался работоспособным, это обеспечивается на уровне RAID 6. При этом используются все те же технологии чередования полос и контроля четности, но в отличие от RAID 5 контрольная сумма вычисляется два раза и копируется на два разных диска. В итоге данные окажутся потерянными только в случае выхода из строя сразу двух жестких дисков. По сравнению с RAID 5 это более дорогое и медленное решение: увеличивается время на вычисление и запись паритетной информации, требуется дополнительное дисковое пространство. На практике RAID 6 почти не используется, так как выход из строя сразу двух дисков – слишком редкий случай, а повысить надежность можно другими способами.

3.3. Составные уровни RAID-массивов

У основных уровней RAID есть свои достоинства и недостатки. Поэтому инженеры решили создать RAID, который бы объединял достоинства нескольких уровней. Составной RAID-массив – это обычно сочетание быстрого RAID 0 с надежным RAID 1, 3 или 5. Итоговый массив действительно обладает улучшенными характеристиками, но и платить за это приходится повышением стоимости и сложностью решения.

Составной RAID строится так: сначала диски разделяются на наборы (set). Затем на основе каждого из наборов строятся простые массивы. А завершается все объединением этих массивов в один мегамассив. Запись типа X + Y означает, что сначала диски объединены в RAID уровня X, а затем несколько массивов RAID X объединены в RAID уровня Y.

RAID 0 + 1 (01) и 1 + 0 (10). RAID 0 + 1 (рис. 3.7) часто называют «зеркалом страйпов», а RAID 1 + 0 (рис. 3.8) – «страйпом зеркал» (русское «чередование» практически не используется, сменившись англоязычным). В обоих случаях используются две технологии – чередование и зеркалирование, но результаты разные.

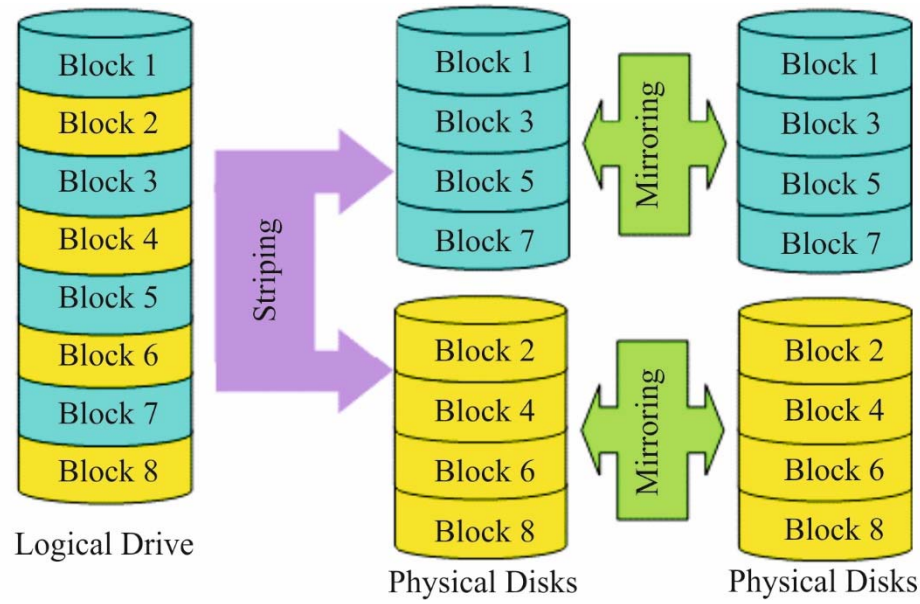

Рис. 3.7

RAID 0 + 1 обладает высокой скоростью работы и повышенной надежностью и является недорогим решением, но по надежности несколько лучше RAID 1 + 0. Основной недостаток массивов 1 + 0 – низкий процент использования емкости накопителей – всего 50 %.

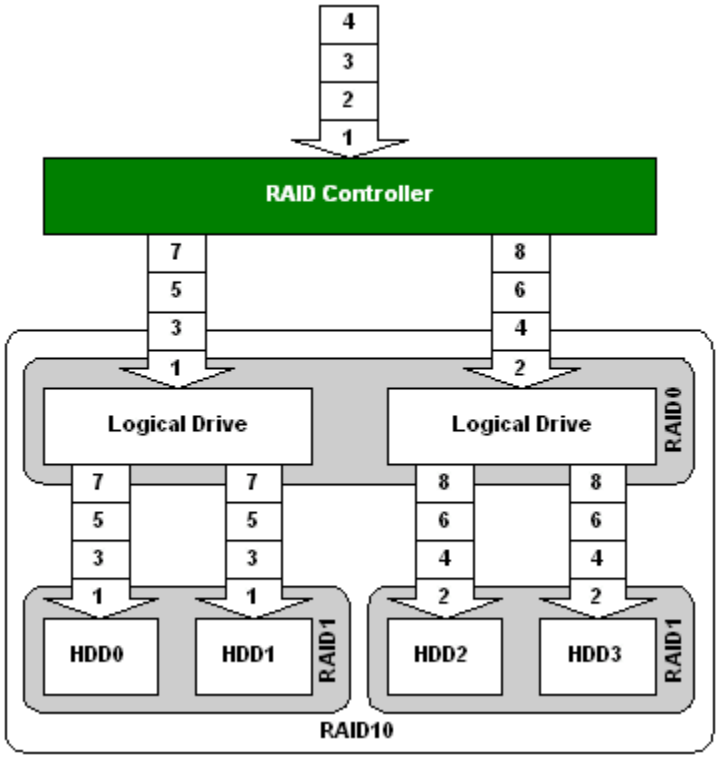

Рис. 3.8

RAID 0 + 3 (03) и 3 + 0 (30). По идее, сочетание чередования и RAID 3 дает выигрыш в скорости, но он довольно мал. Зато система заметно усложняется. Наиболее простой уровень 3 + 0. Из двух массивов RAID 3 строится

страйп, и минимальное количество требуемых дисков – 6. Получившийся RAID 3 + 0 с точки зрения надежности лучше, чем 0 + 3.

Достоинства этих комбинаций – довольно высокий процент использования емкости дисков и высокая скорость чтения данных. Недостатки – высокая цена, сложность системы.

RAID 0 + 5 (05) и 5 + 0 (50). Что будет, если объединить чередование с распределенной четностью с обыкновенным чередованием? Получится быстрая и надежная система. RAID 0 + 5 представляет собой набор страйпов, на основе которых построен RAID 5. Такая комбинация используется редко, так как практически не дает выигрыша ни в чем. Широкое распространение получил составной массив RAID 5 + 0 (рис. 3.9).

Чаще всего это два массива RAID 5, объединенных в страйп. Такая конфигурация позволяет получить высокую производительность при работе с файлами малого размера. Типичный пример – использование в качестве веб-сервера.

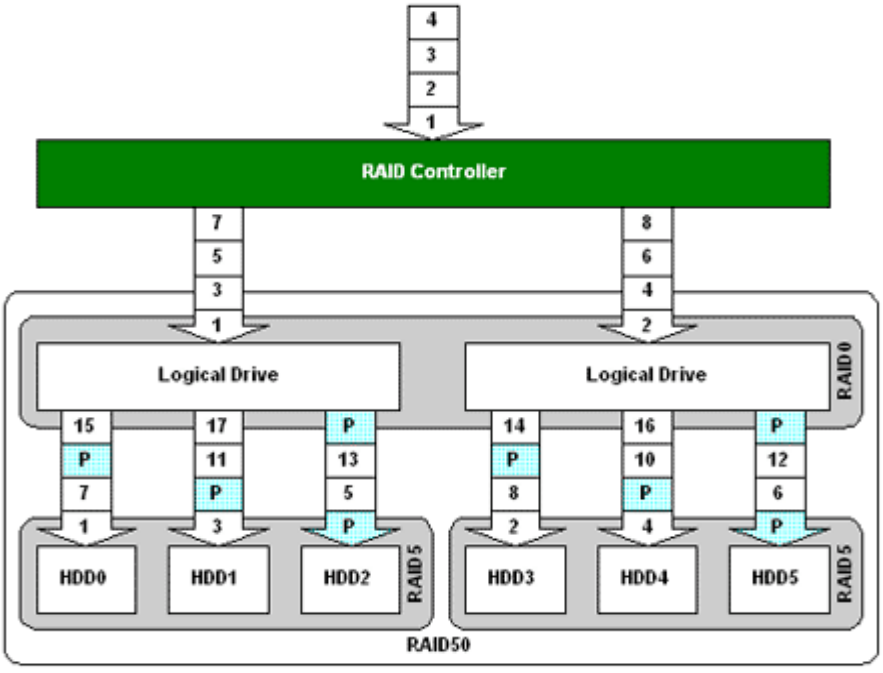

Рис. 3.9

RAID 1 + 5 (15) и 5 + 1 (51) – уровень, построенный на сочетании зеркалирования и чередования с распределенной четностью. Основная цель RAID 15 и 51 – значительное повышение надежности. Массив 1 + 5 продолжает работать при отказе трех накопителей, а 5 + 1 – даже при потере пяти из восьми жестких дисков! Платить приходится большим количеством неиспользуемой емкости дисков и общим удорожанием системы.

Чаще всего для построения RAID 5 + 1 используют два контроллера RAID 5, которые зеркалируют на программном уровне, что позволяет снизить затраты.

3.4. Сравнительные результаты

Распространенные single RAID-массивы

Характеристика	RAID 0	RAID 1	RAID 3	RAID 5	RAID 6
Технология	Чередование	Зеркалирование	Чередование, четность	Чередование, четность	Чередование, четность
Контроллер	Все	Все	Аппаратный	Аппаратный Hi-End	Специализированный
Кол-во дисков	2, 4	2	3 и больше	3 и больше	3 и больше
Доступное рабочее пространство, %	100	50	66 для 3, 75 для 4	66 для 3, 75 для 4	33 для 3 50 для 4 60 для 5
Стойкость при отказе диска	Нет	Высокая	Высокая	Высокая	Очень высокая
Восстановление данных	Нет	Быстрое	Быстрое	Быстрое	Очень быстрое
Скорость случайного чтения	Высокая	Хорошая	Хорошая	Очень хорошая	Очень хорошая
Скорость случайной записи	Высокая	Хорошая	Плохая	Нормальная	Плохая
Скорость линейного чтения	Высокая	Хорошая	Очень хорошая	Очень хорошая	Хорошая
Скорость линейной записи	Высокая	Хорошая	Хорошая	Хорошая	Средняя
Цена	Самая низкая	Низкая	Средняя	Средняя	Высокая

Распространенные multi-RAID-массивы

Характеристика	RAID 0 + 1	RAID 1 + 0	RAID 5 + 0	RAID 5 + 1
Технология	Чередование, зеркалирование	Чередование, зеркалирование	Чередование, четность	Чередование, четность, зеркалирование
Контроллер	Почти все	Почти все	Специализированный	Специализированный
Кол-во дисков	4 min	4 min	6 min	6 min
Доступное рабочее пространство, %	50	50	66 для 2 страйпов по 3 диска	33–40
Стойкость при отказе диска	Очень хорошая	Отличная	Хорошая	Отличная
Восстановление данных	Быстрое	Очень быстрое	Среднее	Быстрое

Окончание таблицы

Характеристика	RAID 0 + 1	RAID 1 + 0	RAID 5 + 0	RAID 5 + 1
Скорость случайного чтения	Очень хорошая	Очень хорошая	Очень хорошая	Очень хорошая
Скорость случайной записи	Хорошая	Хорошая	Хорошая	Хорошая
Скорость линейного чтения	Очень хорошая	Очень хорошая	Очень хорошая	Очень хорошая
Скорость линейной записи	Хорошая	Хорошая	Хорошая	Хорошая
Цена	Относительно высокая	Относительно высокая	Высокая	Очень высокая

4. НАЗНАЧЕНИЕ И ИСПОЛЬЗОВАНИЕ ШИН В ЭВМ

4.1. Общие положения и состав шин

Компьютер состоит из множества различных компонентов, часть из которых размещена на материнской (системной) плате: процессор, оперативная память, жесткий диск, видеокарта, контроллеры, а также разъемы для подключения внешних периферийных устройств (далее ВУ) – таких как экран, мышь, клавиатура, подключаемые флешки и т. д. Всеми этими компонентами должен управлять процессор, передавать и получать данные, отправлять сигналы, изменять состояние. Для этого компоненты ЭВМ должны быть связаны между собой системой проводников (линий), по которым происходит обмен информацией. Этот набор электрических проводов, собранных вместе, называют *шиной* [3]. Шина, связывающая только два устройства, по принципу точка-точка, называется *портом*. Но обычно к шине по одному набору проводников можно подключить несколько ВУ, для чего в ней имеются специальные места (разъемы), называемые *слотами*.

Архитектура любой **шины** имеет следующие компоненты:

– *линии для обмена данными* (шина данных). Шина данных обеспечивает обмен данными между процессором, памятью и картами расширения, установленными в слоты. Чем выше разрядность шины, тем больше данных может быть передано за один такт и тем выше производительность ПК. Компьютеры с процессором семейства Pentium имеют 64-разрядную шину данных;

– *линии для адресации данных* (шина адреса). Шина адреса служит для указания адреса какого-либо устройства, с которым процессор осуществляет обмен данными. Каждый компонент ПК, каждый порт ввода-вывода и ячейка памяти имеют свой адрес;

– *линии управления* (шина управления). По шине управления передаются сигналы: записи/считывания, готовности к приему/передаче данных, подтвер-

ждение приема данных, аппаратного прерывания и др. Все сигналы шины управления служат для обеспечения передачи данных;

– *линии питания* (шины питания) обеспечивают электрическое питание элементов шин;

– *контроллер шины* осуществляет управление процессом обмена данными и служебными сигналами и обычно выполняется в виде отдельной микросхемы либо набора микросхем, называемого Chipset.

Операции на шине называются *транзакциями*. Основные виды – транзакции чтения и транзакции записи или транзакции ввода и транзакции вывода. Шинная транзакция состоит из двух частей: посылка адреса и прием (или посылка) данных.

Связанные с шиной устройства должны работать по определенным правилам, которые называют *протоколами шины*. Некоторые устройства, связанные с шиной, являются активными и могут инициировать передачу информации по шине, а другие – пассивными и ждут запросов. Активное устройство называют *задающим* (bus master), пассивное – *подчиненным* (bus slave). Процессор является задающим устройством, если он требует от контроллера считать или записать информацию. В этом случае контроллер является пассивным устройством. Контроллер становится задающим устройством, если он командует приемом слов в память, ранее считанных им с диска.

Механизм, обеспечивающий как связь, так и взаимодействие устройств компьютера, реализуется с помощью унифицированной совокупности средств связи – *интерфейсов*, которые требуют стандартизации, распространяемой на форматы передаваемых данных, команды, наборы шин, алгоритмы, сигналы и т. д. Интерфейс – это совокупность унифицированных *шин для передачи информации, электронных схем, управляющих прохождением сигналов по шинам, и алгоритмов, управляющих обменом информации*.

Интерфейсы подразделяют на односвязные и многосвязные. Односвязные интерфейсы используют единственную центральную (системную) шину – магистраль, к которой подсоединяются все устройства на основе принципа разделения времени. Так как несколько устройств могут одновременно стать ведущими и пытаться захватить шину, то магистраль должна использоваться в режиме разделения времени. Ведущие устройства снабжаются приоритетом использования магистрали.

Подобная организация имеет два основных преимущества:

– низкую стоимость,

– универсальность.

Поскольку такая шина является единственным местом подсоединения для разных устройств, новые устройства могут быть легко добавлены в систему. Стоимость такой организации получается достаточно низкой, поскольку для реализации множества путей передачи информации используется единственный набор линий шины, разделяемый множеством устройств. Главным недостатком организации с единственной шиной является то, что *шина является узким местом*, ограничивающим максимальную пропускную способность передачи данных.

В коммерческих системах, где обмен данными осуществляется часто, а также в суперкомпьютерах, где необходимые скорости передачи очень высоки из-за высокой производительности процессора, одним из главных вопросов связи устройств является создание системы нескольких шин, способной удовлетворить все запросы. Использование для связи устройств в компьютере нескольких независимых систем шин характерно для многосвязных интерфейсов.

4.2. Виды шин и способы их иерархической организации

Шины в ПК различаются по своему функциональному назначению:

– *системная шина* используется микросхемами Chipset для пересылки информации к процессору и обратно;

– *шина кеш-памяти* предназначена для обмена информацией между процессором и кеш-памятью (уровня *L*2);

– *шина памяти* используется для обмена информацией между оперативной (основной) памятью и процессором;

– *шины ввода-вывода* используются для обмена информацией с периферийными устройствами.

Обычно различные виды шин объединены по иерархическому принципу. Пример укрупненной структуры соединения компонентов ЭВМ с помощью системы шин показан на рис. 4.1, где представлены:

– *шина кеш-памяти* – самая быстрая шина, которая соединяет процессор и *L*2-кеш, называемая *задней шиной* (BSB – backside bus);

– *шина памяти*, предназначенная для доступа к основной памяти и называемая *передней шиной* (FSB – frontside bus). Наличие двух шин связи с памятью (в процессорах Pentium II и выше) повышает производительность, так как процессор может одновременно получать данные с обеих шин. Скорость шины BSB обычно выше скорости шины FSB;

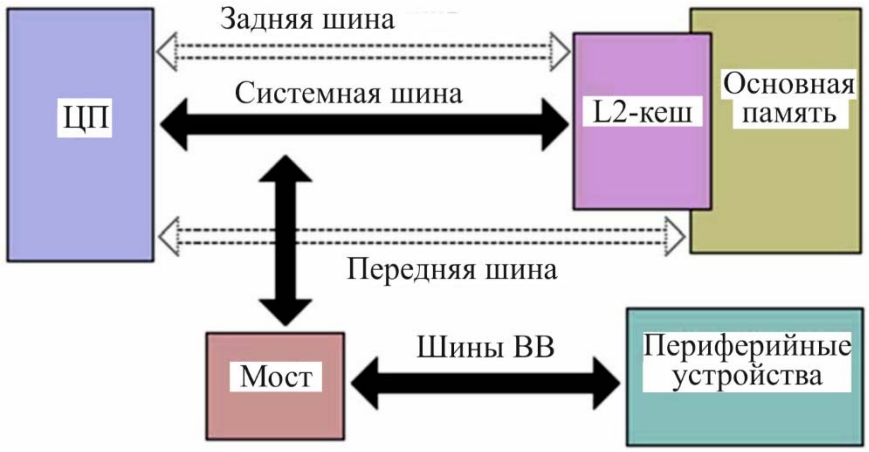

Рис. 4.1

– *системная шина*, обеспечивающая соединение процессора и подсистемы памяти с чипсетом и другими устройствами ЭВМ. Эта шина до 1998 г. работала с частотой синхронизации 66 МГц, а затем частота шины была повышена до 133 МГц и более;

– *локальная шина ввода-вывода* (ШВВ) – быстродействующая шина, используемая для подключения быстрых периферийных устройств к памяти, чипсету и процессору. Такую шину используют видеокарты, дисковые накопители и сетевые интерфейсы. Наиболее распространенными локальными ШВВ являются VESA Local Bus (VLB), AGP (Accelerated Graphics Port) – для связи видеопроцессора с системной памятью ПК;

– *стандартная шина ввода-вывода*, применяемая для медленных ВУ (клавиатура, мышь, модем и др.), а также для совместимости со старыми устройствами. Почти во всех ПК такой шиной является шина ISA (Industry Standard Architecture – стандартная промышленная архитектура).

Наряду с шиной ISA в современных ПК широко применяются шины:

– PCI (Peripheral Component Interconnect – взаимосвязь периферийных компонентов);

– USB (Universal Serial Bus) – универсальная последовательная шина, позволяющая подключать до 127 медленных ВУ с использованием транспортного узла *хаба* (hub);

– IEEE 1394 (FireWire) – скоростная последовательная шина, предназначенная для подключения к ПК цифровых камер, принтеров, телевизоров и других устройств, требующих высокой пропускной способности.

Несколько шин ввода-вывода, соединяющие различные периферийные устройства с процессором, подключаются к системной шине с помощью *моста* (bridge), содержащего схемы для объединения шин и взаимодействия

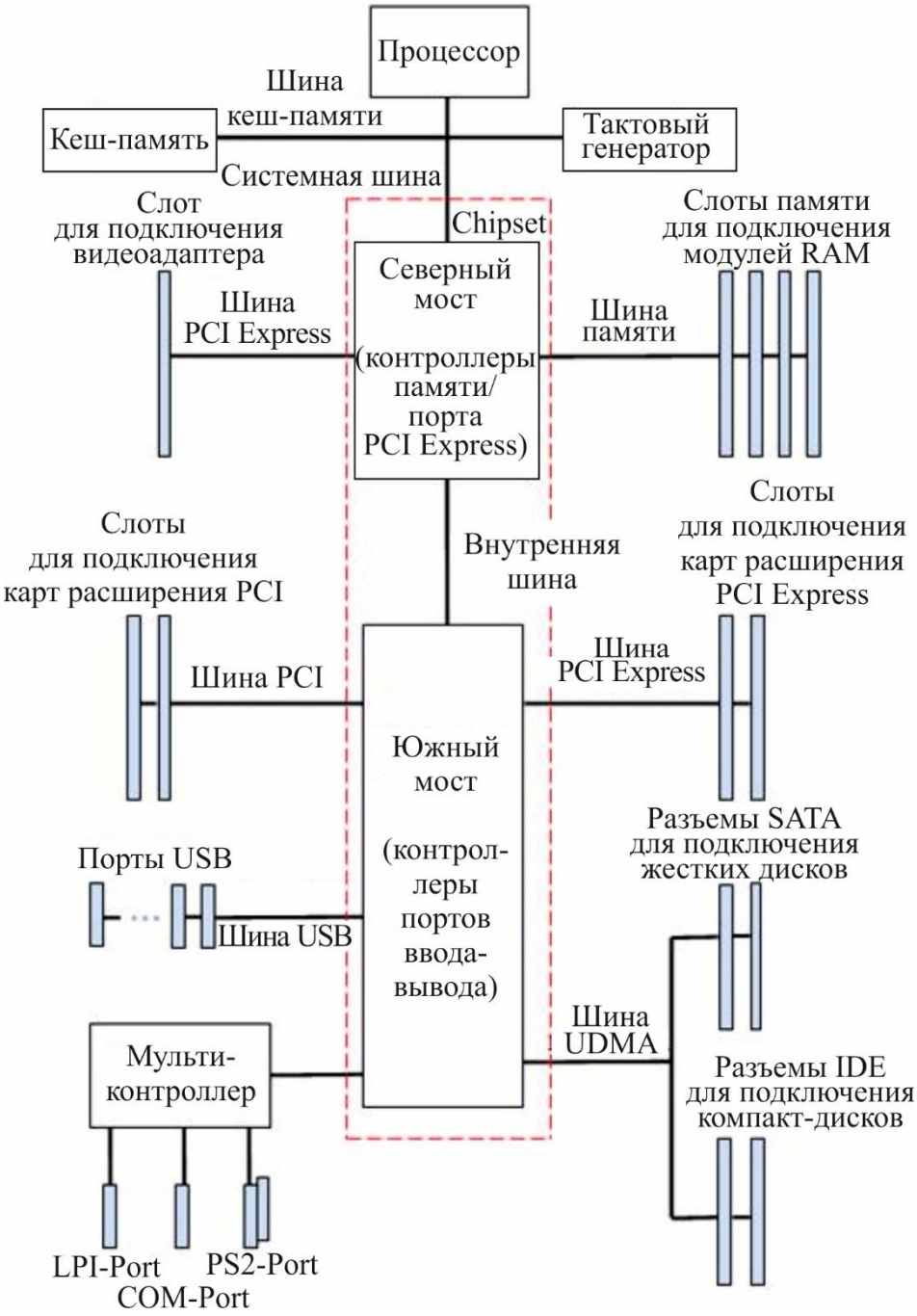

Рис. 4.2

устройств на одной шине с устройствами на другой шине. Мосты реализуются в системном чипсете, как это показано на рис. 4.2, и с их помощью чипсет управляет всеми шинами и обеспечивает правильное взаимодействие всех устройств в ЭВМ.

4.3. Основные характеристики шин

Разрядность (*ширина*) *шины* определяется числом параллельных проводников, входящих в нее. Первая шина ISA для IBM PC была 8-разрядной, т. е. по ней можно было одновременно передавать 8 бит. Используемая сейчас уни-

версальная шина ISA имеет ширину 16. Другие шины ввода-вывода, включая VLB и PCI, имеют ширину 32 бит. Ширина системной шины в ПК с процессорами Pentium IV составляет 64 бит.

Разрядность шины адреса можно определять независимо от разрядности шины данных. Разрядность шины адреса показывает, сколько ячеек памяти можно адресовать при передаче данных. В современных ПК разрядность шины адреса составляет 36 бит, что обеспечивает адресацию памяти емкостью 64 Гбайт.

Скорость шины (bus speed) показывает, сколько битов информации можно передавать по каждому проводнику шины в секунду. Большинство шин передают по одному проводнику 1 бит в такте синхронизации, хотя некоторые шины, например AGP, могут передавать 2 бит данных в такте синхронизации, что удваивает производительность.

Пропускная способность шины определяется количеством байтов информации, передаваемых по шине за секунду. Для определения пропускной способности шины необходимо умножить тактовую частоту шины на ее разрядность. Например, если разрядность шины 64, а тактовая частота 66 МГц, то пропускная способность равна 8 (байт) * 66 МГц = 528 Мбайт/с.

Частота шины – это тактовая частота, с которой происходит обмен данными по шине. Современные шины имеют частоту от 8,3 (ISA) до 133 МГц и даже до 377 МГц.

Одна из причин трудностей, возникающих при разработке шин, заключается в том, что максимальная скорость шины главным образом лимитируется физическими факторами:

– длиной шины;

– количеством подсоединяемых устройств.

Указанные физические ограничения не позволяют произвольно ускорять шины. Требования малой задержки, высокой пропускной способности и подключения разных устройств являются противоречивыми. В современных крупных системах используется целый комплекс взаимосвязанных шин, каждая из которых обеспечивает упрощение взаимодействия различных подсистем и высокую пропускную способность.

Как отмечалось ранее, шины делятся:

– на шины, обеспечивающие организацию связи процессора с памятью;

– шины ввода-вывода;

– системные шины.

Шины ввода-вывода могут иметь большую протяженность, поддерживать подсоединение многих типов устройств и обычно следуют одному из шинных стандартов.

Шины процессор-память, с другой стороны, сравнительно короткие, обычно высокоскоростные и соответствуют организации системы памяти для обеспечения максимальной пропускной способности канала память-процессор. На этапе разработки системы для шины процессор-память заранее известны все типы и параметры устройств, которые должны соединяться между собой, в то время как разработчик шины ввода-вывода должен иметь дело с устройствами, различающимися по задержке и пропускной способности.

С целью снижения стоимости некоторые компьютеры имеют единственную шину для памяти и устройств ввода-вывода. Такая шина часто называется системной. Персональные компьютеры, как правило, строятся на основе одной системной шины в стандартах ISA или PCI. Необходимость сохранения баланса производительности по мере роста быстродействия микропроцессоров привела к двухуровневой организации шин в персональных компьютерах на основе системной и локальной шин.

Разработка системы шин связана с обеспечением ряда функциональных возможностей, характеризующихся определенными параметрами (табл. 4.1).

Таблица 4.1

Возможность	Высокая производительность	Низкая стоимость
Общая разрядность шины	Отдельные линии адреса и данных	Мультиплексирование линий адреса и данных
Ширина (разрядность) данных	Чем шире, тем быстрее (например, 64 бит)	Чем уже, тем дешевле (например, 8 бит)
Размер пересылки	Пересылка нескольких слов имеет меньшие накладные расходы	Пересылка одного слова дешевле
Режим и способ передачи данных	Дуплексный, параллельный	Симплексный, полудуплексный, последовательный
Главные устройства шины	Несколько (требуется арбитраж)	Одно (арбитраж не нужен)
Наличие расщепленных транзакций	Да – отдельные пакеты *запроса* и *ответа* дают большую полосу пропускания (но нужно несколько главных устройств)	Нет – продолжающееся соединение дешевле и имеет меньшую задержку
Тип синхронизации	Синхронные	Асинхронные

Решение о выборе той или иной возможности зависит от целевых параметров стоимости и производительности. Первые три возможности являются очевидными:

– раздельные линии адреса и данных;

– более широкие (имеющие большую разрядность) шины данных;

– режим групповых пересылок (пересылки нескольких слов).

Они дают увеличение производительности за счет увеличения стоимости. Далее на скорость работы шины оказывают влияние режим и способ передачи данных. Существуют следующие режимы передачи данных: симплексный, полудуплексный и полнодуплексный (или просто дуплексный) и способы передачи: параллельный и последовательный.

При симплексном режиме данные передаются только в одном направлении. Используя транспортную аналогию, симплексную передачу можно представить как однонаправленную однополосную дорогу. Сейчас она редко используется па практике.

Полудуплексный режим является самым распространенным. Он похож на однополосную дорогу, по которой движение может осуществляться в обоих направлениях, но не одновременно, а последовательно.

Режим полного дуплекса похож на двухполосную, двунаправленную дорогу. Данные могут передаваться в обоих направлениях одновременно.

Параллельная передача характеризуется тем, что группа битов передается одновременно по нескольким проводникам. Каждый бит передается по собственному проводу. Например, все внутренние коммуникации компьютера с его устройствами осуществляются через параллельную передачу. Это быстрый способ передачи. Однако при больших расстояниях он становится экономически невыгодным не только из-за того, что требует значительно больше кабеля, но и по причине взаимных помех этих проводников.

При последовательной передаче группа битов передается последовательно, один за другим по одному проводнику. Она медленнее, но экономически более выгодна при передаче на большие расстояния.

Также важной характеристикой шины является количество ее главных (задающих) устройств (bus master). Главное устройство шины – это устройство, которое может инициировать транзакцию (передачу) записи или чтения. ЦП, например, всегда является главным устройством шины. Шина может иметь несколько главных устройств, если имеется несколько ЦП или когда контроллеры ввода-вывода могут инициировать транзакции на шине. Если в

наличии несколько таких устройств, то требуется схема арбитража, чтобы решить, кто следующий захватит шину. Арбитраж часто основан либо на схеме с фиксированным приоритетом, либо на более «справедливой» схеме, которая случайным образом выбирает, какое главное устройство захватит шину.

В настоящее время используются два типа шин, отличающиеся способом коммутации:

– *шины с коммутацией цепей* (circuit-switched bus);

– *шины с коммутацией пакетов* (packet-switched bus).

Они получили свои названия по аналогии со способами коммутации в сетях передачи данных.

Шина с коммутацией пакетов при наличии нескольких главных устройств обеспечивает значительно большую пропускную способность по сравнению с шиной с коммутацией цепей за счет разделения транзакции (передачи) на две логические части: *запроса шины* и *ответа*. Такая методика получила название «расщепления» транзакций (split transaction). Транзакция чтения разбивается на транзакцию запроса чтения, которая содержит адрес, и транзакцию ответа памяти, которая содержит данные. Каждая транзакция теперь должна быть помечена (тегирована) соответствующим образом, чтобы ЦП и память могли сообщить, какому адресу соответствует данное.

Шина с коммутацией цепей не делает расщепления транзакций, любая транзакция на ней есть неделимая операция. Главное устройство запрашивает шину, после арбитража помещает на нее адрес и блокирует шину до окончания обслуживания запроса. Большая часть времени обслуживания при этом тратится не на выполнение операций на шине (например, на задержку выборки из памяти). Таким образом, в шинах с коммутацией цепей это время просто теряется.

Расщепленные транзакции делают шину доступной для других главных устройств, пока память читает слово по запрошенному адресу. Это, правда, также означает, что ЦП должен бороться за шину для посылки данных, а память должна бороться за шину, чтобы вернуть данные. Таким образом, *шина с расщеплением транзакций имеет более высокую пропускную способность, но обычно она имеет и большую задержку, чем шина, которая захватывается на все время выполнения транзакции.* Транзакция называется расщепленной, поскольку произвольное количество других пакетов или транзакций могут использовать шину между запросом и ответом.

Последний вопрос связан с выбором типа синхронизации и определяет, является ли шина *синхронной* или *асинхронной*. Если шина синхронная, то она

включает сигналы синхронизации, которые передаются по линиям управления шины, и фиксированный протокол, определяющий расположение сигналов адреса и данных относительно сигналов синхронизации. Поскольку практически никакой дополнительной логики не требуется для того, чтобы решить, что делать в следующий момент времени, эти шины могут быть и быстрыми, и дешевыми. Однако они имеют два главных недостатка. Все на шине должно происходить с одной и той же частотой синхронизации, поэтому из-за проблемы перекоса синхросигналов синхронные шины не могут быть длинными. Обычно шины процессор-память синхронные.

Асинхронная шина, с другой стороны, не тактируется. Вместо этого обычно используется старт-стопный режим передачи и протокол «рукопожатия» (handshaking) между источником и приемником данных на шине. Данные передаются как последовательность нулей и единиц, поэтому приемник должен уметь выделять байт в этом потоке данных. При асинхронной передаче каждый байт обрамляется стартовым и стоповым битами, с помощью которых приемник может их разделить. Эта схема позволяет гораздо проще приспособить широкое разнообразие устройств и удлинить шину, не беспокоясь о перекосе сигналов и о системе синхронизации. В целом асинхронная передача является относительно недорогой, потому что не требует дорогостоящего оборудования.

Если может использоваться синхронная шина, то она обычно быстрее, чем асинхронная, из-за отсутствия накладных расходов на синхронизацию шины для каждой транзакции. Выбор типа шины (синхронной или асинхронной) определяет не только пропускную способность, но также непосредственно влияет на емкость системы ввода-вывода в терминах физического расстояния и количества устройств, которые могут быть подсоединены к шине. Асинхронные шины по мере изменения технологии лучше масштабируются. Шины ввода-вывода обычно асинхронные.

4.4. Стандарты шин ПК

Системные шины предназначены для обмена информацией между CPU, памятью и другими устройствами, входящими в систему. К системным шинам относятся:

• ***GTL***, имеющая тактовую частоту 66, 100 и 133 МГц. Развитие системных шин GTL (Gunning Transceiver Logic – «стреляющая» логика передачи) связано с развитием архитектуры процессоров Intel. Системная шина GTL связывает процессор и северный мост, обеспечивает работу на высоких частотах при хорошей устойчивости к помехам.

В целом системная шина объединяет несколько магистралей: данных, адреса, служебную, питания. Разрядность шины данных определяет производительность процессора. Это параллельная шина, т. е. каждый разряд данных передается по отдельной линии. В процессоре Pentium 4 используется 64-разрядная шина. Адресная шина 36-разрядная, что позволило расширить адресуемую память до 64 Гбайт. Третья группа линий относится к управляющим, с помощью которых чипсет и процессор обмениваются командами и запросами, осуществляют тактирование и синхронизацию, управляют напряжением питания. Для Pentium 4 число линий управления достигает 124. Поэтому в целом для Pentium 4 требуется 224 линии системной шины GTL.

• ***EV6*** – шина, разработанная фирмой AMD, позволяет повысить тактовую частоту до 377 МГц, что обеспечивает ей пропускную способность 2.6 Гбайт/с. EV6 не является шиной в привычном понимании этого слова, а представляет собой просто 64-битный канал обмена между процессором и чипсетом. Обмен с системной памятью, PCI и AGP осуществляется чипсетом, причем каждая шина может работать на своей частоте. Поскольку главным «узким местом» современных процессоров является обмен с системной памятью, повышенная пропускная способность позволит уменьшить время простоя процессора при заполнении линии кеша.

• ***VME*** – шина приобрела большую популярность как шина ввода-вывода в рабочих станциях и серверах на базе RISC-процессоров. Эта шина высоко стандартизована, имеется несколько версий этого стандарта. В частности, VME32 – 32-битовая шина с производительностью 30 Мбайт/с, а VME64 – 64-битовая шина с производительностью 160 Мбайт/с.

Шины ввода-вывода представлены на рис. 4.3 и совершенствуются в соответствии с развитием периферийных устройств ПК.

Шина ISA в течение многих лет считалась стандартом ПК и до сих пор сохраняется в некоторых ПК наряду с современной шиной PCI, несмотря на то что практически не изменилась с момента своего расширения до 16 битов в 1984 г. Корпорация Intel совместно с Microsoft разработала стратегию постепенного отказа от шины ISA. В начале планируется исключить ISA-разъемы на материнской плате, а впоследствии исключить слоты ISA и подключить дисководы, мыши, клавиатуры, сканеры к шине USB, а винчестеры, приводы CD-ROM – к шине IEEE 1394. Однако наличие огромного парка ПК с шиной ISA будет востребовано еще на протяжении некоторого времени.

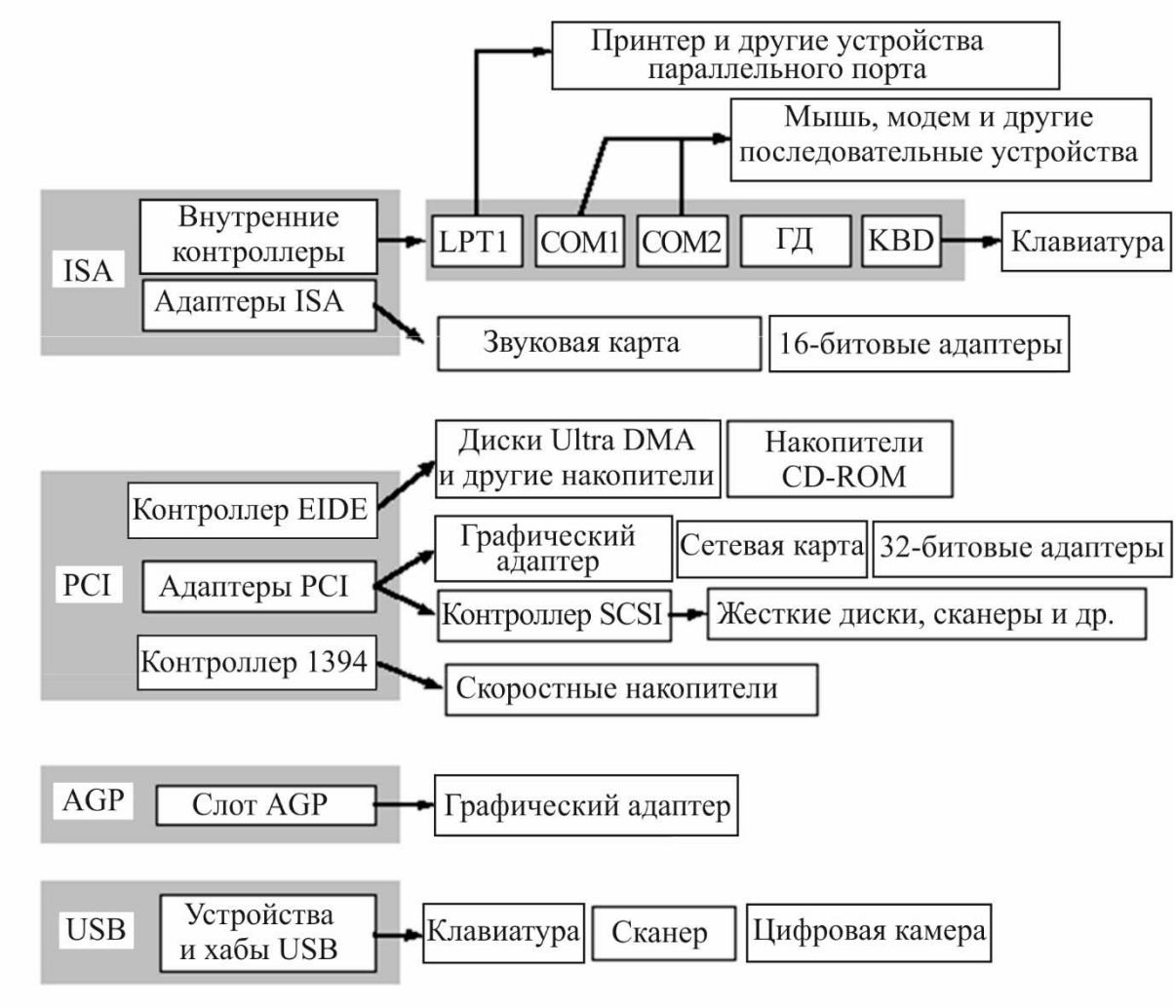

Рис. 4.3

Шина EISA (Extended ISA) стала дальнейшим развитием шины ISA в направлении повышения производительности системы и совместимости ее компонентов. Шина не получила широкого распространения в связи с ее высокой стоимостью и пропускной способностью, уступающей пропускной способности появившейся на рынке шины VLB.

Шина VESA, или **VLB**, предназначена для связи CPU с быстрыми периферийными устройствами и представляет собой расширение шины ISA для обмена видеоданными.

Первая ***локальная шина*** VL-Bus (VLB) появилась в 1992 г. Аббревиатура VESA означает Video Electronics Standards Association, а эта ассоциация была создана в конце 80-х гг. XX в. для решения проблем видеосистем в PC. Основной причиной разработки шины VLB было улучшение производительности видеосистем PC.

Шина VLB представляет собой 32-битовую шину, которая является прямым расширением шины памяти процессора 486. Шина VLB обычно работает

на частоте 33 МГц, хотя в некоторых системах возможна и большая скорость. Применение видеокарты VLB и контроллера ввода-вывода значительно повышает производительность системы по сравнению с системой, имеющей только одну шину ISA.

Появление в 1994 г. процессора Pentium и его локальной шины PCI привело к постепенному «забвению» шины VLB.

Шина PCI (Peripheral Component Interconnect – взаимосвязь периферийных компонентов) была разработана фирмой Intel для процессора Pentium и представляла собой совершенно новую шину. Основополагающим принципом, положенным в основу шины PCI, является применение так называемых мостов (Bridges), которые осуществляют связь между шиной PCI и другими типами шин. В шине PCI реализованы следующие принципы:

• ***Пакетный режим (burst mode)*:** шина PCI может передавать информацию в пакетном режиме, когда после начальной адресации можно подряд передавать несколько наборов данных. Этот режим похож на пакетизацию кеша (cache bursting).

• ***Bus Mastering***, который подразумевает способность внешнего устройства при пересылке данных управлять шиной (без участия CPU). Во время передачи информации устройство, поддерживающее Bus Mastering, захватывает шину и становится главным. В этом случае центральный процессор освобождается для решения других задач, пока происходит передача данных.

Опции высокой полосы пропускания: версия 2.1 спецификации шины PCI допускает расширение до 64 бит и 66 МГц, что повышает текущую производительность в 4 раза (до 266 Мбайт/с). Скорость шины PCI в зависимости от чипсета и материнской платы можно установить как синхронную или асинхронную.

Чаще всего она применяется для контроллеров жестких дисков и графических контроллеров, которые монтируются непосредственно на материнской плате или на картах расширения в слотах шины PCI. В настоящее время шина PCI стала фактическим стандартом среди шин ввода-вывода.

Шина AGP – высокоскоростная локальная шина ввода-вывода, предназначенная исключительно для нужд видеосистемы. Она связывает видеоадаптер (3D-акселератор) с системной памятью ПК. Трафик на шине PCI становится очень напряженным в современных ПК, когда видео, жесткие диски и другие периферийные устройства конкурируют между собой за единственную полосу пропускания ввода-вывода. Чтобы предотвратить насыщение шины

PCI видеоинформацией, фирма Intel разработала новый интерфейс специально для видеосистемы, который называется *ускоренным графическим портом* (Accelerated Graphics Port – AGP). Шина AGP была разработана на основе архитектуры шины PCI, поэтому она также является 32-разрядной. Однако при этом у нее есть дополнительные возможности увеличения пропускной способности, в частности за счет использования более высоких тактовых частот.

Идея реализации AGP довольно проста: создать быстрый специализированный интерфейс между видеочипсетом и системным процессором. Интерфейс реализуется только между этими двумя устройствами, что обеспечивает три основных преимущества: проще реализовать порт, проще повысить скорость AGP и можно ввести в интерфейс специфические для видео усовершенствования. AGP считается портом, а не шиной, так как он объединяет только два устройства (процессор и видеокарту) и не допускает расширения.

Шина PCI Express, или **PCI-e** (Peripheral Component Interconnect Express), на физическом уровне шиной не является, будучи соединением типа точка-точка. Использует программную модель шины PCI и высокопроизводительный физический протокол, основанный на последовательной передаче данных. Разработка стандарта PCI Express была начата фирмой Intel в июле 2002 г.

В отличие от стандарта PCI, использовавшего для передачи данных общую шину с подключением параллельно нескольких устройств, PCI Express, в общем случае, является пакетной сетью с топологией типа звезда. Устройства PCI Express взаимодействуют между собой через среду, образованную коммутаторами, при этом каждое устройство напрямую связано соединением точка-точка с коммутатором.

Шина PCI Express нацелена на использование только в качестве локальной шины. Так как программная модель PCI Express во многом унаследована от PCI, то существующие системы и контроллеры могут быть доработаны для использования шины PCI Express заменой только физического уровня, без доработки программного обеспечения. *Высокая пиковая производительность шины PCI Express позволяет использовать ее вместо шин AGP и тем более PCI.*

Шина USB (универсальная последовательная шина) была разработана лидерами компьютерной и телекоммуникационной промышленности Compaq, DEC, IBM, Intel, Microsoft для подключения периферийных устройств вне корпуса ПК. Скорость обмена информацией по шине USB составляет 12 Мбит/с или 15 Мбайт/с. К компьютерам, оборудованным шиной USB, можно подключать такие периферийные устройства, как клавиатура, мышь, джойстик, принтер,

не выключая питания. Все периферийные устройства должны быть оборудованы разъемами USB и подключаться к ПК через отдельный выносной блок, называемый *USB-хабом*, или *концентратором*, с помощью которого к ПК можно подключить до 127 периферийных устройств (рис. 4.4).

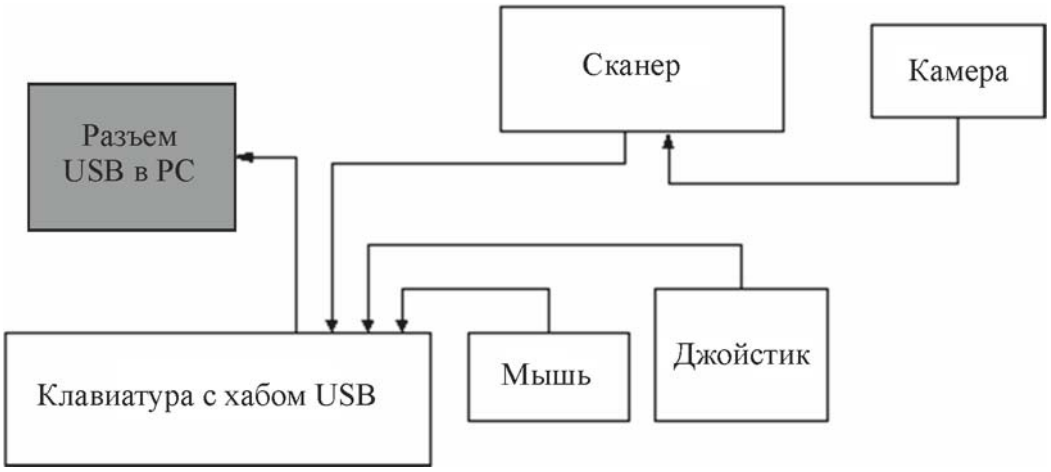

Рис. 4.4

Шина USB полностью поддерживает технологию Plug and Play. Она устраняет необходимость установки карт расширения внутри ПК и последующего реконфигурирования системы. Шина позволяет подключать, конфигурировать, использовать и при необходимости отключать периферийные устройства в то время, когда ПК и другие устройства работают. Не нужно инсталлировать драйверы, выбирать последовательные и параллельные порты, а также определять линии IRQ, DMA-каналы и адреса ввода-вывода. Все это достигается путем управления периферийными устройствами с помощью хост-контроллера на материнской плате или на карте PCI. Это снижает нагрузку на процессор и повышает общую производительность системы. Самим хост-контроллером управляет системное программное обеспечение в составе операционной системы.

Развитие шины USB связано с появлением шины USB 2.0, пропускная способность которой повышается с 12 до 360…480 Мбайт/с. Для нее будут разработаны новые скоростные периферийные устройства, которые расширят диапазон применений ПК. Скорости 12 Мбайт/с вполне достаточно для таких устройств, как телефоны, цифровые камеры, клавиатура, мышь, цифровые джойстики, ленточные накопители, цифровые динамики, сканеры и принтеры. Повышенная пропускная способность USB 2.0 расширит функциональность периферийных устройств, обеспечивая поддержку камер с высокой разрешающей способностью для видеоконференций, а также скоростных сканеров и принтеров следующего поколения.

Шина SCSI (Small Computer System Interface) является одной из наиболее популярных шин ввода-вывода, обеспечивает скорость передачи данных до 320 Мбайт/с и предусматривает подключение к одному адаптеру до восьми устройств: винчестеры, приводы CD-ROM, сканеры, фото- и видеокамеры. Отличительной особенностью шины SCSI является то, что она представляет собой кабельный шлейф. С шинами PC (ISA или PCI) шина SCSI связана через *хост-адаптер* (Host Adapter). Каждое устройство, подключенное к шине SCSI, может инициировать обмен с другими устройством.

Начальный стандарт 1986 г., известный теперь под названием SCSI-1, определял рабочие спецификации протокола шины, набор команд и электрические параметры.

Первоначально SCSI предназначался для использования в небольших дешевых системах и поэтому был ориентирован на достижение хороших результатов при низкой стоимости. Его характерной чертой является простота, особенно в части обеспечения гибкости конфигурирования периферийных устройств без изменения организации основного процессора. *Главной особенностью подсистемы SCSI является размещение в периферийном оборудовании интеллектуального контроллера.*

В 1992 г. этот стандарт был пересмотрен с целью устранения недостатков первоначальной спецификации (особенно в части синхронного режима передачи данных) и добавления новых возможностей повышения производительности, таких как «быстрый режим» (fast mode), «широкий режим» (wide mode) и помеченные очереди. Этот пересмотренный стандарт получил название SCSI-2 и в настоящее время используется большинством поставщиков вычислительных систем.

Шина IEEE 1394 – это стандарт высокоскоростной локальной последовательной шины, разработанный фирмами Apple и Texas Instruments. Шина IEEE 1394 предназначена для обмена цифровой информацией между ПК и другими электронными устройствами, особенно для подключения жестких дисков и устройств обработки аудио- и видеоинформации, а также работы мультимедийных приложений. Она способна передавать данные со скоростью до 1600 Мбайт/с, работать одновременно с несколькими устройствами, передающими данные с разными скоростями, как и SCSI.

Через интерфейс IEEE 1394 можно подключить к компьютеру практически любые устройства, способные работать с SCSI. К ним относятся все виды накопителей на дисках, включая жесткие, оптические, CD-ROM, DVD, цифро-

вые видеокамеры, устройства. Благодаря таким широким возможностям эта шина стала наиболее перспективной для объединения компьютера с бытовой электроникой. В настоящее время уже выпускаются адаптеры IEEE 1394 для шины PCI.

Шина IEEE 1394 (обычно называемая Fire Wire – огненный провод) во многом похожа на шину USB, также являясь последовательной шиной с горячей заменой, но намного быстрее. В IEEE 1394 есть два уровня интерфейса: один для шины на материнской плате компьютера и второй для интерфейса точка-точка между периферийным устройством и компьютером по последовательному кабелю. Простой мост объединяет эти два уровня. Интерфейс шины поддерживает скорости передачи данных в 12.5, 25 или 50 Мбайт/с, а интерфейс кабеля – 100, 200 и 400 Мбит/с, что намного больше скорости шины USB – 1.5 Мбайт/с или 12 Мбит/с. Спецификация 1394b определяет другие способы кодирования и передачи данных, что позволяет повысить скорость до 800 Мбит/с, 1.6 Гбит/с и более. Такая высокая скорость позволяет применять IEEE 1394 для подключения к PC цифровых камер, принтеров, телевизоров, сетевых карт и внешних запоминающих устройств.

5. ОРГАНИЗАЦИЯ СИСТЕМЫ ВВОДА-ВЫВОДА В ВЫЧИСЛИТЕЛЬНОЙ МАШИНЕ

5.1. Назначение и основные требования к системе ввода-вывода

Назначение системы ввода-вывода – это обеспечение взаимодействия центральной части вычислительной машины (ВМ) с внешней средой (пользователи, устройства, процессы), которое реализуется периферийными (или внешними) устройствами (ПУ или ВУ).

Можно выделить четыре класса внешних устройств, широко используемых в ВМ:

1) устройства, обеспечивающие взаимодействие пользователя и ВМ (клавиатура, мышь, светодиодные индикаторы, дисплеи, печатающие и звуковоспроизводящие устройства);

2) устройства внешней памяти, обеспечивающие хранение, ввод и вывод программ и данных;

3) устройства сопряжения с объектами; этот класс устройств крайне разнообразен. Сюда могут входить аналого-цифровые и цифроаналоговые преобразователи (АЦП и ЦАП), модуляторы и демодуляторы, усилители, фильтры и т. д.;

4) сетевое оборудование, обеспечивающее включение ВМ в информационно-вычислительную сеть.

Ни одно из этих устройств не может быть непосредственно подключено к шинам адреса, данных и управления ВМ. Здесь необходимы специальные устройства сопряжения, которые называют контроллерами или адаптерами. С точки зрения ВМ любой контроллер рассматривается как один или несколько портов ввода или вывода со своими конкретными адресами.

Основные проблемы ввода-вывода:

1. Существует большое количество ВУ с различными параметрами, существенно отличающимися:

а) по скорости передачи данных;

б) формату передачи данных (устройства последовательного, параллельного представления информации и т. д.);

в) размерам передаваемых данных (биты, байты, слова, блоки, секторы);

г) количеству выполняемых функций (чтение, запись, перемотка, подсчет, измерение).

2. Различные скорости работы центральной части машины и ВУ (требуется многоуровневая буферная память).

3. Асинхронность работы центральной части машины и ВУ (согласование операций в устройствах).

Требования к системе ввода-вывода:

1. Гарантировать эффективное согласование центральной части машины и ВУ с целью достижения максимальной производительности.

2. Обеспечение распределения ВУ между одновременно выполняемыми задачами в системе (большинство машин работают в многозадачных режимах).

3. Обеспечение управления каждым конкретным внешним устройством.

4. Обеспечение дружественного интерфейса с пользователем.

5.2. Архитектура систем ввода-вывода

Существует два основных способа организации системы ввода-вывода:

1. Прямой ввод-вывод (рис. 5.1).

2. Косвенный (канальный) ввод-вывод (рис. 5.2).

При прямом вводе-выводе работа внешних устройств и памяти управляется центральным процессором, и все они подключаются к одной системной шине, при этом в зависимости от варианта подключения памяти может быть: в случае 1 – общее пространство адресов памяти и портов ВУ, а в случае 2 – раздельные адреса обращения к памяти и ВУ. Совмещение адресного пространства используется в ВМ MIPS и SPARC, а раздельные адреса – в ВМ на основе процессоров Intel.

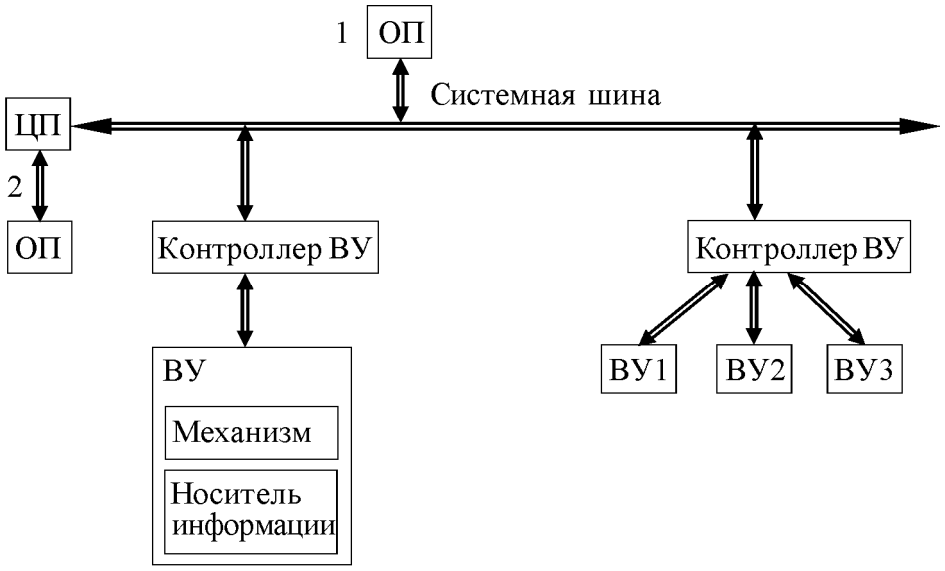

Рис. 5.1

При раздельном методе адресации порты ввода-вывода располагаются в своем собственном адресном пространстве, не совпадающем с адресным пространством памяти. Так, в архитектуре x86 действует собственная нумерация адресов, которая допускает использование до 256 портов ввода и до 256 портов вывода. Адрес порта N представляет собой двухразрядное шестнадцатеричное число в диапазоне 00H…0FFH. Управление записью и чтением со стороны процессора осуществляется с помощью сигналов управления I/OW – «запись в порт вывода» и I/OR – «чтение из порта ввода». Существенно, что запись и чтение памяти управляются при раздельной адресации другой парой сигналов (MEMW и MEMR). Связь портов с программой осуществляется двумя командами ввода-вывода: IN N и OUT N.

При адресации портов ввода-вывода, отображенной на память (в англ. мнемонике метод называется MMIO – memory-mapped I/O) часть адресного пространства отводится под порты ввода-вывода вместо памяти, например адреса от 0xFFFF0000 до 0xFFFFFFFF. Эти адреса расположены в зарезервированной части карты памяти. Каждому устройству ввода-вывода присваивается один или несколько адресов в этом диапазоне.

Такой метод адресации позволяет:

1) упростить системный контроллер, так как отпадает необходимость формирования сигналов I/OW и I/OR;

2) использовать при обращении к портам ВУ все множество команд, обеспечивающих взаимодействие с памятью;

3) иметь практически любое количество портов. Это количество ограничено только размером адресного пространства памяти ВМ.

Перечисленные достоинства достигаются усложнением дешифратора выбора портов и сокращением адресного пространства, отводимого под ячейки памяти.

На рис. 5.2 показан процесс ввода-вывода по канальному способу.

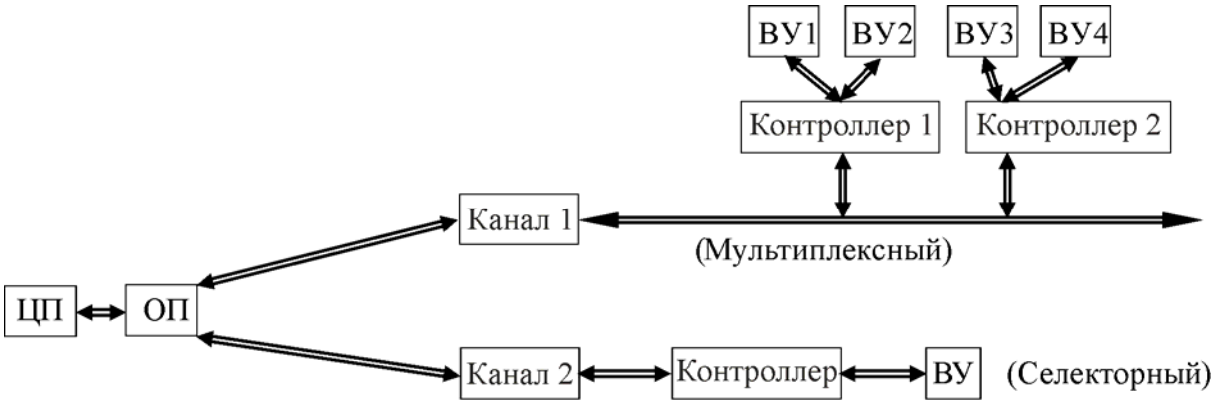

Рис. 5.2

Каналы делятся на следующие:

1. Мультиплексные (обслуживают много ВУ, но медленных).

2. Селекторные (обслуживают мало ВУ, но быстродействующих).

Отличительная особенность канального ввода-вывода – процессор освобождается от управления внешними устройствами, функция процессора заключается в инициализации запуска канальных программ и завершении операции ввода-вывода, выполняемой с помощью канала, по соответствующему признаку из канала (по существу это многопроцессорная система).

5.2.1. Состав и сравнение функций контроллеров и каналов

Контроллер в текущий момент времени выполняет одну команду ввода-вывода, получаемую от процессора или канала, и одновременно обслуживает одно внешнее устройство.

В его функции входит:

1) опознание своего адреса выборки;

2) подтверждение готовности внешних устройств;

3) управление операцией во внешнем устройстве;

4) согласование форматов данных;

5) согласование скоростей передачи (буферизация);

6) фиксация момента и характера операции ввода-вывода.

Контроллер должен содержать:

1. Селектор адреса (логическая схема, выдающая разрешающий сигнал на один адрес).

2. Регистры управления (содержит команду) и состояния. Состояния характеризуются следующими битами: DONE, BUSY, ERROR, часто применяются биты приоритета.

3. Буферные регистры данных, которые служат для согласования форматов и скоростей передачи.

Канал выполняет целую канальную программу из многих команд. Допускает одновременное управление несколькими внешними устройствами.

В его функции входит:

1) опознание своего адреса и подтверждение готовности;

2) прием команд процессора, инициализирующих работу канала, и нахождение в памяти своей канальной программы;

3) поиск контроллера и внешнего устройства, участвующего в операции, и проверка их готовности;

4) запуск канальной программы и управление обменом;

5) сообщение центрального процессора о завершении операции.

Канал представляет собой специализированный процессор с расширенными управлениями и ограниченный арифметическими возможностями.

В целом процесс взаимодействия внешних устройств и центральной части машины определяется интерфейсом ввода-вывода, под которым понимается совокупность сигналов, линий связи и алгоритмов управления, обеспечивающих заданный протокол взаимодействия внешних устройств и процессора. Под протоколом понимается последовательность формирования прямых и квитирующих сигналов взаимодействия (ответный сигнал, выдаваемый после приема прямого сигнала).

5.3. Способы выполнения операций передачи данных

Передача данных в системах ВВ может осуществляться отдельными битами (последовательная передача) либо байтами или словами (параллельная передача). Возможны два режима последовательной передачи данных: синхронный и асинхронный [2], [8].

5.3.1. Последовательная передача данных

При синхронном режиме передаваемые байты данных собираются в кадр, который обрамляется *синхробайтами* (рис. 5.3). Синхробайт – это байт, содержащий заранее известный код, например 0111110, который оповещает приемник о приходе кадра данных.

При его получении приемник начинает прием данных и их преобразование в параллельный формат. Такая организация синхронной последовательной передачи целесообразна лишь для пересылки массивов байтов, а не отдельных символов.

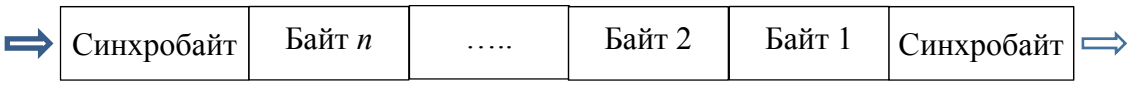

Рис. 5.3

Синхронная последовательная передача реализована в интерфейсе SPI (Serial Peripheral Bus), что переводится как «шина для последовательного подключения периферийных устройств». Ее главное назначение – связать одно главное устройство – Ведущее (Master) – с одним или несколькими Ведомыми (Slave). Ведущий в этом интерфейсе всегда один, и только он устанавливает скорость передачи данных и может формировать тактовые импульсы и другие параметры, такие как полярность и фаза тактирования. Ведущим является ЭВМ или микроконтроллер, а Ведомыми являются ПУ: микросхемы ЦАП и АЦП, модули беспроводной связи, включая приемопередатчики Wi-Fi и Bluetooth и т. д. Этот интерфейс особенно востребован там, где требуется высокая скорость передачи данных и высокая надежность. Интерфейс SPI – самый быстрый из всех имеющихся и самый «легкий» с точки зрения потребляемых ресурсов. Расплатой за это является использование большего количества проводов, чем у других интерфейсов. Здесь их требуется 3 штуки только непосредственно для передачи данных:

1) **MOSI** – Master Output Slave Input (Ведущий передает, Ведомый принимает);

2) **MISO –** Master Input Slave Output (Ведущий принимает, Ведомый передает);

3) **SCK –** Serial Clock (тактовый сигнал).

Кроме того, для корректной работы соединения потребуется еще несколько проводов для подключения к шине нескольких ПУ. Ведущему нужен способ отличать одно от другого: для этого в протокол добавлен сигнал SS (Slave Select). У каждого Ведомого есть для этого отдельный контакт, за состоянием которого он следит. Падение в низкий уровень означает, что Ведущий обращается конкретно к нему, и они начинают активное взаимодействие.

Есть два способа подключения Ведущего и Ведомых. *Первый:* классический, когда несколько Ведомых соединяются с Ведущим параллельно шинами MISO, MOSI и SCK, а SS связан с каждым из них индивидуально. Схема про-

стая и понятная, но требует дополнительно столько пинов, сколько на шине устройств. *Второй*: цепочка или кольцо – SS один на всех, но данные передаются сквозь устройства с одного на другое. Пока SS в низком уровне, данные растекаются по своим местам, после подъема уровня SS устройства начинают работать с принятыми данными. Очевидный плюс – от контроллера требуется меньше пинов, но и большой минус – далеко не все устройства поддерживают сквозную передачу данных.

Преимущества SPI: максимально простой и быстрый. Скорость может достигать десятков мегагерц, что позволяет передавать большие объемы данных в потоковом режиме. Все шины однонаправлены, это упрощает задачу преобразования уровней. Программная реализация тоже максимально проста. *Недостатки:* требуется большее количество проводов и пинов, которое напрямую зависит от количества устройств.

Асинхронная последовательная передача данных означает, что у передатчика и приемника нет общего генератора синхроимпульсов и синхронизирующий сигнал не посылается вместе с данными. В этом случае приемник узнает о моментах начала и завершения передачи данных следующим образом.

Стандартный формат асинхронной последовательной передачи данных между ЭВМ и ВУ содержит *n* пересылаемых битов информации (при пересылке символов *n* равно 7 или 8 бит) и 3–4 дополнительных бита: стартовый бит, бит контроля четности и 1 или 2 стоповых бита (рис. 5.4, *а*). Бит четности может отсутствовать. Когда передатчик бездействует (данные не посылаются на линию), на линии сохраняется уровень сигнала, соответствующий логической 1.

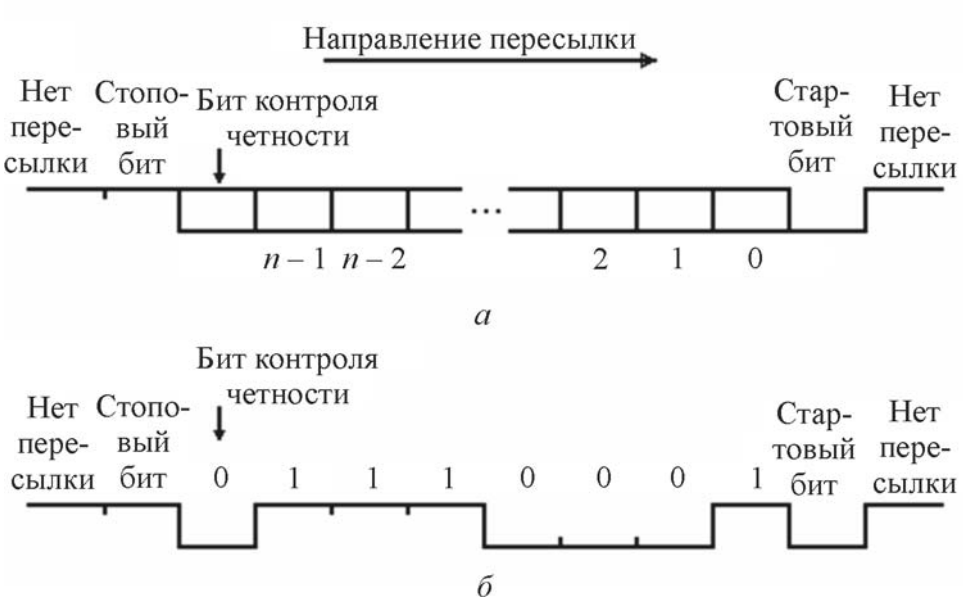

Рис. 5.4

Передатчик начинает пересылку символа посредством генерирования стартового бита, т. е. перевода линии в состояние логического 0 на время, точно равное времени передачи бита. Затем происходит передача битов символа, начиная с младшего бита, за которым следует дополнительный бит контроля по четности. Далее с помощью стопового бита линия переводится в состояние логической 1 (рис. 5.4, *б*). Состояние логической 1 должно поддерживаться в течение промежутка времени, равного 1 или 2 временам передачи бита.

Промежуток времени от начала стартового бита до конца стопового бита называется кадром. Сразу после стопового бита передатчик может посылать новый стартовый бит, если имеется другой символ для передачи; в противном случае уровень логической 1 может сохраняться, пока бездействует передатчик. Новый стартовый бит может быть послан в любой момент времени после окончания стопового бита.

В линиях последовательной передачи данных передатчик и приемник должны быть согласованы по всем параметрам формата, включая номинальное время передачи бита. Для этого в приемнике устанавливается генератор синхроимпульсов, частота которого должна совпадать с частотой аналогичного генератора передатчика. Кроме того, для обеспечения защищенности сигнала от искажения частоты синхроимпульсов приемник должен считывать принимаемый бит в середине его длительности.

Очевидно, что дополнительные биты «старт» и «стоп» уменьшают скорость передачи данных по сравнению с синхронным режимом передачи данных, при котором все байты передаются впритык друг к другу без стартового и стопового битов. Кроме того, скорость передачи данных может уменьшаться из-за наличия промежутков между байтами. При синхронном режиме передачи между передаваемыми данными нет никаких лишних промежутков.

Асинхронная последовательная передача данных реализована в универсальном асинхронном приемопередатчике (Universal Asynchronous Receiver-Transmitter – UART). UART − узел вычислительных устройств, являющийся отдельной микросхемой (например, Intel I8250) или частью контроллера, предназначенный для организации последовательной связи цифровых устройств. UART реализует последовательный ввод-вывод без пересылки тактового сигнала. Вместо этого системы должны заранее договориться о скорости передачи данных, и каждая из них должна локально генерировать свой собственный тактовый сигнал. Хотя эти системные тактовые сигналы могут иметь небольшую погрешность частоты и неизвестное соотношение фаз, UART

обеспечивает надежную асинхронную связь. UART используется в таких протоколах, как RS-232 и RS-485.

Передача данных в UART осуществляется по одному биту в равные промежутки времени. Этот временной промежуток определяется заданной скоростью UART и для конкретного соединения указывается в бодах (что соответствует байтам в секунду). Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19 200; 38 400; 57 600; 115 200; 230 400; 460 800 бод. В современных системах 9600 и 115 200 являются двумя наиболее распространенными скоростями передачи; 9600 встречается там, где скорость не имеет значения, а 115 200 является быстрой стандартной скоростью, хотя более медленной по сравнению с синхронным последовательным вводом-выводом типа SPI.

Как отмечалось ранее (см. рис. 5.4), помимо информационных битов UART автоматически вставляет в поток синхронизирующие метки – *стартовый* и *стоповый биты*. При приеме эти лишние биты удаляются из потока. Обычно стартовый и стоповый биты обрамляют один байт информации (8 бит), при этом младший информационный бит передается первым, сразу после стартового. Некоторые реализации UART используют два стоповых бита при передаче для уменьшения вероятности рассинхронизации приемника и передатчика при плотном трафике. Приемник игнорирует второй стоповый бит, воспринимая его как короткую паузу на линии.

Принято соглашение, что пассивным (в отсутствие потока данных) состоянием входа и выхода UART является логическая 1. Стартовый бит всегда является логическим 0, поэтому приемник UART ждет перепада из 1 в 0 и отсчитывает от него временной промежуток в половину длительности бита (середина передачи стартового бита). Следующие 8 значений являются принятыми данными, последнее значение проверочное (стоп-бит). Значение стоп-бита всегда равно 1. Если реально принятое значение иное, UART фиксирует ошибку.

Для формирования временных интервалов передающий и приемный UART имеют источник точного времени (тактирования). Точность этого источника должна быть такой, чтобы сумма погрешностей (приемника и передатчика) установки временного интервала от начала стартового импульса до середины стопового импульса не превышала половины битового интервала.

5.3.2. Параллельная передача данных

Параллельная передача данных между контроллером и ВУ является по своей организации более простым способом обмена. Для ее организации

помимо шины данных, количество линий в которой равно числу одновременно передаваемых битов данных, используется минимальное количество управляющих сигналов.

В простом контроллере ВУ, обеспечивающем побайтный вывод данных на внешнее устройство [9], в шине связи с ВУ (рис. 5.5) используются всего два управляющих сигнала: «Выходные данные готовы» (к ВУ) и «Данные приняты» (от ВУ).

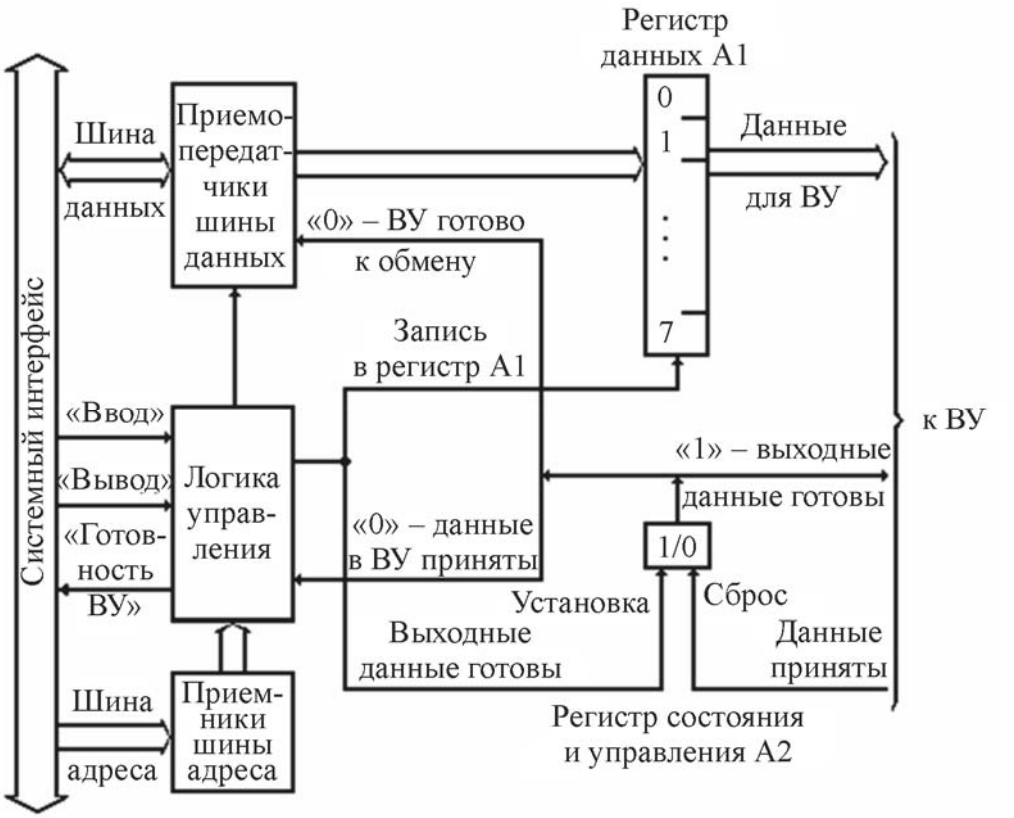

Рис. 5.5

Для формирования управляющего сигнала «Выходные данные готовы» и приема из ВУ управляющего сигнала «Данные приняты» в контроллере используется одноразрядный адресуемый регистр состояния и управления А2. Одновременно с записью очередного байта данных с шины данных системного интерфейса в адресуемый регистр данных контроллера (порт вывода А1) в регистр состояния и управления записывается логическая единица. Тем самым формируется управляющий сигнал «Выходные данные готовы» в шине связи с ВУ.

ВУ, приняв байт данных, управляющим сигналом «Данные приняты» обнуляет регистр состояния контроллера. При этом формируются управляющий сигнал системного интерфейса «Готовность ВУ» и признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных систем-

ного интерфейса посредством стандартной операции ввода при реализации программы асинхронного обмена.

Логика управления контроллера обеспечивает селекцию адресов регистров контроллера, прием управляющих сигналов системного интерфейса и формирование на их основе внутренних управляющих сигналов контроллера, формирование управляющего сигнала системного интерфейса «Готовность ВУ». Для сопряжения регистров контроллера с шинами адреса и данных системного интерфейса в контроллере используются, соответственно, приемники шины адреса и приемопередатчики шины данных.

Блок-схема простого контроллера ВУ, обеспечивающего побайтный прием данных из ВУ [9], приведена на рис. 5.6. В этом контроллере при взаимодействии с внешним устройством также используются два управляющих сигнала: «Данные от ВУ готовы» и «Данные приняты».

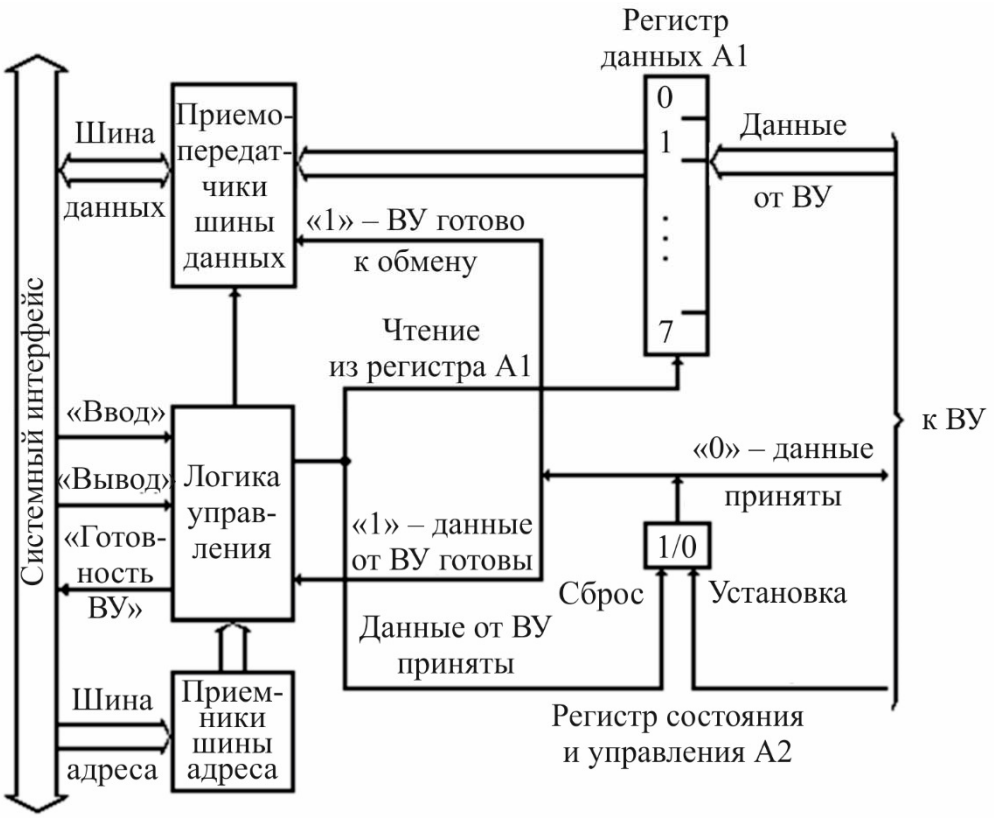

Рис. 5.6

Для формирования управляющего сигнала «Данные приняты» и приема из ВУ управляющего сигнала «Данные от ВУ готовы» используется одноразрядный адресуемый регистр состояния и управления А2. Внешнее устройство записывает в регистр данных контроллера А1 очередной байт данных и управляющим сигналом «Данные от ВУ готовы» устанавливает в единицу регистр состояния и управления А2.

При этом формируются: управляющий сигнал системного интерфейса «Готовность ВУ»; признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных системного интерфейса посредством операции ввода при реализации программы асинхронного обмена.

Тем самым контроллер извещает процессор о готовности данных в регистре А1. Процессор, выполняя программу асинхронного обмена, читает байт данных из регистра данных контроллера и обнуляет регистр состояния и управления А2. При этом формируется управляющий сигнал «Данные приняты» в шине связи с внешним устройством.

5.4. Способы выполнения обмена данными

Способы обмена данными удобно рассматривать, используя особенности взаимодействия процессора с ВУ, а также различных ВУ непосредственно друг с другом. Классификация основных способов выполнения обмена данными

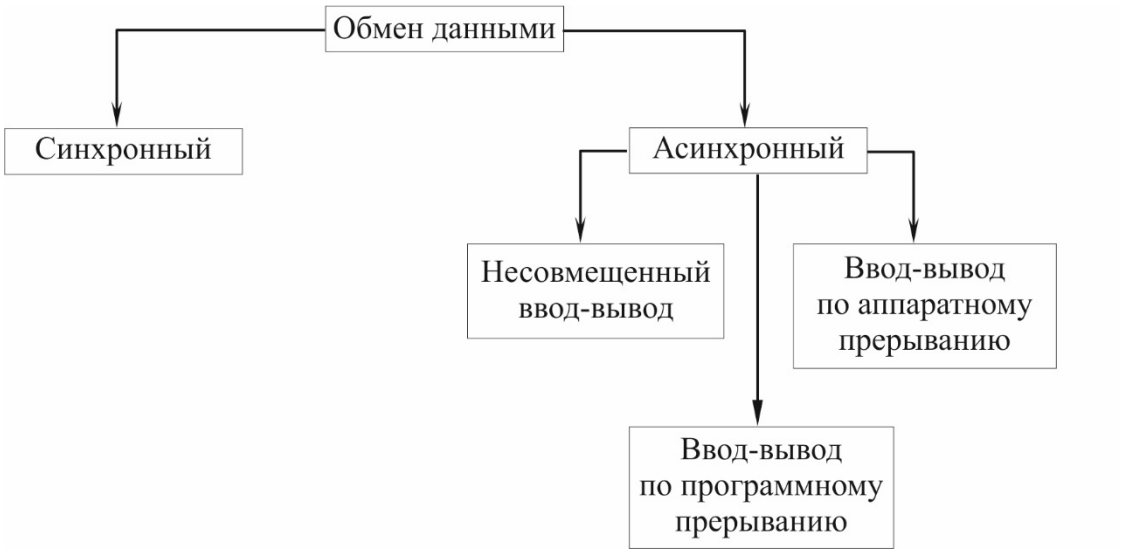

Рис. 5.7

показана на рис. 5.7, а на рис. 5.8–5.11 представлены упрощенные блок-схемы выполнения каждого из способов обмена данными.

5.4.1. Синхронный обмен данными

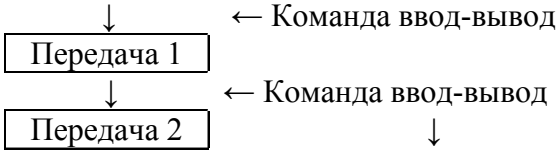

Рис. 5.8

Синхронный обмен данными (рис. 5.8) подразумевает жесткое соответствие сигналов процессора и контроллера ВУ. Это самый быстрый, но ненадежный способ обмена, применяемый только на близкие расстояния.

5.4.2. Программно-управляемый ввод-вывод

В случае программно-управляемого ввода-вывода центральный процессор после выдачи команды ввода-вывода ожидает готовности ВУ, проверяя установку флажка (бита) готовности (говорят, что процессор «висит» на флажке готовности) (рис. 5.9). Этот флажок представляет собой своеобразный семафор, который управляет доступом к данным либо от процессора, либо от ВУ. Этот способ обмена часто называют несовмещенным вводом-выводом, так как во время выполнения операции ввода-вывода процессор находится в режиме ожидания и не работает. Такой способ обмена удобен для внешнего устройства и неэффективен для процессора. Так как ВУ диктует условия, то со стороны ВУ скорость передачи данных – максимально возможная для него. Иногда удается в цикле ожидания вставить выполнение процессором фоновой программы.

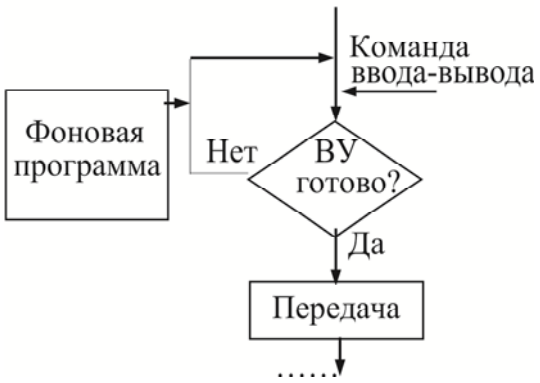

Рис. 5.9

5.4.3. Ввод-вывод по программному прерыванию

В данном случае (рис. 5.10) для осуществления передачи ВУ устанавливает запрос на прерывание работы процессора, и если прерывание возможно, то подпрограмма обслуживания прерывания выполняет передачу данных, по завершении которой происходит возврат на продолжение основной программы (ОП).

В таком режиме обмен данными с ВУ происходит в произвольные моменты времени, определяемые внешней по отношению к ЭВМ средой, и не зависит от программы, выполняемой в ЭВМ. Обмен с прерыванием программы значительно экономит время процессора за счет того, что исчезает необходимость в организации программных циклов ожидания готовности ВУ, на выполнение которых тратится значительное время, особенно при обмене с медленными ВУ.

Рис. 5.10

При реализации ввода-вывода по прерываниям нужно решить две проблемы:

1) как процессору определить, какое из ВУ выставило запрос прерывания;

2) какое из нескольких ВУ, выставивших запрос прерывания, должно быть обслужено в первую очередь.

Существует два основных способа идентификации ВУ, запросивших обслуживания:

– программный опрос регистров состояния (бит «Готовность ВУ») контроллеров всех ВУ;

– использование векторов прерывания.

Организация прерываний с программным опросом готовности предполагает наличие единой подпрограммы обслуживания прерываний от всех ВУ. При этом в конце последнего цикла выполнения очередной команды основной программы процессор проверяет наличие запроса прерывания от ВУ. Если запрос прерывания есть и оно разрешено, то процессор переключается на выполнение подпрограммы обработки прерываний.

Проводится последовательный опрос регистров состояния контроллеров всех ВУ, работающих в режиме прерывания, и когда подпрограмма обнаружит готовое к обмену ВУ, сразу выполняются действия по его обслуживанию. Завершается подпрограмма обработки прерывания после опроса готовности всех ВУ.

Приоритет ВУ в ЭВМ с программным опросом готовности ВУ однозначно определяется порядком их опроса в подпрограмме обработки прерываний. Чем раньше в подпрограмме опрашивается готовность ВУ, тем меньше время реакции на его запрос и выше приоритет. Необходимость проверки готовности всех ВУ увеличивает время обслуживания тех ВУ, которые опрашиваются последними. Это является основным недостатком рассматриваемого способа организации прерываний.

Организация обмена по прерываниям с использованием векторов прерываний позволяет устранить указанный недостаток. При такой организации ВУ, запросившее обслуживания, само идентифицирует себя с помощью вектора прерывания – адреса ячейки основной памяти ЭВМ, в которой хранится адрес начала подпрограммы обслуживания прерывания данного ВУ. Поэтому процессор, получив вектор прерывания, сразу переключается на выполнение требуемой подпрограммы обработки прерывания. В ЭВМ с векторной системой прерывания каждое ВУ должно иметь собственную подпрограмму обработки прерывания.

При этом запрос прерывания в процессор формирует контроллер прерываний, общий для всех устройств, работающих в режиме прерываний (IBM-

совместимые персональные компьютеры). Для обслуживания прерываний от нескольких ВУ контроллер прерываний сравнивает уровень поступившего запроса с уровнем находящегося на обслуживании и, если уровень поступившего запроса больше, разрешает его обслуживание, выдавая сигнал «Запрос на прерывание» в процессор. В ответ процессор формирует управляющий сигнал «Предоставление прерывания», который разрешает контроллеру ВУ, запросившему обслуживание, выдачу вектора прерывания в шину адреса системного интерфейса.

Данный способ обмена удобен для процессора и неэффективен для ВУ, в таком режиме к процессору подключаются медленные устройства с произвольными моментами готовности к передаче данных.

5.4.4. Ввод-вывод по аппаратному прерыванию (прямой доступ к памяти)

В этом режиме обмен данными между ВУ и основной памятью ЭВМ происходит без участия процессора. Обменом в режиме прямого доступа к памяти (ПДП) управляет не программа, выполняемая процессором, а электронные схемы, внешние по отношению к процессору. Обычно схемы, управляющие обменом в режиме ПДП, размещаются в специальном контроллере, который называется контроллером прямого доступа к памяти (КПДП).

Для реализации режима ПДП необходимо обеспечить непосредственную связь КПДП и памяти ЭВМ. В целях сокращения количества линий в шинах ЭВМ контроллер ПДП подключается к памяти посредством шин адреса и данных системного интерфейса. При этом возникает проблема совместного использования шин системного интерфейса процессором и КПДП. Можно выделить два основных способа ее решения: реализацию обмена в режиме ПДП с «захватом цикла» и в режиме ПДП с блокировкой процессора.

Существует две разновидности прямого доступа к памяти с «захватом цикла». Наиболее простой способ организации ПДП состоит в том, что для обмена используются те машинные циклы процессора, в которых он не обменивается данными с памятью. В такие циклы КПДП может обмениваться данными с памятью, не мешая работе процессора. Однако возникает необходимость выделения таких циклов, которая не всегда легко обеспечивается. Причем обмен данными в этом режиме возможен только в случайные моменты времени одиночными байтами или словами.

Более распространенным является ПДП с «захватом цикла» и принудительным отключением процессора от шин системного интерфейса. Для реали-

зации такого режима ПДП системный интерфейс ЭВМ дополняется двумя линиями для передачи управляющих сигналов «Требование прямого доступа к памяти» (ТПДП) и «Предоставление прямого доступа к памяти» (ППДП).

Управляющий сигнал ТПДП формируется КПДП. Процессор, получив этот сигнал, приостанавливает выполнение очередной команды, не дожидаясь ее завершения, выдает на системный интерфейс управляющий сигнал ППДП и отключается от шин системного интерфейса. С этого момента все шины системного интерфейса управляются КПДП. Контроллер, используя шины системного интерфейса, осуществляет обмен одним байтом или словом данных с памятью ЭВМ и затем, сняв сигнал ТПДП, возвращает управление системным интерфейсом процессору. Как только КПДП будет готов к обмену следующим байтом, он вновь «захватывает» цикл процессора и т. д. В промежутках между сигналами ТПДП процессор продолжает выполнять команды программы. Тем самым выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме прерываний. Упрощенная блок-схема реализации этого способа обмена показана на рис. 5.11.

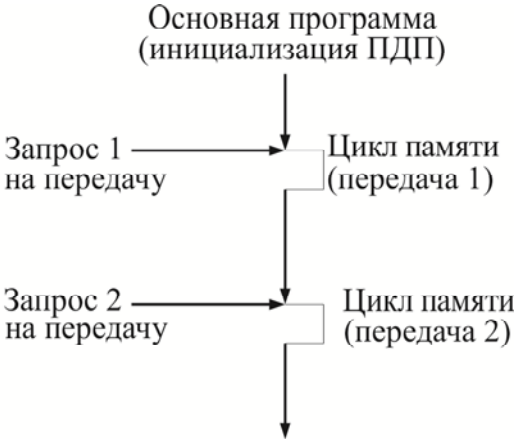

Рис. 5.11

Применение обмена данными с ВУ в режиме ПДП требует предварительной подготовки, а именно для каждого ВУ необходимо выделить область памяти, используемую при обмене, и указать ее размер, т. е. количество записываемых в память или читаемых из памяти байтов (слов) информации. КПДП должен обязательно иметь в своем составе регистр адреса и счетчик байтов (слов), в которые загружается информация перед началом обмена с ВУ в режиме ПДП: в счетчик байтов – количество передаваемых байтов данных, а в регистр адреса – начальный адрес области памяти для передаваемых данных.

Наиболее распространенным является обмен в режиме ПДП с блокировкой процессора. Он отличается от ПДП с «захватом цикла» тем, что управление системным интерфейсом передается контроллеру ПДП не на время обмена одним байтом, а на время обмена блоком данных. Это позволяет уменьшить накладные расходы, связанные с инициализацией контроллера ПДП перед обменом данными.

В целом обмен данными в режиме ПДП обеспечивает параллельную работу процессора и выполнение операций ввода-вывода, он используется для

подключения быстрых ВУ, так как передача идет между памятью и ВУ, и управление передачей выполняется аппаратно.

5.5. Структуры контроллеров ВУ для разных режимов обмена данными

5.5.1. Контроллер несовмещенного ввода-вывода

На рис. 5.12 представлена схема контроллера несовмещенного ввода-вывода (КНВВ). На схеме показаны возможности как ввода данных от ВУ, так и вывода данных из ЦП. Но по сравнению со схемами котроллеров параллельной передачи данных, представленных на рис. 5.5 и 5.6, данная схема упрощена тем, что на ней не полностью показаны схемы, обеспечивающие связь внутренних сигналов контроллера с сигналами системной шины.

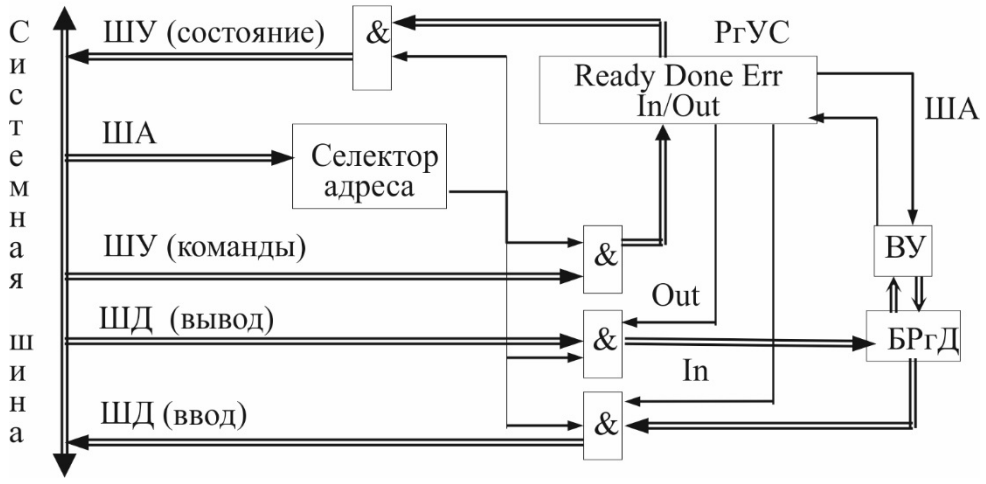

Рис. 5.12

Основные узлы КНВВ:

1) селектор адреса выдает на выходе единицу (разрешающий сигнал для других схем контроллера) только для одного своего адреса;

2) регистр управления и состояния (РгУС), который принимает от шины управления команды ввода-вывода (In, Out) и признак готовности данных от ЦП, а в системный интерфейс по шинам управления передает состояние ВУ до и после выполнения команды: Ready – готовность ВУ к обмену; Done – готовность данных после передачи; Err – наличие ошибки в процессе обмена;

3) буферный регистр данных (БРгД) осуществляет промежуточное хранение данных для согласования форматов и выравнивания скоростей ВУ и процессора.

При выводе данных из ЦП в ВУ процессор проверяет сигнал готовности ВУ к обмену, выдает команду Out по ШУ, а выводимый байт данных по ШД в БРгД контроллера и ожидает установки флага готовности Done, подтверждающего прием байта данных в ВУ. После чего флаг Done сбрасывается.

При вводе данных из ВУ в ЦП процессор после проверки готовности ВУ выдает по ШУ команду In и ожидает установки от ВУ флага готовности Done, свидетельствующего о загрузке байта данных из ВУ в БРгД контроллера. Затем считывает данное из БРгД контролера в регистр процессора (обычно AL) и сбрасывает флаг Done.

Как отмечалось ранее, необходимость ожидания установки флага готовности Done снижает эффективность работы процессора.

5.5.2. Контроллер обмена по программному прерыванию

Упрощенная схема контроллера обмена по программному прерыванию (ОПП) показана на рис. 5.13. Контроллер ОПП функционально использует два узла:

1. Индивидуальный контроллер управления ВУ, аналогичный КНВВ.
2. Общий, или выделенный, контроллер обслуживания прерываний.

На схеме в части КНВВ показаны только биты EI и Done РгУС, по которым формируется сигнал IRQ запроса прерывания в контроллер обслуживания прерываний. Остальные узлы КНВВ и линии связи его внутренних сигналов с линиями системной шины идентичны показанным на рис. 5.12.

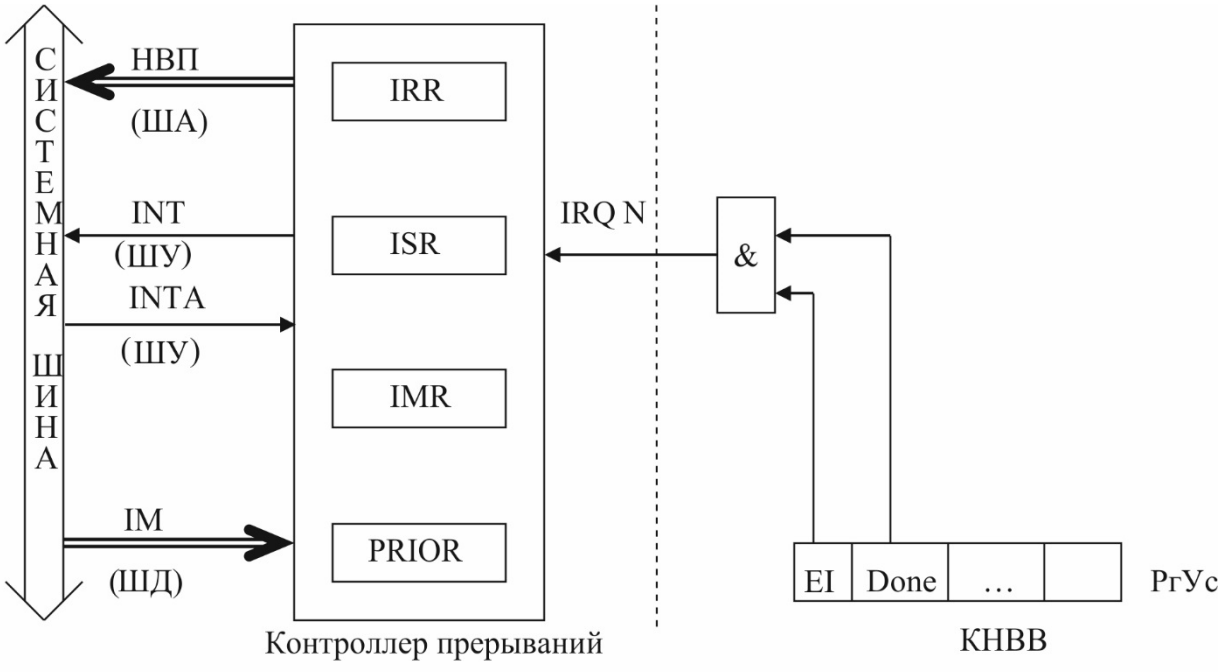

Рис. 5.13

На рис. 5.13 используются обозначения: IRR – регистр приема запросов, принимающий запрос прерывания IRQ N от ВУ, формируемый КНВВ; ISR – регистр обслуживания запросов, содержащий уровень выполнения текущего обработчика прерывания, сравниваемый с уровнем поступившего запроса;

IMR – регистр маскирования прерываний; PRIOR – схема разрешения приоритетов (арбитр), используемая в сложных случаях назначения и сравнения уровней приоритетов программ; IM – маска прерывания (выдается по шине данных ШД); INT – сигнал, выдаваемый по шинам управления (ШУ), для передачи запроса прерывания в процессор; INTA – выдаваемый по ШУ сигнал подтверждения разрешения на прерывание; НВП – номер вектора прерывания (выдается по шине адреса ША); EI – бит разрешения прерывания в РгУС КНВВ; Done – бит завершения операции ввода-вывода в РгУС КНВВ.

Для осуществления передачи ВУ устанавливает запрос IRQ на прерывание выполнения текущей программы. Если уровень запроса IRQ превышает уровень выполняемой программы, то контроллер прерываний выдает запрос прерывания INT процессору, и после завершения текущей команды процессор выдает сигнал INTA разрешения прерывания в контроллер. В ответ на это контроллер прерывания выдает по ША номер вектора прерывания для вызова обработчика, управляющего передачей данных в процессе обслуживания запроса от данного ВУ. Передача данных идет обычным образом через средства КНВВ.

Достоинством обмена данными через прерывание является то, что процессор не висит на флаге ожидания завершения операции ввода-вывода, а может продолжать выполнение программы. С другой стороны, ВУ должно ожидать, пока контроллер получит разрешение от процессора на прерывание от этого ВУ. Поэтому этот режим обмена удобен для процессора и неудобен для ВУ.

5.5.3. Контроллер обмена данными в режиме аппаратного прерывания (или КПДП)

Рассматривается режим прямого доступа к памяти с блочной передачей данных между памятью и быстрым ВУ (HDD, Sound card и т. д.). Управление передачей происходит без участия процессора (процессор должен только инициализировать регистры контроллера).

В обеспечении обмена задействованы КПДП и фрагменты контроллера прерываний (формирует сигнал в процессор о завершении обмена в режиме ПДП) и индивидуального контроллера ВУ, управляющего операциями ввода-вывода в конкретном ВУ по аналогии с КНВВ, дополненного линиями выдачи запроса на передачу в режиме ПДП.

Схема контроллера обмена данными в режиме ПДП показана на рис. 5.14.

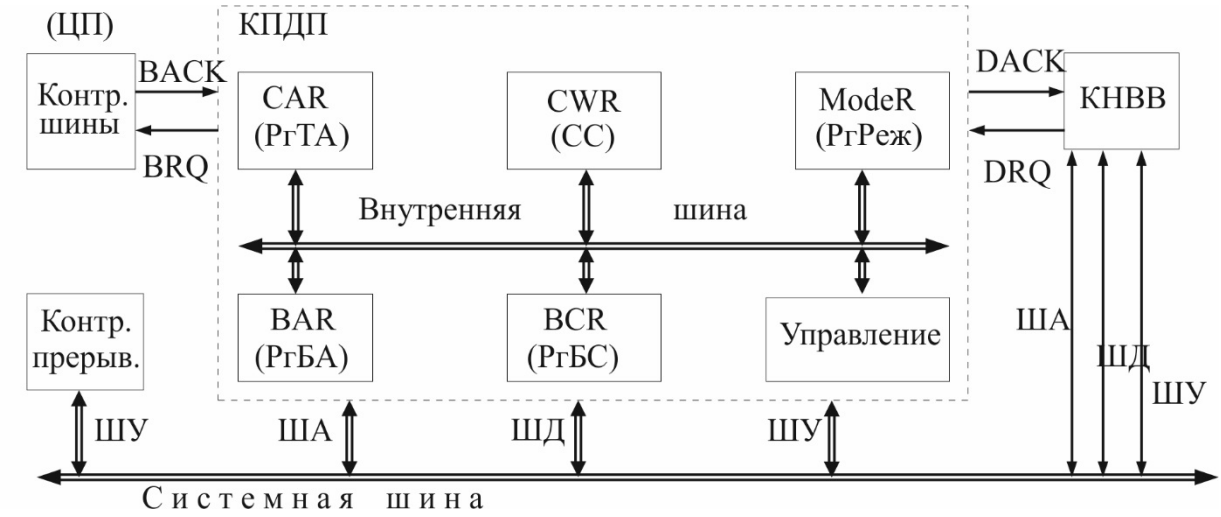

Рис. 5.14

Основные компоненты схемы (рис. 5.14): CAR – регистр текущего адреса (РгТА) – работает в режиме автоинкрементирования: увеличивается на 1 или 2 после каждой передачи байта или слова данных; CWR (СС) – счетчик слов (или байтов), который работает в режиме автодекрементирования: после каждой передачи содержимое регистра уменьшается на 1 (байт) или 2 (слово); ModeR – регистр режима передачи (определяет номер канала (адрес ВУ), характер и направление передачи); BAR – регистр базового адреса, содержит начальный адрес памяти и в процессе передачи не изменяется (служит для средств контроля передачи); BCR – регистр базового счетчика, содержит первоначальное количество передаваемых слов (служит для средств контроля передачи).

Управление – управление организацией передач по шинам КПДП.

По ШУ системного интерфейса в КПДП передаются следующие сигналы: MemR – чтение памяти; IOR – чтение устройства; MemW – запись в память; IOW – запись в устройство.

Алгоритм передачи данных в режиме ПДП:

1. После подготовки устройством данного (или готовности к приему очередного данного) контроллер ВУ выставляет запрос DRQ на передачу данного в режиме прямого доступа.

2. После его получения КПДП выставляет запрос BRQ на возможность захвата шины.

3. В ответ ЦП освобождает шины адреса, данных и шины управления, по которым передаются сигналы управления передачей данного. Затем вырабатывается сигнал BACK, подтверждающий для КПДП право владения системной шиной для выполнения передачи.

4. КПДП вырабатывает подтверждающий сигнал DACK контроллеру ВУ для управления передачей со стороны ВУ.

5. Далее КПДП выставляет на ША адрес текущей ячейки памяти и выдает пару команд: либо {MemR и IOW} – при записи данного из памяти в ВУ, либо {MemW и IOR} – при записи данного из ВУ в память, реализующих передачу данного между ВУ и памятью.

6. Изменяется содержимое регистра РгТА: ТА ++ и счетчика слов: СС --.

7. После этого сбрасываются сигналы запроса на передачу от ВУ и разрешения для КПДП владения шиной: DRQ ← 0, BACK ← 0. После чего управление шиной переходит к ЦП.

Пп. 1–7 реализуют «захват цикла памяти» и выполняются для передачи одного данного. Они повторяются до тех пор, пока не выполнится условие СС = 0, по которому формируется запрос в контроллер прерываний, в результате обслуживания которого процессор узнает о завершении передачи блока данных в режиме ПДП.

В компьютерах iX86 в качестве КПДП используются микросхемы:

1. 8237A – XT (содержит четыре канала с номерами 0–3).

2. 8237A – 5 – AT и выше (содержит две группы по четыре канала, первая – как у XT (для поддержки), а вторая – для работы с двухбайтовыми портами).

Приведем для примера состав регистров и адресацию для 8237A:

• 00h–07h – номера регистров по каждому из каналов.

Каждый канал содержит регистры (для чтения и записи):

– регистр текущего адреса (CAR);

– регистр счетчика слов (CWR);

– регистр базового адреса (BAR);

– регистр базового счетчика слов (BCR).

• 0Ah – регистр маски – разрешает и запрещает работу по каждому из каналов, номера битов:

0, 1 – номер канала;

2 – разрешить, запретить;

3–7 – не используются.

• 0Bh – регистр режима, номера битов:

0, 1 – номер канала;

2, 3 – тип цикла DMA:

00 – цикл проверки;

01 – цикл записи;

10 – цикл чтения;

11 – запрещенная комбинация;

4 – режим инициализации;

5 – направление приращения адреса:

0 – увеличение;

1 – уменьшение;

6, 7 – режим обслуживания:

00 – по требованию;

01 – одиночная передача;

10 – блочная передача;

11 – каскадирование (для двухуровневых контроллеров).

• 0Ch – сброс триггеров байта (для загрузки 16-битных регистров).

• 0Eh – сброс регистра маски

• 81h–8Fh – регистры страниц, текущий адрес равен 16 бит, шина адреса и шина данных (прямого доступа к памяти) равны 20 бит, регистры страниц задают адрес страницы (старшие четыре бита адреса – работают с 1 Мбайт), одна страница – 64 Кбайт.

Для 8237A-5-AT используются 3-байтовые регистры страницы (24 бит) и появляется возможность адресовать до 16 Мбайт оперативной памяти.

6. ПРОГРАММНЫЕ СРЕДСТВА УПРАВЛЕНИЯ ВВОДОМ-ВЫВОДОМ

Управление устройствами ввода-вывода (УВВ) компьютера является одной из главных функций ОС. Операционная система должна:

– передавать устройствам команды;

– перехватывать прерывания;

– обрабатывать ошибки;

– обеспечивать одинаковый интерфейс между устройствами и остальной частью системы.

6.1. Особенности УВВ с точки зрения программного управления

Устройства ввода-вывода делятся на два основных типа: *блок-ориентированные* и *байт-ориентированные*. Блок-ориентированные устройства управляют устройствами прямого доступа, которые хранят информацию в блоках фиксированного размера, каждый из которых имеет свой собственный адрес. Самое распространенное блок-ориентированное устройство – магнит-

ный диск. Адресуемость блоков приводит к тому, что для дисков появляется возможность кеширования данных в оперативной памяти.

Байт-ориентированные устройства не адресуют данные и не позволяют выполнять *операции* поиска данных, они генерируют или потребляют последовательность байтов, представляющую собой символьную информацию. Примерами таких устройств являются терминалы, принтеры, сетевые адаптеры и т. п.

УВВ, или внешнее устройство, обычно состоит из механизма (электромеханический компонент) и электронного компонента, который называется контроллером устройства или адаптером. Некоторые контроллеры могут управлять сразу несколькими устройствами.

Операционная система обычно имеет дело не с устройством, а с контроллером, который, как правило, выполняет простые функции, например, преобразует поток битов в блоки, состоящие из байтов, и осуществляет контроль и исправление ошибок. Каждый контроллер имеет несколько регистров, которые используются для взаимодействия с центральным процессором. ОС выполняет ввод-вывод, записывая команды (типа READ, WRITE) в регистры контроллера. Когда команда принята, процессор оставляет контроллер и занимается другой работой. После завершения выполнения команды контроллер организует прерывание, для того чтобы передать управление процессором операционной системе, которая должна проверить результаты операции. Процессор получает результаты и статус устройства, читая информацию из регистров контроллера.

6.2. Организация программного обеспечения ввода-вывода

При большом разнообразии устройств ввода-вывода, обладающих существенно различными характеристиками, целесообразной идеей организации программного обеспечения ввода-вывода (ПО ВВ) является иерархическая структура, основанная на его разбиении на несколько уровней, при котором нижние уровни (или слои) обеспечивают экранирование особенностей аппаратуры от верхних, а те, в свою очередь, обеспечивают удобный интерфейс для пользователей. При этом нижние слои ПО ВВ должны включать индивидуальные драйверы, написанные для конкретных физических устройств, а верхние слои должны обобщать процедуры управления этими устройствами, предоставляя общий *интерфейс* если не для всех устройств, то для группы устройств, обладающих общими характеристиками, например для всех матричных принтеров.

Более того, можно считать, что ПО ВВ делится не только на горизонтальные слои, но и на вертикальные, которые отражают специфику типов ВУ. Например, могут использоваться вертикальные подсистемы управления дисками, графическими устройствами, сетевыми адаптерами, текстовыми терминалами и т. п.

В общем случае целесообразно ПО ВВ разделить на четыре слоя (рис. 6.1):

1) аппаратный уровень, реализуемый контроллером ВУ и выполняющий обработку ошибок и частично обработку прерываний;

2) драйверы устройств;

3) независимый от устройств слой ОС;

4) пользовательский слой ПО ВВ.

Рис. 6.1

Обработка ошибок. Ошибки следует обрабатывать как можно ближе к аппаратуре. Если контроллер обнаруживает ошибку чтения, то он должен попытаться ее скорректировать. Если это ему не удается, то исправлением ошибок должен заняться драйвер устройства. И только если нижний уровень не может справиться с ошибкой, он сообщает об ошибке верхнему уровню ОС.

Обработка прерываний. Прерывания частично могут обрабатываться на аппаратном уровне, а частично на уровне ОС. В последнем случае они должны быть скрыты как можно глубже внутри операционной системы, чтобы только меньшая часть ОС имела с ними дело. Наилучший способ состоит в разрешении процессу, инициировавшему операцию ввода-вывода, блокировать себя до завершения операции и наступления прерывания. Процесс может блокировать себя, используя, например, вызов DOWN для семафора или вызов RECEIVE для ожидания сообщения. При наступлении прерывания процедура обработки прерывания выполняет разблокирование процесса, инициировавшего операцию ввода-вывода, используя вызовы UP или посылая процессу сообщение. В любом случае эффект от прерывания будет состоять в том, что ранее заблокированный процесс теперь продолжит свое выполнение.

Драйверы устройств. Весь зависимый от устройства код помещается в драйвер устройства. Каждый драйвер управляет устройствами одного типа. Основные свойства драйвера:

– входит в состав ядра ОС, работая в привилегированном режиме;

– непосредственно управляет внешним устройством, взаимодействуя с его контроллером с помощью команд ввода-вывода;

– обрабатывает прерывания от контроллера устройства;

– предоставляет прикладному программисту удобный интерфейс работы с устройством, экранируя от него низкоуровневые детали управления устройством и организации его данных;

– взаимодействует с другими модулями ядра ОС с помощью строго оговоренного интерфейса, описывающего формат передаваемых данных, структуру буферов, способы включения драйвера в состав ОС, способы вызова драйвера, набор общих процедур подсистемы ввода-вывода, которыми драйвер может пользоваться и т. п.

Согласно этим свойствам драйвер вместе с контроллером устройства и прикладной программой воплощают идею многослойного подхода к организации программного обеспечения ввода-вывода.

Если контроллер представляет низкий слой управления устройством, выполняющий операции в терминах блоков и агрегатов устройства (например, передвижение головки дисковода, поиск сектора), то драйвер выполняет более сложные операции, преобразуя данные, адресуемые в терминах номеров цилиндров, головок и секторов диска, в линейную последовательность блоков. В результате прикладная программа работает с данными, преобразованными в

понятную форму – файлами, таблицами баз данных и т. п., не вдаваясь в детали представления этих данных в УВВ.

Драйвер устройства принимает запрос от устройств программного слоя и решает, как его выполнить. Типичным запросом является чтение *n* блоков данных. Если драйвер был свободен во время поступления запроса, то он начинает выполнять запрос немедленно. Если же он был занят обслуживанием другого запроса, то вновь поступивший запрос присоединяется к очереди уже имеющихся запросов и будет выполнен, когда наступит его очередь.

В последнее время по мере развития операционных систем и усложнения структуры системы ввода-вывода наряду с традиционными драйверами в ОС появились так называемые высокоуровневые драйверы, которые располагаются в общей модели системы ввода-вывода над традиционными драйверами. Появление таких драйверов можно считать развитием идеи многоуровневой организации системы ввода-вывода, когда ее функции декомпозируются между несколькими модулями в соседних слоях иерархии (как пример, можно указать семиуровневую модель сетевых протоколов).

Традиционные драйверы, которые стали называть аппаратными или драйверами устройств, освобождаются от высокоуровневых функций и занимаются только низкоуровневыми операциями. Эти низкоуровневые *операции* составляют фундамент, на котором можно построить тот или иной набор операций в драйверах более высоких уровней.

При таком подходе повышается гибкость и *расширяемость* функций по управлению устройством. Например, если различным приложениям необходимо работать с различными логическими модулями управления одним и тем же физическим устройством, то для этого в системе достаточно установить несколько драйверов, использующих один аппаратный драйвер. Несколько драйверов, управляющих одним устройством, но на разных уровнях, можно рассматривать как один многоуровневый *драйвер*.

Высокоуровневые драйверы оформляются по тем же правилам и придерживаются тех же внутренних интерфейсов, что и аппаратные драйверы. Как правило, высокоуровневые драйверы не вызываются по прерываниям, так как взаимодействуют с устройством через посредничество аппаратных драйверов. Аппаратные драйверы после запуска операции ввода-вывода должны своевременно реагировать на завершение контроллером заданного действия путем взаимодействия с системой прерывания. Драйверы более высоких уровней вызываются не по прерываниям, а по инициативе аппаратных драйверов. Не все

процедуры аппаратного драйвера нужно вызывать по прерываниям, поэтому драйвер обычно имеет определенную структуру, в которой выделяется секция обработки прерываний (Interrupt Service Routine – ISR), которая и вызывается от соответствующего устройства диспетчером прерываний.

Независимый от устройств слой операционной системы. Большая часть ПО ВВ является независимой от устройств. Точная граница между драйверами и независимыми от устройств программами определяется системой, так как некоторые функции, которые могли бы быть реализованы независимым способом, в действительности выполнены в виде драйверов для повышения эффективности или по другим причинам.

Типичными функциями для независимого от устройств слоя являются:

– обеспечение общего интерфейса к драйверам устройств;

– именование устройств;

– защита устройств;

– обеспечение независимого размера блока;

– буферизация;

– распределение памяти на блок-ориентированных устройствах;

– распределение и освобождение выделенных устройств;

– уведомление об ошибках.

Рассмотрим некоторые функции данного перечня подробнее. Верхним слоям программного обеспечения неудобно работать с блоками разной величины, поэтому данный слой обеспечивает единый размер блока, например, за счет объединения нескольких различных блоков в единый логический блок. В связи с этим верхние уровни имеют дело с абстрактными устройствами, которые используют единый размер логического блока независимо от размера физического сектора.

При создании файла или заполнении его новыми данными необходимо выделить ему новые блоки. Для этого ОС должна вести список свободных блоков диска. На основании информации о наличии свободного места на диске может быть разработан алгоритм поиска свободного блока, независимый от устройства и реализуемый программным слоем, находящимся выше слоя драйверов.

Важной функцией независимого от устройств слоя является создание среды для остальных компонентов системы, которая облегчала бы их взаимодействие друг с другом. Эта задача решается созданием стандартного внутреннего интерфейса взаимодействия модулей ввода-вывода между собой. Это

облегчает включение новых драйверов и файловых систем в состав ОС. Кроме того, разработчики драйверов и других программных компонентов освобождаются от написания общих процедур, таких как *буферизация* данных и синхронизация нескольких модулей между собой при обмене данными.

Пользовательский слой программного обеспечения ВВ составляют системные вызовы ввода-вывода, которые принимают от пользовательских процессов запросы на *ввод-вывод* и переадресуют их отвечающим за определенный *класс* устройств модулям и драйверам, а также возвращают процессам результаты операций ввода-вывода. Таким образом, этот слой поддерживает пользовательский *интерфейс* ВВ, создавая для прикладных программистов максимум удобств по обращению к внешним устройствам и расположенным на них данным.

Данный слой ПО ВВ содержится в библиотеках, связываемых с пользовательскими программами. Системные вызовы, включающие вызовы ввода-вывода, обычно делаются библиотечными процедурами. Если программа, написанная на языке С, содержит вызов count = write (fd, buffer, nbytes), то библиотечная процедура write будет прямо связана с программой. Набор подобных процедур является частью системы ввода-вывода. В частности, форматирование ввода или вывода выполняется библиотечными процедурами. Примером может служить функция printf языка С, которая принимает строку формата и, возможно, некоторые переменные в качестве входной информации, затем строит строку символов ASCII и делает вызов write для вывода этой строки. Стандартная библиотека ввода-вывода содержит большое число процедур, которые выполняют ВВ и работают как часть пользовательской программы.

Другой категорией программного обеспечения ввода-вывода является подсистема спулинга (spooling). Здесь проблема состоит в том, что одни устройства являются разделяемыми, а другие – выделенными. Спулинг – это способ работы с выделенными устройствами в мультипрограммной системе. Диски – это разделяемые устройства, так как одновременный доступ нескольких пользователей к диску не представляет собой проблему. Принтеры – это выделенные устройства, потому что нельзя смешивать строчки, печатаемые различными пользователями. Наличие выделенных устройств создает для операционной системы некоторые проблемы. Так, для работы с принтером создается специальный процесс-монитор, который получает исключительные права на использование этого устройства. При этом создается специальный каталог, называемый каталогом спулинга. Для того чтобы напечатать файл, пользова-

тельский процесс помещает выводимую информацию в этот файл и помещает его в каталог спулинга. Процесс-монитор по очереди распечатывает все файлы, содержащиеся в каталоге спулинга.

Список литературы

1. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е изд. СПб.: Питер, 2014.

2. Харрис Дэвид М., Харрис Сара Л. Цифровая схемотехника и архитектура компьютера. 2-е изд. Берлингтон: Morgan Kaufman, 2013. 1621 с.

3. Орлов С. А., Цилькер Б. Я. Организация ЭВМ и систем. 2-е изд. СПб.: Питер, 2011.

4. Касперски К. Техника оптимизации программ. Эффективное использование памяти. СПб.: БХВ-Петербург, 2003.

5. Паттерсон Д., Хеннесси Дж. Архитектура компьютера и проектирование компьютерных систем. 4-е изд. СПб.: Питер, 2012.

6. Брамм П., Брамм Д. Микропроцессор 80386 и его программирование: пер. с англ. М.: Мир, 1990. 448 с.

7. Жмакин А. П. Архитектура ЭВМ. 2-е изд. СПб.: БХВ-Петербург, 2010.

8. Гук М. Аппаратные средства IBM PC. Энциклопедия. 2-е изд. СПб.: Питер, 2002.

9. Ершова Н. Ю., Ивашенков О. Н., Курсков С. Ю. Микропроцессоры: учеб. пособие / Петрозаводский гос. ун-т. Петрозаводск, 2011. Электр. форма, HTML.

Владимир Андреевич Кирьянчиков

Организация ЭВМ и систем.
Иерархическая система памяти.
Организация шин. Система ввода-вывода

Учебное пособие

Редактор И. Г. Скачек

Подписано в печать 30.06.22. Формат 60×84 1/16.
Бумага офсетная. Печать цифровая. Печ. л. 6,0.
Гарнитура «Times New Roman». Тираж 63 экз. Заказ .

Издательство СПбГЭТУ «ЛЭТИ»
197022, С.-Петербург, ул. Проф. Попова, 5Ф