МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ, МОЛОДЕЖИ И СПОРТА

ГУ «ЛУГАНСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНТИ ТАРАСА ШЕВЧЕНКО»

Лахно В.А., Могильный Г.А.

АРХИТЕКТУРА КОМПЬЮТЕРА

(Часть 2)

Луганск 2011

УДК 004.031., 004.032.8
ББК 32.973.
Л 297

Рекомендовано Ученым советом
Луганского национального университета имени
Тараса Шевченко
Протокол № 7 от 2011 г.

Рецензенты:

Лахно В.А., Могильный Г.А.
Архитектура компьютера (часть 2): учебное пособие. – Л.: ЛНУ имени Тараса Шевченко, 2011. – 142 с.

Изучение дисциплины "Архитектура компьютера" является одной из важных составляющих профессиональной подготовки современных специалистов в области информационных технологий. Стремительное развитие ЭВМ и другой цифровой электроники приводит к насыщению ими практически всех сфер деятельности человека.

В этих условиях для специалиста компьютерной или программной инженерии необходимо знание основ аппаратной части компьютера, его основных технических характеристик и функциональных возможностей.

Настоящее учебное пособие по дисциплине «Архитектура компьютера» призвано помочь студентам в получении знаний о функциональных составляющих современных компьютеров, их назначении, видах и характеристиках, а также формах представления информации в ЭВМ, основах машинной математики и логики.

МОДУЛЬ 3

МОДУЛЬ 4

Модуль 3
Подсистема памяти. Система прерываний и исключений. Материнские платы и чипсеты.

ГЛАВА 11. ОРГАНИЗАЦИЯ ПОДСИСТЕМЫ ПАМЯТИ В ЭВМ

Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации. Отдельные устройства, входящие в эту совокупность, называются запоминающими устройствами (ЗУ) того или иного типа [2, 5, 12, 16, 17].

Термин "запоминающее устройство" обычно используется, когда речь идет о принципе построения некоторого устройства памяти (например, полупроводниковое ЗУ, ЗУ на жестком магнитном диске и т.п.). Термин "память" применяют, когда хотят подчеркнуть выполняемую устройством памяти логическую функцию или место расположения в составе оборудования ЭВМ (например, оперативная память – ОП, внешняя память и т.п.).

Запоминающие устройства играют важную роль в общей структуре ЭВМ. По некоторым оценкам [18, 19, 20] производительность компьютера на разных классах задач на 40-50% определяется характеристиками ЗУ различных типов, входящих в его состав. К основным параметрам, характеризующим запоминающие устройства, относятся емкость и быстродействие.

11.1. Основные параметры и характеристики запаминающих устройств

Для оценки свойств запоминающих устройств и качества выполняемых ими функций используют ряд

показателей. Рассмотрим основные параметры и характеристики ЗУ.

Одной из основных характеристик ЗУ является ***емкость памяти*** – наибольший объем информации, который одновременно можетхранится в ЗУ. Базовой единицей измерения емкости памяти служит ***бит,*** представляющий собой один разряд – двоичного числа (логические 0 или 1). Добавление к основной единице измерения множителей 2^3, 2^{10}, 2^{20} позволяет получить такие единицы измерения как (1 байт = 8 бит), килобайт (1 Кбайт = 1024 байт), мегабайт (1 Мбайт = 1024 Кбайт = 1048576 байт). В ряде случаев в качестве единицы измерения емкости памяти используют (машинное) слово, содержащее 8, 16 32 бита. Бит хранится элементом памяти, а слово – ячейкой памяти. Ячейка памяти представляет собой совокупность элементов памяти, к которым возможно лишь одновременное обращение.

Для детализации структуры памяти используется показатель организация поминающего устройства, для оценки которого служит произведение числа хранимых слов на их разрядность, что дает емкость памяти ЗУ. Например, два ЗУ с организацией *32*х*32* и *64*х*16* имеют одинаковую емкость памяти 1024 бит.

Быстродействие (***производительность***) ЗУ может характеризоваться временем цикла записи или считывания (выборки) информации и временем обращения к памяти, включающим в себя оба цикла. ***Время цикла записи*** оценивается временным интервалом после появления сигнала записи до занесения входного слова в ячейку памяти, ***время цикла считывания*** – интервалом между моментами явления сигнала чтения и слова на выходе ЗУ. В тех случаях, когда обращение к первому слову передаваемого пакета слов требует больших временных затрат, чем к последующим словам, вводят два параметра: ***время доступа при первом обращении*** и ***темп передач для последующих слов пакета***. Приведенные параметры являются эксплуатационными, или измеряемыми.

Кроме них задают ряд режимных параметров в виде временных соотношений между различными управляющими сигналами, которые необходимы для нормального

функционирования ЗУ. К основным управляющим информационным сигналам следует отнести:

A – адрес, или *адресный код.* Адресный код является номером элемента ячейки памяти, в которых хранится бит, байт или слово информации. Число разрядов адресного кода выражается целой степенью двойки. Например, 12-ти разрядный код $A_{11}A_{10}A_9...A_0$ позволяет обратиться к любому из 2^{12}=4048 элементов памяти ЗУ;

$\overline{CS}$ (*Chip Select*) или $\overline{CE}$ (*Chip Enable*) – сигналы *выбора кристалла,* или микросхемы, активизирующие при нулевом логическом уровне работу данной микросхемы;

$\overline{W}/R$ (*Write/Read*) – сигнал, управляющий режимом работы памяти: $\overline{W}/R = 0$ – запись; $\overline{W}/R = 1$ – чтение;

DI, DO – (*Data Input, Data Output*) – входные и выходные *m*-разрядные данные ЗУ передаваемые по совмещенной или отдельной шинам. Число разрядом *m* определяется организацией ЗУ.

На рис. 11.1 и 11.2 показаны временные зависимости для режимов записи и чтения ЗУ.

Рассмотрим последовательность подачи управляющих сигналов на входы ЗУ. Прежде всего, подается адресный код требуемого элемента (ячейки) памяти, чтобы заблокировать память для доступа к ячейкам других операций. Затем подаются сигнал $\overline{CS} = 0$, которым активизируется работа выбранной микросхемы, и сигнал $\overline{W}/R$ записи/чтения.

Принятые на рис. 11.1 и 11.2 условные обозначения временных интервалов:

$t_{CY(A)}$ – время цикла адресации в режимах записи и считывания;

$t_{SU(A\text{-}WR)}$, $t_{SU(A\text{-}RD)}$ – время установки сигналов записи и чтения относительно адреса;

$t_{SU(A\text{-}CS)}$ – время установки сигналов выбора микросхемы относительно адреса в режимах записи и чтения;

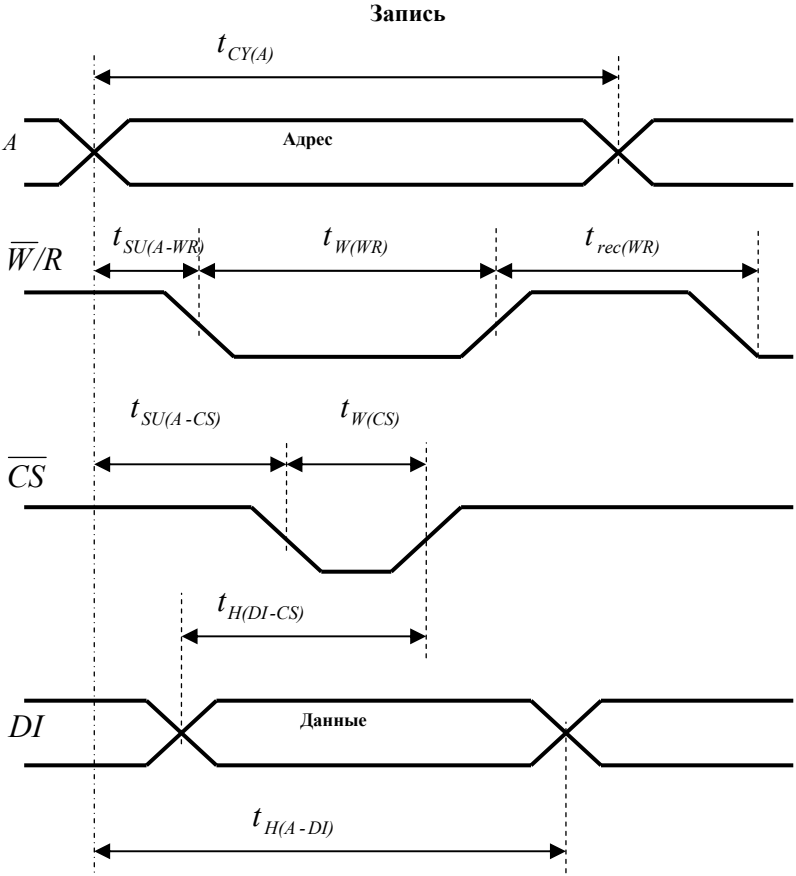


Рис. 11.1. Временная диаграмма работы статического ОЗУ в режиме записи

$t_{W(WR)}$, $t_{W(RD)}$ – длительность сигналов записи и чтения, длительность сигналов выбора микросхемы в режимах записи и чтения;

$t_{W(CS)}$ – длительность сигналов выбора микросхемы в режимах чтения или записи;

$t_{H(A-DI)}$ – время удержания между началом поступления адреса и окончанием поступления данных;

$t_{H(DI\ -CS)}$ – время удержания между началом поступления данных и окончанием сигнала выбора микросхемы;

t_A – время выборки адреса;

t_{CS} – время появления данных относительно начала сигнала выбора кристалла в режиме чтения;

t_{DS} – задержка перехода выхода данных из активного состояния в состояние отключено в режиме чтения.

Временные соотношения между управляющими сигналами задают следующими величинами:

- временем цикла t_{SY}, представляющем собой интервал между началами
- (или окончаниями) сигналов на любом управляющем входе ЗУ;
- временем установления t_{SU}, которое оценивается интервалом между началами сигналов двух различных управляющих сигналов;
- длительностью действия t_W заданного управляющего сигнала;
- временем удержания t_H, для оценки которого используется интервал между началом одного сигнала и окончанием другого;
- временем сохранения t_V, представляющим собой интервал между окончаниями одного и другого сигналов;
- временем доступа t_A, которое оценивается интервалом времени от появления одного из управляющих сигналов до появления информационного сигнала на выходе;
- и рядом др.

Указанные величины могут задаваться для любых управляющих сигналов ЗУ.

11.2. Классификация запоминающих устройств

Запоминающие устройства можно классифицировать по целому ряду параметров и признаков. На рис. 11.3 представлена классификация по типу обращения и организации доступа к ячейкам ЗУ.

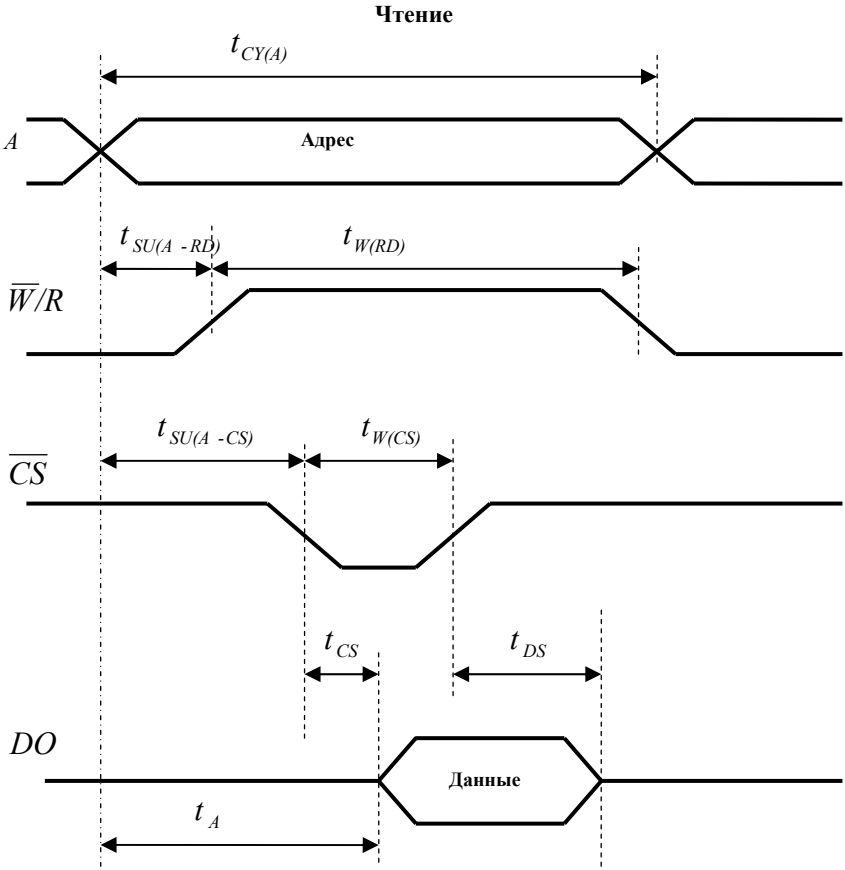


Рис. 11.2. Временная диаграмма работы статического ОЗУ в режиме чтения

По типу обращения ЗУ делятся на устройства, допускающие как чтение, так и запись информации, и постоянные запоминающие устройства (ПЗУ), предназначенные только для чтения записанных в них данных (*ROM - read only memory*). ЗУ первого типа используются в процессе работы процессора для хранения выполняемых программ, исходных данных, промежуточных и окончательных результатов. В ПЗУ, как правило, хранятся системные программы, необходимые для запуска компьютера в работу, а также константы. В некоторых ЭВМ, предназначенных, например, для работы в системах управления по одним и тем же неизменяемым алгоритмам, все программное обеспечение может храниться в ПЗУ.

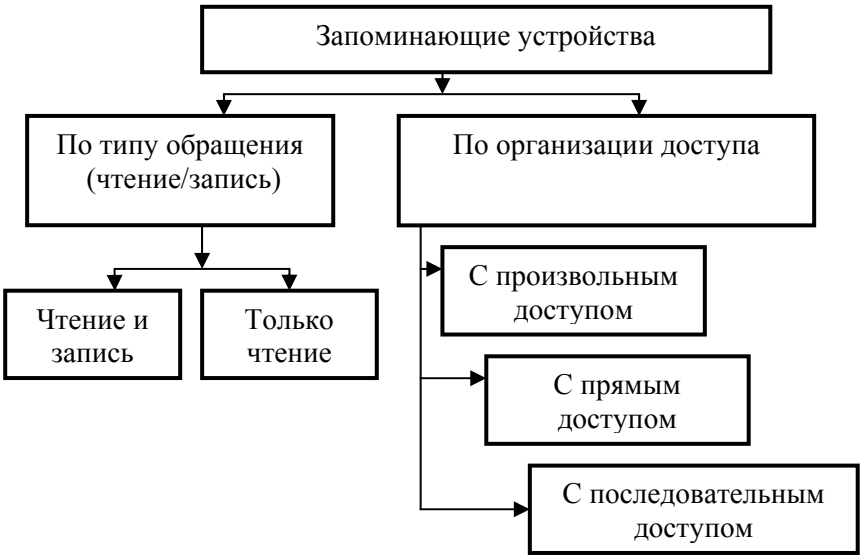


Рис. 11.3. Классификация запоминающих устройств

В ЗУ с произвольным доступом (*RAM - random access memory*) время доступа не зависит от места расположения участка памяти (например, ОЗУ).

В ЗУ с прямым (циклическим) доступом благодаря непрерывному вращению носителя информации (например,

магнитный диск – МД) возможность обращения к некоторому участку носителя циклически повторяется. Время доступа здесь зависит от взаимного расположения этого участка и головок чтения/записи и во многом определяется скоростью вращения носителя.

В ЗУ с последовательным доступом производится последовательный просмотр участков носителя информации, пока нужный участок не займет некоторое нужное положение напротив головок чтения/записи (например, магнитные ленты – МЛ).

По принципу действия различают:

- полупроводниковые ЗУ, выполненные на интегральных микросхемах;
- магнитные ЗУ, среди которых выделяют устройства с движущимися накопителями информации (накопители на гибких и жестких магнитных дисках);
- оптические (лазерные) ЗУ, в которых цифровые данные представлены в виде двоичного рельефа отражающей поверхности диска. Данные записываются и считываются механизмом оптического сканирования, в состав которого входит лазер и система зеркал.

По способу хранения информации ОЗУ разделяют на две подгруппы:

- статические ОЗУ (СОЗУ), в которых состояние элементов памяти при хранении информации остается неизменным;
- динамические ОЗУ (ДОЗУ), в которых состояние элементов памяти (обычно полупроводниковых емкостей) не остается неизменным и требует периодического проведения процесса регенерации.

По способу доступа к информации устройства памяти подразделяют на три группы:

- адресные ЗУ, в которых доступ к ячейке памяти обеспечивается с помощью адресного кода, являющимся по сути дела номером ячейки. К ним относятся оперативные ЗУ (ОЗУ), или ЗУ с произвольной выборкой, и постоянные ЗУ, обеспечивающие

- ЗУ с *последовательным доступом,* в которых осуществляется последовательный просмотр всех ячеек памяти. К этой группе ЗУ относят видеопамять, буфер FIFO (*First-In First-Out* – «первым пришел – первым ушел») и стек, или LIFO (*Last-In First-Out* - «последним пришел - первым ушел»);
- ассоциативные ЗУ, в которых поиск и извлечение информации производятся не по ее расположению в памяти (адресу или месту в очереди), а по некоторому характеристическому признаку самой информации. Таким признаком может служить, например, выделенная совокупность разрядов (поле) слова. Поле входного слова сравнивается с полями всех слов, хранящихся в памяти. Если поля совпадают, то слово считывается из памяти. Основная область применения ассоциативного доступа к данным - *кэш-память,* которая будет рассмотрена ниже.

По организации использования памяти в цифровых системах деляют:

- *внутренние* ЗУ, к памяти которых процессор вычислительной системы непосредственно обращается за командами и данными. К внутренним ЗУ относятся ОЗУ и полупроводниковые ПЗУ
- *внешние* ЗУ, к памяти которых за командами и данными процессор непосредственно обратиться не может. Для использования хранящейся во внешних ЗУ информации ее предварительно передают во внутренние ЗУ.

Очевидно, что идеальное запоминающее устройство должно обладать бесконечно большой емкостью и иметь бесконечно малое время обращения. На практике эти параметры находятся в противоречии друг другу: в рамках одного типа ЗУ улучшение одного из них ведет к ухудшению значения другого. К тому же следует иметь в виду и экономическую целесообразность построения запоминающего устройства с теми

или иными характеристиками при данном уровне развития конкретной технологии. Поэтому в настоящее время запоминающие устройства компьютера строятся по иерархическому принципу (рис. 11.4).

Иерархическая структура памяти позволяет экономически эффективно сочетать хранение больших объемов информации с быстрым доступом к информации в процессе ее обработки.

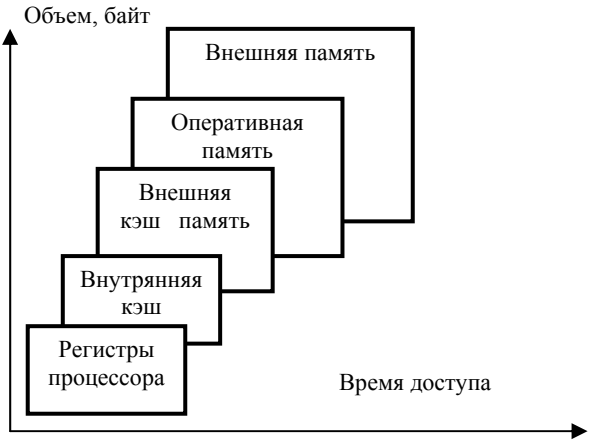


Рис. 11.4. Иерархическая организация памяти в ЭВМ

На нижнем уровне иерархии находится регистровая память (РП) – набор регистров (см. главу 5), входящих непосредственно в состав микропроцессора (CPU). Регистры CPU программно доступны и хранят информацию, наиболее часто используемую при выполнении программы: промежуточные результаты, составные части адресов, счетчики циклов и т.д. Регистровая память имеет относительно небольшой объем (до нескольких десятков машинных слов). РП работает на частоте процессора, поэтому время доступа к ней минимально. Например, при частоте работы процессора 2 ГГц время обращения к его регистрам составит всего 0,5 нс.

Оперативная память – устройство, которое служит для хранения информации (программ, исходных данных, промежуточных и конечных результатов обработки), непосредственно используемой в ходе выполнения программы в процессоре. В настоящее время объем ОП персональных компьютеров составляет несколько от нескольких сотен Мбайт до нескольких Гбайт. Оперативная память работает на частоте системной шины и требует 6-8 циклов синхронизации шины для обращения к ней. Так, при частоте работы системной шины 100 МГц (при этом период равен 10 нс) время обращения к оперативной памяти составит несколько десятков наносекунд (нс).

Для заполнения пробела между РП и ОП по объему и времени обращения в настоящее время используется кэш-память, которая организована как более быстродействующая (и, следовательно, более дорогая) статическая оперативная память со специальным механизмом записи и считывания информации и предназначена для хранения информации, наиболее часто используемой при работе программы. Как правило, часть кэш-памяти располагается непосредственно на кристалле микропроцессора (внутренний кэш), а часть – вне его (внешняя кэш-память). Кэш-память программно недоступна. Для обращения к ней используются аппаратные средства процессора и компьютера.

Внешняя память организуется, как правило, на магнитных и оптических дисках, магнитных лентах. Емкость дисковой памяти достигает сотен гигабайт при времени обращения менее 1 мкс. Магнитные ленты вследствие своего малого быстродействия и большой емкости используются в настоящее время в основном только как устройства резервного копирования данных, обращение к которым происходит редко, а может быть и никогда. Время обращения для них может достигать нескольких десятков секунд.

Регистры процессора составляют его контекст и хранят данные, используемые исполняющимися в конкретный момент командами процессора. Обращение к регистрам процессора происходит, как правило, по их мнемоническим обозначениям в командах процессора.

Независимо от принципов построения в любом ЗУ можно выделить четыре блока, см. рис. 11.5:

- блок поиска слов *SED;*
- накопитель *M;*
- блок разрядных цепей *BFD;*
- блок управления *CO*.

Блоком поиска слов *SED* называется та часть ЗУ, с помощью которой осуществляется отыскание ячейки, в которой хранится искомое слово или в которую надо записать данное слово.

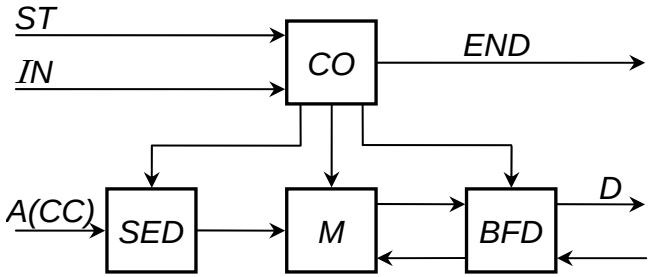


Рис. 11.5. Общая структура ЗУ

Накопителем называется та часть ЗУ, в которой происходит фиксация и хранение слов и из которой при необходимости можно считать записанное ранее слово. Накопитель разбит на отдельные ячейки. В каждой ячейке может храниться часть слова, слово или несколько слов.

Блоком разрядных цепей *BFD* называется та часть ЗУ, по которой при записи слово *D* поступает в накопитель, а при чтении проходит с накопителя на выход ЗУ. В некоторых ЗУ имеются блоки, выполняющие функции как адресных, так и разрядных цепей. Такие блоки называются адресно-разрядными цепями.

Блоком управления *CO* называется та часть ЗУ, которая по стартовому сигналу начала операции *ST* и по коду операции *IN* (чтение или запись) вырабатывает последовательность управляющих сигналов, необходимую для выполнения заданной

операции. Завершение операции фиксируется по сигналу конца операции *END* на специальном выходе *CO*.

По способам построения *SED* ЗУ подразделяются на два вида: с М-поиском (поиском места) и с В-поиском (поиском времени). В ЗУ с М-поиском поиск ячейки осуществляется путем определения по заданному адресу (признаку слова для ассоциативных ЗУ) места (М) в накопителе, которое занимает ячейка с данным адресом (признаком слова). В ЗУ с В-поиском поиск ячейки осуществляется путем определения по заданному адресу (признаку) момента времени (В), когда считывающие элементы могут прочитать (записать) слово из искомой ячейки.

В некоторых типах ЗУ чтение информации сопровождается ее разрушением. Отдельные типы ЗУ позволяют сохранять информацию при отключении энергопитания. Поэтому различают ЗУ с разрушающим и неразрушающим чтением информации, а также ЗУ, сохраняющие и не сохраняющие информацию при отключении питания.

Кэш используется для согласования скорости работы ЦП и основной памяти, см. рис. 11.6. В режиме записи *кэш*-память запоминает копии слов данных, передаваемых процессором в оперативную (основную) память. В режиме считывания происходит обращение к *кэш*-памяти, чтобы сравнить поступающий от процессора теговый адрес с тегами хранящихся в *кэше* слов. Если обнаруживается, что один из тэгов равен теговому адресу, то в *кэше* имеется копия данных адресованной ячейки основной памяти. В этом случае *кеш* формируеи сигнал *Hit=1* (попадание) и выдает данные процессору. В противном случае формируется сигнал *Hit=0,* при котором происходит чтение процессором из основной памяти и запись считываемых данных в *кеш*-память. При *Hit=0* данные в *кеш*-память могут поступать также из процессора.

В вычислительных системах используют многоуровневый *кэш*: *кэш* I уровня (L1), *кэш* II уровня (L2) и т.д. В настольных системах обычно используется двухуровневый *кэш*, в серверных – трехуровневый. *Кэш* хранит команды или данные, которые с большой вероятностью в ближайшее время поступят процессору на обработку. Работа

кэш-памяти прозрачна для программного обеспечения, поэтому кэш-память обычно программно недоступна.

Эксклюзивным называется кэш, в котором данные, хранящиеся в кэш-памяти первого уровня, не обязательно должны быть продублированы в кэшах нижележащих уровней. Инклюзивный кэш – когда любая информация, хранящаяся в кэшах высших уровней, дублируется в кэш-памяти.

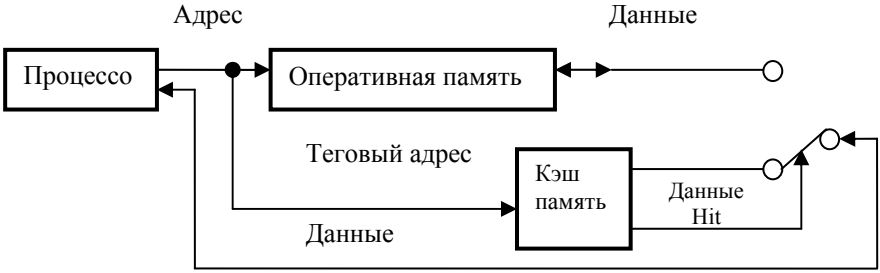


Рис. 11.6. Схема для пояснения принципа работы кэш-памяти

Оперативная память хранит, как правило, функционально-законченные программные модули (ядро операционной системы, исполняющиеся программы и их библиотеки, драйверы используемых устройств и т.п.) и их данные, непосредственно участвующие в работе программ. Оперативная память также используется для сохранения результатов вычислений или иной обработки данных перед пересылкой их во внешнее ЗУ, на устройство вывода данных или коммуникационные интерфейсы.

Каждой ячейке оперативной памяти присвоен уникальный адрес. Организационные методы распределения памяти предоставляют программистам возможность эффективного использования всей компьютерной системы. К таким методам относят сплошную («плоскую»- *flat model*) модель памяти и сегментированную модель памяти (*segmented model*).

Организационные методы распределения памяти позволяют организовать вычислительную систему, в которой

рабочее адресное пространство программы превышает размер фактически имеющейся в системе оперативной памяти, при этом недостаток оперативной памяти заполняется за счет внешней более медленной или более дешевой памяти (винчестер, флэш-память и т.п.) Такую концепцию называют **виртуальной памятью**. При этом линейное адресное пространство может быть отображено на пространство физических адресов либо непосредственно (линейный адрес есть физический адрес), либо при помощи механизма страничной трансляции. Во втором случае линейное адресное пространство делится на страницы одинакового размера, которые составляют виртуальную память. Страничная трансляция обеспечивает отображение требуемых страниц виртуальной памяти в физическое адресное пространство.

Кроме реализации системы виртуальной памяти внешние ЗУ используются для долговременного хранения программ и данных в виде файлов.

11.3. Кэш-память

Для согласования содержимого кэш-памяти и *оперативной памяти* используют три метода записи:

- Сквозная запись (*write through*) – одновременно с кэш-памятью обновляется *оперативная память*.
- Буферизованная сквозная запись (*buffered write through*) – информация задерживается в кэш-буфере перед записью в *оперативную память* и переписывается в *оперативную память* в те циклы, когда ЦП к ней не обращается.
- Обратная запись (*write back*) – используется бит изменения в поле тега, и строка переписывается в *оперативную память* только в том случае, если бит изменения равен 1.

Как правило, все методы записи, кроме сквозной, позволяют для увеличения производительности откладывать и группировать операции записи в *оперативную память*.

В структуре кэш-памяти выделяют два типа блоков данных:

- память отображения данных (собственно сами данные, дублированные из *оперативной памяти*);
- память тегов (признаки, указывающие на расположение кэшированных данных в *оперативной памяти*).

Пространство памяти отображения данных в *кэше* разбивается на строки – блоки фиксированной длины (например, 32, 64 или 128 байт), см. рис. 11.7.

Каждая строка *кэша* может содержать непрерывный выровненный блок байт из *оперативной памяти*. Какой именно блок *оперативной памяти* отображен на данную строку *кэша*, определяется тегом строки и алгоритмом отображения. По алгоритмам отображения *оперативной памяти* в *кэш* выделяют три типа кэш-памяти:

- полностью ассоциативный *кэш*;
- *кэш* прямого отображения;
- множественный ассоциативный *кэш*.

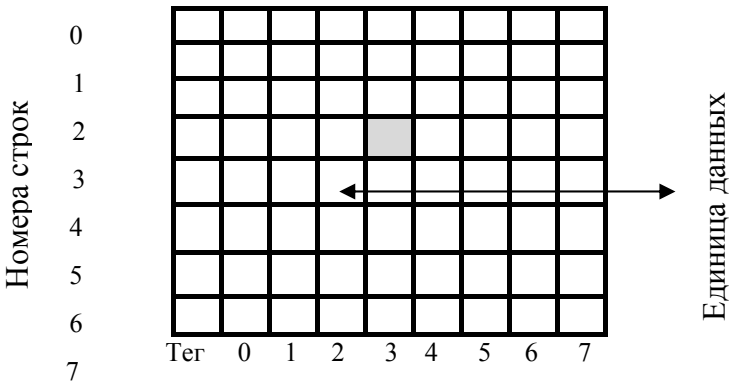


Рис. 11.7. Представление кэш – памяти в виде совокупности строк

Для полностью ассоциативного *кэша* (или кеш-память с произвольной загрузкой – *Fully Associated Cache Memory* –

FASM) характерно, что кэш-контроллер может поместить любой блок *оперативной памяти* в любую строку кэш-памяти, см. рис. 11.8 и 11.9.

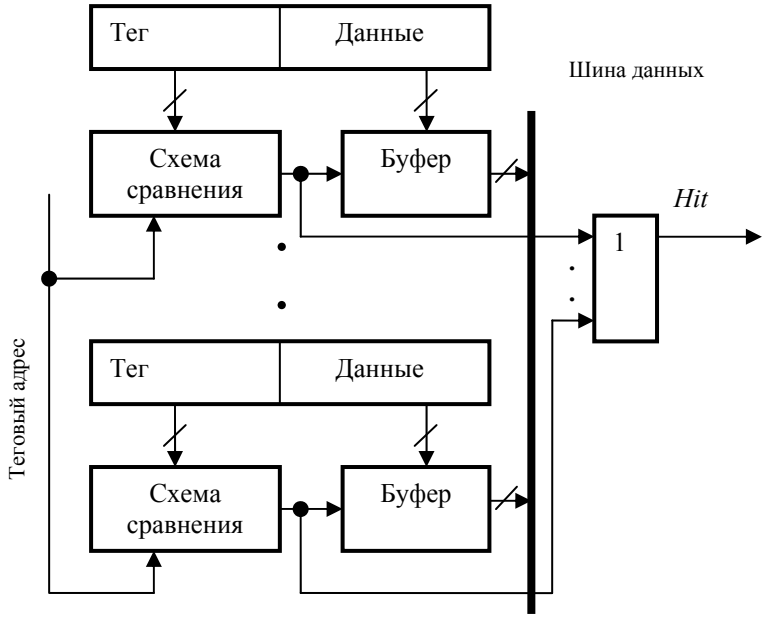


Рис. 11.8. Схема кэш – памяти с произвольной нагрузкой

Физический адрес разбивается на две части:

- смещение в блоке (строке кэша);
- номер блока.

При помещении блока в кэш номер блока сохраняется в теге соответствующей строки. Когда ЦП обращается к кэшу за необходимым блоком, кэш-промах будет обнаружен только после сравнения тегов всех строк с номером блока.

Главное преимущество данной схемы – хорошая утилизация оперативной памяти, т.к. нет ограничений на то, какой блок может быть отображен на ту или иную строку кэш-памяти. К недостаткам следует отнести сложную аппаратную реализацию этого способа, требующую большого количества

компараторов, что приводит к увеличению времени доступа к кэшу и увеличению его стоимости.

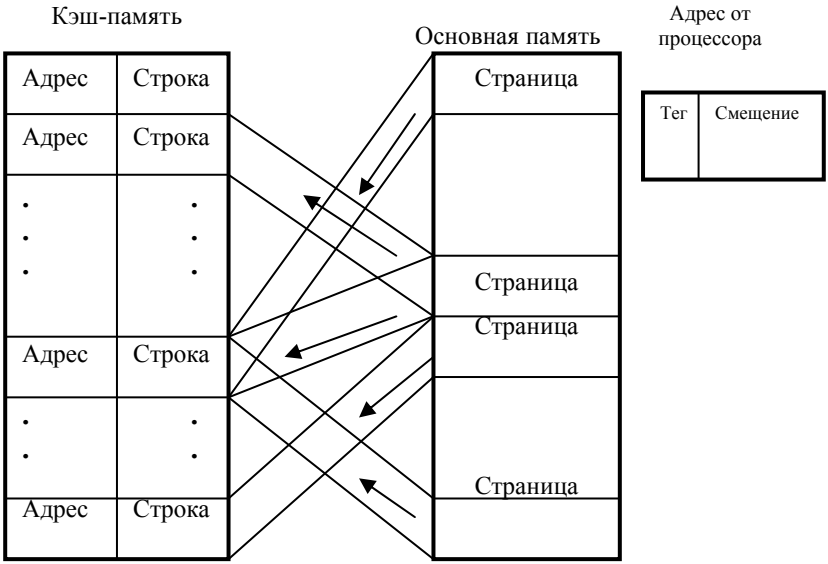


Рис. 11.9. Организация кэш – памяти с произвольной нагрузкой

Альтернативный способ отображения оперативной памяти в кэш – это кэш прямого отображения (или одновходовый ассоциативный кэш). При организации кеша с прямым размещением для одной строки кеш-памяти отводиться несколько страниц основной памяти, см. рис. 11.10 а. Обычно размер страниц составляет 2^n. В этом случае адрес памяти (номер блока) однозначно определяет строку кэша, в которую будет помещен данный блок. Физический адрес разбивается на три части:

- смещение в блоке (строке кэша);
- номер строки кэша;
- тег.

Тот или иной блок будет всегда помещаться в строго определенную строку кэша, при необходимости заменяя собой

хранящийся там другой блок. Когда ЦП обращается к кэшу за необходимым блоком, для определения удачного обращения или кэш-промаха достаточно проверить тег лишь одной строки.

Работа кеш-памяти в режиме считывания показана на рис. 11.10 б. По адресу строки производится считывание. Поле адресов считанной строки сравнивается с теговым адресом, поступившим от процессора. Если теговый адрес совпадает с тегом строки, формируется сигнал *Hit=1* для выдачи слова данных. Выводимое слово определяется по смещению и выводится с помощью мультиплексора.

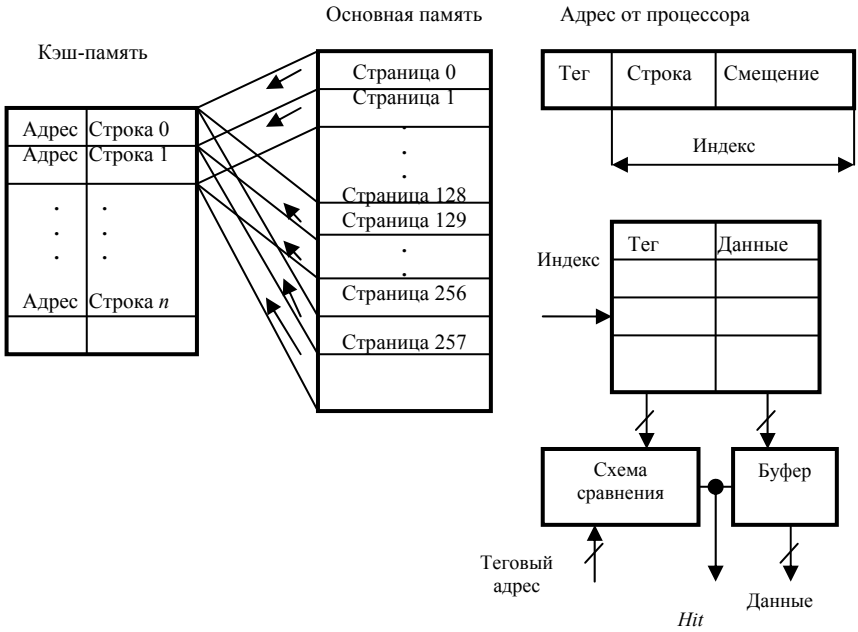


Рис. 11.10. Организация кэш – памяти с прямым размещением

Преимуществами данной схемы являются простота и дешевизна реализации. К недостаткам следует отнести низкую эффективность такого кэша из-за вероятных частых перезагрузок строк Компромиссным вариантом между первой и второйсхемами является множественный ассоциативный кэш или частично-ассоциативный кэш (наборно-ассоциированный кеш), см. рис. 11.11. При этом способе организации кэш-памяти строки объединяются в группы, в которые могут входить 2,4,8,… – строк. В соответствии с количеством строк в таких группах различают 2-входовый, 4-входовый и т.п. наборно-ассоциативный кэш. При обращении к памяти физический адрес разбивается на три части:

- смещение в блоке (строке кэша);
- номер группы (набора);
- тег.

Блок памяти, адрес которого соответствует определенной группе, может быть размещен в любой строке этой группы, и в теге строки размещается соответствующее значение. Таким образом, в рамках выбранной группы соблюдается принцип ассоциативности. Однако, тот или иной блок может попасть только в строго определенную группу, что перекликается с принципом организации кэша прямого отображения. Для того чтобы процессор смог идентифицировать кэш-промах, ему надо будет проверить теги лишь одной группы строк.

Данная схема обладает достоинствами как полностью ассоциативного кэша (хорошая утилизация памяти, высокая скорость), и кэша прямого доступа (простота и дешевизна). Именно поэтому множественный ассоциативный кэш наиболее широко распространен.

Современные процессоры имеют конвейерную архитектуру, при которой блоки конвейера работают параллельно. Таким образом, выборка команды и доступ к данным команды осуществляется на разных этапах конвейера, а использование раздельной кэш-памяти позволяет выполнять эти операции параллельно.

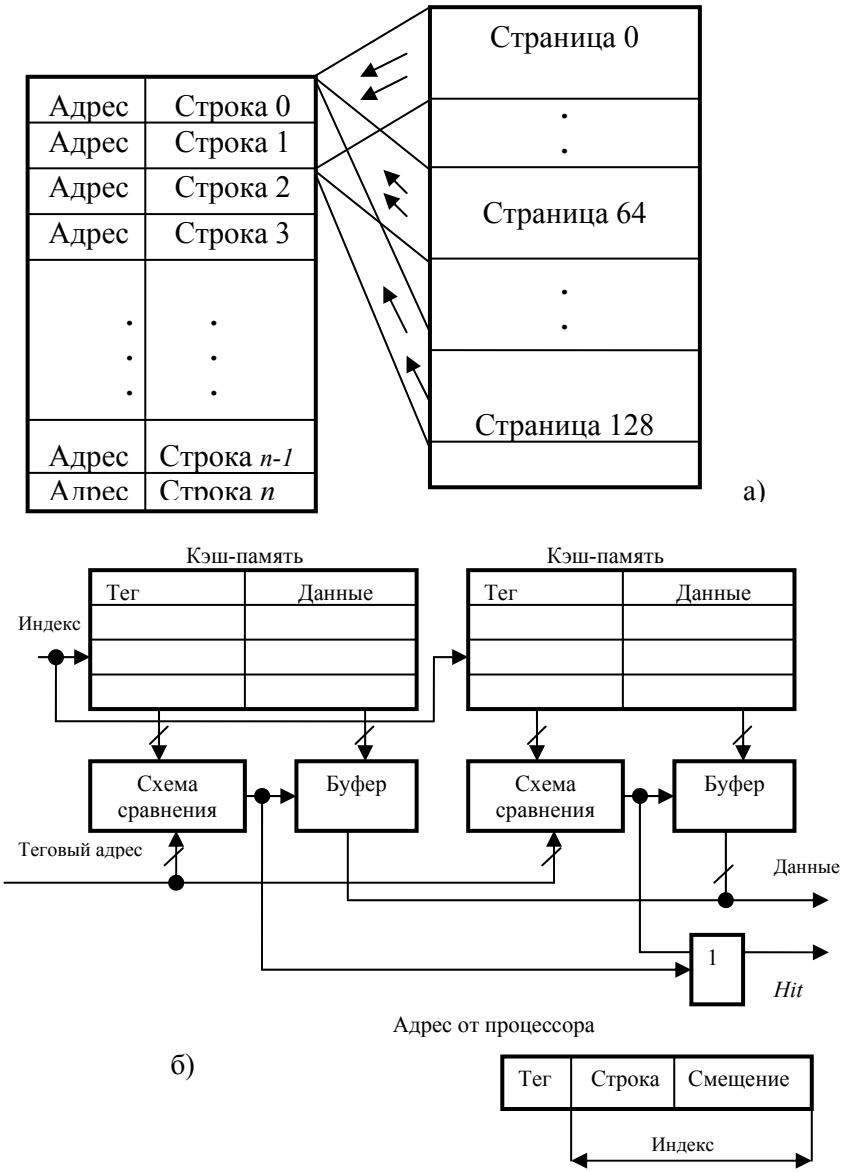


Рис. 11.11. Организация наборно-ассоциативной кэш – памяти

Кэш команд может быть реализован только для чтения, следовательно, не требует реализации никаких алгоритмов обратной записи, что делает этот кэш проще, дешевле и быстрее.

Таким образом, все последние модели IA-32, начиная с Pentium, для организации кэш-памяти первого уровня используют гарвардскую архитектуру.

Критерием эффективной работы кэша можно считать уменьшение среднего времени доступа к памяти по сравнению с системой без кэш-памяти.

11.4. Оперативная память

Характеристики оперативной памяти и особенности ее устройства являются важнейшим фактором, от которого зависит быстродействие компьютера. Это связано с тем, что даже при наличии быстрого процессора скорость выборки данных из памяти может оказаться невысокой. И именно эта невысокая скорость работы с оперативной памятью будет определять быстродействие компьютера. Время цикла работы с памятью t_m обычно заметно больше, чем время цикла центрального процессора t_c. Если процессор инициализирует обращение к памяти, она будет занята в течение времени t_c+t_m и в течение этого промежутка времени ни одно другое устройство, в том числе и сам процессор не смогут работать с оперативной памятью. Таким образом, возникает проблема доступа к памяти. Эта проблема решается специальной организацией оперативной памяти. Принята следующая классификация параллельных компьютеров по архитектуре подсистем оперативной памяти:

- системы с разделяемой памятью, у которых имеется одна большая виртуальная память и все процессоры имеют одинаковый доступ к данным и командам, хранящимся в этой памяти;
- системы с распределенной памятью, у которых каждый процессор имеет свою локальную оперативную память, и к этой памяти у других процессоров нет доступа.

Различие этих двух типов памяти проявляется в структуре виртуальной памяти, то есть памяти, как она "видна" процессору. Физически память обычно делится на части, доступ к которым может быть организован независимо. Различие между разделяемой и распределенной памятью заключается в способе интерпретации адреса.

Для программиста часто бывает важно знать тип оперативной памяти компьютера, на котором он работает. Ведь архитектура памяти определяет способ взаимодействия между различными частями параллельной программы. При выполнении на компьютере с распределенной памятью программа перемножения матриц, например, должна создать копии перемножаемых матриц на каждом процессоре, что технически осуществляется передачей на эти процессоры сообщения, содержащего необходимые данные. В случае системы с разделяемой памятью, достаточно лишь один раз задать соответствующую структуру данных, а затем разместить ее в оперативной памяти.

В большинстве современных компьютеров оперативная память представляет собой динамические модули памяти, которые содержат полупроводниковые БИС ЗУ и организованны по принципу устройств с произвольным доступом.

Массовую оперативную память строят на модулях динамической памяти, а память статического типа используется для построения кеш-памяти в микропроцессоре.

Память динамического типа – DRAM представляет собой набор запоминающих ячеек, которые состоят из конденсаторов и транзисторов, расположенных внутри полупроводниковых микросхем памяти.

Современные технологии *оперативной памяти* в основном используют два схемотехнических решения для повышения быстродействия *DRAM*:

- включение в микросхемы динамической памяти некоторого количества статической памяти;
- синхронная работа памяти и ЦП, т.е. использование внутренней конвейерной архитектуры и чередование адресов.

На протяжении всей истории развития компьютеров создавались различные типы оперативной памяти. Ниже перечислены наиболее распространенные типы DRAM.

FPM DRAM

FPM DRAM - широко использовались в ЭВМ на основе Intel-386 и Intel-486. С появлением МП Pentium была вытеснена EDO DRAM. Ее эффективность обусловлена конвейерной организацией МП. Типичное время доступа при частоте системной шины 66 МГц – 60 нс (35 нс – внутри строки).

EDO DRAM

FPM DRAM стала активно использоваться на компьютерах с процессорами Intel Pentium и выше. Её производительность оказалась на 10 – 15 % выше по сравнению с памятью типа FPM DRAM. Рабочая частота была 40 и 50 МГц, время полного доступа - 60 и 50 нс, а время рабочего цикла – 25 и 20 нс.

SDRAM

Особенностью SDRAM (Synchronous DRAM) является синхронная работа микросхем памяти и процесора, см. рис. 11.12.

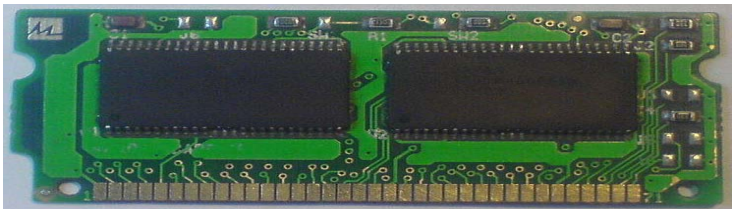


Рис. 11.12. Модуль SDRAM в 72-контактном корпусе SO-DIMM

SDRAM сменила FPM DRAM. Особенностями этого типа памяти являлись использование тактового генератора для синхронизации всех сигналов и использование конвейерной обработки информации. При этом уменьшаются временные задержки в процессе циклов ожидания, и ускоряется поиск данных. Эта синхронизация позволяет контроллеру памяти точно знать время готовности данных. Таким образом, скорость доступа увеличивается благодаря тому, что данные доступны во время каждого такта таймера. Технология SDRAM позволяет использовать множественные банки памяти, функционирующие одновременно, дополнительно к адресации целыми блоками. Микросхемы SDRAM имеют программируемые параметры и свои наборы команд. Длина пакетного цикла чтения-записи может программироваться (1, 2, 4, 8, 256 элементов). Цикл может быть прерван специальной командой без утери данных. Конвейерная организация позволяет инициировать следующий цикл чтения до окончания предыдущего.

Рабочие частоты этого типа памяти могли равняться 66, 100 или 133 МГц, время полного доступа – 40 и 30 нс, а время рабочего цикла – 10 и 7,5 нс.

С этим типом памяти применялась технология Virtual Channel Memory (VCM). VCM использует архитектуру виртуального канала, позволяющую более гибко и эффективно передавать данные с использованием каналов регистра на чипе. Данная архитектура интегрирована в SDRAM.

SDRAM II (DDR)

Synchronous DRAM II, или DDR (Double Data Rate – удвоенная скорость передачи данных), стала развитием SDRAM. DDR основана на тех же принципах, что и SDRAM, но включала некоторые усовершенствования, которые увеличили быстродействие, см. рис. 11.13. DDR давала возможность считывать данные по нарастающему и спадающему фронтам тактового сигнала, выполняя два доступа за время одного обращения стандартной SDRAM, что увеличивает скорость доступа вдвое по сравнению с SDRAM (при одинаковой частоте). Память DDR SDRAM работает на частотах в 100, 133,

166 и 200 МГц, её время полного доступа – 30 и 22,5 нс, а время рабочего цикла – 5, 3,75, 3 и 2,5 нс.

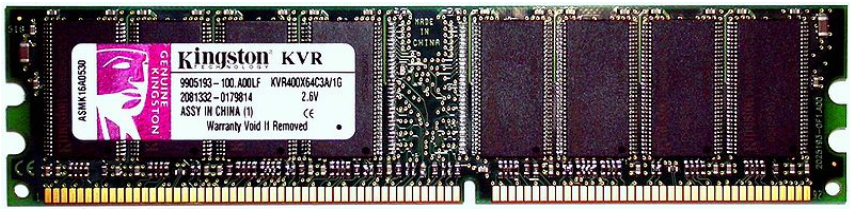


Рис. 11.13. Модуль 1 GiB Kingston DDR-SDRAM со 184 контактами

RDRAM

Тип памяти RDRAM был разработан компанией *Rambus*. Рабочие частоты памяти – 400, 600 и 800 МГц, время полного доступа – до 30 нс, время рабочего цикла – до 2,5 нс.

В основе технологии RDRAM лежит многофункциональный протокол обмена данными между микросхемами, который позволяет передачу данных по упрощенной шине, работающей на высокой частоте. RDRAM представляет собой интегрированную на системном уровне технологию.

Rambus, впервые использованный в графических рабочих станциях в 1995 году, использует уникальную технологию RSL (Rambus Signal Logic – сигнальная логика Rambus), позволяющую использовать частоты передачи данных до 600MHz на обычных системах и материнских платах.

DDR2 SDRAM

DDR2 SDRAM, см. рис. 11.14, стала выпускаться с 2004 года. DDR2 может работать с тактовой частотой шины 200, 266, 333, 337, 400, 533, 575 и 600 МГц. При этом эффективная частота передачи данных соответственно будет 400, 533, 667,

675, 800, 1066, 1150 и 1200 МГц. Время полного доступа - 25, 11,25, 9, 7,5 нс и менее. Время рабочего цикла — от 5 до 1,67 нс.

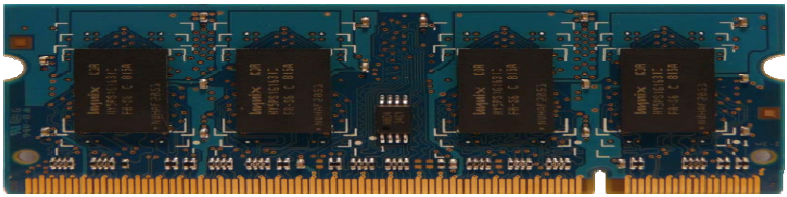


Рис. 11.14. Модуль 1GB DDR2 в 204-контактном корпусе SO-DIMM

DDR3 SDRAM

DDR3 основан на технологиях DDR2 SDRAM со вдвое увеличенной частотой передачи данных по шине памяти. Модули памяти DDR3, отличаются пониженным на 40% энергопотреблением по сравнению с предшественниками (DDR2). Снижение напряжения питания достигается за счёт использования 90-нм (вначале, в дальнейшем 65-, 50-, 40-нм) техпроцесса при производстве микросхем и применения транзисторов с двойным затвором Dual-gate (что способствует снижению токов утечки).

Частота полосы пропускания лежит в пределах от 800 до 2400 МГц (рекорд частоты – более 3000 МГц), что обеспечивает большую пропускную способность по сравнению со всеми предшественниками.

Микросхемы памяти DDR3 производятся исключительно в корпусах типа BGA, см. рис. 11.15.

Рис. 11.15. Модуль 4GB DDR3

11.5. Внешняя память комьютера

Внешняя (долговременная) память – это место длительного хранения данных, не используемых в данный момент в ОЗУ компьютера. Внешняя память является энергонезависимой. Носители внешней памяти, кроме того, обеспечивают транспортировку данных в тех случаях, когда компьютеры не объединены в сети (локальные или глобальные).

Для работы с внешней памятью необходимо наличие накопителя (устройства, обеспечивающего запись и (или) считывание информации) и устройства хранения – носителя.

Основные виды накопителей:

- накопители на гибких магнитных дисках (НГМД);
- накопители на жестких магнитных дисках (НЖМД);
- накопители на магнитной ленте (НМЛ);
- оптические накопители (CD, DVD, Blue-Ray, др.);
- др.

Запоминающие устройства принято делить на виды и категории в связи с их принципами функционирования, эксплуатационно-техническими, физическими, программными и др. характеристиками. Так, например, по принципам функционирования различают следующие виды устройств: электронные, магнитные, оптические и смешанные – магнитооптические. Каждый тип устройств основан на собственной технологии хранения/воспроизведения/записи цифровой информации.

Принцип работы магнитных внешних запоминающих устройств основан на способах хранения информации с использованием магнитных свойств материалов. Как правило, магнитные запоминающие устройства состоят из собственно устройств чтения/записи информации и магнитного носителя, на который, непосредственно осуществляется запись, и с которого считывается информация. Наиболее часто различают: дисковые и ленточные устройства. Общая технология магнитных

запоминающих устройств состоит в намагничивании переменным магнитным полем участков носителя и считывания информации, закодированной как области переменной намагниченности. Дисковые носители, как правило, намагничиваются вдоль концентрических полей – дорожек, расположенных по всей плоскости вращающегося носителя. Намагничивание достигается за счет создания переменного магнитного поля при помощи головок чтения/записи. Головки представляют собой два или более магнитных управляемых контура с сердечниками, на обмотки которых подается переменное напряжение. Изменение величины напряжения вызывает изменение направления линий магнитной индукции магнитного поля и, при намагничивании носителя, означает смену значения бита информации с 1 на 0 или с 0 на 1.

Дисковые устройства делят на гибкие (Floppy Disk) и жесткие (Hard Disk) накопители и носители.

В рамках нашего пособия, мы рассматриваем только оснобенности НЖМД.

Жесткий диск (HDD - Hard Disk Drive или винчестер) устроен следующим образом, см. рис. 11.16, на шпинделе, соединенным с электромотором, расположен блок из нескольких дисков, над поверхностью которых находятся головки для чтения/записи информации.

Рис. 11.6. Схема жесткого диска

При работе головки находятся над поверхностью дисков в воздушном потоке, который создается при вращении дисков.

Основным свойством дисковых магнитных устройств является запись информации на носитель на концентрические замкнутые дорожки с использованием физического и логического цифрового кодирования информации. Плоский дисковый носитель вращается в процессе чтения/записи, чем и обеспечивается обслуживание всей концентрической дорожки, чтение и запись осуществляется при помощи магнитных головок чтения/записи, которые позиционируют по радиусу носителя с одной дорожки на другую.

Для операционной системы данные на дисках организованы в дорожки и секторы. Дорожки (40 или 80) представляют собой узкие концентрические кольца на диске. Каждая дорожка разделена на части, называемые секторами. При чтении или записи устройство всегда считывает или записывает целое число секторов независимо от объёма запрашиваемой информации. Размер сектора на дискете равен 512 байт. Цилиндр – это общее количество дорожек, с которых можно считать информацию, не перемещая головок. Поскольку гибкий диск имеет только две стороны, а дисковод для гибких дисков – только две головки, в гибком диске на один цилиндр приходится две дорожки. В жестком диске может быть много дисковых пластин, каждая из которых имеет две (или больше) головки, поэтому одному цилиндру соответствует множество дорожек. Кластер (или ячейка размещения данных) – наименьшая область диска, которую операционная система использует при записи файла. Обычно кластер – один или несколько секторов.

Накопители на жестких дисках объединяют в одном корпусе носитель (носители) и устройство чтения/записи, а также, нередко, и интерфейсную часть, называемую контроллером жесткого диска.

Ниже приведены основные физические и логические характеристики НЖМД:

1. Тип интерфейса – совокупность линий связи, сигналов, посылаемых по этим линиям, технических средств, поддерживающих линии, а также правила обмена. Серийно выпускаемые внутренние жёсткие диски могут использовать

интерфейсы ATA (IDE и PATA), SATA, eSATA, SCSI, SAS, FireWire, SDIO и Fibre Channel.

2. Емкость – количество данных, которые могут храниться накопителем. Емкость современных жестких дисков (с форм-фактором 3,5 дюйма) на конец 2010 г. достигает 3 Терабайт.

3. Физический размер (форм-фактор). Большинство накопителей для персональных компьютеров и серверов имеют ширину либо 3,5, либо 2,5 дюйма – под размер стандартных креплений для них соответственно в настольных компьютерах и ноутбуках. Также применяются форматы – 1,8 дюйма, 1,3 дюйма, 1 дюйм и 0,85 дюйма.

4. Время произвольного доступа – время, за которое винчестер гарантированно выполнит операцию чтения или записи на любом участке магнитного диска – от 2,5 до 16 мс.

5. Скорость вращения шпинделя - количество оборотов шпинделя в минуту. В настоящее время выпускаются винчестеры со следующими стандартными скоростями вращения: 4200, 5400 и 7200 (ноутбуки), 5400, 7200 и 10 000 (персональные компьютеры), 10 000 и 15 000 об/мин (серверы и высокопроизводительные рабочие станции).

6. Количество операций ввода-вывода в секунду. У современных дисков это около 50 операций/с при произвольном доступе к накопителю и около 100 операций/сек при последовательном доступе.

7. Потребление энергии (важно для мобильных устройств).

8. Уровень шума. Тихими накопителями считаются устройства с уровнем шума около 26 дБ и ниже.

9. Скорость передачи данных (при последовательном доступе). Находится в следующих диапазонах:

- внутренняя зона диска: от 44,2 до 74,5 Мб/с;
- внешняя зона диска: от 60,0 до 111,4 Мб/с.

10. Объём буфера. Буфером называется промежуточная память, предназначенная для сглаживания различий скорости чтения/записи и передачи по интерфейсу. В современных дисках он обычно варьируется от 8 до 64 Мб.

Вопросы для самоконтроля

1. Какая память в ЭВМ является самой быстрой?
2. Какие способы существуют для согласования содержимого кэш-памяти и основной памяти?
3. Перечислите типы кэш-памяти.
4. Какие схемотехнические решения используются для повышения быстродействия *DRAM*?
5. В каких микросхемах динамической памяти используется включение некоторого количества статической памяти?
6. В каких микросхемах динамической памяти используется внутренняя конвейерная архитектура?

Литература для самостоятельной подготовки по теме:
5, 8, 9, 12, 13.

ГЛАВА 12. RISC-ПРОЦЕССОРЫ

В 70-е годы XX века была выдвинута идея создания микропроцессора, оперирующего минимальным набором команд.

Основные черты RISC-архитектуры с аналогичными по характеру чертами CISC-архитектуры отображаются следующим образом (табл. 12.1).

Таблица 12.1

Основные черты архитектуры RISC и CISC

CISC- архитектура	RISC-архитектура
Многобайтовые команды	Однобайтовые команды
Малое количество регистров	Большое количество регистров
Сложные команды	Простые команды
Одна или менее команд за один цикл процессора	Несколько команд за один цикл процессора
Традиционно одно исполнительное устройство	Несколько исполнительных устройств

Одним из важных преимуществ RISC-архитектуры является высокая скорость арифметических вычислений. RISC-процессоры первыми достигли планки наиболее распространенного стандарта IEEE 754, устанавливающего 32-разрядный формат для представления чисел с фиксированной точкой и 64-разрядный формат "полной точности" для чисел с плавающей точкой. Высокая скорость выполнения арифметических операций, в сочетании с высокой точностью вичислений, обеспечивает RISC-процессорам преимущество по быстродействию в сравнении с CISC-процессорами.

Другой особенностью RISC-процессоров является комплекс средств, обеспечивающих безостановочную работу арифметических устройств:

- механизм динамического прогнозирования ветвлений;
- большое количество оперативных регистров;
- многоуровневая встроенная кэш-память.

Организация регистровой структуры – основное достоинство и основная проблема RISC. Практически любая реализация RISC-архитектуры использует трехместные операции обработки, в которых результат и два операнда имеют самостоятельную адресацию – *R1: = R2, R3*. Это позволяет без существенных затрат времени выбрать операнды *R2* и *R3* из адресуемых оперативных регистров и записать в регистр результат операции – *R1*. Кроме того, трехместные операции дают компилятору большую гибкость по сравнению с типовыми двухместными операциями формата "регистр – память" архитектуры CISC. В сочетании с быстродействующей арифметикой RISC-операции типа "регистр – регистр" становятся очень мощным средством повышения производительности процессора.

Вместе с тем опора на регистры является ахиллесовой пятой RISC-архитектуры. Проблема в том, что в процессе выполнения задачи RISC-система неоднократно вынуждена обновлять содержимое регистров процессора, причем за минимальное время, чтобы не вызывать длительных простоев арифметического устройства. Для CISC-систем подобной проблемы не существует, поскольку модификация регистров может происходить на фоне обработки команд формата "память – память".

Существуют два подхода к решению проблемы модификации регистров в RISC-архитектуре:

1) Аппаратный, предложенный в проектах RISC-1 и RISC-2;

2) Программный, разработанный специалистами IBM и Стэндфордского университета.

Принципиальная разница между ними заключается в том, что аппаратное решение основано на стремлении уменьшить время вызова процедур за счет установки дополнительного оборудования процессора, тогда как программное решение базируется на возможностях компилятора и является более экономичным с точки зрения аппаратуры процессора.

Основные особенности RISC-процессоров [16]:

- сокращенный набор команд (от 80 до 150 команд);
- большинство команд выполняется за 1 такт;
- большое количество регистров общего назначения;
- наличие жестких многоступенчатых конвейеров;
- команды имеют простой формат, и используются немногие способы адресации;
- наличие вместительной раздельной кэш-памяти;
- применение оптимизирующих компиляторов, которые анализируют исходный код и частично меняют порядок следования команд.

Самыми крупными разработчиками RISC-процессоров считаются Sun Microsystems (архитектура SPARC – Ultra SPARC), IBM (многокристальные процессоры Power, однокристальные PowerPC – PowerPC 620), Digital Equipment (Alpha – Alpha 21064, 21164, 21264), Mips Technologies (семейство Rxx00 – R 10000), а также Hewlett-Packard (архитектура PA-RISC - PA-8000).

Все RISC-процессоры третьего поколения:

- являются 64-х разрядными и суперскалярными (запускаются не менее 4-х команд за такт);
- имеют встроенные конвейерные блоки арифметики с плавающей точкой;
- имеют многоуровневую кэш-память. Большинство RISC-процессоров кэшируют предварительно дешифрованные команды;
- изготавливаются по КМОП-технологии с 4 слоями металлизации.

Для обработки данных применяется алгоритм динамического прогнозирования ветвлений и метод переназначения регистров, что позволяет реализовать внеочередное выполнение команд.

Повышение производительности RISC-процессоров достигается за счет повышения тактовой частоты и усложнения схемы кристалла. Представителями первого направления являются процессоры Alpha фирмы DEC, наиболее сложными остаются процессоры компании Hewlett-Packard.

Уменьшение набора машинных команд в RISC-архитектуре позволило разместить на кристалле вычислительного ядра большое количество регистров общего назначения. Увеличение количества регистров общего назначения позволило минимизировать обращения к медленной оперативной памяти, оставив для работы с RAM только операции чтения данных из оперативной памяти в регистр и запись данных из регистра в оперативную память, все остальные машинные команды используют в качестве операндов регистры общего назначения.

Ниже перечислены основные аппаратные блоки RISC архитектуры МП:

1. Блок загрузки инструкций включает в себя следующие составные части: блок выборки инструкций из памяти инструкций, регистр инструкций, куда помещается инструкция после ее выборки и блок декодирования инструкций. Эта ступень называется ступенью выборки инструкций.

2. Регистры общего назначения совместно с блоками управления регистрами образуют вторую ступень конвейера, отвечающую за чтение операндов инструкций. Операнды могут храниться в самой инструкции или в одном из регистров общего назначения. Эта ступень называется ступенью выборки операндов.

3. Арифметико-логическое устройство и, если в данной архитектуре реализован, аккумулятор, вместе с логикой управления, которая, исходя из содержимого регистра инструкций, определяет тип выполняемой микрооперации. Источником данных помимо регистра инструкций может быть

4. Набор, состоящий из регистров общего назначения, логики записи и иногда из RAM, образует ступень сохранения данных. На этой ступени результат выполнения инструкций записываются в регистры общего назначения или в основную память.

В RISC-процессоре используется не менее 32 регистров, часто более 100, в то время, как в классических МП обычно 8-16 регистров общего назначения. В результате процессор на 20%-30% реже обращается к оперативной памяти, что также повысило скорость обработки данных. Кроме того, наличие большого количества регистров упрощает работу компилятора по распределению регистров под переменные. Упростилась топология процессора, выполняемого в виде одной интегральной схемы, сократились сроки ее разработки, она стала дешевле.

Классическими представителями RISC архитектуры являются процессоры Alpha 21064 и 21264, см. рис. 12.1.

В RISC-процессорах обработка машинной команды разделена на несколько ступеней, каждую ступень обслуживают отдельные аппаратные средства и организована передача данных от одной ступени к следующей.

Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд.

Выполнение типичной команды можно разделить на следующие этапы:

1. выборка команды – по адресу, заданному счетчиком команд;
2. декодирование команды;
3. выполнение операции, при необходимости обращения к памяти – вычисление физического адреса;
4. обращение к памяти;
5. запоминание результата WB.

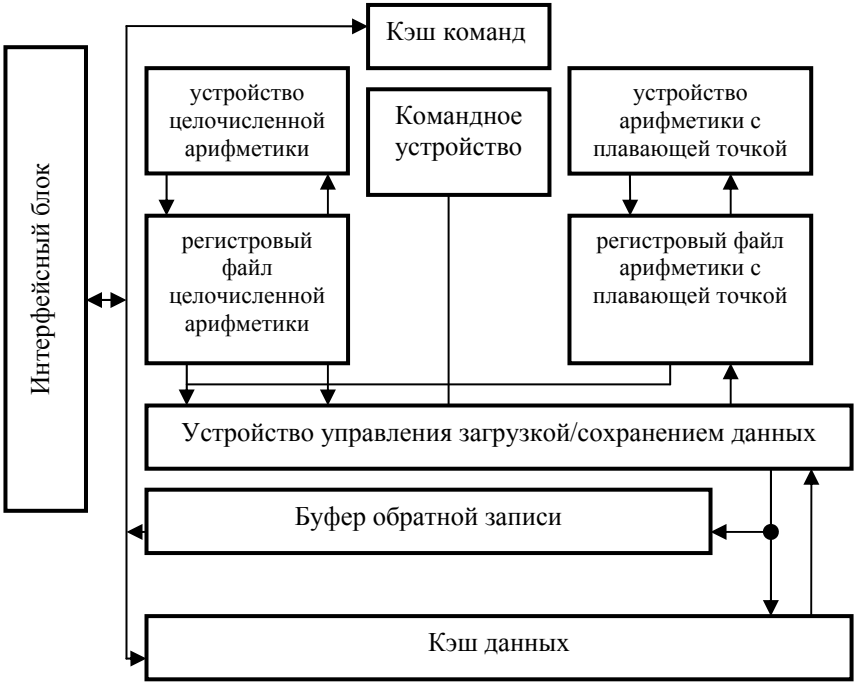


Рис. 12.1. Структура процессора Alpha 21064

Поскольку в процессорах с RISC-архитектурой набор исполняемых команд сокращен до минимума, для реализации более сложных операций приходится комбинировать команды. При этом все команды имеют формат фиксированной длины (например, 12, 14 или 16 бит), выборка команды из памяти и ее исполнение осуществляется за один цикл (такт) синхронизации. Система команд RISC-процессора предполагает возможность равноправного использования всех регистров процессора. Это обеспечивает дополнительную гибкость при выполнении ряда операций.

Следует отметить, что в начале 1990-х годов XX-го века, RISC-архитектуры позволяют получить большую

производительность, чем CISC, за счет использования суперскалярного и ***VLIW***-подхода, а также за счет возможности повышения тактовой частоты и упрощения кристалла с высвобождением площади под кэш.

VLIW – это набор команд, реализующий горизонтальный микрокод. Несколько (4-8) простых команд упаковываются компилятором в длинное слово. Такое слово соответствует набору функциональных устройств. ***VLIW***-архитектуру обычно рассматривают как статическую суперскалярную архитектуру, поскольку распараллеливание кода производится на этапе компиляции, а не динамически во время исполнения.

Имеющиеся реализации ***VLIW*** подхода, например, *VLIW Multiflow*, не получили широкого распространения, кроме – AP-120B/FPS-164/FPS-264 компании Floating Point Systems. Эти процессоры в 80-е годы применялись при проведении научно-технических расчетов. Команда в этих системах содержала ряд полей, каждое из которых управляло работой отдельного блока процессора, так что все командное слово определяло поведение всех блоков процессора.

Вопросы для самоконтроля

1. С чем связано появление RISC-процессоров?
2. Основные особенности RISC-процессоров.
3. Назовите фирмы-разработчики RISC процессоров.
4. Области применения RISC-процессоров.

Литература для самостоятельной подготовки по теме:
5, 8, 16.

ГЛАВА 13. МАТЕРИНСКИЕ ПЛАТЫ И ЧИПСЕТЫ

Материнская плата (*motherboard*) – сложная многослойная печатная плата, на которой устанавливаются основные компоненты компьютера (ЦП, модули ОЗУ и контроллер ОЗУ, загрузочное ПЗУ, контроллеры базовых интерфейсов ввода-вывода).

Материнская плата, как правило, содержит разъёмы (слоты) для подключения дополнительных контроллеров, для подключения которых обычно используются шины USB, PCI и PCI-Express.

Также, на материнской плате установлен чипсет (chipset) – специальный набор микросхем. Например, в ПК чипсет, размещаемый на материнской плате выполняет роль связующего компонента, обеспечивающего совместное функционирование подсистем памяти, центрального процессора, устройств ввода-вывода и т.д.

Как правило, чипсет материнских плат ПК состоит из двух основных микросхем, см. рис. 13.1:

1) контроллер-концентратор памяти (Memory Controller Hub – MCH) или северный мост (northbridge). MCH обеспечивает взаимодействие ЦП с памятью. Соединяется с ЦП высокоскоростной шиной, см. главу 14. В современных ЦП контроллер памяти может быть интегрирован непосредственно в ЦП. В MCH некоторых чипсетах может интегрироваться графический процесор;

2) контроллер-концентратор ввода-вывода (Controller Hub, ICH) или южный мост (southbridge). ICH обеспечивает взаимодействие между ЦП и жестким диском, картами PCI, низкоскоростными интерфейсами PCI Express, интерфейсами IDE, SATA, USB и т.д.

Материнские платы характеризуются форм-фактором (ФФ). ФФ – стандарт, определяющий размеры материнской платы для компьютера, места ее крепления к корпусу; расположение на ней интерфейсов шин, портов ввода/вывода, сокета центрального процессора (если он есть) и слотов для оперативной памяти, а также тип разъема для подключения блока питания.

Форм-фактор носит рекомендательный характер. Однако подавляющее большинство производителей предпочитают соблюдать спецификацию. Ниже перичислены основные форм-факторы материнских плат.

Устаревшие: Baby-AT (1990 г.); AT (1984 г.); LPX (1987 г.).

Современные: ATX (1995 г.); microATX (1997 г.); Flex-ATX (1999 г.); NLX (1997 г.); WTX (1999 г.), CEB (2005 г.).

Внедряемые: Mini-ITX и Nano-ITX (2003 г.); Pico-ITX (2007 г.); BTX (2004 г.), MicroBTX и PicoBTX (2007 г.).

Существуют материнские платы, не соответствующие никаким из существующих форм-факторов. Обычно это обусловлено либо тем, что производимый компьютер узкоспециализирован, либо желанием производителя материнской платы самостоятельно производить и периферийные устройства к ней, либо невозможностью использования стандартных компонентов (например Apple Computer, Commodore, Silicon Graphics, Hewlett Packard).

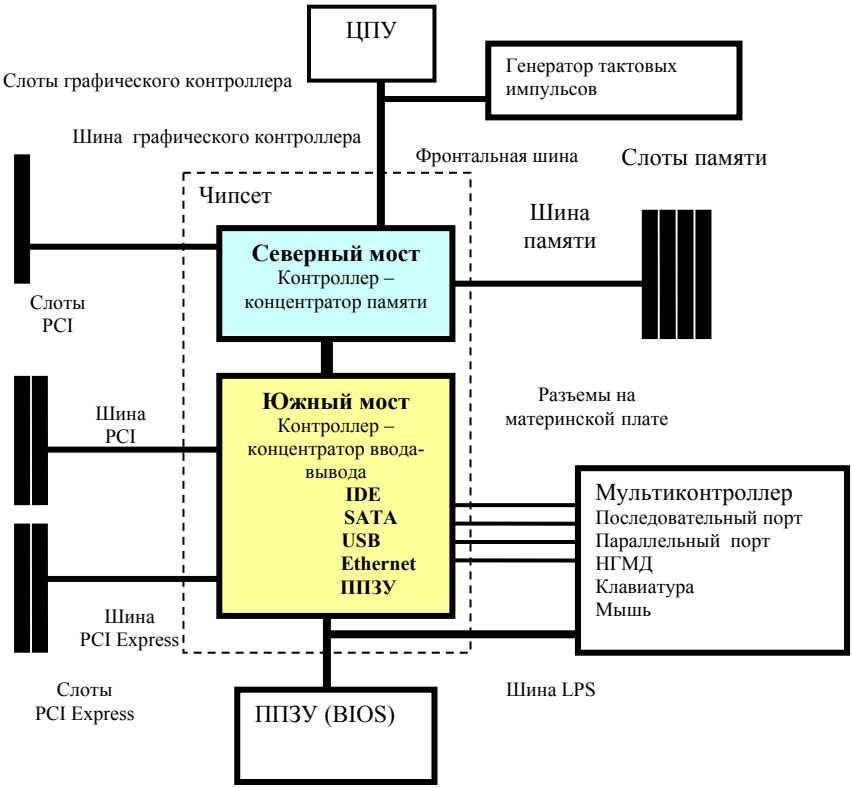


Рис. 13.1. Компоненты материнской платы

На рис. 13.2 показаны основные компоненты материнской платы.

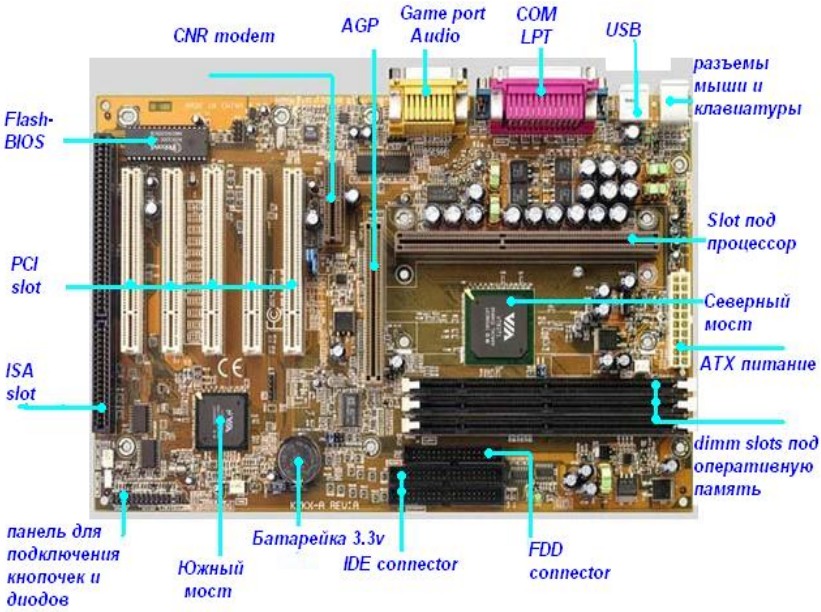


Рис. 13.2. Основные компоненты материнской платы

Как указывалось выше, на чипсет (ChipSet) возлагается основная нагрузка по обеспечению центрального процессора данными и командами, а также, по управлению прерферией – видео карты, звуковая система, оперативная память, дисковые накопители и различные порты ввода/вывода. В последних разработках в состав микросхем наборов для интегрированных плат стали включаться и контроллеры внешних устройств. Внешне микросхемы чипсета выглядят, как самые большие после процессора, по количество выводов от нескольких десятков до двух сотен. Название набора обычно происходит от маркировки основной микросхемы, например, i810,i810E,i440BX, VIA Apollo pro 133A, SiS630, UMC491, i82C437VX и т.п. При этом используется только код микросхемы внутри серии: например, полное наименование SiS471 – SiS85C471. Последние разработки используют и собственные имена; в ряде случаев это – фирменное название

(INTEL,VIA, Viper) Тип набора в основном определяет функциональные возможности платы: типы поддерживаемых процессоров, структура объем кэша, возможные сочетания типов и объемов модулей памяти, поддержка режимов энергосбережения, возможность программной настройки параметров и т.п. На одном и том же наборе может выпускаться несколько моделей системных плат, от простейших до довольно сложных с интегрированными контроллерами портов, дисков, видео и т.д.

Разъём центрального процессора – гнездовой или щелевой разъём, предназначенный для установки центрального процессора. Использование разъёма вместо прямого распаивания процессора на материнской плате упрощает замену процессора для модернизации или ремонта компьютера. Разъём может быть предназначен для установки собственно процессора или CPU-карты. Каждый разъём допускает установку только определённого типа процессора или CPU-карты.

Тип разъемов конструктивно представляет пластиковый разъем с зажимающей защелкой, расположенной сбоку корпуса разъема, предназначенной для предотвращения самопроизвольное выпадения процессора. При установке процессора защелка должна быть максимально поднята вверх.

Ниже приведены основные типы разъемов для процессоров Intel и AMD.

Intel

Socket 7 – стандартный ZIF (Zero Input Force) – разъёмом с 296 контактами, использующийся всеми процессорами класса P5 - Intel Pentium, AMD K5 и K6, Cyrix 6x86 и 6x86MX и Centaur Technology IDT-C6.

Socket 8 – нестандартный ZIF имеет 387 контактов и несовместим с Socket 7, и предназначен для установки в него процессора класса P6 - Pentium Pro. Так как ядро процессора и кэш были объединены на одном кристалле то и форма его получилась прямоугольной а не квадратной как у Socket 7.

Socket 370 – нестандартный ZIF несовместим ни с Socket 7, ни с Socket 8, предназначен для установки в него более дешевого прототипа P6 Celeron.

Socket 423, 478 – Pentium 4 и Celeron.

Socket 479 – Pentium M и Celeron M.

PAC418 – Itanium

PAC611 – Itanium 2, HP PA-RISC 8800 и 8900.

Socket B (LGA 1366) – Core i7 с интегрированным трехканальным контроллером памяти.

Socket H (LGA 1156) – Core i7/Core i5/Core i3 с интегрированным двуканальным контроллером памяти.

Socket J (LGA 771) – Intel Xeon серий 50xx, 51xx, 53xx, 54xx.

Socket M – Core Solo, Core Duo и Core 2 Duo.

Socket N – Dual-Core Xeon LV.

Socket P – замена Socket 479.

AMD

Super Socket 7 – AMD K6-2, AMD K6-2+, AMD K6-III, Rise mP6, Cyrix MII/6x86MX.

Socket 754 – Athlon 64 нижнего уровня и Sempron с поддержкой только одноканального режима работы с памятью типа DDR.

Socket 939 – Athlon 64 и Athlon 64 FX с поддержкой двухканального режима работы с памятью типа DDR.

Socket 940 – Opteron.

Socket AM2 – с поддержкой памяти типа DDR2, но не совместим с Socket 940.

Socket AM2+ – замена для Socket AM2, прямая и обратная совместимость с сокетом AM2 для всех планируемых материнских плат и процессоров.

Socket AM3 – с поддержкой памяти типа DDR3. Замена для Socket AM2+.

Рассмотрим устройство чипсетов на примерах Intel P55 Express (2004 г.) и Intel P67/H67 2010 г.

Ключевые характеристики чипсета Intel P55 Express выглядят следующим образом:

- поддержка процессоров Core i7 и Core i5, при подключении к этим процессорам по шине DMI (с пропускной способностью ~2 ГБ/с);

- до 8 портов PCIEx1 (PCI-E 2.0, но со скоростью передачи данных PCI-E 1.1);
- до 4 слотов PCI;
- 6 портов Serial ATA II на 6 устройств SATA300 (SATA-II);
- 14 устройств USB 2.0 (на двух хост-контроллерах EHCI) с возможностью индивидуального отключения;
- MAC-контроллер Gigabit Ethernet (подключение PHY-контроллера i82577/82578 осуществляется через любой свободный порт PCIEx1 чипсета);
- High Definition Audio (7.1);

прочее.

По сравнению с предшествующим поколением чипсетов (ICH10R), у Intel P55 Express, произошли в основном лишь количественные изменения: 14 портов USB вместо 12, 8 портов PCI-E вместо 6. Но у Intel P55 Express, есть одно новшество более системного характера: теперь «периферийный» контроллер PCI Express тоже соответствует второй версии стандарта. Однако соответствует он ей лишь в части новых технологий этой версии, а вот скорость работы принудительно выставлена на уровне PCI-E 1.1 – 250+250 МБ/с (до 250 МБ/с в каждом из двух направлений одновременно). Таким образом, контроллеры с первыми реализациями ожидаемых вскоре USB 3.0 и Serial ATA III предстоит подключать к портам, работающим на прежней скорости.

Новые чипсеты Intel P67/H67 (рис. 13.3) сохранили прежний уровень функциональности. Но архитектурно P6 один-единственный мост, подключаемый к процессору по шине DMI (только теперь новой версии), а также по специальному интерфейсу FDI в случае «интегрированного» чипсета, и главной задачей которого является обеспечить функционирование основной массы периферийных устройств на материнской плате.

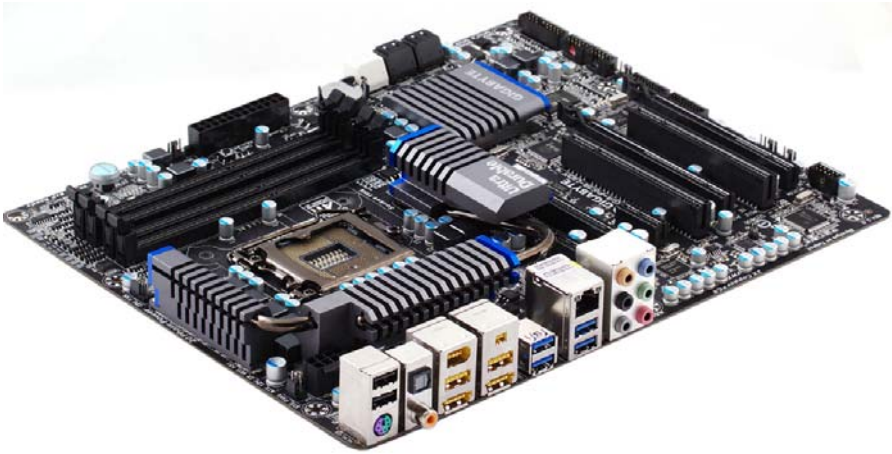


Рис. 13.3. Системная плата Gigabyte P67A-UD5 на чипсете Intel P67

Базовые характеристики чипсета Intel P67/H67 выглядят следующим образом:

- поддержка всех новых процессоров на ядре Sandy Bridge при подключении к этим процессорам по шине DMI 2.0 (с пропускной способностью ≈4 ГБ/с);
- до 8 портов PCIEx1 (PCI-E 2.0);
- 2 порта Serial ATA III на 2 устройства SATA600 и 4 порта Serial ATA II на 4 устройства SATA300;
- 14 устройств USB 2.0 (на двух хост-контроллерах EHCI) с возможностью индивидуального отключения;
- MAC-контроллер Gigabit Ethernet и специальный интерфейс (LCI/GLCI) для подключения PHY-контроллера (i82579 для реализации Gigabit Ethernet, i82562 для реализации Fast Ethernet);
- High Definition Audio (7.1);
- прочее.

Вопросы для самоконтроля

1. Назначение материнской платы?
2. Назначение южного и северного моста.
3. Что такое слот и форм-фактор?
4. Основные характеристики чипсетов.
5. Какие основные типы разъемов для процессоров Intel и AMD Вы знаете?

Литература для самостоятельной подготовки по теме:
5, 8, 16, 20.

ГЛАВА 14. СИСТЕМА ПРЕРЫВАНИЙ И ИСКЛЮЧЕНИЙ В АРХИТЕКТУРЕ IA-32

14.1. Система прерываний и исключений процессора

Прерывания и исключения – это события, которые указывают на возникновение в системе или в выполняемой в данный момент задаче определенных условий, требующих вмешательства процессора. Возникновение таких событий вынуждает процессор прервать выполнение текущей задачи и передать управление специальной процедуре либо задаче, называемой обработчиком прерывания или обработчиком исключения. Различные синхронные и асинхронные события в системе на основе ЦП IA-32 можно классифицировать следующим образом [5, 13, 20], см. рис. 14.1.

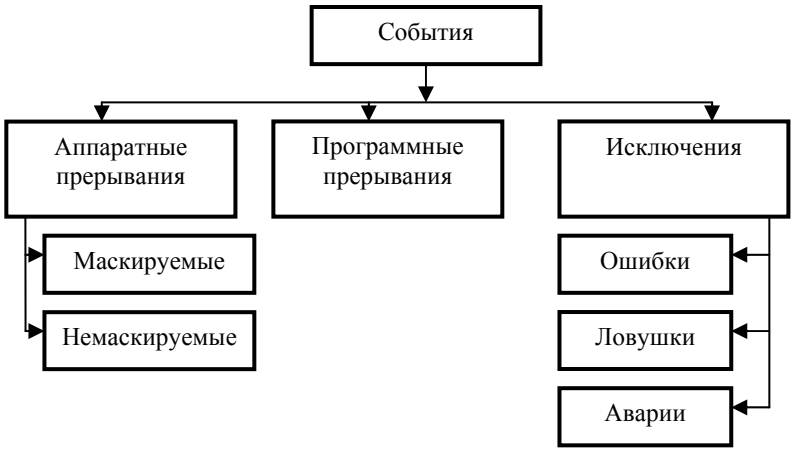


Рис. 14.1. Классификация событий в системе на основе ЦП IA-32

Система прерывания появилась в процессорах ЭВМ второго поколения, которые использовались, в основном, в качестве программных устройств управления различными объектами.

Основными причинами появления системы прерывания являются:

- стремление уменьшить простои ЭВМ при возникновении внештатных ситуаций в процессоре (попытки деления на ноль, использование несуществующей команды, сбой в устройстве и т.д.),
- желание загрузить полезной работой процессор, во время, когда он ожидает сигнал от управляемого объекта, т.е. желание реализовать фоновую работу ЭВМ.

Не смотря на то, что система прерывания появилась для разрешения двух, разных по сущности проблем, в современных ЭВМ используется единый механизм прерывания. (В истории развития архитектур специализированных ЭВМ имелись примеры использования двух раздельных систем прерывания.)

В современных ЭВМ, не смотря на единый механизм прерывания, различают особенности процедуры обработки прерываний при внештатных ситуациях в процессоре и при приходе сигналов прерывания от внешних устройств.

С переходом на многопрограммные режимы работы система прерывания стала обязательным компонентом всех ЭВМ.

Система прерывания – это эффективный способ реализации контрольных и управляющих функций операционной системы для поддержки заданных режимов работы ЭВМ, как аппаратно-программного комплекса.

Процедура прерывания заключается в переходе на подпрограмму обработки прерывания с возможностью возврата в основную программу.

В этом изложении процедура прерывания совпадает с процедурой "вызова процедуры с возвратом". Выполнение каждой команды (кроме NOT) меняет последействие, записывая результат выполнения в память, РОН или в регистр состояния.

Команды вызова процедуры с возвратом расставляются программистом с учетом такого последействия, Но прерывание – событие, в общем случае, случайное по отношению к программе. Программа обработки прерывания может изменить контекст программы, что может привести к нарушению корректности выполнения последующих команд основной программы.

По этой причине, в процедуру прерывания включают этап сохранение контекста программы, а при возвращении из процедуры – этап его восстановления.

Сохранение контекста может быть реализовано схемными, программными или схемно-программными способами. Наиболее часто используют схемно-программное сохранение контекста. При этом контекст делится на основную часть и дополнительную.

В основную часть входят коды условий и биты управления, собранные в регистр состояния программы (*PSW* – Program Status Word или PS). Для МП Intel – это регистр флагов (*EFLAGS*).

В дополнительную часть входит содержимое РОН и ячеек памяти.

Основная часть сохраняется аппаратно, обычно в стек, Содержимое РОН сохраняется и восстанавливается программой обработки прерывания, причем сохраняются и восстанавливаются только РОНы, используемые программой обработки прерывания.

Так как адресные пространства программ обработки прерывания и основной программы всегда разделяются, сохранение ячеек памяти в процедуре прерывания не предусмотрено.

Ловушка (*Trap*) – это реакция системы на появление в работе процессора нештатных ситуаций: попытка деления на ноль, выявление несуществующего адреса и т. д. К ловушкам относятся также специальные команды программировуемого прерывания.

Прерывание (Interrupt) – это реакция системы на запрос внешнего устройства (сигнал прерывания) по выполнению процессором определенной процедуры управления.

Таким образом, ловушки выявляется схемами фиксации особых ситуаций (штатных или не штатных в процессоре), требующих контроля со стороны операционной системы. При этом с каждой схемой фиксации особых ситуаций связанны определенные программы обработки прерывания.

Сигналы прерывания – это сигналы от внешних, по отношению к процессору, устройств в моменты, требующих управляющих действий со стороны операционной системы.

Основными вопросами реализации системы прерывания являются:

- прием сигналов прерывания и ловушек. А затем, выделение приоритетного сигнала;
- определение момента выполнения процедуры прерывания;
- выбор процедуры прерывания (модели сохранение контекста и перехода на программу обработки прерывания);
- выбор процедуры возврата из процедуры прерывания.

Прием сигналов (запросов) прерывания производится для определения конкретной программы обработки прерывания. В большинстве ЭВМ система прерывания предусматривает 256 программ обработки ситуаций, вызвавшей запрос на прерывание. Каждая программа обработки прерывания связана с определенным источником прерывания. Определение источника сигнала прерывания является определением программы обработки прерывания.

Однопроцессорная система не может выполнять программы обработки прерывания одновременно по нескольким запросам. Поэтому используется приоритетное выделение принимаемых сигналов прерывания. Здесь возможны варианты.

Прием сигналов может производиться по общему проводу. В этом варианте, для выявления источников сигналов прерывания производится последовательный опрос процессором всех возможных источников сигнала прерывания.

Последовательность опроса определяет приоритет сигнала прерывания устройства. При опросе все устройства,

пославшие сигналы прерывания, в своих ответах определяют программы обработки прерывания.

Альтернативным способом приоритетного определения источника сигнала прерывания является прием сигналов прерываний от внешних устройств по индивидуальным проводам и фиксации их на отдельном регистре прерывания.

В этом случае в систему прерывания вводится схема приоритетного анализа поступивших сигналов. На рис. 14.2 представлена упрощенная схема механизма системы прерывания. В этой схеме рассматривают только два крайних (по реализации выделения сигналов) классов системы прерывания.

Эта схема задает только жесткий приоритет сигналов прерывания. Но, важность первоочередной обработки причин прерываний может меняться. Для этого в систему прерывания вводят регистр маски, на те прерывания, приоритеты которых желательно менять.

Сигнал прерывания, в общем случае, не является сигналом немедленного действия. Обычно сигнал прерывания (от внешних устройств) или ловушки (от схем контроля выполнения операций) просто фиксируется в соответствующем разряде регистра прерывания.

Выполнение перехода на программу обработки ситуации всегда происходит определенная задержка (время реакции системы прерывания).

Здесь возможны несколько вариантов в выборе момента начала процедуры прерывания.

1. Программы состоят из последовательностей автономных участков (процедур) с минимальным набором передаваемой информации. Обычно это содержимое памяти или РОН. Это точки с минимумом контекста. Программист может отмечать эти точки особыми командами, для использования их в качестве команд прерывания. Но это решение значительно увеличивает среднее время реакции системы прерывания.

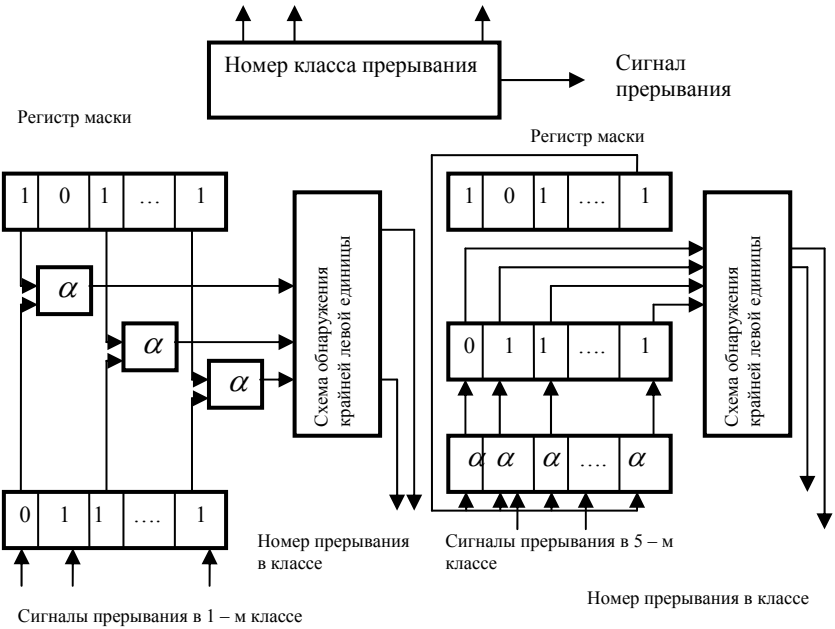


Рис. 14.2. Упрощенная схема механизма системы прерывания

2. Среднее время реакции системы прерывания значительно (в разы) сокращается, если процедуру прерывания производить сразу после окончания текущей команды.

3. Имеется еще возможность уменьшения времени реакции системы прерывания, если процедуру прерывания начинать сразу после окончания не команды, а текущей части микрооперации, во время которой появился запрос прерывания. Но при этом приходится дополнительно сохранять результат текущей микрооперации.

В подавляющем большинстве архитектур ЭВМ процедура прерывания производится по окончании текущей команды.

На рис. 14.3 показана упрощенная схема цикла процессора выполнения команды с фиксацией ловушки (внештатной ситуации) и проверки сигнала прерывания.

Выполнение команд в процессоре является циклическим. Начинаем рассмотрение с процедуры выборки команды. Это начало цикла. Результатом этого этапа является выборка и анализ полей команды, адресов операндов и результата. Следующими этапами являются этапы выборка операндов. Последующие этапы производят дешифрацию кода операции команды и передачу команды в блоки выполнения. Последним этапом является сохранением результата и записью признака результата в регистр состояния.

Внешние прерывания генерируются по аппаратному сигналу, поступающему от периферийного оборудования, когда оно требует обслуживания. Процессор определяет необходимость обработки внешнего прерывания по наличию сигнала на одном из контактов *INTR#* или *NMI#*. При появлении сигнала на линии *INTR#* внешний контроллер прерываний (например, 8259A) должен предоставить процессору вектор (номер) прерывания.

Прерывания, которые генерируются при поступлении сигнала на вход *INTR#*, называют маскируемыми аппаратными прерываниями. Бит *IF* в регистре флагов позволяет заблокировать (*замаскировать*) обработку таких прерываний.

Прерывания, генерируемые сигналом *NMI#*, называют немаскируемыми аппаратными прерываниями. Немаскируемые прерывания не блокируются флагом IF. Пока выполняется обработчик немаскируемого прерывания, процессор блокирует получение немаскируемых прерываний до выполнения инструкции IRET, чтобы исключить одновременную обработку нескольких немаскируемых прерываний.

Прерывания всегда обрабатываются на границе инструкций, т.е. при появлении сигнала на контакте *INTR#* или *NMI#* процессор сначала завершит выполняемую в данный момент инструкцию (или итерацию при наличии префикса повторения), а потом начнет обрабатывать прерывание. Помещаемый в стек обработчика адрес очередной инструкции позволяет корректно возобновить выполнение прерванной программы.

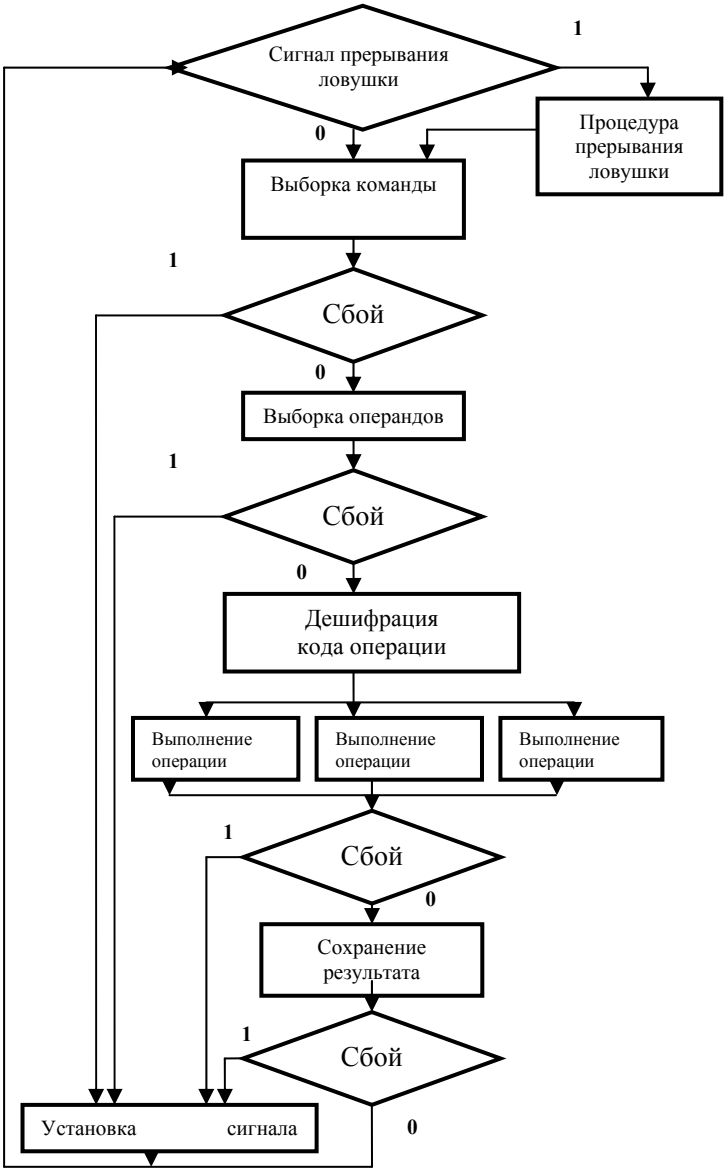


Рис. 14. 3. Схема цикла процессора выполнения команды с фиксацией ловушки

Исключения являются для процессора внутренними событиями и сигнализируют о каких-либо ошибочных условиях при выполнении той или иной инструкции. Источниками исключений являются три типа событий:

- генерируемые программой исключения, позволяющие программе контролировать определенные условия в заданных точках программы;
- исключения машинного контроля, возникающие в процессе контроля операций внутри чипа и транзакций на шине процессора (Pentium 4);
- обнаруженные процессором ошибки в программе (деление на ноль, нарушение правил защиты, отсутствие страницы и т.п.)

14.2. Расширенный программируемый контроллер прерываний (APIC)

Начиная с модели Pentium, процессоры IA-32, содержат встроенный расширенный программируемый контроллер прерываний (APIC).

APIC предназначен для регистрирования прерываний от источников внутри процессора или от внешнего контроллера прерываний и передачи их ядру процессора для последующей обработки.

Встроенный APIC различает следующие источники прерываний.

1. От локальных устройств. Прерывания, генерируемые по фронту или уровню сигнала (например, контроллер прерываний типа 8259A).
2. От внешних устройств. Прерывания, генерируемые по фронту или уровню сигнала, который поступает от устройства, подключенного к внешнему контроллеру прерываний.
3. Межпроцессорные (IPI). В многопроцессорных системах один из процессоров может прервать другой при

4. От таймера APIC. Встроенный APIC содержит таймер, который можно запрограммировать на генерацию прерывания по достижении определенного отсчета.

5. От таймера монитора производительности.

6. От термодатчика. Многие процессоры содержат встроенный блок температурного контроля, который можно запрограммировать на генерацию прерываний.

7. Внутренние ошибки APIC. Встроенный APIC может генерировать прерывания при возникновении внутренних ошибочных ситуаций (например, при попытке обратиться к несуществующему регистру APIC).

Источники 1, 4, 5, 6, 7 считаются локальными источниками прерываний и обслуживаются специальным набором регистров APIC, называемым таблицей локальных векторов (*LVT* - local vector table).

Источники 2 и 3 обрабатываются APIC через механизм сообщений, передаваемых либо по выделенной трехпроводной шине APIC или по системной шине (например, PCI).

14.3. Обработка прерываний на основе контроллера 8259A

Контроллер прерываний 8259A представляет собой устройство, реализующее до восьми уровней запросов на прерывания, с возможностью программного маскирования и изменения порядка обслуживания прерываний.

Контроллер прерываний состоит из следующих блоков, см. рис. 14.4:

RGI – регистр запретов прерываний; хранит все уровни, на которые поступают запросы IRQx.

PRB – схема принятия решений по приоритетам; схема идентифицирует приоритет запросов и выбирает запрос с наивысшим приоритетом.

ISR – регистр обслуживаемых прерываний; сохраняет уровни запросов прерываний, находящиеся на обслуживании контроллера прерываний.

RGM – регистр маскирования прерываний; обеспечивает запрещение одной или нескольких линий запросов прерывания.

BD – буфер данных; предназначен для сопряжения с системной шиной данных.

RWCU – блок управления записью/чтением; принимает управляющие сигналы от микропроцессора и задает режим функционирования контроллера прерываний.

CMP – схема каскадного буфера-компаратора; используется для включения в систему нескольких контроллеров.

CU – схема управления; вырабатывает сигналы прерывания и формирует трехбайтовую команду CALL для выдачи на шину данных.

Один контроллер 8259A способен обслуживать прерывания от 8 источников. Для обслуживания большего количества устройств используется каскадное включение контроллеров.

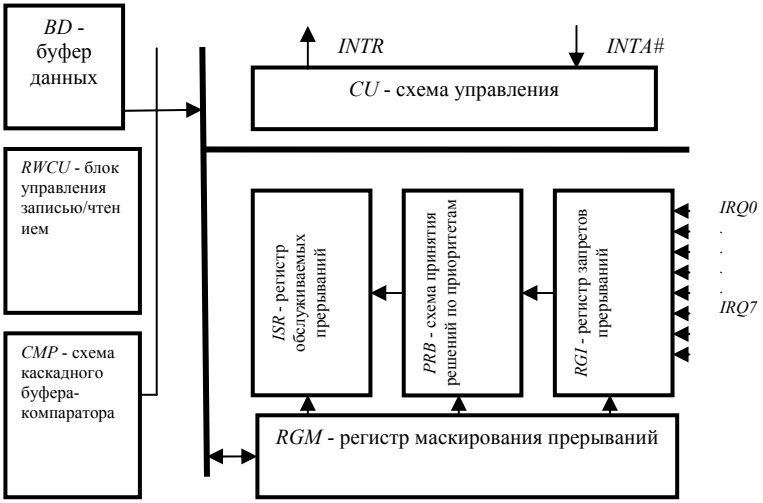


Рис. 14.4. Структура контроллера прерываний 8259A

Поскольку в каждый момент времени может поступить более чем один запрос на прерывание, контроллер прерываний имеет схему приоритетов. В основном режиме – режиме полного вложения, – до тех пор, пока установлен разряд в регистре *ISR*, соответствующий запрашиваемому прерыванию, все последующие запросы с таким же или более низким приоритетом игнорируются, подтверждаются лишь запросы с более высоким приоритетом.

В циклическом режиме используется круговой порядок использования приоритетов. Последнему обслуженному запросу присваивается низший приоритет, следующему по кругу - наивысший, что гарантирует обслуживание остальных устройств до очередного обслуживания данного устройства.

Контроллер допускает маскирование отдельных запросов прерываний, что позволяет устройствам с более низким приоритетом получить возможность генерировать прерывания. Режим специального маскирования разрешает прерывания всех уровней, кроме уровней, обслуживаемых в данный момент.

Вопросы для самоконтроля

1. Какие исключительные ситуации могут возникать при работе ЭВМ?
2. Чем маскируемые прерывания отличаются от немаскируемых?
3. В каком регистре контроллера прерываний сохраняются уровни запросов прерываний, находящиеся на обслуживании?
4. В каких режимах работы контроллера прерываний подтверждаются лишь запросы с более высоким приоритетом?
5. В каких режимах работает контроллер прямого доступа к памяти?

Литература для самостоятельной подготовки по теме:
5, 8, 16, 20.

Модуль 4
Типы и характеристики интерфейсов. Устройства ввода/ вывода.

ГЛАВА 15. ТИПЫ И ХАРАКТЕРИСТИКИ ИНТЕРФЕЙСОВ

15.1. Системные интерфейсы

Интерфейс – это аппаратное и программное обеспечение (элементы соединения и вспомогательные схемы управления, их физические, электрические и логические параметры), предназначенное для сопряжения систем или частей системы (программ или устройств). Под сопряжением подразумеваются следующие функции:

- выдача и прием информации;
- управление передачей данных;
- согласование источника и приемника информации.

Системные интерфейсы, как и прочие так же имеют различные спецификации, соответственно и характеристики у них будут разные.

SATA – (Serial ATA) – интерфейс с последовательной передачей, предназначен для работы с накопителями данных. Конектор интерфейса имеет семь контактов. На данный момент интерфейс имеет три системных стандарта и один для подключения внешних устройств.

• SATA 1 – первый вариант стандарта, его пропускная способность составляет 1,2 Гбит/с. Работы шины осуществляется на частоте 1,5 ГГц.

• SATA 2 – стал вторым вариантом данного стандарта, в этот раз он работает в два раза быстрее своего предшественника, а именно пропускная способность 2,4 Гбит/с, а частота шины 3 ГГц.

• SATA 3 – самый новый стандарт, в теории способен передавать информацию на скорости в 6 Гбит/с, но в практике достигнуто лишь 4,8 Гбит/с. Однако данную спецификацию интерфейса разработчики хотят назвать как – SATA 6Gb/s.

IDE – (Integrated Drive Electronics) – параллельный интерфейс, предназначен для подключения накопителей информации к материнской плате. Интерфейс IDE увидел свет еще в далеком 1986 году, его разработчиком как не удивительно была компания WD (Western Digital). Максимальный поддерживаемый объем диска для IDE было 137 Гб.

ATA – (dvanced Technology Attachment) – практически это тот же IDE, только в более обновленной версии. Имеется 2 версии интерфейса, с 40 проводным шлейфом и с 80 проводным. В первом случае максимальная скорость передачи данных составляет 66 МБ/с, во втором от 66,7 МБ/с.

PCI – (Peripheral component interconnect) – интерфейс, а точнее даже шина для осуществления ввода и вывода между материнской платой и периферийными устройствами. Первая версия PCI появилась в 1992 году, её разработчиком был процессорный гигант – компания Intel. Первая версия работала на частоте 33 МГц, а пропускная способность составляла 80 Мбайт/с. Шина может быть как 32, так и 64 разрядной, например, в компьютерах Macintosh зачастую используется 64 разрядная шина. Максимальная пропускная способность интерфейса PCI на сегодняшний день составляет 4096 Мбайт/с.

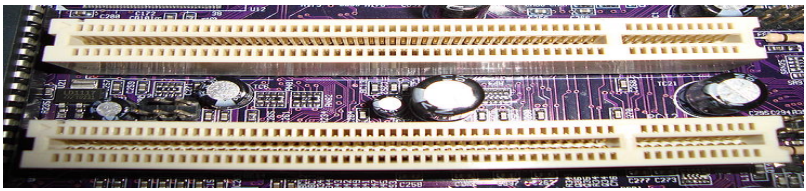


Рис. 15.1. Разъемы 32-разрядные PCI на материнской плате

На одной шине PCI может быть не более четырех устройств (слотов). Мост шины PCI (PCI Bridge) – это

аппаратные средства подключения шины PCI к другим шинам. Host Bridge – главный мост – используется для подключения PCI к системной шине (шине процессора или процессоров). Peer-to-Peer Bridgem – одноранговый мост – используется для соединения двух шин PCI. Две, и более, шины PCI применяются в мощных серверных платформах – дополнительные шины PCI позволяют увеличить количество подключаемых устройств.

Шина PCI все обмены трактует как пакетные: каждый кадр начинается фазой адреса, за которой может следовать одна или несколько фаз данных. Количество фаз данных в пакете неопределенно, но ограничено таймером, определяющим максимальное время, в течении которого устройство может пользоваться шиной. Каждое устройство имеет собственный таймер, значение для которого задается при конфигурировании устройств шины.

В каждом обмене участвуют два устройства - инициатор обмена (Initiator) и целевое устройство (Target). Арбитражем запросов на использование шины занимается специальный функциональный узел, входящий в состав чипсета системной платы. Для согласования быстродействия устройств-участников обмена предусмотрены два сигнала готовности *IRDY#* и *TRDY#.*

Шина имеет версии с питанием 5 В, 3.3 В. Также существует универсальная версия (с переключением линий +V I/O с 5 В на 3.3 В). Ключами являются пропущенные ряды контактов 12, 13 и 50, 51. Для 5 В-слота ключ расположен на месте контактов 50, 51; для 3 В - 12, 13; для универсального - два ключа: 12, 13 и 50, 51. Ключи не позволяют установить карту в слот с неподходящим напряжением питания. 32-битный слот заканчивается контактами A62/B62, 64-битный – A94/B94.

В отличие от адаптеров остальных шин, компоненты карт PCI расположены на левой поверхности плат. По этой причине крайний PCI-слот обычно разделяет использование посадочного места адаптера с соседним ISA-слотом (Shared slot).

Шина PCI являлась до последнего времени второй (после ISA) по популярности применения. В современных системах происходит отказ от шин ISA, и шина PCI выходит на главные позиции. Некоторые фирмы для этой шины выпускают

карты-прототипы, но, конечно же, доукомплектовать их периферийным адаптером или устройством собственной разработки гораздо сложнее, чем карту ISA. Здесь сказываются и более сложные протоколы, и более высокие частоты (8 МГц у шины ISA против 33 или 66 МГц у шины PCI). Также шина PCI обладает плохой помехоустойчивостью, поэтому для построения измерительных систем и промышленных компьютеров используется все еще относительно редко.

На некоторых системных (материнских) платах имеется небольшой разъем, который называется Media Bus. Он расположен позади разъема шины PCI одного из слотов. На этот разъем выводятся сигналы обычной шины ISA, и предназначен он для того, чтобы на графическом адаптере с шиной PCI можно было разместить и недорогой чипсет звуковой карты, предназначенный для шины ISA. Этот разъем, а тем более и такие комбинированные аудио-видео карты, широкого распространения не получили.

AGP – (Accelerated Graphics Port) – интерфейс для подключения видеокарты к материнской плате, см. рис. 15.2. Спецификации AGP появились в 1997 году, тогда Intel выпустил первую версию описания, включающую две скорости: 1x и 2x. Во второй версии (2.0) появился AGP 4x, а в 3.0 – 8x.

Рассмотрим все варианты подробнее:

AGP 1x – это 32-битный канал, работающий на частоте 66 МГц, с пропускной способностью 266 Мбайт/с, что в два раза выше полосы PCI (133 Мбайт/с, 33 МГц и 32-бит).

AGP 2x – 32-битный канал, работающий с удвоенной пропускной способностью 533 Мбайт/с на той же частоте 66 МГц за счет передачи данных по двум фронтам, аналогично DDR памяти (только для направления «к видеокарте»).

AGP 4x – такой же 32-битный канал, работающий на 66 МГц, но в результате дальнейших ухищрений была достигнута учетверенная «эффективная» частота 266 МГц, с максимальной пропускной способностью более 1 ГБ/с.

AGP 8x – дополнительные изменения в этой модификации позволили получить пропускную способность уже до 2.1 ГБ/с.

Рис. 15.2. Интерфейс AGP

PCI Express (**PCIe**) – интерфейс, последователь вышеупомянутого AGP. По принципу работы программной модели PCIe похож на PCI. В данном случае используется последовательная передача данных. Разработкой занималась все та же Intel, в продажу устройства с данным интерфейсом поступили в конце 2002 года. В данном случае количество модификаций больше чем у AGP, существует еще 16x и 32x. Однако в повседневной жизни практически везде используется лишь модификация 16x. Разъемы PCI Express по ширине и форме не сильно отличаются от PCI и располагаются в тех же местах на системной плате, см. рис. 15.2.

PCI Express построен на принципах симплексной технологии, а это означает, что сигналы идут одновременно, в противоположных направлениях и по отдельным парам проводов – итого две пары, называемые линией. Стандарт декларирует пропускную способность симплексной линии на отметке 2,5 Гбит/с в одну сторону или, соответственно, 5 Гбит/с в обе стороны. Однако эти значения масштабируемы.

Рис. 15.3. Разъемы PCIe на материнской плате

Указанные значения пропускной способности являются идеальными – то есть в реальной жизни они, к сожалению, не достигаемы.

При инициализации линии, связывающей две точки (отправителя и адресата), происходит пересылка специальных начальных последовательностей. В этих последовательностях могут быть закодированы параметры соединения. Во время сеанса связи могут возникнуть и такие неприятности, как несоответствие работы тактовых генераторов передающей и приемной стороны. Для того чтобы поправить положение, периодически в потоке данных передаются специальные корректирующие последовательности. Степень необходимости внедрения корректирующих последовательностей определяется исходя из разницы между показателями тактовых генераторов.

В любом случае дополнительные кодирующие последовательности, как и необходимость внедрения избыточности с целью синхронизации, негативно сказываются на производительности, так как неизбежно влекут за собой

временные потери. Для организации связи в реальном времени такой путь не подходит, поэтому разработчикам таких критичных к латентности систем придется изобретать собственные протоколы обмена.

Можно сказать, что на данном этапе последовательная передача, как таковая, заканчивается. Единственное соединение, представляющее собой линию PCI Express, две пары проводов, – этого не достаточно для обеспечения высокой пропускной способности. Поэтому линии привычно выстраивают в ряд - их может быть 32, 16, 12, 8, 4 и 2. В итоге, вся последовательность данных, которую необходимо передать, распределяется на все имеющиеся линии «веером» – передача параллельная, но не синхронная. Если имеется 12 линий, то первый байт блока данных передается по первой линии, второй – по второй, и т. д., а тринадцатый байт – снова по первой. Теоретически шина с 32 линиями способна выдать пропускную способность 20 Гбит/с, от которых отнимаем 20% – 16 Гбит/с, или же по 8 Гбит/с в каждую сторону.

Более высокий уровень стека PCI Express отвечает за корректность передачи данных. Получив от самого верхнего уровня иерархии, Transaction, для передачи пакет данных, алгоритм Data Link присоединяет к последнему номер последовательности и его контрольную сумму. Кроме того, Data Link отвечает и за информирование остальных уровней стека о состоянии канала связи. Третья важная функция Data Link – управление энергопотреблением.

Наиболее интеллектуальный уровень PCI Express – Transaction Layer, который задействует четыре различных адресных пространства. Это память (адрес может быть как 32-, так 64-разрядным), сообщения, конфигурация и ввод/вывод.

Кроме соответствия традиционным требованиям энергосбережения, стандарт PCI Express обладает и эксклюзивными механизмами управления питания – это ASPM (*Active State Power Management*). ASPM обладает достаточно большой автономностью и способен переводить устройство в оптимальный режим работы без инструкций свыше (со стороны ПО). Стандарт PCI Express считает устройство неактивным, если за время, равное 7 мкс, с ним не было никакого обмена

данными. Как только возникает потребность в обмене, устройство возвращается в рабочее состояние.

15.2. Интерфейсы накопителей

Интерфейсы накопителей связывают сам накопитель с контроллером, подключенным к какой-либо системной шине или интерфейсу передачи данных. Для подключения дисковых и других накопителей в настоящее время применяются преимущественно параллельные интерфейсы. Опуская давно вышедшие из употребления интерфейсы ST506/412 (Shugart, Seagate Technology) и ESDI (Enhanced Small Computer Interface), рассмотрим два наиболее распространенных семейства интерфейсов: IDE (ATA-1), ATAPI, EIDE, ATA-3) и SCSI (Fast/Wide, Ultra 2, Fibre Channel и SSA).

ATA-1 IDE (ATA-1 ИДЕ), исторически первый IDE интерфейс для жестких дисков с поддержкой 2-х устройств на шине.

Первой реализацией интерфейса IDE стала версия ATA-1, в которой поддерживалось всего два устройства на шине, причем эти устройства должны были быть жесткими дисками. В соответствии со спецификацией к одному разъему IDE можно подключить два устройства, используя их соединение в виде цепочки (daisy chain) по схеме «управляющий» (master) – «управляемый» (slave). Режим работы устройства (master или slave) задается на этих устройствах при помощи механического переключателя. Таким образом реализована примитивная адресация устройств на шине IDE.

Интерфейс IDE поддерживал режимы PIO Mode 1 и 2, а также режим DMA. Максимальная скорость передачи данных по шине в режиме PIO Mode 2 составляла 8,3 Мбайт/с.

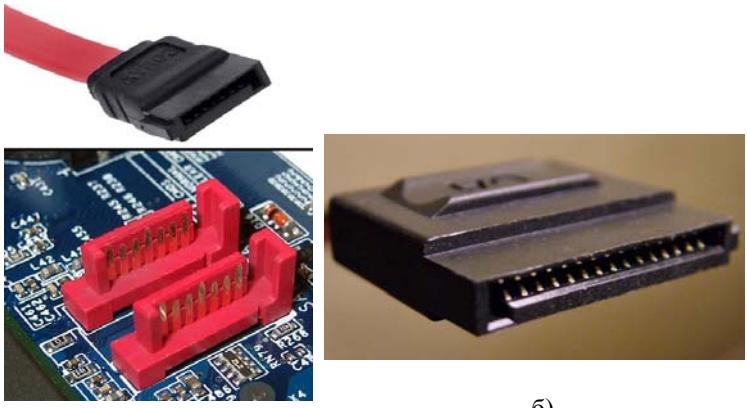
Спецификация интерфейса EIDE, претерпев изменения, направленные на повышение производительности и надежности передачи данных, получила название ATA-3 (или Fast ATA-3). Во-первых, в линию передачи введено терминирование. Во-вторых, обеспечена поддержка технологии предупреждения

отказов жестких дисков SMART (Self-Monitoring Analysis and Reporting Technology). Следующим шагом в развитии интерфейса IDE стала разработанная компанией Quantum спецификация UltraATA, которая на сегодняшний день поддерживается подавляющим большинством новых моделей накопителей и материнских плат. В ней применен новый протокол передачи данных UltraDMA (синонимы: UDMA, UltraDMA/33, UDMA/33). За счет передачи информации по фронту и срезу тактового сигнала скорость передачи в UltraATA составила 33,3 Мбайт/с, что в два раза выше достигаемой в Multi Word DMA Mode 2. Благодаря сохранению прежней тактовой частоты обеспечена обратная совместимость с существующими стандартами IDE. Протокол UltraDMA/33 активно использует Bus Mastering DMA для передачи данных, что снижает загрузку процессора. Для проверки целостности данных используются контрольные коды CRC (Cyclical Redundancy Code). Длина интерфейсного кабеля по-прежнему должна быть менее 45 см (на практике рекомендуется использовать как можно более короткий кабель, но не короче 15 см между отдельными устройствами).

В качестве дальнейшего развития UltraDMA компания Quantum при поддержке Intel и Western Digital предложила спецификацию UltraDMA/66 с пиковой производительностью 66 Мбайт/с. В настоящее время полную поддержку производителей получил новый протокол ATA/100 со скоростью до 100 Мб/с. Большой интерес вызывает появление накопителей с интерфейсом SerialATA. Главные достоинства этого интерфейса – сочетание высоких скоростей передачи данных (уже в первых моделях – до 187,5 Мб/с) с использованием удобных тонких кабелей.

Стандарт SATA/150 обеспечивает пропускную способность до 1,5 Гбит/с (без учета кодирования 8В / 10В). Стандарт SATA/300 обеспечивает пропускную способность до 3 Гбит/с (без учета кодирования 8В/10В). Каждое устройство работает на отдельном кабеле. Стандарт предусматривает горячую замену устройств и функцию очереди команд. SATA-устройства используют два разъема: 7-контактный – для подключения *шины* данных и 15-контактный – для

подключения питания, см. рис. 15.4 а) и б). Передача данных происходит в дуплексном режиме по двум парам проводником (одна пара – на прием, другая – на передачу) с использованием дифференциального кодирования сигналов.



а) б)

Рис. 15.4. 7- и 15 - контактный разъем передачи данных Serial ATA

Кроме перечисленных *интерфейсов*, для подключения накопителей используются универсальные периферийные *интерфейсы*, речь о которых пойдет в следующем разделе.

15.3. Интерфейсы SCSI, RS-232C, IEEE 1284, USB, FireWire

Интерфейс *SCSI* был разработан в конце 1970-х годов и предложен организацией Shugart Associates. Первый стандарт на этот интерфейс был принят в 1986 г.

SCSI определяет только логический и физический уровень. Устройства, подключенные к шине *SCSI*, могут играть две роли: Initiator (ведущий) и Target (ведомый), причем одно и то же устройство может быть как ведущим, так и ведомым. К

шине может быть подключено до восьми устройств. Каждое устройство на магистрали имеет свой адрес (SCSI ID) в диапазоне от 0 до 7. Одно из этих устройств – хост-адаптер *SCSI*. Ему обычно назначают SCSI ID=7. Хост-адаптер предназначен для осуществления обмена с процессором. Хост-адаптер, как правило, имеет разъемы для подключения как встраиваемых, так и внешних SCSI-устройств, см. рис. 15.5.

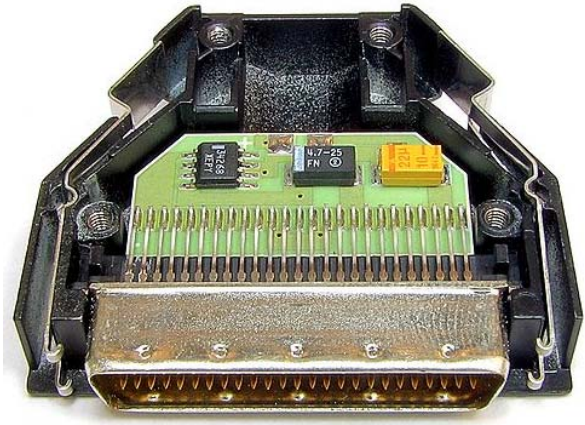


Рис. 15.5. Разъемы SCSI-terminator

Стандарт *SCSI* определяет два способа передачи сигналов – синфазный и дифференциальный. В первом случае сигналы на линиях имеют ТТЛ-уровни, при этом длина кабеля ограничена 6 м. Версии шины *SCSI* с дифференциальной передачей сигнала ("токовой петлей") дают возможность увеличить длину шины до 25 м.

Стандарт SCSI-2 был предложен в 1989 году и существовал в двух вариантах – Fast SCSI и Wide SCSI. Fast SCSI характеризуется удвоенной пропускной способностью (до 10 МБайт/сек). Wide SCSI в дополнение к этому имеет удвоенную разрядность шины (16 бит), что позволяет достичь скорости передачи до 20 МБ/сек. При этом максимальная длина кабеля ограничивалась тремя метрами.

Также в этом стандарте была предусмотрена 32-х битная версия Wide SCSI, которая позволяла использовать два

шестнадцатибитных кабеля на одной шине, но эта версия не получила распространения.

Стандарт SCSI-3 (или Ultra SCSI) появился в 1992 году. Пропускная способность шины составила 20 МБайт/сек для восьмибитной шины и 40 МБайт/сек – для шестнадцатибитной. Максимальная длина кабеля так и осталась равной трём метрам.

В период с 1997 г. По 2010 г. Появились следующие разновидности Ultra SCSI:

Ultra-2 SCSI – 1997 г.

Ultra-3 SCSI – 1999 г.

Ultra-320 SCSI – 2001 г.

Ultra-640 SCSI – 2003 г.

Чтобы гарантировать качество сигналов на магистрали *SCSI*, линии шины должны быть с обеих сторон согласованы при помощи набора согласующих резисторов, или терминаторов. Терминаторы должны быть установлены на хост-адаптере и на последнем устройстве магистрали. Обычно используют один из трех методов согласования:

- пассивное согласование при помощи резисторов;
- FPT (Force Perfect Termination) – улучшенное согласование с исключением перегрузок при помощи ограничительных диодов;
- активное согласование при помощи регуляторов напряжения.

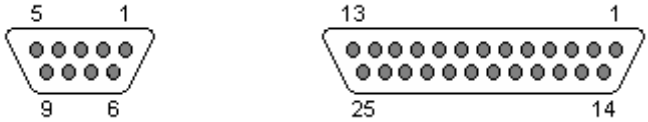
Обмен данными между устройствами на шине *SCSI* происходит в соответствии с протоколом высокого уровня на основе стандартного списка команд – CCS (Common Command Set). Этот универсальный набор команд обеспечивает доступ к данным с помощью адресации логических, а не физических блоков. С внедрением в спецификацию CSS команд, поддерживающих приводы CD-ROM, коммуникационные устройства, сканеры и др. (стандарт SCSI-2), стала осуществимой работа практически с любыми блочными устройствами.

Интерфейс RS-232C

Интерфейс RS-232C предназначен для подключения аппаратуры, передающей или принимающей данные. В роли аппаратуры передачи данных (АПД) может выступать компьютер, принтер, плоттер и другое периферийное оборудование. В роли АКД обычно выступает модем. Конечной целью подключения является соединение двух устройств АПД.

Стандарт на последовательный интерфейс RS-232C был опубликован в 1969 г. Ассоциацией электронной промышленности (EIA). Первоначально этот интерфейс использовался для подключения ЭВМ и терминалов к системе связи через модемы, а также для непосредственного подключения терминалов к машинам. Сейчас интерфейс RS-232C активно вытесняется интерфейсом USB.

Обычно ПК имеют в своем составе два интерфейса *RS-232C*, которые обозначаются COM1 и COM2. Возможна установка дополнительного оборудования, которое обеспечивает функционирование в составе PC четырех, восьми и шестнадцати интерфейсов *RS-232C*. Для подключения устройств используется 9-контактный (DB9) или 25-контактный (DB25) разъем, см. рис. 15.6.



а) б)

Рис. 15.6. 9 (DB9) и 25 (DB25) – контактный разъем для подключения устройств к RS-232C

Основные принципы обмена информацией по интерфейсу RS-232C заключаются в следующем:

Обмен данными обеспечивается по двум цепям, каждая из которых является для одной из сторон передающей, а для другой – приемной.

В исходном состоянии по каждой из этих цепей передается двоичная единица, т.е. стоповая посылка. Передача стоповой посылки может выполняться сколь угодно долго.

Передаче каждого пакета данных предшествует передача стартовой посылки, т.е. передача двоичного нуля в течение времени, равного времени передачи одного бита данных.

После передачи стартовой посылки обеспечивается последовательная передача всех разрядов данных, начиная с младшего разряда. Количество битов может быть 5, 6, 7 или 8.

После передачи последнего бита данных возможна передача контрольного разряда, который дополняет сумму по модулю 2 переданных разрядов до четности или нечетности. В некоторых системах передача контрольного бита не выполняется.

После передачи контрольного разряда или последнего бита, если формирование контрольного разряда не предусмотрено, обеспечивается передача стоповой посылки. Минимальная длительность посылки может быть равной длительности передачи одного, полутора или двух бит данных.

Обмен данными по описанным выше принципам требует предварительного согласования приемника и передатчика по скорости (длительности бита) (300-115200 бит/с), количеству используемых разрядов в символе (5, 6, 7 или 8), правилам формирования контрольного разряда (контроль по четности, по нечетности или отсутствие контрольного разряда), длительности передачи стоповой посылки (1 бит, 1,5 бит или 2 бит).

Спецификация RS-232C для электрических характеристик сигналов определяет, что высокий уровень напряжения от +3В до +12В (при передаче – до +15В) считается логическим "0", а низкий уровень напряжения от 3В до 12В (при передаче – до 15В) считается логическим "1". Диапазон сигналов – 3В:+3В обеспечивает защиту от помех и стабильность передаваемых данных.

Интерфейс IEEE 1284

Стандартный интерфейс параллельного порта получил свое первоначальное название по имени американской фирмы Centronics – производителя принтеров.

Параллельный порт Centronics – порт, используемый с 1981 года в персональных компьютерах фирмы IBM для подключения печатающих устройст. Изначально этот порт был разработан только для симплексной (однонаправленной)

передачи данных, так как предполагалось, что порт должен использоваться только для работы с принтером, см. рис. 15.7.

Рис. 15.7. Кабельный 36-контактный разъём для подключения внешнего устройства (IEEE 1284-B)

Впоследствии разными фирмами были разработаны дуплексные расширения интерфейса (byte mode, EPP, ECP). Затем был принят международный стандарт IEEE 1284, описывающий как базовый интерфейс Centronics, так и все его расширения.

Стандарт IEEE 1284 определяет работу параллельного интерфейса в трех режимах: Standard Parallel Port (*SPP*), Enhanced Parallel Port (*EPP*) и Extended Capabilities Port (*ECP*). Каждый из этих режимов предусматривает двустороннюю передачу данных между компьютером и периферийным устройством.

Режим ***SPP*** (Стандартный параллельный порт) используется для совместимости со старыми принтерами, не поддерживающими IEEE 1284. Режиму *SPP* соответствуют три программно доступных регистра:

- порт BASE+0 – *SPP* Data – регистр данных,
- порт BASE+1 – SPP Status – регистр состояния,
- порт BASE+2 – SPP Control – регистр управления.

Для устройства LPT1 базовым адресом (BASE) в пространстве портов ввода-вывода обычно является 378h.

В настоящее время стандарт IEEE-1284 не развивается и активно вытесняется интерфейсом USB. Также, альтернативой параллельному интерфейсу является сетевой интерфейс Ethernet.

Интерфейс USB

Спецификация периферийной шины USB была разработана лидерами компьютерной и телекоммуникационной промышленности – Compaq, DEC, IBM, Intel, Microsoft, NEC и Northern Telecom для подключения компьютерной периферии вне корпуса ПК с автоматическим автоконфигурированием (Plug&Play). Первая версия стандарта появилась в 1996 г.

Основная цель стандарта, поставленная перед его разработчиками – создать реальную возможность пользователям работать в режиме Plug&Play с периферийными устройствами. Кроме этого, желательно питание маломощных устройств подавать с самой шины. Скорость шины должна быть достаточной для подавляющего большинства периферийных устройств. Попутно решается историческая проблема нехватки ресурсов на внутренних шинах IBM PC совместимого компьютера – контроллер USB занимает только одно прерывание независимо от количества подключенных к шине устройств.

Возможности USB следуют из ее технических характеристик:

- Высокая скорость обмена (full-speed signaling bit rate) – 12 Mb/s;
- Максимальная длина кабеля для высокой скорости обмена – 5 m;
- Низкая скорость обмена (low-speed signaling bit rate) – 1.5 Mb/s;
- Максимальная длина кабеля для низкой скорости обмена – 3 m;
- Максимальное количество подключенных устройств (включая размножители) – 127;

- Возможно подключение устройств с различными скоростями обмела;
- Отсутствие необходимости в установке пользователем дополнительных элементов, таких как терминаторы для SCSI;
- Напряжение питания для периферийных устройств – 5 V;
- Максимальный ток потребления на одно устройство – 500 mA.

Особенно удобен этот интерфейс для подключения часто подключаемых/отключаемых устройств. Конструкция разъемов для USB, см. рис. 15.8, рассчитана на многократное сочленение/расчленение.

USB Тип A
а)

USB Тип B
б)

Рис. 15.8. Разъемы USB

Шина позволяет подключить к ПК до 127 физических устройств. Каждое физическое устройство может, в свою очередь, состоять из нескольких логических.

Интерфейс *USB* 1.1 декларирует два режима:

- низкоскоростной подканал (пропускная способность - 1,5 Мбит/с), предназначенный для таких устройств, как мыши и клавиатуры;
- высокопроизводительный канал, обеспечивающий максимальную пропускную способность 12 Мбит/с, что может использоваться для подключения внешних накопителей или устройств обработки и передачи аудио- и видеоинформации.

Спецификация USB 2.0 выпущена в апреле 2000 года. USB 2.0 отличается от USB 1.1 введением режима Hi-speed.

Для устройств USB 2.0 регламентировано три режима работы:

Low-speed, 10-1500 Кбит/с (используется для интерактивных устройств: клавиатуры, мыши, джойстика);

Full-speed, 0,5-12 Мбит/с (аудио-, видеоустройства);

Hi-speed, 25-480 Мбит/с (видеоустройства, устройства хранения информации).

В спецификации USB 3.0 разъёмы и кабели обновлённого стандарта физически и функционально совместимы с USB 2.0. Кабель USB 2.0 содержит в себе четыре линии (см. рис. 15.8) – пару для приёма/передачи данных, плюс и ноль питания. В дополнение к ним USB 3.0 добавляет еще четыре линии связи (две витых пары), в результате чего кабель стал гораздо толще. Новые контакты в разъемах USB 3.0 расположены отдельно от старых на другом контактном ряду, см. рис. 15.9.

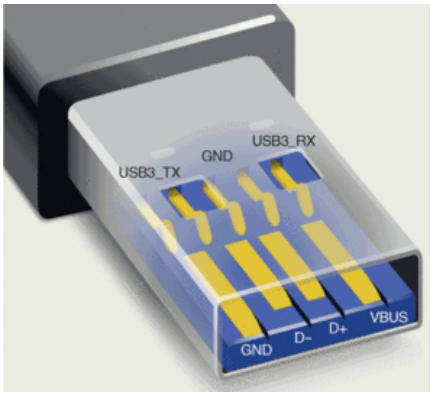


Рис. 15.9. Разъемы USB 3.0

Спецификация USB 3.0 повышает максимальную скорость передачи информации до 4,8 Гбит/с.

Интерфейс IEEE 1394 - FireWire

FireWire – универсальный, удобный в реализации и использовании скоростной интерфейс, обеспечивающий подключение самых разнообразных устройств.

Основные характеристики шины можно свести к следующим показателям:

- скорость передачи данных до 400 Mbits/s по стандарту IEEE-1394a и 800 Mbits/s по стандарту IEEE-1394b, согласованному в 1394 Trade Association в конце мая 2001 года;
- 16-ти разрядный адрес позволяет адресовать до 64K узлов на шине;
- предельная теоретическая длина шины 224 метра;
- "горячее" подключение/отключение без потери даннях;
- автоматическое конфигурирование, аналогичное Plug&Play;
- произвольная топология шины – по аналогии с локальными сетями может использоваться как "звезда" так и общая шина (только в виде цепочки, в отличие от сети на коаксиальном кабеле);
- отсутствие терминаторов;
- возможность обмена с гарантированной пропускной способностью.

Максимальное расстояние между двумя устройствами в цепочке по IEEE-1394a – 4.5 м, по IEEE-1394b - 100 м.

Передача данных осуществляется по тонкому и гибкому кабелю (до 4,5 метров длиной) со скоростью до 400 Мбит/с (то есть 50 МБ/с).

Топология IEEE-1394 позволяет как древовидную, так и цепочечную архитектуру, а также комбинацию из того и другого. Поэтому легко строить любые варианты подключения различных устройств к шине. Стандарт предусматривает архитектурное разделение шины на 2 основных блока – кабельная часть и контроллер (контроллеры). Так как контроллеров может быть несколько, эту часть также называют

объединительной (backplane – дословно задний план, кросс-плата и т.п.).

Адрес узла на "дереве" 16-ти разрядный, что позволяет адресовать до 64К узлов. К каждому узлу может быть подключено до 16-ти конечных устройств. На объединительной панели (backplane) может быть подключено до 63 узлов к одному мосту (bridge) шины. Так как под идентификатор номера шины (моста) отведено 10 разрядов, то общее количество узлов и составляет 64К.

Каждый узел обычно предусматривает подключение 3-х устройств, хотя собственно стандарт разрешает подключение до 27 устройств. Устройства могут быть подключены через стандартные кабели длиной до 4.5 метра.

Физические адреса (ID) устройствам назначаются при подаче питания на контроллер шины и устройства, подключенные к ней, после общего сброса шины, а также при "горячем" подключении устройства к шине. Адреса присваиваются в порядке последовательности обнаружения и/или подключения устройств. Никакая установка перемычек или переключателей на самих устройствах не требуется.

Стандартный кабель для IEEE-1394 состоит из 2 витых пар передачи сигналов шины, двух проводов питания и все это заключено в экранированную оболочку. Провода питания рассчитаны на ток до полутора ампер и напряжение от 8 до 40 вольт. На рис. 15.10 показан один из вариантов кабеля IEEE-1394.

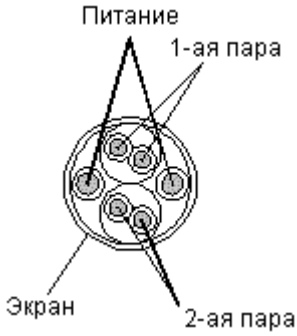


Рис. 15.10 Вариант кабеля IEEE-1394

В отличие от USB, где применение различных типов разъемов регламентировано типом устройств, в FireWire все несколько по-другому. Здесь разъемы подразделяются по тому, нужно ли устройству питание от шины или нет. В том случае, когда нет необходимости в питании, используется 4-х контактный разъем (как правило, такой применяется в видеокамерах). Если же устройству может потребоваться питание от шины, то используется 6-и контактный разъем. Большинство компьютерных устройств рассчитано именно на него.

15.4. Безпроводные интерфейсы

Инфракрасный интерфейс IrDA

Первая версия стандарта IrDA (*Infra-Red Data Assotiation*) была принята в 1994 году Ассоциацией инфракрасной передачи данных. Интерфейс IrDA позволяет соединяться с периферийным оборудованием без кабеля при помощи ИК-излучения с длиной волны 850-900 нм (номинально - 880 нм). Порт IrDA дает возможность устанавливать связь на коротком расстоянии до 1 метра в режиме "точка-точка".

Устройство инфракрасного интерфейса подразделяется на два основных блока: преобразователь (модули приемника-детектора и диода с управляющей электроникой) и кодер-декодер. Блоки обмениваются данными по электрическому интерфейсу, в котором в том же виде транслируются через оптическое соединение, за исключением того, что здесь они пакуются в кадры простого формата - данные передаются 10bit символами, с 8bit данных, одним старт-битом в начале и одним стоп-битом в конце данных.

Сам порт IrDA основан на архитектуре коммуникационного COM-порта ПК, который использует универсальный асинхронный приемо-передатчик UART (*Universal Asynchronous Receiver Transmitter*) и работает со скоростью передачи данных 2400 -115200 bps.

Связь в IrDA полудуплексная, т.к. передаваемый ИК-луч неизбежно засвечивает соседний PIN-диодный усилитель приемника. Воздушный промежуток между устройствами позволяет принять ИК-энергию только от одного источника в данный момент.

Схема обращения устройств представляет собой протокол обмена данными, где есть фазы запросов (*Request*) и ответов (*Response*). Обмен информацией идет только с первичным устройством, которое всегда выступает инициатором соединения, однако его роль может играть любое из устройств, поддерживающих необходимые для этого функции. Каждое устройство имеет 32-битный адрес, вырабатываемый случайным образом при установлении соединения. Каждому кадру в пределах соединения ведущее устройство при старте присваивает 7-битный адрес соединения. Для возможных, но нежелательных случаев, когда два устройства имеют одинаковый адрес, предусмотрен такой механизм, когда ведущее устройство дает команду всем подчиненным устройствам изменить их адреса.

Протокол управления каналом IrLMP (*Link Management Protocol*) является обязательным, однако его некоторые особенности могут быть опциональны. Протокол IrLMP содержит два компонента: LM-IAS (Link Management Information Access Service) и LM-MUX (Link Management MUltipleXer).

Каждое устройство IrDA содержит таблицу сервисов и протоколов, доступных в настоящий момент. Эта информация может запрашиваться у других устройств. LM-IAS управляет информационной базой так, что станции могут запросить, какие службы предоставляются. Эта информация хранится в виде объектов, с каждым из которых связан набор атрибутов.

LM-MUX выполняет мультиплексирование каналов поверх одного соединения, устанавливаемого протоколом IrLAP. С этой целью определяется множество точек доступа канала LSAP (*Link Service Access Point*) каждая с уникальным идентификатором (селектором). Таким образом, каждое из LSAP-соединений определяет логически различные информационные потоки. LM-MUX функционирует в двух

режимах: мультиплексирования и эксклюзивном. Первый режим позволяет разделять одно физическое соединение нескольким задачам. В этом случае управление потоком должно быть обеспечено протоколами верхнего уровня (например, TinyTP) или непосредственно приложением. Второй режим отдает все ресурсы одному единственному приложению. Также IrLMP предусматривает три варианта доступа:

- с установлением предварительного соединения,
- без установления предварительного соединения;
- режим сбора информации о возможностях, сервисах и приложениях удаленного устройства.

В последнее десятилетие ИК порты на ПК уступили место интерфейсам Bluetooth и Wi-Fi.

Однако, вполне возможно, что инфракрасный порт вскоре сможет пережить ренесанс. Японские разработчики в 2008 г. смогли кардинально улучшить параметры интерфейса, в первую очередь, скорость обмена, сохранив одно из его важных достоинств – полную невосприимчивость к электромагнитным помехам. Новая технология отличается использованием полупроводниковых лазеров вместо обычных светодиодов. Это позволило увеличить скорость обмена до 1 Гбит/с, что в 250 раз больше, чем теоретический предел портов IrDA, равный 4 Мбит/с, и более чем в 60 раз превышает возможности VFIR. Помимо повышенной скорости, новинку отличает улучшенная помехоустойчивость. Новый IrDA интерфейс может найти применение в ПК и мобильных устройствах, особенно телефонах.

Интерфейс Bluetooth

В начале 1998 года Ericsson, IBM, Intel, Toshiba и Nokia - крупнейшие компании компьютерного и телекоммуникационного рынка – объединились для совместной разработки технологии беспроводного соединения мобильных устройств. 20 мая 1998 года произошло официальное представление специальной рабочей группы (SIG – *Special Interest Group*), призванной обеспечить беспрепятственное внедрение технологии, получившей название Bluetooth. Сейчас

группа включает в себя более 1400 компаний (в том числе 3COM/Palm, Axis Communication, Motorola, Compaq, Dell, Qualcomm, Lucent Technologies, UK Limited, Xircom и др.), принимающих участие в работе над бесплатной открытой спецификацией Bluetooth.

Устройства, использующие стандарт Bluetooth, функционируют в диапазоне 2,45 ГГц ISM (*Industrial, Scientific, Medical* – промышленный, научный и медицинский диапазон) и способны передавать данные со скоростью до 720 кбит/с на расстояние до 10 метров и передачу 3 голосовых каналов. Такие показатели достигаются при использовании мощности передачи 1 мВт и задействованном механизме переключения частоты, предотвращающем интерференцию. Если принимающее устройство определяет, что расстояние до передающего устройства менее 10 м, оно автоматически изменяет мощность передачи до уровня, необходимого при данном расположении устройств. Устройство переключается в режим экономии энергии в том случае, когда объем передаваемых данных становится мал или передача прекращается.

Технология использует FHSS – скачкообразную перестройку частоты (1600 скачков/с) с расширением спектра. При работе передатчик переходит с одной рабочей частоты на другую по псевдослучайному алгоритму. Для полнодуплексной передачи используется дуплексный режим с временным разделением (TDD). Поддерживается изохронная и асинхронная передача данных и обеспечивается простая интеграция с TCP/IP. Временные интервалы (Time Slots) развертываются для синхронных пакетов, каждый из которых передается на своей частоте радиосигнала.

Энергопотребление устройств Bluetooth должно быть в пределах 0.1 Вт. Каждое устройство имеет уникальный 48-битовый сетевой адрес, совместимый с форматом стандарта локальных сетей IEEE 802.

Диапазон 2.45 гГц является не лицензируемым и может свободно использоваться всеми желающими. Управляет им лишь Федеральная комиссия по коммуникациям (FCC - *Federal Communication Commission*), ограничивая часть диапазона, которую может использовать каждое устройство. Устройства

стандарта Bluetooth способны соединяться друг с другом, формируя пикосети, в каждую из которых может входить до 256 устройств. При этом одно из устройств является ведущим (Master), еще семь – ведомыми (Slave), остальные находятся в дежурном режиме. Пикосети могут перекрываться, а к ресурсам ведомых устройств может быть организован доступ. Перекрывающиеся пикосети могут образовать распределенную сеть, по которой могут мигрировать данные.

В отличие от технологии инфракрасной связи IrDA (Infrared Direct Access), работающей по принципу "точка-точка" в зоне прямой видимости, технология Bluetooth разрабатывалась для работы как по принципу "точка-точка", так и в качестве многоточечного радиоканала, управляемого многоуровневым протоколом, похожим на протокол мобильной связи GSM.

Bluetooth стала конкурентом таким технологиям, как IEEE 802.11, HomeRF и IrDA, хотя последняя и не предназначена для построения локальных сетей, но является самой распространенной технологией беспроводного соединения компьютеров и периферийных устройств.

Спецификация Bluetooth 3.0 была принята 21 апреля 2009 года. Она поддерживает теоретическую скорость передачи данных до 24 Мбит/с. Её основной особенностью является добавление AMP (*Асимметричная Мультипроцессорная Обработка*), дополнение к 802.11 как высокоскоростное сообщение. Модули с поддержкой спецификации Bluetooth 3.0 соединяют в себе две радиосистемы: первая обеспечивает передачу данных в 3 Мбит/с (стандартная для Bluetooth 2.0) и имеет низкое энергопотребление; вторая совместима со стандартом 802.11 и обеспечивает возможность передачи данных со скоростью до 24 Мбит/с (сравнима со скоростью сетей Wi-Fi). Выбор радиосистемы для передачи данных зависит от размера передаваемого файла. Небольшие файлы передаются по медленному каналу, а большие – по высокоскоростному. Bluetooth 3.0 использует более общий стандарт 802.11 (без суффикса), то есть не совместим с такими спецификациями Wi-Fi, как 802.11b/g или 802.11n.

В 2010 г. организация Bluetooth Special Interest Group одобрила новый интерфейс Bluetooth 4.0, и теперь

производители техники могут применять его в своих устройствах.

По сравнению со стандартом Bluetooth 3.0 произошли следующие изменения. Во-первых, увеличена скорость передачи данных - теперь она достигает 24 Мбит/с. Во-вторых, значительно увеличился радиус действия модуля Bluetooth – он превышает 100 метров. Третье – это снижение энергопотребления модуля Bluetooth. Данный параметр важен для портативных устройств.

Вопросы для самоконтроля

1. Что такое интерфейс?
2. Назовите основные функции интерфейсов ЭВМ.
3. Системные интерфейсы ЭВМ и их особенности.
4. Перечислите основные особенности интерфейса AGP.
5. Чем отличаются интерфейсы PCI и PCI Express.
6. Какие шины расширения используются в архитектуре ПК в настоящее время?
7. Назовите особенности стандартного последовательного интерфейса RS-232C.
8. Перечислите особенности интерфейса USB 3.0.
9. Назовите протоколы, входящие в стек IrDA, и их назначение.
10. Охарактеризуйте топологию интерфейсов USB и FireWire.
11. Сравните технические характеристики интерфейсов USB и FireWire.

Литература для самостоятельной подготовки по теме:
5, 8, 16, 20.

ГЛАВА 16. УСТРОЙСТВА ВВОДА-ВЫВОДА

Поскольку устройствам ввода/вывода посвящено большое количество специальной литературы [3, 5, 8, 12, 17, 20] кратко остановимся на некоторых конструктивных аспектах наиболее распространенных из них.

Можно выделить следующие основные функциональные классы устройств ввода-вывода (УВВ):

1. УВВ, предназначенные для связи с пользователем, к ним, прежде всего, относят клавиатуры, сканеры, а также манипуляторы – мыши, трекболы и джойстики, мониторы, принтеры, графопостроители и т.п.;
2. Устройства связи с объектом управления (АЦП, ЦАП, датчики, цифровые регуляторы, реле и т.д.);
3. Средства передачи данных на большие расстояния (средства телекоммуникации – модемы, сетевые адаптеры и др.).

Основным устройством ввода информации в компьютер является **клавиатура**, которая представляет собой совокупность механических датчиков, воспринимающих давление на клавиши и замыкающих тем или иным образом определенную электрическую цепь. В настоящее время распространены два типа клавиатур: с механическими или с мембранными переключателями. В первом случае датчик представляет собой традиционный механизм с контактами из специального сплава. Во втором случае переключатель состоит из двух мембран – верхней (активной) и нижней (пассивной). Мембраны разделены третьей прокладкой (тоже мембрана).

Как правило, внутри корпуса любой клавиатуры, кроме датчиков клавиш, расположены электронные схемы дешифрации и микроконтроллер. Обмен информации между клавиатурой и системной платой осуществляется по специальному последовательному интерфейсу 11-битовыми блоками. Основной принцип работы клавиатуры заключается в

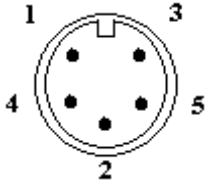
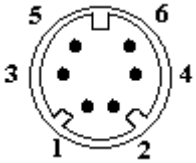
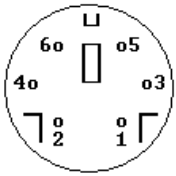
сканировании переключателей клавиш. Замыканию и размыканию любого из этих переключателей соответствует уникальный цифровой код – скан-код. Подключение клавиатуры к системной плате выполняется посредством электрически идентичных разъемов 5 DIN5 или 6 mini-DIN (PS/2), см. табл. 16.1 или шины USB.

Первая **компьютерная мышь** была представлена 9 декабря 1968 года Дугласом Энджельбартом на компьютерной конференции Fall Joint. Распространение мыши получили благодаря росту популярности программных систем с графическим интерфейсом пользователя.

Первая мышь при движении вращала два колеса, которые были связаны с осями переменных резисторов. Перемещение курсора такой мыши вызывалось изменением сопротивления переменных резисторов.

Таблица 16.1

Распайка разъемов AT (5 DIN) и PS/2

<table>
<tr><th colspan="2">Нумерация разъемов</th><th>Назначение разъемов</th></tr>
<tr><td colspan="2"> 
</td><td>5 DIN
1. Тактовая частота
2. Данные
3. N/A (не используется)
4. GND ("земля")
5. Питание +5V</td></tr>
<tr><td>PS/2
 
</td><td>6 mini-DIN
 
</td><td>PS/2 и 6 mini-DIN
1. Данные
2. N/A (не используется)
3. GND ("земля")
4. Питание +5V
5. Тактовая частота
6. N/A (не используется)</td></tr>
</table>

Сегодня наиболее распространенной конструкцией мыши является полностью оптическая конструкция. С помощью светодиода и системы линз, фокусирующих его свет, под мышью подсвечивается участок поверхности. Отраженный от этой поверхности свет, в свою очередь, собирается другой

линзой и попадает на приемный сенсор микросхемы процессора обработки изображений. Этот чип делает снимки поверхности под мышью с высокой частотой и обрабатывает их. Опираясь на анализ череды последовательных снимков, представляющих собой квадратную матрицу из пикселей разной яркости, интегрированный *DSP*-процессор (*Digital signal processor*) высчитывает результирующие показатели, свидетельствующие о направлении перемещения мыши вдоль осей *X* и *Y*, и передает результаты своей работы на периферийный интерфейс. Основные характеристики, обеспечивающие надежность работы оптических мышей, определяются техническими параметрами применяемых сенсоров.

Первые мыши подключались к ПК через специальную плату-адаптер (т.н. мыши с шинным интерфейсом – *bus mouse*). Затем большое распространение получил способ подключения мыши через последовательный интерфейс *RS-232C*. В 1987 году компания IBM выпустила серию персональных компьютеров PS/2, в котором был представлен выделенный последовательный интерфейс для подключения мыши с разъемом 6 mini-DIN, см. табл. 16.1. В 2002 году в спецификации Microsoft PC 2002 было предложено отказаться от этих портов в пользу универсального интерфейса USB.

Трекбол представляет собой «перевернутую» оптико-механическую мышь – в движение приводится не сам корпус устройства, а только его шар. Это позволяет существенно повысить точность управления курсором и, кроме того, экономить место, поэтому трекболы часто используют в ноутбуках.

Сенсорная панель (*touchpad* или *trackpad*) – это устройство ввода, применяемое в ноутбуках, служит для перемещения курсора в зависимости от движений пальца пользователя. Используется в качестве замены компьютерной мыши. Сенсорные панели различаются по размерам, но обычно их площадь не превосходит 50 см^2. Работа сенсорной панели основана на измерении емкости пальца или измерении емкости между сенсорами. Емкостные сенсоры расположены вдоль вертикальной и горизонтальной осей панели, что позволяет

определять положение пальца с нужной точностью. Преимуществами сенсорных панелей являются:

- отсутствует необходимость в ровной поверхности, как для мыши;
- расположение сенсорной панели, как правило, фиксировано относительно клавиатуры;
- для перемещения курсора на весь экран достаточно лишь небольшого перемещения пальца;
- работа с ними не требует особого привыкания, как, например, в случае с трекболом.

Недостатком же сенсорных панелей является низкое разрешение, что затрудняет работу в графических редакторах.

Джойстик является аналоговым координатным устройством ввода информации, выполняемым обычно в виде двух реостатных датчиков с питанием +5В. Рукоятка джойстика связана с двумя переменными резисторами, изменяющими свое сопротивление при ее перемещении. Один резистор определяет перемещение по координате *X*, другой – по *Y*. Джойстик обычно подключается к адаптеру игрового порта, расположенному на многофункциональной плате ввода-вывода (*Multi I/O Card*).

Сканером называется устройство, которое позволяет вводить в компьютер образы изображений, представленных в виде текста, рисунков, слайдов, фотографий или другой графической информации. Сканеры можно классифицировать по следующим критериям:

1. По степени прозрачности вводимого изображения с оригинала:
   - непрозрачные оригиналы (фотографии, рисунки, страницы книг и журналов), при этом изображение снимается в отраженном свете;
   - прозрачные оригиналы (слайды, негативы, пленки), при этом обрабатывается свет, прошедший через оригинал.
2. По кинематическому механизму сканера:
   - ручные сканеры, в которых задача ровного и равномерного перемещения сканирующей головки по соответствующему изображению возлагается на пользователя;

- планшетные сканеры – сканирующая головка перемещается относительно бумаги с помощью шагового двигателя;
- рулонные сканеры – отдельные листы документов протягиваются через устройство так, что сканирующая головка остается на месте;
- проекционные сканеры – вводимый документ кладется на поверхность сканирования изображением вверх, при этом блок сканирования также находится сверху, а перемещается только сканирующее устройство.

3. По типу вводимого изображения:
   - черно-белые (штриховые или полутоновые);
   - цветные.

В черно-белых сканерах изображение освещается белым светом, получаемым, как правило, от флуоресцентной лампы. Отраженный свет через редуцирующую линзу попадает на фоточувствительный элемент (ПЗС-линейка или ПЗС-матрица). Каждая строка сканируемого изображения соответствует определенным значениям напряжения на ПЗС, см. рис. 16.1.

На полупроводниковой поверхности находится тонкий (0,1-0,15 мкм) слой диэлектрика (обычно окисла), на котором располагаются полоски проводящих электродов (из металла или поликристаллического кремния). Эти электроды образуют линейную или матричную регулярную систему, причем расстояния между электродами столь малы, что существенными являются эффекты взаимного влияния соседних электродов. Принцип работы ПЗС основан на возникновении, хранении и направленной передаче зарядовых пакетов в потенциальных ямах, образующихся в приповерхностном слое полупроводника при приложении к электродам внешних электрических напряжений [20].

Эти значения напряжения преобразуются в цифровую форму через АЦП (для полутоновых сканеров) или через компаратор (для двухуровневых «штриховых» сканеров).

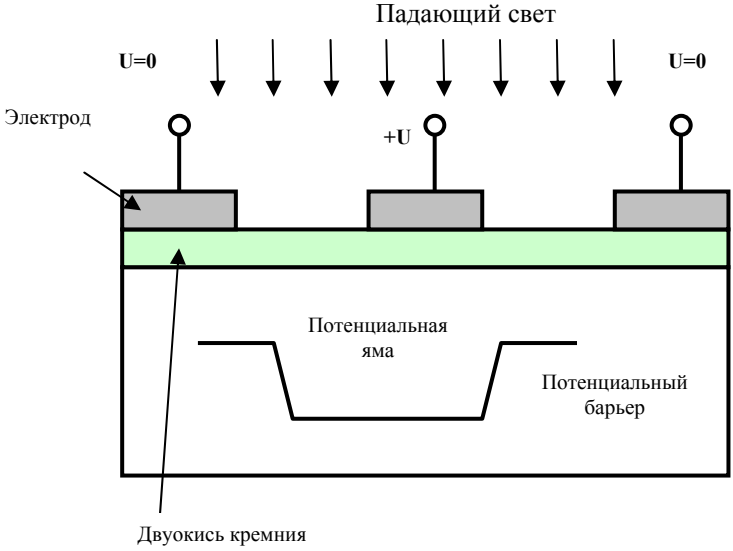


Рис. 16.1. Принципиальное устройство ПЗС матрицы

Для сканирования цветных изображений существует несколько технологий. Например, сканируемое изображение поочередно освещается красным, зеленым и синим цветом, так что страница сканируется за три прохода. В ряде сканеров, например, Epson и Sharp, смена цвета происходит для каждой строки, что позволяет избежать проблем с «выравниванием» пикселей при разных проходах.

В конструкции сканеров Hewlett Packard (HP) и Ricoh сканируемое изображение освещается источником белого света, а отраженный свет через редуцирующую линзу попадает на трехполосную ПЗС-линейку через систему специальных фильтров, разделяющих свет на три компоненты: красный, синий, зеленый.

Для связи с компьютером сканеры, как правило, используют один из универсальных периферийных интерфейсов: SCSI, IEEE 1284 или USB.

Для унификации прикладного ПО сканеров в 1992 г. рядом компаний (Kodak, HP, Logitech и др.) была разработана спецификация TWAIN [20].

Монитор (дисплей) – устройство визуализации текстовой или графической информации без ее долговременной фиксации.

По типу отображаемой информации мониторы делят на алфавитно-цифровые (в настоящее время не используются) и графические. По способу формирования изображения графические дисплеи делят на векторные (не используются в ПК) и растровые. В векторном дисплее изображение строится из элементарных отрезков векторов (в случае ЭЛТ – электронный луч непрерывно "вырисовывает" контур изображения, собирая его из этих векторов). В растровых дисплеях изображение получают с помощью матрицы точек (в случае ЭЛТ – электронные лучи пробегают по строкам экрана, подсвечивая требуемые точки своим цветом).

Наиболее широкое распространение получили мониторы на базе электронно-лучевых трубок (ЭЛТ) и на основе жидких кристаллов (ЖК).

Принцип действия ЭЛТ – мониторов заключается в том, что испускаемый электродом (электронной пушкой) пучок электронов, попадая на экран, покрытый люминофором, вызывает его свечение. На пути пучка электронов находятся дополнительные электроды: отклоняющая система (определяет направление пучка) и модулятор (регулирует яркость получаемого изображения). В случае цветного *монитора* имеются три электронных пушки с отдельными схемами управления, а на поверхность экрана нанесен люминофор трех основных цветов: R (*red*) – красный, G (*green*) – зеленый, B (*blue*) – синий. Чтобы каждая пушка попадала только по люминофору своего цвета, используется теневая маска. Электронный луч периодически сканирует весь экран, образуя близкорасположенные строки развертки. По мере движения луча по строкам видеосигнал, подаваемый на модулятор, изменяет яркость определенных пикселей, образуя видимое изображение. В цикле сканирования луч движется по зигзагообразной траектории от левого верхнего угла экрана к

нижнему правому. Прямой ход луча по горизонтали осуществляется сигналом строчной (горизонтальной) развертки, а по вертикали – сигналом кадровой (вертикальной) развертки.

Очевидно, наиболее важными параметрами для монитора являются: частота кадровой развертки, частота строчной развертки и полоса пропускания видеосигнала. Частота кадровой развертки во многом определяет устойчивость изображения (отсутствие мерцаний). Современные *мониторы* поддерживают кадровые развертки в диапазоне 60-160 Гц. Частота строчной развертки определяется произведением частоты вертикальной развертки на количество выводимых строк в одном кадре с учетом обратного хода (разрешение по вертикали), типичное значение – 30-64 кГц (отражает количество строк, которое монитор может воспроизвести за одну секунду). Полоса видеосигнала определяется произведением разрешения по горизонтали с учетом обратного хода на частоту строчной развертки (отражает число точек в строке, которое монитор может воспроизвести за одну секунду). К важным факторам, определяющим четкость изображения, относят также размеры точек люминофора, а точнее – расстояние между ними (*dot pitch*), типичное значение – 0,25-0,28 мм.

Работа ЖК-мониторов основана на свойстве некоторых веществ проявлять анизотропию в текучем («жидком») состоянии. Первый ЖК-монитор был продемонстрирован американской фирмой RCA в 1966 году. Для изготовления ЖК-мониторов используют так называемые нематические кристаллы - оптически одноосные жидкие кристаллы, имеющие дальний ориентационный порядок и свободные в перемещении. В отсутствии электрического поля молекулы этого вещества образуют скрученные спирали (обычно 90º). В результате такой ориентации молекул плоскость поляризации проходящего света поворачивается. Если же к прозрачным электродам приложено напряжение, спираль молекул распрямляется (они ориентируются вдоль поля), при этом поворота плоскости поляризации проходящего света не происходит. Используя подходящим образом ориентированный пленочный

поляризатор, можно добиться, чтобы в первом случае ЖК-элемент пропускал проходящий свет, а во втором – нет.

При использовании Twisted-Nematic (*TN*)-элементов контраст равен 3:1 (освещенная точка в три раза светлее темной). Молекулы элемента Super-Twisted-Nematic (*STN*) закручены на угол от 180 до 270 градусов. Контраст при использовании STN-элементов составляет 10:1 и выше, но в них проявляется некоторый сдвиг цветов. Для устранения цветовых ошибок в применяемых в настоящее время Triple STN-элементов (*TSTN*), называемых также FSTN – Film STN предусмотрена специальная полимерная пленка между стеклом и поляризатором – третий слой (отсюда Triple). Экран ЖК-дисплея имеет либо заднюю подсветку (*backlight* или *backlit*), либо боковую (*sidelight* или *sidelit*). Каждая точка изображения на ЖК-дисплее – соответствующий TSTN-элемент, а весь экран – матрица этих элементов.

Для адресации ЖК-элементов можно использовать два метода: прямой (пассивный) и косвенный (активный). При прямой адресации элементов каждая выбираемая точка изображения активируется подачей напряжения на соответствующий проводник-электрод для строки (общий для целой строки) и на проводник-электрод для столбца (общий для всего столбца). Матрицы с пассивным управлением («пассивные матрицы») имеют недостаточный контраст изображения, т.к. электрическое поле возникает не только в точке пересечения адресных проводников, но и на всем пути распространения тока. Эта проблема решается при использовании так называемых активных матриц, когда каждой точкой изображения управляет свой независимый электронный переключатель (как правило, TFT).

При применении активных матриц большое значение имеют такие параметры, как малое время отклика (типичное значение – 1-8 мкс) и большой угол зрения (75º-120º).

При подключении мониторов к видеокарте используются в основном два типа разъемов: разъем DB-15 с аналоговым видеосигналом и опционально с цифровым интерфейсом DDC и разъем DVI (*Digital Visual Interface*),

позволяющий передавать как аналоговый видеосигнал, так и цифровой.

Под ***принтером*** обычно подразумевают устройство вывода данных, преобразующее информацию в удобную для чтения форму на бумаге. Как правило, классификацию принтеров проводят по следующим критериям:

1. По способу печати:
   - последовательные – печатный документ формируется символ за символом;
   - строчные – при печати устройство формирует сразу всю строку целиком;
   - страничные – на бумагу наносится изображение сразу всей страницы.
2. По используемой технологии печати:
   - ударные (для переноса красящего вещества используется механический удар);
   - безударные.

В настоящий момент к ударным принтерам относят матричные конструкции. В них печатающая головка из 9, 18 или 24 игл, приводится в движение электромагнитами. Головка крепится к каретке и перемещается вместе с ней по направляющим параллельно бумаге вдоль печатаемой строки. Часть игл матрицы приводится в движение, и они «ударяют» по красящей ленте, находящейся между головкой и бумагой, формируя, таким образом, печатаемый символ. Недостаток этих принтеров – низкая скорость печати и высокий уровень шума. Достоинство – нежесткие требования к качеству бумаги.

К безударным относят струйные чернильные принтеры. У них головка движется в горизонтальной плоскости над бумагой. Печатающая головка содержит сопла, через которые подаются чернила. У разных моделей количество сопел может варьироваться от 12 до 64. Различные технологии струйных принтеров отличаются способом выбрасывания чернильной капельки из сопла. В принтерах Cannon и HP используется технология *bubble-jet* (или *thermal ink jet*). В каждом сопле находится нагревательный элемент (тонкопленочный резистор). При резком нагревании образуется чернильный паровой пузырь, который выталкивает из сопла очередную порцию чернил. В

принтерах Epson используется технология *piezo ink jet*. Выбросом капли из сопла управляет диафрагма из пьезоэлемента. Под действием электрического поля пьезоэлемент деформируется и выталкивает каплю из сопла. Несомненным преимуществом перед матричными принтерами является низкий уровень шума при работе, а недостатком достаточно высокие требования к качеству бумаги. В целом, необходимо отметить, что расходные материалы для данной технологии являются самыми дорогими, по сравнению с принтерами других технологий печати.

Другой популярной безударной технологией является технология электрографической печати, которая используется в так называемых лазерных принтерах. Луч полупроводникового лазера формирует электронное изображение на фотоприемном барабане. Барабану предварительно сообщается некий статический заряд. Таким образом, освещаемые и неосвещаемые лазером участки барабана имеют разный заряд. К заряженным участкам прилипают частицы порошкообразного тонера. При соприкосновении бумаги с барабаном на ней остается отпечаток, который фиксируется за счет нагрева частиц тонера до температуры плавления. Лазерные принтеры имеют высокую скорость печати и высокую разрешающую способность. Недостатком является высокая цена принтеров и необходимость использования качественной бумаги.

Для управления принтером используются специальные языки. Для матричных и струйных принтеров наибольшее распространение получил язык ESC/P. Для лазерных и некоторых струйных принтеров основными языками управления являются PCL фирмы HP и PostScript фирмы Adobe.

PCL (*Printer Command Language*) – язык управления принтером разработанный компанией HP. В настоящий момент PCL используется только в Microsoft и HP.

Postscript был Adobe Systems в начале 80-х гг. В Postscript используется модель изображения текста (или рисунков) на чистой странице. Когда страница готова, она выводится на печать и начинается «прорисовка» изображения очередной страницы. Это есть не что иное, как метод компиляции. Каждый документ Postscript обычно представляет

собой программу, которая печатает на принтере (или отображает на экране монитора) следующие друг за другом страницы.

Для подключения *принтеров* используют RS-232C, IEEE 1284 или USB.

В заключении отметим, что в последнее время разработчики различных компаний все чаще предлагают альтернативные конструкции устройств ввода/вывода. Приведем только один пример. Специалисты Массачусетского технологического института предложили устройство названное Mouseless. Суть подхода заключается в использовании традиционной для владельцев ноутбуков технологии управления, но без участия мыши как таковой. Лазерная система вмонтированная в торец портативного компьютера, следит за перемещение руки, которая будто лежит на классической мышке, и фиксирует перемещения, воспринимая стуки пальцев по столу в качестве кликов. Полученные с камеры сведения обрабатываются специально разработанным ПО, который трансформирует движение руки в движение курсора на экране.

Впрочем, эксперты Intel полагают, что уже в течение следующих десяти лет человек сможет полностью отказаться от использования клавиатуры и мыши, применяя для управления лишь силу мысли. На данный момент ведутся исследования в области регистрации волновой активности с целью определения технологии ее трансформации для сопряжения с управляющими чипами. Теоретически установленные на поверхности головного мозга микросхемы смогут воспринимать мысли, интерпретировать их и передавать в виде команд для компьютерных систем.

Вопросы для самоконтроля

1. Дайте классификацию периферийных устройств.
2. Сформулируйте основные принципы работы устройств ввода.
3. Перечислите классы сканеров.

4. Назовите типы дисплеев, физические принципы формирования изображения.

5. Дайте сравнительную оценку различных технологий вывода на печать.

6. Какие новые направления развития устройств ввода вывода Вы знаете?

Литература для самостоятельной подготовки по теме:
5, 8, 16, 20.

ГЛАВА 17. НОВЫЕ НАПРАВЛЕНИЯ РАЗВИТИЯ КОМПЬЮТЕРОВ

Перспективных направлений связанных с развитием архитектуры ЭВМ в настоящее время достаточно много. Мы остановимся только на наиболее интенсивно развивающихся направлениях исследований – квантовые компьютеры, нанотехнологии, нейрокомпьютеры и фотоника.

Квантовые компьютеры

Идея создания квантового компьютера (КК) принадлежит американскому физику Ричарду Фейнману. В 1958 году, моделируя на компьютере квантовые процессы, он понял, что для решения квантовых задач со многими частными производными объем памяти классического компьютера совершенно недостаточен. Уже при решении задачи с 1000 электронными спинами в памяти должно быть достаточно ячеек, чтобы хранить 2^{1000} переменных. А гигабайт – это всего лишь 2^{30}. Реальный прорыв в становлении идеи КК произошел в 1995 году, когда американский математик Шор переложил для квантового компьютера алгоритм вычисления простых множителей больших чисел. Шор показал, что если классический компьютер для нахождения множителей числа из 1000 двоичных знаков должен сделать 2^{1000} операций, то квантовому компьютеру для этого понадобится всего 1000^3 операций.

Теоретических моделей квантового компьютера множество. Проблема, скорее, в том, чтобы найти разумные пути создания реального прибора. Существует несколько подходов к осуществлению идеи такого устройства. Первый вариант – это импульсный ядерный магнитно-резонансный (ЯМР) спектрометр высокого разрешения. Спины ядер, входящих в состав атомов, в свою очередь образующих исследуемую в ЯМР – спектрометре молекулу – это Q-биты, единицы измерения квантовой информации. Каждое ядро имеет

свою частоту резонанса в данном магнитном поле. При воздействии импульсом на резонансной частоте одного из ядер оно начинает эволюционировать, остальные же ядра «молчат». Для того чтобы заставить эволюционировать второй атом, надо взять другую частоту и дать импульс на ней. Иными словами, процесс вычислений управляется импульсами переменного магнитного поля. Например, 1000^3 (то есть миллиард) операций в алгоритме Шора для 1000-разрядного числа – это миллиард воздействий на отдельные спины и на их пары. При этом в молекуле есть прямая связь между спинами, и поэтому она является идеальной заготовкой для квантового компьютера, а сам спектрометр – просто готовый «процессор» для этого компьютера. Однако в настоящее время удается работать с системами с общим числом спинов не более пяти-семи, в то время как для решения полномасштабных задач их необходимо порядка 1000.

Другой подход основан на использовании т.н. ионных ловушек, или «подвешенных» в вакууме ионов. За изобретение ионных ловушек ученому Боннского университета Паулю в свое время была присуждена нобелевская премия [18,19,20]. Эти ионные ловушки удалось «растянуть» и получить одномерный ионный кристалл, удерживаемый и в осевом, и в радиальном направлении внешними полями. У каждого иона кристалла берутся два уровня энергии – это один Q-бит; между собой эти ионы связаны через колебания внутри одномерного кристалла, который имеет набор резонансных частот. Эксперименты в этом направлении активно проводились в Инсбрукском университете (Австрии) и Лос-Аламосской лаборатории (США) [21, 22]. На сегодняшний день получена цепочка из 30 ионов.

И третий подходм – квантовый компьютер на твердом теле. Например, австралийский физик Кейн высказал предположение, что делать квантовый компьютер будет основан точно на том кремнии, на котором сегодня работает традиционная микроэлектроника. В нужных местах на расстояниях порядка 100 ангстрем располагают атомы фосфора – обычная примесь в кремнии. Если на таком расстоянии расположить два атома фосфора, то облака внешних электронов немного пересекутся, что необходимо для их

взаимодействия, и атомы смогут обмениваться состояниями. Один атом управляет электронами другого. Над этими атомами делаются 50-ангстремные электроды, и с помощью напряжения на этом электроде меняют резонансную частоту спина ядра атома фосфора. Конструкция похожа на полевой транзистор – как бы те же затворы, только вместо тока – состояния атома. У квантового компьютера будет, возможно, и квантовый канал связи, основанный на эффекте, который называется «квантовая телепортация». Принцип квантовой телепортации основан на эффекте запутывания квантовых состояний двух частиц, который анализировался еще в 1935 году Эйнштейном – Подольским – Розеном. Запутанные состояния возникают при взаимодействии двух квантовых частиц и последующем их разъединении; при этом они оказываются в некоем «запутанном» состоянии, в котором состояние первой частицы строго коррелировано с состоянием второй.

Сейчас в исследования квантовых компьютеров вкладываются десятки миллионов долларов. Конечно, это даже нельзя сравнивать с теми деньгами, которые идут на разработку традиционных компьютеров и даже в исследования по нанотехнологиям. Пока над квантовыми вычислениями работают небольшие коллективы в лабораториях таких гигантов, как IBM и Intel. Много экспериментов проводится в крупных центрах, особенно в Лос-Аламосе (США), в университетах по всему миру: в Инсбруке (Австрия), Бонне (Германия).

В 2009 году канадская компания D-Wave продемонстрировала первый работающий квантовый компьютер Orion [21, 22]. Таким образом, реальные квантовые вычисления стали возможными на десятки лет раньше, чем планировалось ранее.

Orion способен выполнять одновременно 64 тыс. операций. Основной частью любого квантового компьютера является квантовый регистр, являющийся совокупностью некоторого числа кубитов – квантовых единиц информации. Кубит, в отличие от обычного бита, который может принимать значение 0 или 1, может одновременно находиться в разных квантовых состояниях, представляющих суперпозицию 0 и 1.

До ввода информации в компьютер все кубиты регистра должны быть приведены в основные базисные состояния (т.н. операция инициализации). Далее каждый кубит подвергается селективному воздействию (импульсами внешнего электромагнитного поля или иным путем), и весь регистр переходит в суперпозицию базисных состояний.

У Orion регистр состоит всего из 16 кубит. Затем информация обрабатывается квантовым процессором, выполняющим последовательность квантовых логических операций. Результатом преобразования информации на выходе компьютера является новая суперпозиция состояний, которую можно далее преобразовать к виду, пригодному для дальнейшего использования.

О процессоре сообщается лишь то, что это новый тип аналогового процессора с масштабируемой архитектурой и, что он основан на квантово-механических принципах.

D-Wave Systems заявила, что квантовый компьютер не будет конкурентом нынешним, скорее, он предназначен для решения задач с огромным количеством исходной информации и большим числом переменных. Такие задачи характерны для систем криптографии и безопасной передачи данных, биологии и медицины, моделирования квантовых систем, оптимизации различных процессов.

Нанотехнологии

Нанотехнологии – это технологии, оперирующие величинами порядка нанометра. Это технологии манипуляции отдельными атомами и молекулами, в результате которых создаются структуры сложных спецификаций. Слово «нано» (в древнегреческом языке «карлик») означает миллиардную часть единицы измерения и является синонимом бесконечно малой величины, в сотни раз меньшей длины волны видимого света и сопоставимой с размерами атомов.

В настоящее время работы в области нанотехнологий ведутся в четырех основных направлениях:

- молекулярная электроника;
- биохимические и органические решения;

- квазимеханические решения на основе нанотрубок;
- квантовые компьютеры.

В 2007 (данные в докризисный период [21, 22]) году на долю США приходилось примерно около 32% всех мировых инвестиций в нанотехнологии (Европейский Союз – примерно 16%, Япония – 20%). Исследования в этой сфере активно ведутся также в странах бывшего СССР, Австралии, Канаде, Китае, Южной Корее, Израиле, Сингапуре и Тайване. Если в 2000 году суммарные затраты стран мира на подобные исследования составили примерно 800 млн. долларов, то в 2010 году они увеличились в 3,5. По мнению экспертов [22], чтобы нанотехнологии стали реальностью, ежегодно необходимо тратить не менее 1 трлн. долларов.

В последнее время резко увеличилось количество публикаций о новых достижениях в области нанотехнологий. Самые свежие новости можно найти, например, на сайте http://www.nanonewsnet.com/.

Наиболее значимые практические результаты достигнуты в области молекулярной электроники. Она логически близка к традиционной полупроводниковой электронике. Методами молекулярной электроники из углеводородных соединений удается получить аналоги диодов и транзисторов, а, следовательно, и основные булевы модули «И», «ИЛИ» и «НЕ», из которых затем можно строить схемы любой сложности. Подобный подход позволяет сохранить преемственность архитектурных решений.

В 1999 году сотрудники компании HP и Калифорнийского университета в Лос-Анджелесе смогли получить действующий молекулярный вентиль [22]. Его толщина составляет всего одну молекулу. Первоначально он умел либо только открываться, либо только закрываться.

В 2001 году в Йельском университете был продемонстрирован вентиль который может принимать любое из двух положений, что позволяет произвольно записывать в него 0 или 1. В настоящее время, судя по публикациям [21, 22], ведутся работы по объединению вентилей в регистры.

По мнению аналитиков, предел миниатюризации для традиционной кремниевой электроники наступит через 7-10 лет.

Используя органическую молекулу и химические внутренние процессы, удалось уменьшить размер транзистора до 1-2 нанометров. При создании подобных транзисторов использовалась техника «самосборки», когда молекулы фактически сами присоединяются одна к другой с помощью электродов, сделанных из золота. Это позволило уменьшить размер канала до 1-2 нм, причем использованная методика относительно недорога и позволяет увеличить плотность транзисторов на единицу площади.

Ученые компании Philips (2002) разработали нанотранзистор, использующий эффект сверхпроводимости. Конструкция транзистора состоит из арсенида индия и алюминиевых сверхпроводящих контактов, а заряд переносится не электронами, а куперовскими парами. Куперовская пара – спаренные электроны с противоположно направленными спинами. В новых элементах ток в канале между стоком и истоком регулируется напряжением на затворе.

Арсенид-индиевые полупроводники размерами от 10 до 100 нм ученые получили с помощью сложного процесса выпаривания.

Свою конструкцию нанотранзисторов предложили ученные из Питтсбургского университета (США). Группа исследователей использовала такие материалы-диэлектрики, как алюминат лантана и титанат стронция. Соединенные вместе, эти материалы обретали способность проводить позитивные электрические заряды.

В Калифорнийском Университете в результате научных изысканий была разработана технология формирования нанопроводников из различных материалов, с четким, на уровне атомов, разграничением слоев. Такое разграничение является критическим условием при создании высокоэффективных транзисторов.

Для формирования полупроводниковых компонентов часто используются гетерогенные структуры – включающие, например, кремний и германий. Но до сих пор не было возможности получения нанопроводников с четкими границами

между слоями этих материалов, диффундировавших друг с другом, тем самым нарушая оптимальные условия для использования их в качестве транзистора. Кроме того, новая технология, в отличие от традиционной, предполагает вертикальное, а не горизонтальное размещение слоев, формирующих транзистор. Эта особенность может помочь в сокращении места, занимаемого каждым логическим вентилем, и обеспечить возможность дальнейшего наращивания количества транзисторов.

Активные исследования в области проектирования нанотранзисторов проводятся учеными в Южной Корее. Шесть атомов углерода – именно столько их задействовано в предложенном ими наноэлементе, обладающем всеми свойствами обычного транзистора. Такой нанотранзистор – это кардинально новое решение, поскольку его использование позволит увеличить производительность в разы, существенно снизив при этом потребление энергии – считают корейские разработчики.

Серийный выпуск нанотранзисторов пока откладывается на несколько лет, поскольку лишь 15% полученных и апробированных сегодня корейскими учеными параметров транзистора, соответствовали требуемым рабочим характеристикам и изобретение нуждается в некоторых уточнениях и доработках. Второе препятствие к вводу в эксплуатацию – отсутствие технологии, позволяющей интегрировать нанотранзисторы в существующие сегодня микросхемы.

Впервые нанотрубки были открыты в лабораториях компании NEC (*Nippon Electric Corporation*) Япония. Нанотрубка представляет собой цилиндрическую структуру толщиной порядка 10 атомов, см. рис. 17.1, которая в зависимости от размера и формы может обладать проводящими либо полупроводниковыми свойствами. Например, если трубка прямая, она является проводником, а если скручена или изогнута – полупроводником. Необычные электрические свойства нанотрубок сделают их одним из основных материалов наноэлектроники.

Компания NEC в 2007 г. сообщила об успешной разработке транзисторов на базе углеродных нанотрубок (*carbon nanotube, CNT*), получаемых при помощи технологий струйной печати. Свойства *CNT*-транзисторов можно задавать, изменяя длину и плотность нанотрубок.

В двух американских университетах в Сан-Диего и Клемсон удалось сделать транзистор полностью из углеродных нанотрубок, разветвленных в форме буквы "*Y*", см. рис. 17.2. Размер нанотранзистора – несколько сотен микрон, что примерно в 100 раз меньше компонентов, используемых в сегодняшних микропроцессорах.

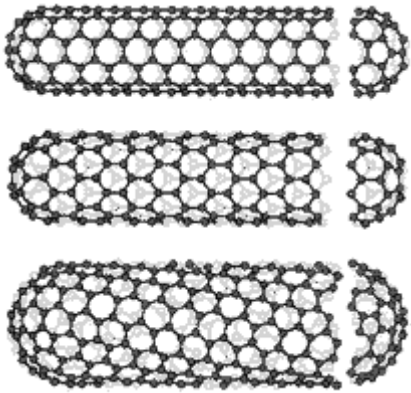


Рис. 17.1. Схема углеродных нанотрубок

Совсем недавно ученым подразделения IBM Research удалось получить углеродную мономолекулярную структуру в виде нанотрубки, которая полностью реализует один из трех основных логических элементов – элемент логического отрицания «NOT». При этом отмечается еще одна особенность созданного элемента: выходной сигнал у него выше, чем входной, приблизительно в полтора раза.

На нанотрубках уже созданы реальные конструкции дисплеев. Например, японская корпорация ISE изготавливает стадионные табло, основанные на панелях с полевой эмиссией.

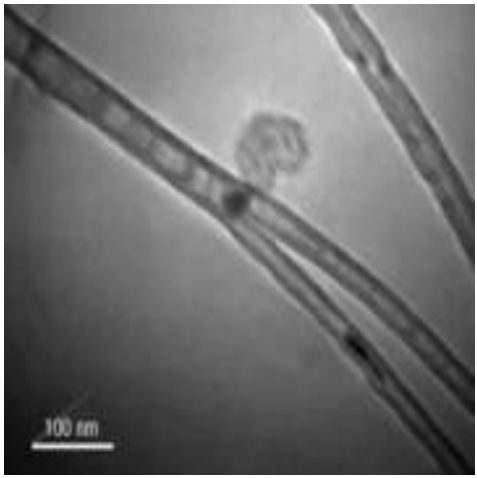


Рис. 17. 2. Нанотранзистор на углеродных нанотрубках (Изображение из журнала New Scientist)

Фирма Motorola продемонстрировала действующий прототип нового цветного дисплея, в котором используется множество нанотрубок. Прототип дисплея имеет размер 4,7 дюйма по диагонали и дает оптическое разрешение в 128x96 пикселей. Он должен стать элементом 42-дюймового телевизионного экрана высокой четкости изображения с разрешением 1280x720 пикселей.

Фотоника

Фотоника – это технология излучения, передачи, регистрации света при помощи волоконной оптики и оптоэлектроники.

Кремниевая фотоника – это соединение оптических технологий с традиционными кремниевыми, широко используемыми для производства разнообразных микросхем. Ключевая особенность такой гибридной технологии заключается в преобразовании электрических сигналов в свет и обратно, то есть превращение электронов в фотоны и наоборот.

Корпорация Intel первой создала прототип гибридной оптической системы передачи данных на основе кремния с пропускной способностью до 50 Гбит в секунду.

Это значительное достижение в области кремниевой фотоники, напрямую связанное с перспективой создания гибридных компьютерных микросхем и целых систем с использованием оптических проводников. В настоящее время чипы и прочие компоненты вычислительной техники и другой электроники связаны друг с другом посредством металлических дорожек на печатных платах или просто с помощью проводов. При этом любой металл, включая медь, обладает некоторым сопротивлением, в результате чего с увеличением протяжённости проводника уровень сигнала падает. Из-за этих потерь конструкторы вынуждены располагать электронные элементы как можно ближе друг к другу. Кроме того, скорости обмена данными по медным проводам достигли физического предела – чрезвычайно сложно добиться передачи сигнала достаточной силы на сколь-нибудь значительные расстояния на скоростях в 10 Гбит/с и выше.

Созданная в лабораториях Intel экспериментальная оптическая система передачи данных на основе кремниевых лазеров позволит заменить традиционные проводники сверхтонкими и лёгкими оптическими волокнами, способными передавать огромные объёмы информации на большие расстояния без существенных потерь.

Прототип системы Silicon Photonics Link с пропускной способностью 50 Гбит/с включает в себя кремниевый передатчик и приёмник. Передатчик состоит из четырёх гибридных кремниевых лазеров и оптических модуляторов, преобразующих данные в световые лучи, способные транслировать данные на скорости до 12,5 Гбит/с каждый. Эти четыре луча соединяются в один при помощи мультиплексора и передаются по единому оптоволоконному кабелю. Затем луч света поступает в приёмник, демультиплексор разделяет его на четыре канала, а фотодетекторы преобразуют световые потоки обратно в электрический сигнал.

Пропускная способность кремниево-оптических сетей позволит передавать трехмерный видеосигнал на экран величиной во во всю стену с таким высоким разрешением, что будет создаваться полное впечатление присутствия в комнате актёров из фильма или собеседников по телеконференции.

Нейрокомпьютеры

Термин «нейрокомпьютер» употребляется для обозначения всего спектра работ в рамках подхода к построению систем искусственного интеллекта, основанного на моделировании элементов, структур, взаимодействий и функций различных нервной системы. Так как в настоящее время исследования в этой области ведутся в основном на уровне моделей нейронных сетей, то понимание термина «нейрокомпьютеры» сужают, ставя знак равенства между ним и нейронными сетями.

В зависимости от способа реализации моделей нейронных сетей выделяют 4 уровня нейрокомпьютеров.

Уровень 0. **Теоретический.** Работы, в которых в той или иной форме (математической, алгоритмической, словесной и т.д.) представлено описание моделей нейронных сетей.

Уровень 1. **Программный.** Модели нейронных сетей, программно реализованные на обычных последовательных компьютерах.

Уровень 2. **Программно-аппаратный.** Сопроцессоры для ускорения моделирования нейронных сетей.

Уровень 3. **Аппаратный.** Физически реализованные модели нейронных сетей.

Специфичность нейросетевых операций, а также сверхпараллельность структуры и функционирования моделей нейронных сетей чрезвычайно замедляют их реализацию на обычных последовательных компьютерах. Потребность в выполнении большого объема исследовательских работ и быстром функционировании появившихся прикладных систем привели к появлению специализированных вычислительных устройств для эффективного моделирования нейронных сетей – нейрокомпьютеров в узком смысле слова. Такая трактовка, соответствующая уровням 2 и 3 по приведенной классификации, получила широкое распространение.

Элементарным строительным элементом нейронной сети (НС) является нейрон, который осуществляет взвешенное

суммирование поступающих на его вход сигналов $x_1, x_2, ..., x_n$, см. рис. 17.3. Результат такого суммирования образует промежуточный выходной сигнал, который преобразуется активационной функцией в выходной сигнал нейрона – *У*. По аналогии с электронными системами активационную функцию можно считать нелинейной усилительной характеристикой искусственного нейрона, имеющей большой коэффициент усиления для слабых сигналов (с падающим усилением для больших возбуждений). Коэффициент усиления вычисляется как отношение выходного сигнала нейрона к вызвавшему его небольшому приращению взвешенной суммы входных сигналов. Кроме этого для обеспечения увеличения вычислительной мощности многослойными НС, по сравнению с однослойными, необходимо чтобы активационная функция между слоями была нелинейной, т.е. как показано в учитывая ассоциативность операции умножения матриц любую многослойную нейросеть без нелинейных активационных функций можно свести к эквивалентной однослойной нейросети, которые весьма ограничены по своим вычислительным возможностям. Но вместе с этим наличие нейлинейностей на выходе нейрона не может служить определяющим критерием, хорошо известны и успешно работают нейросети и без нелинейных преобразований на выход, получившие название нейросети на линиях задержки.

Множество выходных сигналов нейронов сети $y_1, y_2, ..., y_N$ называют вектором выходной активности, или паттерном активности нейронной сети. Веса связей нейронов сети удобно представлять в виде матрицы *W*, где ω_{ij} - вес связи между *i*- и *j*-м нейронами. В процессе функционирования (эволюции состояния) сети осуществляется преобразование входного вектора в выходной, т.е. некоторая переработка информации, которую можно интерпретировать, например, как функцию гетеро – или автоассоциативной памяти. Конкретный вид выполняемого сетью преобразования информации обусловливается не только характеристиками нейроподобных элементов, но и особенностями ее архитектуры, т.е. той или иной топологией межнейронных связей, выбором

определенных подмножеств нейроподобных элементов для ввода и вывода информации или отсутствием конкуренции, направлением и способами управления и синхронизации информационных потоков между нейронами и т.д.

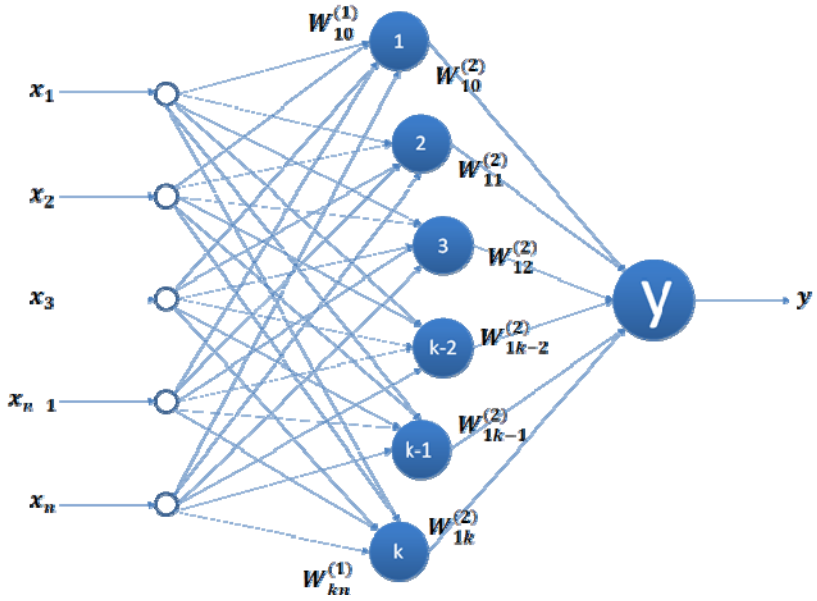


Рис. 17.3. Общий вид нейрона

Среди основных преимуществ НС в [22, 23] отмечены: инвариантность методов синтеза НС к размерности пространства признаков и размерам НС, адекватность современным перспективным технологиям, отказоустойчивость в смысле монотонного, а не катастрофического изменения качества решения задачи в зависимости от числа вышедших из строя элементов.

Как было отмечено, нейрокомпьютер – это вычислительная система с MSIMD архитектурой, т.е. с параллельными потоками одинаковых команд и множественным потоком данных.

Вопросы построения и применения НС выходят за рамки данного учебного пособия, заинтересованному читателю мы можем порекомендовать обратиться к литературе [21, 22, 23].

Вопросы для самоконтроля

1. Сформулируйте основные тенденции развития технологий используемых в микропроцессорах.
2. Какие идеи были положены в основу квантовых компьютеров?
3. Что такое нанотехнологии?
4. В каких направлениях происходит развитие нанотехнологий?
5. Приведите примеры использования нанотехнологий в конструкции узлов ЭВМ.
6. Что такое фотоника? Расскажите о применении фотоники в области телекоммуникаций.
7. Дайте определение нейрокомпьютера.

Литература для самостоятельной подготовки по теме:
5, 8, 16, 20, 21, 22.

ГЛАВА 18. КОРОТКО ОБ АССЕМБЛЕРЕ

Сегодня на рынке программного обеспечения существует огромное количество продуктов написанных на языках высокого уровня. На этом фоне программирование на низкоуровневом языке – ассемблере – может показаться чем-то устаревшим и нерациональным. Однако следует отметить, что ассемблер фактически является языком процессора, а значит, без него не обойтись, пока существуют процессоры. Основными достоинствами программирования на ассемблере являются максимальное быстродействие и минимальный размер получаемых программ. Кроме того, ассемблер позволяет программисту выполнять действия, которые либо вообще нельзя реализовать на других языках, и, в частности, на языках высокого уровня, или выполнение которых займет слишком много машинного времени в случае привлечения дорогостоящих средств языка высокого уровня.

Недостатки зачастую обусловлены склонностью современного рынка ПО к предпочтению количества качеству. По мере увеличения своего размера программа на ассемблере теряет наглядность. Это связано с тем, что в ассемблерных программах следует уделять много внимания деталям. Язык требует планирования каждого шага ЭВМ. В случае небольших программ это позволяет сделать их оптимальными с точки зрения эффективности использования аппаратных средств. Однако, в случае создания сложных приложений огромное количество деталей может помешать, добиться оптимальности программы в целом. Для программирования на ассемблере необходимо очень хорошо знать структуру компьютера и работу аппаратных устройств. Из всего вышесказанного можно сделать вывод, что на языке ассемблер можно сделать любое приложение, любую программу, но для написания сложных приложений лучше использовать языки высокого уровня, такие как C++, C# или Паскаль, которые позволят сосредоточиться на

самом задании, и не нужно будет учитывать и оценивать особенности устройства и микропроцессора.

Поскольку в рамках данного учебного пособия, мы не преследуем цель детально рассмотреть все особенности программирования на ассемблере (более детально этот вопрос рассматривается в книгах В.И. Юрова, А.С. Крупника, В.Ю.Пирогова и др.), в данном разделе приведены основы работы в ассемблере с несложными примерами, которые легко воспроизвести имея под рукой любой компьютер.

18.1. Структура программы на ассемблере

Программа на ассемблере представляет собой совокупность блоков памяти, называемых сегментами памяти. Программа может состоять из одного или нескольких таких блоков-сегментов. Каждый сегмент содержит совокупность предложений языка, каждое из которых занимает отдельную строку кода программы.

Предложения ассемблера бывают четырех типов:

1. команды или инструкции, представляющие собой символические аналоги машинных команд. В процессе трансляции инструкции ассемблера преобразуются в соответствующие команды системы команд микропроцессора;
2. макрокоманды – оформляемые определенным образом предложения текста программы, замещаемые во время трансляции другими предложениями;
3. директивы, являющиеся указанием транслятору ассемблера на выполнение некоторых действий. У директив нет аналогов в машинном представлении;
4. строки комментариев, содержащие любые символы. Ком-ментарии игнорируются транслятором.

Допустимыми символами при написании текста программ являются:

1. все латинские буквы: **A-Z**, **a-z**. При этом заглавные и строчные буквы считаются эквивалентными;
2. цифры от **0** до **9**;
3. знаки **?**, **@**, **$**, **_**, **&**;
4. разделители **, . [ ] ( ) < > { } + / * % ! ' " ? \ = # ^.**

Предложения ассемблера формируются из *лексем*, представляющих собой синтаксически неразделимые последовательности допустимых символов языка, имеющие смысл для транслятора.

Лексемами являются:

- *идентификаторы* – последовательности допустимых символов, использующиеся для обозначения таких объектов программы, как коды операций, имена переменных и названия меток. Идентификатор может состоять из одного или нескольких символов. В качестве символов можно использовать буквы латинского алфавита, цифры и некоторые специальные знаки – _, ?, $, @. Идентификатор не может начинаться символом цифры. Длина идентификатора может быть до 255 символов, хотя транслятор воспринимает лишь первые 32, а остальные игнорирует. Регулировать длину возможных идентификаторов можно с использованием опции командной строки **mv**. Кроме этого существует возможность указать транслятору на то, чтобы он различал прописные и строчные буквы либо игнорировал их различие (что и делается по умолчанию). Для этого применяются опции командной строки **/mu, /ml, /mx**;
- *цепочки символов* – последовательности символов, заключенные в одинарные или двойные кавычки;

целые числа в одной из следующих систем счисления: *двоичной, десятичной, шестнадцатеричной*.

Практически каждое предложение содержит описание объекта, над которым или при помощи которого выполняется некоторое действие. Эти объекты называются *операндами*. *Операнды* – это объекты (например, регистры или ячейки памяти), на которые действуют инструкции или директивы, либо это объекты, которые определяют или уточняют действие инструкций или директив.

Операнды могут комбинироваться с арифметическими, логическими, побитовыми и атрибутивными операторами для расчета некоторого значения или определения ячейки памяти, на которую будет воздействовать данная команда или директива.

Можно провести следующую классификацию операндов:

- постоянные, или непосредственные, операнды;
- адресные операнды;
- перемещаемые операнды;
- счетчик адреса;
- регистровый операнд;
- базовый и индексный операнды;
- структурные операнды.

Под каждую архитектуру процессора и под каждую ОС или семейство ОС существует свой Ассемблер. Существуют также так называемые «кросс-ассемблеры», позволяющие на машинах с одной архитектурой (или в среде одной ОС) ассемблировать программы для другой целевой архитектуры или другой ОС, и получать исполняемый код в формате, пригодном к исполнению на целевой архитектуре или в среде целевой ОС.

Наиболее известными ассемблерами для операционной системы DOS являлись Borland Turbo Assembler (TASM) и Microsoft Macro Assembler (MASM). Также в своё время был популярен простой ассемблер A86.

Изначально они поддерживали лишь 16-битные команды (до появления процессора Intel 80386). Более поздние версии TASM и MASM поддерживают и 32-битные команды, а также все команды, введённые в более современных процессорах, и системы команд, специфических для конкретной архитектуры (такие как, например, MMX, SSE, 3DNow! и т. д.).

Удобство программирования, скромные системные требования и высокая скорость трансляции обеспечивали TASM'у лидерство на протяжении всего существования MS-DOS. В настоящее время Borland (Codegear) прекратила распространение своего ассемблера.

При появлении операционной системы Microsoft Windows появилось расширение TASM, именуемое TASM32, позволившее создавать программы для выполнения в среде Windows.

Microsoft поддерживает свой продукт под названием Microsoft Macro Assembler. Кроме того, Стивен Хатчессон создал пакет для программирования на MASM под названием «MASM32».

MASM'у (MASM32) посвящено множество книг, что упрощает процесс обучения, а в сети можно найти множество исходных текстов ассемблерных программ и библиотек, освобождающих программиста от необходимости изобретать велосипед. Также MASM является выходным языков для многих дизассемблеров (Sourcer, IDA Pro). Все это делает MASM транслятором номером один в программировании под Windows/Intel.

Flat assembler (FASM) – ассемблер с предельно упрощенным синтаксисом. FASM - свободно распространяемый многопроходной ассемблер, написанный Томашем Грыштаром. Помимо базового набора инструкций процессора и сопроцессора FASM поддерживает наборы инструкций MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, SSE4a, AVX и 3DNow!, а также EM64T и AMD64 (включая AMD SVM и Intel SMX). Компиляция программы в fasm состоит из 2 стадий: препроцессирование и ассемблирование.

На стадии препроцессора раскрываются все макросы, символические константы, обрабатываются директивы препроцессора.

В отличие от стадии ассемблирования, препроцессирование выполняется только 1 раз.

Единственное существенное отличие от формата, принятого в других ассемблерах (MASM, TASM в режиме совместимости с MASM) – значение ячейки памяти всегда записывается как [label_name], а просто label_name означает адрес (то есть порядковый номер) ячейки. Это позволяет обходиться без ключевого слова offset. Также в fasm при переопределении размера операнда вместо byte ptr пишется просто byte, вместо word ptr – word и т. д. Не позволяется использовать несколько квадратных скобок в одном операнде, таким образом вместо [*bx*][*si*] необходимо писать [*bx+si*]. Эти изменения синтаксиса привели к более унифицированному и лёгкому для чтения коду.

Благодаря модульной архитектуре, приспособить fasm к генерации кода под другие платформы очень легко. В последнии версии введена поддержка AMD64/EM64T, fasm уже многократно протестирован на создание правильных объектных файлов под Win64, так что он один из немногих открытых средств разработки для платформы Win64.

18.2. Примеры программирования на ассемблере

18.2.1. Встроенный ассемблер

В данном разделе мы рассмотрим примеры использования ассемблерных вставок в программы, написанные на языках высокого уровня.

Встроенный ассемблер позволяет включать в ваши программы на языке C# или C++ операторы ассемблера. Инструкции встроенного ассемблера компилируются и ассемблируются с вашей программой, и вам не потребуется писать отдельные модули.

Чтобы включить в код C++ инструкции ассемблера, используйте ключевое слово asm и следующий формат:

asm код_операции операнды;

где "код_операции" – допустимая инструкция процессора 80x86;

"операнды" содержат операнды (операнд), допустимые для указанной операции (константы, переменные и метки). Концом оператора asm является символ; или новая строка.

После точки с запятой на той же строке может размещаться новый оператор asm, но на следующей строке оператор продолжаться не может. Для включения нескольких операторов *asm* их можно заключить в фигурные скобки (первая скобка должна быть на той же строке, что и asm):

```
asm {
pop ax; pop ds
iret
}
```

Ассемблерная часть оператора копируется непосредственно в вывод и включаются в операторы языка ассемблера, которые Borland C++ или VC++ (VC#) генерирует для инструкций C++ (C#). Все идентификаторы C++ (C#) заменяются на соответствующие эквиваленты ассемблера. Каждый оператор asm рассматривается как оператор C++ (C#).

Пример 1. Операция сложения.

Команды сложения достаточно простые. Команда *INC* осуществляет инкремент, то есть увеличение содержания операнда на единицу, например *INC EAX* или *INC DWORD PTR [EBX]*. Команда *INC* устанавливает флаги**, SF, ZF, AF, PF** в зависимости от результата сложения.

Команда *ADD* позволяет сложить два операнда. Результат помещается в первый операнд (приемник). Например, при выполнении команды *ADD EBX,10* к содержимому регистра *EBX* добавляется число 10. после выполнения команды результат будет находиться в регистре *EBX* . Первый операнд может быть регистром или переменной. Второй операнд (источник) – регистром, переменной или числом. Невозможно выполнить операцию сложения одновременно над двумя переменными.

Команда *XADD* похожа на команду *ADD*, но с некоторым отличием: она предварительно переносит операнд – приемник в операнд – источник, а затем результат сложения помещает в операнд-приемник. Первый операнд может быть регистром и переменной, второй – только регистром. Действие на флаги аналогично команде *ADD.*

Команда *ADC* осуществляет сложение двух операндов подобно команде *ADD*. С помощью этой команды можно осуществлять сложение чисел, когда результат превышает 32 бита или изначально длина операндов превышает 32 бита. Допустим, что первое число содержится в паре регистров *EDX:EAX*, а второе число – в паре *ECX:EBX*. Тогда сложение этих чисел можно представить следующим образом:

ADD EAX, EBX
ADC EDX, ECX

Результат сложения будет содержаться в паре регистров *EDX:EAX*.

Для первого примера мы детально рассмотрим ход его выполнения в **C++Builder 6(2009)** и **Visual Studio 2008**.

В качестве компилятора используем Visual Studio 2008. В окне New Project выбираем *Win32 Console Application* (см. рис. 18.1). Введем имя проекта, например *Ki*. Нажмите кнопку OK. В окне Win32 Application Wizard нажмите кнопку Next (см. рис. 18.2).

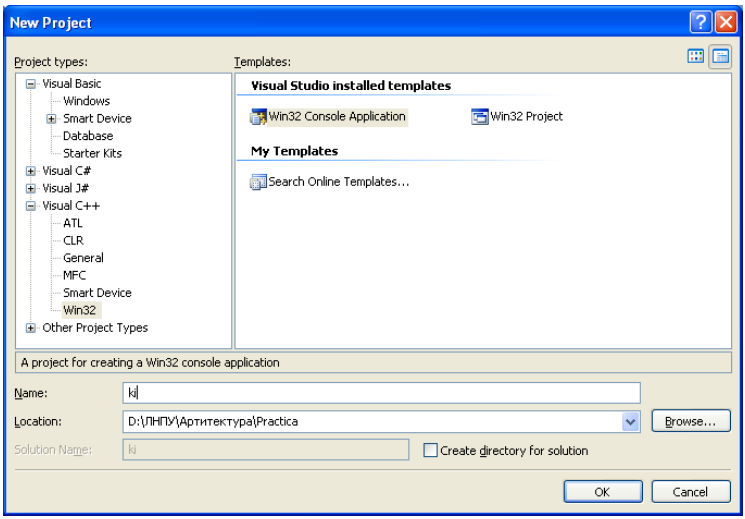


Рис. 18.1. Окно New Project

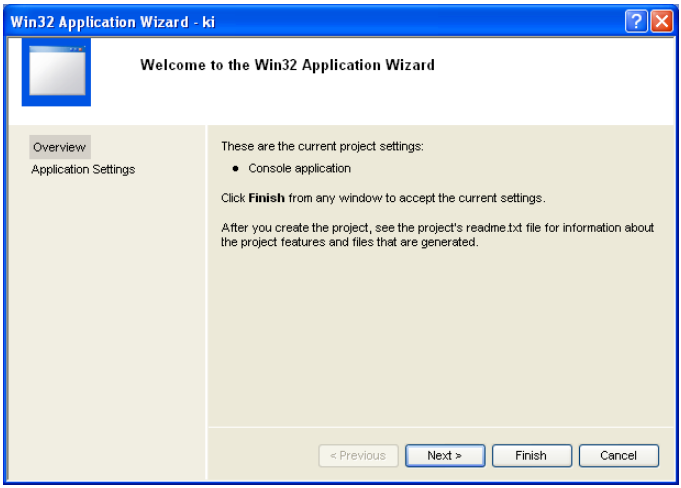


Рис. 18.2. Окно Win32 Application Wizard

В окне кодов (см. рис. 18.3) введем следующие коды примера.

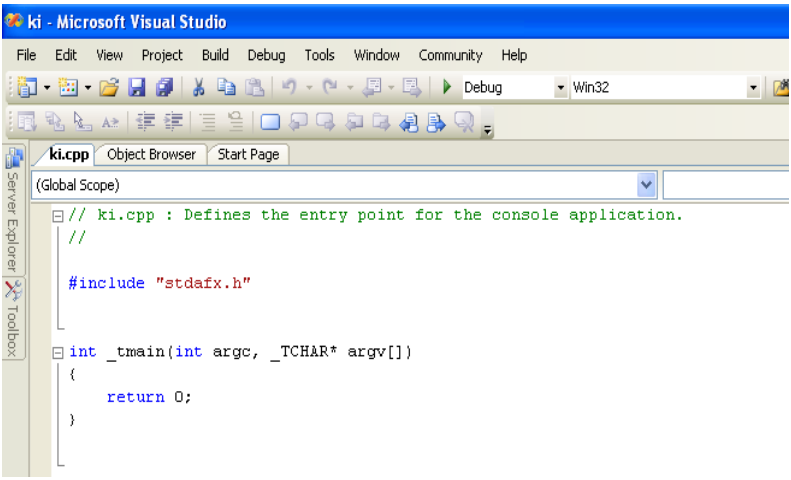


Рис. 18.3. Окно кодов

На рис. 18.4 показан результат виполнения программы.

```
#include "stdafx.h"
#include <windows.h>
```

```
#include <conio.h>
#include <stdio.h>
int a,b,c;
DWORD d,e,f;
void main ()
{
        a=100; b=-200;
        f=0;
        d=0xffffffff;
        e=0x10;
        __asm{
//Сложение положительного и отрицательного чисел
                MOV EAX, a
                ADD EAX, b
                MOV c, EAX
//Сложение больших  чисел
                MOV EAX, e
                ADD d, EAX
                ADC f,0
        }
        printf("%d %x %x", c,f,d);
        getch ();
        }
```

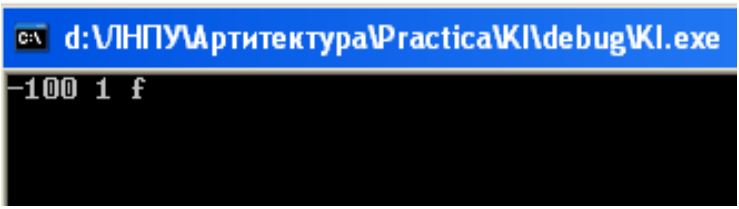


Рис. 18.4. Результат выполнения программы

В начале ассемблерного кода выполняется сложение двух чисел со знаком. При этом *b<0 (b=-200).* Результат помещаем в переменную *c* и выводим как знаковое число – результат – 100**.**

Значения переменных *d* и *e* не превышают 32-битовой границы, но результат в ней не помещается. Следовательно, мы вынуждены учитывать перенос бита за 32-битную границу. Для этого используем команду *ADC f,0.* Второй операнд равен *0*, и значение переменной *f* также сначала равно *0*. Выведем отдельно старшую и младшую части 64-битного числа. В результате на экране видим ***1 f.***

Пример 2. Операция вычитания.

Команда декремент *DEC* уменьшает содержимое операнда (регистра или переменной) на 1. Она так же воздействует, как и инкремент на флаги, **SF, ZF, AF, PF.**

Команда *SUB* вычитает из левого операнда правый и воздействует на флаги **CF, SF, ZF, AF, PF**. Левый операнд может быть регистром или переменной, правый – регистром, или числовой константой. Для команды безразлично, являются ли числа знаковыми или нет.

Команда *SBB* аналогичная *SUB*, но дополнительно вычитает из приемника флаг **CF**. Ее используют для работы с числами, длина которых превышает 32 бита.

Пусть первое число содержится в паре регистров *EDX*:*EAX*, а второе в паре *ECX:EBX.* Тогда вычитание этих чисел можно описать парой команд:

```
SUB EAX,EBX
SBB EDX,ECX
```

Результат вычитания будет содержаться в паре *EDX:EAX.*

Листинг операции вычитания (C++Builder 2009)

```
#include <vcl.h>
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
#pragma argsused
```

```
int a,b,c;
__int64 i,j,k;
void main ()
{
        a=100; b=-200;
        i=0x1fffffff;
        j=0x1ffffffb;
        _asm {
//вычитание  32 – битных  чисел
MOV  EAX, a
SUB EAX, b
MOV c, EAX
// вычитание  64 – битных  чисел
MOV EAX, DWORD PTR i
MOV EDX, DWORD PTR i+4
MOV EBX, DWORD PTR j
MOV ECX, DWORD PTR j+4
SUB EAX,EBX
SBB EDX,ECX
MOV  DWORD PTR k,EAX
MOV  DWORD PTR k+4,EDX
                }
printf("%d %I64x",c,k);
 getch();
 }
```

На рис. 18.5 показан результат виполнения программы.

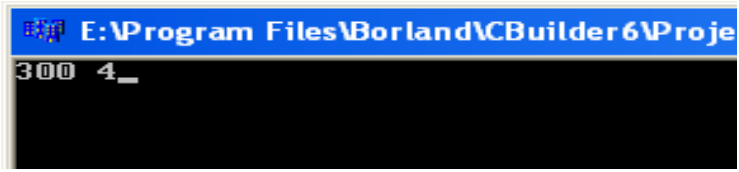


Рис. 18.5. Результат выполнения программы

Выполняя вычитание для 32-битных чисел, мы действуем аналогично операциям сложения.

Для 64-битных чисел, используем переменные, которые имеют тип __int64. Они занимают 8 байт, и компилятор поддерживает действия с ними.

Для операций на ассемблере необходимо использовать пары регистров *EDX:EAX* и *ECX:EBX*. Таким образом, для того, чтобы получить результат вычитания, мы используем пару команд – *SUB/SBB*. Результат помещается в переменную, имеющую тип __int64. Для загрузки 64-битового числа необходимо использовать две команды *MOV*, загружая сначала младшую, а затем старшую часть числа.

Пример 3. Операции умножения и деления.

В отличие от операций сложения и вычитания умножение чувствительно к знаку числа. Существуют две команды умножения:

MUL – для умножения чисел без знака;

IMUL – для умножения чисел со знаком.

Разберем их подробнее.

Команда *MUL*. Единственным операндом этой команды может быть регистр или переменная. Здесь важен размер этого операнда (источника). Если операнд однобайтовый, то он умножается на *AL*, соответственно результат будет помещен в регистр *AX*, в зависимости от того, превосходит он один байт или нет. Если результат не превышает один байт, то флаги **OF** и **CF** будут равны нулю, в противном случае 1. Если операнд двухбайтовый, то он умножается на *AX*, а результат будет помещен в пару регистров *DX:AX*. Соответственно, если результат поместится полностью в *AX*, то есть содержимое *DX* будет равно 0, то нулю будут равны и флаги **CF** и **OF**. Если операнд - источник будет иметь длину 4-е байта, то он умножается на EAX, а результат должен быть помещен в пару регистров *EDX:EAX*. Если содержимое *EDX* после выполнения *умножения* окажется равным нулю, то нулевое значение будет и у флагов **OF** и **CF**.

Команда *IMUL* имеет три различных формата.

Первый формат полностью аналогичный формату команды *MUL*. Второй формат:

IMUL operand1, operand2

operand1 должен быть регистром, *operand2* может быть числом, регистром или переменной. В результате операции умножения *operand1*operand2* результат размещается в *operand1*. Однако мы можем получить число, которое не поместится в приемнике. В данном случае флаги **CF** и **OF** будут равны 1 (0 в противном случае).

Третий формат:

IMUL operand1,operand2, operand3

В данном случае *operand* (регистр или переменная) умножаются на *operand3,* и результат размещается в *operand1*. Если при выполнении умножения возникает переполнение, то устанавливаются флаги **CF** и **OF**.

Листинг операции умножения (C++Builder 2009)

```
#include <vcl.h>
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
#pragma argsused
DWORD a;
__int64 b;
int c,e;
void main( )
{
        a=100000;
        c= -1000;
        __asm {

//умножение чисел без знака
        MOV EAX, 100000
        MUL DWORD PTR a;
```

```
        MOV DWORD PTR b,EAX
        MOV DWORD PTR b+4, EDX
//умножение чисел со знаком
        IMUL EAX, c, 1000
        MOV e, EAX
        }
        printf("%I64d  %d" ,b,e);
        getch();
}
```

На рис. 18.6 показан результат виполнения программы.

Рис. 18.6. Результат выполнения программы

При умножении чисел без знака мы используем два 32-битных числа. Результат операции должен разместиться в паре регистров *EDX: EAX*. Затем мы размещаем значения из этой пары регистров в 64-битной переменной.

При выполнении умножения чисел со знаком мы используем формат команды с тремя операндами. Содержимое переменной *c* умножается на *1000*. Результат размещается в регистре *EAX*.

Деление беззнаковых чисел осуществляется с помощью команды *DIV*. Команда имеет всего один операнд – делитель. Делитель может быть регистром или ячейкой памяти. В зависимости от размера делителя выбирается и делимое.

Возможны следующие варианты:

1. Делитель имеет размер 1 байт. В этом случае делимое помещается в регистр *AX*. Результат деления (частное) окажется в регистре *AL*, в регистре же *AH* будет остаток от деления.

2. Делитель имеет размер 2 байта. Результат операции будет помещен в регистр *AX*. Остаток от деления – в регистр *DX*.

4. Делитель имеет размер 4 байта. Делимое будет находиться в паре регистров *EDX: EAX*. Результат операции будет помещен в регистр *EAX*. Остаток от деления – в регистр *EDX*.

Команда знакового деления *IDIV* полностью аналогичная команде *DIV*,. Существенно, что для команд деления значения флагов арифметических операций не определены. В результате деления может возникнуть либо переполнение, либо деление на 0. В этом случае возникает исключение – будет вызвана специальная процедура, которая и должна обработать возникшую ситуацию. Такую обработку должна обеспечить операционная система.

Листинг операции деления (C++Builder 2009)

```
#include <vcl.h>
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
#pragma argsused
DWORD a,b,c;
void main( )
{
        a=100000;
        __asm{
//беззнаковое деление
        MOV EAX, a
        MOV EDX, 0
        MOV EBX, 20
        DIV EBX
        MOV b, EAX;  //часттное
        MOV c, EDX;  //остаток
        }
        printf("% d % d ",b,c);
        getch();
}
```

На рис. 18.7 показан результат виполнения программы.

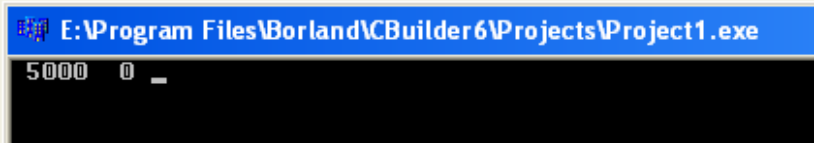


Рис. 18.7. Результат выполнения программы

Пример 3. Способы адресации памяти (Visual Studio 2008).

```
#include "stdafx.h"
#include <windows.h>
#include <conio.h>
#include <stdio.h>
BYTE ar[6] = {1,12,128,50,200,10};
BYTE a,b,c,d;
WORD e;
DWORD f;
void main()
{
        __asm
        {
        MOV AL,BYTE PTR ar
        MOV a,al
        LEA EBX,ar
        MOV AL,BYTE PTR[ EBX]
        MOV b,AL
        MOV AL,BYTE PTR [EBX]+3
        MOV c,AL
        MOV EDX,1
        MOV AL,BYTE PTR [EBX][EDX]+2
        MOV d,AL
        MOV AX,WORD PTR [EBX]+2
        MOV e,AX
        MOV EAX,DWORD PTR [EBX]+2
        MOV f,EAX
```

```
        }
        printf("%u %u %u %u %u %u",a,b,c,d,e,f);
        getch ();
}
```

На рис. 18.8 показан результат виполнения программы.

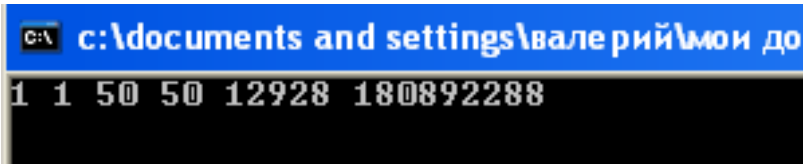


Рис. 18.8. Результат выполнения программы

Данный пример построен, в основном, на непрямой адресации. В непрямой адресации используется один из регистров общего названия. В данном случае применяется регистр *EBX*, но может быть использован любой другой регистр. Формат команды таков, что можно менять адрес ячейки памяти, не меняя содержимого самого регистра: *[EBX]+2*. В данном случае результирующий адрес получается путем сложения адреса, хранящегося в регистре *EBX*, и числа 2. Для изменения адреса обращения можно использовать и другой *регистр: [EBX] [EDX]+2*. В этом случае результирующий адрес получается путем сложения содержимого регистра *EBX*, регистра *EDX*, а также числа *2*. Выражение *[EBX] [EDX]+2* можно заменить и на [EBX+EDX+2].

Следует отметить следующее:

- то адресное пространство и память, которую мы охватываем, является собственностью только нашей программы. Здесь нет места для других программ. Но и мы не можем выйти за рамки отведенной нам области, и обратится к коду другой программы или области памяти, занятой ядром операционной системы;

реальный адрес формулируется не только из 32-битной величины, которую мы используем, но и с помощью сегментных регистров.

18.2.2. Примеры в MASM и FASM

Пример 1. Сравнение MASM и FASM

Этот простой пример позволяет наглядно увидеть некоторые различия при использовании двух самых популярных ассемблеров для ОС Windows.

MASM32

```
.386
.model flat, stdcall
      option casemap :none   ; case
sensitive
     include
\masm32\include\windows.inc
     include \masm32\include\user32.inc
     include
\masm32\include\kernel32.inc
     includelib \masm32\lib\user32.lib
     includelib \masm32\lib\kernel32.lib
   .code
.data
MsgBoxCaption  db "Win32
Программа",0
MsgBoxText     db "Привет
пользователь пк!",0
 .code
start:
    invoke MessageBox, NULL, ADDR
MsgBoxText,\
ADDR MsgBoxCaption, MB_OK

    invoke ExitProcess, NULL
end start
```

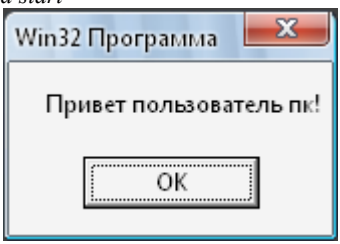


Рис. 18.9. Результат выполнения программы MASM32

FASM

```
format PE GUI 4.0

include
'c:\fasmw\INCLUDE\WIN32AX.INC'


.data
MsgBoxCaption  db "Win32
Программа",0
MsgBoxText     db "Привет
пользователь пк!",0
 .code
start:
      invoke  MessageBox, NULL,
MsgBoxText , MsgBoxCaption ,MB_OK
      invoke  ExitProcess, NULL
.end start
```

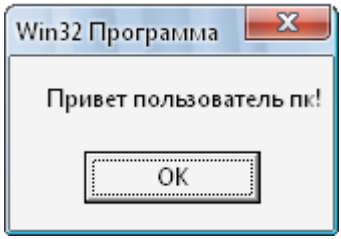


Рис. 18.10. Результат выполнения программы FASM

Разберем листинги.

MASM32

.386 указывает, что используются команды процессора не выше 386.

include user32.inc, *include kernel32.inc*, *include WINDOWS.INC* – перечень подключаемых модулей, которые находятся в библиотеках *user32.dll*, *kernel32.dll*.

Секция *.data* содержит данные (которые в случае необходимости могут быть извлечены).

MsgBoxCaption это метка (смещение). Число, 0 обозначает конец строки.

MsgBoxText аналогично *MsgBoxCaption*.

Секция код (*.code*) – набор команд процесора, выстроенных определенным образом, для того чтобы получилось, что нибудь полезное.

Start: это метка (смещение) которая сообщает компилятору где начинать выполнение программы.

Invoke команда, придуманная Стивеном Хатчессон (Австралия) для того чтобы упростить написание под Windows. У *ExitProcess* один аргумент *NULL* или просто 0, с MessageBox сложнее – 4 аргумента.

ADDR MsgBoxText – передача адреса метки MsgBoxText. Команда ADDR действительна только в контексте директивы *INVOKE*. Вы не можете использовать ее, чтобы присвоить адрес метки регистру или переменной. В данном примере, вы можете использовать *OFFSET* вместо *ADDR*.

MB_OK тип окна с одной кнопкой **ОК.**

FASM

format PE GUI 4.0 – строка описывает формат выходящего файла. Писать не обязательно если вы делаете стандартный windows exe файл, если надо dll, или не стандартный exe файл то надо указать дополнительные параметры например: *format PE GUI 4.0 dll*

include '...:\fasm\INCLUDE\WIN32AX.INC' – библиотека.

В FASM *ADDR* писать не обязательно.

Пример. 2. Арифметические в FASM

```
format PE GUI 4.0
entry start                    ;Точка входа в программу
INCLUDE 'E:\FASM\INCLUDE\win32ax.inc'
INCLUDE 'E:\FASM\INCLUDE\encoding\win1251.inc'
INCLUDE 'E:\FASM\INCLUDE\api\user32.inc'
section '.data' data readable writable
formats db "       %d ",0
result db 256 dup(?)           ;Когда будем преобразовавать
число строку,
                               ;сюда сохраним результат
section '.code' code readable executable
start:
                               ;Сложения чисел 2 и 2
mov eax,2                      ;Перемещаем в eax число 2
mov edx,2                      ;Перемещаем в edx число 2
add eax,edx                    ;Складываем содержимое eax и
edx (2+2).
                               ;Выводим результат в eax.
invoke wsprintf,result,formats,eax          ;Преобразуем
число(результат) в
                               ;строку для того чтобы можно
                               ;было вывести его на экран.
invoke MessageBox,0,result,"Сложение",MB_OK      ;Выводим
результат
                               ;на экран.
                               ;Вычитание 5 из 10
mov eax,10                     ;Перемещаем в eax число 10
mov edx,5                      ;Перемещаем в edx число 5
sub eax,edx                    ;Вычитаем из содержимого eax
edx (10-5).
                               ;Выводим результат в eax.
invoke wsprintf,result,formats,eax      ;Преобразуем
                               число(результат)
                               ;в строку для того чтобы
                               ;можно было вывести его на
                               экран.
```

```
                                        ;Выводим результат на экран.

invoke MessageBox,0,result,"Вычитание",MB_OK

                                        ;Умножение 2 на 3
mov ax,2                                ;Перемещаем в ах число 3
imul ax,3                       ;Умножаем содержимое ах на 3 (2*3).
                                ;Результат будет находиться в еах.
invoke wsprintf,result,formats,eax      ;Преобразуем
                                        число(результат)
                                ;в строку для того чтобы можно
                                ;было вывести его на экран.

                                ;Выводим результат на экран.

invoke MessageBox,0,result,"Умножение",MB_OK
invoke ExitProcess,0                    ;Выход из программы
section '.idata' import data readable
library kernel32,'KERNEL32.DLL',user32,'USER32.DLL'
INCLUDE 'E:\FASM\INCLUDE\api\kernel32.inc'
```

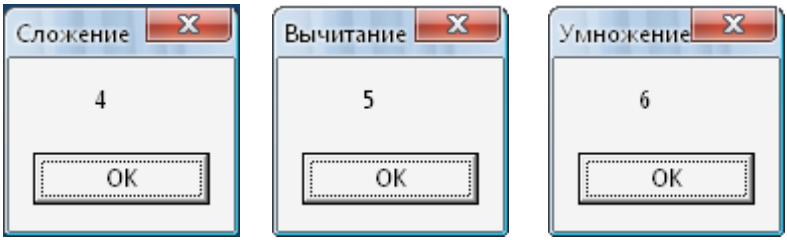


Рис. 18.11. Результат выполнения программы

Директива *INCLUDE* как бы вставляет в указанное место текст из другого файла. Откройте файл *WIN32AX.INC* в папке **INCLUDE** при помощи блокнота или в самом FASM и убедитесь, что мы автоматически подключили к нашей программе еще и текст из *win32a.inc, macro/if.inc*, и общий набор библиотек функций Windows. При помощи подключаемых файлов мы организуем некое подобие языка

высокого уровня: дабы избежать рутины описания каждой функции вручную, мы подключаем целые библиотеки описания стандартных функций Windows.

Далее у нас обозначена секция данных – *.data*. В этой секции мы объявляем переменные. Команда *db* означает "определить байт" (*define byte*). Следующая секция – исполняемый код программы – *.code*. В начале секции стоит метка *start:*. Она означает, что именно с этого места начнет исполняться наша программа. Команда (макроинструкция) *invoke* вызывает встроенную в Windows API-функцию MessageBox.

Вопросы для самоконтроля

1. Перечислите основные средства программирования на ассемблере.
2. Какие операторы ассемблера используются для выполения арифметических операций?

Литература для самостоятельной подготовки по теме:
5, 11, 13.

ЛИТЕРАТУРА

1. Апокин И.А. Развитие вычислительных машин / Апокин И.А., Майстров Л.Е. - М.: Наука, 1990. – 262 с.
2. Бройдо В. Л. Архитектура ЭВМ и систем / Бройдо В. Л., Ильина О.П. : Учебник для вузов. 2-е издание. - СПб.: Питер, 2008. – 720 с.
3. Гилмор Ч. Введение в микропроцессорную технику / Гилмор Ч. - М.: Мир. 1984. – 340 с.
4. Дроздов Е.А. Основы построения и функционирования вычислительных систем / Дроздов Е.А., Пятибратов А.П. - М.:Энергия, 1973. – 368 с.
5. Жмакин А. П. Архитектура ЭВМ / Жмакин А. П. - БХВ-Петербург, 2006. – 320 с.
6. Каган Б.М. Электронные вычислительные машины и системы / Каган Б.М. - М.: Энергоатомиздат. 2001 – 212 с.
7. Карцев М.А. Арифметика цифровых машин/ Карцев М.А. - М.:Наука, 1969. – 134 с.
8. Кузин А.В. Архитектура ЭВМ и вычислительных систем / Кузин А.В., Пескова С.А. - Инфра-М, 2010. – 240 с.
9. Лахно В.А. Прикладная теория цифровых автоматов / Лахно В.А., Могильный Г.А., Петров А.С.- Луганск: ВНУ, 2009. – 248 с.
10. Малиновский Б.Н. История вычислительной техники в лицах / Малиновский Б.Н. – К.: фирма "КИТ", ПТОО "А.С.К.", 1995. – 384 с.
11. Савельев А.Я. Арифметические и логические основы цифровых автоматов / Савельев А.Я. - М.: Высшая школа. 1980. – 312 с.
12. Смирнов А.Д. Архитектура Вычислительных машин / Смирнов А.Д. - М.:Наука, 1990. – 320 с.
13. Таненбаум Э. Архитектура компьютера / Таненбаум Э. - СПб.: Питер, 2002. – 704 с.

14. Точи Р. Цифровые системы. Теория и практика, 8-е издание / Точи Р., Уидмер Дж., Нил С. - М.:Вильямс, 2004. – 1024 с.
15. Чу Я. Организация ЭВМ и микропрограммирование / Чу Я. - М.: Мир. 1975 – 234 с.
16. Evans, James S. and Richard H. Eckhouse, Alpha RISC Architecture for Programmers . Upper Saddle River, N.J.: Prentice Hall PTR, 1999.
17. Gerrit A. Blaauw, Frederick P. Brooks, Jr. Computer Architecture: Concepts and Evolution Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. 1997.
18. Обучение в интернет. Бесплатное дистанционное обучение информатике, телекоммуникациям, основам электронного бизнеса / [электронный ресурс]. – 2011. – http://www.lessons-tva.info
19. Computer Architecture / [электронный ресурс]. – 2011. http://www.psut.edu.jo/sites/qaralleh/CA /ca_doc/Computer %20Architecture_review.pdf
20. Квантовые компьютеры / [электронный ресурс]. – 2011. http://www.wikipedia.org
21. Перспективы развития компьютерной техники / [электронный ресурс]. – 2011. – http://pcterra.org/pers.html
22. Нейронные сети / [электронный ресурс]. – 2011. – http://www.neuroproject.ru/neuro.php
23. Нейронные сети / [электронный ресурс]. – 2011. – http://www.statsoft.ru /home /textbook/modules /stneunet.html

Учебное пособие

АРХИТЕКТУРА КОМПЬЮТЕРА
(Часть 2)

Авторы

Лахно Валерий Анатольевич, доцент, кандидат технических наук
Могильный Геннадий Анатольевич, доцент, кандидат технических наук

Оригинал макет: Лахно В.А.

Видавництво Державного закладу
«Луганський національний університет
імені Тараса Шевченка»
вул. Оборонна, 2, м. Луганськ, 91011. Тел./факс: (0642) 58-03-20