Министерство образования и науки Российской Федерации

Государственное бюджетное образовательное учреждение
высшего профессионального образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

О.В. ВЕСЕЛОВ, А.В. БАКУТОВ

МАЛЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Учебное пособие

Допущено Учебно-методическим объединением по образованию в области автоматизированного машиностроения в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки «Автоматизация технологических процессов и производств»

Владимир 2012

УДК 004.2
ББК 32.973
В38

Рецензенты:
Кандидат технических наук, доцент зав. кафедрой электротехники, электроники и автоматики Московского государственного технологического университета «Станкин»
В. В. Филатов

Кандидат технических наук, зав. сектором конструкторского отдела Государственного научно-производственного предприятия «Крона» (г. Владимир)
Ю. В. Черкасов

Печатается по решению редакционно-издательского совета ВлГУ

Веселов, О. В.

В38 Малые вычислительные системы : учеб. пособие / О. В. Веселов, А. В. Бакутов ; Владим. гос. ун-т имени Александра Григорьевича и Николая Григорьевича Столетовых. – Владимир : Изд-во ВлГУ, 2012. – 360 с.

ISBN 978-5-9984-0281-4

Разработано в соответствии с Государственным образовательным стандартом Министерства образования Российской Федерации по специальностям 220301 – автоматизация технологических процессов и производств, 220401 – мехатроника, направлениям 220700 – автоматизация технологических процессов и производств, 221000 – мехатроника и робототехника. Изложены вопросы архитектуры малых вычислительных систем, принципов их работы и построения.

Предназначено для студентов специальностей 220301, 220700 дневной формы обучения, а также ориентировано на студентов заочной формы обучения и студентов Центра реабилитации инвалидов.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

Табл. 28. Ил. 173. Библиогр.: 17 назв.

УДК 004.2
ББК 32.973

ISBN 978-5-9984-0281-4

ПРЕДИСЛОВИЕ

Понятие «компьютер» в настоящее время стало расхожим. Кто только не поминает его по поводу и без. Говоря о процессорах, памяти, принтерах, сканерах, многие даже не имеют представление о том, как они устроены, как работают, по какому принципу строится память, для каких задач используются и многое другое. Это одна категория людей. Есть и другая категория, для которой эти понятия являются принципиальными. В их задачу входит проектирование систем, используемых для управления в различных отраслях.

Но не смотря на обилие литературы, связанной с компьютером как с таковым и микропроцессорами и микроконтроллерами, все же остается ниша, относящаяся к средствам вычислительной техники и именуемая малыми вычислительными системами. Такими системами, которые уже не компьютер, но еще не микроконтроллер.

Учебное пособие рассчитано на студентов, аспирантов, преподавателей, изучающих устройство, применение и обслуживание высокопроизводительных вычислительных комплексов и сетей, отличается от других учебных пособий новым подходом к изложению материала, учитывающим особенности проектирования и эксплуатации малых вычислительных систем, новейшие разработки в области коммуникаций и распределённых вычислений.

В пособии последовательно рассматриваются не только организация, устройство и принципы функционирования вычислительных устройств, но и такие темы, как их эффективность и качество.

В состав пособия входят не только теоретическое описание малых вычислительных систем, но и методические указания по курсовому проектированию и выполнению лабораторных и практических работ.

Логически учебное пособие можно разделить на несколько частей. В первой части рассматриваются общие характеристики и основные понятия малых вычислительных систем, приводятся конфигурации локальных вычислительных систем и их качественные характеристики. Во второй части рассматривается физическая реализация среды передачи данных, устройство и классификация коммуникационного оборудования, представлены основные понятия архитектуры «клиент-сервер». В третьей части подробно изложены согласование и конфигурация линий связи, структуры распространенных интерфей-

сов и протоколов, принципы организации протоколов, рекомендации по управлению исполнительными устройствами и чтению информации с внешних устройств.

В методических указаниях для курсового проектирования изложены вопросы построения малых вычислительных систем, объединённых в единую сеть для управления технологическими процессами, рассматривается вопрос создания алгоритма и протокола работы сети, излагаются общие требования к проекту, производится описание основных этапов выполнения курсового проекта, приведены варианты заданий и правила оформления пояснительной записки.

В методических указаниях для лабораторных работ изучаются устройство и настройка персонального компьютера, типы сетевых топологий, рассматриваются вопросы монтажа, настройки и исследования производительности вычислительных сетей, даются указания для создания собственного интернет-ресурса.

Как видно из перечисления тем разделов, в учебном пособии наиболее подробно рассмотрены основы функционирования и проектирования современных вычислительных сетей на основе малых вычислительных систем.

Учебное пособие по дисциплине "Вычислительные машины, системы и сети" весьма полезно для студентов, аспирантов и преподавателей технических вузов и позволяет получить не только теоретические, но и практические знания в актуальной и востребованной области – малых вычислительных систем.

Раздел 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Глава 1

СИСТЕМЫ СЧИСЛЕНИЯ И АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Изучение систем счисления и арифметических операций, используемых в ЭВМ, очень важно для понимания того, как производится обработка числовых данных в вычислительных машинах. Знание систем счисления, как и математики вообще, может пригодиться и в повседневной жизни.

ПОЗИЦИОННЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

Во всех вычислительных машинах используются *позиционные системы счисления.* В таких системах число представляется в виде последовательности цифр, позиции каждой из которых присвоен определенный вес. Например, величину D 4-разрядного десятичного числа $d_3d_2d_1d_0$ можно получить следующим образом:

$$D = d_3 \cdot 10^3 + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 \cdot 10^0 .$$

Каждая цифра d_i имеет вес, равный 10^i. Поэтому, например, величина числа 6851 может быть представлена так:

$$6851 = 6 \cdot 1000 + 8 \cdot 100 + 5 \cdot 10 + 1 \cdot 1 .$$

Для того чтобы иметь возможность оперировать как положительными, так и отрицательными степенями числа 10, в десятичной системе счисления используют десятичную точку. Таким образом, число $d_1d_0.d_{-1}d_{-2}$ имеет значение

$$D = d_1 \cdot 10^1 + d_0 \cdot 10^0 + d_{-1} \cdot 10^{-1} + d_{-2} \cdot 10^{-2} .$$

Например, значение числа 34.85 вычисляется так:

$$34.85 = 3 \cdot 10 + 4 \cdot 1 + 8 \cdot 0.1 + 5 \cdot 0.01 .$$

В любой позиционной системе счисления позиция каждой цифры d_i имеет вес b_i, где b есть основание системы счисления. Общая форма записи чисел в таких системах счисления выглядит следующим образом:

$$^{d}p-1^{d}p-2\ldots^{d}1^{d}0.^{d}-1^{d}-2\ldots^{d}-n,$$

где p — число цифр, расположенных слева, а n — число цифр, расположенных справа от точки, отделяющей целую часть от дробной части числа. Значение числа представляет собой сумму произведений отдельных цифр на основание системы счисления в соответствующей степени:

$$D = \sum_{p-1 \geq i \geq -n} d_i \cdot b^i , \qquad (1.1)$$

где b — основание системы; p — число цифр, расположенных слева; n — число цифр, расположенных справа от точки.

В случае отсутствия точки в числе предполагается, что она находится справа от правой крайней цифры.

Любое число в позиционной системе счисления, если не учитывать возможных нулей в крайних позициях (разрядах), представляется единственным образом (очевидно, что число 34.85 равно числу 034.85000 и т. п.). Крайняя слева цифра в таком числе называется цифрой старшего разряда, а крайняя справа — цифрой младшего разряда.

Почти во всех вычислительных машинах используется двоичная система счисления — система, основание которой равно двум. Цифры 0 и 1, используемые в такой системе, называют битами, каждый бит d_i имеет вес 21. Ниже приведены примеры двоичных чисел и их десятичных эквивалентов, в которых для указания системы счисления используется нижний индекс:

$1001_2 = 1 \cdot 16 + 0 - 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 17_{10}$
$101010_2 = 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 42_{10}$
$110.011_2 = 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 + 0 \cdot 0.5 + 1 \cdot 0.25 + 1 \cdot 0.125 = 6.375_{10}$.

ВОСЬМЕРИЧНЫЕ И ШЕСТНАДЦАТЕРИЧНЫЕ ЧИСЛА

Для пользователей ЭВМ кроме систем счисления с основаниями 2 и 10 важны также и системы с другими основаниями. В частности, удобное представление чисел в ЭВМ обеспечивают восьмеричная и шестнадцатеричная системы счисления, в которых используются основания 8 и 16 соответственно.

В табл.1.1 приведены двоичные числа от 0 до 1111 и их восьме-

ричные, десятичные и шестнадцатеричные эквиваленты. Для записи чисел в восьмеричной системе счисления требуется 8 цифр, и поэтому используются цифры от 0 до 7 десятичной системы. Для шестнадцатеричной системы требуется 16 цифр, так что наряду с десятичными цифрами от 0 до 9 здесь используют латинские буквы A, B, C, D, E, F.

Восьмеричные и шестнадцатеричные системы счисления удобны для представления двоичных чисел. Поскольку последовательность из 3 бит (двоичных цифр) имеет восемь комбинаций, то каждое такое сочетание двоичных цифр можно однозначно представить с помощью одной восьмеричной цифры (табл. 1.1).

Таблица 1.1

Представление чисел

Двоичные числа	Десятичные числа	Восьмеричные числа	Шестнадцатеричные числа	Двоичный эквивалент
0	0	0	0	0000
1	1	1	1	0001
10	2	2	2	0010
11	3	3	3	0011
100	4	4	4	0100
101	5	5	5	0101
110	6	6	6	0110
111	7	7	7	0111
1000	8	10	8	1000
1001	9	11	9	1001
1010	10	12	*A*	1010
1011	11	13	*B*	1011
1100	12	14	*C*	1100
1101	13	15	*D*	1101
1110	14	16	*E*	1110
1111	15	17	*F*	1111

Аналогично строка из бит (двоичных цифр) может быть представлена одной шестнадцатеричной цифрой. Таким образом, двоичные целые числа легко преобразовать в восьмеричные (или шестнадцатеричные). Для этого двоичное число, начиная от двоичной точки и двигаясь влево, делится на группы из 3 (или 4) бит и каждая группа заменяется соответствующей восьмеричной (или шестнадцатеричной) цифрой. Приведем два примера:

101011000110_2=101 011 000 $110_2 = 5306_8$
101011000110_2= 1010 1100 $0110_2 = AC6_{16}$
11001110101001_2=011 001 110 101 $001_2 = 31651_8$
1011001110101001_2= 1011 0011 1010 $1001_2 = B3A9_{16}$

Из этих примеров следует, что с целью получения общего числа битов в строке, кратного трем или четырем, к двоичному числу можно добавить слева необходимое число нулей.

Если в двоичном числе содержатся цифры справа от двоичной точки, то они также могут быть преобразованы. Для этого, как и в предыдущем случае, начиная от двоичной точки, но двигаясь вправо, производится формирование групп по три бита в каждой. Если возникает необходимость, то к числу добавляется (теперь уже справа) соответствующее число нулей, как показано в примере, приведенном ниже:

$11.1010011011_2 = 011.101\ 001\ \ 101\ 100_2 = 3.5154_8$
$11.1010011011_2 = 0011.1010\ 0110\ 1100_2 = 3.A6C_{16}$

Контрольные вопросы

1. Как представляются числа в позиционной системе счисления?
2. Какая система счисления применяется в вычислительных машинах?
3. Переведите несколько чисел из десятичной системы в двоичную и из двоичной в десятичную.

Преобразования чисел из восьмеричной и шестнадцатеричной систем счисления в двоичную также выполняются достаточно легко. Для этого каждую восьмеричную или шестнадцатеричную цифру необходимо заменить соответствующим двоичным эквивалентом из 3 или 4 бит.

СПОСОБЫ ПРЕОБРАЗОВАНИЯ ЧИСЕЛ

Из приведенных ниже примеров следует, что предварительное преобразование в двоичную систему позволяет также преобразовывать числа из восьмеричной системы счисления в шестнадцатеричную, и наоборот:

$1573_8 = 001\ \ 101\ \ 111\ 0112 = 0011\ 0111\ \ 1011_8 = 37B_{16}$

$A748_{16} = 1010\ 0111\ 0100\ 1000_2 = 001\ 010\ 011\ 101\ 001\ 000_2 = 123510_8$

$3.1458 = 011.001\ 100\ 1012 = 0011.0011\ 0010\ 1000_2 = 3.328_{16}.$

В программном обеспечении большинства ЭВМ двоичные числа представляют в виде восьмеричных и шестнадцатеричных чисел. Шестнадцатиразрядное двоичное число может принимать 65 536 значений, которым в восьмеричной системе счисления соответствуют числа в пределах от 0 до 177777, а в шестнадцатеричной — от 0 до FFFF.

Выбор восьмеричного или шестнадцатеричного представления двоичных чисел почти полностью определяется требованиями, предъявляемыми к документации. В отношении аппаратных средств использование той или иной системы оказывает влияние только на то, будут ли лампочки и переключатели, размещенные на передней панели ЭВМ, сгруппированы по три или по четыре.

В общем случае преобразование числа из одной системы счисления в другую нельзя произвести путем простой замены. Для этого необходимо выполнить ряд арифметических операций. Рассмотрим, как преобразуются числа из системы с любым основанием в числа системы с основанием 10 и обратно при помощи десятичной арифметики.

Ранее мы выяснили, что значение числа в любой системе счисления определяется по формуле

$$D = \sum_{p-1 \geq i \geq -n} d_i \cdot b^i \ .$$

Таким образом, значение числа может быть найдено путем преобразования каждой его цифры в десятичный эквивалент и выполнения действий десятичной арифметики в соответствии с формулой. Некоторые примеры таких операций приведены ниже:

$1BE8_{16} = 1 \cdot 16^3 + 11 \cdot 16^2 + 14 \cdot 16^1 + 8 \cdot 16^0 = 7144_{10}$

$F1AC_{l6} = 15 \cdot 16^3 + 1 \cdot 16^2 + 10 \cdot 16^1 + 12 \cdot 16^0 = 61868_{10}$

$437.5_8 = 4 \cdot 8^2 + 3 \cdot 8^1 + 7 \cdot 8^0 + 5 \cdot 8^{-1} = 287.625_{10}$

$122.1_3 = 1 \cdot 3^2 + 2 \cdot 3^1 + 2 \cdot 3^0 + 1 \cdot 3^{-1} = 17.\overline{3}_{10}$

Наличие черты над цифрой 3 в последнем примере указывает на то, что эта цифра находится в периоде.

Общая формула преобразования целых чисел в десятичные имеет вид:

$D = ((...((d_p\_1) \cdot b + d_p\_2) \cdot b + ...) \cdot b + d_1) \cdot b + d_0$, где d_0 — остаток.

В соответствии с этой формулой сначала производится сложение цифры старшего разряда с нулем и умножение суммы на *b*. Затем к полученному результату прибавляется следующая цифра, а сумма вновь умножается на *b* и так далее до тех пор, пока не будут использованы все цифры преобразуемого числа. Например, можно записать:

$$F1AC_{16} = (((15) \cdot 16 + 1) \cdot 16 + 10) \cdot 16 + 12.$$

Общая формула, приведенная выше, сама по себе не представляет большого интереса, однако благодаря ей получен очень удобный метод преобразования десятичного числа *D* в число системы с основанием *b*. Рассмотрим, что произойдет, если правую часть формулы разделить на *b*. Так как первое слагаемое в правой части формулы кратно *b*, то частное от деления равно

$$Q = (...((d_{p-1}) \cdot b + d_{p-2}) \cdot b + ...) \cdot b + d_1.$$

Следовательно, d_0 можно вычислить путем деления *D* на *b*. Более того, частное *Q* имеет ту же форму записи, что и исходная формула. Поэтому последовательное выполнение операций деления на *b* приводит к получению последовательных цифр числа *D*. Ниже приведены некоторые примеры:

179/2=89 остаток 1 (цифра младшего разряда)

89/2 = 44 остаток 1

44/2 = 22 остаток 0

22/2=11 остаток 0

11/2 = 5 остаток 1

5/2 = 2 остаток 1

2/2=1 остаток 0

1/2 = деление прекращает, поскольку делимое стало меньше делителя (единица становится цифрой старшего разряда)

$179_{10}=10110011_2$

467/8 = 58 остаток 3 (цифра младшего разряда)

/8 = 7 остаток 2

/8=0 остаток 7 (цифра старшего разряда)

$467_{10} = 723_8$

3417/16 = 213 остаток 9 (цифра младшего разряда)

/16= 13 остаток 5

/16 = 0 остаток 13 (цифра старшего разряда)

$3417_{10}=D59_{16}$

Контрольные вопросы

1. Сформулируйте правила и способы преобразования чисел.
2. Преобразуйте произвольные числа из одной формы в другую.

СЛОЖЕНИЕ И ВЫЧИТАНИЕ ЧИСЕЛ

При сложении и вычитании недесятичных чисел вручную используются те же методы, которые изучаются в средней школе для выполнения действий над десятичными числами, различие состоит только в том, что применяют другие таблицы сложения и вычитания. Табл. 1.2 представляет собой таблицу сложения и вычитания двоичных чисел. Для того чтобы произвести сложение двух двоичных чисел *X* и *У*, вначале мы складываем биты их младших разрядов и исходный перенос $C_{вх}$, равный нулю, получая в соответствии с таблицей значения суммы {x + y + $C_{вх}$) и переноса $C_{вых}$. Затем мы продолжаем обработку битов справа налево, включая перенос каждой колонки в сумму следующей колонки. Ниже приведены два примера десятичного сложения и соответствующие им примеры двоичного сложения, где переносы показаны в виде последовательности битов С:

C		101111000
X	190	10111110
Y	+ 141	+10001101
X+Y	331	101001011
C		001011000
X	173	10101101
Y	+ 44	+ 00101100
X + Y	217	11011001

Вычитание двоичных чисел производится аналогичным образом, только вместо переносов используется заем:

B		001111100
X	_ 229	_ 11100101
Y	46	00101110
X-Y	183	10110111

Таблица 1.2

Таблица сложения и вычитания двоичных чисел

$C_{вх}$,$b_{вх}$	x	y	$x+y+c$	$C_{вых}$	$x-y-b_{вх}$	$b_{вх}$
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
1	0	0	1	0	1	1
1	0	1	0	1	0	1
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Операцию вычитания в вычислительных машинах очень часто применяют для сравнения двух чисел. Например, если выполнение операции *X*-*У* приводит к возникновению заема из старшего разряда, то число *X* меньше числа *У*, в противном случае число *X* больше или равно числу *У*. Таблицы сложения и вычитания могут быть получены для восьмеричных и шестнадцатеричных чисел, а также для чисел с любым другим основанием. Однако лишь у небольшого числа специалистов по вычислительной технике возникнет желание их запомнить. Таблицы вычитания не нужны совсем, так как данную операцию, как показано ниже, всегда можно выполнить, используя систему

представления чисел дополнениями и операцию сложения. В общем случае сложение (или вычитание) любой колонки цифр может быть произведено путем преобразования цифр данной колонки в десятичные, десятичного сложения и последующего преобразования результата с целью получения соответствующих цифр суммы и переноса в недесятичной системе счисления. (Перенос производится каждый раз, когда значение суммы в колонке превышает или равно основанию системы.) Поскольку сложение выполняется в десятичной системе счисления, необходимо знать только таблицу десятичного сложения; единственно, что мы должны дополнительно помнить,— это как осуществляется преобразование десятичных цифр в недесятичные и обратно. Последовательность шагов для сложения вручную двух шестнадцатеричных чисел приведена ниже:

C	1 1 0 0	1 1 0 0
X	$+1\ 9\ B\ 9_{16}$	1 9 11 9
Y	$\underline{C\ 7\ E\ 6_{16}}$	12 7 14 6
X+Y	$E\ 1\ 9\ F_{16}$	14 17 25 15
		14 16+1 16+9 15
		E 1 9 F

До настоящего времени мы имели дело только с положительными числами, однако существует также достаточно много способов представления и отрицательных чисел. При каждодневных расчетах мы обычно пользуемся системой представления чисел в прямом коде, однако в большинстве ЭВМ применяется система представления в виде дополнений. Дальнейшее описание охватывает практически все важнейшие системы представления чисел.

Контрольные вопросы

1. Сформулируйте правила сложения и вычитания чисел.
2. Проведите операции сложения и вычитания произвольных чисел.

ПРЕДСТАВЛЕНИЕ ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ

Представление чисел в прямом коде. Наиболее широко используемым способом представления десятичных чисел является прямой код. Числа при таком представлении состоят из величины числа и его знака. Таким образом, десятичные числа +98, −57, +123.5 и −13 интерпретируются как обычно.

В десятичной системе в случае отсутствия у числа символа зна-

ка число считается положительным, т. е. имеет знак « + ». Нуль может быть представлен двумя способами: + 0 и −0; однако значения, соответствующие этим представлениям, совпадают.

Систему представления чисел в прямом коде довольно просто применить к двоичным числам, если для представления знака ввести дополнительный разряд. Традиционно для этой цели используется старший разряд (плюсу соответствует нулевое значение этого разряда, минусу — единичное). Приведем несколько примеров 8-разрядных двоичных чисел в прямом коде и их десятичных эквивалентов:

$00101011_2 = +\ 43_{10}$ $\quad$ $10101011_2 = -43_{10}$

$01111111_2 = +127_{10}$ $\quad$ $11111111_2 = -127_{10}$

$00000000_2 = +\ 0_{10}$ $\quad$ $10000000_2 = -0_{10}$

Система представления в прямом коде содержит одинаковое число положительных и отрицательных чисел. Так, *n*-разрядное число в прямом коде может находиться в пределах от $-(2^{n-1}-1)$ до $+(2^{n-1}-1)$, причем нуль может быть представлен двумя способами.

Сложение двух чисел в прямом коде производится по правилам, которые изучались в средней школе. Если знаки чисел совпадают, то их величины складываются, а результату присваивается знак слагаемых. В случае несовпадения знаков из большей величины вычитается меньшая, а результату присваивается знак большей величины. Для того чтобы произвести вычитание чисел в прямом коде, необходимо изменить знак вычитаемого на противоположный и выполнить сложение.

Системы представления чисел дополнениями. В вычислительных машинах отрицательные числа обычно представляются в виде дополнений. Если в системе представления чисел в прямом коде со знаком для получения отрицательного числа достаточно изменить знак положительного числа, то в системах представления чисел дополнениями для получения отрицательного числа необходимо образовать его дополнение. Получить дополнение числа сложнее, чем изменить его знак, однако сложение и вычитание двух чисел в системе с дополнениями может производиться непосредственно, без проверки знаков и абсолютных значений, необходимой в системе представления в прямом коде. Рассмотрим две системы представления чисел дополнениями, одна из которых называется системой представления дополнением до основания системы счисления, а другая — системой представления неполным дополнением до основания системы счисления.

В любой системе с использованием дополнений мы имеем дело с фиксированным количеством цифр, или разрядов, например n. В дальнейшем будем считать, что основанием системы является n, а числа имеют следующий вид:

$$D = d_{n-1}d_{n-2}...d_1d_0$$

т. е. точка находится справа и числа являются целыми. Если при выполнении какой-либо операции будет получен результат, для записи которого требуется более n разрядов, то цифры старших разрядов отбрасываются. Если дополнение числа D образовать дважды, то в результате обязательно будет получено число D.

Представление чисел дополнением до основания системы счисления. В десятичной системе счисления дополнение до основания есть дополнение до десяти. В этой системе дополнение любого n-разрядного числа получается путем его вычитания из числа 10^n. Ниже приведены некоторые примеры 4-разрядных десятичных чисел и результаты их вычитания из числа 10000:

Десятичное число	Дополнение до десяти
1849	8151
2067	7933
5374	4626
100	9900
8151	1849

В общем случае дополнение любого n-разрядного числа D до основания b системы счисления может быть получено путем вычитания D из b^n. Если D находится в пределах от 1 до b^n - 1, то при вычитании получается другое число в тех же пределах. Если D равно нулю, то результат вычитания равен b^n и имеет вид 100... 00 при общем числе разрядов, равном n+1. Отбросив цифру старшего разряда, получим 0. Следовательно, в системе представления чисел дополнением до основания системы счисления существует только одно представление нуля.

Из предыдущих рассуждений можно сделать вывод, что для получения дополнения какого-либо числа до основания системы счисления необходимо выполнить операцию вычитания. Однако ее можно избежать, если b^n записать как (b^n—1) + 1, а b^n—D как ((b^n - 1) - D) + + l. Число b^n-1 имеет вид $mm…mm$, где $m = b$ -1, а число разрядов равно n. Например, 10 000 = 9 999+1.

Если для какой-либо цифры d определить дополнение как b-1-d, то (b^n - 1)-D можно получить путем образования дополнений для всех

цифр числа *D*. Следовательно, дополнение числа *D* до основания системы счисления получается в результате образования дополнений его отдельных цифр до *b* - 1 и последующего прибавления единицы. В этом нетрудно убедиться на рассмотренных выше примерах дополнений до основания 10. Дополнения цифр, используемых в двоичной, восьмеричной, десятичной и шестнадцатеричной системах счисления, приведены в табл. 1.3.

Таблица 1.3

Дополнения отдельных цифр

Цифра	Дополнение			
	двоичное	восьмеричное	десятичное	шестнадцатеричное
0	1	7	9	F
1	0	6	8	E
2	-	5	7	D
3	-	4	6	C
4	-	3	5	B
5	-	2	4	A
6	-	1	3	9
7	-	0	2	8
8	-	-	1	7
9	-	-	0	6
A	-	-	-	5
B	-	-	-	4
C	-	-	-	3
D	-	-	-	2
E	-	-	-	1
F	-	-	-	0

Представление чисел дополнением до двух. Для двоичных чисел дополнение до основания системы счисления называется дополнением до двух. В системе представления дополнением до двух число является положительным, если значение старшего разряда равняется нулю, и отрицательным, если оно равняется единице. Десятичный эквивалент двоичного числа, представленного дополнением до двух, вычисляется так же, как и для числа без знака, за исключением того,

что вес старшего значащего разряда в данном случае равен - 2^{n-1}, а не $+2^{n-1}$. Представляемые числа находятся в диапазоне от -(2^{n-1}) до +(2^{n-1}-1). Ниже приведены некоторые примеры 8-разрядных двоичных чисел и их дополнений до двух

$17_{10}=00010001_2$		$-99_{10}=10011101_2$	
11101110	дополнения	01100010	дополнения
+ 1	двоичных цифр	+ 1	двоичных цифр
$11101111_2=-17_{10}$		$01100011_2=99_{10}$	
$119_{10}=01110111_2$		$-127_{10}=10000001_2$	
10001000	дополнения	01111110	дополнения
+ 1	двоичных цифр	+ 1	двоичных цифр
$10001001_2=-119_{10}$		$01111111_2=127_{10}$	
$0_{10}=00000000_2$		$-128_{10}=10000000_2$	
11111111	дополнения	01111111	дополнения
+ 1	двоичных цифр	+ 1	двоичных цифр
$00000000_2=0_{10}$		$10000000_2=-128_{10}$	

В системе представления дополнением до двух нуль считается положительным, так как в знаковом разряде его представления находится 0. Такая договоренность согласуется с теорией чисел и всеми языками программирования. Так как в системе представления дополнением до двух существует только одно представление нуля, в ней имеется одно «лишнее» отрицательное число -2^{n-1}, у которого нет положительного эквивалента (по абсолютной величине). Как показано в последнем примере, попытка преобразовать такое число в соответствующее ему положительное число приводит к получению исходного числа. Тем не менее когда при арифметических операциях это число появляется как промежуточный результат, окончательный результат вычислений все же оказывается правильным (если, конечно, не предпринимались попытки получить из указанного числа число с противоположным значением).

Система представления чисел дополнением до двух, или в дополнительном коде, применяется во всех мини- и микро-ЭВМ, а также почти во всех больших ЭВМ, используемых в настоящее время.

Представление чисел неполным дополнением до основания системы счисления. В системе представления чисел неполным дополнением до основания дополнение n-разрядного числа D получается путем его вычитания из b^n - 1. Дополнение может быть также получено путем образования дополнения отдельных цифр числа D, но без последующего прибавления единицы как в системе представления дополнением до основания системы счисления. Для десятичной системы счисления такое дополнение называется дополнением до девяти. Ниже приведены некоторые примеры:

Десятичное число	Дополнение до девяти
1849	8150
2067	7932
5374	4625
7100	2899
8150	1849

Представление чисел дополнением до единицы. Для двоичных чисел неполное дополнение до основания системы счисления называется дополнением до единицы, или обратным кодом. Как и в случае представления дополнением до двух, при таком представлении старший разряд является знаковым и его значение для отрицательных чисел равно единице, а для положительных - нулю. Таким образом, существуют два представления нуля - положительный нуль (00...00) и отрицательный нуль (11...11). Представления положительных чисел в системах с дополнениями до единицы и до двух совпадают, тогда как представления отрицательных чисел отличаются на 1. При вычислении десятичного эквивалента числа, записанного как дополнение до единицы, старшему разряду приписывается вес $-(2^{n-1}-1)$, а не -2^{n-1} Представляемые числа находятся в диапазоне - $(2^{n-1}-1)/+(2^{n-1}-1)$. Ниже приведены некоторые примеры 8-разрядных двоичных чисел и их дополнений до единицы:

$17_{10}= 00010001_2$, $119_{10} = 01110111_2$

$11101110_2=-17_{10}$ $10001000_2= -119_{10}$

$-99_{10}=10011100_2$ $-127_{10}= 10001000_2$

$01100011_2=99_{10}$ $01111111_2 ,= 127_{10}$

$0_{10} = 00000000_2$, (положительный нуль)

$0_{10} = 11111111_2$, (отрицательный нуль)

Основными достоинствами системы представления дополнением до единицы являются ее симметричность и простота образования дополнений. Однако сложение чисел в такой системе выполняется не-

сколько сложнее, чем в системе представлением дополнением до двух. Кроме того, в машине, производящей операции над дополнениями чисел до единицы, необходимо с помощью аппаратных средств осуществлять две проверки на нуль (так как имеются два представления нуля) или преобразовывать код 11... 11 в 00...00.

Представление чисел в коде со смещением 2^{m-1}. В системе представления чисел в коде со смещением 2^{m-1} число X, находящееся в диапазоне от -2^{m-1} до $+2^{m-1}-1$, записывается в виде m-разрядного двоичного представления числа $X + 2^{m-1}$ (которое всегда неотрицательно и имеет значение меньше 2^m). Диапазон чисел, которые могут быть представлены в данной системе, является тем же, что и диапазон чисел, представляемых с помощью m-разрядных дополнений до двух. Действительно, представления любого числа в обеих системах полностью совпадают, за исключением знаковых разрядов, имеющих всегда противоположные значения.

Контрольные вопросы

1. Сформулируйте правила представления чисел в отрицательном коде.
2. Примените операции представления к произвольным числам.

СЛОЖЕНИЕ И ВЫЧИТАНИЕ ЧИСЕЛ В ДОПОЛНИТЕЛЬНОМ КОДЕ

Правила сложения. Табл. 1.4, в которой приведены десятичные числа и их эквиваленты в различных системах представления, позволяет понять, почему для выполнения арифметических операций наиболее предпочтительной является двоичная система представления чисел в дополнительном коде. Если начать счет с числа 10002 (-810) и осуществлять его в прямом направлении, то можно заметить, что каждое последующее число, включая последнее число 01112(+710), может быть получено путем прибавления к предыдущему числу единицы без учета каких-либо переносов за пределы четвертого разряда. Этого нельзя сказать о числах, представленных в прямом и обратном кодах. Так как обычное сложение является по существу более сложным вариантом счета, то числа, представленные в дополнительном коде, могут складываться по правилам двоичного сло-

жения, но без учета каких-либо переносов за пределы старшего разряда.

Таблица 1.4

Десятичные и 4-разрядные двоичные числа

Десятичное число	Дополнительный код	Обратный код	Прямой код	Код со смещением 2^{m-1}
-8	1000	0111	1000	0000
-7	1001	1000	1111	0001
-6	1010	1001	1110	0010
-5	1011	1010	1101	0011
-4	1100	1011	1100	0100
-3	1101	1100	1011	0101
-2	1110	1101	1010	0110
-1	1111	1110	1001	0111
0	0000	0000	0000	1000
1	0001	0001	0001	1001
2	0010	0010	0010	1010
3	0011	0011	0011	1011
4	0100	0100	0100	1100
5	0101	0101	0101	1101
6	0110	0110	0110	1110
7	0111	0111	0111	1111

Если результат не выходит за границы диапазона чисел, которые могут быть представлены в рассматриваемой системе, сумма всегда будет получаться правильной. Это подтверждается примерами сложения десятичных чисел и соответствующих им 4-разрядных двоичных чисел, представленных в дополнительном коде:

```
   +3      0011      -2      1110
 + +4    + 0100    + -6    + 1010
   +7      0111      -8     1|1000

   +6      0110      +4      0100
 + -3    + 1101    + -7    + 1001
   +3     1|0011     -3      1101
```

Переполнение. Если при выполнении операции сложения получают результат, который выходит за пределы диапазона представляемых чисел, то имеет место переполнение. Сложение двух чисел с разными знаками никогда не приводит к переполнению, чего нельзя сказать о сложении двух чисел с одинаковыми знаками, например:

-3	1101	+5	0101
+ -6	+ 1010	+ +6	+ 0110
-9	1|0111 = +7	+11	1011=-5

-8	1000	+7	0111
+ -8	+ 1000	+ +7	+ 0111
-16	1|0000 = +0	+14	1110=-2

Существует простое правило, позволяющее выявить наличие переполнения: при сложении переполнение происходит только в том случае, если слагаемые имеют одинаковые знаки, а знак суммы отличается от знака слагаемых. Иногда правило наличия переполнения формулируют, используя понятие переносов, возникающих при выполнении операции сложения: переполнение при сложении происходит в том случае, если значения битов переноса в знаковый разряд и из знакового разряда различны. Внимательное изучение табл. 1.4. показывает, что эти два правила эквивалентны.

Правило наличия переполнения можно также сформулировать следующим образом: прибавление к какому-либо числу положительного числа приводит к переполнению, если указатель пройдет границу между позициями +7 и −8.

Большинство ЭВМ для обнаружения переполнения имеют встроенные аппаратные средства.

Правила вычитания. Вычитание чисел в дополнительном коде может производиться так, как если бы они являлись обычными двоичными числами без знака. Можно также сформулировать и соответствующие правила выявления переполнения. Однако непосредственное вычитание чисел в дополнительном коде производится довольно редко. Обычно, для того чтобы вычесть одно число из другого, вначале путем образования дополнения до двух получают противополож-

ное значение вычитаемого, а затем складывают его с уменьшаемым, используя обычные правила сложения. Вычитание, реализуемое таким способом, можно выполнить, пользуясь всего одной операцией сложения. Для этого формируют обратный код вычитаемого и складывают его с уменьшаемым при начальном переносе, равном 1, а не 0.

1 − начальный перенос

	+4		0100		0100
−	+3	−	0011	+	1100
	+1				1|0001

1 − начальный перенос

	+3		0011		0011
−	+4	−	0100	+	1011
	-1				1111

1 − начальный перенос

	+3		0011		0011
−	-4	−	1100	+	0011
	+7				0111

Контрольные вопросы

1. Сформулируйте правила сложения и вычитания чисел в дополнительном коде.
2. Проведите операции сложения и вычитания произвольных чисел.

СЛОЖЕНИЕ И ВЫЧИТАНИЕ ЧИСЕЛ В ОБРАТНОМ КОДЕ

Наличие переполнения при вычитании можно установить путем изучения знаков уменьшаемого и обратного кода вычитаемого, используя при этом то же правило, что и в случае сложения. Для обнаружения переполнения можно также рассмотреть переносы в знаковый разряд и из знакового разряда и воспользоваться соответствующим правилом, сформулированным для сложения. Последний способ позволяет выявить переполнение без учета знаков уменьшаемого, обратного кода вычитаемого и разности.

Попытка получения противоположного значения для «лишнего» отрицательного числа приводит (в соответствии с правилами, сформулированными выше) к переполнению при прибавлении 1:

$$
\begin{array}{rl}
-(-8)=-1000= & 0111 \\
+ & \\
 & \underline{0001} \\
 & 1000 = -8
\end{array}
$$

Тем не менее данное число может использоваться при выполнении вычитания и сложения, если окончательный результат не выходит за пределы допустимого диапазона чисел.

Числа в дополнительном коде и двоичные числа без знака. Так как числа в дополнительном коде складываются и вычитаются в соответствии с теми же основными алгоритмами, что и числа без знака той же длины, то вычислительной машине для выполнения операций над числами обоих видов необходим всего один тип команды сложения или вычитания. Однако результаты, полученные при выполнении такой команды сложения или вычитания, должны интерпретироваться программой по-разному в зависимости от того, какими числами оперирует машина – числами со знаком (т.е. от −8 до +7) или без знака (от 0 до 15).

В заключение отметим, что при сложении чисел без знака наличие переноса из старшего разряда или заема для данного разряда указывает на то, что результат операции выходит за пределы допустимого диапазона чисел. При сложении чисел со знаком, представленных в дополнительном коде, на возникновение такой ситуации указывает соблюдение условия переполнения, которое было определено ранее. Использование переноса из старшего разряда в данном случае является неуместным в том смысле, что переполнение может произойти независимо от того, имеется перенос или нет.

Для рассмотрения правил сложения чисел, представленных в обратном коде, обратимся к табл. 1.4. Если мы будем вести счет в прямом направлении начиная с числа 1000_2(−710), то каждое последующее число в обратном коде получается путем прибавления единицы к предыдущему числу за исключением перехода от числа 11111(-0) к числу 0001_2 (+110).

Для продолжения правильного счета при получении числа 1111_2 необходимо прибавить к нему цифру 2, а не 1. Отсюда вытекает правило сложения чисел в обратном коде: сложение таких чисел сводится к обычному двоичному сложению, но когда осуществляется переход через 1111_2, к результату следует прибавить дополнительную 1.

Переход при сложении через 1111_2 может быть легко зафиксирован по наличию переноса из знакового разряда. Таким образом, можно достаточно просто сформулировать правило сложения чисел, представленных в обратном коде: следует выполнить обычное двоичное сложение, а при наличии переноса из знакового разряда прибавить к результату 1. Это правило часто называют правилом циклического переноса. Ниже приведены примеры сложения чисел в обратном коде:

+3	0011	+4	0100	5	0101
+4	0100	-7	1000	-5	1010
+7	0111	-3	1100	-0	1111
-2	1101	+6	0110	-0	1111
-5	1010	-3	1100	-0	1111
-7	10111	+3	10010	-0	11110
	1		1		1
	1000		0011		1111

Если следовать сформулированному выше правилу двухшагового сложения, то при сложении какого-либо числа с обратным кодом этого числа всегда получается отрицательный нуль. Действительно, выполнение операции сложения в соответствии с этим правилом приводит к получению положительного нуля только в том случае, если оба слагаемых являются положительными нулями.

Некоторые сумматоры позволяют производить сложение чисел, представленных в обратном коде, за один шаг. С этой целью выход переноса старшего разряда сумматора соединяется непосредственно со входом переноса младшего разряда. Если до начала сложения выходной сигнал переноса не будет установлен в нуль, это может вызвать появление неоднозначности или даже неустойчивость системы; в результате может получиться либо положительный, либо отрицательный нуль. Во всех остальных случаях при использовании циклического переноса неоднозначности не возникает.

Как и для чисел в дополнительном коде, простейший способ вычитания чисел в обратном коде состоит в том, чтобы получить дополнение вычитаемого и выполнить сложение. Правила обнаружения переполнения при сложении и вычитании чисел в обратном коде явля-

ются такими же, как и для чисел в дополнительном коде. Операции над такими числами могут выполняться на вычислительных машинах, предназначенных для работы с дополнительными кодами, путем использования команд «Сложение с переносом» и «Вычитание с переносом».

Контрольные вопросы

1. Сформулируйте правила сложения и вычитания чисел в обратном коде.
2. Проведите операции сложения и вычитания произвольных чисел.

ДВОИЧНОЕ УМНОЖЕНИЕ

В данном разделе рассматриваются алгоритмы умножения чисел со знаком и без знака. Многие вычислительные машины позволяют реализовать эти или подобные алгоритмы в машинных командах. Однако в малых ЭВМ с целью экономии аппаратных средств такие команды обычно не предусматриваются, поэтому алгоритмы умножения реализуют программным способом с использованием простых операций сложения и сдвига.

Для описания алгоритмов в данном и последующих разделах книги используется расширенный язык программирования паскаль, который позволяет производить обращение к отдельным разрядам слова.

Рассматриваемые здесь алгоритмы могут быть легко транслированы на язык ассемблера большинства ЭВМ.

Умножение чисел без знака. В средней школе мы изучали, что умножение чисел производится путем сложения ряда сдвинутых относительно друг друга множимых с учетом цифр множителя. Как показано ниже, этот способ можно также использовать и для получения произведения двух двоичных чисел без знака.

	1011	*множимое*
11	×1101	*множитель*
× 13	1011	
33	0000	*сдвинутые*
11	1011	*множимые*
	1011	
143	1001111	*произведение*

Формирование сдвинутых множимых при двоичном умножении выполняется довольно просто, так как возможными цифрами множителя могут быть только 0 и 1.

Вместо получения всех сдвинутых множимых и последующего их сложения более удобным оказывается прибавление каждого сдвинутого множимого по мере его формирования к частичному произведению. Пример умножения с использованием этого метода показан на рис. 1.1.

1011	множимое
× 1101	множитель
00000000	частичное произведение
1011	сдвинутое множимое
00001011	частичное произведение
1011	сдвинутое множимое
00110111	частичное произведение
1011	сдвинутое множимое
10001111	произведение

Рис. 1.1. Умножение с использованием частичных произведений

Когда в ЭВМ производится умножение m-разрядного слова на n-разрядное, для представления окончательного произведения требуется самое большее $n+m$ разрядов. Поэтому типичный алгоритм умножения позволяет перемножить два n-разрядных слова и получить $2n$-разрядное двойное слово произведения.

Умножение чисел со знаком. Умножение чисел со знаком можно выполнить, используя умножение чисел без знака и обычные правила, которые изучаются в средней школе: следует умножить величины чисел и сделать произведение положительным, если знаки сомножителей совпадают, или отрицательным, если знаки разные. Этот способ очень удобен для умножения чисел в прямом коде.

В системе представления чисел в дополнительном коде для получения величины отрицательного числа требуется выполнить операцию образования дополнения, а если должен быть получен отрица-

тельный результат, то необходимо образовать дополнение произведения без знака. Такие сложности приводят к необходимости поиска более эффективного способа умножения чисел, представленных в дополнительном коде.

В общем случае умножение чисел без знака выполняется как последовательность операций сложения без знака над множимыми, сдвинутыми относительно друг друга. Величина сдвига множимого для каждого шага соответствует весу разряда множителя. Разряды множителя, представленного в дополнительном коде, обладают тем же весом, что и разряды множителя без знака, за исключением старшего разряда, вес которого является отрицательным. Из этого следует, что умножение чисел в дополнительном коде может выполняться в виде последовательности операций сложения над сдвинутыми относительно друг друга множимыми, за исключением последнего шага, в качестве которого используется вычитание. Тогда становится возможным получить алгоритм умножения чисел в дополнительном коде.

Контрольные вопросы

1. Сформулируйте правила сложения и вычитания чисел.
2. Проведите операции сложения и вычитания произвольных чисел.

Деление чисел без знака. Простейший алгоритм двоичного деления также основан на методе деления, который изучают в средней школе. Как при десятичном, так и при двоичном делении, с целью определения числа, кратного сдвинутому делителю и подлежащего вычитанию, мы мысленно сравниваем уменьшенное делимое с числами, кратными делителю. В случае десятичного деления (рис. 1.2) мы сначала выбираем число 11 как наибольшее число, кратное 11, которое меньше 21, а затем выбираем число 99 как наибольшее число, кратное 11, которое меньше 107. В случае двоичного деления выбор чисел производится несколько проще, так как выбрать можно только нуль или сам делитель. Однако для того чтобы выбрать правильный сдвинутый делитель, здесь требуется выполнить операцию сравнения.

ДВОИЧНОЕ ДЕЛЕНИЕ

```
  19          1011        частное
11) 217   1011) 11011001  делимое
 11          1011         сдвинутый делитель
 107          0101        уменьшенное делимое
  99          0000        сдвинутый делитель
   8          1010        уменьшенное делимое
              0000        сдвинутый делитель
             10100        уменьшенное делимое
              1011        сдвинутый делитель
             10011        уменьшенное делимое
              1011        сдвинутый делитель
              1000        остаток
```

Рис. 1.2. Пример длинного деления

В процессорах деление без знака дополняет операцию умножения. Для типичного алгоритма деления делимым является двойное слово, а делителем одинарное; частное и остаток получаются в виде одинарных слов. Если при выполнении такого деления окажется, что делитель равен нулю, или для представления частного потребуется более одного слова, то происходит переполнение. Вторая ситуация имеет место только в том случае, если делитель меньше или равен старшему слову делимого.

Деление чисел со знаком. Как и в случае умножения, существует несколько методов, позволяющих выполнять деление непосредственно над числами, представленными в дополнительном коде. Эти методы часто реализуются схемным путем. Однако для программ, написанных на языке ассемблера, простейший способ деления чисел в дополнительном коде заключается в преобразовании этих чисел в положительные, выполнении деления с восстановлением остатка и последующем приведении результата к соответствующему.

Алгоритм деления с восстановлением остатка в данном случае является упрощенным, поскольку отрицательные числа, получаемые в результате пробных вычитаний, теперь могут быть представлены в пределах длины слов делимого и делителя. Приспосабливая рассмотренный выше алгоритм для деления чисел со знаком, необходимо учесть, что делитель, остаток и частное состоят из 15 разрядов и знака, а делимое — из 31 разряда и знака.

Контрольные вопросы

1. Сформулируйте правила двоичного деления чисел.
2. Проведите операции двоичного деления произвольных чисел.

Глава 2

СТРУКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

ШИНЫ

Шина - это группа проводников (металлических полосок на печатной плате), объединенных по функциональному назначению, проходящих через всю систему, к которым подключаются все устройства, образующие вычислительную систему. Линии шины подразделяются на три группы: шина адреса, шины данных и шина управления. На рис. 2.1 показано, как выглядит система на базе процессора 80286. В центре находится процессор, а все остальные устройства соединяются с ним отдельными линиями связи. Практически такую систему реализовать нельзя, так как микросхема 80286 имеет ограниченное число контактов, связывающих ее с внешним миром. Если применить общую схему, показанную на рис. 2.1, то придется ограничить число внешних устройств, так как для них потребуются отдельные контакты, процессор 80286 подключается к *шине,* как показано на рис. 2.2.

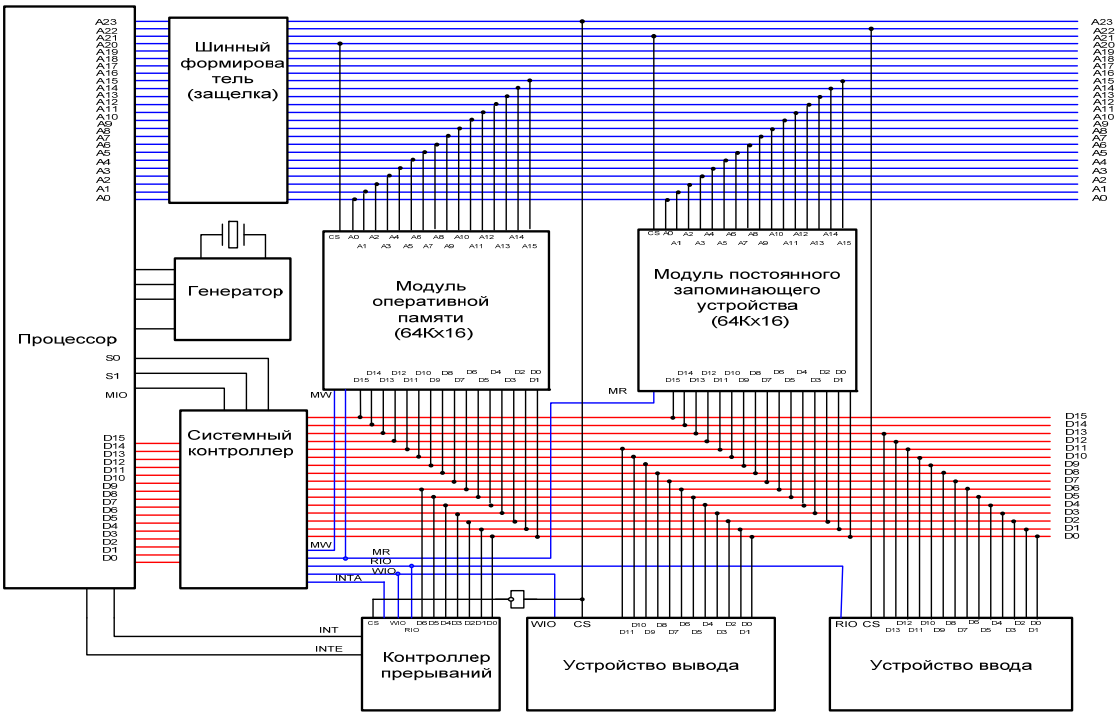


Рис. 2.1. Система на базе процессора 80286

Каждому устройству назначается адрес, а таким устройствам, как память - диапазон адресов. Для взаимодействия с конкретным устройством процессор выдает на 24 линии шины адреса логические значения (0 и 1), соответствующие битам адреса этого устройства. Хотя адрес подается во все устройства, только адресуемое устройство распознает свой собственный и реагирует на него. Если процессор выдает информацию, он заставляет логические значения на 16 линиях шины данных соответствовать передаваемому слову.

Если же процессор принимает информацию, то значения на линиях шины данных устанавливает адресуемое устройство. В обоих случаях процессор сигналами на линиях шины управления показывает, принимает он информацию или передает и является ли адресуемое устройство памятью или устройством ввода-вывода.

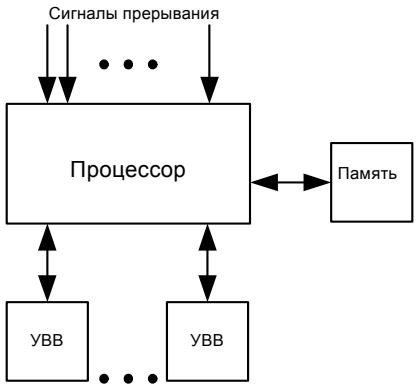


Рис. 2.2. Шина процессора

Линии шины. На рис. 2.3 показаны сигналы процессора 80286, относящиеся к шине. Адрес выдается на линии A_0 - - A_{23}. В операциях с памятью адрес является физическим (см. гл. 5). В реальном режиме адреса содержат только 20 бит, а не 24. Поэтому при работе процессора в реальном режиме для памяти важны только линии A_0 – A_{19}. В операциях ввода-вывода адрес является номером порта. Так как номера портов имеют длину 16 бит, для устройства ввода-вывода важны только линии A_0 – A_{15} .

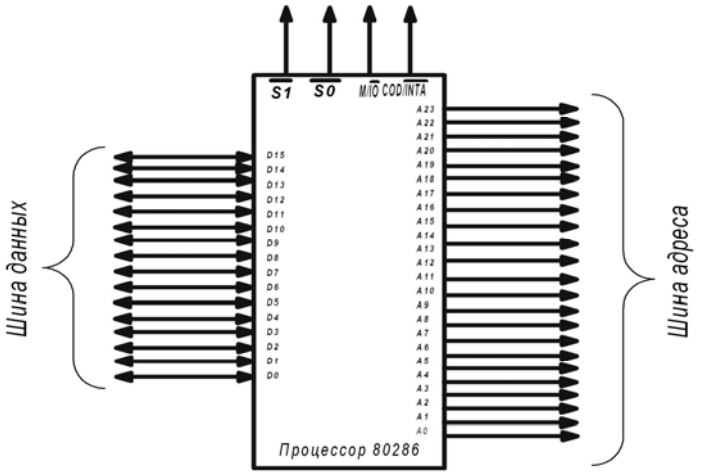


Рис. 2.3. Интерфейс шины процессора 80286

Линии D_0 - D_{15} несут данные, которые передаются или принимаются в *цикле шины.* Четыре *линии состояния* S1, S0, COD/INTA и M/IO показывают характер цикла шины и служат линиями управления шиной.

В таблице показано, как интерпретировать линии состояния цикла шины. Сигналы S0 и S1 показывают, что делает процессор: принимает (считывание или ввод), передает (запись или вывод) или что-то еще (прерывание, останов или отключение). Когда процессор передает или принимает, сигнал M/IO показывает, производится операция с памятью или с устройством ввода-вывода. Когда процессор считывает из памяти, сигнал COD/INTA показывает, производит он выборку команд (т.е. кода) или считывание данных.

Отметим черту над сигналом S1 на рис. 2.3. Она означает, что на линии действует логическая инверсия сигнала, т.е. НЕ S1. Ради простоты мы опускаем эти детали из рассмотрения (но не из рисунков).

Коды состояния цикла шины

Операция	$\overline{S1}$	$\overline{S0}$	M/$\overline{IO}$	COD/$\overline{INTA}$	Цикл шины
			0	1	Ввод
Процессор принимает	0	1	1	0	Считывание из памяти (данные)
				1	Считывание из памяти (команды)
Процессор выдает	1	0	0	1	Вывод
			1	0	Запись в память
			0	0	Подтверждение прерывания
Остальные	0	0	1	0	Останов или отключение*

*A1=1 для останова, A1 = 0 для отключения.

Временная диаграмма циклов шины. В системе на базе процессора 80286 для измерения времени применяются три единицы: *такты системной синхронизации, циклы процессора* и *циклы шины.*

Основным временным сигналом в системе является сигнал системной синхронизации CLK. Подобно ударам метронома он помогает всем компонентам системы действовать в правильные моменты времени. Период повторения сигнала CLK называется тактом системной синхронизации. При частоте синхронизации 16 МГц такт составляет 62,5 нс.

Два такта системной синхронизации образуют цикл процессора, причем первый из них называется фазой 1, а второй - фазой 2 (рис. 2.4). Следовательно, процессор 80286 работает на частоте, которая в два раза меньше частоты системной синхронизации. Например, при работе процессора на частоте 8 МГц частота системной синхронизации равна 16 МГц, а цикл процессора длится 125 нс. Далее мы покажем, почему частота системной синхронизации в два раза выше частоты работы процессора.

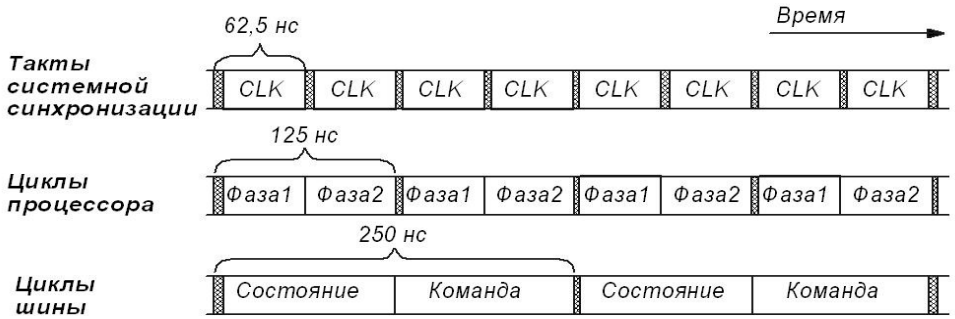


Рис. 2.4. Временная диаграмма шины

Каждый цикл шины включает в себя два действия и потому занимает по крайней мере два цикла процессора. В первом цикле процессор выдает состояние шины и адресную информацию, что объясняет название этого цикла - *цикл состояния.* Подключенные к шине устройства должны разобраться, как им реагировать на обращение процессора. В следующем цикле процессора, называемом *командным циклом,* процессор и адресуемое устройство производят передачу данных.

Некоторые устройства (ввода-вывода или памяти) не могут отреагировать к моменту окончания командной части цикла шины. Можно, конечно, уменьшить системную синхронизацию так, чтобы самое медленное устройство успевало отреагировать вовремя. Однако есть другое, более приемлемое решение. У процессора предусмотрен вход готовности КЕАБУ. Если устройство сигнализирует о неготов-

ности во время командной части цикла шины, процессор расширяет ее на дополнительный цикл процессора, который называется *состоянием ожидания.* Если устройство продолжает сигнализировать о неготовности, вводятся дополнительные состояния ожидания (рис. 2.5).

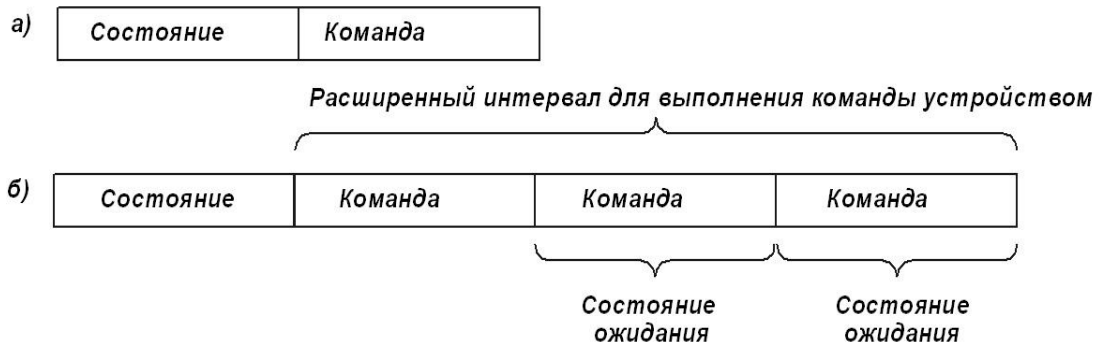


Рис. 2.5. Расширение цикла шины с помощью состояний ожидания: а - обычный цикл шины, б - цикл шины, расширенный на два состояния ожидания

Вместо введения состояний ожидания только для тех устройств, которые нуждаются в них, можно добавить фиксированное число состояний ожидания к каждому циклу шины. Представляется, что этот прием не лучше понижения частоты системной синхронизации, но иногда он предпочтительнее. Дело в том, что многие команды оперируют только регистрами и не требуют никаких перадач по шине. На такие команды состояния ожидания не повлияют. Более того, команды типа деления DIV выполняются столь долго, что состояние ожидания пренебрежимо мало. Измерения по бенчмарк – программам показали, что добавление одного состояния ожидания к каждому циклу шины ухудшает производительность процессора примерно на 25 *%,* хотя время цикла шины увеличивается на 50 *%.*

Подведем итоги. Минимальной измеримой единицей времени является такт системной синхронизации. Каждый цикл процессора занимает точно два такта системной синхронизации. Каждый цикл шины имеет длительность, равную по крайней мере двум циклам процессора. Циклы шины состоят из командной части (один или несколько циклов процессора) и части состояния (один цикл процессора). Командная часть расширяется сверх одного цикла процессора путем введения состояний ожидания.

Генерирование импульсов синхронизации. Импульсы системной синхронизации вырабатывает не процессор 80286, а микросхема 82284 генератора синхронизации. На рис. 2.6 показано ее подключение к процессору. Частота сигнала системной синхронизации определяется резонансной частотой кварца. Для изменения ее необходимо заменить только кварц.

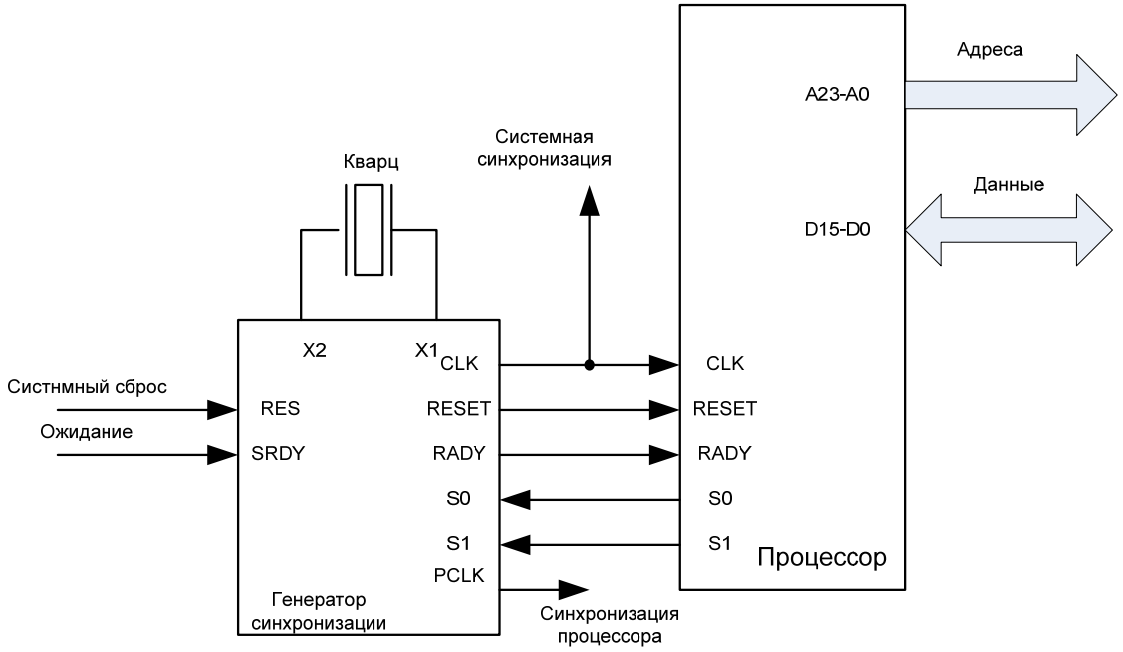


Рис. 2.6. Подключение генератора синхронизации к процессору

Кроме системной синхронизации CLK генератор формирует импульсы, соответствующие циклам процессора PCLK для обслуживания остальной части системы. Чтобы сигнал PCLK правильно отражал один цикл процессора, а не фазу 2 одного цикла и фазу 1 следующего, необходимо синхронизировать генератор 82284 и процессор. Для этого все сигналы системного сброса подаются в микросхему 82284, а затем поступают в процессор. Эти сигналы и обеспечивают синхронизацию двух микросхем.

Еще одна функция генератора синхронизации касается состояний ожидания. Для введения состояния ожидания процессор требует, чтобы сигнал READY отсутствовал в точный момент цикла шины и в течение определенного интервала. При подаче сигналов READY в генератор он помогает в синхронизации, поэтому временная точность в запросах состояний ожидания может быть меньше.

Контрольные вопросы

1. На какие группы подразделяются линии шины?
2. Дайте определение компьютерной шины.
3. Какие единицы применяются для измерения времени в системе на базе процессора?
4. Какие существуют циклы процессора?
5. Какую функцию выполняют импульсы системной синхронизации?

Конвейеризация шины. Стоимость компьютерной памяти зависит от ее быстродействия: чем меньше время обращения, тем выше удельная стоимость. В процессоре 80286 применяется интересный способ, называемый *конвейеризацией шины,* для повышения производительности при медленной памяти, хотя для этого приходится усложнять схемы.

При обсуждении цикла шины мы предполагали, что процессор выдает адрес на шину адреса в начале части состояния цикла шины и сохраняет его в течение всего цикла шины. Мы предполагали также, что данные помещаются на шину в командной части и сохраняются на ней до конца цикла шины. Но теперь представим себе, что адрес выдается немного раньше начала цикла шины и не будем требовать данных даже немного спустя после начала командной части цикла. Очевидно, при этом на реакцию устройствам отводится больше времени. Рассмотрим этот способ подробнее.

На рис. 2.7, *а* показано, что в действительности происходит на трех частях шины процессора (линиях состояния, шине адреса и шине данных). Предполагается, что частота синхронизации процессора составляет 8 МГц. Видно, что адрес опережает цикл шины на один такт системной синхронизации (62,5 нс). На такой же интервал данные "залезают" в следующий цикл шины. (Отметим, что здесь используется дополнительная разрешающая способность по времени, обеспечиваемая двойной частотой системной синхронизации.)

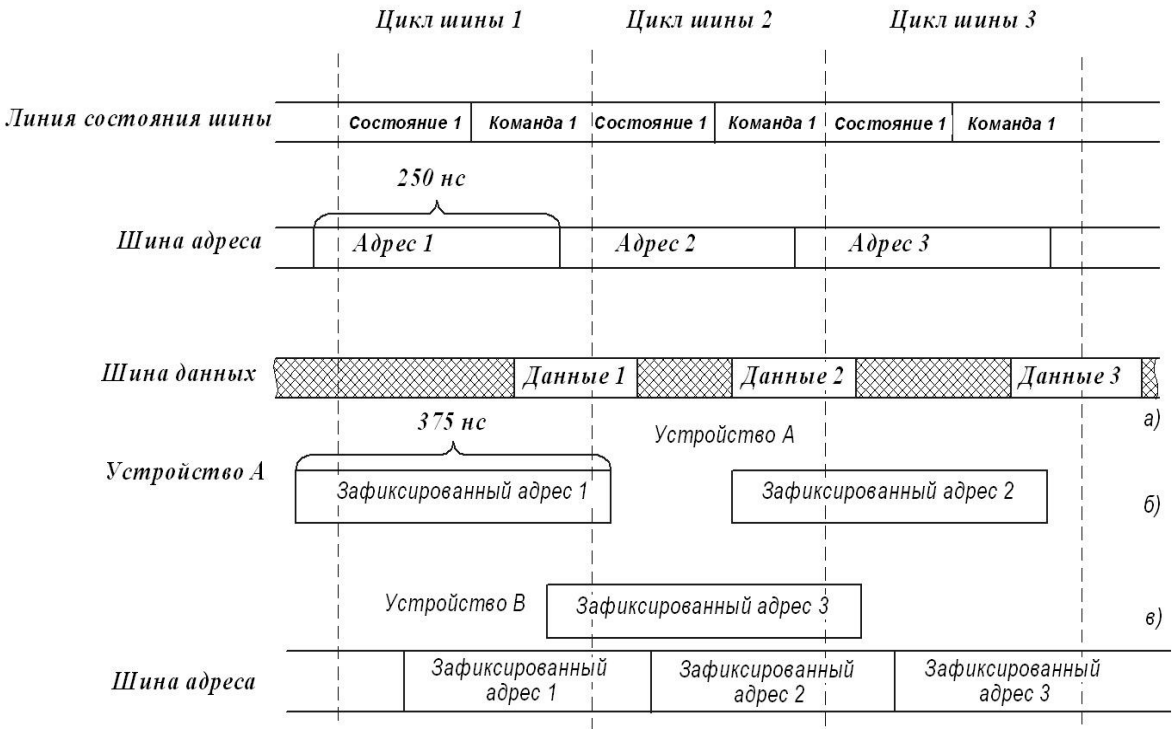


Рис. 2.7. Временная диаграмма конвейерной шины

Большинство устройств работают неправильно, если их адрес исчезает в середине цикла шины. Но на рис. 2.7 видно, что именно это и происходит на шине процессора, когда адрес для одного цикла шины освобождает место для следующего адреса. Устройство должно зафиксировать свой адрес, как только он появляется на шине адреса, и должно помнить о том, что оно адресовано до окончания цикла шины. Предположим, что каждое устройство подключено к шине адреса через *защелку* (см. микросхему ALS573 на рис. 2.3). Когда на входе *строба* C появляется сигнал, защелка запоминает значения восьми сигналов на входах D_0 –D_7 . Запомненный байт подается на входы Q_0 - Q_7, если активен сигнал *разрешения выхода* OE. Изменения входных сигналов не влияют на выходные сигналы до нового стробирования. При необходимости три микросхемы ALS573 запоминают все 24 бита адреса.

Мы предполагаем, что каждое устройство имеет свою защелку, поэтому адресуемое устройство стробирует свою защелку, когда адрес впервые появляется на шине, и адрес сохраняется защелкой до тех пор, пока устройство не заканчивает операцию с данными. Предполагается, что устройство А адресуется в первом цикле шины, устрой-

ство В - во втором и снова устройство А - в третьем. Теперь на одну передачу по шине каждое устройство имеет не 250 нс (продолжительность цикла шины), а 375 нс. При этом не введено ни одного состояния ожидания и частота синхронизации не уменьшена.

На первый взгляд кажется, что мы получили что-то из ничего. Подозрительный читатель заметит, что никакого выигрыша от защелок нет. Конечно, выигрыша нет, если два цикла шины обращаются к одному и тому же устройству, один сразу после другого (рис. 2.8).

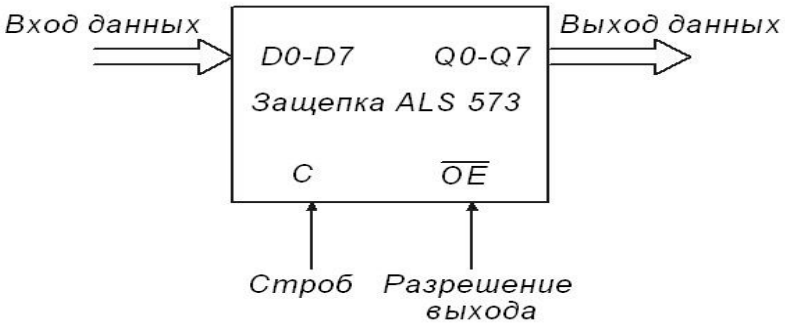


Рис. 2.8. 8-битная защелка ALS573

В случае устройств ввода-вывода программист может легко обеспечить, чтобы команды IN и OUT, обращающиеся к одному и тому же порту, не были слишком близки друг к другу. Поэтому потребуются дополнительные схемы для введения состояния ожидания во второй цикл каждой пары циклов шины, в которых обращения производятся к одному и тому же запоминающему устройству. Мы вернемся к этому вопросу при обсуждении динамических ЗУПВ.

Еще один недостаток конвейеризации шины заключается в том, что в каждом устройстве должна быть своя защелка. Конечно, мы можем упростить интерфейс устройств с шиной, если предусмотреть одну защелку для всей системы, но при этом теряются достоинства конвейеризации (см. временную диаграмму на рис. 2.7, *в*). В этом случае шина процессора 80286 напоминает шины других микропроцессоров, например 8086. Защелки в каждом устройстве не нужны и не требуется вводить состояния ожидания в последовательных циклах шины с обращением к одному и тому же устройству.

Буферирование шины данных. В большой вычислительной системе мощности процессора 80286 может не хватить для управления всеми устройствами, подключенными к шине. Для шины адреса

эта проблема решается с помощью мощной защелки (см. выше). Однако защелка для шины данных не подходит, так как в зависимости от типа цикла шины данные передаются по одним и тем же линиям в обоих направлениях. Для шины данных вместо защелки требуется *приемопередатчик.* Это мощный двунаправленный усилитель, который передает сигналы в любом направлении, превращая выходные сигналы в более мощные, чем входные.

Приемопередатчик ALS245 показан на рис. 2.9. Он имеет два набора линий данных $A_0 – A_7$ и $B_o – B_7$.

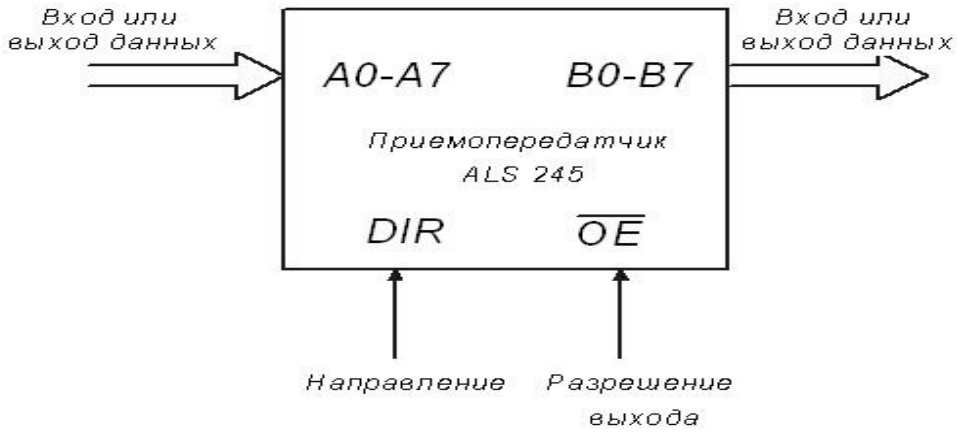


Рис. 2.9. 8-битный приемопередатчик ALS245

Когда сигнал DIR (направление) активен, данные передаются слева направо - от A к B. Если же DIR = 0, данные передаются справа налево - от B к A. В любом случае для производства передачи сигнал OE (разрешение выхода) должен быть активным.

Контрольные вопросы

1. В чем заключается конвейеризация шин?
2. Какими функциями обладает строб?
3. Для чего необходимо буферирование шины данных?

Шинные команды. Может показаться, что для стробирования защелок, разрешения приемопередатчиков и декодирования состояний шины в точные моменты времени потребуются сложные схемы. Фактически же все эти действия осуществляет микросхема 82288, называемая контроллером шины. На рис. 2.10 показано, как подключить контроллер шины.

Контроллер шины воспринимает состояние шины с выходов S0, S1 и M/IO процессора. Он также воспринимает сигнал READY для правильного ввода состояния ожидания. Контроллер превращает входные сигналы в отдельные сигналы *шинных команд:* считывание из памяти MRDC, запись в память MWTC, считывание ввода-вывода IORC, запись ввода-вывода IOWC и подтверждение прерывания INTA. Пользоваться отдельными сигналами намного проще, чем "сырыми" сигналами состояния от процессора 80286.

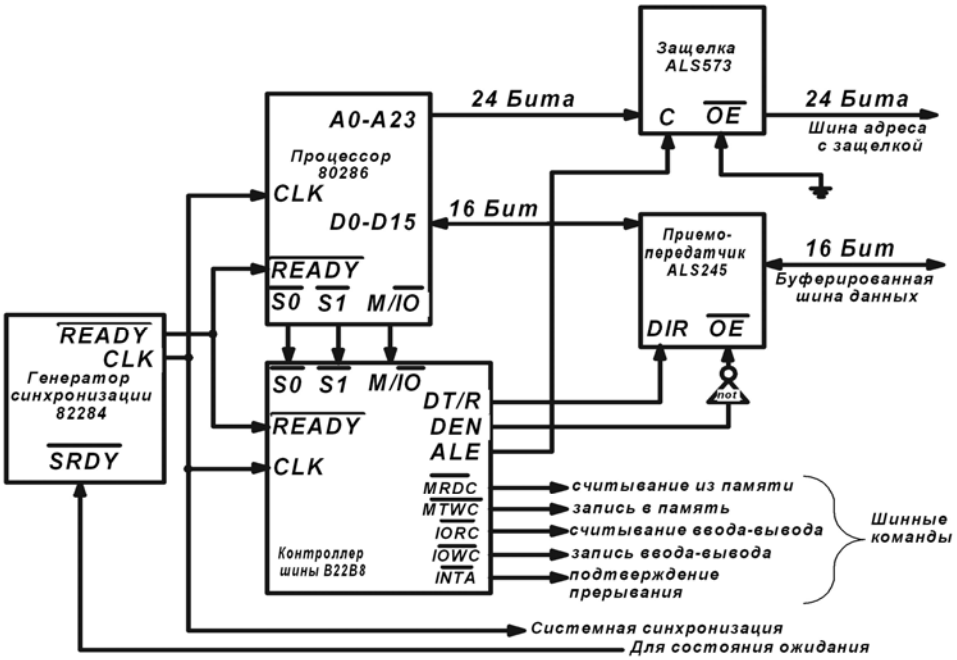


Рис. 2.10. Применение контроллера шины 82288

Контроллер шины формирует сигнал разрешения защелки адреса ALE для стробирования защелок на шине адреса в правильные моменты времени. Мы не будем пользоваться входами OE защелок, поэтому они заземляются: земля (0В) соответствует логическому значению FALSE (ложь), а V_{CC}(+5В) – логическому значению TRUE (*истина)*. (V_{CC} - постоянное напряжение от источника питания.)

Сигнал *разрешения данных D*EN контроллера шины разрешает приемопередатчики на шине данных, когда происходит передача данных. Сигнал *передачи/приема данных D*T/R показывает приемопередатчикам правильное направление передачи информации.

Теперь временная диаграмма становится довольно простой. Вот основные этапы операции считывания:

1. Фиксируется адрес.
2. Задается направление включения приемопередатчиков шины данных, разрешается работа приемопередатчиков и выдается соответствующая команда считывания (IORS или MRDC).
3. Теперь адресуемое устройство может посылать данные.
4. Снимается сигнал на командной линии и запрещается работа приемопередатчиков шины данных.

Основные операции записи также довольно просты:

1. Фиксируется адрес, задается направление включения приемопередатчиков и разрешается их работа.
2. Выдается соответствующая команда записи (IOWC или MWTC).
3. Адресуемое устройство может теперь принять данные.
4. Снимается сигнал на командной линии.
5. Запрещается работа приемопередатчиков.

Теперь, когда сформировалась система коммуникаций, можно перейти к рассмотрению процессора, основного устройства вычислительной системы (рис. 2.11).

Схема управления выборкой команд выполняет чтение команд из памяти и их дешифрацию. В первых микропроцессорах были невозможны одновременное выполнение предыдущей команды и выборка следующей команды, так как процессор не мог совмещать эти операции. Но уже в 16-разрядных процессорах появляется так называемый конвейер (очередь) команд, позволяющий выбирать несколько следующих команд, пока выполняется предыдущая. Два процесса идут параллельно, что ускоряет работу процессора. Конвейер представляет собой небольшую внутреннюю память процессора, в которую при малейшей возможности (при освобождении внешней шины) записывается несколько команд, следующих за исполняемой. Читаются эти команды процессором в том же порядке, что и записываются в конвейер (это память типа FIFO, First In – First Out, первый вошел – первый вышел). Правда, если выполняемая команда предполагает переход не на следующую ячейку памяти, а на удаленную (с меньшим или большим адресом), конвейер не помогает, и его приходится сбрасывать. Но такие команды встречаются в программах сравнительно редко.

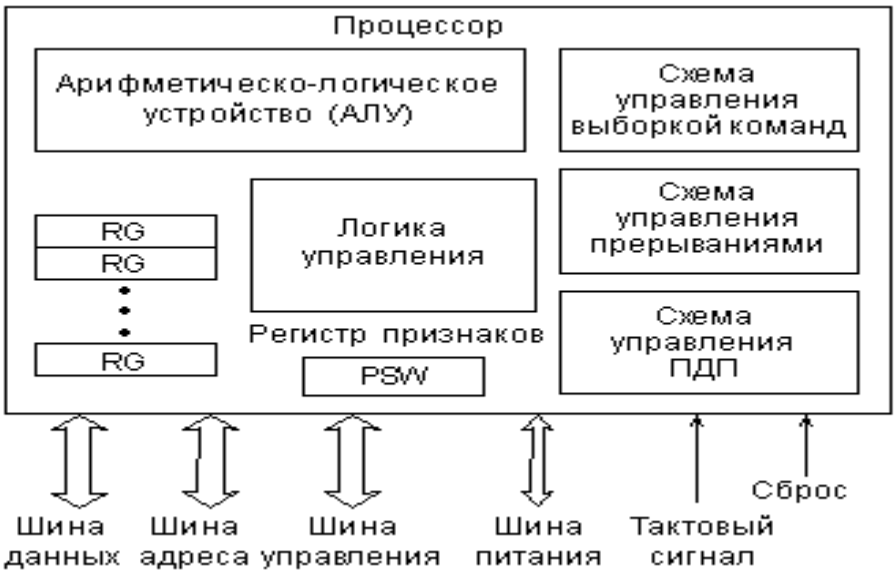


Рис. 2.11. Внутренняя структура микропроцессора

Развитием идеи конвейера стало использование внутренней кэш-памяти процессора, которая заполняется командами, пока процессор занят выполнением предыдущих команд. Чем больше объем кэш-памяти, тем меньше вероятность того, что ее содержимое придется сбросить при команде перехода. Понятно, что обрабатывать команды, находящиеся во внутренней памяти, процессор может гораздо быстрее, чем те, которые расположены во внешней памяти. В кэш-памяти могут храниться и данные, которые обрабатываются в данный момент, это также ускоряет работу. Для большего ускорения выборки команд в современных процессорах применяют совмещение выборки и дешифрации, одновременную дешифрацию нескольких команд, несколько параллельных конвейеров команд, предсказание команд переходов и некоторые другие методы.

Регистры процессора представляют собой по сути ячейки очень быстрой памяти и служат для временного хранения различных кодов: данных, адресов, служебных кодов. Операции с этими кодами выполняются предельно быстро, поэтому в общем случае чем больше внутренних регистров, тем лучше. Кроме того, на быстродействие процессора сильно влияет разрядность регистров. Именно разрядность регистров и АЛУ называется внутренней разрядностью процессора, которая может не совпадать с внешней разрядностью.

По отношению к назначению внутренних регистров существует два основных подхода. Первого придерживается, например, компания

„Intel”, которая каждому регистру отводит строго определенную функцию. С одной стороны, это упрощает организацию процессора и уменьшает время выполнения команды, но с другой — снижает гибкость, а иногда и замедляет работу программы. Например, некоторые арифметические операции и обмен с устройствами ввода/вывода проводятся только через один регистр — аккумулятор, в результате чего при выполнении некоторых процедур может потребоваться несколько дополнительных пересылок между регистрами.

Регистр признаков (регистр состояния) занимает особое место, хотя он также является внутренним регистром процессора. Содержащаяся в нем информация — это не данные, не адрес, а слово состояния процессора.

Схема управления прерываниями обрабатывает поступающий на процессор запрос прерывания, определяет адрес начала программы обработки прерывания (адрес вектора прерывания), обеспечивает переход к этой программе после выполнения текущей команды и сохранения в памяти (в стеке) текущего состояния регистров процессора. По окончании программы обработки прерывания процессор возвращается к прерванной программе с восстановленными из памяти (из стека) значениями внутренних регистров.

Схема управления прямым доступом к памяти служит для временного отключения процессора от внешних шин и приостановки работы процессора на время предоставления прямого доступа запросившему его устройству.

Логика управления организует взаимодействие всех узлов процессора, перенаправляет данные, синхронизирует работу процессора с внешними сигналами, а также реализует процедуры ввода и вывода информации.

Таким образом, в ходе работы процессора схема выборки команд выбирает последовательно команды из памяти, затем эти команды выполняются, причем в случае необходимости обработки данных подключается АЛУ. На входы АЛУ могут подаваться обрабатываемые данные из памяти или из внутренних регистров. Во внутренних регистрах хранятся также коды адресов обрабатываемых данных, расположенных в памяти. Результат обработки в АЛУ изменяет состояние регистра признаков и записывается во внутренний регистр или в память (как источник, так и приемник данных указываются в

составе кода команды). При необходимости информация может переписываться из памяти (или из устройства ввода/вывода) во внутренний регистр или из внутреннего регистра в память (или в устройство ввода/вывода).

Содержимое указателя (счетчика) команд изменяется следующим образом. В начале работы системы (при включении питания) в него заносится раз и навсегда установленное значение. Это первый адрес программы начального запуска. Затем после выборки из памяти каждой следующей команды значение указателя команд автоматически увеличивается (инкрементируется) на единицу (или на два в зависимости от формата команд и типа процессора). То есть следующая команда будет выбираться из следующего по порядку адреса памяти. При выполнении команд перехода, нарушающих последовательный перебор адресов памяти, в указатель команд принудительно записывается новое значение — новый адрес в памяти, начиная с которого адреса команд опять же будут перебираться последовательно.

Контрольные вопросы

1. Перечислите основные этапы операции считывания.
2. Опишите внутреннюю структуру микропроцессора.
3. В чем различие между памятью FIFO и LIFO?
4. Для чего необходима кеш-память?
5. Какой элемент процессора является ячейкой быстрой памяти?
6. Как изменяется содержимое счетчика команд?

Глава 3

ПАМЯТЬ

В настоящее время существует большое количество различных типов ЗУ, используемых в вычислительных системах. Эти устройства различаются принципом действия, логической, конструктивной и технологической реализацией. Большое количество существующих типов ЗУ обусловливает различия в структурной и логической организации (систем) памяти. Требуемые характеристики памяти достигаются не только за счет применения ЗУ с соответствующими характеристиками, но в значительной степени за счет особенностей ее структуры и алгоритмов функционирования. Память почти всегда является "узким местом", ограничивающим производительность вычислительной системы. Поэтому в ее организации используется ряд приемов, улучшающих временные характеристики памяти и, следовательно, повышающих производительность в целом.

КЛАССИФИКАЦИЯ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Для понимания принципов организации устройств и систем памяти рассмотрим один из возможных вариантов классификации ЗУ, представленный на рис. 3.1. В нем устройства памяти подразделяются по двум основным критериям: назначению (роли или месту в иерархии памяти) и принципу организации.

КЛАССИФИКАЦИЯ ЗУ ПО НАЗНАЧЕНИЮ

При разделении ЗУ по назначению можно рассматривать два вида: оперативная обработка информации и хранение информации во времени большем, чем один цикл процессора, внутренние и внешние ЗУ ЭВМ. Такое деление первоначально основывалось на различном конструктивном расположении их в ЭВМ.

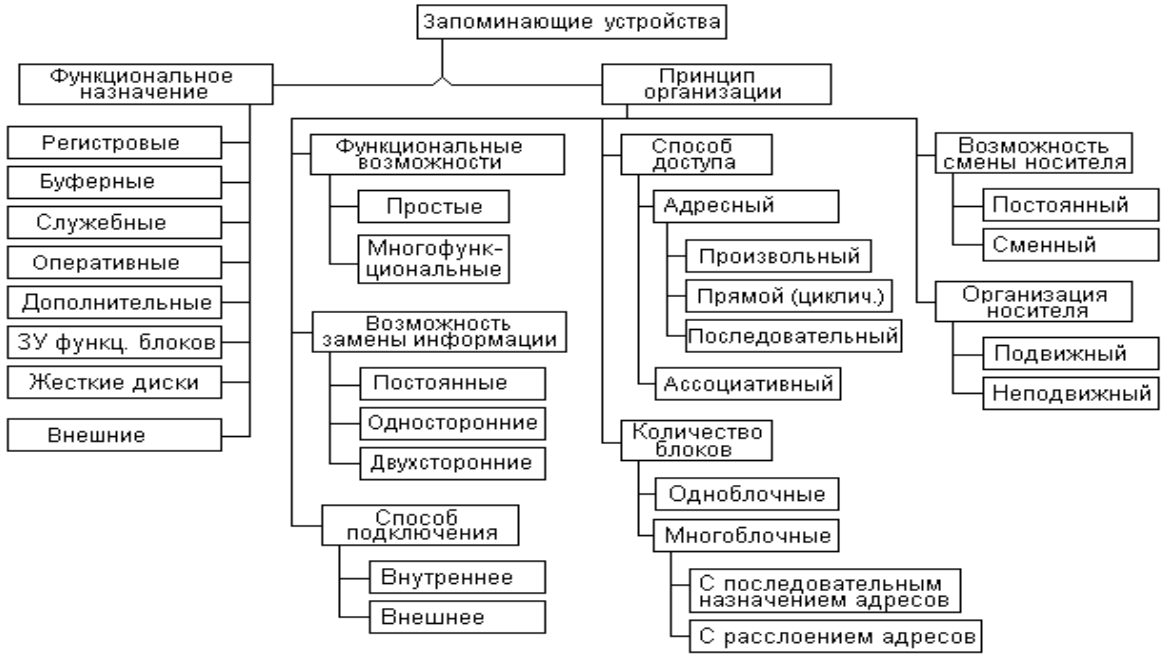


Рис. 3.1. Классификация запоминающих устройств

В настоящее время, например, накопители на жестких магнитных дисках, традиционно относимые к внешним ЗУ, конструктивно располагаются непосредственно в основном блоке компьютера. Поэтому разделение на внешние и внутренние ЗУ имеет в ряде случаев относительный, условный характер. Обычно к внутренним ЗУ относят устройства, непосредственно доступные процессору, а к внешним – такие, обмен информацией которых с процессором происходит через внутренние ЗУ.

Общий вид иерархии памяти ЭВМ представлен на рис. 3.2. На нем показаны различные типы ЗУ, причем поскольку рисунок обобщенный, то не все представленные на нем ЗУ обязательно входят в состав ЭВМ, а характер связей между устройствами может отличаться от показанного на рисунке.

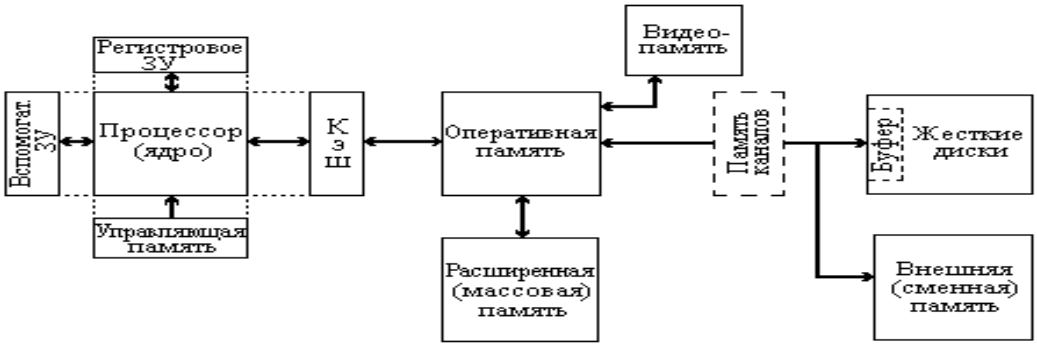


Рис. 3.2. Возможный состав системы памяти ЭВМ

Иерархия ЗУ:

1. Верхнее место в иерархии памяти занимают регистровые ЗУ, которые входят в состав процессора и часто рассматриваются не как самостоятельный блок ЗУ, а просто как набор регистров процессора. Такие ЗУ в большинстве случаев реализованы на том же кристалле, что и процессор, и предназначены для хранения небольшого количества информации, которая обрабатывается в текущий момент времени или часто используется процессором. Это позволяет сократить время выполнения программы за счет использования команд типа регистр-регистр и уменьшить частоту обменов информацией с более медленными ЗУ ЭВМ. Обращение к этим ЗУ производится непосредственно по командам процессора.

2. Следующую позицию в иерархии занимают буферные ЗУ. Их назначение состоит в сокращении времени передачи информации между процессором и более медленными уровнями памяти компьютера. Буферная память может устанавливаться на различных уровнях, но здесь речь идет именно об указанном ее местоположении. Ранее такие буферные ЗУ в отечественной литературе называли сверхоперативными, сейчас это название практически полностью вытеснил термин "кэш-память", или просто кэш.

Принцип использования буферной памяти во всех случаях сводится к одному и тому же. Буфер представляет собой более быстрое (а значит, и более дорогое), но менее емкое ЗУ, чем то, для ускорения работы которого он предназначен. При этом в буфере размещается только та часть информации из более медленного ЗУ, которая используется в настоящий момент. Если доля h обращений к памяти со стороны процессора, удовлетворяемых непосредственно буфером (кэшем), высока (0,9 и более), то среднее время для всех обращений оказывается близким ко времени обращения к кэшу, а не к более медленному ЗУ.

Пусть двухуровневая память состоит из кэш и оперативной памяти, как показано на рис. 3.3. И пусть, например, время обращения к кэшу tc = 1 нс (10 - 9 с), время tm обращения к более медленной памяти в десять раз больше – tm = 10 нс, а доля обращений, удовлетворяемых кэшем, h = 0,95. Тогда среднее время обращения к такой двухуровневой памяти $T_{ср}$ составит $T_{ср} = 1 \cdot 0.95 + 10(1 – 0.95) = 1.45$ нс, т.е. всего на 45 %, больше, времени обращения к кэшу. Значение h за-

висит от размера кэша и характера выполняемых программ и иногда называется отношением успехов или попаданий (hit ratio).

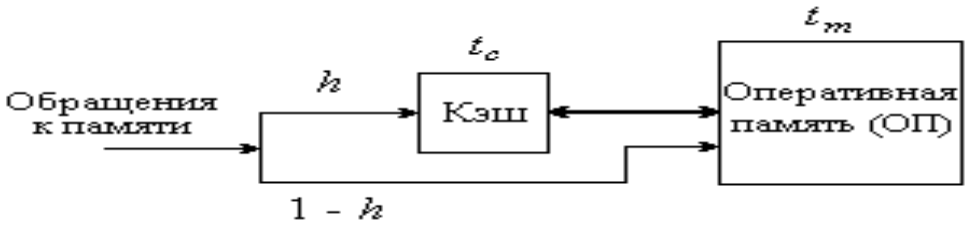


Рис. 3.3. К расчету среднего времени обращения:
tc – время обращения к кэш-памят; tm – время обращения к ОП;
h – доля обращений, обслуживаемых кэш-памятью;
1 – h – доля обращений, обслуживаемых ОП

Размеры кэш-памяти существенно изменяются с развитием технологий. Так, если в первых ЭВМ, где была установлена кэш-память, во второй половине 1960-х годов (большие ЭВМ семейства IBM-360) ее емкость составляла всего от 8 до 16 Кбайт, то уже во второй половине 1990-х годов емкость кэша рядовых персональных ЭВМ составляла 512 Кбайт. Причем сама кэш-память может состоять из двух (а в серверных системах даже трех) уровней: первого (L1) и второго (L2), также различающихся своей емкостью и временем обращения.

Конструктивно кэш уровня L1 входит в состав процессора (поэтому его иногда называют внутренним). Кэш уровня L2 либо также входит в микросхему процессора, либо может быть реализован в виде отдельной памяти. Как правило, на параметры быстродействия процессора большее влияние оказывают характеристики кэш-памяти первого уровня.

Время обращения к кэш-памяти, которая обычно работает на частоте процессора, составляет от десятых долей до единиц наносекунд, т.е. не превышает длительности одного цикла процессора.

3. Еще одним (внутренним) уровнем памяти являются служебные ЗУ. Они могут иметь различное назначение.

Одним из примеров таких устройств являются ЗУ микропрограмм, которые иногда называют управляющей памятью, другим – вспомогательные ЗУ, используемые для управления многоуровневой памятью. В управляющей памяти, использующейся в ЭВМ с микропрограммным управлением, хранятся микропрограммы выполнения команд процессора, а также различных служебных операций.

Вспомогательные ЗУ для управления памятью (например, теговая память, используемая для управления кэш-памятью, буфер переадресации TLB – translation location buffer) представляют собой различные таблицы, используемые для быстрого поиска информации в разных ступенях памяти, отображения ее свойств, очередности перемещения между ступенями и пр.

Емкости и времена обращения к таким ЗУ зависят от их назначения. Обычно это небольшие (до нескольких Кбайт), но быстродействующие ЗУ. Специфика назначения предполагает недоступность их командам процессора.

4. Следующий уровень иерархии памяти – оперативная память. Оперативное ЗУ (ОЗУ) является основным запоминающим устройством ЭВМ, в котором хранятся выполняемые в настоящий момент процессором программы и обрабатываемые данные, резидентные программы, модули операционной системы и т.п. Название оперативной памяти также несколько изменялось во времени. В некоторых семействах ЭВМ ее называли основной памятью, основной оперативной памятью и пр. В англоязычной литературе также используется термин RAM (random access memory), означающий память с произвольным доступом.

Эта память используется в качестве основного запоминающего устройства ЭВМ для хранения программ, выполняемых или готовых к выполнению в текущий момент времени, и относящихся к ним данных. В оперативной памяти располагаются и компоненты операционной системы, необходимые для ее нормальной работы. Информация, находящаяся в ОЗУ, непосредственно доступна командам процессора при условии соблюдения требований защиты.

Оперативная память реализуется на полупроводниках (интегральных схемах), стандартные объемы ее составляют (в начале 2000-х годов) сотни мегабайт – единицы гигабайт, а времена обращения – единицы – десятки наносекунд.

5. Еще одним уровнем иерархии ЗУ может являться дополнительная память, которую иногда называли расширенной, или массовой. Первоначально (1970-е годы) эта ступень использовалась для наращивания емкости оперативной памяти до величины, соответствующей адресному пространству (например, 24-битного адреса) команд, с помощью подключения более дешевого и емкого, чем ОЗУ, запоминающего устройства.

Это могла быть ферритовая память или даже память на магнитных дисках. Конечно, она была более медленной, а хранимая в ней информация сперва передавалась в оперативную память и только оттуда попадала в процессор. При записи путь был обратный.

Затем, в ранних моделях ПЭВМ, дополнительная память также использовалась для наращивания емкости ОЗУ и представляла собой отдельную плату с микросхемами памяти. А еще позже термин „дополнительная память” (extended или expanded memory) стал обозначать область оперативного ЗУ с адресами выше одного мегабайта. Конечно, этот термин применим только к IBM PC совместимым ПЭВМ.

6. В состав памяти ЭВМ входят также ЗУ, принадлежащие отдельным функциональным блокам компьютера. Формально эти устройства непосредственно не обслуживают основные потоки данных и команд, проходящие через процессор. Их назначение обычно сводится к буферизации данных, извлекаемых из каких-либо устройств и поступающих в них.

Типичным примером такой памяти является видеопамять графического адаптера, которая используется в качестве буферной памяти для снижения нагрузки на основную память и системную шину процессора.

Другими примерами таких устройств могут служить буферная память контроллеров жестких дисков, а также память, использовавшаяся в каналах (процессорах) ввода-вывода для организации одновременной работы нескольких внешних устройств.

Емкости и быстродействие этих видов памяти зависят от конкретного функционального назначения обслуживаемых ими устройств. Для видеопамяти, например, объем может достигать величин, сравнимых с оперативными ЗУ, а быстродействие – даже превосходить быстродействие последних.

7. Следующей ступенью памяти, ставшей фактически стандартом для любых ЭВМ, являются жесткие диски. В этих ЗУ хранится практически вся информация, которая используется более или менее активно, начиная от операционной системы и основных прикладных программ и кончая редко используемыми пакетами и справочными данными.

Емкость этой ступени памяти, которая может включать в свой состав до десятков дисков, обеспечивая хранение очень большого ко-

личества данных, зависит от области применения ЭВМ. Типовая емкость жесткого диска, составляющая на начало 2000-х годов десятки гигабайт, удваивается примерно каждые полтора года.

Со временами обращения дело обстоит несколько иначе: компоненты этого времени, обусловленные перемещением блока головок чтения-записи, уменьшаются сравнительно медленно (примерно вдвое за 10 лет). Компонента, обусловленная временем подвода сектора и зависящая от скорости вращения шпинделя диска, также уменьшается с ростом этой скорости примерно такими же темпами. А скорость передачи данных растет значительно быстрее, что связано с увеличением плотности записи информации на диски.

8. Все остальные запоминающие устройства можно объединить с точки зрения функционального назначения в одну общую группу, охарактеризовав ее как группу внешних ЗУ. Под словом “внешние” следует подразумевать то, что информация, хранимая в этих ЗУ, в общем случае расположена на носителях, не являющихся частью собственно ЭВМ. Под это определение подпадают гибкие диски, компакт-диски, накопители на сменных магнитных дисках и магнитооптические диски, твердотельные (флэш) диски и флэш-карты, стримеры, внешние винчестеры и др. Естественно, что параметры этих устройств достаточно различны. Функциональное назначение их обычно сводится либо к архивному хранению информации, либо к переносу ее одного компьютера к другому.

Некоторые сомнения в принадлежности к данной категории могут вызвать сменные диски, устанавливаемые в салазки (rack). Такие диски, действительно, лучше отнести к предыдущей (седьмой) группе.

Особенности организации ЗУ определяются, в первую очередь, используемыми технологиями, логикой их функционирования, а также некоторыми другими факторами. Эти особенности и соответствующие разновидности ЗУ перечисляются ниже.

КЛАССИФИКАЦИЯ ЗУ ПО ПРИНЦИПУ ОРГАНИЗАЦИИ

1. По функциональным возможностям ЗУ можно разделять:
   - на простые, допускающие только хранение информации;
   - многофункциональные, которые позволяют не только хранить, но и перерабатывать хранимую информацию без участия процессора непосредственно в самих ЗУ [2].

Подход, используемый во второй группе ЗУ, в принципе позволяет создать производительные системы с параллельной обработкой данных. В частности, похожие подходы используются в различных частях видеотракта компьютера.

2. По возможности изменения информации различают ЗУ:

- постоянные (или с однократной записью);
- полупостоянные (с перезаписью или перепрограммируемые);
- приборы с зарядовой связью.

В постоянных ЗУ (ПЗУ) информация заносится либо при изготовлении, либо посредством записи (или, как иначе называют эту процедуру, программирования или прожига), которая может быть выполнена только однократно. В ходе такой записи изменяется сам носитель информации, например, пережигаются проводники в микросхемах ПЗУ или формируются лунки в отражающем слое CD-ROM.

Полупостоянными называют ЗУ, которые имеют существенно различные времена записи и считывания информации. Наиболее распространенными типами таких ЗУ являются перепрограммируемые постоянные ЗУ или компакт-диски с перезаписью – CD-RW. Время записи в устройствах этих типов значительно превышает время считывания информации.

Двусторонние ЗУ имеют близкие значения времен чтения и записи. Типичными представителями таких ЗУ являются оперативные ЗУ и ЗУ на жестких дисках.

3. По способу доступа различают ЗУ:

- с адресным доступом;
- ассоциативным доступом.

При адресном доступе для записи или чтения место расположения информации в ЗУ определяется ее адресом. Логически адрес может иметь различные структуры. Например, в оперативных ЗУ адрес представляет собой двоичный код, одна часть разрядов которого указывает строку матрицы элементов памяти, а другая – столбец этой матрицы. На пересечении заданных строки и столбца находится искомая информация. В ЗУ на магнитных дисках адрес может представлять собой либо комбинацию номеров цилиндра, головки и сектора (так называемая CHS-геометрия), либо логический номер сектора (LBA-адресация). Возможны и иные варианты.

В любом случае заданный адрес отрабатывается схемами доступа ЗУ (дешифратором, блоком позиционирования головок и т.п.) таким образом, что в операции участвует соответствующая адресу область матрицы элементов памяти, запоминающей среды или носителя информации.

При этом, в зависимости от того, как именно срабатывает механизм доступа, различают следующие виды адресного доступа:

- произвольный;
- прямой (циклический);
- последовательный.

Термин “память с произвольным доступом” (random access memory – RAM) применяют к ЗУ, в которых выбор места хранения информации производится непосредственным подключением входов и выходов элементов памяти (через буферы, усилители и логические элементы) к входным и выходным шинам ЗУ. Это наиболее быстрый вид адресного доступа, применяемый в оперативных ЗУ и кэш-памяти.

При прямом (циклическом) доступе непосредственной коммутации связей оказывается недостаточно. В таких ЗУ обычно происходит еще и перемещение данных относительно механизма чтения/записи, механизма чтения/записи относительно данных или и то и другое. Физически это может быть как механическое перемещение, например, в жестких дисках, перемещение областей намагниченности, как в ЗУ на магнитных доменах, перенос зарядов и др.

С логической точки зрения такие ЗУ можно сопоставить набору сдвигающих регистров, информация в которых сдвигается циклически и может вводиться в регистр или выводиться из него только в одном из разрядов. Термины “циклический” и “прямой” доступ близки по содержанию, хотя термин “прямой доступ” имеет более широкий смысл.

Последовательный доступ характерен для ЗУ, использующих в качестве носителя информации (запоминающей среды) магнитную ленту, например для стримеров. В таких ЗУ для доступа к блоку данных необходимо переместить носитель так, чтобы участок, на котором располагается требуемый блок данных, оказался под блоком головок чтения/записи.

Кроме того, при всех формах адресного доступа адресуемым элементом может быть не только байт или слово (как в оперативной

памяти и кэш-памяти), но целый блок данных. Это обычно связано либо с конструктивными особенностями ЗУ, либо с большим временем доступа.

При ассоциативном доступе место хранения информации при чтении и записи определяется не адресом, а значением некоторого ключа поиска. Каждое записанное и хранимое в ассоциативной памяти слово имеет поле ключа. Значение этого ключа сравнивается со значением ключа поиска при чтении данных из памяти. В случае совпадения сравниваемых значений информация считывается из памяти.

Ассоциативная память эффективна для решения задач, связанных с поиском данных. Однако ее использование ограничено в силу сравнительно высокой ее сложности.

Действительно, с аппаратной точки зрения сам поиск может быть организован по-разному: последовательно по разрядам ключевых полей или параллельно по всем ключам во всем массиве памяти. Второй способ, конечно, более быстрый, но требует соответствующей организации (ключевой части) памяти, которая должна иметь для этого в ключевой части каждого хранимого слова схемы сравнения. Именно поэтому такая память существенно более дорогая, чем оперативная и используется в основном для решения задач, требующих быстрого поиска в небольших объемах информации.

Одним из частых применений ассоциативной памяти является быстрое преобразование логических (линейных) адресов данных в физические (т.е. адреса ячеек памяти), выполняемое, например, так называемым буфером трансляции адресов. Другой близкой задачей является определение того, имеется ли требуемая информация в верхних уровнях ЗУ или необходима ее подкачка из более медленных ЗУ.

4. По организации носителя различают ЗУ:

- с неподвижным носителем;
- с подвижным носителем.

В первых из них носитель механически неподвижен в процессе чтения и записи информации, что имеет место, например, в оперативных и кэш ЗУ, твердотельных дисках, ЗУ с переносом зарядов и др.

Для ЗУ второй группы чтение и запись информации сопровождаются механическим перемещением носителя, что обычно имеет место в различных ЗУ с магнитной записью, например в жестких и гибких дисках.

Однако возможны и иные варианты. Например, фирмой IBM разрабатывается ЗУ с механическим перемещением записывающих и считывающих элементов (микроигл) и неподвижным носителем информации (пластиковой пленкой).

5. По возможности смены носителя ЗУ могут быть:

- с постоянным носителем;
- со сменным носителем.

В ЗУ первого вида носитель является частью самого устройства и не может быть извлечен из него в процессе нормального функционирования (оперативные ЗУ, жесткие диски).

В ЗУ второй группы носитель не является собственной частью устройства и может устанавливаться в ЗУ и извлекаться из него в процессе работы (гибкие диски, CD-ROM-дисководы, карты памяти, магнито-оптические диски).

6. По способу подключения к системе ЗУ делятся:

- на внутренние (стационарные);
- внешние (съемные).

В первом случае ЗУ, как правило, является обязательным компонентом вычислительной системы, устанавливается в корпусе системы (например, оперативная память) или интегрируется с другими ее компонентами (например, кэш-память).

Во втором случае устройство подключается к системе дополнительно и представляет собой отдельный блок. Подключение (и отключение) таких ЗУ в зависимости от особенности их реализации может производиться как при выключенной системе – так называемое “холодное подключение”, так и в работающей системе – “горячее подключение”.

Последний вариант в серверных системах предусматривают и для стационарных ЗУ (жестких дисков).

7. По количеству блоков, образующих модуль или ступень памяти, можно различать ЗУ:

- одноблочные;
- многоблочные.

Такое разделение может представлять интерес в том случае, когда в многоблочное ЗУ входят блоки (или банки памяти), допускающие возможность параллельной работы. В этом случае за счет одновременной работы блоков можно повысить общую производительность модуля (ступени) ЗУ, иначе называемую его пропускной спо-

собностью и измеряемую количеством информации, которое модуль может записать или считать в единицу времени.

Но возможность одновременной работы блоков еще не означает, что они именно так и будут работать. Чтобы это произошло, необходимо обращения системы к памяти более или менее равномерно распределять по различным блокам. Достичь этого можно различными способами, например запустить параллельные задачи или процессы (threads), работающие с разными блоками, либо разместить информацию, относящуюся к одному процессу, в разных блоках.

Однако, поскольку параллельные процессы в действительности выполняются параллельно только в многопроцессорных системах (в крайнем случае, в гиперпоточных архитектурах), то часто используют второй путь, прибегая к так называемому чередованию (interleave) адресов между блоками, т.е. последовательные адреса или группы адресов адресного пространства назначают в различные блоки памяти так, как это показано на рис. 3.4, *б*. На этом рисунке представлена память, состоящая из двух блоков, но на практике известны системы, допускающие расслоение по шестнадцати блокам.

Ясно, что в случае такого назначения адресов при выполнении какой-либо программы обращения к памяти будут распределяться по блокам достаточно равномерно. А при обмене блоком данных с другой ступенью памяти обращения по последовательным адресам тем более будут попадать в различные блоки памяти.

Рассматривая расслоение адресов, можно отметить его аналогию с некоторыми режимами работы RAID-контроллеров.

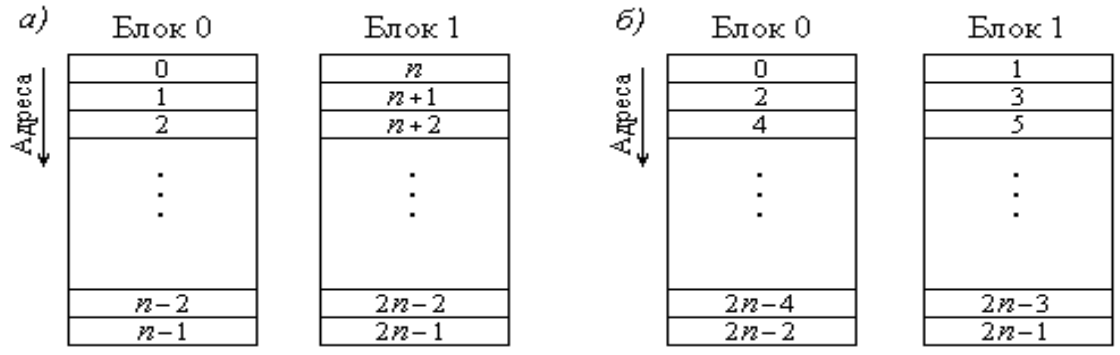


Рис. 3.4. Распределение адресов адресного пространства памяти по блокам: а – последовательное; б – с расслоением по блокам

Конечно, за пределами приведенной классификации остались такие довольно представительные признаки, как физические принципы реализации, уровень потребляемой мощности, радиационная устойчивость и некоторые другие, которые в определенных случаях могут иметь немаловажное значение.

Контрольные вопросы

1. По каким критериям классифицируются устройства памяти?
2. В чем назначение буферной ЗУ?
3. Для чего предназначены регистровые ЗУ?
4. Чем характеризуется память RAM?

Организация ЗУ с произвольным доступом. Запоминающие устройства характеризуются рядом параметров, определяющих возможные области применения различных типов таких устройств. К основным параметрам, по которым производится наиболее общая оценка ЗУ, относятся их информационная емкость Е, время обращения Т и стоимость С.

Состав и структура микросхем оперативных ЗУ в процессе совершенствования технологий их изготовления подверглись определенным изменениям.

Первые полупроводниковые оперативные ЗУ строились на схемах малой и средней степени интеграции и включали в себя несколько различных типов микросхем: собственно матрицы элементов памяти, усилители чтения-записи, дешифраторы и при необходимости регистры (адреса и данных).

Позднее с появлением больших интегральных схем (БИС) и повышением частоты их работы, использование раздельных типов микросхем перестало быть оправданным по следующим причинам. Во-первых, количество элементов памяти в матрицах возросло настолько, что число выводов, требующееся для выбора элемента памяти и равное сумме количества строк и количества столбцов матрицы, стало очень большим (несколько тысяч). Во-вторых, длина соединений между микросхемами больше, чем длина соединений внутри микросхемы, что увеличивает время прохождения сигнала и реактивные со-

ставляющие (емкость и индуктивность, или перекрестные помехи), а следовательно, уменьшает быстродействие памяти.

Поэтому микросхемы памяти стали включать в себя не только элементы памяти, но и всю остальную электронику управления: дешифраторы, усилители, буферные регистры, схемы управления. Такой состав БИС памяти приводил к известной аппаратной избыточности строящихся на их основе модулей памяти.

Действительно, первые БИС памяти имели логическую организацию вида N одноразрядных слов, или N × 1, где N - количество адресов (одноразрядных слов) микросхемы. Следовательно, каждый разряд модуля памяти, построенного на таких микросхемах, включал свои собственные дешифраторы, буферные регистры и схемы управления, одного комплекта которых при традиционной организации было достаточно для целого модуля. Однако такое дублирование оправдывалось достигаемыми характеристиками памяти, а его стоимость не была чрезмерной (электроника обрамления составляла не более 5 – 15 % общей площади кристалла микросхемы).

Впоследствии разрядность хранимых в микросхеме слов была увеличена и составляет на сегодня от 4 до 16 разрядов, т.е. N × 4, N × 8, N × 16. Понятно, что относительная доля избыточных схем обрамления при этом падает.

На функциональных схемах микросхема памяти изображается обычным прямоугольником с левым и правым полями, как показано на рис. 3.5.

Микросхема имеет три группы входов: адресные входы, вход(ы) данных и управляющие входы.

Количество адресных входов (A0÷Ak) определяется емкостью и организацией микросхемы памяти, а также способом подачи адреса. Нетрудно видеть, что емкость микросхемы ECx, равная произведению количества адресов (слов) N на разрядность хранимых слов *n*, не определяет однозначно требуемое число адресных входов. Для адресации любого из N слов требуется адрес разрядностью log2N. Например, для адресации микросхемы емкостью ECx = 128 Мбит, имеющей организацию 16M × 8 (адресов × бит), достаточно log216M = = log2 (24 · 220) = 24 разряда.

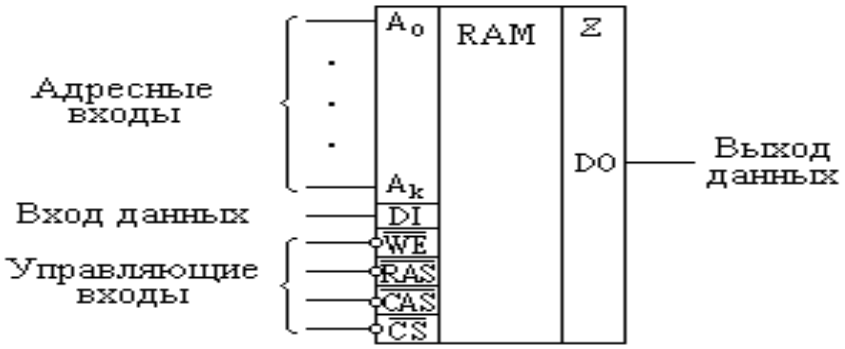


Рис. 3.5. Микросхема памяти

Способ подачи адреса также оказывает влияние на количество адресных входов микросхемы. Так, распространенный в динамических оперативных ЗУ прием мультиплексирования адресных входов, состоящий в поочередной подаче на одни и те же адресные входы сначала старшей части (половины) адреса - адреса строки (Row Address), а затем - младшей части - адреса столбца (Column Address), позволяет уменьшить вдвое количество требующихся адресных входов. Конечно, это несколько увеличивает время обращения к памяти, но оказывается экономически (да и схемотехнически) оправданным.

В статических ЗУ все разряды адреса подаются на адресные входы одновременно.

Количество входов данных (DI - Data Input) равно разрядности хранимых слов. Количество выходов данных (DO - Data Output) также равно разрядности хранимых слов. Однако во многих случаях входы и выходы данных объединяются, что позволяет уменьшить вдвое количество выводов данных у микросхем памяти, а также упростить их подключение к шинам данных.

Для этого выходы микросхем памяти (или объединенные входы/выходы) обычно имеют специальный выходной каскад, позволяющий подключать к одной шине выходы нескольких микросхем без использования дополнительных сборок ИЛИ. Есть два варианта организации таких выходов: выход с тремя устойчивыми состояниями (или z-выход) и выход с открытым коллектором. Тип выхода отмечается специальным значком в верхней части правого поля изображения микросхемы. На рис. 3.5. показан z-выход.

Выход данных, реализованный по схеме с открытым коллектором, как правило, инверсный.

Управляющие входы могут заметно различаться как по назначению, так и по обозначениям для разных типов микросхем памяти.

Во всех случаях присутствует вход управления режимом обращения: чтение или запись. Частым его обозначением является WE# (Write Enable - разрешение записи). Вход этот обычно инверсный (это и обозначает символ #), т.е. режим записи включается при нулевом значении сигнала на данном входе, а при единице на входе производится чтение.

Другим общим сигналом, имеющимся почти во всех микросхемах, является сигнал выбора микросхемы - CS# (Chip Select). Этот вход также обычно является инверсным и при единичном значении на нем микросхема переходит во "выключенное" состояние (выход данных микросхемы переходит в состояние высокого выходного сопротивления, если он является z-выходом, или в состояние "1", если это инверсный выход с открытым коллектором). При нулевом значении сигнала на входе CS# микросхема находится в активном состоянии.

В динамических ОЗУ при мультиплексировании адресных входов используются два управляющих входа сигналов строба: RAS# (Row Address Strobe - строб адреса строки) и CAS# (Column Address Strobe - строб адреса столбца, или колонки). Сигналы на этих входах переводятся в активное состояние (в "0") в тот момент, когда на адресных входах установлен адрес строки или адрес столбца соответственно.

Структурная схема БИС динамического ОЗУ показана на рис. 3.6.

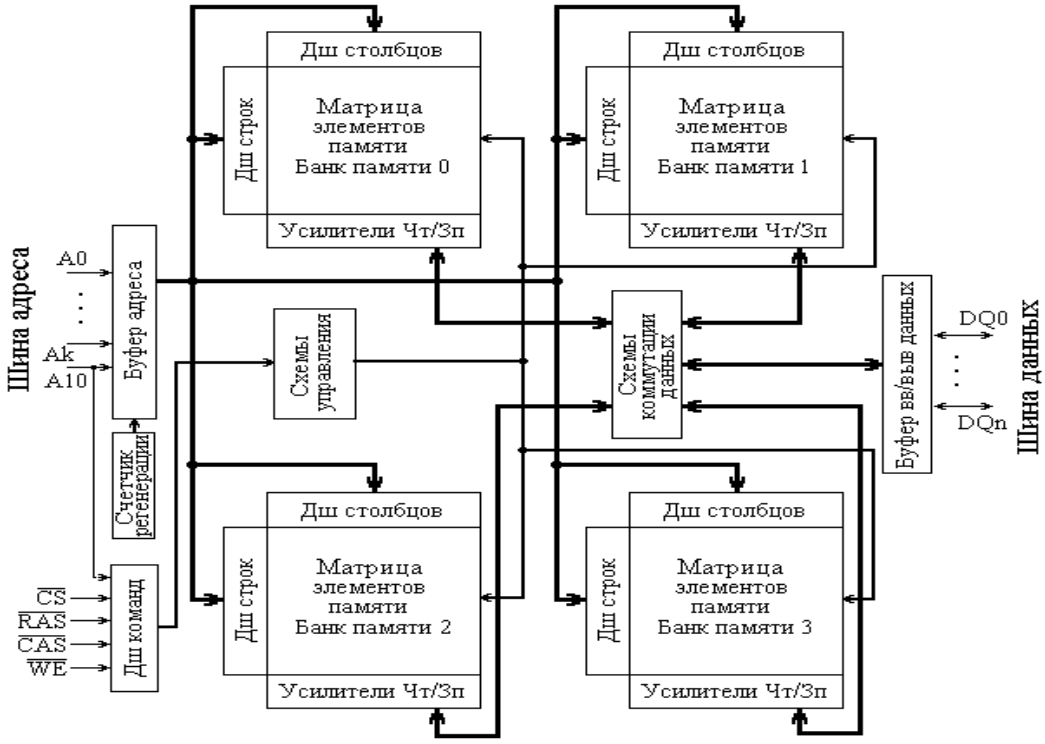


Рис. 3.6. Структурная схема динамического ОЗУ

Основными ее компонентами являются четыре банка памяти, представляющие собой матрицы элементов памяти с дешифраторами строк и столбцов и усилителями чтения-записи. Кроме собственно банков памяти в состав ОЗУ входят:

- буфер адреса, фиксирующий адреса строки и столбца;
- счетчик регенерации, формирующий адрес строки, в которой должна выполняться очередная регенерация;
- дешифратор команд, определяющий, какое действие (команду) должна выполнить микросхема в соответствии с поданными управляющими сигналами (и сигналомA10);
- схемы управления, формирующие управляющие сигналы для остальных узлов микросхемы;
- схемы коммутации данных, передающие читаемые или записываемые данные из/в банки памяти;
- буфер ввода/вывода данных, обеспечивающий связь микросхемы памяти с шиной данных.

Адресный сигнал A10 выделен среди других адресных линий, так как он имеет специальное назначение: при подаче адреса столбца этот сигнал указывает на особенности выполнения последующего (пакетного) чтения или записи, задавая (при единичном значении) режим так называемого автоматического подзаряда банка памяти.

Контрольные вопросы

1. От чего зависит количество входов данных DI?
2. Чем определяется количество адресных входов?
3. Какую функцию выполняет сигнал CS?
4. Что входит в состав ОЗУ?

ПРИНЦИПЫ РАБОТЫ ЗУ

В статических ЗУ (Static Random Access Memory – SRAM) в качестве элемента памяти используется триггер, что, конечно, сложнее, чем конденсатор с транзисторным ключом динамического ЗУ. Поэтому статические ЗУ обладают меньшей плотностью хранения информации.

Однако триггер со времен первых компьютеров был и остается самым быстродействующим элементом памяти. Поэтому статическая память позволяет достичь наибольшего быстродействия, обеспечивая время доступа в единицы и даже десятые доли наносекунд, что и обусловливает ее использование в ЭВМ, главным образом, в высших ступенях памяти – кэш-памяти всех уровней.

Главными недостатками статической памяти являются ее относительно высокая стоимость и высокое энергопотребление.

Конечно, в зависимости от используемой технологии, память будет обладать различным сочетанием параметров быстродействия и потребляемой мощности. Например, статическая память, изготовленная по КМОП-технологии (CMOS память), имеет низкую скорость доступа, со временем порядка 100 нс, но зато отличается очень малым энергопотреблением. В ПЭВМ такую память применяют для хранения конфигурационной информации компьютера при выключенном напряжении сети (в этой же микросхеме размещают и часы, отсчитывающие реальное время). Питание такой памяти осуществляется от небольшой батарейки, которая может служить несколько лет.

Основными разновидностями статической памяти (SRAM) с точки зрения организации ее функционирования являются асинхронная (Asynchronous), синхронная пакетная (Synchronous Burst) и синхронная конвейерно-пакетная (Pipeline Burst) память.

Первой появилась асинхронная память (рис. 3.7), интерфейс этой памяти включает шины данных, адреса и управления. В состав сигналов последней входят:

CS# (Chip Select) – сигнал выбора микросхемы;

WE# (Write Enable) – сигнал разрешения записи;

OE# (Output Enable) – сигнал включения выходов для выдачи данных.

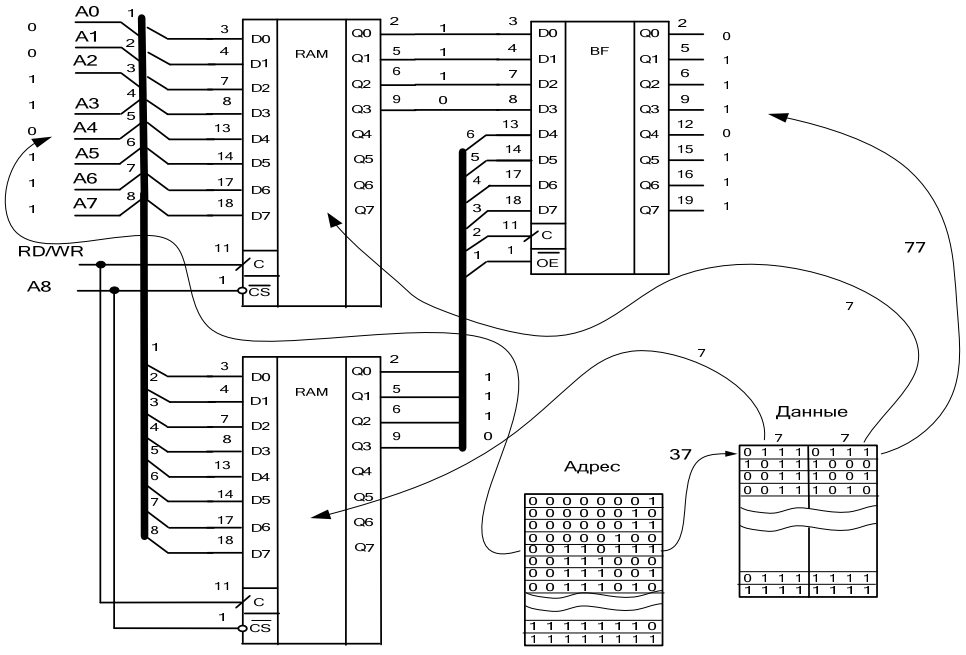


Рис. 3.7. Асинхронная память

Все сигналы управления инверсные, т.е. их активный (вызывающий соответствующее действие) уровень низкий. При единичном значении сигнала OE# выход микросхемы переходит в состояние высокого выходного сопротивления.

Временные диаграммы циклов чтения и записи приведены на рис. 3.8 и не требуют особых пояснений. Цикл записи может быть организован и несколько иначе, чем показано на 3.8, *б*), в случае удержания во время цикла высокого уровня сигнала OE#.

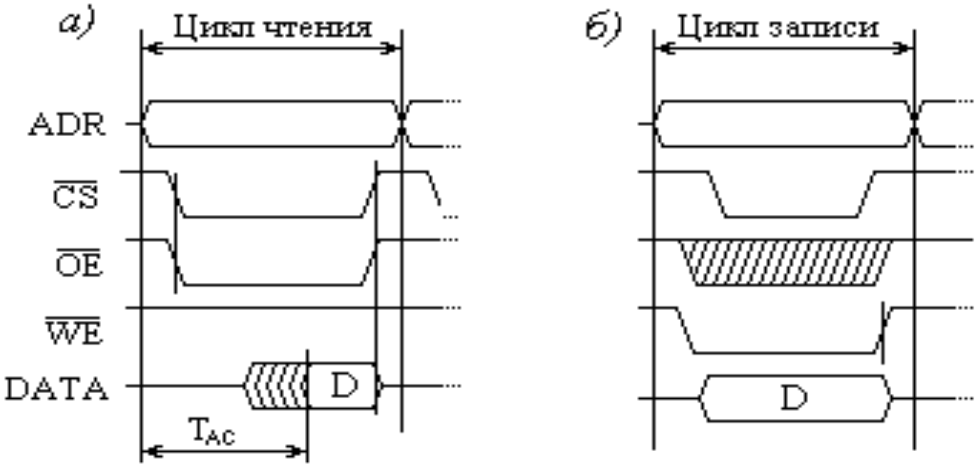


Рис. 3.8. Временные диаграммы циклов чтения и записи

Время доступа TAC у типовых микросхем составляет порядка 10 нс. Поэтому реально такие микросхемы могут работать на частотах, близких к частоте системной шины, только если эти частоты не превышают 66 МГц.

Несколько позже появилась синхронная пакетная статическая память (SBSRAM), ориентированная на выполнение пакетного обмена информацией, который характерен для кэш-памяти. Эта память включает в себя внутренний счетчик адреса, предназначенный для перебора адресов пакета, и использует сигналы синхронизации CLK, как и синхронная DRAM-память.

Для организации пакетного обмена, помимо имеющихся у асинхронной памяти управляющих сигналов CS#, OE# и WE#, в синхронную память также введены сигналы ADSP# (Address Status of Processor) и CADS# (Cache Address Strobe), сопровождающие передачу адреса нового пакета, а также сигнал ADV# (Advance) продвижения на следующий адрес пакета. Пакетный цикл всегда предусматри-

вает передачу четырех элементов, так как внутренний счетчик имеет всего 2 бита, причем перебор адресов в пределах пакета может быть последовательным или с расслоением (чередованием) по банкам (при использовании процессоров семейства x86).

Временные диаграммы пакетных циклов чтения и записи приведены на рис. 3.9. Обращения к синхронной памяти могут быть и одиночными. В этом случае низкому уровню сигнала ADSP#, указывающему на передачу адреса, соответствует высокий уровень сигнала CADS#, а не низкий, как при пакетном цикле. Параметр TQK характеризует время задержки данных относительно синхронизирующего сигнала.

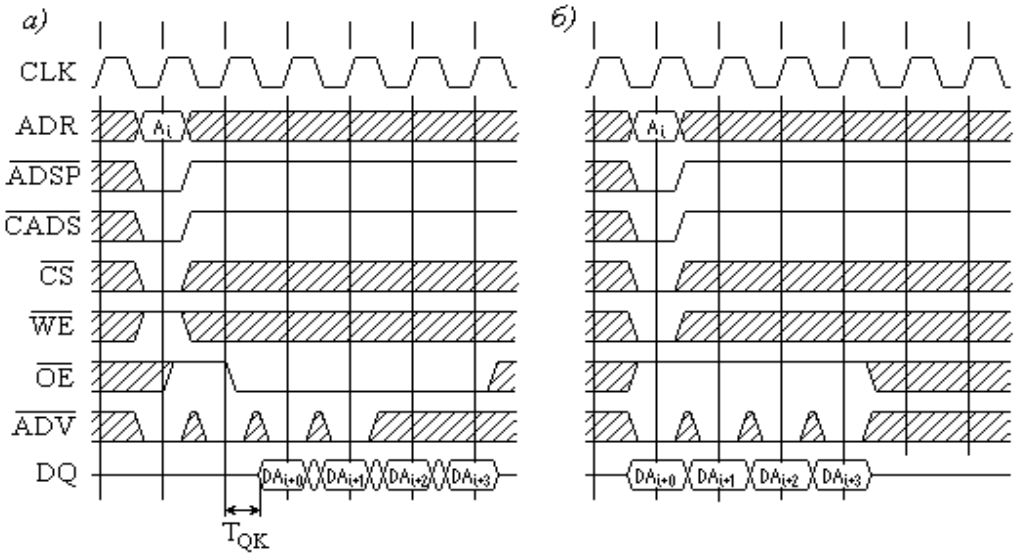


Рис. 3.9. Пакетные циклы чтения: а – записи; б – синхронной пакетной статической памяти (SBSRAM)

Следующим шагом в развитии статической памяти явилась конвейерно-пакетная память PBSRAM, обеспечивающая более высокое быстродействие, чем SBSRAM. В нее были введены дополнительные внутренние буферные регистры данных (здесь можно провести аналогию с EDO DRAM памятью) адреса, а в ряде модификаций предусмотрена возможность передачи данных на двойной скорости по переднему и заднему фронтам синхросигнала и используются сдвоенные внутренние тракты записи и чтения. Это позволило получить время обращения порядка 2 - 3 нс и обеспечить передачу данных пакета без задержек на частотах шины более 400 МГц.

Внутренняя логика позволяет переключаться с циклов чтения на циклы записи, и наоборот, без дополнительных задержек, кроме того, анализируется совпадение адресов записи и чтения для исключения избыточных операций.

Динамические полупроводниковые ЗУ с произвольным доступом. Как отмечалось выше, в качестве оперативных ЗУ в настоящее время чаще используются динамические ЗУ с произвольным доступом (DRAM). Такое положение обусловлено тем, что недостатки, связанные с необходимостью регенерации информации в таких ЗУ и относительно невысоким их быстродействием, компенсируются другими показателями: малыми размерами элементов памяти и, следовательно, большим объемом микросхем этих ЗУ, а также низкой их стоимостью.

Широкое распространение ЗУ этого типа проявилось также и в разработке многих его разновидностей: асинхронной, синхронной, RAMBUS и других. Основные из них рассматриваются далее.

Асинхронная динамическая память DRAM. В процессе совершенствования технологии производства изменялась и логика функционирования динамических ОЗУ.

Первые такие ЗУ, которые впоследствии стали называть асинхронными динамическими ОЗУ, выполняли операции чтения и записи, получив лишь запускающий сигнал (обычно, сигнал строба адреса) независимо от каких-либо внешних синхронизирующих сигналов. Диаграмма простых (не пакетных) циклов чтения и записи для таких ЗУ представлена на рис. 3.10, *а* и 3.10, *б* соответственно. Любой цикл (чтения или записи) начинается по спаду (фронту “1” →“0”) сигнала RAS#.

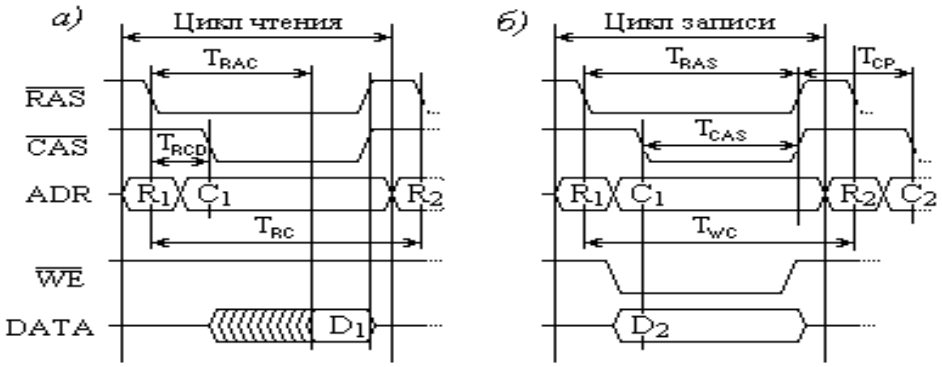


Рис.3.10. Временные диаграммы простых циклов: а – чтения; б – записи (асинхронной) динамической памяти

Как видно из диаграмм, адрес на шины адреса поступает двумя частями: адрес строки (обозначенный как R_1 или R_2) и адрес столбца (C_1 и C_2). В момент, когда на адресной шине установилось требуемое значение части адреса, соответствующий сигнал строба (RAS# или CAS#) переводится в активное (нулевое) состояние.

В цикле чтения (сигнал WE# во время этого цикла удерживается в единичном состоянии) после подачи адреса строки и перевода сигнала CAS# в нулевое состояние начинается извлечение данных из адресованных элементов памяти, что показано на диаграмме сигнала DATA как заштрихованная часть. По истечении времени доступа TRAC (RAS Access Time – задержка появления данных на выходе DATA по отношению к моменту спада сигнала RAS#) на шине данных устанавливаются считанные из памяти данные. Теперь после удержания данных на шине в течение времени, достаточного для их фиксации, сигналы RAS# и CAS# переводятся в единичное состояние, что указывает на окончание цикла обращения к памяти.

Цикл записи начинается так же, как и цикл чтения, по спаду сигнала RAS# после подачи адреса строки. Записываемые данные выставляются на шину данных одновременно с подачей адреса столбца, а сигнал разрешения записи WE# при этом переводится в нулевое состояние (известен и несколько иной цикл “задержанной” записи). По истечении времени, достаточного для записи данных в элементы памяти, сигналы данных, WE#, RAS# и CAS# снимаются, что говорит об окончании цикла записи.

Помимо названного параметра TRAC – времени доступа по отношению к сигналу RAS# (его значение для микросхем второй половины 90-х годов XX столетия составляло от 40 до 80 нс), - на диаграмме (рис. 3.11) указаны еще несколько времен:

TRCD – минимальное время задержки между подачей сигналов RAS# и CAS# (RAS-to-CAS Delay);

TRAS и TCAS – длительности (активного уровня) сигналов RAS# и CAS#;

TRC и TWC – длительности циклов чтения и записи соответственно;

TRP и TCP – времена подзаряда строки и столбца соответственно (время подзаряда определяет минимальную задержку, необходимую перед подачей очередного сигнала RAS# или CAS# после снятия (подъема в “1”) текущего).

Значения времен TRC и TWC для памяти (1990-х годов) составляли порядка 50 – 100 нс, так что на одно (полное) обращение уходило от 5 до 7 циклов системной шины в зависимости от ее частоты, особенностей используемого чипсета и собственно быстродействия памяти. Так, для системной шины с частотой 66 МГц длительность цикла составляет порядка 15 нс, что для 5 – 7 циклов дает диапазон 75 – 100 нс, если же частота системной шины составляла 100 МГц, то 5 циклов занимают 50 нс.

Подача адреса двумя частями удлиняет цикл обращения к памяти. Вместе с тем большинство обращений непосредственно к оперативной памяти производится по последовательным адресам.

Действительно, как отмечалось выше, до 90 % и более обращений процессора к памяти удовлетворяются кэш-памятью. Те обращения, которые не могут быть удовлетворены кэшем, вызывают обмен информацией между ОП и кэшем. При этом передачи выполняются блоками по 32 байта (4 цикла по 8 байт, в процессорах Intel 486 это были строки по 16 байт – 4 цикла по 4 байта), расположенными в последовательных адресах и называемыми строками кэша. Обмен информацией между оперативной памятью и внешними устройствами обычно выполняется целыми блоками, что также предполагает обращения по последовательным адресам.

Поскольку адрес строки является старшей частью адреса, то для последовательных адресов памяти адрес строки одинаков (исключение составляет переход через границу строки). Это позволяет в (пакетном) цикле обращений по таким адресам задать адрес строки только для обращения по первому адресу, а для всех последующих задавать только адрес столбца. Такой способ получил название FPM (Fast Page Mode – быстрый страничный режим) и мог реализовываться обычными микросхемами памяти при поддержке контроллера памяти чипсета, обеспечивая сокращение времени обращения к памяти для всех циклов пакета кроме первого. Получающаяся при этом временная диаграмма пакетного цикла чтения представлена на рис. 3.11.

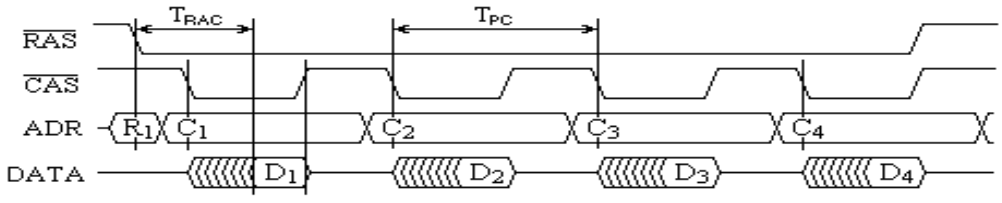


Рис. 3.11. Временная диаграмма цикла чтения последовательных адресов динамической памяти DRAM в режиме FPM

Как видно из рисунка, цикл чтения первого слова пакета выполняется так же, как и одиночное обращение. Второй и последующие циклы чтения оказываются короче первого из-за отсутствия фазы подачи адреса строки, и их длительность определяется минимально допустимым периодом следования импульсов CAS# – TPC (Page CAS Time). Соотношение длительностей первого и последующих циклов при частоте системной шины может достигать 5 : 3, откуда и обозначение 5-3-3-3, используемое как характеристика памяти (и чипсета) и указывающее, что первый из циклов пакета занимает по времени 5 циклов системной шины, а последующие – по 3 цикла.

Длительность (низкого уровня) импульса CAS# определяется временем не только извлечения данных из памяти, но и удержания их на выходе микросхемы памяти. Последнее необходимо для фиксации прочитанных данных (контроллером памяти), так как данные присутствуют на выходе только до подъема сигнала CAS#. Поэтому следующей модификацией асинхронной динамической памяти стала память EDO (Extended Data Output – растянутый выход данных). В микросхеме EDO памяти на выходе был установлен буфер-защелка, фиксирующий данные после их извлечения из матрицы памяти при подъеме сигнала CAS# и удерживающий их на выходе до следующего его спада. Это позволило сократить длительность сигнала CAS# и соответственно цикла памяти, доведя пакетный цикл до соотношения с циклами системной шины 5-2-2-2 (т.е. сократить длительность второго и последующих циклов в 1,5 раза только за счет выходного регистра-буфера). Временная диаграмма для режима EDO показана на рис. 3.12, а сам этот режим иногда называют гиперстраничным (Hyper Page Mode).

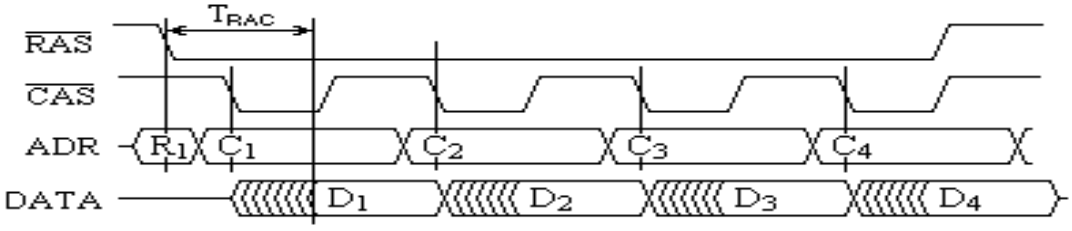


Рис. 3.12. Временная диаграмма цикла чтения последовательных адресов динамической памяти DRAM в режиме EDO

Синхронная динамическая память SDRAM. Синхронная динамическая память обеспечивает большее быстродействие, чем асинхронная при использовании аналогичных элементов памяти. Это позволяет реализовать пакетный цикл типа 5-1-1-1 при частоте системной шины 100 МГц и выше.

Основные сигналы интерфейса SDRAM схожи с сигналами интерфейса асинхронной памяти. Главные их отличия сводятся к появлению ряда новых сигналов:

1. У памяти SDRAM присутствует синхросигнал CLK, по переднему фронту которого производятся все переключения в микросхеме. Кроме этого сигнала имеется также сигнал CKE (Clock Enable), разрешающий работу микросхемы при высоком уровне, а при низком – переводящий ее в один из режимов энергосбережения.

2. В интерфейсе SDRAM имеются сигналы выбора банка BS0 и BS1 (Bank Select), позволяющие адресовать конкретные обращения в один из четырех имеющихся в микросхемах SDRAM банков (массивов элементов) памяти.

3. Присутствуют сигналы DQM маски линий данных, позволяющие блокировать запись данных в цикле записи или переключать шину данных в состояние высокого выходного сопротивления (z-состояние) при чтении.

4. Имеет место специфическое использование одной из адресных линий в момент подачи сигнала CAS#. Значение сигнала на этой линии задает способ подзаряда строки банка.

Кроме того, SDRAM-память сразу ориентирована на выполнение пакетных передач данных, причем длина пакета задается при инициализации микросхем памяти.

Выигрыш в производительности SDRAM достигается за счет более гибкого управления процессами чтения и записи, возможности задания параметров и лучших алгоритмов работы контроллера памяти.

Во-первых, в SDRAM памяти процессы синхронны, что позволяет более жестко упорядочить их во времени.

Во-вторых, микросхемы SDRAM имеют внутреннюю мультибанковую организацию. Это позволяет применять приемы, повышающие пропускную способность памяти. В частности, можно прибегнуть к чередованию, или расслоению, (interleave) адресов. Кроме того,

оказывается возможным в ряде случаев так спланировать порядок обработки обращений к памяти, чтобы уменьшить их времена. Однако это не совсем простая задача, так как необходимо учитывать временные ограничения на различные сочетания следующих друг за другом операций, длину пакетных циклов, особенности выполнения подзарда. Поэтому чипсеты различных производителей обеспечивают разную пропускную способность памяти.

Действительно, в общем случае процедура записи или чтения в SDRAM-памяти выполняется в три этапа:

1. Сначала при подаче сигнала RAS# происходит выбор нужной строки, или в терминах, принятых для этой памяти, выполняется команда активации банка.
2. Затем выполняются требуемые операции записи или чтения и передачи данных.
3. После записи или чтения строку, к которой выполнялось обращение, надо закрыть (выполнить подзаряд банка), иначе нельзя будет обратиться к новой строке этого же банка (вновь его активировать).

Именно за счет первого и последнего этапов и можно добиться ускорения работы памяти. Если очередное обращение к данному банку будет адресоваться к той же строке, то ее можно не закрывать, что позволит не выполнять заново команду активации банка.

Контрольные вопросы

1. Каковы принципиальные отличия памяти DRAM от памяти SDRAM?
2. Что является главным недостатком статической памяти?
3. Какие дополнительные сигналы введены в синхронную память?
4. Чем определяется длительность импульса CAS?
5. В чем преимущество памяти SDRAM?

Глава 4

РЕЖИМЫ РАБОТЫ

В этой главе мы рассмотрим три базовых режима работы вычислительной системы:

1) прерываний;
2) прямой доступ к памяти;
3) ожиданий.

У каждого режима есть особенности, но режим ожиданий используется как самостоятельная функция, так и в составе двух предыдущих режимов.

Каналы запросов прерывания (IRQ) используются различными устройствами для сообщения процессору о том, что должен быть обработан определенный запрос.

ПРЕРЫВАНИЯ

Каналы прерываний представляют собой проводники на системной плате и соответствующие контакты в разъемах. После получения IRQ компьютер приступает к выполнению специальной процедуры его обработки, первым шагом которой является сохранение в стеке содержимого регистров процессора. Затем происходит обращение к *таблице векторов прерываний,* в которой содержится список адресов памяти, соответствующих определенным номерам (каналам) прерываний. В зависимости от номера полученного прерывания запускается программа, относящаяся к данному каналу.

Указатели в таблице векторов определяют адреса памяти, по которым записаны программы-драйверы для обслуживания платы, пославшей запрос. Например, для сетевой платы вектор прерывания содержит адрес сетевых драйверов, предназначенных для работы с ней; для контроллера жесткого диска вектор указывает на программный код, обслуживающий контроллер.

После выполнения необходимых действий по обслуживанию устройства, пославшего запрос, процедура обработки прерывания восстанавливает содержимое регистров процессора (извлекая его из стека) и возвращает управление компьютером той программе, которая

выполнялась до возникновения прерывания.

Благодаря прерываниям компьютер может своевременно реагировать на внешние события. Например, всякий раз, когда с последовательного порта в систему поступает новый байт, вырабатывается IRQ.

Аппаратные прерывания имеют *иерархию приоритетов:* чем меньше номер прерывания, тем выше приоритет. Прерывания с более высоким приоритетом имеют преимущество перед прерываниями с более низкими приоритетами и могут "прерывать прерывания". В результате в компьютере может возникнуть несколько прерываний, "вложенных" друг в друга.

При генерации большого количества прерываний стек может переполниться и тогда ПК зависнет. Если такая ошибка возникает слишком часто, попытайтесь исправить ситуацию, увеличив размер стека.

По шине ISA запросы на прерывание передаются в виде *перепадов логических уровней,* причем для каждого из них предназначена отдельная линия, подведенная ко всем разъемам. Каждому номеру аппаратного прерывания соответствует свой проводник. Системная плата не может определить, в каком слоте находится пославшая прерывание плата, поэтому возможно возникновение неопределенной ситуации в том случае, если несколько плат используют один канал. Чтобы этого не происходило, система настраивается так, что каждое устройство (адаптер) использует свою линию (канал) прерывания. Применение одной линии сразу несколькими разными устройствами в большинстве случаев недопустимо.

Для того чтобы это было возможно, при разработке устройства необходимо предпринять специальные меры. Совместное использование линий прерывания предусмотрено лишь для немногих плат адаптеров. Это связано с принципиальными недостатками способа передачи IRQ в шине ISA. В компьютерах с шиной MCA IRQ передается в виде статического логического уровня, что позволяет разделить во времени прерывания, передаваемые по одной линии. В принципе, в компьютере с шиной MCA все платы можно настроить на одно и то же прерывание без возникновения конфликтов между ними. Однако для повышения производительности системы их лучше разделять как можно большими временными интервалами.

Внешние аппаратные прерывания часто называются *маскируе-*

мыми прерываниями, т.е. их можно отключить ("замаскировать") на время, пока процессор выполняет другие критические операции. Организация эффективной обработки прерываний – задача для программиста.

Поскольку в шине ISA совместное использование прерываний обычно не допускается, при установке новых плат может обнаружиться недостаток линий прерываний Если две платы используют одну и ту же линию IRQ, то их нормальную работу нарушит возникший конфликт. Ниже рассмотрены прерывания стандартных устройств и свободные линии.

Прерывания в 8-разрядной шине ISA. В компьютерах PC и XT с 8-разрядным процессором 8088 имеется 8 внешних аппаратных прерываний. Стандартное распределение этих прерываний, пронумерованных от 0 до 7, показано на рис. 4.1.

В компьютере с 8-разрядной шиной ISA имеющихся прерываний (ресурсов) часто катастрофически не хватает. Попытка установить в компьютер PC/XT несколько устройств, требующих обработки своих прерываний, может привести к тому, что разрешить проблему нехватки прерываний вы сможете единственным способом — вынуть реже всего используемую плату адаптера.

Для того чтобы не возникало проблем при генерации фактически несуществующего IRQ2, конструкторы выделили дополнительное прерывание IRQ9 для заполнения образовавшейся бреши. Это означает, что любая добавленная в компьютер плата, для которой характерно использование прерывания IRQ2, на самом деле будет использовать IRQ9. Это следует учитывать, чтобы случайно не назначить прерывание IRQ9 другому устройству.

Поскольку прерывание IRQ2 теперь используется непосредственно на системной плате, линия IRQ 9 подключается к тем контактам слотов, которые обычно используются для IRQ 2. Плата, настроенная на IRQ 2, фактически использует IRQ 9. Соответственно исправлена и таблица векторов прерываний. Такой подход обеспечивает совместимость со структурой прерываний PC/XT, а платы адаптеров, настроенные на IRQ 2, могут функционировать нормально. Отметим, что линии прерываний 0, 1, 2, 8 и 13 не выведены на разъемы шины и не используются платами адаптеров. Линии прерываний 8, 10, 11, 12, 13, 14 и 15 подключены ко второму контроллеру. Они могут исполь-

зоваться только адаптерами с 16-разрядным разъемом, поскольку подведены к контактам в "расширенных" частях слотов. Линия IRQ9 подключена к разъему 8-разрядного слота вместо IRQ2 и доступна 8-разрядным платам, которые используют ее как линию IRQ2.

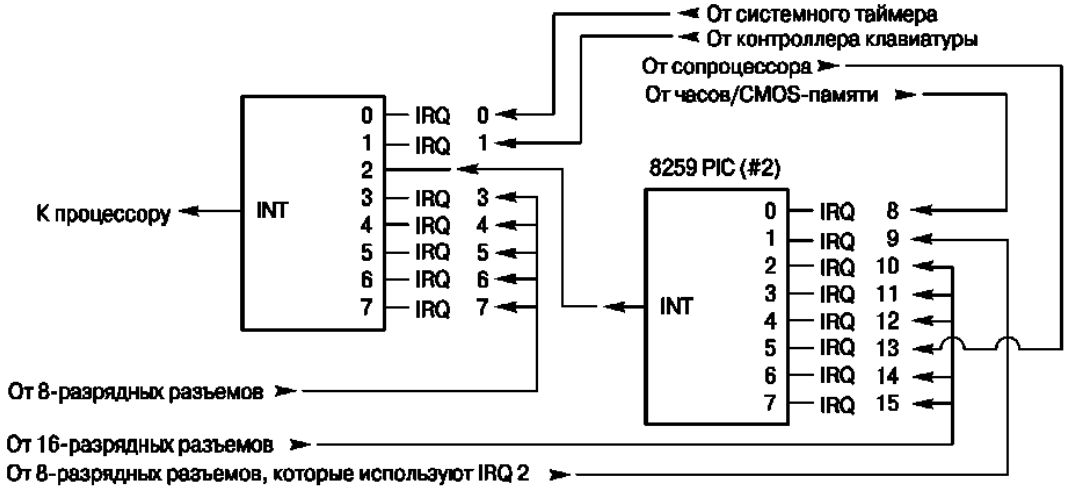


Рис. 4.1. Распределение прерываний

Вычислительная система, работающая на основе рассмотренных выше принципов, была бы весьма неэффективна. Другими словами, объем выполняемой работы был бы намного меньше потенциальных возможностей машины. Действительно, большинство внешних устройств работает гораздо медленнее центрального процессора, и поэтому в течение длительного времени, пока данные не будут введены в ЭВМ, процессор вынужден простаивать, не выполняя вычислений, а только проверяя, все ли данные приняты с внешних устройств.

Предположим, что вычислительная система считывает с перфоленты с помощью устройства считывания до 500 символов в секунду (один символ за 2000 мкс). Если допустить, что цикл обращения к памяти составляет одну микросекунду, то программа, изображенная на рис. 4.2, вероятно, потребует около 20 мкс на выполнение цикла шага 2. Самое быстродействующее устройство чтения перфоленты передает один символ каждые 2000 мкс (500 раз в 1с). То есть в течение 2000 мкс CPU полезно используется только около 20 мкс, или 1% времени. Остальные 99 % времени CP11 простаивает в ожидании приема знака от устройства чтения перфоленты, что демонстрирует неэффективность использования центрального процессора.

Введение прерываний позволяет существенно увеличить эффективность использования СРТЛ за счет выполнения одних программ в

то время, когда задерживается исполнение других. Ниже будет показано, что система передачи данных по шине ввода-вывода может оказаться весьма эффективной, если устройства подключаются к шине автономно, а связь по ней устанавливается по принципу «рукопожатия».

Предположим, что набор автономных устройств на шине ввода-вывода включает центральный процессор. Последний должен быть спроектирован так, чтобы его работа продолжалась до тех пор, пока не понадобится связь с внешним устройством или пока внешнее устройство не выдаст сигнала связи с центральным процессором. Таким образом, центральный процессор должен инициировать работу внешнего устройства, а затем прервать с ним связь до тех пор, пока эта работа не будет выполнена. Например, сначала центральный процессор запускает устройство чтения перфоленты на выполнение команды ввода, инициируя считывание перфоленты. После этого CPU может находиться в состоянии выполнения пользовательских программ, пока устройство чтения не выдаст сигнал о том, что символ считан и находится в регистре данных контроллера устройства чтения перфоленты.

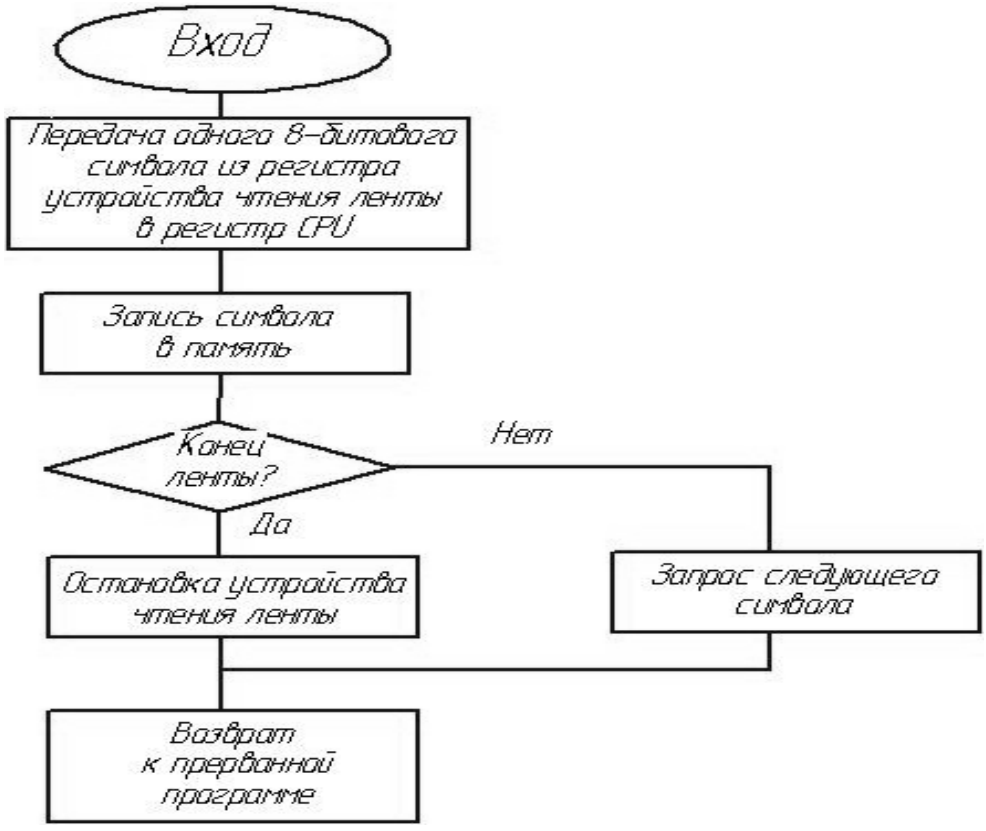


Рис. 4.2. Программа обслуживания устройства чтения перфоленты

Вместе с тем, когда контроллер намерен установить связь с процессором, а последний обрабатывает программу, не имеющую ничего общего с устройством чтения перфоленты, процессор должен прервать программу, выполняемую в данный момент, если ему самому необходимо обратиться к устройству чтения перфоленты. Центральный процессор вычислительных систем, спроектированных подобным образом (т. е. процессор любой современной ЭВМ), должен потратить около 20 мкс для обслуживания устройства чтения перфоленты, после чего он возвращается к прерванной программе. Поэтому, учитывая распределение времени между выполняемой программой и программой обработкой данных, принимаемых от устройства чтения с перфоленты, центральный процессор будет занят все 100 % времени полезной работой.

Рассмотрим более общий случай. Пусть центральный процессор занят обработкой программы P1, когда поступает запрос на прерывание этой программы от внешнего устройства. Это вызывает переключение центрального процессора на выполнение другой программы, например P2, являющейся программой управления данным внешним устройством. Программа P2 может быть названа *подпрограммой обслуживания внешнего устройства.* Для устройства чтения перфоленты эта подпрограмма может быть представлена блок-схемой, изображенной на рис. 4.2. После окончания выполнения подпрограммы обслуживания центральный процессор должен вернуться к программе P1. Если система прерываний достаточно эффективна (а иначе она и не нужна), то она должна иметь механизм возврата и безошибочного перехода от P1 к P2 и затем обратно. Создание такой системы представляет определенные трудности и должно являться одним из основных элементов разработки вычислительной системы.

Было бы непрактично использовать команду типа «Обратить внимание на любое внешнее устройство, требующее прерывания», так как это замедлило бы выполнение всех программ, находящихся в памяти ЭВМ. Поэтому механизм прерывания должен быть полностью автоматическим и выполняться аппаратно, хотя при этом может использоваться и поддержка со стороны специального программного обеспечения. Проблема передачи управления от про-

граммы P1 к программе P2 и обратно аналогична той, которая была рассмотрена в гл. 2 (задача входа в подпрограмму и выхода из нее). Переход к подпрограмме должен выполняться здесь также автоматически и аппаратно, но не по команде CP11, а по сигналу, передаваемому процессору от внешнего устройства. Если продолжить сравнение с задачей входа в подпрограмму, то можно сказать, что аппаратные средства обеспечивают вход без какой-либо поддержки программного обеспечения. Возврат же к прерванной программе совершается при помощи команды возврата, подобной команде возврата из подпрограммы.

Состояние процессора. Состояние процессора в любой момент времени прежде всего характеризуется:

– содержанием всех регистров CPU;

– состоянием всех сигналов управления в аппаратуре.

После прерывания программы и выполнения подпрограммы обслуживания устройства CPU должен вернуться точно к такому же состоянию, которое было в момент прерывания. Как только это случится, прерванная программа будет продолжать работу, как будто ничего не произошло.

Содержимое всех регистров должно быть сохранено в памяти.

Для этого наиболее естественным является использование специального раздела памяти вычислительной системы. Чтобы ни одна программа не могла воспользоваться этим разделом, все обращения к нему прикладных программ должны блокироваться. Однако состояние машинных сигналов управления не может быть сохранено, потому что операция по их сохранению сама же будет их разрушать. К счастью, эти сигналы управления возвращаются к определенному состоянию перед окончанием исполнения каждой команды, в то время как выполнение машинного цикла находится между моментом завершения соответствующей исполняющейся команды и началом выборки следующей. Именно в этот момент CPU опознает возникновение прерывания. Если внешнему устройству и придется ждать несколько микросекунд, пока процессор обработает его запрос на прерывание, то это не будет сказываться на его работе, потому что такие ситуации происходят только эпизодически (один раз за 2000 мкс, как в примере с устройством чтения перфоленты). Однако то, что внеш-

нее устройство должно ждать, пока не установится связь между ним и CPU, подчеркивает, что эта связь устанавливается по принципу «рукопожатия», а не односторонней синхронизации.

Реакция процессора на прерывание. В заключение рассмотрения реакций вычислительных систем на запросы на прерывание необходимо дать полное представление о всех основных операциях, которые должны выполняться чисто аппаратно или по возможности с оптимальным применением как аппаратных, так и программных средств. Сначала рассмотрим вычислительную систему, которая учитывает наличие только одного внешнего устройства, что, несмотря на нереальность ситуации, может дать представление об основных принципах более сложных систем. Все усложнения, которые вытекают из наличия в системе большого числа внешних устройств, будут рассмотрены ниже.

Соответствующая часть памяти системы представлена на рис. 4.3. Предположим, что при выполнении программы P устройство чтения перфоленты выставляет запрос на прерывание, когда центральный процессор обрабатывает команду из ячейки N—1. На этой стадии программный счетчик (PC) содержит число N, которое является адресом следующей команды выполняемой программы P. CPU обеспечит переход к ячейке S, которая является началом программы обслуживания прерывания, при условии, что число S будет записано в PC в качестве адреса выборки следующей команды. Система должна обеспечить сохранение содержимого PC как адреса возврата (адреса связки). Это почти полностью совпадает с тем, что происходит при выполнении соответствующей команды входа в подпрограмму. После окончания команды, выполняющейся в момент поступления запроса на прерывание, аппаратные средства должны обеспечить запись содержимого PC в память, загрузку PC из ячейки S и, наконец, запуск выборки команды. Всякий раз, когда происходит аппаратное запоминание информации в памяти, используются фиксированные адреса. Разработчиками аппаратных средств часто, например, используется нулевая ячейка памяти. Ее удобно использовать, поскольку она находится на границе диапазона адресов памяти. Это позволяет ввести несущественное ограничение — программы не должны использовать ячейки памяти с младшими адресами.

<table>
<tr><td>Адрес</td><td>Содержимое памяти</td></tr>
<tr><td>0</td><td>N</td></tr>
<tr><td>1</td><td>S</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td rowspan="3">Программа P</td></tr>
<tr><td>N-1
N</td></tr>
<tr><td>S</td><td rowspan="2">Программа обслуживания устройства чтения перфоленты</td></tr>
<tr><td></td></tr>
</table>

Рис. 4.3. Отображение распределения памяти программы прерывания

Возможен более гибкий вариант системы, когда число аппаратно копируется из программного счетчика в нулевую ячейку памяти, а сам программный счетчик загружается из ячейки 1. Хотя аппаратно используются только фиксированные адреса, обслуживающая программа может быть размещена в любом месте памяти, если начальный адрес программы обслуживания (S) будет предварительно загружен в ячейку 1.

Центральный процессор в этом случае выбирает команду из ячейки S и тем самым запускает программу обслуживания устройства чтения перфоленты. Если несколько первых команд этой программы являются командами типа «Запомнить [регистр] в памяти», то содержимое всех регистров CPU будет записано в память. Когда выполнение программы обслуживания будет почти закончено, последние несколько ее команд должны быть командами типа «Загрузить регистры из памяти». Они загрузят регистры CPU точно такой же информацией, какая в них содержалась в момент прерывания программы P. Самая последняя команда должна быть командой типа «Косвенный переход по ячейке 0». Эта команда скопирует число N из ячейки 0 в программный счетчик, так что следующая выбранная команда будет командой из программы P, которая должна была выполняться, если бы не произошло прерывание.

Рассмотренная методика с некоторыми изменениями использована в нескольких вычислительных системах. Так, например, в ячейке 0 аппаратным способом сохраняется содержимое программного счетчика, а содержимое ячейки 1 используют как команду, т. е. содержи-

мое ячейки 1 загружается в регистр команд вместо загрузки в PC. В ячейке 1 находится команда «Передать управление ячейке S», что соответствует входу в программу обслуживания внешнего устройства. Если последняя занимает старшие адреса памяти, то в ячейке 1 должна будет находиться команда косвенного перехода.

Другой способ реализации прерывания использует стек. После распознавания запроса на прерывание аппаратно обеспечивается проталкивание содержимого PC в стек и выборка адреса обслуживающей программы из фиксированной ячейки памяти. В машинах со стеком несколько первых команд являются командами «Протолкнуть [регистр] в стек», которые обеспечивают сохранение содержимого регистров CPU. Последние команды программы, являясь командами «Вытолкнуть содержимое стека в регистр CPU, восстанавливают содержимое регистров. Самая последняя команда возврата загружает PC из вершины стека, чтобы обеспечить возврат к прерванной программе. Применяя с некоторыми вариациями этот способ к машинам, имеющим небольшое число накапливающих регистров, можно аппаратно организовать стек из всех регистров CPU, которые участвуют в прерывании. Это обеспечит сокращение времени, необходимого для выполнения задачи. В таких машинах команда «Возврат из прерывания», загрузив из стека программный счетчик, загружает из стека накапливающие регистры. Машины с большим числом накапливающих регистров не используют автоматическую организацию стековой памяти, потому что обычно быстрее выбрать и выполнить команды для сохранения содержимого нескольких регистров, используемых в программе обслуживания, чем делать то же самое для всех регистров. Сохранение и восстановление содержимого регистров играет первостепенную роль в критические моменты работы системы прерываний.

Вообще говоря, нужно учитывать, что приходится аппаратно сохранять содержимое PC и некоторых из неадресуемых (программно недоступных) регистров CPU (регистр кодов условия или регистр состояния). Сохранить содержимое адресуемых регистров CPU можно двумя способами — аппаратно и программно. Обычно использование программного способа проще, однако для ЭВМ, имеющей малое число адресуемых регистров, это время может возрастать, так как перед каждой командой сохранения требуется выполнение цикла выборки. Для выполнения одного цикла выборки необходим по крайней мере

один цикл памяти, а в 8-битовых вычислительных системах может потребоваться и три цикла памяти. В машинах с большим числом адресуемых регистров программный метод много гибче и обычно позволяет достигать более высоких скоростей выполнения программ.

Контрольные вопросы

1. Опишите три базовых режима работы вычислительной системы.
2. Для чего используются прерывания?
3. Как реализуется иерархия приоритетов прерываний?
4. Чем характеризуется состояние процессора в определенный момент времени?
5. Какие проблемы может принести генерация большого количества прерываний?

Передача сигналов прерывания по шине ввода-вывода. Если центральный процессор допускает прерывания, то в шине ввода-вывода должна быть введена линия запроса на прерывание. Внешнее устройство может использовать эту линию для передачи сигнала в CPU, указывающего на то, что CPU должен прервать исполняемую программу и перейти к обслуживанию внешнего устройства. Для этой же цели в шине ввода-вывода имеется линия подтверждения прерывания, используемая для того, чтобы CPU подтвердил принятие запроса и сообщил об этом внешнему устройству. Эти две линии используются при реализации метода «рукопожатия». Обычно внешнее устройство прекращает запрашивать прерывание, когда получено подтверждение, а аппаратные средства CPU вызвали программу обслуживания устройства. Устройство чтения перфоленты, например, запрашивая программу обслуживания, вводит символ в регистр CPU. Символ будет находиться в регистре данных контроллера этого устройства; именно его появление является причиной запроса на прерывание.

Линия запроса на прерывание и линия подтверждения прерывания — две основные линии в шине ввода-вывода во всех вычислительных системах, использующих систему прерывания. Большее число линий встречается в тех случаях, когда требуется выполнение дополнительных функций.

Прерывание в системе с большим количеством внешних устройств. Механизм прерывания, рассмотренный для случая с одним устройством, может быть теперь распространен на случай с большим числом устройств в системе ввода-вывода. Внешние устройства системы реагируют точно так же, как это описано выше, однако суммарная реакция усложняется при их объединении. Система должна определить, какое устройство выставило запрос на прерывание и какое обслуживается программой. Система должна также решить проблему, какое устройство необходимо обслужить, если запрос на прерывание выставили одновременно несколько устройств.

Определение источника прерывания. Первой проблемой, с которой «сталкивается» устройство, требующее прерывания, является то, что оно «считает себя» единственным устройством, находящимся в подобном состоянии, и не учитывает возможности одновременных запросов от других устройств. В принципе могут быть два подхода к решению этой проблемы: во-первых, применение специальных программных средств и, во-вторых, введение дополнительных аппаратных средств. При разработке вычислительных машин, например с середины и до конца 1960-х годов, относительно дорогим аппаратным средствам для решения этой проблемы предпочитали программные средства. Последние разработки свидетельствуют о тенденции к расширению применения аппаратных средств и в первую очередь для распознавания запросов. Кроме относительного снижения стоимости аппаратных средств поддержанию этой тенденции способствует также и то, что программное решение обычно медленнее соответствующих аппаратных решений, а скорость обработки прерываний является важнейшим аспектом снижения затрат времени на обслуживание и повышение эффективности системы.

Определение источника прерывания программными средствами. Предположим, что линия шины запроса на прерывание является простой линией соединения со всеми устройствами подобно любой другой линии шины. Когда устройство запрашивает прерывание по сигналу на этой линии, аппаратура обеспечивает вход в обслуживающую программу. В предыдущем обсуждении обслуживающая программа рассматривалась как программа обслуживания устройства, но в системе с большим количеством внешних устройств она является *программой обслуживания прерывания,* и ее функции состоят в опо-

знавании прерывающего устройства и переходе к его программе обслуживания. Проблема опознавания решается часто только программными средствами. Программа обслуживания прерывания обычно *опрашивает* устройства по очереди, чтобы найти то из них, которое запрашивает прерывание. Опрос может быть выполнен путем или использования команд пропуска, или чтения содержимого регистра состояния каждого устройства в регистр CPU с последующей проверкой бита запроса на прерывание. Пример программы, использующей команды пропуска, показан на рис. 4.4. Предполагается, что в программах обслуживания устройств имеются команды сохранения содержимого регистров. Линия пропуска в шине ввода-вывода является единственным схемным средством в этой системе опознавания устройства. Она экономит время, которое понадобилось бы для работы программы опроса устройств по очереди и проверки их состояния в регистре CPU, когда последовательность команд CPU определяет, установлен ли бит запроса на прерывание. Выполнение подобной программы заняло бы много времени.

АДРЕС	КОМАНДА
N	Пропустить, если устройство A не требует прерывания
N + 1	Перейти к программе обслуживания устройства A
N + 2	Пропустить, если устройство B не требует прерывания
N + 3	Перейти к программе обслуживания устройства B
N + 4	Пропустить, если устройство C не требует прерывания
N + 5	Перейти ...

Рис. 4.4. Программа опроса устройств, использующая пропуски

Команды пропуска представляют собой, конечно, команды ввода с кодом команды, указывающим, что логическую 1 на линии пропуска установит то устройство, которое запрашивает прерывание.

В тех вычислительных системах, в которых и память, и внешние устройства находятся на общей шине ввода-вывода и где нет специальных команд ввода-вывода, должны быть опрошены регистры состояния каждого устройства. Система часто способна выполнять это одной командой (подобно пропуску, использованному выше, но работающей быстрее, так как и проверка, и переход выполняются в одной и той же команде). Предположим, в системе возник запрос на преры-

вание от некоторого внешнего устройства и старший бит регистра состояния установлен в единицу внешним контроллером. Если CPU использует содержимое регистра состояния как число, то запрос на прерывание делает это число отрицательным. Команда «Перейти, если число в ячейке N отрицательно», являющаяся стандартной командой условного перехода, осуществляет переход к программе обслуживания устройства, если N – адрес его регистра состояния. Программа опроса устройств такой машины представляет собой список команд условного перехода, что и показано на рис. 4.5.

АДРЕС	КОМАНДА
N	Перейти, если A отрицательно
N + 1	Перейти, если B отрицательно
N + 2	Перейти, если C отрицательно
N + 3	Перейти ...

Рис. 4.5. Опрос устройств, использующий условные переходы

Определение источника прерывания техническими средствами. Опознавание устройства техническими средствами происходит в системе, использующей *векторы прерывания.* Здесь контроллер внешнего устройства, запрашивающего прерывание, устроен таким образом, что при получении от CP1) подтверждения запроса контроллер выставляет на некоторые специальные линии шины так называемый вектор (двоичное число), представляющий собой указатель программы обслуживания внешнего устройства. Это позволяет CP11 очень быстро находить обслуживающую программу. Рис. 4.6 показывает одну из таких систем, являющуюся расширением системы.

Для каждого внешнего устройства в системе резервируются две ячейки памяти. Первая используется для хранения содержимого программного счетчика (адреса возврата), а вторая – для хранения адреса начала программы обслуживания внешнего устройства. Когда CPU подтверждает запрос на прерывание, он использует вектор для генерации четного 16-битового адреса. Затем содержимое программного счетчика записывается в ячейке памяти с этим адресом. PC загружается из следующей (нечетной) ячейки. Начинается выборка команды – первой команды программы обслуживания устройства, другими сло-

вами, программы, к выполнению которой переходит CPU. Процесс этот очень быстрый, занимающий не более двух циклов памяти (от 1 до 2 мкс). По сравнению с системами опроса устройств время, необходимое CPU для ввода в действие программы обслуживания устройства, не зависит от количества внешних устройств в системе.

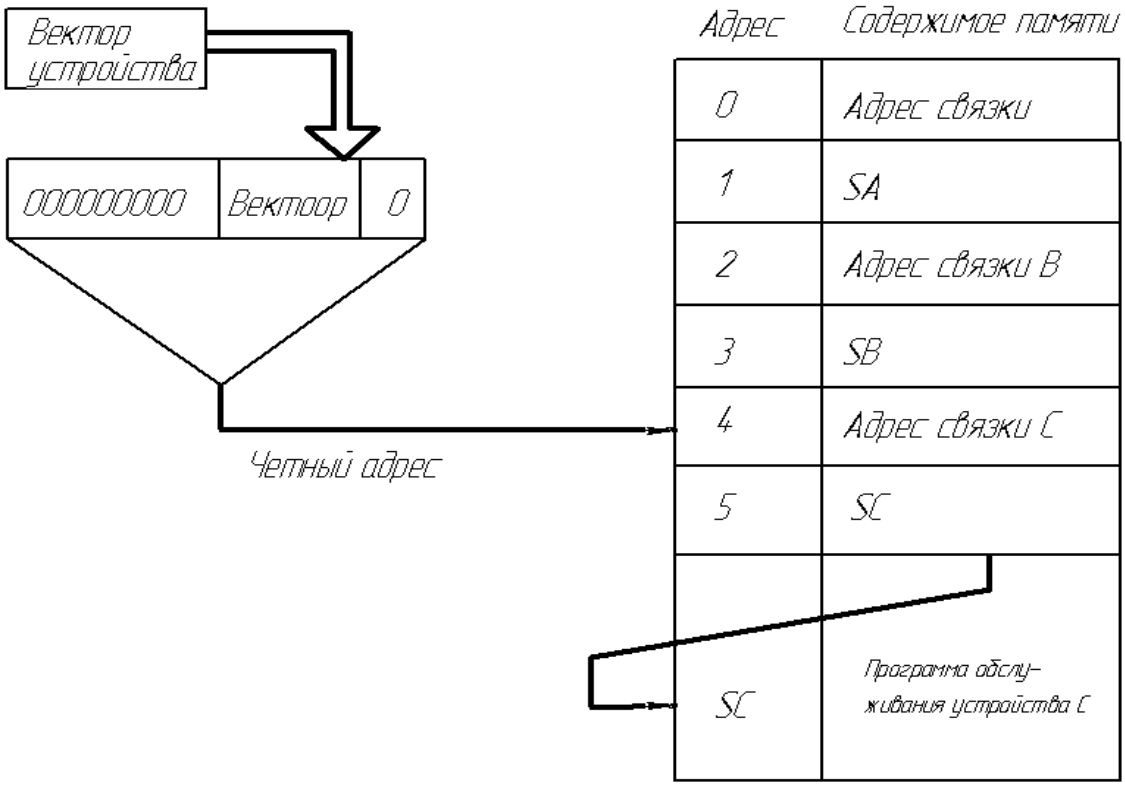


Рис. 4.6. Использование вектора устройства

В вычислительных системах со стеком содержимое программного счетчика при прерывании засылается в стек. Адрес начала программы обслуживания устройства выбирается из ячейки памяти, адрес которой сформирован по вектору устройства, как показано на рис. 4.6.

В некоторых случаях в шине ввода-вывода могут быть предусмотрены специальные линии для векторов; например, для 64 устройств должно быть шесть таких линий. Часто в качестве линий для векторов используются линии адреса или данных шины ввода-вывода. В этом случае длина вектора может доходить до 16 бит.

Одновременные запросы на прерывание. В системе с несколькими внешними устройствами (т.е. в реальной системе) возможен случай запроса на прерывание от двух устройств или более во время выполнения CPU какой-либо команды.

Таким образом, в конце выполнения этой команды CPU будет иметь дело с двумя запросами, называемыми *одновременными,* хотя фактически они могут возникнуть не в одно и то же время. Запросы одновременны в том смысле, что представляются вместе центральному процессору тогда, когда последний находится па стадии распознавания запроса на прерывание. Это показано на рис. 4.7. Система должна быть способна решить, какое устройство обслужить, а какое оставить ждать; другими словами, она должна установить *приоритеты устройств.*

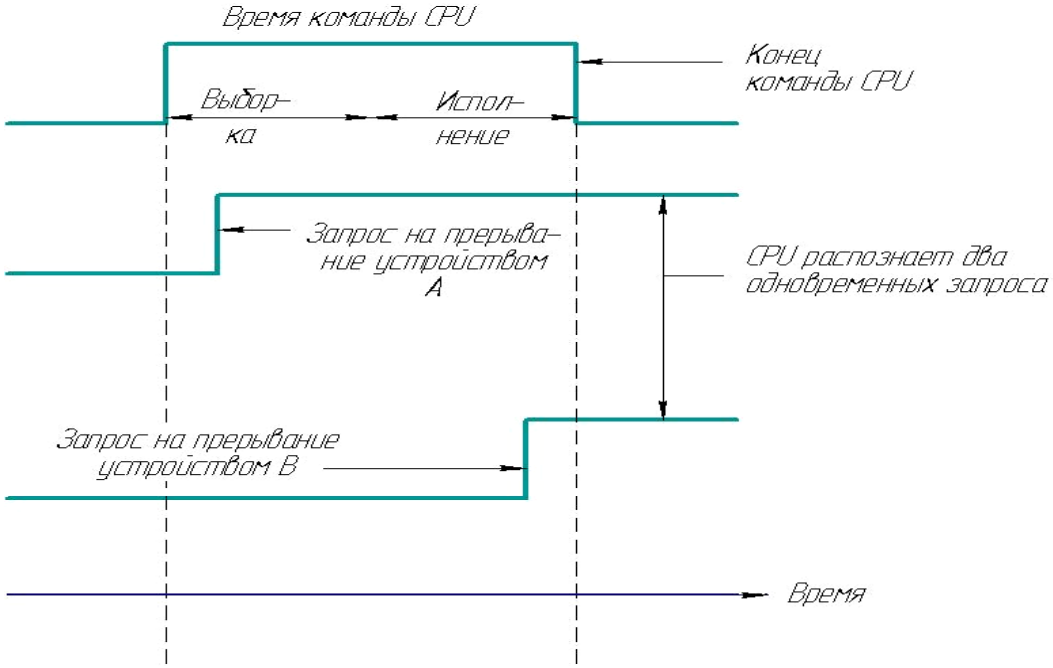


Рис. 4.7. Распознавание запроса на прерывание

Программное управление приоритетами. В программах опознавания устройств, т.е. в программах опроса, рассмотренных ранее, приоритеты установлены программно, так как устройства опрашиваются в определенной последовательности. Устройство с более высоким приоритетом опрашивается раньше других. Интересно отметить, что устройство, запросившее прерывание первым, может быть обслужено не первым. Предположим, у устройства с низким приоритетом возник запрос на прерывание и ЭВМ переходит к программе опроса устройств. Далее предположим, что запрос на прерывание возник у устройства с высоким приоритетом. Когда оно опрашивается первым, запрос на прерывание обнаруживается, и поэтому именно это устрой-

ство будет обслужено первым. Устройство же, первым запросившее прерывание, будет обслужено после выполнения программы обслуживания устройства с более высоким приоритетом.

Большое преимущество программной установки приоритетов заключается в том, что приоритеты находятся здесь под контролем программиста и, следовательно, могут быть изменены и приспособлены к требованиям вычислительной системы. Устройство с высоким приоритетом может быть опрошено несколько раз во время выполнения программы опроса, если это необходимо.

Аппаратное управление приоритетами. Существуют два основных метода аппаратного управления приоритетами. Первый использует линию подтверждения прерывания, идущего *по цепочке* от CPU, как показано на рис. 4.8. Когда CPU вырабатывает сигнал подтверждения прерывания, последний посылается только устройству с наивысшим приоритетом (устройству A на рис. 4.8). Если это устройство не запрашивает прерывания, сигнал проходит к следующему устройству по цепочке и т.д. Только в том случае, если устройство запрашивает прерывание, оно захватывает сигнал подтверждения и препятствует его пересылке дальше по цепочке. Прерывающее устройство не только захватывает сигнал подтверждения прерывания, но и отвечает соответствующим образом, например выставляет вектор прерывания на адресные линии шины.

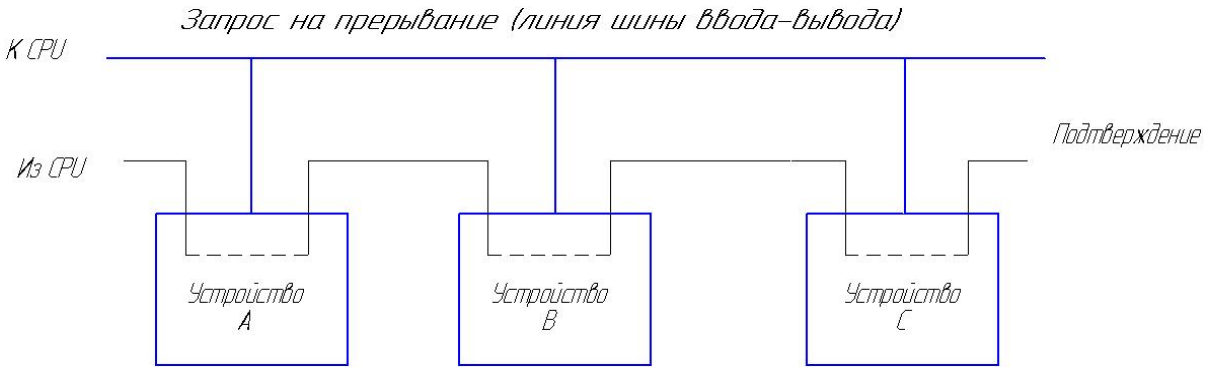


Рис. 4.8. Подтверждение прерывания, идущего по цепочке

Второй способ заключается в использовании нескольких линий запроса на прерывание. Если каждое устройство связано с CPU своей собственной линией запроса на прерывание, то при помощи простой системы можно реализовать ситуацию, когда к CPU проходит запрос

только наивысшего приоритета. Тот факт, что каждый запрос идет по своей индивидуальной линии, также означает, что в системе происходит опознавание устройства. Таким образом, нет необходимости в векторе для опознавания устройства.

Практически при аппаратном управлении приоритетами чаще всего используется комбинация двух описанных способов. Такие системы используют, скажем, восемь или 16 пар линий запроса – подтверждения прерывания. Линии запроса на прерывание устанавливаются на приоритетный уровень специальной схемой, а линия подтверждения прерывания, соответствующая каждому приоритетному уровню, идет по цепочке через внешние устройства на этом уровне. Подобную систему обычно называют системой *многоуровневого прерывания.* В цепочке на любом уровне каждое устройство, присоединенное к системе, имеет свой единственный приоритет. Так как на каждом приоритетном уровне в цепочке в основном более одного устройства, то в системе используются векторы для переключения CPU на работу программы обслуживания внешних устройств.

Программно-аппаратное управление приоритетами. Большой недостаток аппаратной системы управления приоритетами заключается в том, что однажды установленные приоритеты фактически фиксируются. Существуют комбинированные системы, в которых пытаются соединить гибкость программных средств с высокими скоростями аппаратных реализаций. В основном эти системы используют два регистра — *регистр прерываний* и *регистр маски прерывания.* Внешние по отношению к СРИ аппаратные средства здесь подобны средствам многоуровневого прерывания, описанным выше. Внутри CPU линии запроса па прерывание оканчиваются в регистре прерываний. Запрос на прерывание от любого устройства на одном уровне вызывает установку соответствующего бита в регистре прерываний в единицу.

Регистр маски прерывания – это адресуемый регистр; таким образом, его содержимое может быть изменено командой программы. Соответствующие биты двух регистров логически умножаются (операция И). Следовательно, CPU сможет распознать запрос на прерывание только в том случае, когда соответствующий бит регистра маски установлен в единицу. Схемы CPU позволяют распознавать только запрос наивысшего приоритета. В любое время программа может за-

претить запросы на прерывание на любом уровне, установив в нуль соответствующий бит регистра маски прерывания. На рис. 4.9 на уровнях 0, 1, 2, 5 и 7 показано это запрещение.

Путем запрета некоторого уровня N и разрешения уровня N—1 приоритет прерывания на уровне N—1 можно сделать выше, чем на уровне N. Таким образом, программа управляет аппаратной установкой приоритетов. Этот способ применяется все чаще в современных вычислительных системах, сложность технических средств которых постоянно растет.

В многоуровневой системе устройства на каждом приоритетном уровне могут быть программно опрошены. Это устраняет необходимость генерации вектора внешними устройствами. Программа опроса для определенного уровня запускается аппаратными средствами. Приоритетная схема СРСГ генерирует вектор длиной 4 бит (в 16-уровневой системе). Этот вектор представляет собой число, соответствующее самому высокому приоритетному уровню запроса на прерывание. Вектор используется аппаратными средствами для запуска программы опроса — программы обслуживания прерывания на этом приоритетном уровне. Обычно в многоуровневой системе на каждом уровне находятся только несколько внешних устройств. Программа опроса для каждого уровня выполняется, следовательно, быстрее программы опроса всех внешних устройств. Эта система быстрее программной, рассмотренной выше, но медленнее аппаратной векторной системы.

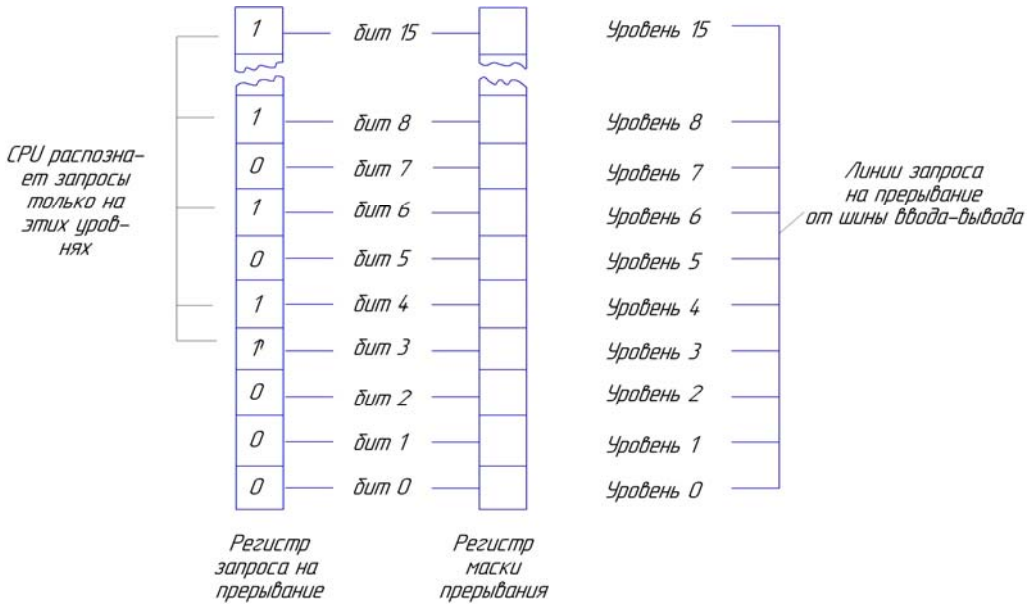


Рис. 4.9. Многоуровневая система прерывания

Вложенные прерывания. Для некоторых внешних устройств в вычислительной системе может существовать определенное критическое время, в течение которого должен быть передан ответ на запрос на прерывание, чтобы предотвратить системную ошибку. Например, дисковое запоминающее устройство может пересылать в память CPU по одному слову каждые 20 мкс. Поэтому оно должно обслуживаться по крайней мере каждые 20 мкс во время передачи информации от него в память. Предположим, что устройство, работающее гораздо медленнее, чем дисковое, такое, как устройство чтения перфоленты, запрашивает и получает обслуживание прерывания. Предположим далее, что дисковое запоминающее устройство запрашивает прерывание уже после того, как CPU начнет выполнять программу обслуживания устройства чтения перфоленты. Если эта программа закончит работу в пределах 20 мкс, не возникает никаких проблем. Если же она не уложится в это время, необходимо разрешить дисковому устройству прервать обслуживание устройства чтения перфоленты, которое может быть остановлено в промежутке между передачей символов на любое необходимое время. Далее, нет ничего необычного в том, что CPU выполняет программу, которая может оказаться программой обслуживания устройства (но CPU «не знает», какого именно), а контроллер диска делает запрос на прерывание. Если запрос разрешен, то по окончании обслуживания диска все описанные системы обработки прерываний позволят осуществить возврат к программе обслуживания устройства чтения перфоленты. Но можно ли по окончании обслуживания последнего вернуться к программе, которую оно прервало? Подобную ситуацию называют вложением прерываний. Для ее реализации требуется такая система обработки прерываний, которая позволяет сохранять адреса возврата через программы обслуживания в начальную программу. Возможно, ранее читателю было трудно понять, зачем для каждого устройства предусмотрены две ячейки памяти (вместо одной общей) для адреса возврата. Причина этого кроется в том, что при наличии одной общей ячейки для адреса возврата не представляется возможным вложение прерываний. Поскольку каждое устройство имеет ячейку, в которой хранится адрес возврата из своей программы обслуживания, то может быть разрешено прерывание подпрограммы обслуживания прерывания.

В вычислительных системах, использующих стек, адрес возвра-

та при прерывании вводится в стек. Затем совершается переход к программе обслуживания устройства или к программе обслуживания прерываний. Если и эта программа прервана, содержимое программного счетчика снова проталкивается в стек. Это не разрушает другую информацию, находящуюся в стеке, и при условии достаточно большой глубины стека допускается вложение неограниченного количества программ обслуживания.

Разрешение и запрет прерываний. В вычислительных системах насчитываются три уровня разрешения и запрета прерываний. На самом высоком уровне запрещены все запросы на прерывание. На втором уровне могут быть запрещены прерывания от одного или нескольких уровней (в системе с многоуровневыми прерываниями) при помощи очистки бит в регистре маски прерывания. На самом низком уровне запрос на прерывание, идущий к CPU от некоторых внешних устройств, может быть запрещен установкой бита запрета прерывания в регистре состояния контроллера внешнего устройства.

С помощью специальной схемы большинство вычислительных систем может установить CPU в состояние, не способное к приему прерываний в течение некоторого времени. Как только прерывание опознано, схема не только сохраняет адрес возврата, о чем было сказано выше, но и переводит CPU в состояние, не способное к приему прерываний. Это может быть сделано установкой бита запрета прерывания в регистре состояния CPU(регистр кодов условия или слово состояния программы), или очисткой регистра маски прерывания, или схемным путем. Если запрет прерывания достигнут установкой бита запрета прерывания или очисткой регистра маски, то запрет продолжается до тех пор, пока программа не изменит соответствующие биты в регистре. Если запрет осуществляется схемным путем, то его длительность обычно соответствует одной команде после опознавания прерывания. Если состояние запрета прерываний необходимо на время выполнения нескольких команд, программист должен позаботиться о том, чтобы первой командой обслуживания прерывания была команда «Запрет прерываний».

В машинах, использующих опрос устройств, программа опроса устанавливает регистр маски прерывания так, что программа не прерывается устройствами одного и того же приоритетного уровня, но

может быть прервана устройствами более высокого приоритетного уровня. Машины с одноуровневой системой прерывания выполняют опрос устройств без прерываний, так как прерывание может привести только к повторному запуску программы опроса.

Если внешнее устройство закончило передачу своих данных, оно запрашивает прерывание для того, чтобы информировать об этом CPU, если у последнего есть для него другая работа. Если другой работы не ожидается, CPU запрещает внешнему устройству запрашивать прерывание для информирования о том, что оно свободно. Это проще всего сделать с помощью бита разрешения прерывания в регистре состояния контроллера устройства. Когда программа CPU должна игнорировать запросы на прерывание от какого-либо устройства, она устанавливает бит запрета прерывания в регистре состояния этого устройства. Устройство не будет запрашивать прерывание до тех пор, пока программа снова не установит бит разрешения прерывания.

Контрольные вопросы

1. Как передаются прерывания по шине ввода-вывода?
2. В чем различия между определением источника прерывания программным и аппаратным методами?
3. Как реагирует система на одновременные запросы на прерывание?
4. Какими способами реализуются приоритеты устройств?

Прямой доступ к памяти. В системах с большим объемом данных, передаваемых внешним устройствам или от них, выполнение программ пересылки данных между CPU и внешними устройствами может занимать большую часть времени CPU. В других системах, где пересылка данных способна осуществляться на очень большой скорости, время между отдельными пересылками может оказаться недостаточным для функционирования системы прерывания. В этих случаях выгодно применять специальную аппаратно реализованную схему, позволяющую внешним устройствам передавать данные непосредственно в память или из нее. Программы, исполняемые в CPU, не прерываются на то время, когда внешнее устройство посылает пли получает данные. Все современные вычислительные системы допускают прямой доступ к памяти. В одних случаях это дополнение к базовой системе, а других – часть базовой системы.

Общие характеристики ПДП. Системы, включающие ПДП, отличаются как своими программными средствами, так и техническими от систем без ПДП.

Программные средства. Предположим, что требуется передать N слов данных от внешнего устройства в память системы. Программа обслуживания прерывания для внешнего устройства устанавливает параметры, необходимые для передачи:

а) количество слов данных, которые должны быть переданы;

б) начальный адрес передаваемых слов данных.

Передаются ли данные в память или из нее с восходящими или ни сходящими адресами — вопрос внутренней организации системы. Затем программа обслуживания посылает команду внешнему устройству, которое инициирует передачу, а CPU возвращается назад к прерванной программе. В то время как CPU продолжает выполнять свою программу, внешнее устройство пересылает данные в память или из нее. Когда пересылки данных «внешнее устройство — память» завершатся, устройство снова прервет CPU. Программа обслуживания установит флаг, означающий, что данные находятся в памяти, и программа, требующая эти данные, может быть запущена. Таким образом, программа обслуживания инициирует пересылку данных, но не управляет ею.

Схемные средства. В системе с ПДП память доступна и внешним устройствам, и CPU (а не одному CPU, как в системах без ПДП). Для того чтобы различные устройства имели доступ к памяти, система снабжена некоторыми аппаратно реализованными схемами. Запросы на доступ к памяти подобны запросам к CPU на прерывание. Поэтому система, управляющая запросами на доступ к памяти, обычно очень похожа на систему, управляющую запросами на прерывание CPU. Устройство, требующее доступа к памяти, должно сделать запрос на доступ. Система назначает устройству приоритет, и запрос наивысшего приоритета удовлетворяется. Запросы от CPU на доступ к памяти трактуются так же, как любой другой запрос. Обычно CPU имеет более низкий приоритет, чем внешние устройства. Поэтому в системе с ПДП CPU должен быть устроен таким образом, чтобы он мог временно приостановить работу, если память занята.

Методы организации ПДП. Виды передачи данных при наличии ПДП могут быть различны в разных системах или в одной и той же системе. Эти виды обычно выбираются так, чтобы достичь максимальной пропускной способности системы, что так же важно, как и работа с памятью системы на максимально возможной скорости.

Метод пропуска цикла. Метод пропуска цикла, вероятно, является наиболее общим. При его использовании каждому устройству, включая CPU (рис. 4.10), предоставляется один цикл обращения к памяти. К концу цикла система производит выборку текущих запросов на доступ, а в конце цикла обращения к памяти разрешается доступ устройству с наивысшим текущим приоритетом. Данный метод правильнее назвать одноцикловым в отличие от других, предоставляющих более одного цикла обращения к памяти. Название «пропуск цикла» происходит от того, что запросы на цикл, идущие от CPU, как бы пропускаются.

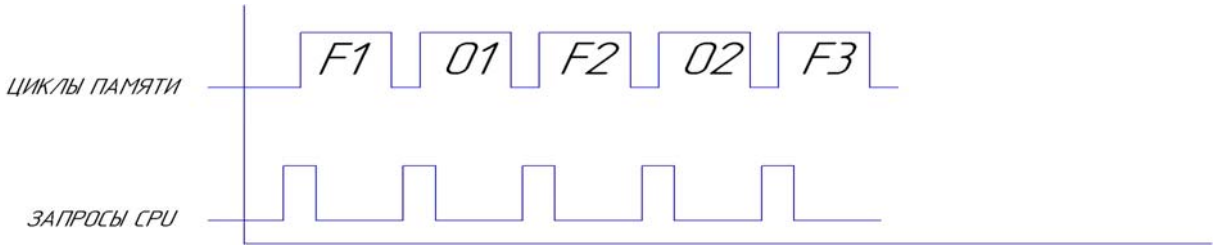


Рис. 4.10. Запросы на циклы обращения к памяти, идущие только от CPU

Рис. 4.11 показывает схему доступа к памяти CPU, выполняющего последовательность команд с обращением к памяти.

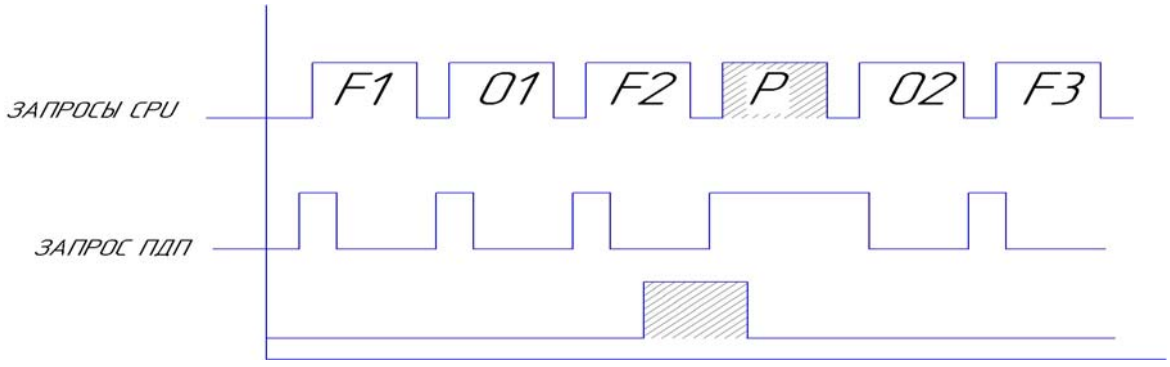


Рис. 4.11. Цикл, захваченный внешним устройством

Предполагается, что аппаратная логика CPU работает быстрее цикла обращения к памяти, так что CPU обусловливает выполнение

операций, связанных с обращением к памяти, почти на максимальной скорости. Это предположение вполне реалистично для большинства систем. Каждая команда CPU требует двух обращений к памяти: F_n — для выборки команды, O_n — для выборки операнда. Конечно, практически для каждой выборки может понадобиться более одного обращения, например для выборки команд увеличенной длины или выборки операнда при использовании косвенной адресации в памяти. Каждый раз, когда CPU запрашивает доступ к памяти, он выставляет запрос на цикл обращения к памяти. Этот запрос подтверждается контроллером доступа к памяти и снимается CPU сразу же после его разрешения. Внешнее устройство в любое время может также запросить доступ к памяти, и если его приоритет выше приоритета CPU, то устройство, как и показано, получает доступ в ближайшем цикле. Запрос от CPU сохраняется до тех пор, пока не получит подтверждения. Заметим, что для получения доступа к памяти внешнее устройство не должно ждать, пока CPU выполнит свои команды. Максимальное время ожидания внешнего устройства, если оно не обладает высшим приоритетом в системе, представляет собой время одного цикла обращения к памяти. В течение времени доступа внешнего устройства CPU бездействует. Если внешнее устройство запрашивает ПДП в то время, когда команда CPU находится в процессе выполнения, то время выполнения этой команды может увеличиться до времени одного цикла обращения к памяти.

Ситуация, показанная на рис. 4.11, в некотором отношении соответствует наихудшему случаю. В любой обычной программе не все команды представляют собой команды обращения к памяти, так что ситуация, при которой память полностью занята CPU, ненормальна. Чаще всего память может быть свободна, и если в это время внешнее устройство делает запрос на цикл, он сразу же может быть разрешен. Хотя можно было бы создать систему, в которой для внешнего устройства отводились бы циклы, не требующиеся CPU, но этого не делают. Внешним устройствам дается возможность ПДП, так как они не способны ждать в течение времени, превышающего очень малое время между пересылками без того, чтобы не вызвать системную ошибку. Поэтому невозможно заставить их ждать неопределенное время, необходимое для выполнения «лишнего» цикла, не требующегося программой в CPU.

Иногда на практике внешнее устройство захватывает цикл тогда, когда CPU мог бы использовать его, а иногда цикл внешнего устройства заставляет CPU ждать время, меньшее времени одного цикла, необходимого для доступа. В других случаях внешнее устройство использует лишний цикл памяти. Эффект действия ПДП на время выполнения команды непредсказуем и, следовательно, непредсказуемо также время выполнения части или всей программы в случае использования ПДП. Программы, в которых требуется точно определять время выполнения, не могут работать совместно с ПДП. В таких системах все измерения в реальном времени должны осуществляться внешними устройствами с прерыванием.

Когда имеет место передача с ПДП, внешнее устройство должно выставлять запрос на использование шины. Если запрос разрешен, устройство посылает адрес в память системы, занимает линию направления, чтобы указать памяти либо на операцию считывания, либо на операцию записи, и затем посылает или получает данные. Эти передачи осуществляются либо на основе получения подтверждения, либо другим способом, который реализован в системе. Описанные выше действия повторяются при каждой пересылке с ПДП, пока все данные не будут переданы, после чего устанавливается бит в регистре состояния внешнего устройства, указывая тем самым на завершение передачи. Обычно при установке этого бита устройство выставляет CPU запрос на прерывание. Программа обслуживания обнаруживает установку бита «Завершение ПДП» в регистре состояния и выполняет соответствующие действия.

В некоторых ранних системах, спроектированных в то время, когда регистровые схемы были дорогими, счетчик слов и адрес данных хранятся не в регистрах внешнего устройства, а в памяти системы. Когда внешнему устройству разрешается передача с ПДП, контроллер памяти производит выборку адреса данных из фиксированной для каждого устройства ячейки. Этот адрес временно хранится в регистре контроллера памяти. Затем осуществляется передача данных и выбирается из памяти счетчик слов, содержимое которого уменьшается и снова записывается в память. Если содержимое счетчика слов становится равным нулю, посылается сигнал контроллеру внешнего устройства, который, в свою очередь, выставляет запрос на прерывание. Контроллер памяти, используя вектор устройства (обычно не-

сколько бит), определяет адрес пары ячеек памяти, содержащих счетчик слов и адрес данных для внешнего устройства. Этот вектор устройства используется так же, как и вектор прерывания в подобной ситуации, которая требует двух или трех циклов обращения к памяти для каждой передачи данных: один цикл для самой передачи данных и два — для управления. В некоторых универсальных вычислительных системах используется именно этот метод, но поскольку счетчик слов и адрес данных могут быть записаны в таких системах в одном слове, для передачи данных потребуются только два цикла обращения к памяти.

Пакетный режим передачи. В некоторых случаях можно достигнуть большего эффекта, если предоставить устройству доступ к памяти на время выполнения некоторого числа последовательных циклов, вместо того чтобы позволить ему вступить в «конкурентную борьбу» с другими устройствами на относительно большой период времени. В такой системе внешнее устройство может получить доступ к памяти и передать пакет данных на максимальной скорости. Если внешнее устройство не может получать или принимать данные на той максимальной скорости, на которую способна память, то выигрыша при использовании этого способа передачи с ПДП не будет. Внешнее устройство, которое не может поддерживать максимальную скорость передач, должно позволить другим устройствам получить циклы обращения к памяти. Если одно устройство имеет доступ к памяти, система приостанавливает работу, пока это устройство не освободит память. Однако если основная память системы имеет много входов, то пакетный решим передач через один вход не нарушает работы остальной части системы. Пакетный режим используется, например, для быстрых передач типа «память – память» между памятью системы и локальной памятью устройств, снабженных буферами, если обе памяти могут работать на одной и той же скорости.

Реализация ПДП. Передачи с ПДП выполняются по системе каналов, подобных шине системы ввода-вывода. Следовательно, в тех системах, где память присоединена к CPU специальной шиной памяти, ПДП выполняется на специальном канале ПДП, который связывает все внешние устройства, но обходит CPU. Обычно лишь немногие

внешние устройства присоединены к каналу ПДП. В подобных системах ПДП обычно является дополнительным устройством, так как аппаратура ПДП может быть совершенно отделена от технических средств остальной системы.

В тех вычислительных системах, где память присоединена к шине ввода-вывода, передачи с ПДП обычно используют стандартные линии шины ввода-вывода. Для управления этими передачами предназначено несколько дополнительных линий. Например, некоторые микропроцессоры снабжены специальной линией управления, с помощью которой внешние устройства с ПДП могут заставить CPU приостановить работу и освободить шину ввода-вывода. По этой линии можно заблокировать схемы шины в CPU, и тогда управление шиной перейдет к контроллеру.

Автономный блок ПДП. Некоторые системы имеют канал ПДП, совершенно независимый от шины ввода-вывода. Этот канал включает:

а) линии данных, по одной для каждого бита в слове;

б) линии адреса, обычно 16 для адресации памяти объемом в 64 К;

в) линии управления – указания направления, «рукопожатия», запроса на ПДП и подтверждения.

Если к каналу ПДП присоединено больше одного устройства, необходимо определить значения приоритетов, чтобы одновременные запросы могли быть удовлетворены определенным образом. В некоторых системах это достигается тем, что каждое устройство снабжено отдельной линией запроса, обусловливающей создание многоуровневой системы. Другие системы содержат одну линию запроса и определяют приоритеты с помощью линии подтверждения, идущей по цепочке. Последний метод используется в современных системах чаще.

Контрольные вопросы

1. Какие существуют методы организации ПДП?
2. В чем заключается метод пропуска цикла?
3. Как выполняется передача данных с ПДП?
4. Что включает в себя автономный блок ПДП?

Глава 5

МИКРОПРОЦЕССОР

Микропроцессор с момента появления претерпел множественные изменения. Менялась тактовая частота, появляется встроенная память, увеличивается число внутренних регистров и разрядность шины данных, изменяется система команд и способы адресации, но в сути своей он остается тем же самым вычислительным устройством с базовым наборов функциональных элементов.

Для изучения структуры и принципов работы целесообразно рассматривать более простой процессор, из линейки x86 ранних версий, лишенный множества элементов, хотя и существенных, но все же затрудняющих понимание его работы.

Арифметико-логическое устройство (АЛУ) - центральная часть процессора, выполняющая арифметические и логические операции.

АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО

АЛУ реализует важную часть процесса обработки данных. Она заключается в выполнении набора простых операций. Операции АЛУ подразделяются на три основные категории: арифметические, логические и операции над битами. Арифметической операцией называют процедуру обработки данных, аргументы и результат которой являются числами (сложение, вычитание, умножение, деление). Логической операцией именуют процедуру, осуществляющую построение сложного высказывания (операции И, ИЛИ, НЕ). Операции над битами обычно подразумевают сдвиги.

АЛУ состоит из регистров, сумматора с соответствующими логическими схемами и элемента управления выполняемым процессом. Устройство работает в соответствии с сообщаемыми ему именами (кодами) операций, которые при пересылке данных нужно выполнить над переменными, помещаемыми в регистры.

Арифметико-логическое устройство функционально можно разделить на две части:

а) микропрограммное устройство (устройство управления), задающее последовательность микрокоманд (команд);

б) операционное устройство (АЛУ), в котором реализуется заданная последовательность микрокоманд (команд).

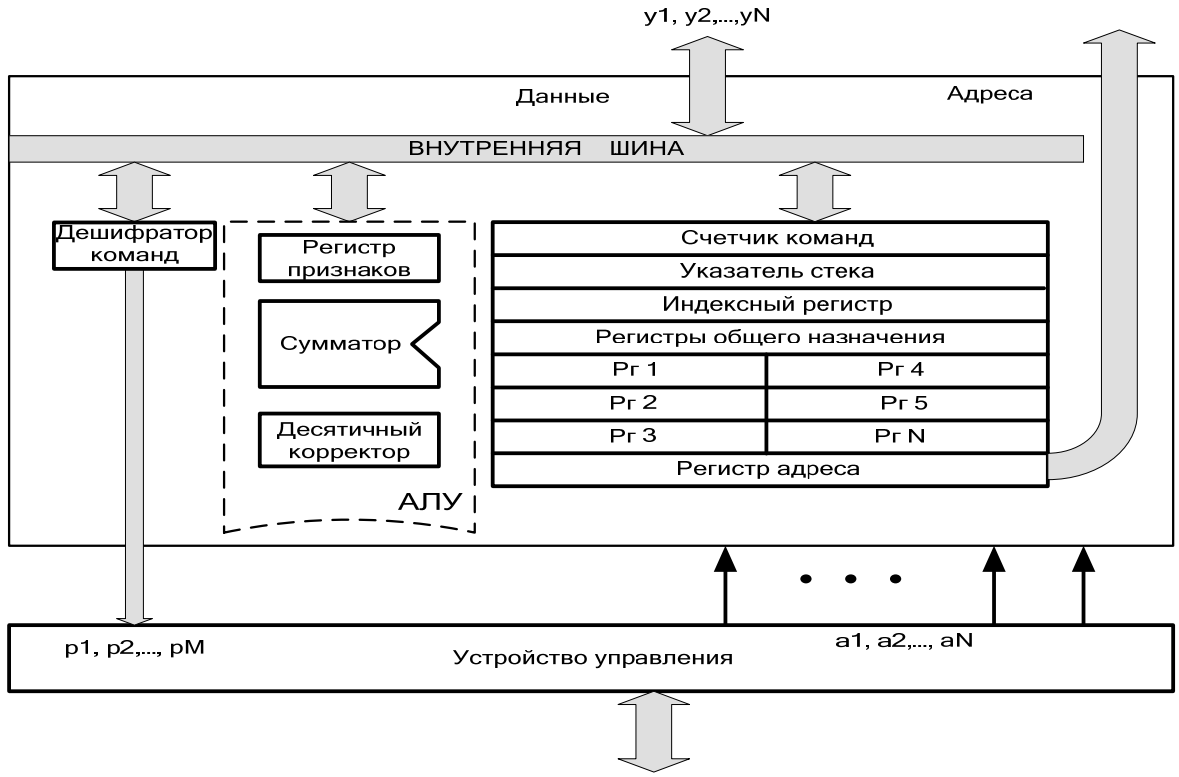


Рис. 5.1. Архитектура микропроцессора

Структурная схема АЛУ и его связь с другими блоками машины показаны на рис. 5.1. В состав АЛУ входят регистры Рг1 – Рг7, в которых обрабатывается информация, поступающая из оперативной или пассивной памяти N1, N2, ...NS; логические схемы, реализующие обработку слов по микрокомандам, поступающим из устройства управления.

Закон переработки информации задает микропрограмма M, которая записывается в виде последовательности микрокоманд A1,A2, ..., An-1,An. При этом различают два вида микрокоманд: внешние, т. е. такие, которые поступают в АЛУ от внешних источников и вызывают в нем те или иные преобразования информации (на рис. 5.1 микрокоманды A1,A2,..., An), и внутренние, которые генерируются в АЛУ и воздействуют на микропрограммное устройство, изменяя естественный порядок следования микрокоманд. Например,

АЛУ может генерировать признаки в зависимости от результата вычислений j, w, Q и др. (j – признак переполнения, w – признак отрицательного числа, Q – признак равенства 0 всех разрядов числа). На рис. 2.1 эти микрокоманды обозначены p1, p2,..., pm.

Результаты вычислений из АЛУ передаются по кодовым шинам записи y1, y2, ..., ys, в ОЗУ.

Функции регистров, входящих в АЛУ:

Рг1 – сумматор (или сумматоры) – основной регистр АЛУ, в котором образуется результат вычислений;

Рг2, Рг3 – регистры слагаемых, сомножителей, делимого или делителя (в зависимости от выполняемой операции);

Рг4 – адресный регистр (или адресные регистры), предназначен для запоминания (иногда и формирования) адреса операндов и результата;

Рг6 – k индексных регистров, содержимое которых используется для формирования адресов;

Рг7 – вспомогательные регистры, которые по желанию программиста могут быть аккумуляторами, индексными регистрами или использоваться для запоминания промежуточных результатов.

Часть операционных регистров является программно-доступной, т.е. они могут быть адресованы в команде для выполнения операций с их содержимым. К ним относятся сумматор, индексные регистры, некоторые вспомогательные регистры.

Остальные регистры программно-недоступные, так как они не могут быть адресованы в программе. Операционные устройства можно классифицировать по виду обрабатываемой информации, способу обработки информации и логической структуре.

АЛУ десятично-двоичных чисел. Это арифметико-логическое устройство применяется в дешевых моделях микрокалькуляторов. АЛУ в этих приборах имеет более сложный характер. Во первых, оно состоит из нескольких блоков по четыре разряда, а во-вторых, автоматически осуществляется так называемая десятичная коррекция чисел. Например:

8+9=17=10+7 1000+1001=10001=1010+0111
9+9=18=10+8 1001+1001=10010=1010+1000
5+4=9=0+9 0101+0100=1001=0000+1001
8+7=15=10+5 1000+0111=1111=1010+0101

Как видим, происходит лишь небольшое усложнение АЛУ, а все остальные операции остаются теми же, что и в целочисленном АЛУ. Из-за усложнения двоично-десятичных арифметико-логических устройств они работают гораздо медленнее, чем аналогичные целочисленные АЛУ и АЛУ с плавающей точкой. Однако при этом устройство управления процессором упрощается за счет отсутствия команд конверсии двоичных чисел в двоично-десятичные и обратно.

АЛУ для чисел с плавающей точкой. При проведении операций с плавающей точкой логика расчетов усложняется. Дело в том, что операции приходится выполнять на числах, имеющих не только разные мантиссы, но и разные порядки. Поэтому перед проведением операций над вещественными числами нужна нормализация, т.е. приведение двух вещественных чисел к одному порядку (обычно большему по величине из двух чисел). Для этих целей в арифметико-логическом устройстве с плавающей точкой отдельно производятся действия с порядком, отдельно - с мантиссой. Нормализация происходит следующим образом.

Находится разность порядков большего и меньшего числа. Мантисса меньшего числа сдвигается вправо на число бит, равное разности, полученное на шаге 1.

После этого производятся обычные целочисленные операции с мантиссой. Далее, после получения результата вычислений иногда производится коррекция мантиссы числа с плавающей точкой. Алгоритм коррекции следующий: убираются все незначащие нули в левой части мантиссы. Для этого осуществляется сдвиг влево мантиссы на n разрядов (n – число незначащих нулей слева). После этого число n вычитается из порядка.

Как правило, операцию коррекции вызывают принудительно, а не запускают автоматически.

При работе этого устройства необходимо, чтобы ему правильно передавался и порядок, и мантисса числа. Именно поэтому в большинстве устройств для проведения операций с плавающей точкой все операнды и результаты, а также промежуточные числа хранились в единообразной форме. Обычно ею является формат вещественных чисел с расширенной точностью длиной 80 бит (10 байт). Преобразование чисел в этот формат и из него в формат других вещественных и целых чисел осуществляется устройством управления сопроцессора.

Целочисленное АЛУ. Целочисленное арифметико-логическое устройство является, наверное, первым универсальным АЛУ. Это АЛУ могло работать с целыми числами и вещественными числами с фиксированной точкой.

Несмотря на большое число команд микропроцессора, это устройство фактически все команды сводит к девяти элементарным операциям. Все они приведены в табл. 5.1.

Таблица 5.1

Элементарные операции АЛУ

Операция	Обозначение	Количество операндов	Подсистема выполнения
Сложение	+	2	Сумматор
Вычитание	-	3	Сумматор и регистр
Логическое умножение	И,&, and^	2	Логические схемы
Логическое сложение	ИЛИ, V, 1, or	2	Логические схемы
Сдвиг влево	<<	2	Регистр
Сдвиг вправо	>>	2	Регистр
Инверсия (НЕ)	not	1	Логические схемы
Увеличение на 1, инкремент	++,inc	1	Сумматор
Уменьшение на 1, декремент	--,dec	1	Сумматор

Именно эти операции выполняются за один такт микропроцессора (см. синхронизирующие импульсы, тактовую частоту) и имеют наибольшую скорость выполнения. Они являются единственными командами для MISC-процессоров. Фактически все другие операции осуществляются с помощью этих девяти базовых. Так, умножение восьмиразрядных целых чисел A и B выполняется по следующему алгоритму:

1. Обнуляется результат.
2. Если последний разряд числа B – единица, то к результату прибавляется число A.

3. Число A сдвигается на разряд влево, а число B – на разряд вправо.

Повторяются шаги со второго по третий семь раз.

Заметим, что сдвиг влево на один разряд соответствует умножению на два, а сдвиг вправо на один разряд – целочисленному делению на два.

Команда изменения знака числа будет следующей:

Вначале происходит инверсия числа.

После этого производится инкремент результата (т.е. к нему прибавляется единица).

Таким образом, число переводится в дополнительный код. Команда определения знака числа основывается просто на проверке самого старшего бита.

Контрольные вопросы

1. Какие операции реализует АЛУ?
2. Каковы функции регистров, входящих в АЛУ?
3. За счет чего упрощается устройство управления процессором в АЛУ десятично-двоичных чисел?
4. Как происходит нормализация в АЛУ для чисел с плавающей точкой?
5. Перечислите элементарные операции АЛУ.

Элементная база АЛУ. На уровне логических схем АЛУ состоит из логических элементов, сумматоров, триггеров и некоторых других элементов.

Логический элемент - электронная схема, реализующая элементарную переключающую функцию. При реализации функций переключения входные переменные соответствуют входным сигналам, а выходной сигнал представляет собой значение функции. Всего существует десять логических элементов, реализующих десять логических (элементарных или сложных) функций.

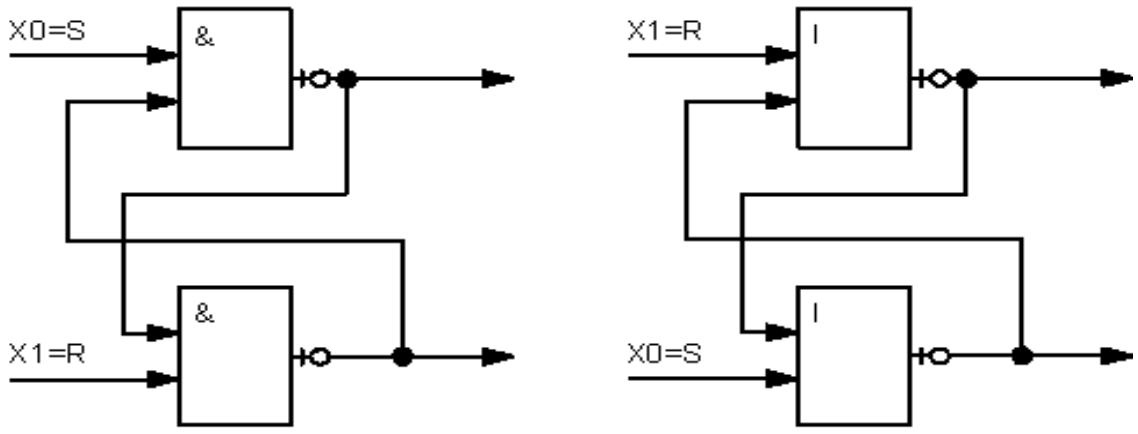
Логическая схема может реализовать сложную функцию алгебры логики, а может входить в состав другого функционального блока процессора (сумматора, дешифратора, регистра, триггера).

Триггер - электронная схема с двумя устойчивыми состояниями, предназначенная для хранения одного бита информации. Триггер пе-

реходит из одного устойчивого состояния в другое при воздействии некоторого входного сигнала. Триггер имеет вход для установки в состояние 0 (X0) и в 1 (X1). На выходе выдается состояние триггера в прямом (Y) и в инверсном (Y1) виде. В компьютерах используют синхронизируемые и несинхронизируемые триггеры. Синхронизируемый триггер – это триггер, изменение состояния которого осуществляется только в момент подачи сигнала синхронизации V.

В зависимости от способа управления различают несколько типов триггеров: D- (с одним входом); RS- (с двумя входами); T- (со счетным входом); RST- (с двумя входами и счетным выходом) триггеры, и универсальные триггеры: JK- и DF-триггеры.

Триггеры, как правило, выполняются на логических элементах ИЛИ-НЕ, И-НЕ (рис. 5.2). Если триггеры выполняются на логических элементах И-НЕ, то это триггеры с инверсным управлением; если на элементах ИЛИ-НЕ, то это триггеры с прямым управлением.



а) *б)*

Рис. 5.2. Схема реализации триггера – защелки на элементах И-НЕ (а) и ИЛИ-НЕ (б).

RS-триггер - двухвходовый триггер с раздельными входами для установки в 0 или 1 (рис. 5.3). При подаче единичного сигнала на вход R (-X0) триггер переходит в состояние 0 (Y=0, Y1=1), а при подаче на вход S (=X1) единичного сигнала – в состояние 1 (Y=1, Y1=0). Одновременная подача единичного сигнала на оба входа запрещена. Обычно RS-триггеры бывают синхронизируемыми (вход для синхронизации - V).

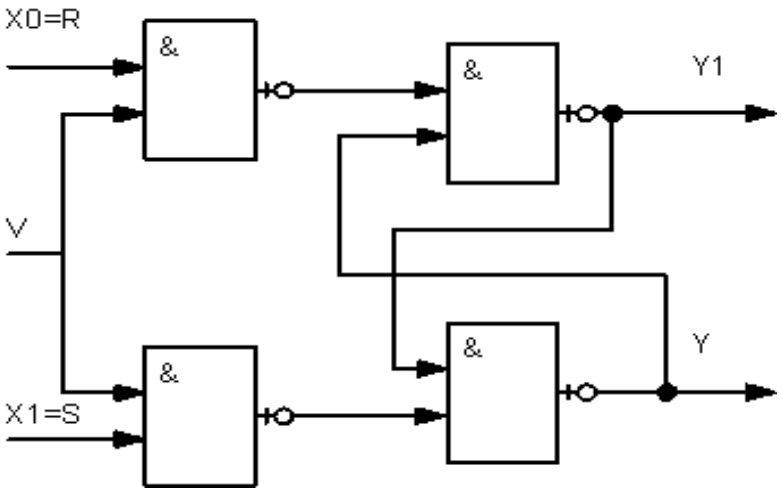


Рис. 5.3. Схема реализации RS-триггера на элементах И-НЕ

Т-триггер - одновходовый триггер со счетным входом: информация подается одновременно на два входа. При подаче сигнала состояние триггера меняется на противоположное (рис. 5.4). Он, как правило, является несинхронизируемым и позволяет не только хранить информацию, но и осуществлять сложение по модулю 2.

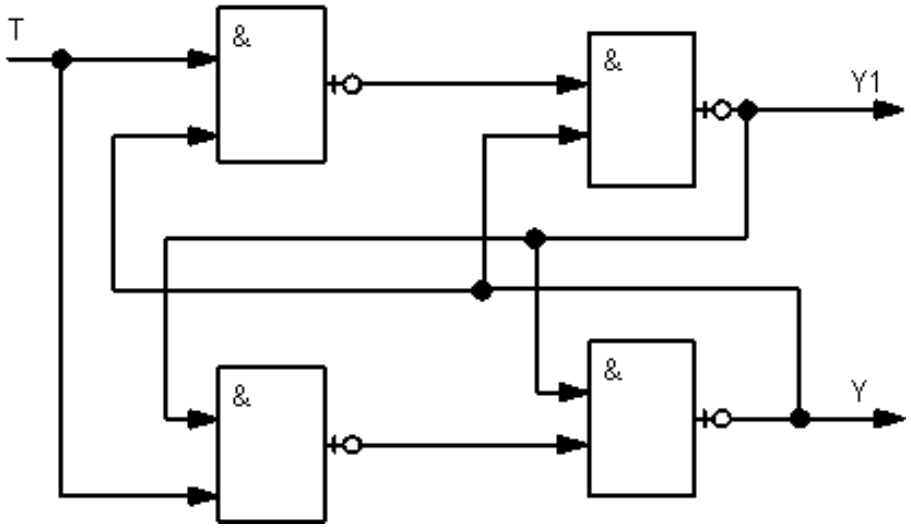


Рис. 5.4. Схема реализации Т-триггера

D-триггер выполняет функцию задержки входного сигнала на один такт синхронизации (рис. 5.5). Сигнал, появившийся на входе D (=X0) в момент времени T, задерживается в нем и появляется на выходе Y в момент времени T+1.

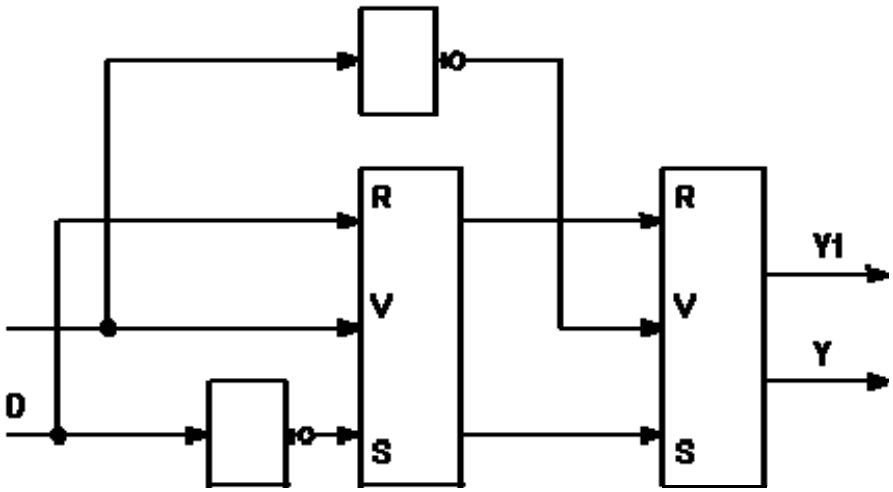


Рис. 5.5. Схема реализации D-триггера

JK-триггер – двухвходовый триггер, допускающий раздельную установку состояния 0 и 1, а также смену текущего состояния (режим со счетным входом), осуществляемую при подаче на оба входа единичного сигнала. Вход K в этом триггере соответствует входу R (=X0) RS-триггера, а вход J – S (=X1).

DF-триггер – двухвходовый триггер, позволяющий по одному входу реализовать режим D-триггера, а по другому – модифицировать режим работы. Вход D соответствует X1, а F – X0. При F=0 DF-триггер сохраняет текущее состояние. Сигнал F=1 устанавливает триггер в состояние 0. При D=1 и F=1 триггер устанавливается в состояние 1.

Триггеры с неустойчивыми состояниями называются вибраторами. Схема с одним неустойчивым состоянием (триггер Шмидта, одновибратор) генерирует импульсный сигнал определенной длительности. Схема с двумя неустойчивыми состояниями называется мультивибратором и служит для генерации последовательности прямоугольных сигналов. Он используется тактовым генератором.

Регистр – схема для приема, хранения и передачи *n*-разрядного блока данных Он используется для промежуточного хранения, сдвига, преобразования и инверсии данных. Регистры выполняются на триггерах и логических элементах. Их число и тип определяются разрядностью слова и назначением регистра. Если регистр не требует предварительного сброса данных (т.е. установки всех его ячеек в нуль), то новые данные заменяют в нем старые. Схема регистра показана на рис. 5.6.

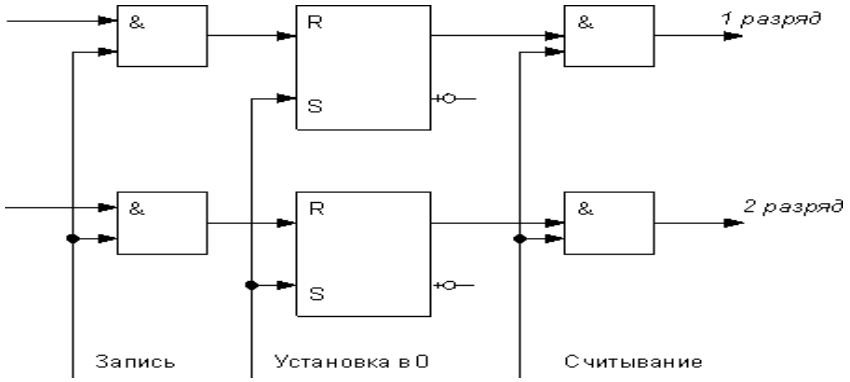


Рис. 5.6. Реализация регистра

Сумматор - схема, выполняющая операцию арифметического сложения двух чисел. Различают одно- и многоразрядный (на всю длину суммируемых слов) сумматоры, полный сумматор (с приведением переносов) и полусумматор (с запоминанием переносов).

В зависимости от числа входов сумматоры бывают двухвходовыми и трехвходовыми. В двухвходовом одноразрядном сумматоре на вход подаются два разряда суммируемых чисел, а на выходе формируется результат сложения разрядов по модулю 2 и перенос в следующий разряд. В трехвходовом одноразрядном сумматоре на вход подаются два разряда суммируемых чисел и результат переноса из суммирования предыдущих разрядов, а на выходе – результат суммирования по модулю 2 и перенос в следующий разряд (рис. 5.7).

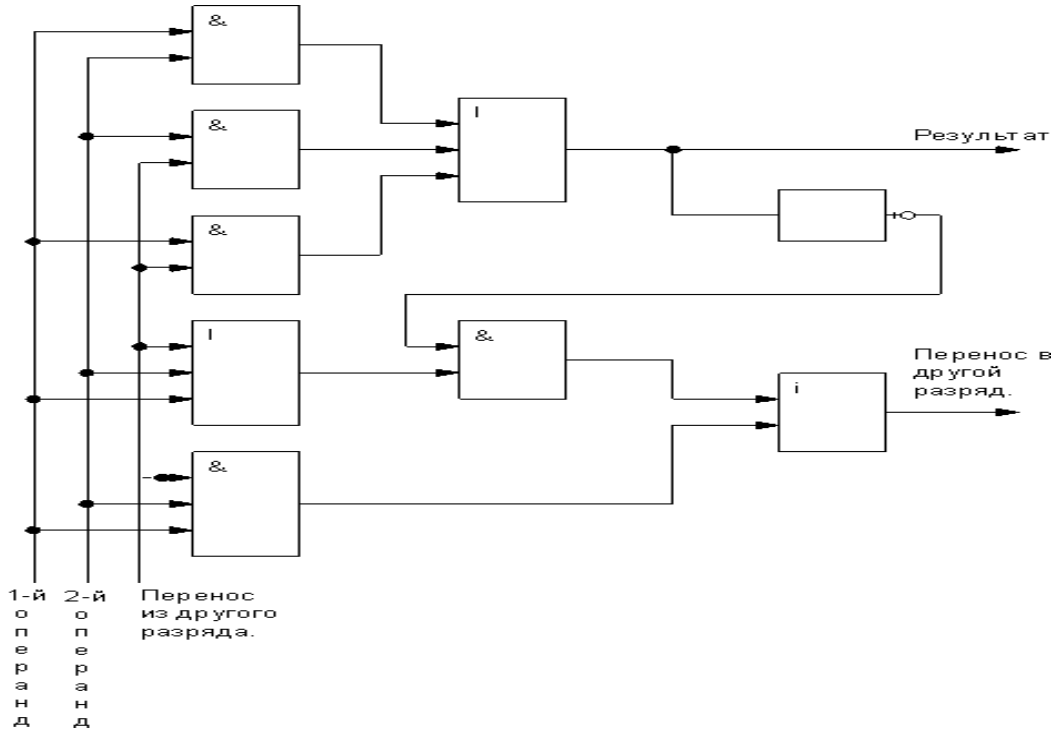


Рис. 5.7. Реализация одноразрядного сумматора с переносом знака

Дешифратор – логическая схема, преобразующая входное *n*-разрядное двоичное слово в единичный сигнал на одном из 2^n входов. Обратную функцию выполняет схема, называемая шифратором. Дешифраторы широко используются в устройствах управления для управления работой микропроцессорами.

Коды условий – это флаги, автоматически устанавливаемые в 1 или 0 в соответствии с состоянием АЛУ. Обычно они отражают состояние слова на выходе из АЛУ в конце выполнения команды, так как большинство CPU производит передачи данных в основном через АЛУ. Флаги условий изменяются от машины к машине, но четыре флага:

– флаг нуля (Z);
– флаг отрицательного значения (N);
– флаг переноса (C);
– флаг переполнения (V)

представлены в большинстве типов ЭВМ. Некоторые случаи их применения рассмотрены в этой главе.

Контрольные вопросы

1. На каких логических элементах выполняются триггеры?
2. Как называется триггер с неустойчивым состоянием?
3. Для чего используются регистры?
4. Где нашли широкое применение дешифраторы?

Флаг нулевого значения. Флаг нуля Z устанавливается в 1, если результат операции АЛУ нуль; в противном случае он сбрасывается в нуль. Большинство команд, выполняющих передачу данных через АЛУ, воздействует на состояние Z-флага.

В системе команд обычно есть команда условного перехода «Перейти, если результат предыдущей команды нуль». Этот тип команды (переход по нулю) проверяет состояние Z-флага и осуществляет переход, если флаг установлен в 1. Вариации этой команды, такие как «Перейти, если не нуль», «Перейти к подпрограмме, если нуль» (или не нуль), «Возврат из подпрограммы, если нуль» (или не нуль), проверяют состояние Z-флага и производят соответствующий переход. Команды типа «Перейти, если содержимое регистра R равно нулю» обычно передают содержимое регистра через АЛУ для уста-

новки флагов и затем производят переход согласно состоянию Z-флага. Заметим, что команды условного перехода используют состояние флагов, но, как правило, их не меняют.

В следующих разделах описывается использование Z-флага совместно с другими флагами.

Флаг отрицательного значения. Флаг отрицательного значения (N) отражает состояние самого старшего значащего бита на выходе из АЛУ в конце операции. Предполагается, что слово на выходе АЛУ представлено двоичным дополнением. Следовательно, если знаковый бит числа 1, то N-флаг устанавливается в 1, в противном случае он сбрасывается в нуль.

Флаг используется для команд условного перехода, во многом таких же, как и при использовании Z-флага. Проверка в этом случае производится на отрицательный или неотрицательный результат. Этот флаг также используется в сочетании с другими флагами, как описано ниже.

Флаг переноса. Флаг переноса (C) устанавливается, если есть перенос из старшего значащего бита во время выполнения арифметической или логической операции АЛУ, а также тогда, когда операция вычитания предполагает производить заем от бита более высокого порядка, чем старший значащий бит слова (MSB).

Во многих ЭВМ команды циклического сдвига, сдвига вправо и влево выдвигают бит из слова в бит переноса. Это целесообразно при работе с операндами увеличенной длины.

Флаг переполнения. Флаг переполнения (V) используется для различных целей при разных обстоятельствах в различных типах ЭВМ. Он устанавливается в 1, когда результат арифметической операции находится вне диапазона машинного слова.

В некоторых ЭВМ этот флаг не может быть сброшен в нуль аппаратурой АЛУ, когда она обнаруживает, что число лежит в пределах диапазона слова. Такой тип V-флага обычно используется в операции логического умножения с битом разрешения прерывания в регистре кодов условия. Когда результат операции И равен 1, запрашивается прерывание. Этот бит разрешения при переполнении действует точно

таким же образом, как и бит разрешения прерывания в регистре состояния внешнего устройства. Обычно прерывания этого типа, называемые *внутренними прерываниями,* имеют более высокий приоритет, чем запросы на прерывание от внешних устройств. Для программирования программы обслуживания прерывания при переполнении, способной совершать любые действия, использована стандартная система приоритетов прерывания. Этот тип флага переполнения очень полезен при выполнении арифметических вычислений; он обеспечивает почти мгновенную реакцию, когда числа выходят из нужного диапазона.

В других ЭВМ флаг переполнения устанавливается в нуль, как объяснено выше, но сбрасывается в нуль, когда результат следующей арифметической операции лежит в пределах диапазона машинного числа. Этот тип флага переполнения обычно не связан с системой прерываний. Он используется для обнаружения условий перехода, когда необходимо знать условия, установленные предыдущей командой в программе.

Лучшие вычислительные системы имеют оба типа флага переполнения, хотя часто не каждый из них доступен для использования программистом.

Другие флаги. Некоторые типы ЭВМ имеют флаги для указания таких условий, как четность последнего операнда, пересылаемого в машине, флаги переполнения и исчезновения порядка для операций с плавающей точкой и другие специальные флаги, полезные в отдельных системах.

Случается, что некоторые ЭВМ не содержат даже четырех флагов: Z, N, C, V. Однако это может оказаться обманчивой видимостью. Программисту обычно разрешено обрабатывать флаги только в тех машинах, где их можно изменять командами, входящими в состав системы команд. Это команды типа «Установить V», «Сбросить С» или «Передать содержимое регистра К в регистр кодов условия». Подобные команды могут отсутствовать в некоторых системах команд, содержащих, однако, команды типа «Перейти, если содержимое регистра К равно нулю». Хотя в подобной машине может и не быть Z-флага, нужного программисту, такой флаг существует в аппаратуре и используется в ней для выполнения команды условного перехода.

Об этом не упоминается в литературе для пользователей, но такие флаги можно найти в схемах CPU.

В других ЭВМ Z- и N-флаги заменены или сосуществуют с флагами, указывающими отношение величин двух операндов, введенных в АЛУ. Это флаги «Больше, чем» (>) или «Меньше, чем» (<), которые устанавливаются в 1 или сбрасываются в нуль комбинациями V, Z и N.

Перенос и переполнение при исполнении команд арифметики. В малых машинах арифметические расчеты часто требуют выполнения операций со словами двойной длины или большей. Программирование этих операций может быть значительно облегчено, если команды сложения, флаги переноса и переполнения имеют необходимые характеристики. Рассмотрим сложение двух чисел тройной длины, как показано на рис. 5.8. В системе команд СРТЛ нет команды сложения слов тройной длины, таким образом, программирование этой операции должно выполняться обычными командами сложения.

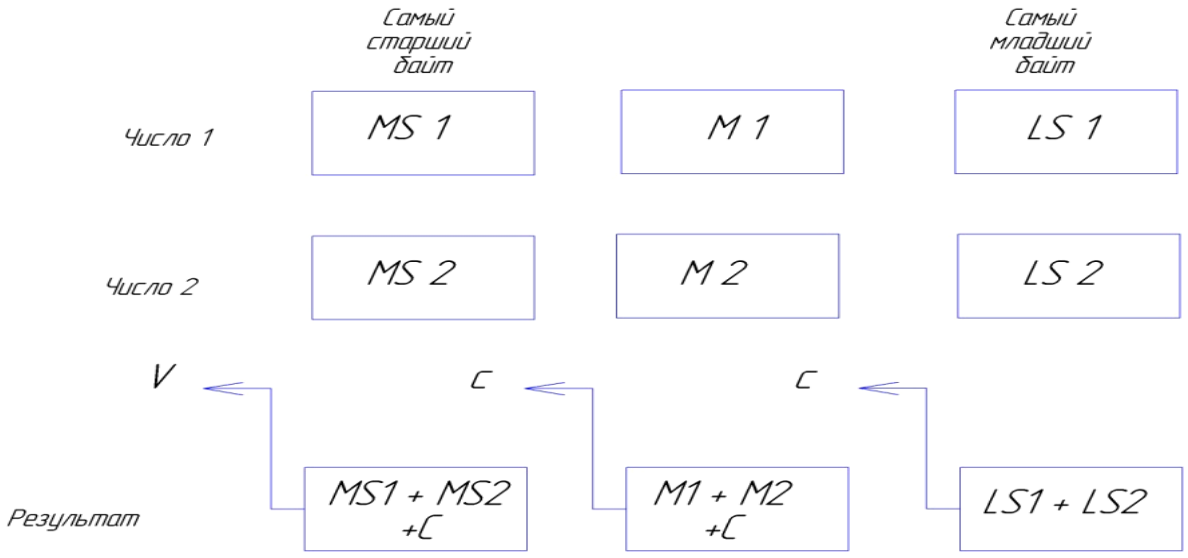


Рис. 5.8. Сложение слов тройной длины

Сложение осуществляется в три этапа:

Этап 1. Сложение самых младших байтов, которые могут установить соответствующий флаг переноса.

Этап 2. Сложение средних байтов плюс перенос от сложения на этапе 1. Это сложение также может установить соответствующий флаг переноса.

Этап 3. Сложение самых старших байтов плюс перенос от сложения на этапе 2. Это сложение может установить флаг переполнения, если результат будет превышать диапазон 3-байтового слова. Фактически это условие обычного переполнения.

Во многих системах команд ЭВМ имеются различные команды сложения, с помощью которых программируются все три этапа, приведенные выше.

Этап 1 требует использования команд:

Прибавить А к В. Не прибавлять перенос.
Установить или сбросить флаг переноса.
Переполнение несущественно.

Этап 2 требует использования команд:

Прибавить А к В. Прибавить также перенос.
Установить или сбросить флаг переноса.
Переполнение несущественно.

Этап 3 требует использования команд:

Прибавить А к В. Прибавить также перенос.
Установить или сбросить флаг переполнения.
Перенос несуществен.

Такие команды сложения имеются в системах команд ряда ЭВМ; некоторые из них включены в системы команд малых машин. Однако часто при работе с малыми машинами программист должен сам проверять положение флагов переполнения и переноса, а также использовать условные переходы для команд, выполняющих прибавление переноса, когда это необходимо.

При вычитании слов увеличенной длины необходимо выполнять аналогичные правила. Флаг переноса в этом случае устанавливается при заеме из более старшего байта. Средний и самый старший байты вычитаются с помощью команд вычитания байтов и переноса.

Применение флагов в операциях выравнивания. В машинах, где операция выравнивания должна программироваться, флаги могут быть использованы для обнаружения конца этой операции. Предположим, положительное число нужно выровнять по левому разряду,

для чего его можно сдвигать влево до тех пор, пока оно не станет отрицательным. Далее сдвигаем число вправо на один разряд и получаем выровненный формат. При каждом сдвиге влево порядок уменьшается на единицу и увеличивается на единицу при каждом сдвиге вправо.

Флаг переноса и сдвиги операндов увеличенной длины. Для операндов увеличенной длины требуются точно такие же действия, что и для операндов единичной длины. Во многих ЭВМ команды сдвига действуют так, как если бы флаг переноса был бы частью сдвигаемого слова. Бит, выдвигаемый из машинного слова как при левом, так и при правом сдвиге, становится битом переноса. При циклических сдвигах бит переноса считается размещенным между концами сдвигаемого слова. Все это очень полезно при программировании сдвигов операндов двойной длины или большей. Например, требуется сдвинуть на один разряд влево операнд двойной длины в машине с длиной слова в 4 бит. Один из методов показан на рис. 5.9. Этот метод использует команду левого циклического сдвига, при котором флаг переноса находится между битами концов сдвигаемого слова.

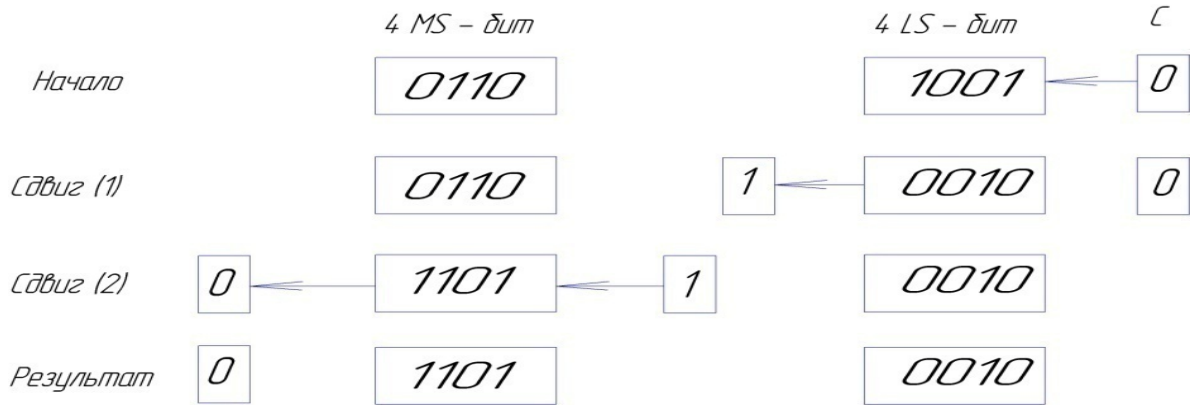


Рис. 5.9. Сдвиг влево слова двойной длины

Для каждого сдвига операнда двойной длины на один разряд требуются три команды:

1. Сбросить флаг переноса в нуль.

2. Циклический сдвиг влево на один разряд младшей части слова. Бит переноса (0) сдвигается в младший разряд младшей части слова. Одновременно старший бит этой части слова сдвигается в бит переноса.

3. Циклический сдвиг влево на один разряд старшей части слова. При этом старший бит младшей части слова, находящийся в бите переноса, сдвигается в младший разряд старшей части слова. Одновременно старший бит слова двойной длины сдвигается в бит переноса, но это уже не имеет отношения к результату. Этот пример показывает, что программист в состоянии решать задачи, используя обычный набор команд, если нет команд, необходимых непосредственно ему.

Одно из главных отличий основных наиболее распространенных больших ЭВМ от малых при программировании на уровне языка ассемблера заключается в том, что первые обладают очень широким набором команд. Система команд малых машин весьма ограничена.

Арифметическое сравнение (числа со знаком). При арифметических действиях часто требуется сравнить два числа и выбрать дальнейшие действия в зависимости от результата этого сравнения. Условные переходы типа «Перейти, если A>B» или «Перейти, если A<B» часто включены в системы команд ЭВМ. Сравнение двух чисел выполняется вычитанием одного числа из другого и установкой кодов условия, соответствующих результату (табл. 5.2).

Таблица 5.2

Сравнение чисел

A	B	A-B	Результат	N	V	Результат	Z	Результат	
100 -80	80 -100	+20 +20	A>B A>B	0 0	0 0	N'.V'=1 N'.V'=1	0 0	(N'.V'). Z'=1	(N исключающее ИЛИ V)'. Z=1
80 100	-100 -80	+180 +180	A>B A>B	1 1	1 1	N.V=1 N.V=1	0 0	(N.V). Z'=1	
100 -80 80 -100	100 -80 80 -100	0 0 0 0	A=B A=B A=B A=B	0 0 0 0	0 0 0 0	N'.V'=1 N'.V'=1 N'.V'=1 N'.V'=1	1 1 1 1	(N'.V'). Z=1	Z=1
80 -100 -80 -100	100 -80 100 +80	-20 -20 -180 -180	A<B A<B A<B A<B	1 1 0 0	0 0 1 1	N.V'=1 N.V'=1 N'.V=1 N'.V=1		N.V'=1 N'.V=1	N Исключающее ИЛИ V=1

Многие системы команд имеют команду «Сравнить», которая выполняет все вышеуказанные действия, но не помещает результат в ячейку памяти или регистр. Оба операнда остаются неизменными.

Условия A>B или A<B могут быть получены из установки флагов N и V при условии, что флаг V не того типа, который остается установленным раз и навсегда. Флаг V, используемый для этих целей, должен устанавливаться в 1 или сбрасываться в нуль автоматически в зависимости от результата каждой арифметической операции. На рис. 5.10 показана таблица результатов вычитания двух чисел с десятичными величинами 100 и 80 соответственно в 8-битовом АЛУ. Числовой диапазон этой машины от +127 до —127, так что сумма чисел 180 дает переполнение. Таблица результатов поясняется рис. 5.11, где показаны четыре области машинных чисел, из которых получаются комбинации N и V, показанные в табл. 5.2. Заметим, что в областях переполнения знак, обнаруживаемый машиной, противоположен знаку правильного результата.

Результаты, показанные на рис. 5.10, могут быть обобщены.

1. Результат вычитания A−B, где A<B, есть отрицательное число.

Это число находится в машине:

а) как отрицательное и в области машинных чисел;

б) положительное и в области переполнения.

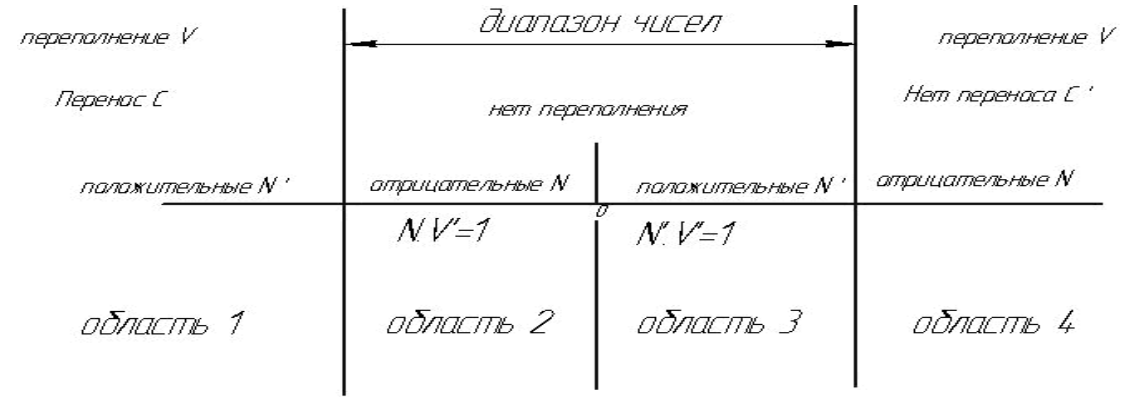


Рис. 5.10. Диапазоны машинных чисел

Эти условия применимы ко всем числам в машине и могут быт записаны следующим образом:

A<B, (если NV' = 1) ИЛИ (]N'V'=1),

т.е. A<B, если N $\oplus$V=1

2. Результат вычитания двух чисел A–B, где A>B, – положительное число, большее нуля. Это число находится в машине:

а) как положительное и в ненулевой области машинных чисел;

б) отрицательное и в области переполнения.

В этом случае число не может быть нулем.

Эти условия могут быть записаны следующим образом:

A>B, если *(N'V'Z'*=1*)* ИЛИ (NVZ' = 1),

т.е. A>B, если (N $\oplus$V)'Z' = 1.

Условие A = B может быть получено, когда при вычитании B из A флаг Z устанавливается в 1. Таким образом, A = B, если Z=1.

В некоторых машинах для получения условий «Больше, чем» «Меньше, чем» и установки соответствующих флагов используется аппаратура. Эти машины имеют в системе команд команды «Перейти, если больше, чем» или «Перейти, если меньше, чем». Там, где подобных команд нет, но такие условные переходы требуются по смыслу, и должны быть созданы программно.

Логическое сравнение (числа без знаков). Часто необходимо сравнивать две группы бит, не представляющих собой числа. Например, программа может выполнять поиск одного символа в 8-битовой таблице кодов ASC 11. В этом случае группы бит могут трактоваться как положительные числа с первым знаковым битом в 9-й позиции. Считается, что в 8-битовой машине каждое слово может содержать число от нуля до десятичного 255. Если сравниваются два таких числа вычитанием одного из другого, то они представляют собой 9-битовые числа в 8-битовой машине. Флаги переполнения и отрицательной величины не отражают эти условия для 9-битовых слов в 8-битовой машине. Следовательно, в этом случае наличие V- и N-флагов несущественно.

Если положительное число B вычитается из другого положительного числа A, результат — либо положительное число (включающее нуль), либо отрицательное, но не переполнение. Положительный результат указывает, что A = B или A>B, а отрицательный – что A<B. Когда эти числа имеют длину 9 бит, условия должны быть выведены путем интерпретации установки флагов 8-битовой машины. Это просто, если ЭВМ имеет Z- и C-флаги.

Рассмотрим случай, когда A<B, т. е. результат вычитания A−B является отрицательным 9-битовым словом. Конечно, девятого бита слова в 8-битовой машине нет. Однако так как девятый бит в обоих словах A и B — нуль (оба они положительны), то девятый бит результата может быть единицей только в том случае, когда между восьмым и девятым битами происходит перенос. Этот перенос устанавливает флаг 8-битовой машины в 1. Верно и обратное: девятый бит результата будет положительным, если переноса не происходит. Следовательно,

если C = 1, то A<B,

если C = 0, то A>B или A = B.

Различие между A>B и A = B устанавливается с помощью Z-флага. Ясно, что при A = B Z будет установлен в 1 вычитанием A−B, но если A>B, вычитание сбрасывает Z в нуль. Условие, при котором девятый бит 1, а другие восемь бит нули, что также дает установку флага Z в 1, не может иметь места, так как в этом случае получился бы результат равный −256, выходящий за диапазон максимальной величины B, равной +255. Полностью условия могут быть записаны так:

A>B, если C = 0 И Z = 0, т. е. если C'Z'=1;

A = B, если C = 0 И Z = 1, т. е. если C'Z = 1; A<B, если C = 1.

Для определения приведенных выше условий нет необходимости совершать вычитание и интерпретировать установку флагов — можно использовать логическое сравнение двух групп бит. Принцип этого метода описан ниже.

Предположим, два 8-битовых слова A и B сравниваются по битам, начиная с самого старшего бита каждого слова. Если два слова идентичны, то идентичны и соответствующие им биты. Если в этом случае соответствующие биты каждого слова связать функцией «Исключающее ИЛИ», то ее значение будет нулевым для каждой позиции бита.

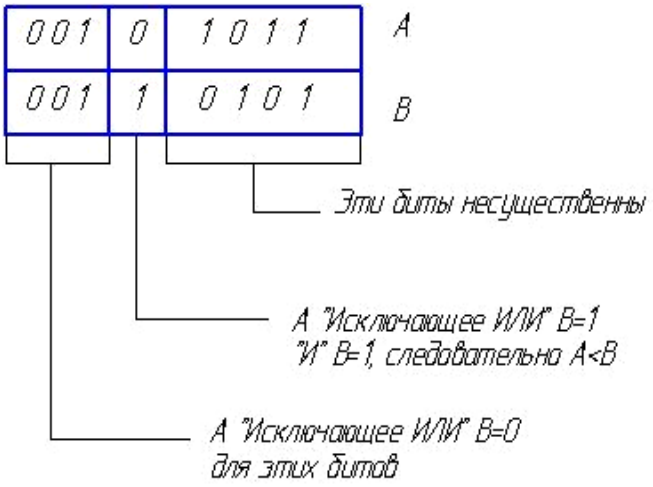


Рис. 5.11. Логика обнаружения отношения A<B

Если же A и B не идентичны, как на рис. 5.11, то существует по крайней мере одна позиция, где соответствующие биты различны, в одном слове — единица, в другом нуль. Единица в самом старшем из неидентичных бит будет находиться в слове, представляющем большее число. Для того чтобы проигнорировать сравнение меньших бит этих слов, можно использовать схему определения приоритетов, подобную той, которая использовалась для определения высшего приоритета запроса на прерывание в многоуровневой системе прерываний. На рис. 5.12 показан логический блок сравнения. Он имеет два 8-битовых слова на входах и три выхода для указания условий A>B, A = B и A<B.

Рис. 5.12. Логический блок сравнения

Другие применения флагов. Когда истинный смысл флагов не существенен для какой-либо отдельной программы, последняя может использовать их для любых целей, для обозначения любых условий. Флаги могут быть использованы, например, в процессе передачи информации от одной подпрограммы к другой. Предположим, программа читает с телетайпа символы в коде ASC 11, при этом требуется обработать только символы, лежащие в диапазоне от A до Z. Один из методов основан на составлении подпрограммы, определяющей, лежит ли символ в требуемом диапазоне. Подпрограмма устанавливает, скажем, флаг V, если символ лежит в этом диапазоне, и сбрасывает его в противном случае. Затем происходит возврат к основной программе, содержащей в качестве следующей команды команду «Перейти, если V сброшен». Таким же образом могут быть использованы все флаги, если имеются команды для их установки, сброса и проверки.

Контрольные вопросы

1. Для чего используются флаги?
2. В чем заключается логическое сравнение?
3. Как используются флаги в процессе передачи информации от одной подпрограммы к другой?

Стек. *Стеком* называется одномерная структура данных, добавление элемента в которую (или исключение элемента из которой) производится с одного конца, называемого *верхушкой* стека. Стек работает по принципу *LIFO* (last-in, first-out) – «поступивший последним обслуживается первым».

Пример стека, состоящего из блока памяти и переменной sp, называемой *указателем стека,* приведен на рис. 5.13. В этом примере указатель стека всегда задает свободную ячейку, находящуюся непосредственно над верхним элементом стека. Ниже будет рассмотрен пример организации стека, в котором указатель стека задает непосредственно верхний элемент.

При выполнении операции загрузки элемента в стек данные записываются на место, определяемое указателем стека, а сам указатель стека устанавливается таким образом, что задает следующую свободную ячейку блока памяти. Данные изымаются из стека с помощью операции извлечения элемента из стека, при выполнении которой указатель стека возвращается назад на один шаг.

Стек организован подобно тому, как устроена «корзина с задачами» некоего доведенного до изнеможения ученого. Как только появляется новая задача, текущая задача опускается в корзину, а ученый приступает к выполнению новой задачи. Когда новая задача будет решена, ученый вернется к предыдущей задаче.

Если задач слишком много, некоторые из них могут оказаться утерянными или забытыми. Подобным образом будет переполняться и стек, если в него загрузить слишком много элементов. Такая ситуация возникает вследствие ограниченного размера блока памяти, выделяемого для организации стека. Что происходит при переполнении стека, зависит от способов его программной реализации. В случае правильно составленной программы стек переполняться не будет, поскольку в такой программе предусмотрено средство обнаружения переполнения стека. Однако если такое средство отсутствует, переполнение стека почти всегда приводит к ошибкам в работе программы. При работе со стеком может возникнуть ситуация, которую довольно трудно себе представить, − это попытка извлечения из стека элемента в то время, когда он пуст. Такую ситуацию можно назвать антипереполнением. Если не предусмотреть средства обнаружения антипереполнения, то последствия для программы могут оказаться такими же

катастрофическими, как и при возникновении ситуации переполнения стека.

Один из способов реализации стека состоит в описании одномерного массива stack и использовании переменной sp целочисленного типа в качестве указателя стека. Тип элементов стека может быть любым; так, в нижеследующем примере употребляются элементы стека литерного типа:

```
CONST stackSize= 100;
VAR
stack: ARRAY [1..stackSize] OF char;
sp : integer;
• • •
```

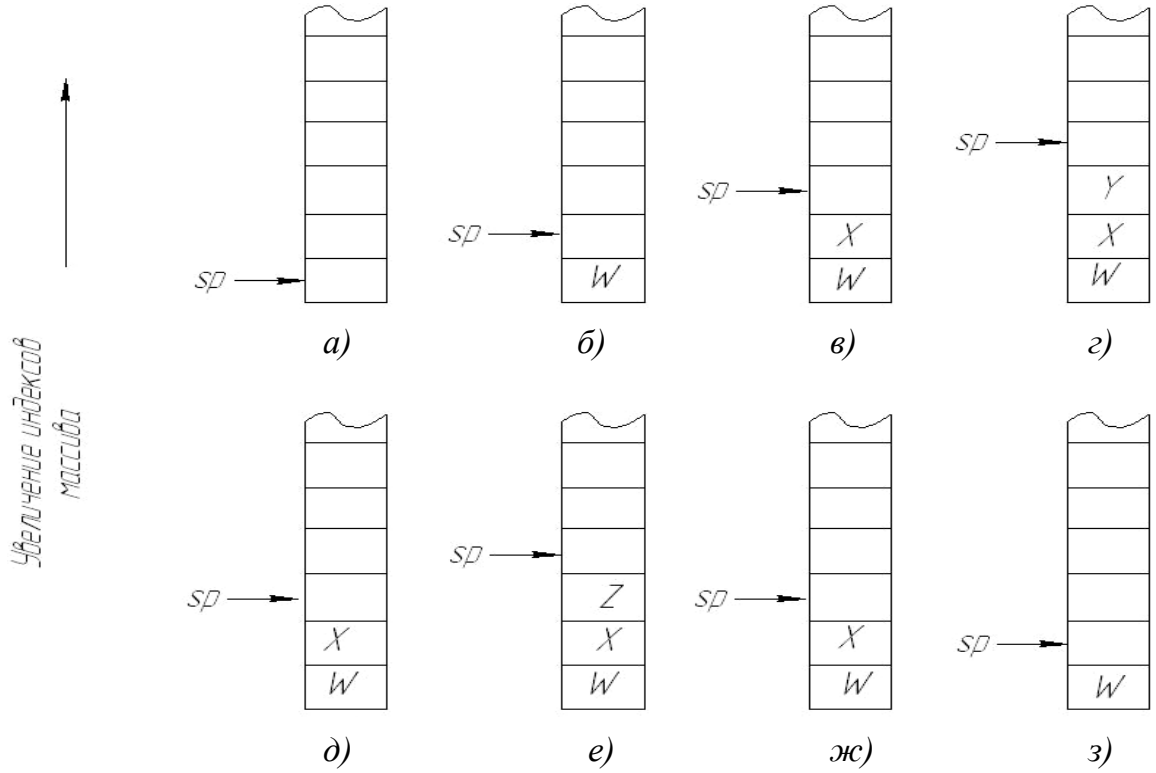


Рис. 5.13. Состояния стека: а – исходное; б − после загрузки элемента W; в − после загрузки элемента X; г − после загрузки элемента Y; д − после извлечения элемента Y; е −после загрузки элемента Z; ж − после извлечения элемента Z; з − после извлечения элемента X

Если в программе используется стек, то в ней должна быть предусмотрена его начальная установка в состояние «пусто». Для нашего примера потребовалось бы написать следующий фрагмент программы:

```
BEGIN
    sp := 1;
    • • •
```

Для загрузки элемента в стек достаточно выполнить следующую последовательность операторов:

```
{Загрузка элемента x в стек}
 stack [sp] : = x;
 sp := sp+1;
```

а для извлечения элемента из стека можно воспользоваться такими операторами:

```
{Извлечение элемента x из стека}
sp := sp—1;
x := stack [sp];
```

Однако если мы попытаемся поместить в стек слишком много элементов, то при выполнении первой последовательности операторов возникнут ошибки в программе. Можно предусмотреть, как показано в следующем примере, специальную проверку на переполнение стека.

```
{Проверка на переполнение и загрузка элемента x в стек}
IF sp<=stackSize THEN
    BEGIN stack [sp] := x; sp := sp+1 END
ELSE
   {Переполнение, сообщение об ошибке};
```

Подобным же образом, прежде чем извлечь элемент из стека, следует убедиться в том, что стек не пуст. Такой контроль предусмотрен включением в программу следующей последовательности операторов:

```
{Проверка наличия элементов и извлечение элемента x из стека}
IFsp>l THEN
    BEGIN sp: = sp—1; x := stack [sp] END
ELSE
   {Антипереполнение, сообщение об ошибке};
```

Представлен пример простой программы, в которой стек используется для изменения порядка расположения литер в строке.

В предыдущих примерах указатели стека всегда указывали на ближайшую к верхнему элементу свободную ячейку. Для чтения данных, находящихся в верхушке стека, без удаления из него элемента не

требуется изменять текущее значение указателя стека, например:
{Чтение данных из верхушки стека без извлечения элемента} x : = stack[sp-1];

Для облегчения чтения данных из верхушки стека желательно устанавливать указатель стека таким образом, чтобы он определял верхний элемент стека. Кроме того, можно организовать изменение значения указателя стека в противоположном направлении. Для этого начальная установка должна быть проведена так, чтобы указатель стека определял последний элемент массива, используемого для организации стека. Удаление элемента из стека будет сопровождаться увеличением значения указателя стека, а загрузка элемента - уменьшением. В стеке stackL числа хранятся в возрастающем порядке, в верхушке этого стека располагается наибольшее число (рис. 5.14).

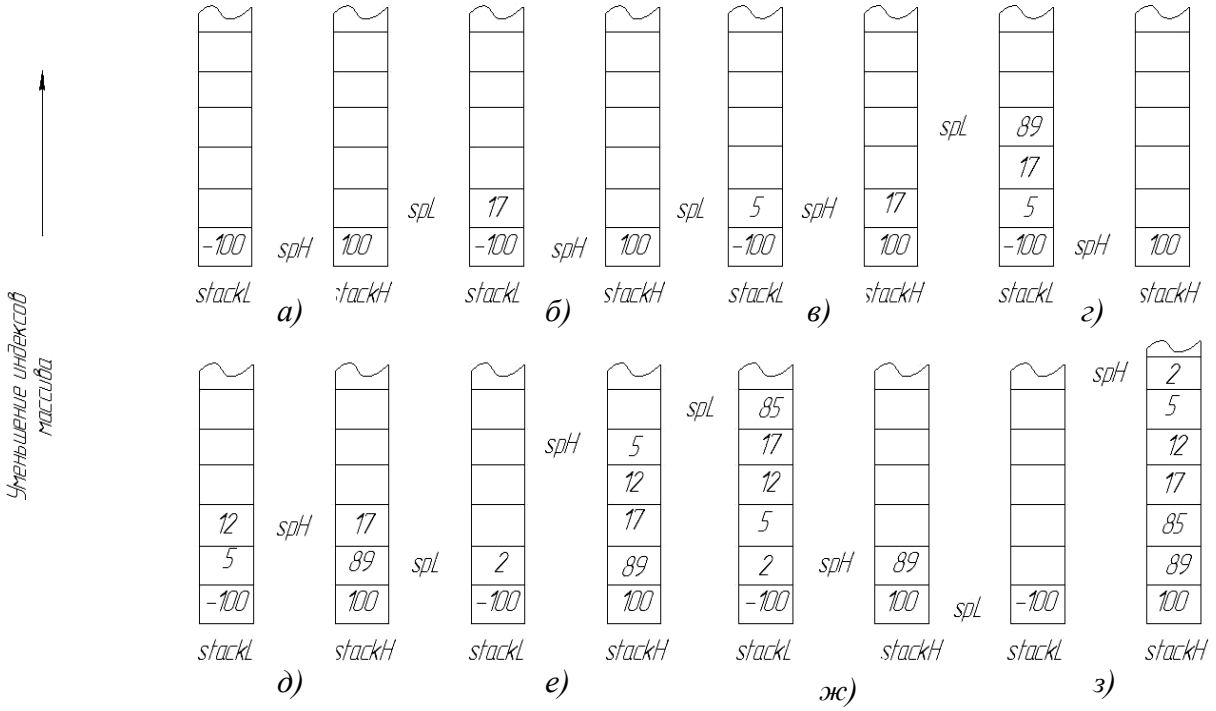


Рис. 5.14. Содержимое стеков, используемых в программе сортировки:
а − начальное состояние; б − з − после ввода чисел 17, 5, 89, 12, 2, 85, — 100

В стеке stackH числа располагаются в порядке их убывания - в верхушке этого стека находится наименьшее число. Кроме того, число в верхушке стека stackL всегда либо меньше, чем число в верхушке стека stackH, либо равно ему. Чтобы записать новое число, потребуется передать данные из стека stackL в стек stackH или из стека stackH в стек stackL таким образом, чтобы значение нового числа лежало между значениями чисел, расположенных в верхушках стеков stackL

и stackH. Теперь можно поместить новое число в любой стек (в программе для этого использован стек stackL); при этом числа в стеках остаются упорядоченными. Когда все исходные числа будут таким способом обработаны, следует передать все числа из стека stackL в стек stackH, а затем извлекать одно за другим числа из стека stackH; упорядоченные исходные числа будут появляться в порядке их возрастания.

Структуры данных типа стека успешно применяются в некоторых областях системного программирования. Так, стеки используются при реализации алгоритмов вычисления выражений в компиляторах и интерпретаторах; в них сохраняются промежуточные результаты вычислений. В программах, написанных на языках высокого уровня (например, на языке Паскаль) и имеющих блочную структуру, локальные данные и другая информация используются в стеке. В блочно-структурированных программах, написанных на языках высокого уровня, параметры процедур обычно передаются с использованием стека. Такой же способ передачи параметров иногда применяется и в программах, составленных на языке ассемблера. Кроме того, как будет отмечено ниже, в микропроцессорах применяется аппаратно реализуемый стек, в котором при обращении к подпрограммам и при обработке прерываний сохраняются адрес возврата и информация о состоянии программы. В программах на языке ассемблера в качестве значения указателя стека sp обычно используется абсолютный адрес ячейки памяти, а не индекс элемента массива.

Очереди. Очередь — одномерная структура данных, в один конец которой добавляют элементы, а изымают с другого конца. Для организации структуры такого типа необходимо располагать блоком памяти и двумя переменными, используемыми в качестве указателей начала (head) и конца (tail) очереди; при этом реализуется принцип FIFO (first-in, first-out) — «поступивший первым обслуживается первым».

При выполнении операции добавления элемента в очередь (рис. 5.15) данные записываются в ячейку, определяемую значением переменной tail, а значение переменной tail изменяется таким образом, чтобы она указывала следующую свободную ячейку блока памяти, доступную для организации очереди. Выполнение операции ис-

ключения элемента из очереди состоит в чтении данных, местоположение которых определяется значением переменной head, и изменении значения переменной head таким образом, чтобы она указывала следующий элемент очереди. Очередь оказывается пустой, когда значения переменных head и tail равны.

При использовании языка Паскаль очередь может быть описана и установлена в начальное состояние следующим образом:

```
CONST queueSize=100;
VAR queue : ARRAY [L..queueSize] OF char;
    head, tail: integer;
BEGIN
    head := 1; tail : = 1;
```

Как и в стеке, элементы очереди могут принадлежать любому типу данных.

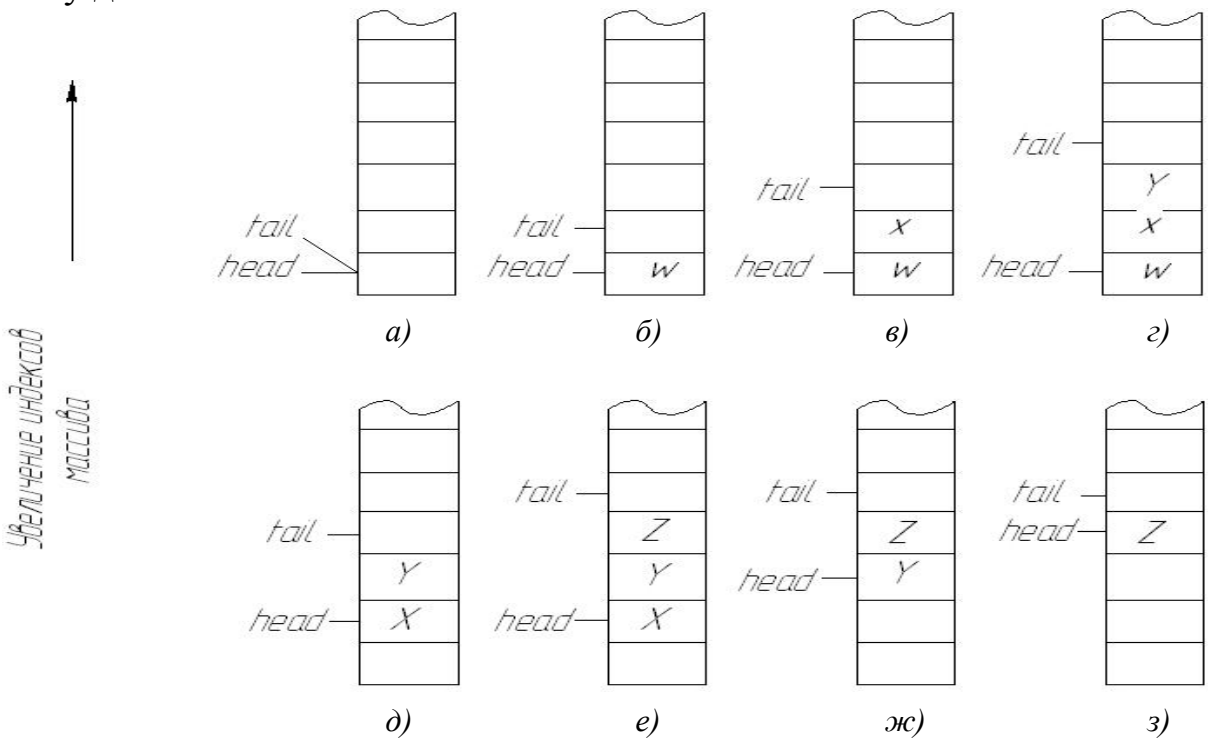


Рис. 5.15. Состояния очереди: а − исходное; б − после добавления элемента W; в − после добавления элемента X; г − после добавления элемента Y; д − после исключения элемента W; е − после включения элемента Z; ж − после исключения элемента X; з − после исключения элемента Y

Очередь, организованная описанным выше способом, имеет ограниченное применение, поскольку массив, состоящий из queueSize

элементов, будет исчерпан, как только в очередь будет добавлен элемент, местоположение которого в массиве определится индексом, равным значению константы queueSize. Теперь в очередь не удастся добавить ни одного элемента, даже если большинство находящихся в ней элементов будет исключено. Необходимо каким-то образом использовать память, освобождающуюся по мере того, как элементы исключаются из очереди, чтобы переполнение очереди наступало тогда, когда действительно введено слишком много элементов. Этого нетрудно достичь, если представить массив в виде кольцевого буфера, схема которого изображена на рис. 5.16. Такой буфер по достижении его конца начинает заполняться с начала. Для добавления и исключения элемента x, если очередь организована в виде кольцевого буфера, можно использовать следующие последовательности операторов:

```
{Добавление элемента x}
   queue [tail] := x; tail := tail+1;
   IF tail>queueSize THEN tail: = 1;
{Исключение элемента x}
   x := queue [head]; head := head+1;
   IF head>queueSize THEN head: = 1;
```

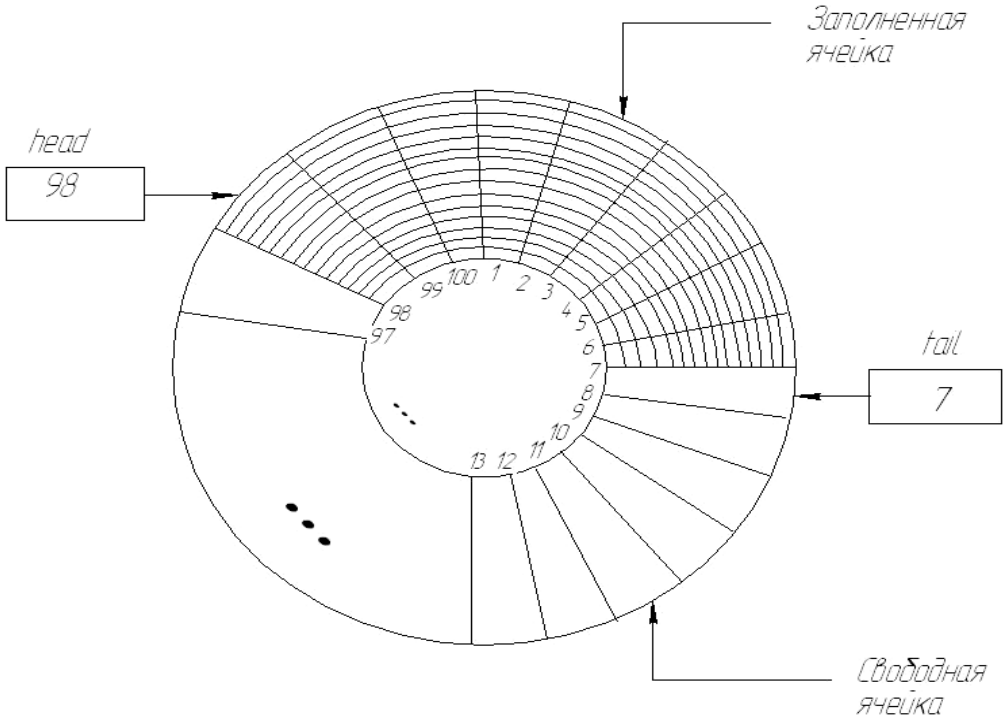


Рис. 5.16. Организация очереди в виде кольцевого буфера

Очередь, как и стек, может переполниться, если мы попытаемся включить в нее слишком много элементов. Состояние переполнения очереди может быть обнаружено, когда конец очереди «догонит» ее начало. При рассматривании рис. 5.15 может создаться впечатление, что значение переменной tail всегда больше значения переменной head (или равно ему) и, следовательно, переполнение очереди можно обнаружить путем проверки условия tail<head. Однако в связи с тем, что очередь имеет кольцевую организацию, условие tail<head может оказаться истинным и в режиме нормального функционирования. Таким образом, следует выполнять проверку равенства tail+1 = head, прежде чем включать в очередь новый элемент. Эту проверку можно производить с помощью следующего фрагмента программы:

```
{Проверка переполнения очереди и включение в нее элемента x}
    temp := tail+1;
    IF temp>queueSize THEN temp :=1;
    IF temp=head THEN {Переполнение, сообщение об ошибке}
    ELSE BEGIN queue [tail] := x; tail := temp END;
```

Если значение переменной tail на единицу меньше значения переменной head, нельзя добавлять элементы в очередь, несмотря на то что имеется одна свободная ячейка в памяти. Иначе критерием заполнения очереди было бы выполнение равенства head = tail, которое является истинным и для пустой очереди. Следовательно, в очереди может содержаться только queueSize-1 значащих элементов. При этом удобно считать, что в очереди может находиться от 0 до queueSize-1 элементов и всего имеется queueSize соответствующих состояний очереди.

В надлежащим образом составленной программе перед исключением элемента из очереди должна выполняться проверка наличия в ней элементов. Такая проверка реализуется в следующем фрагменте программы:

```
{Проверка наличия элементов и исключение элемента x}
   IF head = tail THEN {Очередь пуста, сообщение об ошибке}
   ELSE BEGIN
      x := queue[head]; head := head+1;
     IF head>queueSize THEN head := 1;
  END;
```

Существует также другой, отличный от рассмотренного способ

организации очереди, при котором вместо того чтобы при исключении элемента из очереди изменять значение указателя на начало очереди, все элементы очереди перемещаются на один шаг к ее началу. Такой прием редко используется в программах, поскольку он приводит к значительным дополнительным затратам времени на последовательное перемещение каждого элемента очереди. Однако этот способ организации очереди довольно широко применяется при ее аппаратной реализации, поскольку сдвиговый регистр может работать таким образом, чтобы все перемещения элементов выполнялись параллельно.

Структуры данных типа «очередь» широко используются в операционных системах больших ЭВМ. Их применяют для хранения запросов на обслуживание в порядке их поступления. По мере освобождения нужных ресурсов запросы обслуживаются по принципу «поступивший первым обслуживается первым». Кроме того, очереди являются наиболее естественными структурами данных в программном обеспечении, используемом для выполнения ввода-вывода данных в любых ЭВМ. Когда устройства и программы имеют различные временные характеристики, можно прибегнуть к очередям для временного хранения данных, поступающих от быстродействующего устройства или какой-либо программы. Позже эти данные без изменения порядка их поступления могут быть переданы по запросу другому устройству или программе.

Контрольные вопросы

1. Перечислите способы организации стека.
2. Что представляет собой структура данных типа «очередь»?
3. В чем заключается функция указателя стека?
4. Как необходимо устанавливать указатель стека?
5. Для чего необходимо выполнять проверку переполнения очереди?

Глава 6

АППАРАТНЫЕ СРЕДСТВА

УСТРОЙСТВА ЦИФРОВОГО ВВОДА ВЫВОДА

Интерфейс с шиной интервального таймера. В первом примере подключения к шине устройств ввод-вывода мы рассмотрим интерфейс программируемого интервального таймера 8254. Микросхема 8254 показана на рис. 6.1. Фактически в ней есть три таймера с номерами 0 - 2. Импульсы синхронизации для работы таймеров подаются на входы CLK0, CLK1 и CLK2. Каждый таймер формирует сигнал прерывания на одном из выходов OUT0, OUT 1 и OUT 2.

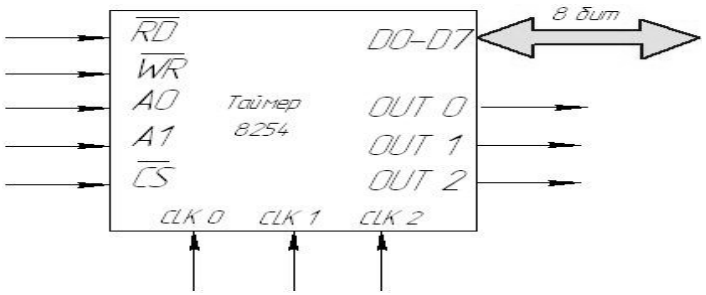


Рис. 6.1. Программируемый интервальный таймер 8254

Таймер 8254 имеет много режимов работы. Для разделения времени в мультизадачной системе необходимо выбрать режим *генератора частоты*. В этом режиме таймер 8254 периодически формирует сигналы прерываний. Продолжительность периода можно задавать программно. Мы не будем подробно обсуждать режимы работы микросхемы 8254; желающие познакомиться с ними могут обратиться к справочным материалам.

В микросхеме 8254 имеется регистр управления, который определяет режим, и регистры для хранения периодов трех таймеров. Доступ ко всем регистрам осуществляется по линиям D_0 - D_7. Сигналы на линиях A_0 и A_1 выбирают, какой из регистров подключается к линиям данных (см. таблицу). Сигналы RD и RW производят считывания из адресуемого регистра или запись в него. Вход выбора кристалла CS разрешает считывание или запись.

Адресация регистров таймера 8254

A1	A0	Выбирают
0	0	Период 0
0	1	Период 1
1	0	Период 2
1	1	Регистр управления

Мы подключим каждый из регистров к разным портам ввода-вывода процессора, чтобы программа могла производить считывание и запись командами Ш и СШТ соответственно. Назначим адреса портов следующим образом:

8 *n* период 0

8 n + 2 период 1

8 *n* + 4 период 2

8 *n* + 6 регистр управления

Так как номера портов состоят из 16 бит, старшие 16 бит номера порта будут *n*, следующие два бита будут выбирать регистр, а младший бит будет 0.

Как же дешифрировать адреса? Можно, например, построить схему из логических элементов для распознавания номера ***n***. Такой вариант мы видели в предыдущем разделе, где значение ***n*** было очень простым (все единицы). Но здесь мы воспользуемся более гибким способом.

На рис. 6.2 показан программируемый компаратор тождества ALS526. Как и при программировании ППЗУ, в компаратор можно запрограммировать любое 16-битное число Q. Если 16-битное число на входах P_0 - P_{15} равно запрограммированному числу Q и если микросхема разрешена по входу G, то на выходе будет сигнал TRUE.

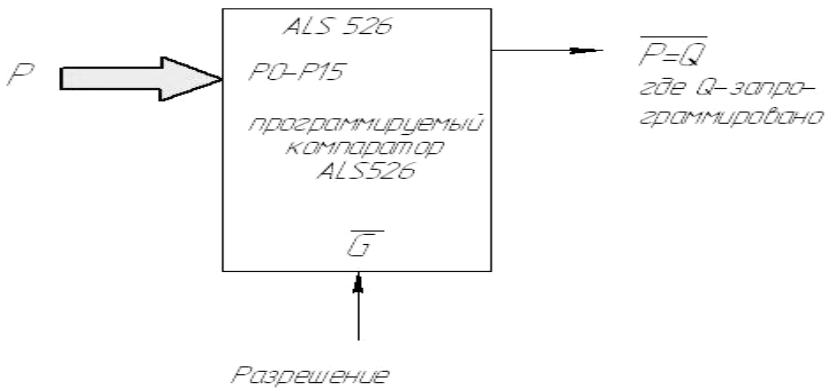


Рис. 6.2. Программируемый 16-битный компаратор ALS526

На рис. 6.3 показано, как подключить таймер к шине. Биты A_1 и A_0 с шины адреса выбирают соответствующий регистр 8254. Биты A_3 - A_{15} сравниваются со значением *n*, которое запрограммировано в биты Q_3 -Q_{15} программируемого компаратора. Бит A_0 сравнивается с Q_0 , который должен быть запрограммирован на нуль. Входы C, P_1 и P_2 не используются, поэтому они заземлены. Биты Q_1 и Q_2 должны быть запрограммированы на нуль.

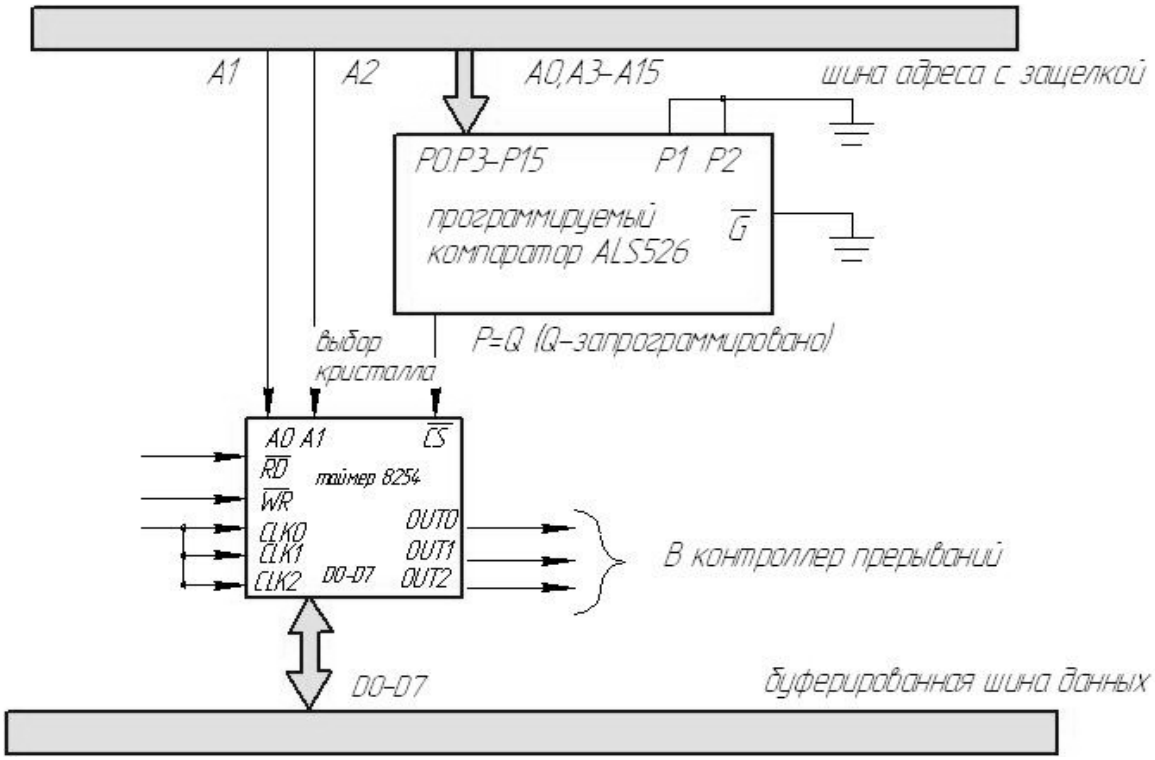


Рис. 6.3. Подключение таймера 8254 к шине

Все три входа синхронизации подключены к сигналу синхронизации процессора PCLK (здесь не требуется более высокая разрешающая способность сигнала CLK), а сигналы IORC и IOWC подаются от контроллера шины. Выходные сигналы таймера 8254 подаются в контроллер прерываний 8259A, который рассматривается далее.

Ввод-вывод, отображенный на память. Предположим, что на рис. 6.3 к таймеру 8254 вместо сигналов ввода-вывода IORC и IOWC подключены командные сигналы памяти MRDC и MWTC. Предположим далее, что памяти по физическим адресам, равным номерам портов, нет, поэтому два устройства на шине одновременно реагировать не могут. Поскольку процессор 80286 осуществляет операции

считывания/записи памяти аналогично соответствующим операциям ввода-вывода, регистры таймера 8254 будут "выдавать себя" за память. Считывание из памяти будет считыванием из регистров таймера, а запись в память - записью в регистры. Так реализуется *ввод-вывод, отображенный на память.*

Преимущество ввода-вывода, отображенного на память, по сравнению с обычным вводом-выводом заключается в том, что можно использовать все средства преобразования адреса виртуального режима для размещения каждого устройства ввода-вывода, отображенного на память, в его собственном сегменте. После этого каждое устройство можно защищать отдельно, а программам управления устройствами разрешается доступ только к "своим" устройствам, а не ко всем.

Недостаток ввода-вывода, отображенного на память, состоит в том, что иногда требуются более сложные схемы дешифрирования адреса. Так как номера портов имеют 16 бит, схема на рис. 6.3 реагирует только на сигналы адреса A_0 - A_{15}. Физические адреса состоят из 24 бит, поэтому при отображении таймера 8254 на память он появится по множеству различных физических адресов, если не ввести схемы контроля лишних 8 бит.

Контрольные вопросы

1. Для чего используется режим генератора частоты?
2. Как реализуется ввод-вывод, отображенный на память?
3. В чем преимущества и каковы недостатки ввода-вывода, отображенного на память?

Подключение контроллера прерываний. Процессор 80286 имеет два входа прерываний: NMI - немаскируемое прерывание и INTR - запрос прерывания. Сигнал на входе NMI заставляет процессор вызвать процедуру прерывания для прерывания с номером 2. Сигнал INTR позволяет внешним устройствам передать в процессор по шине данных однобайтный номер прерывания. Благодаря этому процессор может обслуживать до 256 различных типов прерываний, не требуя 256 входов запросов прерываний. Обычно на вход INTR подается выходной сигнал программируемого контроллера прерываний 8259A, а не запросы прерывающих устройств.

Контроллер собирает запросы прерываний от множества устройств и передает их по одному в процессор. Он воспринимает запросы прерываний от внешних устройств (включая и другие микросхемы 8259A), учитывает их приоритеты, а затем сигнализирует процессору по входу INTR. Реагируя на сигнал, процессор заканчивает выполнение текущей команды, а затем инициирует два цикла шины *подтверждения прерывания* INTA.

Первый цикл шины INTA информирует контроллер прерываний о том, что процессор распознал запрос прерывания. Он также отводит время контроллеру для взаимодействия с подключенными к нему другими контроллерами. В течение второго цикла шины INTA контроллер выдает в процессор по шине 8-битный номер прерывания. После этого процессор вызывает соответствующую процедуру обработки прерывания.

Подключения контроллера прерываний в системе показаны на рис. 6.4. Запросы прерываний подаются в него по восьми входам IR_0 - IR_7. Следовательно, один контроллер может обработать до восьми видов прерываний (восьми различных номеров прерываний). При наличии более восьми типов прерываний потребуется несколько контроллеров.

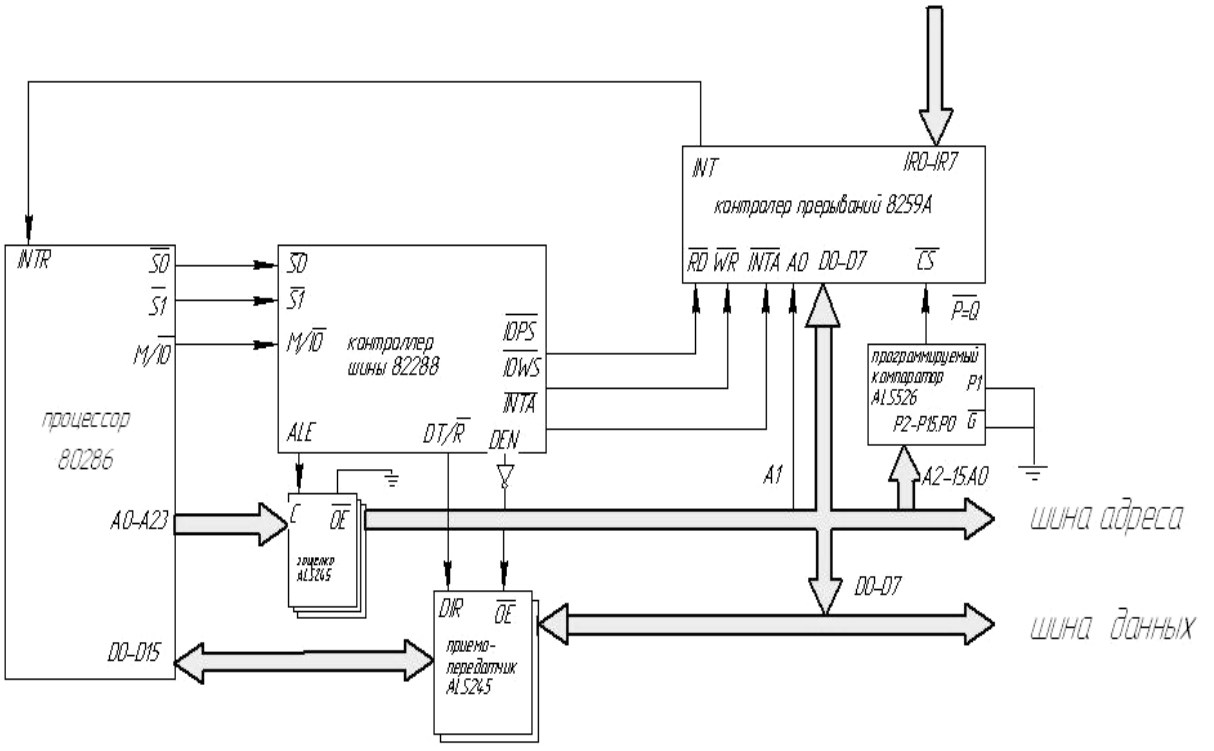


Рис. 6.4. Подключение контроллера прерываний 8259A к процессору 80286

Контроллер 8259A требует два порта ввода-вывода, чтобы программы могли считывать и устанавливать разнообразные режимы его работы, обращаться к его внутренним регистрам и сообщать об окончании процедуры прерывания. Вход A_0 определяет, какой из двух портов адресуется, а сигналы RD, WR, CS и D действуют так же, как и в таймере.

Каскадирование контроллеров прерываний. Чтобы обрабатывать более восьми видов прерываний, требуется несколько контроллеров прерываний. Только один из них, называемый *ведущим,* соединяется со входом INTR процессора. Остальные контроллеры, называемые *ведомыми,* подключаются на входы запросов прерываний ведущего контроллера. На рис. 6.5 показано, как соединить пять контроллеров прерываний для того, чтобы обработать до 36 различных типов прерываний. Четыре прерывания от устройств подаются в ведущей контроллер, а остальные 32 прерывания вначале проходят через ведомый. Линии, соединяющие пять контроллеров прерываний с шиной процессора, на рисунке не показаны. Чтобы контроллеры прерываний действовали согласованно (см. далее), каждый из них должен знать свое предназначение. Сигнал на входе SP/EN сообщает контроллеру, является он ведущим или ведомым. Кроме того, во внутреннем регистре ведущего контроллера зафиксировано, какие входы запросов прерываний подсоединены к ведомым контроллерам, а какие непосредственно к устройствам. Во внутреннем регистре каждого ведомого контроллера хранится номер входа IR ведущего контроллера, к которому он (ведомый контроллер) подсоединен.

Предположим, что устройство запрашивает прерывание через один из ведомых контроллеров. Последний сформирует запрос прерывания через ведущий контроллер. Затем процессор 80286 отреагирует двумя циклами шины подтверждения прерывания INTA.

В течение первого цикла шины INTA ведущий контроллер подтверждает запрос прерывания ведомого контроллера, выдавая его номер входа IR на линии каскадирования CAS. Эти линии соединяют ведущий контроллер со всеми ведомыми. Такое подтверждение необходимо потому, что запросить прерывание могут одновременно несколько ведомых контроллеров.

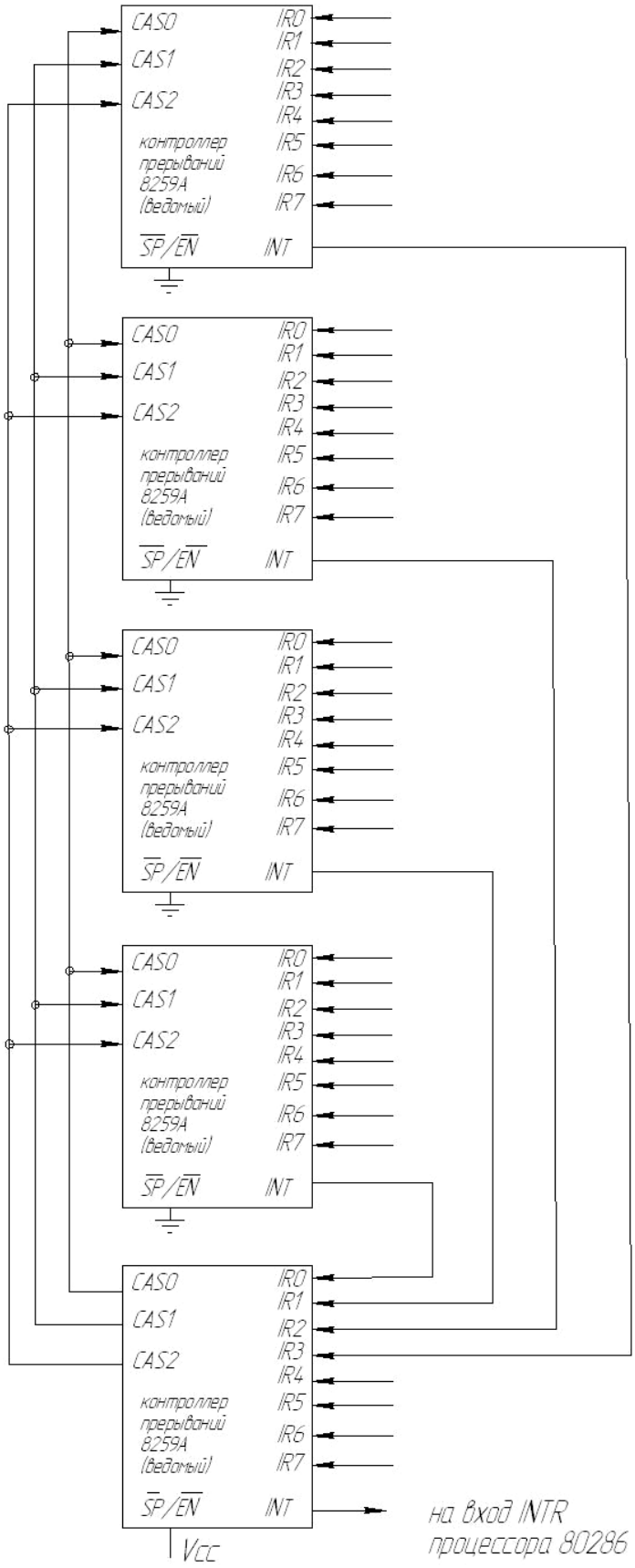


Рис. 6.5. Каскадное включение контроллеров прерываний

Во втором цикле шины INTA выбранный ведомый контроллер помещает номер прерывания прерывающего устройства на шину данных, чтобы его мог считать процессор. После этого процессор реагирует вызовом соответствующей процедуры прерывания, как и в случае одного контроллера прерываний.

Особенности работы. Микросхемы 8254 и 8259A были разработаны до появления процессора 80286. К сожалению, они не могут работать с частотой синхронизации 8 МГц (системная синхронизация 16 МГц). Для обеспечения надежной работы простых схем необходимо уменьшить частоту системной синхронизации до 10 МГц. Но можно оставить частоту системной синхронизации 16 МГц, если встроить дополнительные схемы для введения состояний ожидания и задержек в те циклы шины, в которых активны микросхемы 8254 или 8259A. Такие схемы приведены в литературе.

Контрольные вопросы

1. Какие входы прерываний имеет процессор 80286?
2. Каковы функции сигналов INTR и NMI?
3. Опишите циклы шины подтверждения прерывания INTA.
4. В чем заключается различие между ведущим и ведомыми контроллерами прерываний?
5. Опишите принцип работы контроллера прерываний.

Прямой доступ к памяти. Самый быстрый способ передачи данных между устройством ввода-вывода и памятью через процессор 80286 заключается в использовании команд INS и OUTS. Например, следующая программа считывает 2048 слов из устройства ввода-вывода (например, с дискового накопителя) в память:

```
MOV  DX, portnum ;   portnum- это номер устройства
MOV  CX, 2048      ;   2048 слов = 4096 байт
LES   DI, memva    ;   memva - это виртуальный адрес
                   ;          приемника
REP   INSW         ;     выполнение операции считывания
```

Каждое слово вначале считывается в процессор в цикле шины ввода, а затем записывается в память в цикле шины записи в память.

Так как цикл шины длится два цикла процессора и для передачи требуются два цикла шины, скорость передачи составляет (при частоте синхронизации процессора 8 МГц):

$$\frac{(8\textit{Мтактов}/\textit{с})(2\textit{байта}/\textit{слово})}{4\textit{такта}/\textit{слово}} = 4\text{Мбайт}/\text{с}.$$

Желательно передавать слово из устройства ввода-вывода в память за один цикл шины, а не за два. Скорость передачи данных составила бы 8 Мбайт/с, что равно максимальной пропускной способности шины. Высокая пропускная способность особенно важна в системах виртуальной памяти, где необходимо передавать большие сегменты с диска и на диск.

Для практической реализации поставленной задачи требуется прямой канал в память, не проходящий через процессор 80286. Именно для организации каналов прямого доступа к памяти (ПДП) в процессоре предусмотрен вход запроса шины HOLD и выход подтверждения запроса шины HLDA. Когда устройство формирует сигнал HOLD, процессор заканчивает текущий цикл шины, прекращает управление шиной и выдает сигнал HLDA. После этого устройство, выдавшее сигнал HOLD, может управлять шиной до снятия сигнала HOLD.

Процессор 80286 полностью освобождает шину. Запросившее шину устройство несет ответственность за генерирование сигналов состояния шины, управление шиной адреса и распознавание запросов состояний ожидания. Таким образом, устройство может управлять шиной данных в циклах записи в память (прямая передача в память) и контролирует шину данных в циклах считывания из памяти (прямая передача из памяти).

Очевидно, для полного управления шиной потребуется очень сложная схема. Вместо введения такой схемы в каждое устройство, которое хочет осуществить ПДП, она сосредоточена в одной микросхеме 82258, называемой контроллером ПДП. Эта микросхема содержит четыре канала ПДП, поэтому может обслужить до четырех устройств.

Назначение канала ПДП - образовать поток обращений к памяти. Именно канал отвечает за производство правильного обращения к памяти (считывание или запись) и в правильную ячейку. Подключен-

ное к каналу устройство ввода-вывода отвечает только за выдачу своих данных на шину и прием данных с шины по сигналам канала.

Канал ПДП генерирует требуемый поток обращений, выполняя *канальную программу,* которую он считывает из памяти (рис. 6.6). Канальная программа представляет собой последовательность *командных блоков.* Каждый командный блок определяет операцию памяти (считывание, запись), реализуемую каналом ПДП. Последний канальный командный блок в канальной программе содержит команду STOP.

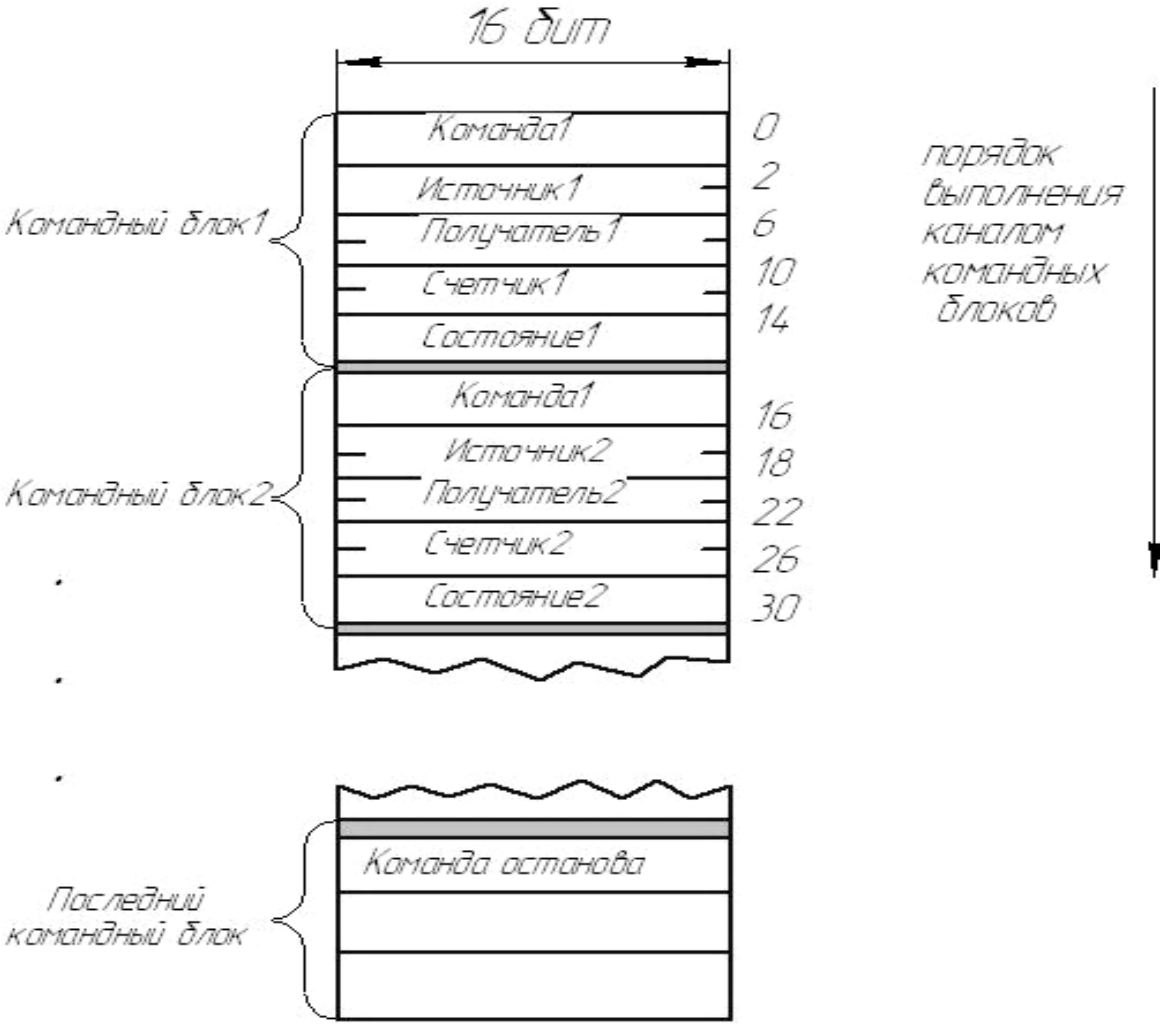


Рис. 6.6. Канальная программа

Каждый командный блок содержит физические адреса источника и (или) приемника в передаче (обычно используется только один адрес) и число передаваемых байт. Следовательно, одна канальная программа может задать передачу с участием нескольких областей памяти. Такая возможность рассеивать входные данные по памяти и

собирать данные для вывода называется *рассеивающим считыванием* и *собирающей записью.* Мы не будем подробно обсуждать программирование канала, а сосредоточимся на аппаратном интерфейсе.

Контроллер ПДП имеет несколько внутренних регистров, часть из которых дублируется для каждого из четырех каналов. Для инициирования передачи ПДП программа процессора 80286 должна вначале загрузить в регистр указателя команды нужного канала начальный адрес канальной программы. После этого программа оповещает подключенное к каналу устройство ввода-вывода о начале передачи. Выдав оповещение, процессор может заняться другими делами. Контроллер ПДП прервет процессор, когда канальная программа закончится.

Микросхема 82258 показана на рис. 6.7. Она имеет сигналы DREQ (запрос ПДП), DACK (подтверждение ПДП) и EOD (конец ПДП) для каждого канала. Когда устройство ввода-вывода готово передавать слово, оно сигнализирует каналу по входу DREQ. Канал показывает свою готовность к передаче выходным сигналом DACK. После этого устройство считывает с шины данных при выводе или выдает данные на шину данных при вводе.

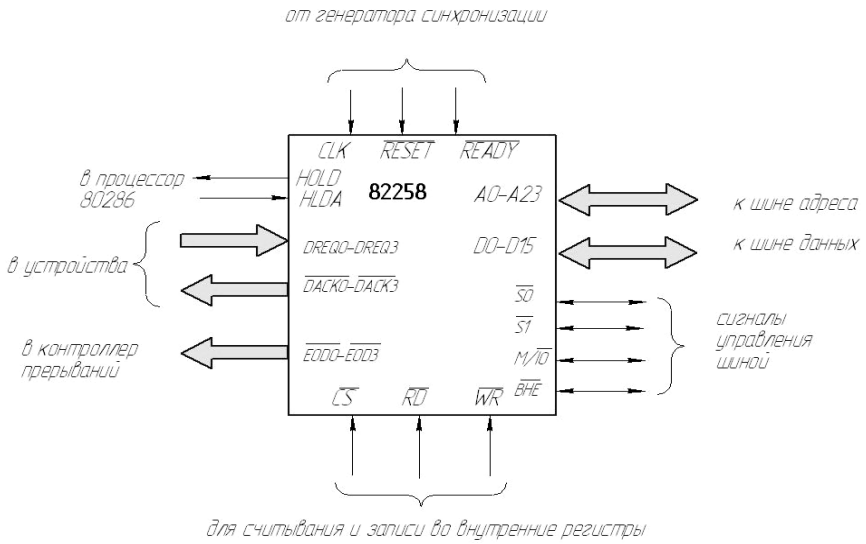


Рис. 6.7. Контроллер прямого доступа к памяти

Последовательные сигналы DREQ вызывают обращения к последовательным словам в памяти, как определено в канальной программе. Само устройство ввода-вывода не знает, куда в память помещаются его данные и откуда берутся выводимые данные. Адресацией управляет контроллер ПДП по команде канальной программы. Когда

канальная программа достигает конца, контроллер ПДП запрашивает прерывание по соответствующему выходу EOD.

На рис. 6.8 показано, как подключить контроллер ПДП к системе на базе процессора 80286. Ради простоты показан всего один канал, а генератор синхронизации и контроллер прерываний опущены.

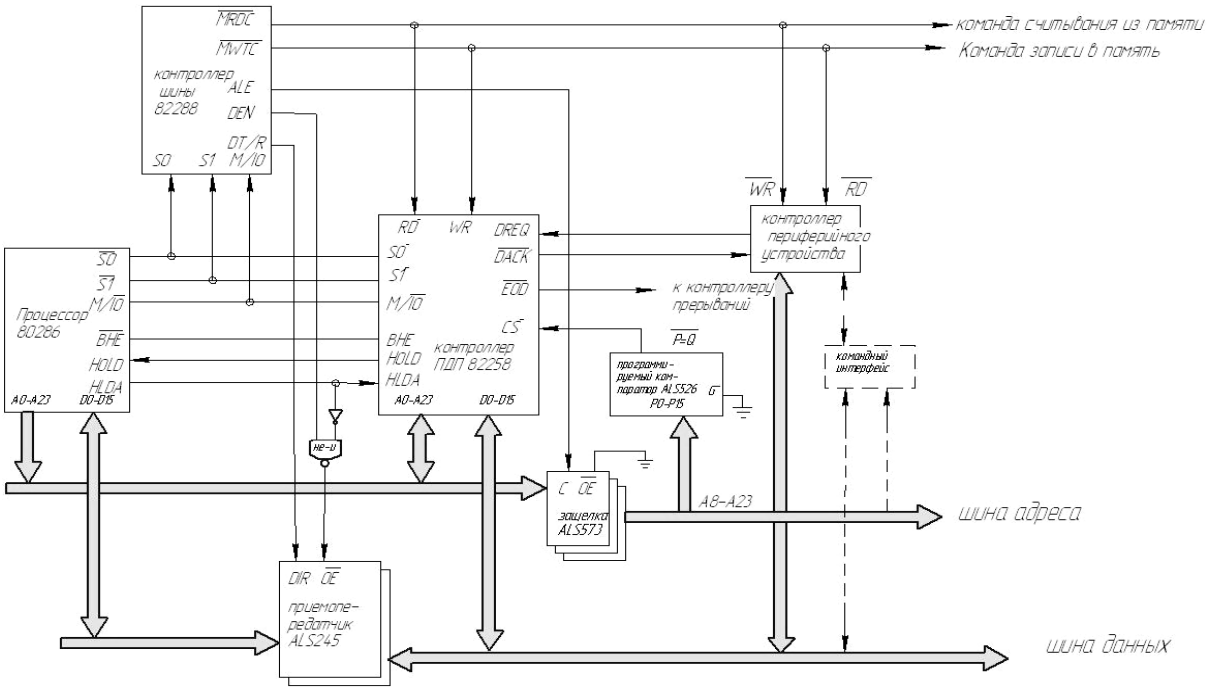


Рис. 6.8. Подключение контроллера ПДП 82258

Мы уже рассмотрели сигнальные линии HOLD, HLDA, DREQ, DACK и EOD. Сигнал BHE обсуждается в следующем разделе. Отметим, что линия HLDA запрещает приемопередатчики шины данных, когда процессор не управляет шиной. Когда контроллер ПДП получает управление шиной, он формирует сигналы состояния шины S0, S1 и M/IO и адреса $A_0 \ldots A_{23}$.

Контроллер ПДП подключен так, что к его внутренним регистрам можно обращаться, применяя ввод-вывод, отображенный на память. Поэтому на его входы RD и WR поданы сигналы MRDC и MWTC. Для адресации внутренних регистров используются линии адреса A_0 - A_7. Остальные линии адреса с помощью программируемого компаратора формируют сигнал выбора кристала.

Контроллер периферийного устройства (диска, ленты и т. п.) подключен к линиям DREQ и DACK, а также к шине данных. Сигналы MRDC и MWTC сообщают, является передача ППД вводом (запи-

сью в память) или выводом (считыванием из памяти). Для производства передач ПДП контроллер периферийного устройства не нужно подключать к шине адреса.

Контрольные вопросы

1. Опишите принцип работы контроллера ПДП.
2. Какие операции реализуются каналом ПДП?
3. Какие функции выполняет сигнал DREQ?
4. Что такое канальная программа?

Устройства аналогового ввода вывода. Большинство периферийных контроллеров для реализации ПДП требуют обычного интерфейса ввода-вывода для команд (поиск, перемотка, запуск ввода-вывода и др.) и состояния (конец ленты, ошибка диска и т. п.). Этот интерфейс показан на рис. 6.8 штриховыми линиями.

При построении управляющих и измерительных систем можно использовать специализированные микросхемы ЦАП и АЦП. Микросхемы этих устройств должны подключаться к шинам микроЭВМ. В справочных материалах оговариваются не только параметры микросхем, но и описывается их структура. Принципиальным является наличие или отсутствие в структуре АЦП встроенного регистра, с помощью которого можно обеспечить подключение его к шинам. Если регистр во внутренней структуре АЦП отсутствует, то необходимо между выводами микросхемы и каналом установить внешний регистр. Вход тактовой частоты можно использовать как управляющий. При этом он должен быть подключен к управляемому генератору. Необходимо обратить внимание на усилители А1-А4. К ним могут быть предъявлены особые требования. Тип этих усилителей выбираем из справочника и обеспечиваем полную схему включения. Аналогичные требования предъявляются и ко всем остальным микросхемам.

Физические параметры объектов, которые мы исследуем при обработке сигналов, обычно непрерывно изменяются. Например, рассмотрим изменение температуры атмосферы во времени. Поскольку температура меняется непрерывно, теоретически возможно производить измерение через бесконечно малые промежутки времени. Однако принимая в расчет объем памяти, необходимый для хранения данных измерения, и время на их обработку, невольно задумаешься,

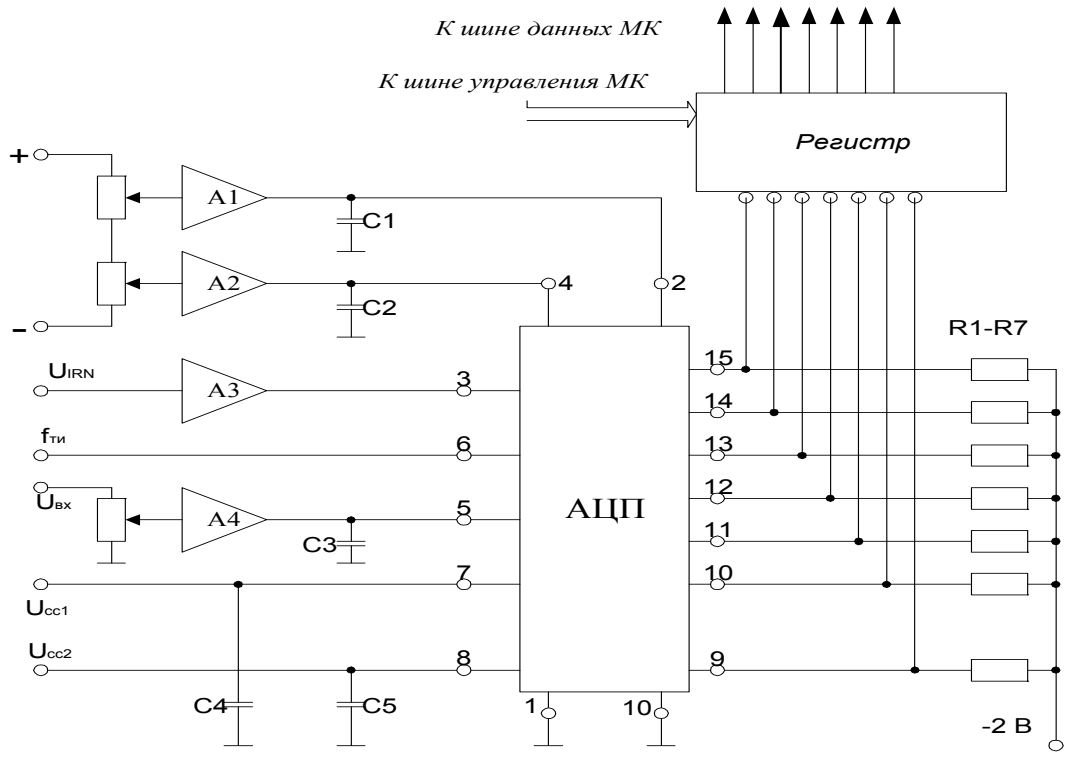


Рис. 6.9. Подключение АЦП 1107ПВ3

насколько подробные измерения нам необходимы. Значение температуры не может внезапно измениться в течение одной секунды или минуты. Следовательно, допустимы измерения через более длительные интервалы времени, что в конечном итоге сокращает объем данных. Чем меньше объем данных, тем меньше времени затрачивается на их обработку в компьютере при меньшем объеме памяти. То же самое можно сказать и о степени точности измерения. Пусть температура атмосферы в данный момент равна 25.27854 °C. Нет смысла в проведении измерения с такой высокой точностью. Вполне достаточно определить степень точности до одной десятой градуса, т.е. 25.3 °C. В настоящее время в метеорологическом центре данные о температуре атмосферы по всей стране собираются каждый час и измерения производятся с точностью до одной десятой градуса. Этого вполне достаточно.

Сигнал, выражающий непрерывно изменяющуюся величину, называется аналоговым сигналом, а ступенчатое представление сигнала — дискретизацией. Дискретизация может производиться как по

времени, так и по значению величины сигнала. В первом случае ее часто называют операцией получения выборки, во втором — квантованием. Если сигнал, подвергнутый дискретизации по времени и по значению, затем представляется в цифровом виде, то такое преобразование аналогового сигнала в цифровой называется аналого-цифровым преобразованием.

Аналоговый сигнал, полученный от датчика, посредством аналого-цифрового преобразователя (АЦП) преобразуется в числовые значения в двоичной системе счисления, т.е. предстает в виде нулей и единиц. Например, при записи на компакт-диск звуковой сигнал преобразуется и под воздействием лазерного луча записывается в виде цифрового сигнала. Частота выборки звукового сигнала равна 44,1 кГц, а число цифр в записываемом числе равно 16. Цифры на диске записываются в виде наличия или отсутствия углубления, называемого питом. Вы, наверное, знаете, что по сравнению с записью обычного аналогового сигнала на кассете или пластинке цифровая запись характеризуется высоким отношением сигнал-шум и широким динамическим диапазоном (отношение минимального сигнала к максимальному неискаженному сигналу) и обеспечивает высокое качество воспроизведения звука. Но чтобы его воспроизвести, цифровой сигнал необходимо снова преобразовать в аналоговый. Этот процесс называется цифро-аналоговым преобразованием.

Компьютеры обладают высокой скоростью вычисления и обработки информации. Поэтому в последнее время заметно возрастает их использование для целей обработки цифровых сигналов по сравнению с традиционным методом обработки аналоговых сигналов посредством электронной аппаратуры. Что касается аналого-цифрового преобразования, важно предусмотреть оптимальное количество уровней квантования, а также установить необходимую частоту выборки.

ПРОБЛЕМА ВЫБОРКИ

Схема, поясняющая процесс преобразования и используемые устройства, представлена на рис. 6.10.

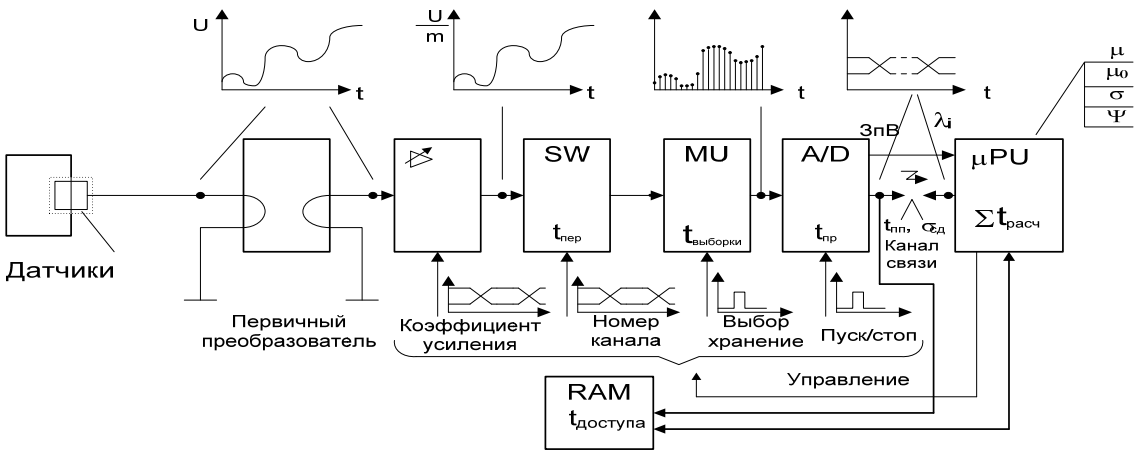


Рис. 6.10. Преобразование сигнала

В процессе преобразования аналогового сигнала в цифровой очевидно, что чем шире интервал дискретизации выборки и грубее квантование, тем меньше требуется данных для того, чтобы представить сигнал (рис. 6.11). Однако если сигнал представлен слишком малым объемом данных, то возникает опасность потерять информацию, которую сигнал содержит. Одна из самых основных проблем, с которой мы с самого начала сталкиваемся при обработке сигналов, – это проблема выбора интервала дискретизации выборки.

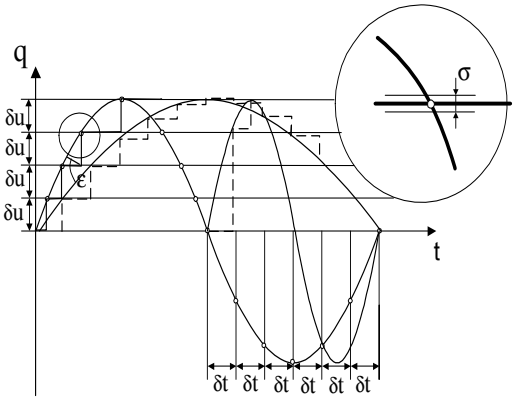


Рис.6.11. Кватование сигнала

Для сигнала, представленного на рис. 6.11, необходимо задавать амплитуду и период повторения. При этом выполняется квантование по уровню и дискретизация по времени. С точки зрения микропроцессорной реализации, целесообразно использовать равномерный шаг квантования, но при этом величина погрешности ε аппроксимации сигнала зависит от его периода. В этом случае для сохранения равноточной аппроксимации необходимо изменять число дискрет. Конкретное число дискрет или величина шага квантования, рассчитывается для каждого сигнала исходя из минимума погрешно-

сти представления. Для преобразования сигнала из цифровой формы в непрерывную и наоборот, используются специальные микросхемы.

Давайте рассмотрим выбор частоты дискретизации.

Выбор интервала дискретизации по времени свяжем с теоремой Котельникова – Шенона*. В соответствии с этой теоремой, если непрерывная функция *q(t)* удовлетворяет условиям Дирихле (ограничена, кусочно-непрерывна и имеет конечное число экстремумов) и ее спектр ограничен некоторой частотой f_m, то существует такой максимальный интервал Δ_T между отсчетами, при котором имеется возможность безошибочно восстановить дискретизируемую функцию *q(t)* по дискретным отсчетам. Этот максимальный интервал

$$\Delta_T = \frac{\pi}{\omega_m} = \frac{1}{2f_m}.$$

Действительно, для непрерывной функции *q(t)* можно записать прямое и обратное преобразование Фурье:

$$S(j\omega) = \int_{-\infty}^{\infty} q(t) e^{-j\omega t} dt, \tag{6.1}$$

$$q(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(j\omega) e^{-j\omega t} d\omega. \tag{6.2}$$

По условиям теоремы спектр функции *q(t)* ограничен частотой ω_m, т.е.

$$S(j\omega) \neq 0 \text{ при } -\omega_m \le \omega \le \omega_m \tag{6.3}$$
$$S(j\omega) = 0 \text{ при } |\omega| > \omega_m.$$

Тогда выражение (6.2) можно представить в виде

$$q(t) = \frac{1}{2\pi} \int_{-\omega_m}^{\omega_m} S(j\omega) e^{-j\omega t} d\omega. \tag{6.4}$$

Спектр $S(j\omega)$ функции *q(t)* с учетом соотношений (6.3) можно рассматривать как функцию, заданную на интервале $(-\omega_m, \omega_m)$. Разложим функцию на этом интервале в ряд Фурье:

$$S(j\omega) = \sum_{n=-\infty}^{\infty} C_n e^{j\frac{\pi n \omega}{\omega_m}}, \tag{6.5}$$

* Дядюнов, А.Н. и др. Адаптивнык системы сбора и передачи аналоговой информации. – М. : Машиностроение, 1988. – 288 с.

где

$$C_n = \frac{1}{2\omega_m} \int_{-\omega_m}^{\omega_m} S(j\omega) e^{-j\frac{\pi n\omega}{\omega_m}} d\omega. \qquad (6.6)$$

Далее обратимся к выражению (6.4) и положим $t = - n\Delta_T$, где $\Delta_T = \pi / m$. Тогда

$$q(-nT) = \frac{1}{2\pi} \int_{-\omega_m}^{\omega_m} S(j\omega) e^{-j\frac{\pi n\omega}{\omega_m}} d\omega. \qquad (6.7)$$

Умножим обе части выражения (6.7) на π / m

$$\frac{\pi}{\omega} q(-nT) = \frac{1}{2\omega_m} \int_{-\omega_m}^{\omega_m} S(j\omega) e^{-j\frac{\pi n\omega}{\omega_m}} d\omega. \qquad (6.8)$$

Сравнивая это выражение с соотношением (6.6), получим

$$C_n = \frac{\pi}{\omega} q(-n\Delta_T). \qquad (6.9)$$

Подставляя вместо C_n полученное выражение (6.9) в формулу (6.5), имеем

$$S(j\omega) = \sum_{n=-\infty}^{\infty} \frac{\pi}{\omega_m} q(-n\Delta_T) e^{j\frac{\pi n\omega}{\omega_m}}. \qquad (6.10)$$

В итоге получаем

$$q(t) = \frac{1}{2\pi} \int_{-\omega_m}^{\omega_m} e^{j\omega t} d\omega \sum_{n=-\infty}^{\infty} \frac{\pi}{\omega_m} q(-n\Delta_T) e^{j\frac{\pi n\omega}{\omega_m}} =$$

$$= \frac{1}{2\omega_m} \sum_{n=-\infty}^{\infty} q(n\Delta_T) \int_{-\omega_m}^{\omega_m} e^{j\omega(t-n\Delta_T)} d\omega.$$

Но

$$\int_{-\omega_m}^{\omega_m} e^{j\omega(t-n\Delta_T)} d\omega = \frac{1}{j(t-n\Delta_T)} e^{j\omega(t-n\Delta_T)} \Big|_{-\omega_m}^{\omega_m} = \frac{2\sin[\omega_m(t-n\Delta_T)]}{t-n\Delta_T}$$

Окончательно получим

$$q(t) = \sum_{n=-\infty}^{\infty} q(n\Delta_T) \frac{\sin[\omega_m(t-n\Delta_T)]}{\omega_m(t-n\Delta_T)}. \qquad (6.11)$$

Из этого выражения следует, что непрерывная функция *q(t),* имеющая спектр, ограниченный частотой ω_m, может быть представлена в виде суммы, каждое слагаемое которой выражается функцией

$$y(t) = \frac{\sin[\omega_m (t - n\Delta_T)]}{\omega_m (t - n\Delta_T)}. \qquad (6.12)$$

Эта функция называется функцией отсчета. Рассмотрим следующие свойства функции отсчета:

- в момент времени $t=n\Delta_T$ функция отсчетов достигает своего наибольшего значения, равного единице;

- в моменты времени, кратные Δ_T ($t=(n \pm l)\Delta_T$, где l - любое число), функция отсчетов обращается в нуль.

Из формулы (6.11) следует, что сумма в каждый n-й момент времени определяется только одним n-м слагаемым, так как все остальные слагаемые в этот момент времени обращаются в нуль. Другими словами, функция $q(t)$ в дискретные моменты времени может быть представлена своими дискретами $x(n\Delta_T)$. Внутри же промежутка Δ_T функция $q(t)$ определяется всеми слагаемыми. С другой стороны, если заданы дискреты $x(n\Delta_T)$, то функция $q(t)$ может быть полностью восстановлена суммированием функций $x(n\Delta_T)y(t)$. Для восстановления функции $q(t)$ необходимо подать на вход фильтра с верхней границей пропускания ω_m импульсов типа δ-функции с амплитудой, соответствующей значениям непрерывной функции в точках отсчета и следующих друг за другом с периодом Δ_T.

На практике использования теоремы В.А. Котельникова наталкиваемся на ряд трудностей. Во-первых, представление непрерывной функции в виде дискрет через промежуток времени Δ_T не позволяет воспроизводить процесс, развивающийся по времени. Действительно, каждая новая дискрета $q(n\Delta_T)$ при восстановлении меняет всю непрерывную функцию на всем предшествующем интервале за исключением отсчетных значений. Во-вторых, реальный процесс, который описывается функцией $q(t)$, имеет, как правило, начало и конец. Но ограниченная во времени функция не может иметь ограниченный спектр и, следовательно, теорема В.А. Котельникова для такого сигнала не применима. Поэтому эту теорему можно рассматривать как приближенную для реальных сигналов в том смысле, что можно ввести разумные допущения на ширину реального спектра. Эту ширину реального спектра можно определить как интервал частот, вне которого спектральная плотность меньше некоторой заданной величины.

Если в выражении (6.11) ограничиться конечным числом членов, то это приводит к появлению ошибки:

$$\varepsilon = \left| q(t) - \sum_{-k}^{k} q(n\Delta_T) \frac{\sin[\omega_m (t - n\Delta_T)]}{\omega_m (t - n\Delta_T)} \right|. \qquad (6.13)$$

Не приводя здесь подробных рассуждений, отметим, что для ε справедливо следующее соотношение:

$$\varepsilon \le \frac{\sqrt{2}}{\pi} P \left| \sin \pi f_m t \right| \sqrt{\frac{1}{f_m} \frac{T}{T^2 - t^2}} = \varepsilon^* P >, \qquad (6.14)$$

где P - полная энергия, которую имеет сигнал $q(t)$; $(-T, T)$ – интервал квантования сигнала $q(t)$; $\varepsilon^* = \frac{\sqrt{2}}{\pi} \left| \sin \pi f_m t \right| \sqrt{\frac{1}{f_m} \frac{T}{T^2 - t^2}}$ - коэффициент ошибки; $f_m = \frac{1}{2\pi} \omega_m$.

Для определения граничной частоты f_m можно поступить следующим образом. Для рассматриваемого сигнала $q(t)$ ограничим его спектр значением f_m, которое определим из условия, что энергия суммы отброшенных гармоник не превышает энергии ошибки. Тогда полную энергию $P_э$ сигнала $q(t)$ можно представить следующим образом:

$$P_э = P_m + P_0, \qquad (6.15)$$

где P_m - энергия сигнала, ограниченного частотой f_m; P_0 - энергия отброшенных гармоник.

Полную энергию $P_э$ сигнала $q(t)$ можно найти по его спектральной плотности:

$$P_э = \int_0^\infty [q(t)]^2 dt = \frac{1}{2\pi} \int_0^\infty S(\omega) S^*(\omega) d\omega, \qquad (6.16)$$

где $S^*(\omega)$ - функция, комплексно сопряженная с функцией $S(\omega)$.

Аналогично, энергия P_m сигнала $q(t)$, ограниченного частотой f_m

$$P_m = \frac{1}{2\pi} \int_0^{\omega_m} S(\omega) S^*(\omega) d\omega. \qquad (6.17)$$

Пусть ошибка восстановления сигнала по его квантованным значениям не превышает величины $(\pm n\Delta P)$ (где n - коэффициент, а ΔP - шаг шкалы уровней). Если допустить, что любое значение ошиб-

ки в пределах заданной величины ($\pm n\Delta P$) равновероятно, то среднее значение энергии ошибки можно представить в виде

$$P_0 = \frac{1}{2n\Delta_p} \int_{-n\Delta_p}^{n\Delta_p} q^2 dq = \frac{(n\Delta P)^2}{3}. \qquad (6.18)$$

Аналогично, если положить, что любое значение сигнала *q(t)* в пределах от 0 до *k* (где *k* - максимальное число шагов шкалы уровней) равновероятно, то среднее значение полной энергии

$$P_э = \frac{1}{k\Delta P} \int_{0}^{k\Delta P} q^2 dq = \frac{(k\Delta P)^2}{3}. \qquad (6.19)$$

Из выражений (6.18), (6.19) следует, что

$$P_0 = \frac{n^2}{k^2} P_э, \qquad (6.20)$$

где – относительная ошибка.

Подставляя выражение (6.20) в формулу (6.15), получим

$$\left(1-\Delta^2\right) P_э = P_m. \qquad (6.21)$$

Если в уравнении (6.15) заменить $P_э$, P_m их значениями, вычисленными для заданного сигнала *q(t)* соответственно по формулам (6.16), (6.17), то полученные соотношения будут определять связь между относительной ошибкой Δ и частотой f_m, которая и используется для выбора граничной частоты f_m.

Контрольные вопросы

1. Как изменяются требования к контрольно-измерительной аппаратуре в зависимости от измеряемой величины?
2. Как влияет отсутствие регистра на структуру АЦП?
3. Что такое дискретизация?
4. С чем связан выбор интервала дискретизации по времени?
5. Какие существуют ограничения в практическом применении теоремы В.А. Котельникова?

Глава 7

ОСНОВЫ ПРОГРАММИРОВАНИЯ

При программировании малых ЭВМ чаще всего применяют команды пересылки и процессорные команды, использующие регистры и простые режимы адресации.

Реализация программ основывается не только на записи кодов команд, но и на знании принципов выполнения команд и методов адресации. Чтобы избежать ошибок при программировании, важно знать типы команд, как выполняется команда и какие ресурсы микропроцессора привлекаются для ее выполнения.

Обычно малые ЭВМ не имеют достаточного набора команд, чтобы выполнить каждую операцию одной командой. Мерой сложности ЭВМ является число операций, выполняемых одной командой, и гибкость команд при выполнении определяемых ими операций.

ВИДЫ КОМАНД

Практически во всех ЭВМ команды подразделяются следующим образом.

Передачи между регистрами, памятью и регистрами, ячейками памяти, в/из стека.

Арифметика — сложение, вычитание, умножение, деление и изменение знака. Имеются три типа чисел: целые (с фиксированной точкой), с плавающей точкой, десятичные (или BCD).

Логика — поразрядные операции ИЛИ, И, исключающее ИЛИ и дополнение.

Сдвиг и ротация — сдвиг вправо, сдвиг влево, ротация вправо, ротация влево, сдвиги и ротация нескольких слов (или байт).

Индексирование и счет—инкремент и декремент.

Манипуляции с битами — селективные установка, сброс и проверка бит в слове или байте. Часто используются с интерфейсными регистрами ВВ.

Зацикливание — комбинация инкремента или декремента, сравнения, проверки и разветвления; целью является повторяющееся выполнение сегмента программы.

Переход — обычно команды выбираются из смежных ячеек памяти, но иногда необходимо нарушить естественный порядок команд. Эту функцию выполняют команды переходов. Они разделяются:

1) на *безусловные,* когда переход осуществляется независимо от состояния ЦП;

2) *условные (разветвления),* которые выполняются или не выполняются в зависимости от некоторой комбинации флажков PSW. Иногда они реализуются как комбинация пропуска, означающего пропуск следующей команды, с последующим безусловным переходом;

3) *переходы к подпрограммам и возвраты.* Адрес возврата запоминается переходом и используется возвратом. Могут быть безусловными и условными.

Передачи ВВ — для инициирования и выполнения передач данных при ВВ, проверки состояния ВВ, выдачи приказов на ВВ и т.д.

РЕЖИМЫ

Для взаимодействия с различными компонентами в ЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров ЦП и интерфейсных регистров ВВ. Для этого каждой запоминающей ячейке присваивается адрес, т. е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров ЦП, а иногда и отдельные адресные пространства для интерфейсных регистров ВВ и внутренней памяти. Это означает, что два или четыре запоминающих элемента могут иметь адрес 0: ячейка памяти, регистр ЦП, регистр ВУ и ячейка внутренней памяти. Выбор любого из этих элементов определяется командой.

Адресное пространство регистров ЦП. Обычно в составе ЦП приходится идентифицировать 4 – 32 регистра, что требует от 2 до 5 бит.

Адресное пространство памяти. Максимальное число ячеек памяти равно 2^n, где n – число бит в адресе. Говорят, что ЭВМ *адресует байт,* если каждый байт в памяти имеет различный адрес, или *адресует слово,* если каждое слово имеет различный адрес, а каждый байт не имеет. В большинстве 8-битных МП адрес памяти имеет 16 бит, поэтому максимальная емкость памяти составляет 2^{16} байт (при

адресации байт) или 2^{16} слов (при адресации слов). Если слово длиннее байта, то основное достоинство ЭВМ с адресацией слов — увеличенная емкость памяти, но так как символ внешнего кода, например ASCII, может содержаться в байте, то многие операции эффективнее выполняются в ЭВМ с адресацией байт. Почти во всех современных МП используется адресация байт.

Основная память некоторых микро-ЭВМ разделена на *страницы.* В этом случае младшие биты (8 – 10 бит) определяют позицию внутри страницы, которая в свою очередь определяется старшими битами. Например, старшие 6 бит определяют одну из 64 страниц, а младшие 10 бит – один из 1024 байт (или слов) в странице. Сегментация на страницы применяется в некоторых способах адресации. *Базовой* называется страница, начинающаяся с нулевого адреса памяти. *Текущей* называется страница, которая содержит адрес, находящийся в программном счетчике.

Стек, или внутренняя память. Иногда кроме основной памяти имеется стек, или внутренняя память. К нему можно обращаться, используя отдельное от основной памяти адресное пространство. Хотя стеки и очень важны, сейчас предпочитают организовывать их в основной памяти.

Интерфейсные регистры ВВ. Часто адреса интерфейсных регистров ВВ находятся в том же адресном пространстве, что и память, но иногда они имеют отдельное адресное пространство. В последнем случае обычно имеется $2^3 – 2^8$ адресов, длина которых составляет 3 – 8 бит. Если регистры ВВ находятся в адресном пространстве памяти, систему памяти следует спроектировать так, чтобы она не реагировала на адреса интерфейсных регистров ВВ.

Так как память хранит команды и данные, для ЭВМ разработано множество способов обращения к памяти, называемых *режимами адресации.* Ниже определены наиболее важные режимы адресации. В каждой микро-ЭВМ реализованы только некоторые из приведенных режимов. Данные определения не являются стандартными и могут меняться в различных руководствах.

Прямая адресация. Адрес определяется как часть команды.

Регистровая адресация. Это разновидность прямой адресации, когда операнд находится в регистре, а адрес регистра является частью команды.

Косвенная адресация. В ячейке, адрес которой определяется как часть команды, содержится адрес. Этой ячейкой может быть либо регистр, и тогда говорят про *регистровую косвенную адресацию,* либо ячейка памяти. (В части руководств регистровая косвенная адресация называется прямой адресацией). В некоторых ЭВМ допускается несколько и даже неопределенное число косвенных указателей; в этом случае ячейка обращается к другой ячейке, которая обращается к новой ячейке и так далее, каждое обращение представляет собой *уровень косвенности.*

Базовая адресация. Адрес образуется в результате сложения содержимого ячейки памяти или регистра с определенным числом, называемым *смещением.* Базовую адресацию можно использовать вместе с косвенной адресацией. Она в основном применяется при обращении к массивам или для перемещения программы в памяти.

Индексирование. Индексированием называется инкремент или декремент адреса, когда ЭВМ обращается к смежным или равномерно распределенным адресам. Этого можно достичь путем последовательного изменения заданного адреса либо путем прибавления к фиксированному адресу числа, на которое производится инкремент или декремент. В некоторых 16-битных МП, например 8086 фирмы „Intel", допускаются одновременные базовая адресация и индексирование (окончательный адрес равен сумме смещения базового адреса и индекса). Индексирование в основном применяется для последовательной адресации элементов в массивах.

Автоинкремент/автодекремент. Разновидность индексирования, когда в команде производится автоматическое увеличение на 1 (или уменьшение на 1) индекса.

Адресация базовой страницы. Разновидность косвенной адресации, когда указанный адрес представляет собой адрес ячейки в базовой странице, а содержимое этой ячейки – нужный адрес.

Адресация текущей страницы. Разновидность косвенной адресации, когда указанный адрес представляет собой адрес ячейки в текущей странице, а содержимое этой ячейки – нужный адрес.

Относительная адресация. Адрес равен сумме числа и текущего содержимого программного счетчика. Число представляет собой адрес операнда относительно текущей команды и обычно является частью команды, но может содержаться в рабочем регистре.

Непосредственная адресация. Информация называется непосредственной, если она является частью команды; поэтому для получения информации адресация не нужна.

Имеются способы адресации, представляющие комбинации приведенных выше, например косвенная относительная адресация.

При выполнении любой программы процессор обращается к памяти, в которой хранятся команды и данные. В командах преобразований данных определяются адреса, которые указывают местоположение необходимых данных, а в командах передачи управления – адреса команд, которым передается управление, т.е. адреса переходов. Способ, или метод, определения в команде адреса операнда или адреса перехода называется режимом адресации, или просто адресацией.

В зависимости от внутренней структуры процессора и способа обращения к памяти и внешним устройствам различают следующие структуры команд: одноадресные (1А), двухадресные (2А), трехадресные (3А), безадресные (БА), команды с большой длиной слова (VLIW – БДС) (рис. 7.1).

По структуре команды содержат, как правило, код операции (КОП) и операнды (А1, А2 и т.д.). Еще в литературе их могут характеризовать как одно-, двух- или трехбайтные.

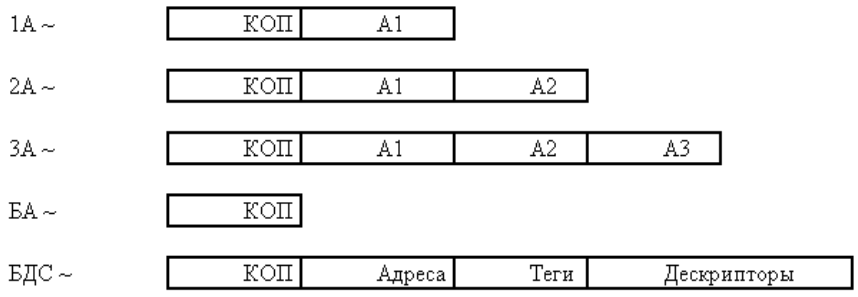


Рис. 7.1. Структура команд

Операндами могут быть адреса регистров, ячеек памяти, конкретные данные. Принципиальное значение имеют методы адресации, учитывающие структуру процессора, которая строится исходя из этих методов.

Поэтому в настоящее время в процессорах применяется много различных режимов адресации, направленных на достижение следующих целей:

– определение адреса памяти наименьшим числом бит, что сокращает длину команд;

– вычисление адреса относительно текущей команды (так называемая относительная адресация), обеспечивающее загрузку программ без модификаций в любую область памяти;

– осуществление доступа к ячейкам памяти, адреса которых вычисляются при выполнении программы, что упрощает доступ к регулярным структурам данных;

– оперирование адресами в такой форме, которая наиболее удобна для таких структур данных, как массивы и стеки [1].

В качестве операндов в различных микропроцессорах используются регистры, которые могут обозначаться буквами латинского алфавита A, B, C, D либо буквой и цифрой R1, R2, R3, парой букв AL, AH и, как правило, таких регистров не менее восьми. Рассматривая основы программирования, мы не будем придерживаться конкретного микропроцессора, поскольку система команд каждого из них опирается на языковое подмножество языка Assembler, имеющего одинаковое представление и однотипное использование. Команды, заданные кодом операции, используют эти регистры в разных комбинациях. Запись команд является аббревиатурой слов, характеризующих операцию, например move (двигать) имеет запись MOV, adition (сложить) – ADD, substrakt (вычесть) - SBB и т.д.

Основные виды адресации для малых вычислительных систем: прямая, косвенная, непосредственная, индексная и её вариации.

Наиболее общий формат двухоперандной команды приведен на рис. 7.2 (закрашены необязательные байты команды).

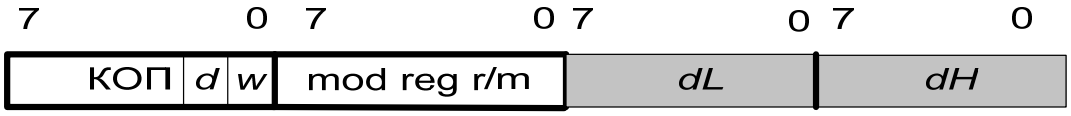


Рис. 7.2. Общий формат команды

Первый байт (их четыре) команды содержит код операции (КОП) и два однобитных поля – направления передачи данных *d* и размер слова *w*. Поле *d* определяет направление передачи: если *d*=1, то направление в МП, а если *d*=0, то направление из МП. Само направление относится ко второму операнду — регистру, определяе-

мому полем *reg* второго байта команды. Этот байт называется постбайтом (или просто байтом) режима адресации. Поле *w* идентифицирует тип (размер) операнда: если *w* = 1, команда оперирует словом, а если *w*=0,– байтом. Таким образом, в зависимости от значений полей *d* и *w* имеются четыре возможности (табл. 7.1).

Таблица 7.1

Назначение полей d и w

Передача, или операция	*d*	*w*
Байт из регистра в память или регистр	0	0
Слово из регистра в память или регистр	0	1
Байт в регистр из памяти или регистра	1	0
Слово в регистр из памяти или регистра	1	1

Участвующие в операции регистры или регистр и ячейку памяти идентифицирует постбайт, состоящий из трех полей: *mod* – режим, *reg* – регистр и *r/m* – регистр/память. Поле *reg* определяет второй операнд, обязательно находящийся в регистре. Способ кодирования внутренних регистров МП в поле *reg* представлен в табл. 7.2.

Таблица 7.2

Адресация внутренних регистров МП

Поле *reg* (и *r/m)*	8-битные регистры	16-битные регистры
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Из таблицы видно, что регистры данных AX – DX, а также указательные SP, BP и индексные регистры SI, DI адресуются одинако-

вым образом. Данное обстоятельство подчеркивает правомерность объединения всех этих регистров в группу общих регистров.

Поле *mod* определяет используемый режим адресации. В частности, оно показывает, как интерпретируется поле *r/m* для нахождения первого операнда. Если *mod*—11, операнд содержится в регистре, а в остальных случаях первый операнд находится в памяти.

Когда поле *mod* содержит 11 (регистровая адресация), поле *r/m* определяет 8- или 16-битный регистр в соответствии с табл. 7.2.

В остальных случаях адресуется память *m* и поле *mod* определяет, как используется смещение *disp* (необязательное), находящееся во втором и третьем байтах команды

mod = 00 – смещение отсутствует; disp=0,
01 - disp=*dL,* команда содержит 8-битное смещение, которое расширяется со знаком до 16 бит;
10 - disp=*dH, dL* – команда содержит два байта смещения (табл. 7.3).

Таблица 7.3

Формирование эффективного адреса памяти

Поле *r/m*	Эффективный адрес	
000	EA=(BX)+(SI)	+ *disp*
001	EA=(BX)+(DI)	+*disp*
010	EA=(BP)+(SI)	+*disp*
011	EA=(BP)+(DI)	+*disp*
100	EA= (Sl)	+*disp*
101	EA= (DI)	+*disp*
110	EA=(BP)	+*disp*
111	EA=(BX)	+*disp*

Контрольные вопросы

1. В чем заключается принцип адресации?
2. Опишите существующие режимы адресации.
3. Какие существуют виды команд?

РЕГИСТРОВАЯ АДРЕСАЦИЯ

При использовании регистровой адресации операнд находится в одном из общих регистров МП, а в некоторых командах – в одном из сегментных регистров. Регистр может быть определен в байте кода операции или в постбайте режима адресации. В двухоперандных командах определяются два регистра. Регистры идентифицируются 3-битными полями *reg* и *r/m* (когда *mod* = 11). В зависимости от значения бита *w* в операции участвует 8- или 16-битный регистр (или регистры).

Команды, оперирующие содержимым регистров, оказываются наиболее короткими и выполняются за минимальное время, так как не требуют цикла шины для обращения к памяти.

В ассемблерных программах регистры обозначаются зарезервированными именами: AL, AH, AX, BL, BH, BX и т. д.

Примеры ассемблерных команд с регистровой адресацией приведены ниже.

```
MOV AX, SI      ; Передать содержимое  SI в AX
ADD DI, BX      ; Прибавить содержимое BX к DI
AND CL, AX      ;Ошибка — несоответствие  размеров
XOR AL, AH      ;Исключающее ИЛИ регистров AL и AH
```

Принцип выполнения команды пересылки из регистра в регистр микропроцессора для восьмибитных регистров показан на рис. 7.3.

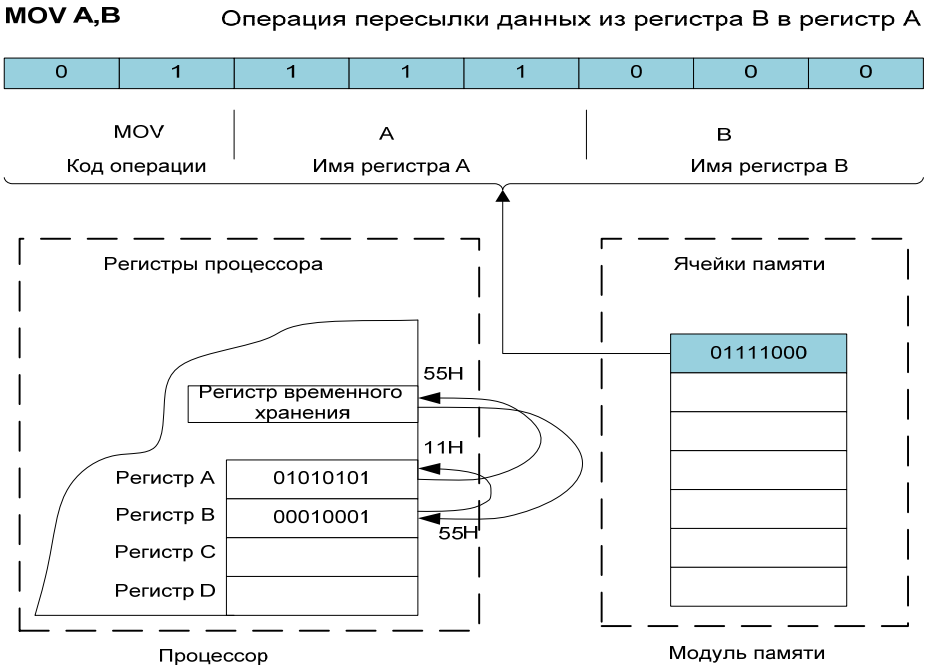


Рис. 7.3. Выполнение команды пересылки данных

Выполнение команды осуществляется в следующей последовательности. Первоначально содержимое регистра A перемещается в регистр временного хранения. Затем освободившийся регистр A заполняется содержимым регистра B. После чего из временного регистра содержимое перемещается в регистр B. Таким образом выполнение команды пересылки завершается.

НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ

Непосредственные операнды (называемые иногда литералами) представляют собой константы длиной 8 или 16 бит, содержащиеся в командах. Доступ к таким операндам осуществляется очень быстро, так как они находятся во внутренней очереди команд, и циклов шины для считывания операндов из памяти не требуется. Непосредственные операнды изменить в ходе выполнения команды невозможно.

Данный режим предусмотрен для большинства двухоперандных команд. Наличие в командах постбайта режима адресации позволяет манипулировать непосредственными операндами и содержимым регистров или ячеек памяти. Однако в МП нет команд непосредственной загрузки сегментных регистров и включения константы в стек, что вызывает некоторые неудобства при программировании. Стандартный непосредственный операнд имеет длину 16 бит, а короткий – 8 бит (при необходимости он расширяется со знаком до 16 бит).

Константы в командах применяются для нескольких целей, например для инициализации регистров и переменных в памяти, в качестве масок для манипуляций отдельными битами, для сравнения переменных с граничными значениями и т. д.

Непосредственные операнды находятся в конце формата команд после смещения, если оно имеется, причем первым следует младший байт константы (рис. 7.4).

Примеры записи команд с непосредственными операндами на языке ассемблера приведены ниже.

```
SUB   AL,   30H    Вычесть из содержимого AL число 48
MOV   CL,   10     Загрузить в регистр CL число 10;
AND   AX,   0F00H  Выделить старшие 4 бита AX;
XOR   DH    1      Инвертировать младший бит DH;
CMP   BL,   40H    Сравнить содержимое BL с числом 64
```

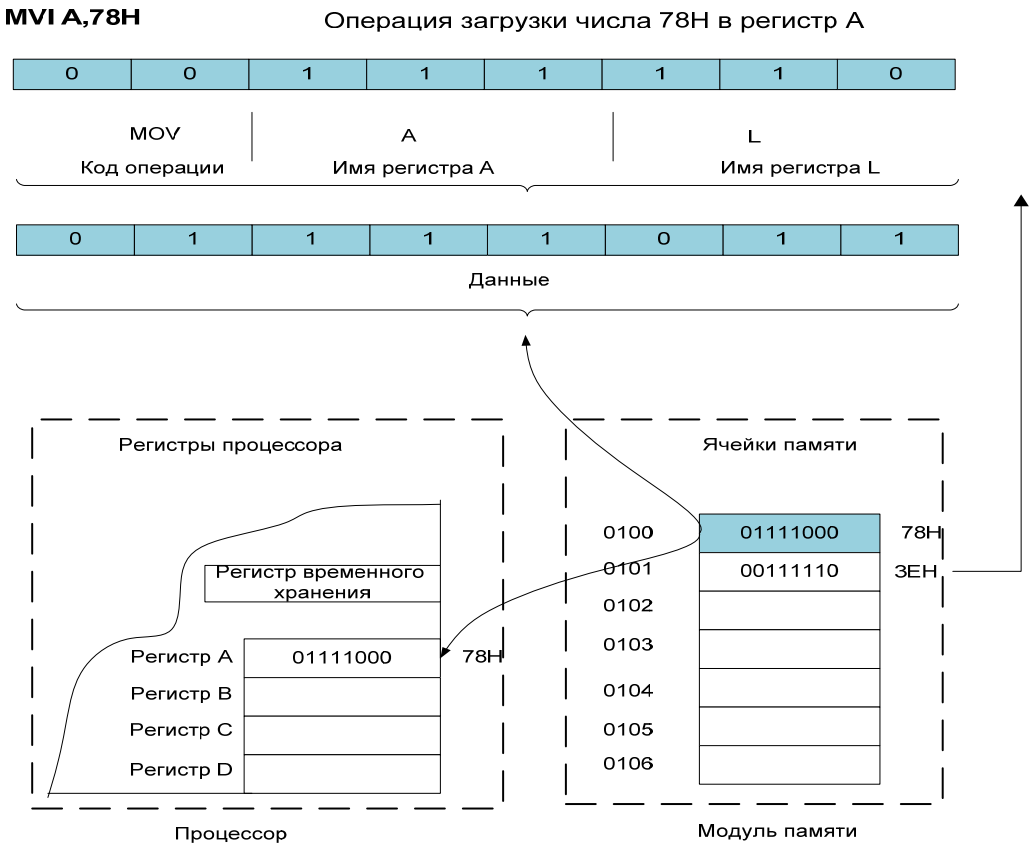


Рис.7.4. Выполнение команды непосредственной загрузки данных

ПРЯМАЯ АДРЕСАЦИЯ

Прямая адресация представляет собой простейший режим адресации – эффективный адрес EA берется прямо из поля смещения команды, и никакие регистры для его вычисления не привлекаются. Этот режим применяется для обращения к простым переменным, которые называются также скалярами.

Примеры ассемблерных команд с прямой адресацией приведены ниже.

```
MOVAX, GAMMA    Загрузить в AX переменную GAMMA
ADD TEMP, BL    Прибавить (BL) к переменной TEMP
```

КОСВЕННАЯ РЕГИСТРОВАЯ АДРЕСАЦИЯ

Разновидностью данного режима является длинная прямая адресация, в которой команда содержит базовый адрес сегмента и смещение. В этом случае обеспечивается прямой доступ к операнду с любым логическим (и физическим) адресом. Однако длинная прямая адресация допускается только в командах межсегментных переходов и вызовов и не может применяться в командах, оперирующих

данными. Метки, которым передают управление команды с данным режимом адресации, должны иметь тип FAR.

В этом режиме эффективный адрес EA операнда находится в одном из базовых или индексных регистров (рис. 7.5). Из четырех адресных регистров BP, BX, SI и DI в косвенной адресации могут использоваться только регистры BX, SI и DI. Косвенная адресация через указатель базы BP моделируется при помощи базовой адресации с нулевым смещением.

Данный режим с некоторыми вариантами применяется во всех современных процессорах. Он позволяет вычислять адреса во время выполнения программы, что часто требуется, например, для обращения к различным элементам регулярных структур данных.

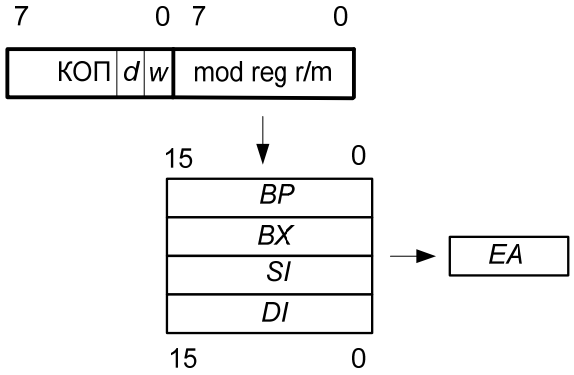


Рис. 7.5. Регистровая косвенная адресация

При модификации содержимого регистра одна и та же команда оперирует различными ячейками памяти. Для изменения содержимого регистра применяется команда загрузки эффективного адреса LEA, а также все арифметические и логические команды.

Отметим, что в командах безусловного перехода и вызова подпрограммы CALL с регистровой косвенной адресацией допускается указывать любой 16-битный общий регистр (рис. 7.6).

Примеры записи команд с косвенной регистровой адресацией на языке ассемблера приведены ниже.

ADD	AX,[DI]	Прибавить к AX содержимое ячейки памяти, адресуемой DI
MOV	[SI],CL	Передать (CL) в ячейку памяти, адресуемую SI
MOV	DI,[DI]	Передать в DI содержимое ячейки памяти, адресуемой DI
JMP	AX	Перейти по адресу из AX
CALL	BX	Вызвать подпрограмму по адресу из BX

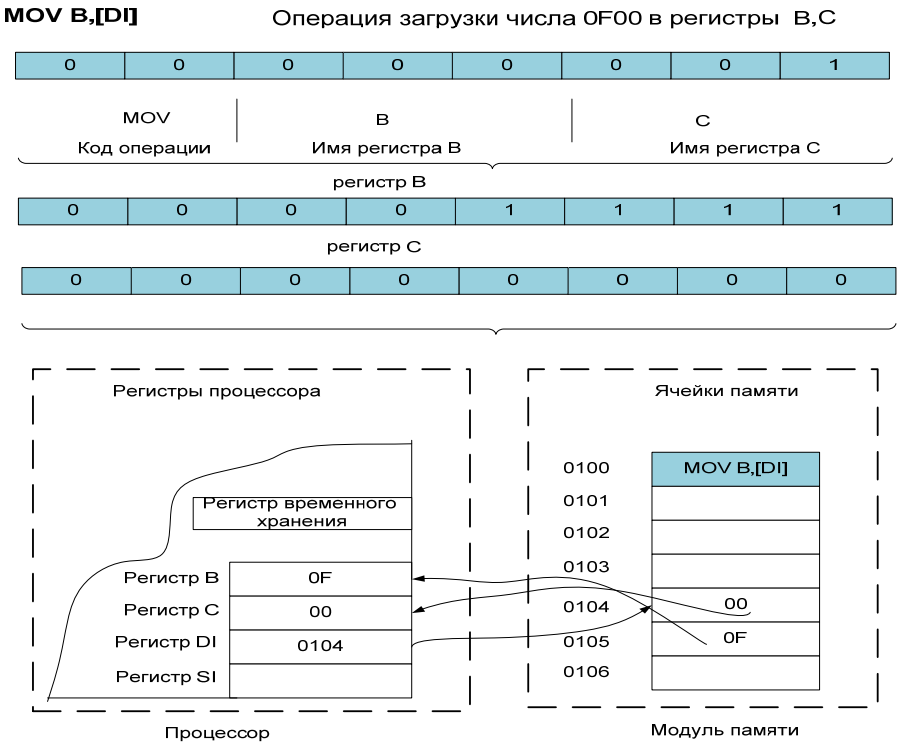


Рис. 7.6. Выполнение регистровой косвенной адресации

БАЗОВАЯ АДРЕСАЦИЯ

При базовой адресации (называемой также адресацией по базе, или адресацией типа «база плюс смещение») эффективный адрес операнда является суммой значения смещения и содержимого регистров BX или BP. Напомним, что при определении BP в качестве базового адреса шинный интерфейс обращается к операнду в текущем сегменте стека (если нет префикса замены сегмента). Это делает базовую адресацию с регистром BP очень удобным средством обращения к данным, находящимся в стеке, что требуется, например, при передаче подпрограммам параметров через стек.

Основное применение базовой адресации связано с доступом к элементам структур данных, когда смещение (номер) элемента известно при ассемблировании программы, а базовый (начальный) адрес структуры должен вычисляться при выполнении программы.

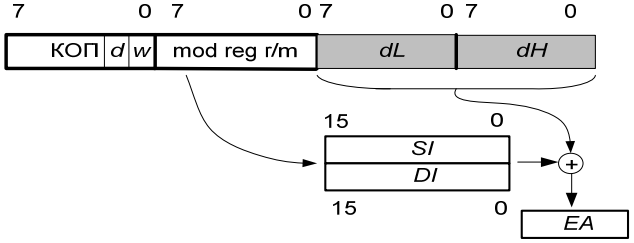


Рис. 7.7. Базовая адресация

Таким образом, структура данных может размещаться в различных областях памяти, а модификация содержимого базового регистра обеспечивает доступ к этим областям. Базовый регистр указывает на начало структуры данных, а требуемый элемент адресуется с помощью смещения (расстояния) от базы (рис. 7.8). Смещения, содержащиеся в команде, могут иметь длину 8 или 16 бит и интерпретируются как знаковые целые. Размер смещения программа-ассемблер выбирает автоматически с учетом атрибутов операндов.

В языке ассемблера используются два обозначения базовой адресации. Первое обозначение имеет вид [B_{REG}] DISP и соответствует адресации структуры данных типа «запись» в языке Паскаль. Второе обозначение имеет вид [B_{REG}+DISP] и явно указывает на необходимость суммирования содержимого регистра и смещения при вычислении эффективного адреса. В обоих случаях обозначение B_{REG} подразумевает один из базовых регистров.

Примеры записи команд с базовой адресацией на языке ассемблера приведены ниже.

MOV AX, [BP] 10 Обе команды передают шестое слово массива,

MOV AX, [BP+10] адресуемого BP

ADD [BX]TEMP, CX Прибавить CX к слову TEMP в массиве, адресуемом BX

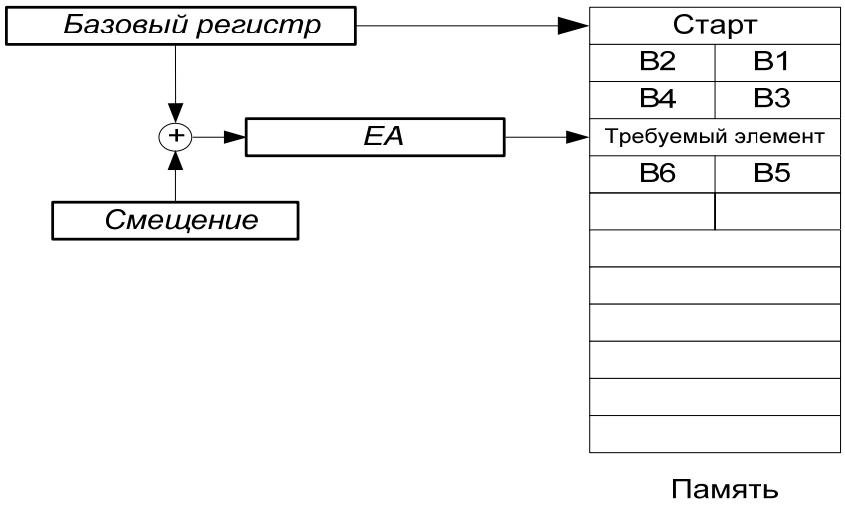


Рис. 7.8. Пример базовой адресации

В режиме индексной адресации (называемой также адресацией с индексированием, адресацией типа «база плюс индекс») эффективный адрес вычисляется как сумма смещения, находящегося в команде, и содержимого регистров SI или DI (рис. 7.9). Данный режим обычно применяется для обращения к элементам одномерных массивов. Смещение определяет при ассемблировании начальный адрес массива, а значение в индексном регистре – нужный элемент. Так как все элементы массива имеют одинаковый размер, простые манипуляции над содержимым индексного регистра позволяют обращаться к любому элементу массива.

ИНДЕКСНАЯ АДРЕСАЦИЯ

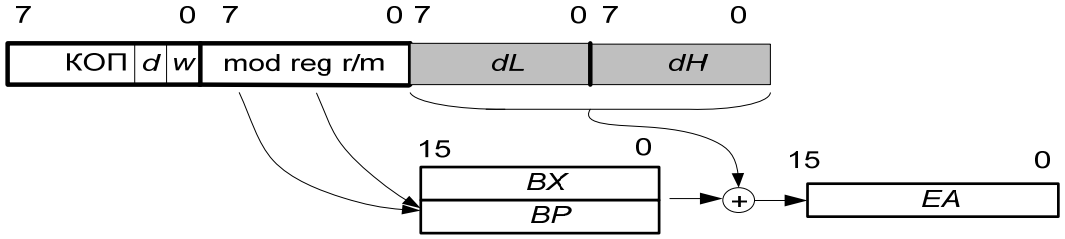


Рис. 7.9. Индексная адресация

По существу режимы базовой и индексной адресации в МП К1810 аналогичны. Это объясняется тем, что базовые адреса и индексные значения имеют одинаковую длину 16 бит.

В языке ассемблера индексная адресация обозначается в виде ADDR16 [I_{REG}], как это принято в языках высокого уровня. Здесь ADDR16 – 16-битное смещение, а I_{REG} обозначает один из индексных регистров: SI или DI.

Примеры команд с индексной адресацией на языке ассемблера приведены ниже.

```
MOV   ARRAY[SI]      Передать AX в элемент массива
ADD   CX, ROW[DI]    Прибавить к CX элемент массива
XOR   DL,            Операция с элементом массива
```

Данный режим называется также базово-индексной адресацией, или адресацией типа «база плюс индекс плюс смещение». Здесь эффективный адрес равен сумме содержимого базового регистра, индексного регистра и смещения, находящегося в команде, (рис. 7.10). Базовая индексная адресация является наиболее гибким режимом, так как два компонента адреса можно определять и варьировать при выполнении программы.

БАЗОВАЯ ИНДЕКСНАЯ АДРЕСАЦИЯ

Регистры BP и BX используются как базовые, а регистры SI и DI — как индексные, т. е. всего получается четыре.

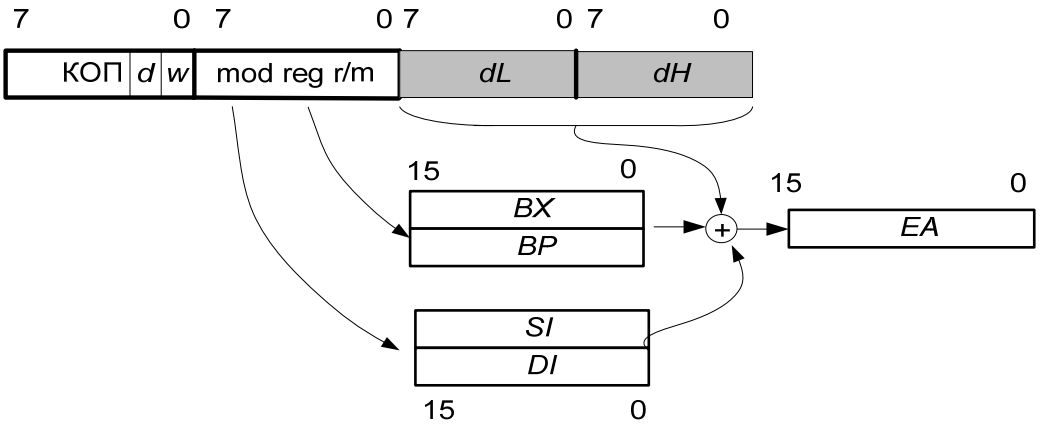


Рис. 7.10. Базовая индексная адресация

С появлением флеш-памяти большого объема возникла необходимость использовать адресацию вида PAE.

Аббревиатура PAE расшифровывается как Physical Address Extension (расширение физического адреса). Впервые технология была представлена в Pentium Pro. Наличие PAE определяется с помощью инструкции cpuid: источник eax = 1, установленный в 1-й бит 6-го регистра edx, говорит о поддержке PAE процессором. Включается же PAE установкой пятого бита регистра CR4 в единицу.

С помощью PAE 32-битный указатель может адресовать физическую память, находящуюся выше адреса 0xFFFFFFFF (2^{32} - 1, четыре гигабайта), а точнее, позволяет адресовать до шестидесяти четырех (2^{36}) гигабайт физической памяти. Казалось бы, как это возможно? Поговорим об этом чуть позже. Сначала посмотрим на этот процесс с точки зрения аппаратной части. Когда процессор производит чтение/запись значения ячейки по какому-либо адресу, он выдает адрес ячейки на шину адреса и генерирует сигнал чтения/записи. (Все, конечно, намного сложнее, но не будем углубляться дальше). Процессор, поддерживающий PAE, имеет 4 дополнительных (к существовавшим ранее тридцати двум) контакта, на которые подается адрес. $2^{32} * 2^4 = 2^{36}$. Таким образом, процессор может обращаться к адресам, лежащим за пределами значения 2^{32}.

Самое главное понять, что любая страница может быть отображена на физический адрес, лежащий выше четырех гигабайт, т.е., например, линейный адрес 0x0 может быть преобразован в физический адрес 0x100000000. Чтобы познакомиться с этим процессом поближе, рассмотрим структуры, которые использует процессор для

трансляции линейного адреса. С одной стороны, изменения этих структур минимальны. Все также есть каталог таблиц (далее именуемый просто "каталог") и таблицы страниц. Однако кое-что добавлено, кое-что изменено. Теперь адрес разбивается на 4 части, и правило "10-10-12" заменено на "2-9-9-12". На индексирование элементов таблицы страниц и каталога теперь выделено 9 бит. Помимо этого появилась новая структура -- Page Directory Pointer Table (PDPT). CR3 теперь содержит адрес таблицы каталогов, а не каталога. О ней более подробно ниже (рис 7.11).

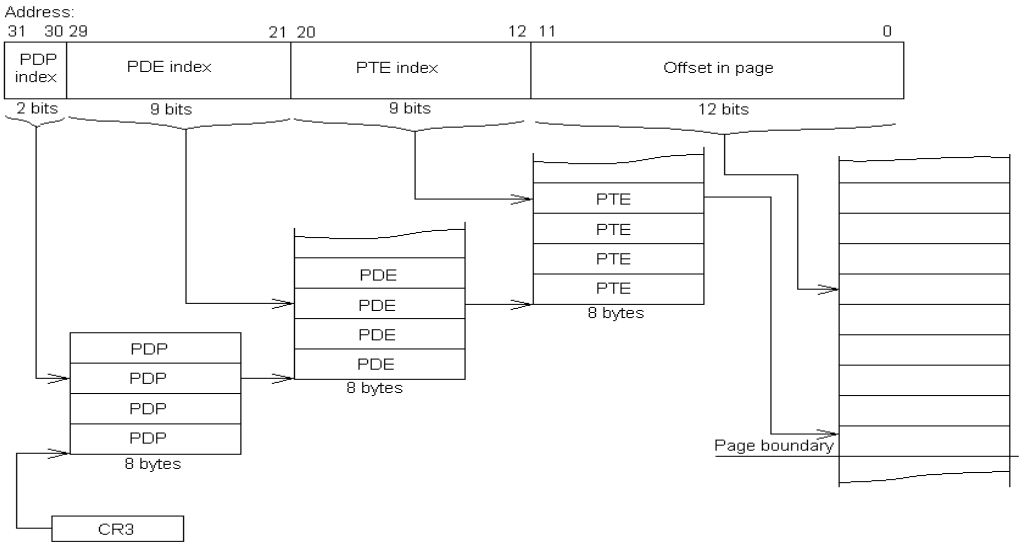


Рис. 7.11. Интерпретация бит линейного адреса

Для начала разберем формат элемента таблицы страниц. Из него, как вы знаете, в конце концов берутся верхние 20 бит физического адреса страницы в обычном (10-10-12) преобразовании адреса (рис.7.12.)

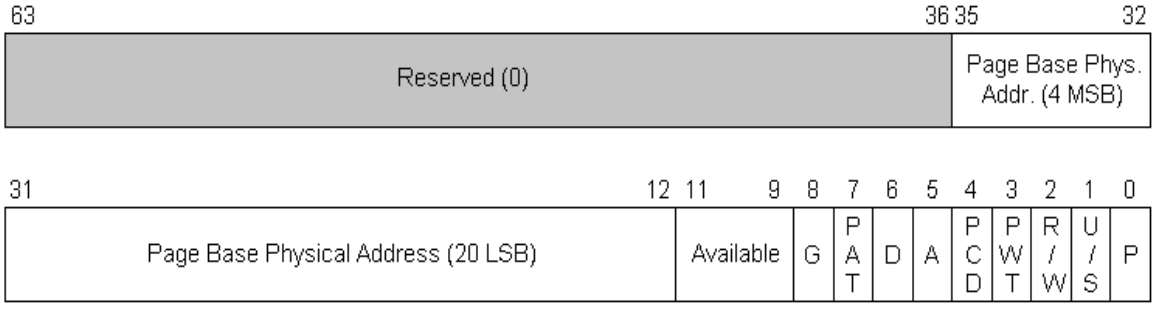


Рис. 7.12. Элемент таблицы страниц

Первое, что бросается в глаза, – размер элемента страницы увеличен с тридцати двух бит до шестидесяти четырех. В принципе это

является основным моментом преобразования с использованием PAE: старшие 20 бит линейного адреса заменяются на старшие 24 бита физического. При размере страниц 4 килобайта это дает $2^{24} * 2^{12} = 2^{36}$ бит физического адреса. Логично предположить, что размер таблицы страниц в таком случае должен увеличиться вдвое. Однако это не так. Теперь таблица страниц вместо привычных тысячи двадцати четырех элементов размером 4 байта, содержит пятьсот двенадцать, размером 8 байт, а значит, что для индексации элемента таблицы страниц теперь достаточно девяти бит линейного адреса ($2^9 = 512$). Таким образом, элементы таблицы страниц теперь выровнены по границе 8 байт относительно базового адреса таблицы страниц. Сама же таблица страниц осталась выровненной по границе 4 килобайта. Значения флагов, находящихся в младших двенадцати битах, не изменились.

Элемент каталога претерпел аналогичные изменения (рис.7.13). Он также расширен до восьми байт, а значит, что таблицы страниц теперь могут размещаться за пределами адреса 0xFFFFFFFF. Выбор элемента каталога осуществляется аналогично выбору элемента таблицы страниц – 9 бит (биты с двадцать первого по двадцать девятый) умножаются на 8 и складываются с базовым адресом каталога. Каталог выровнен по границе 4 килобайта. Значения флагов, находящихся в младших двенадцати битах, не изменились.

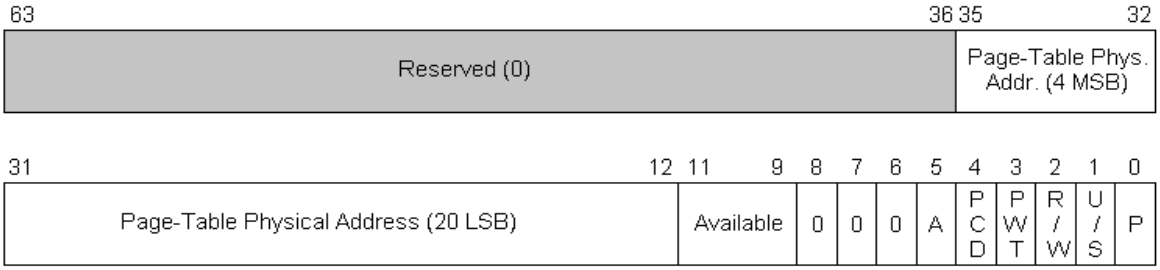


Рис. 7.13. Элемент каталога таблиц.

Итак, подытожим то, что мы имеем: 12 младших бит линейного адреса – смещение в странице, следующие за ними 9 бит – индекс в таблице страниц, следующие 9 – индекс в каталоге таблиц. 12 + 9 + 9= = 30. Что же происходит с оставшимися двумя битами?

В игру вступает Page Directory Pointer Table – таблица каталогов. Дело в том, что теперь каталогов может быть несколько, а именно 4. Физические адреса каталогов хранятся в таблице каталогов – массиве

из четырех 64-битных элементов, каждый из которых содержит 24 старших бита физического адреса одного из каталогов таблиц. Таким образом, каждый элемент таблицы каталогов представляет собой как бы регистр CR3, но с небольшими отличиями. Во-первых, CR3 содержал 20 старших бит физического адреса каталога таблиц, а элемент таблицы каталогов содержит 24 старших бита, что позволяет каталогам размещаться в памяти выше четырех гигабайт. Во-вторых, каждый элемент таблицы каталогов имеет бит присутствия P, позволяющий выгружать каталог таблиц из памяти. Теперь вспомним о двух незадействованных ранее старших битах линейного адреса. Их значение используется как индекс в таблице каталогов, что можно видеть на рис. 7.14.

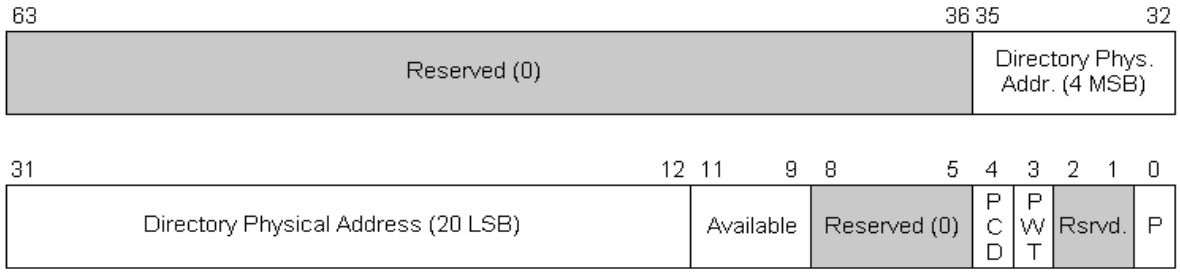


Рис. 7.14. Элемент таблицы каталогов.

Нетрудно догадаться, что CR3 (отныне именуемый PDPTR(Page Directory Pointer Table Register)) содержит физический адрес таблицы каталогов (рис.7.15). Соответственно изменен его формат. Старшие 27 бит CR3 теперь содержат физический адрес таблицы каталогов, два из младших пяти бит управляют кэшированием, три зарезервировано, а значит, адрес таблицы каталогов таблиц должен быть выровнен по границе 32 байта. Так как размер CR3 не изменился, таблица каталогов должна находиться в пределах 32-битного адресного пространства.

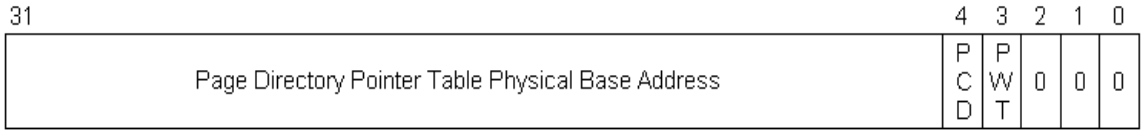


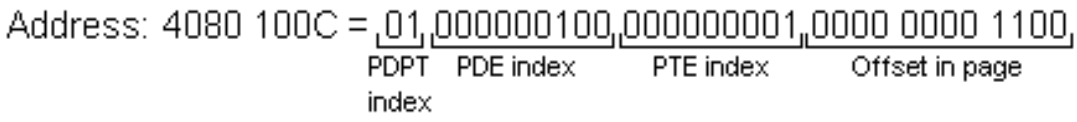
Рис.7.15. Регистр CR3

При использовании PAE 32-битное адресное пространство как бы разбивается на 4 части, каждую часть обслуживают свой каталог и

таблица страниц. Это происходит за счет того, что количество элементов в каталоге и таблице уменьшено вдвое. И если раньше один каталог и 1024 таблицы страниц на каждый элемент каталога охватывали 32-битное адресное пространство целиком, то теперь каждый каталог содержит 512 (2^9) указателей на таблицы страниц, таблица страниц, в свою очередь, содержит 512 (2^9) физических адресов. Значит, каталог и таблица страниц теперь охватывают $2^9 * 2^9 * 2^{12}$ (размер страницы не изменился) равен 2^{30}, т.е. 1 гигабайту. Таким образом, адреса 0 - 3FFF FFFF обслуживает первый элемент таблицы каталогов таблиц, 4000 0000 - 7FFF FFFF – второй, 8000 0000 - BFFF FFFF – третий и C000 0000 - FFFF FFFF – четвертый.

Пример

Допустим, у нас есть адрес:



Первым делом процессор извлечет индекс таблицы каталогов из двух старших бит (в нашем случае это "1", т.е. второй элемент) и выделит адрес каталога из соответствующего элемента. Затем преобразование осуществляется как обычно: следующие 9 бит определяют индекс PDE в каталоге (в нашем случае "4"), полученный PDE содержит базу таблицы страниц, следующие 9 бит – индекс PTE (в нашем случае "1"). Из этого элемента PTE будет извлечен физический адрес страницы. Допустим, это будет 1 0000 0000. Теперь процессор складывает полученное значение с двенадцатью младшими битами линейного адреса. В результате мы получим физический адрес 1 0000 000C, который и будет подан на шину адреса. Чтобы сделать процесс преобразования более наглядным, приведем алгоритм преобразования линейного адреса в физический на псевдо-C:

```
qword get_phys_addr(dword addr)
{
  dword PDT_addr,         /* Физический адрес таблицы каталогов */
      PDT_element_index, /* Индекс элемента таблицы каталогов */
      PDT_element_addr,  /* Физический адрес элемента таблицы каталогов */
```

```
    PDE_index,     /* Индекс элемента каталога таблиц */
    PTE_index;     /* Индекс элемента таблицы страниц */
qword PD_addr,   /* Физический адрес каталога таблиц */
    PDE_addr,      /* Физический адрес элемента каталога таблиц */
    PT_addr,       /* Физический адрес таблицы страниц */
    PTE_addr,      /* Физический адрес элемента таблицы страниц */
    phys_addr;     /* Преобразованный физический адрес */

PDT_addr = CR3
PDT_addr &= F F F F F F E 0/* Нас интересуют только старшие 27 бит CR3 */
PDT_element_index = addr & C 0 0 0 0 0 0 0 /* Выделим из адреса индекс элемента таблицы каталогов */
PDT_element_addr = PDT_addr + PDT_element_index * 0x40 /* Получим физический адрес элемента таблицы каталогов */
PD_addr = *PDT_element_addr /* Из элемента таблицы каталогов получим адрес каталога */
PD_addr &= F F F F F F 0 0 0 /* Нас интересуют лишь старшие 24 бита физического адреса каталога */

PDE_index = addr & 11 1111 1110 0000 0000 0000 0000 0000 /* Выделим из адреса индекс элемента каталога */
PDE_addr = PD_addr + PDE_index * 0x8; /* Получим физический адрес элемента каталога */
PT_addr = *PDE_addr /* Из элемента каталога получим адрес таблицы страниц */
PT_addr &= F F F F F F 0 0 0 /* Нас интересуют лишь старшие 24 бита физического адреса таблицы страниц */

PTE_index = addr & 1 F F 0 0 0 /* Выделим из адреса индекс элемента таблицы страниц */
PTE_addr = PT_addr + PTE_index * 0x8; /* Получим физический адрес элемента таблицы страниц */
phys_addr = *PTE_addr /* Из элемента каталога таблиц получим физический адрес страницы */
phys_addr &= F F F F F F 0 0 0 /* Нас интересуют лишь старшие 24 бита физического адреса */
```

phys_addr += addr & 1111 1111 1111 /* Добавим к физического адресу страницы смещение в странице, и физический адрес готов */
return phys_addr
}

PAE предлагает еще одну опцию преобразования адреса: с увеличенным размером страниц. Что будет, если избавиться от таблицы страниц, оставив только таблицу каталогов и сами каталоги, а значение физического адреса брать прямо из элемента каталога? В этом случае у нас останется "свободным" 21 нижний бит линейного адреса, что приведет к созданию страниц размером 2 мегабайта. При этом старшие 11 бит линейного адреса в процессе преобразования будут заменены на 15 старших бит физического, взятых из соответствующего элемента каталога (рис 7.16). Младшие 21 бит линейного адреса при этом являются смещением в странице.

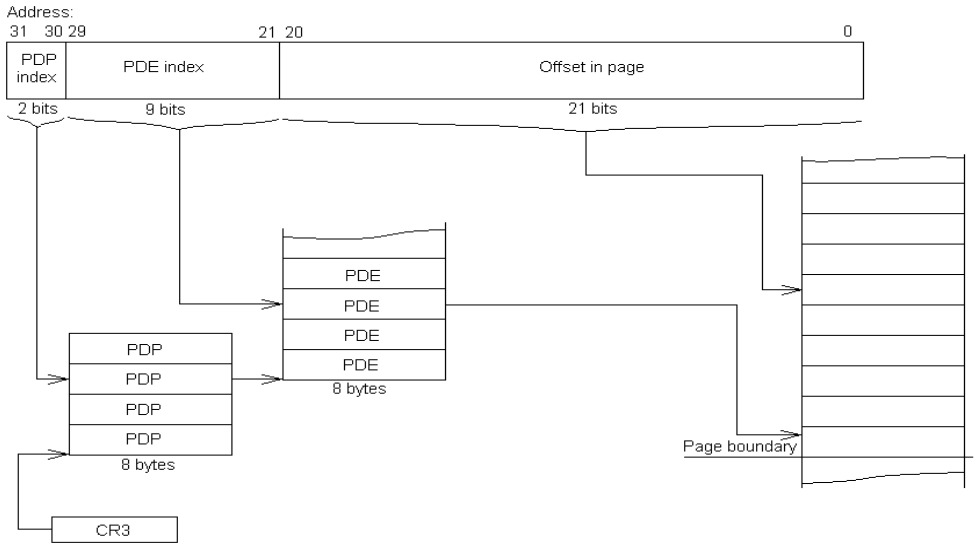


Рис. 7.16. Интерпретация бит линейного адреса с размером страниц 2 мегабайта

Формат элемента каталога в этом случае будет следующим (рис. 7.17).

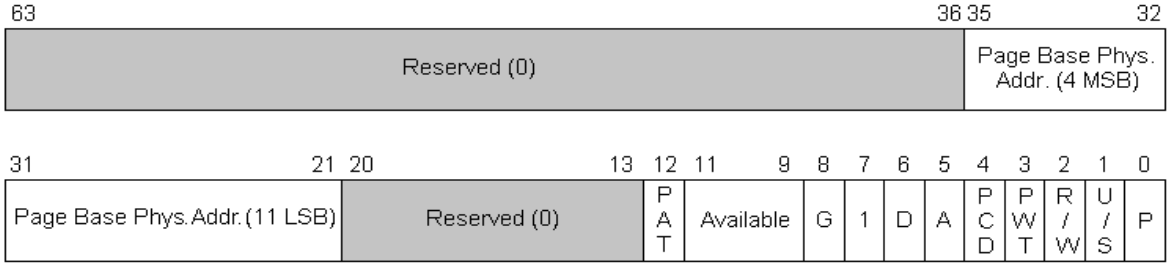


Рис. 7.17. Элемент каталога таблиц при размере страниц 2 мегабайта

Выбор размера страницы осуществляется флагом PS элемента каталога. Значение "0" означает, что каталог ссылается на таблицу страниц, размер страницы при этом остается 4 килобайта, "1" означает, что размер страницы, подчиненной элементу каталога, равен двум мегабайтам, и физический адрес следует брать прямо из элемента каталога. Таким образом возможно смешивать четырехкилобайтные и двухмегабайтные страницы в одном каталоге. Размещение данных в страницах разного размера имеет два преимущества.

1. Большие объемы данных лучше держать в страницах размером 2 мегабайта, так как это потенциально позволит уменьшить количество исключений неприсутствия страницы при работе с данными.

2. Преобразованные линейные адреса четырехкилобайтных и двухмегабайтных страниц хранятся в разных TLB-кэшах, что позволяет каждому TLB-кэшу содержать больше преобразованных адресов одновременно.

Таким образом PAE позволяет адресовать до шестидесяти четырех гигабайт физического адресного пространства, отображая 32-битные адреса на 36-битное физическое адресное пространство. Это происходит путем замены старших двадцати бит линейного адреса на двадцать четыре бита физического адреса, позволяя странице, таблице страниц, каталогу таблиц, таблице каталогов, либо всему 32-битному пространству отображаться в любую область 36-битного адресного пространства. При этом возможен выбор размера страниц между четырьмя килобайтами и двумя мегабайтами.

Контрольные вопросы

1. Какую функцию выполняет флаг PS?
2. Для чего служат регистры-индексы?
3. В чем заключаются различия между прямой и косвенной адресацией?
4. Какой тип адресации является самым эффективным?

Раздел 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

Глава 8

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

Актуальность цикла лабораторных работ по дисциплине «Вычислительные машины, системы и сети» обусловливается тем, что информация в настоящее время выходит на первый план среди прочих ресурсов объектов экономики различных форм собственности и назначения. Это диктуется необходимостью экономить трудовые, материальные и финансовые ресурсы. В настоящее время информационные процессы являются активными силами взаимосвязи внутри экономических объектов хозяйствования и между ними. Такие процессы в основном строятся на использовании разнообразных технологических решений и информационных систем.

Порядок выполнения и защиты лабораторных работ

Лабораторные работы выполняются в компьютерном классе РГТЭУ на макетах, стендах и компьютерах. При проведении лабораторных работ необходимо следовать правилам техники безопасности и внутренним инструкциям. Ниже описаны основные этапы проведения лабораторной работы: выполнение, оформление отчета, защита.

1. Выполнение лабораторной работы

Порядок выполнения:

- изучается инструкция по выполнению работы;
- уясняются цель работы и последовательность действий;
- уточняются у преподавателя непонятные моменты;
- подготавливаются необходимые таблицы;
- выполняются действия согласно пунктам инструкции по выполнению работы;
- основные действия и выводы конспектируются и затем заносятся в отчет.

2. Оформление отчета

Отчет представляется в электронном и текстовом виде, оформляется индивидуально каждым студентом, разрешается в виде исключения вручную аккуратным почерком, используя выделения подчеркиванием и цветом.

Отчет по каждой работе должен содержать:

1. Титульный лист.
2. Оглавление.
3. Упорядоченное изложение хода выполнения работы, выводы и данные по пунктам, заполненные таблицы.
4. Список литературы.

3. Защита лабораторной работы

Для защиты лабораторной работы студент должен:

- представлять цель и порядок выполнения работы;
- изучить практический и теоретический материал согласно вопросам к защите;
- ответить на вопросы к защите и дополнительные вопросы по данной теме.

Защищенную лабораторную работу подписывает преподаватель с указанием даты защиты работы.

Выполненные в полном объеме лабораторные работы являются допуском к зачету (экзамену). Студенты, не защитившие всех лабораторных работ, к зачету не допускаются.

Лабораторная работа № 1. *МАТЕРИНСКАЯ ПЛАТА ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА*

1. Цель работы

Изучить строение материнской платы, составить краткое описание ее составляющих.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

Материнская плата – сложная многослойная печатная плата, на которой устанавливаются основные компоненты персонального компьютера либо сервера начального уровня (центральный процессор, контроллер ОЗУ и собственно ОЗУ, загрузочное ПЗУ, контроллеры

базовых интерфейсов ввода-вывода). Именно материнская плата объединяет и координирует работу таких различных по своей сути и функциональности комплектующих, как процессор, оперативная память, платы расширения и всевозможные накопители. Это основа, которая не только служит для соединения отдельных комплектующих, но и определяет все важнейшие характеристики, связанные с производительностью (рис. 8.1).

Рис. 8.1. Материнская плата ASUS-P5WD2

Чипсет (chipset) – это микросхема, являющаяся связующим звеном между всеми компонентами материнской платы. От параметров чипсета во многом зависят технические и эксплуатационные характеристики компьютера: производительность, устойчивость работы, типы подключаемых процессоров и модулей памяти.

Чипсет, фактически являющийся набором системной логики, выступает в роли менеджера: в его задачи входит согласование работы процессора и остальных устройств путем обслуживания сигналов, приходящих от них по специальным устройствам, которые называются шинами. Основной связкой, влияющей на производительность, является цепочка «процессор – чипсет – оперативная память». Скорость работы всей системы зависит от самого медленного устройства. Например, если частота системной шины 133 МГц, а частота шины-

чипсет 66 МГц, то становится ясно, что каждый второй такт процессор просто простаивает, а при условии всевозможных задержек как со стороны памяти, так и со стороны процессора эта величина становится еще более значимой.

Поскольку основная функция материнской платы – «наводить мосты» между устройствами, то неудивительно, что главные составляющие любого чипсета также называют «мостами». Каждый из двух имеющихся в любом чипсете «мостов» – это специальный чип – микросхема. У каждого из двух «мостов» существует свой четко очерченный круг задач: «северный» мост соединяет между собой процессор, оперативную память и видеошину AGP, т.е. фактически он отвечает за всю «начинку» ПК. В свою очередь второй, «южный», мост отвечает за работу с шиной PCI и всеми подключенными к компьютеру периферийными устройствами.

«Гнездо» для установки 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.

Слоты для подключения питания бывают двух типов: AT и ATX. Все зависит от класса материнской платы. По размеру они одинаковы, но конструктивные различия не дают их перепутать между собой. В основном сейчас используются ATX-питание для материнских плат.

Разъемы-слоты стандартов PCI, PCI-E предназначены для подключения 32-разрядных устройств. Как правило, их четыре. Разъемы PCI – обычно самые короткие на плате, белого цвета, разделенные своеобразной «перемычкой» на две неравные части. Устройствами, устанавливаемыми в PCI–слот, могут быть: видеокарта, звуковые карты типа PCI, внутренние модемы, платы контроллера (SCSI), платы подключения USB составляющих, сетевые карты.

Разъемы-слоты типа ISA предназначены для подключения 16-разрядных устройств. Гораздо более слабые в отношении пропускной способности, чем слоты PCI, слоты' ISA, по сути дела, — наследство,

оставшееся еще со времен компьютеров типа 386. Но именно эти слоты, а не их более быстрые собратья PCI, являются наиболее дефицитными. Именно к ним до сих пор подключается громадное количество дополнительных карт: звуковые платы, внутренние модемы, сетевые карты, специализированные платы сканеров и т.д. Их на плате бывает от 2 до 4 под Socet 7, и от 1 до 2 под Slot 1. По внешнему виду слоты ISA напоминают слоты PCI, только они почти в полтора раза длиннее и цвет их не белый, а черный. В наше время почти прекращен выпуск устройств под ISA слоты.

Слоты для установки оперативной памяти от слотов для установки плат отличаются наличием специальных замочков-«защелок». На платах Pentium MMX, как правило, предусмотрена установка двух типов памяти – более старого формата SIMM (72 контакта) и быстрой памяти типа DIMM (168 контактов). Слоты DIMM значительно короче слотов SIMM. Количество слотов этих типов может варьироваться от 2 до 4. Возможен вариант, когда на материнской плате вы найдете разъемы под один тип памяти. В конце XX в. выпуск памяти SIMM полностью прекращен, что позволило перейти на более быструю и в данный момент дешевую память DIMM. Для компьютеров типа Pentium IV используют новый тип разъемов RIMM.

Контроллеры портов. Под «портами» понимаются разъемы на задней стенке компьютера, предназначенные для подключения таких внешних устройств, как принтер, мобильный дисковод большой емкости (для этого предусмотрен так называемый параллельный порт, или LPT), а также внешнего модема, манипулятора типа «мышь» (через два последовательных порта – COM1 и COM2 с разъемами 9 и 25 штырьков). Параллельный порт (LPT) всегда один, число же портов COM может варьироваться от 2 до 4. Во многих платах Pentium II и выше, соответствующих стандарту ATX, имеются еще и специальные разъемы для подключения мыши и клавиатуры – круглые разъемы типа PS/2. Последовательный порт USB позволяет подключить к компьютеру множество внешних устройств, соединенных в своеобразную цепочку (первое устройство включается в компьютер, второе подключается к первому и т.д.). Устройств в формате USB с каждым днем становиться все больше и больше: модемы, принтеры, сканеры, манипуляторы „мышь” и т.д.

Интерфейс IDE, или ATA (AT Attachment – подключаемый к AT) – простой и недорогой интерфейс для PC AT. Все функции по управлению накопителем обеспечивает встроенный контроллер, а 40-проводной соединительный кабель является фактически упрощенным сегментом 16-разрядной магистрали AT-Bus (ISA). Простейший адаптер IDE содержит только адресный дешифратор, все остальные сигналы заводятся прямо на разъем ISA. Адаптеры IDE обычно не содержат собственного BIOS – все функции поддержки IDE встроены в системный BIOS PC AT. Однако интеллектуальные и кэширующие контроллеры могут иметь собственный BIOS, подменяющий часть или все функции системного. Основной режим работы устройств IDE – программный обмен (PIO) под управлением центрального процессора, однако все современные винчестеры EIDE поддерживают обмен в режиме DMA, а большинство контроллеров – режим Bus Mastering.

Модификации IDE-интерфейса. На данный момент их насчитывается четыре:

Обычный IDE, или ATA.

В ***ATA-2*** были введены дополнительные сигналы (IORDY, CSEL и т.п.), режимы PIO 3-4 и DMA, команды остановки двигателя. Был также расширен формат информационного блока, запрашиваемого из устройства по команде Identify.

В ***ATA-3*** увеличена надежность работы в скоростных режимах (PIO 4 и DMA 2), введена технология S.M.A.R.T. (Self Monitoring Analysis And Report Technology – технология самостоятельного следящего анализа и отчета), позволяющая устройствам сообщать о своих неисправностях.

Стандарт ***Ultra ATA*** (называемый также ATA-33 и Ultra DMA-33) предложен фирмами Intel и Quantim. В нем повышена скорость передачи данных (до 33 Мб/с), предусмотрено стробирование передаваемых данных со стороны передатчика (в прежних ATA стробирование всегда выполнялось контроллером) для устранения проблем с задержками сигналов, а также введена возможность контроля передаваемых данных (метод CRC).

Все четыре разновидности имеют одинаковую физическую реализацию – 40-контактный разъем, но поддерживают разные режимы работы, наборы команд и скорости обмена по шине. Все интерфейсы совместимы снизу вверх (например, винчестер ATA-2 может работать с контроллером ATA, но не все режимы контроллера ATA-2 возможны для винчестера ATA).

Стандарт SATA (Serial ATA) — последовательный интерфейс обмена данными с накопителями информации. SATA является развитием параллельного интерфейса ATA (IDE), который после появления SATA был переименован в PATA (Parallel ATA). SATA-устройства используют два разъёма: 7-контактный (подключение шины данных) и 15-контактный (подключение питания). Стандарт SATA предусматривает возможность использовать вместо 15-контактного разъёма питания стандартный 4-контактный разъём Molex.

4. Ход лабораторной работы

1. Изучить теоретический материал, записав основные моменты лабораторной работы.
2. Произвести описание элементов материнской платы компьютера в табличной форме.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Обзор элементов материнской платы персонального компьютера.
4. Выводы по работе.

6. Контрольные вопросы

1. Что такое материнская плата, каковы ее назначение, состав, и фирмы-производители?
2. Кабели, разъемы, их назначение, описание.
3. Характеристики чипсетов. Фирмы-производители.

7. Литература

1. *Леонтьев, В.П.* Новейшая энциклопедия персонального компьютера 2007 / В.П. Леонтьев. – М. : Олма Медиа Групп, 2007. – 734 с. (Новейшая энциклопедия) – ISBN 978-5-373-00483-1.
2. Новейший чипсет от NVidia // 3DNEWS.RU : Ежедневный обзор новейших компьютерных технологий. 2011. URL: http://www.3dnews.ru/reviews/storage/abouthdd (дата обращения: 22.05.2012).
3. Новая история x86 // iXBT.com : Сайт о высоких технологиях. 2011. URL: http://www.ixbt.com/cpu/x86-cpus.shtml (дата обращения: 15.03.2012).

Лабораторная работа № 2. *СБОРКА ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА*

1. Цель работы

Научиться собирать персональный компьютер из комплектующих**.**

2. Приборы и материалы

Универсальная отвертка со сменными насадками, пинцет радиомонтажный, пассатижи, кусачки, комплектующие ПК (корпус со встроенным блоком питания, переходник с 20- контактного разъема блока питания на 24- контактный разъем материнской платы, материнская плата, графическая карта, процессор, кулер, теплопроводная паста, модули оперативной памяти, DVD – ROM, жесткий диск, Floppy – дисковод), ОС Windows XP SP3.

3. Краткие теоретические сведения

Существуют две формы системного блока – горизонтальная и вертикальная.

Вертикальную форму называют minitower, что в переводе означает "башня". В зависимости от дизайна может располагаться рядом с монитором, в столе или под ним.

Горизонтальную форму системного блока называют desktop. На таком системном блоке как правило сверху располагается монитор. Такая форма используется обычно в офисе.

На передней панели системного блока можно наблюдать две кнопки:

- кнопка Power**.** Ее нажимают, включая, выключая компьютер;
- кнопка Reset используется для экстренной перезагрузки компьютера. Ей приходится пользоваться в случае, когда компьютер „зависает” и когда никакие другие способы не помогают привести компьютер "в чувство".

Также на передней стороне можно найти следующее:

- индикаторы – лампочки, показывающие некоторые параметры работы ПК. Один из них показывает, работает компьютер или нет, т.е. горит в тот момент, когда компьютер включен. Другой индикатор по-

казывает состояние жесткого диска: он постоянно зажигается и гаснет в тот момент, когда происходит чтение или запись на жесткий диск.

- дисководы – это устройства, работающие со сменными информационными носителями. Там можно найти небольшой дисковод, который служит для работы с дисками емкостью 1.44 Мб. Также там, как правило сверху, расположены дисковод CD-ROM /RW или DVD-ROM/RW, предназначенные для чтения или записи информации на компакт-диски.
- разъемы. Их можно найти, как правило, как на передней, так и на задней панели системного блока, они предназначены для работы с внешними устройствами. Имеются такие разъемы, как USB, гнездо скоростного порта FireWare, а также можно найти круглое гнездо, куда подключаются наушники.

Основные компоненты системного блока:

- Корпус – весьма важный компонент, входящий в устройство компьютера. Бывает разных размеров и форм-факторов. К выбору корпуса системного блока следует подойти внимательно. В принципе, чем корпус больше и тяжелее, тем лучше: будет легче обеспечить хорошее охлаждение и низкий уровень шума. Рекомендуются корпуса только известных брэндов, например InWin, Thermaltake, Chieftec, Asus и др.

Блок питания – один из самых важных компонентов системного блока компьютера. Вы можете сэкономить на чем угодно, но только не на блоке питания. Как ни странно, но качество блока питания косвенно можно определить на вес – чем тяжелее, тем лучше. Возьмите в одну руку дешевый безымянный блок питания, а в другую дорогой брэндовый, и вы все поймете. Качественные радиаторы и трансформаторы достаточно тяжелые. Блок питания обеспечивает питание всех компонентов системного блока, и качество этого питания оказывает существенное влияние на „здоровье” всех комплектующих. Некачественный блок питания может являться причиной нестабильной работы компьютера и даже выгорания дорогостоящих комплектующих. Брэндовые корпуса обычно комплектуются достаточно качественными блоками питания. При выборе блока питания также необходимо обращать внимание на его мощность, например для офисного компь-

ютера достаточно будет 300 Вт, а для игрового может и 500 Вт не хватить. Микропроцессор – это главное вычислительное устройство компьютера, именно он выполняет команды, из последовательности которых состоят программы. От быстродействия процессора во многом зависит производительность компьютера. Быстродействие процессора определяется частотой, на которой он работает, количеством ядер и архитектурой. Сейчас на рынке присутствуют два основных брэнда: Intel и AMD. Выбор процессора определяется задачами, для решения которых покупается компьютер. Топовые модели обычно нужны для игр, видеообработки и подобных задач. (ofnet.ru)

- Корпусной вентилятор. Он необязателен, но желателен для улучшения охлаждения.
- Модули оперативной памяти. Оперативная память (ОЗУ – оперативное запоминающее устройство, RAM) – это быстродействующая память компьютера. Именно с этой памятью напрямую работает процессор. После выключения компьютера хранимая в ней информация стирается. С учетом „прожорливости" современных программ правило такое: чем больше оперативной памяти, тем лучше. На данный момент оптимальным объемом оперативной памяти, пожалуй, будет 2 - 4 Гбайта.
- Видеокарта (видеоадаптер, видеоплата, videocard, videoadapter) занимается обработкой и выводом графической информации на монитор. В видеокарте имеется свой специализированный графический процессор, который занимается обработкой 2D/3D графической информации. Это позволяет снизить вычислительную нагрузку на центральный процессор (CPU). Для офисных приложений подойдет практически любая видеокарта (даже встроенная в материнскую плату), а вот для игрушек придется раскошелиться. Выбирать игровую видеокарту следует предварительно определившись с набором игр, в которые хотелось бы поиграть. Выбирая типовую видеокарту убедитесь, что мощности вашего блока питания будет достаточно.
- Модем
- Сетевая карта. Через сетевую карту компьютер подключается к локальной или глобальной сети (Интернет). В настоящее время сетевые платы, как правило, интегрируются в материнские платы.

• CD- или DVD- накопитель (CD/DVD-ROM). Бывают как пишущие, так и не пишущие. Могут отличаться скоростью чтения и записи.

• Жесткий диск (накопитель на жестких магнитных дисках, harddisk, HDD) – это устройство долговременной памяти, данные при выключении питания не стираются, скорость работы намного ниже, чем у оперативной памяти, а емкость намного выше. Установленные программы, документы, музыка и фильмы хранятся именно на жестком диске. Его емкость измеряется в гигабайтах: чем больше, тем лучше, хотя для большинства офисных применений достаточно 40 – 80 гигабайт.

• Материнская плата – основной компонент системного блока, так как она объединяет все перечисленные устройства, а также содержит дополнительные компоненты: сетевой адаптер, видеокарту, звуковую карту, устройства ввода-вывода и пр.

4. Ход лабораторной работы

4.1. Установка процессора на материнскую плату

1. Организовать достаточно просторное рабочее место, приняв меры против статического электричества особенно в зимнее время. Например, разместим на рабочем месте заземленный металлический лист любой толщины.
2. Достать из упаковки саму материнскую плату в антистатическом пакете и поролоновую прокладку, оставив все остальное в упаковке.
3. Достать осторожно материнскую плату из антистатического пакета, положив ее на него и проложив под ним поролоновую прокладку.
4. Приступить к установке процессора (рис. 8.2):

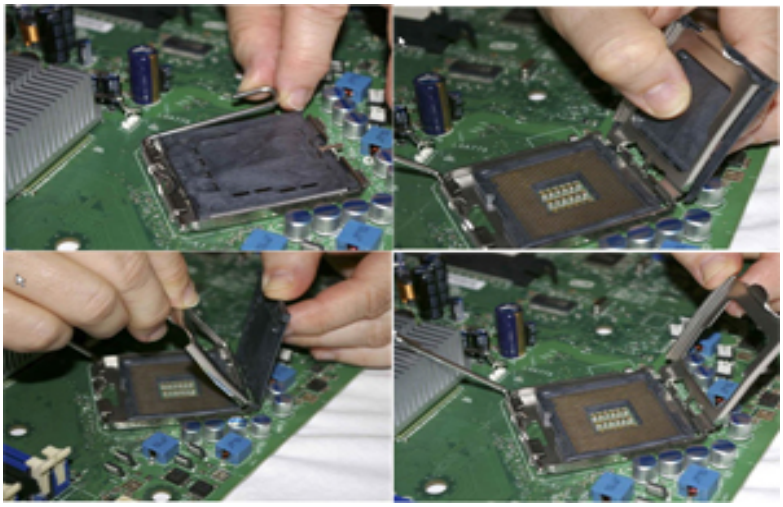


Рис. 8.2. Установка процессора

– отвести рычаг процессорного гнезда, надавливая на него и отводя в сторону от разъема;

– открыть пластину крепления. Не допускать касания руками контактов процессорного разъема;

– снять защитную крышку с пластины крепления. (Не выбрасывайте защитную крышку, устанавливайте ее на разъем, из которого удален процессор!);

– открыть примерно на 120° пластину крепления без защитной крышки;

– взять процессор и снять с него защитную крышку, придерживая его только за края и не допуская касания руками контактов;

– взять процессор большим и указательным пальцами, которые должны находиться у выемок процессорного разъема. Совместить выемки с выступами на процессорном разъеме. Опустить процессор вертикально вниз, избегая перекоса или его проскальзывания в разъеме;

– закрыть пластину крепления;

– нажать на пластину крепления до закрытия и фиксации рычага процессорного гнезда.

4.2. Установка модулей памяти:

1. Отвести боковые защелки на слотах модулей памяти, совместить выемки на модулях с выступами на слотах (рис. 8.3).

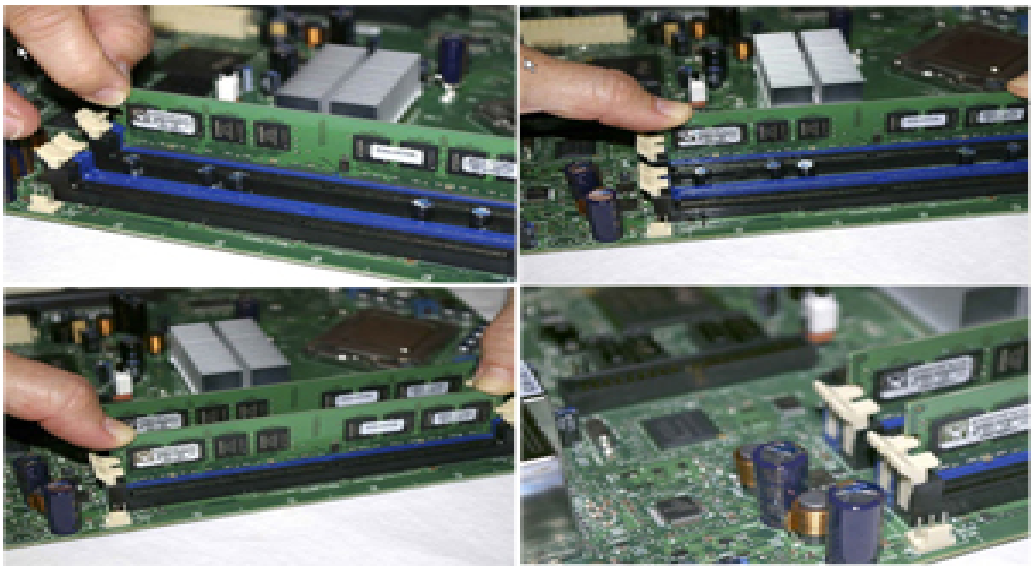


Рис. 8.3. Установка модулей памяти

2. Вставить вертикально модули и надавить сверху на края модуля до закрытия защелки.

3. Повторить предыдущие операции для второго канала (для двухканального варианта).

4. Проверить, закрылись ли защелки на всех установленных модулях памяти.

4.3. Установка на процессор теплоотвода с вентилятором

1. Протереть крышку процессора от накопившейся пыли.

2. Выдавить из тюбика на центр крышки процессора небольшое количество пасты (рис. 8.4).

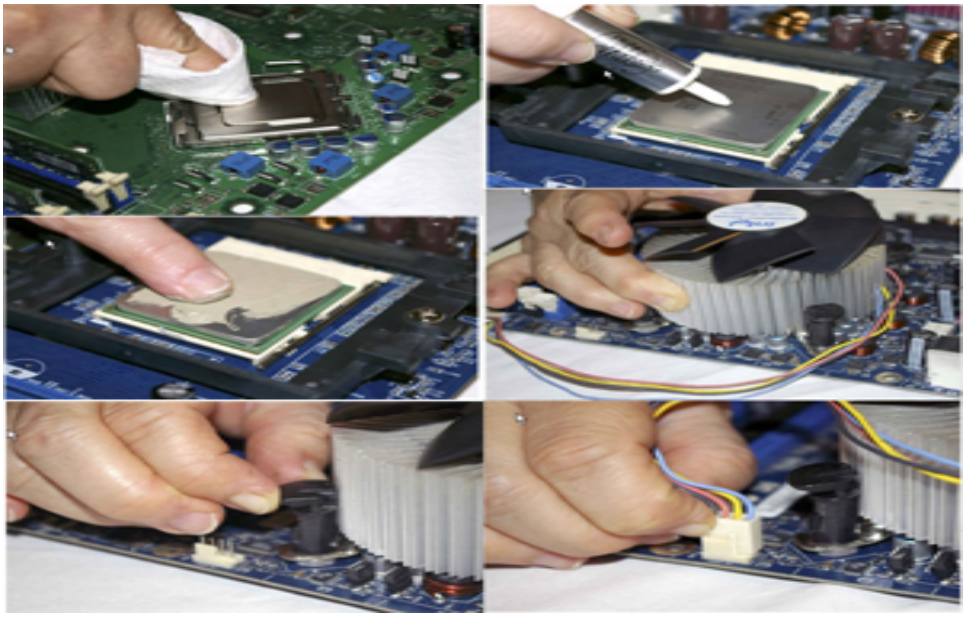


Рис.8.4. Установка кулера

3. Нанести пасту тонким слоем по поверхности крышки процессора от ее центра к краям**,** чтобы сквозь слой просматривалась крышка процессора. Паста создает тепловой барьер, и чем он тоньше, тем эффективнее теплопередача. Основное назначение нанесения пасты – улучшение теплоотвода путем сглаживания микронеровностей от механической обработки крышки процессора и теплоотвода.

4. Снять аккуратно с нижней поверхности теплоотвода фирменную теплопроводную пленку.

5. Установить теплоотвод с вентилятором таким образом, чтобы его крепежные элементы совместились с отверстиями в материнской плате.

6. Нажать на головку каждого крепежного элемента (по диагонали), придерживая теплоотвод. При правильном нажатии должен быть слышен щелчок.

7. Прокрутить отверткой в соответствии с инструкцией головку каждого крепежного элемента для его закрепления.

8. Подсоединить разъем кабеля от вентилятора к соответствующему разъему на материнской плате.

4.4. Установка материнской платы в корпус

1. Ввинтить в корпус в места, совпадающие с крепежными отверстиями материнской платы, стойки для ее установки (рис.8.5).

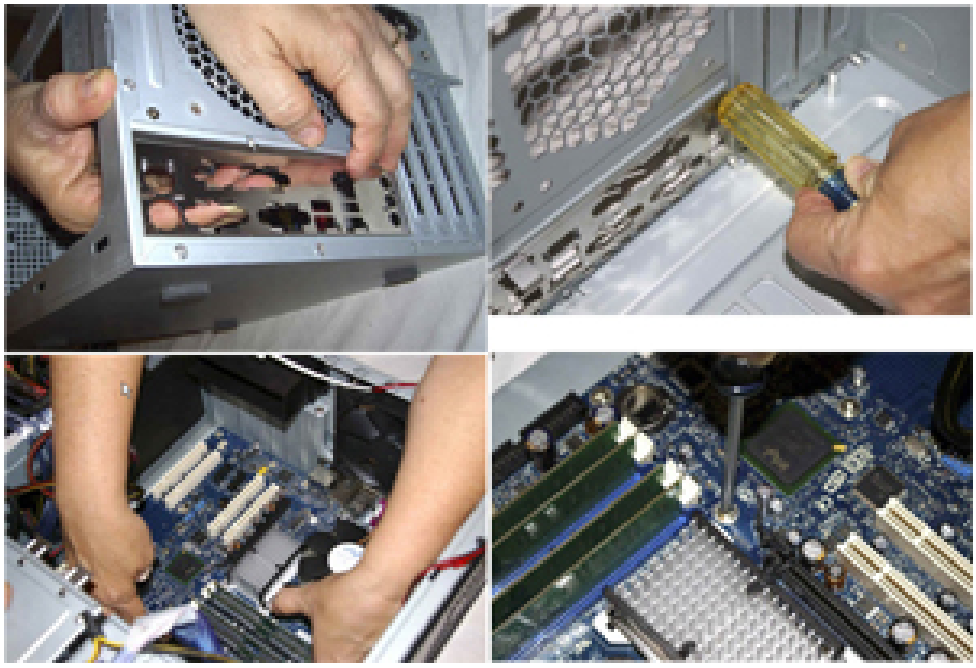


Рис.8.5. Установка материнской платы

2. Снять старую (поставляемую с корпусом) панель и установить новую для входных и выходных разъемов материнской платы с тыльной стороны корпуса.

3. Устанавить материнскую плату таким образом, чтобы крепежные отверстия в ней совпали с резьбовыми отверстиями стоек и разъемы прошли через новую панель.

4. Закрепить винтами материнскую плату.

4.5. Установка DVD-ROM/DVD-Writer

1. Открыть заглушку отсека для DVD-ROM/DVD-Writer и при необходимости снять металлическую перегородку корпуса, препятствующую установке устройства.

2. Установить перемычки сзади устройства в положение **MASTER** для **DVD-Writer** и **SLAVE** – для **DVD-ROM**.

3. Подсоединить кабель данных (**IDE**) к устройству.

4. Вставить устройство в корпус и закрепить его винтами.

5. Подсоединить кабель питания устройства.

4.6. Подсоединение основных кабелей

1. Подсоединить главный кабель питания.
2. Подсоединить дополнительный (**12V**) кабель питания.
3. Подсоединить ответный разъем **IDE** кабеля.
4. Подсоединить последовательно в соответствии с инструкцией на материнскую плату кабели схемы управления системным блоком.

4.7. Установка графической карты

1. Отвинтить (зависит от конструкции корпуса) заглушки для окон плат расширения (рис.8.6).

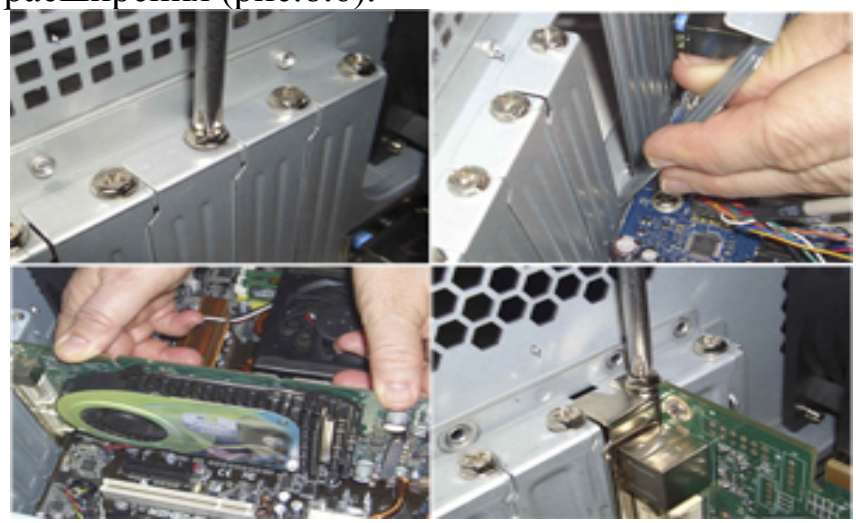


Рис. 8.6. Установка графической платы

2. Освободить от заглушки окно для графической карты.
3. Установить в слот графическую карту.
4. Закрепить винтом (зависит от конструкции корпуса) графическую карту.

4.8. Установка жесткого диска (HDD, винчестера)

1. Вставить один или несколько **HDD**. в специальные отсеки и закрепить.
2. Подсоединить к **HDD** кабель данных.
3. Подсоединить к **HDD** кабель питания.
4. Подсоединить ответный разъем кабеля данных к соответствующему разъему на материнской плате.

4.9. Установка других плат расширения, дополнительных разъемов, подключение кабелей и вентилятора корпуса, закрепление кабелей

1. Установить в слоты необходимые платы расширения (рис. 8.7).

Рис. 8.7. Установка плат расширения

2. Достать из комплекта материнской платы планки с дополнительными разъемами **USB** и **FW**, закрепить их в соответствующих окнах.

3. Подключить кабели дополнительных разъемов, а также звуковой (audio) кабель к соответствующим разъемам на материнской плате.

4. Подключить питание корпусного вентилятора.

5. Закрепить все кабели.

4.10. Установка операционной системы

1. Войти в соответствии с инструкцией на материнскую плату в **BIOS.**

2. Проверить, определила ли **BIOS** установленные **HDD**.

3. Выбрать в **BIOS** загрузку с **DVD-ROM/DVD-Writer.**

4. Вставить в привод загрузочный диск Windows и следовать инструкциям на мониторе.

5. Выбирать после полной установки Windows через соответствующую опцию **BIOS** загрузку с **HDD**.

6. Вставить в привод диск из комплекта материнской платы и установить необходимые драйверы и утилиты.

7. Вставить в привод диск из комплекта графической карты и установить необходимые драйверы.

8. Вставить в привод диск с последними версиями BIOS, утилит, драйверов и выполнить обновления, следуя инструкциям на материнскую плату и графическую карту.

9. Создать **1-й образ** диска, как правило, **C:** , например, программой **Acronis True Image Home v. 10.0.**

10. Обновить Windows [3].

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Описание процесса сборки персонального компьютера.
4. Выводы по работе.

6. Контрольные вопросы

1. Для чего необходимо устанавливать перемычки «MASTER» и «SLAVE»?
2. Для чего необходима термопаста?
3. Что такое форм-фактор?
4. Для чего необходима установка драйверов?
5. Каково функциональное назначение BIOS?

7. Литература

1. Как самому собрать или модернизировать компьютер? // WWW.JAROSLAFF.NET: Информационный пользовательский портал. 2007. URL: http://www.jaroslaff.net/modules.php?name=Articles&pa=showarticle&articles_id=918 (дата обращения: 22.04.2012).

2. *Ваулина, Е. Ю.* Мой компьютер : толковый слов. : свыше 3000 слов и устойчивых словосочетаний русского языка / Е. Ю. Ваулина. – М. : Эксмо, 2005. – 493 с. (Библиотека словарей). – ISBN 5-699-04753-0.

3. Устройство компьютера // WWW.USTROISTVO-PK.RU: Информационный пользовательский портал. 2010. URL: http://www.ustroistvo-pk.ru/index.php?id=sistemnik (дата обращения: 22.04.2012).

Лабораторная работа № 3. *ОПТИМИЗАЦИЯ И ВЫБОР НАСТРОЕК BIOS*

1. Цель работы

Изучить основные настройки конфигурации BIOS и основные разделы BIOS. Определить основы оптимизации BIOS. Настроить BIOS для оптимальной работы с ПК.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

BIOS (Basic Input Output System, базовая система ввода-вывода) – это набор базовых программ для проверки оборудования во время запуска, загрузки операционной системы, а также поддержки обмена данными между устройствами. Базовая система ввода-вывода хранится в постоянном запоминающем устройстве, благодаря чему ее программы могут быть выполнены при включении компьютера. Программы базовой системы ввода-вывода определяют общую производительность компьютера и в большинстве случаев остаются недоступными для пользователей.

Фактически BIOS – это небольшая микросхема, в которой записан набор микрокоманд, контролирующих работу устройств на материнской плате. В самом простом случае при запуске (включении) компьютера процессор выдает в BIOS сигнал, который инициирует загрузку низкоуровневой микропрограммы BOOT – ROUTINE. С этого момента и начинается собственно загрузка компьютера: сначала запускается программа самотестирования POST (Power – On Self Test), которая проверяет работоспособность процессора, оперативной памяти, а также вспомогательных и периферийных элементов. Далее программа BOOT – ROUTINE производит поиск и инициализацию других BIOS, которые могут быть установлены на других платах. Например, свою BIOS имеют почти все современные видеокарты. После проверки оборудования BIOS распределяет системные ресурсы между найденными устройствами. Им назначаются номера запросов

на прерывания IRQ, каналы прямого доступа к памяти DMA, адреса портов ввода-вывода I/O и т.д. Результат этой работы BIOS отображается в загрузочной таблице, где перечислены все основные найденные устройства и назначенные им ресурсы.

Загрузочная таблица «проскакивает» достаточно быстро – если не следить специально, то момент ее появления практически незаметен. Тем не менее особенно при сбоях в работе оборудования эта таблица может быть источником информации, очень полезной для локализации области поиска возникающих ошибок. Так, если устройство опознано в BIOS, но не работает при загруженной операционной системе, то, скорее всего, это проблема правильного подбора драйверов. Если же устройство реально существует, но не «прописано» в загрузочной таблице – значит, проблемы возникают на аппаратном уровне.

После распределения ресурсов BIOS запускает микропрограмму BOOTSTRAP LOADER, которая является начальным загрузчиком. Эта программа в заданном порядке перебирает устройства, которые могут содержать носители с загрузочной записью, и при нахождении таковых инициирует исполнение соответствующей команды, приводящей к началу загрузки соответствующей операционной системы.

Показанный выше алгоритм работы BIOS является наиболее стандартным, но существует множество дополнений и изменений, связанных с работой каждой конкретной BIOS.

Сейчас можно выделить двух производителей BIOS: компании Phoenix и American Megatrends. Первая выпускает BIOS под марками AWARD и PHOENIX, а вторая – AMIBIOS.

Разные марки BIOS имеют естественные различия, несмотря на одинаковые задачи, которые стоят перед ними. В первую очередь это связано с возможностью ручной настройки и, как следствие, предоставляемыми командами. Наиболее «слабым» в этом плане является BIOS PHOENIX – она обладает минимальными настроечными функциями, доступ к которым весьма затруднен. Как правило, эта BIOS используется в многопроцессорных системах, рабочих станциях и на некоторых материнских платах, выпускаемых компанией Intel.

Сложно сказать, какая марка BIOS – AWARD или AMIBIOS – представлена в нашей стране шире. Например, марка AMIBIOS очень распространена в старых компьютерах, хотя и часть новых, собранных на не очень дорогих материнских платах, также использует BIOS

этой марки. BIOS одной марки могут различаться версиями. С ростом возможностей, закладываемых в материнскую плату, растет потребность в увеличении возможностей управления ими. Кроме того, как и любое программное обеспечение, микропрограммы для BIOS постоянно обновляются, оптимизируя и расширяя исходно заложенные в него параметры.

Попасть в окно настроек BIOS, которое также называется BIOS Setup, можно только в самом начале загрузки компьютера в процессе его самотестирования до того момента, как появится загрузочная таблица и будет вызвана программа загрузки операционной системы из boot-сектора носителя. Для входа в окно настроек в большинстве BIOS используется клавиша Delete, однако в некоторых случаях помогают и другие клавиши, например F1, F2 и т.д. Название нужной клавиши пишется в сообщении «Press Del to Enter Setup», или «Hit F2 to Enter Setup», возникающем сразу после тестирования памяти. Иногда сообщение может выглядеть так: «Del: Setup».

Окно настройки BIOS, как правило, представляет собой меню со списком разделов, в которых находятся команды управления параметрами. Некоторые BIOS поддерживают работу с мышью и имеют вид набора папок, в которых хранятся команды.

Пример

STANDARD CMOS SETUP
Стандартные предустановки CMOS

Date (mn/date/year) - для изменения даты в системных часах.

Time (hour/min/sec) - для изменения времени в системных часах.

Hard disk C: (Жесткий Диск C:) - номер вашего первичного (главного) жесткого диска.

Cyln - число цилиндров на вашем жестком диске.

Head - число головок. Wpcom - предкомпенсация при записи.

Lzone - адрес зоны парковки головок.

Sect - число секторов на дорожку.

Size - объем диска. Автоматически вычисляется согласно числу цилиндров, головок и секторов. Выражается в мегабайтах.

Floppy drive A (дисковод для дискет A) - устанавливается тип дисковода для дискет, который будет использоваться в качестве привода A.

Floppy drive B (тип дисковода B) - аналогично предыдущему.

Primary display (Первичный дисплей) - Тип стандарта отображения, который вы используете.

Keyboard (Клавиатура): Installed-установлена. Если изменить на "not installed", эта опция укажет BIOS на отмену проверки клавиатуры во время стартового теста, что позволяет перезапускать PC с отключенной клавиатурой (файл-серверы и т.п.) без выдачи сообщения об ошибке теста клавиатуры.

ADVANCED CMOS SETUP
Дополнительные предустановки

Typematic Rate Programming - программирование скорости автоповтора нажатой клавиши. По умолчанию – Disabled. Следующие два пункта определяют, как программируется клавиатура:

- Typematic Rate Delay (msec) - задержка автоповтора, начальное значение: 500 мс. Начальная задержка перед стартом автоповтора символа, т.е., сколько времени вы должны удерживать клавишу нажатой, чтобы ее код начал повторяться.
- Typematic Rate (Chars/Sec) - частота автоповтора (символов в секунду). Начальное значение 15.
- Memory Test Tick Sound - щелчок при прохождении теста памяти. Рекомендуется устанавливать Enabled для того, чтобы слышать, что процесс загрузки выполняется нормально.
- Memory Parity Error Check - проверка ошибок четности памяти. Рекомендуется установить Enabled. Дополнительная возможность проверки бита ошибки в памяти. Все (или почти все) PC проверяют память во время работы. Каждый байт памяти имеет дополнительный девятый разряд, который при каждом обращении к ОЗУ по записи устанавливается таким образом, чтобы общее число единиц было нечетным. При каждом обращении по чтению проверяется признак нечетности. При обнаружении ошибки возникает немаскируемое прерывание NMI, которое вы не можете заблокировать. ЭВМ прекращает работу и на экране отображается сообщение об ошибке ОЗУ обычно в виде сообщения вида: PARITY ERROR AT 0AB5:00BE SYSTEM HALTED.

Wait for <F1> If Any Error-ждать нажатия F1 в случае любой ошибки. Когда при начальной загрузке обнаруживается ошибка, PC просит вас нажать F1- только в случае нефатальных ошибок. Если установлено в Disabled - система печатает предупреждение и продолжает загрузку без ожидания нажатия клавиши. Рекомендуется устанавливать Enabled.

System Boot Up Num Lock-включение дополнительной клавиатуры при загрузке в цифровой режим. Определяет, будет ли включен режим NumLock при начальной загрузке ЭВМ. Одним это нравится, другим - нет.

Floppy Drive Seek at Boot - поиск на флоппи-диске при загрузке. Рекомендуется устанавливать в Disabled для более быстрой загрузки и для уменьшения опасности повреждения головок.

System Boot Sequence - последовательность начальной загрузки системы: на каком дисководе вначале искать ОС. Для более быстрой загрузки рекомендуется C:,A: этот же метод пригоден и для того, чтобы посторонние не могли загрузить ваш компьютер с дискеты, если ваш autoexec.bat начинается с процедуры доступа к системе. Установка A:,C: нужна в том случае, если пользователь не знает, как ему сконфигурировать CMOS, иначе при какой-либо неудаче большинство пользователей не будут знать, что им делать, если невозможно загрузиться с дискеты. Однако следует быть внимательным - вам следовало бы знать, что эта установка включается и отключается, и быть готовым к этому. Если дорожка с начальным загрузчиком на вашем жестком диске будет повреждена (но не будет полностью отсутствовать), вы сможете загрузиться с дискеты. Аналогично, легко обмануться, считая, что вы загружаетесь с дискеты, заведомо чистой от вирусов, в то время как на самом деле загрузка происходит с инфицированного жесткого диска.

External Cache Memory - внешняя кэш-память. Устанавливается Enabled, если имеется кэш-память. Одна из наиболее часто встречающихся ошибок при работе с CMOS SETUP - если при наличии кэш-памяти вы блокируете ее. Производительность системы при этом значительно падает. Это - кэш между CPU и системной шиной. При установке Enabled и отсутствии реально установленной кэш-памяти система будет "заморожена" большую часть времени.

Password Checking Option - опция проверки пароля. Установка

пароля на доступ к системе или к меню SETUP. Рекомендуется в тех случаях, когда ЭВМ используется совместно несколькими пользователями и вы не хотите, чтобы кто-то (друзья, сестра и т.д.) изменяли установки BIOS.

BootSector Virus Protection - защита сектора загрузки от вирусов. В действительности это не совсем защита от вирусов. Все, что эта функция делает, - всякий раз, когда к сектору начальной загрузки обращаются по записи, выдает предупреждение на экран и позволяет вам либо разрешить запись, либо запретить ее.

AUTO CONFIGURATION WITH BIOS DEFAULTS
Автоконфигурация со значениями BIOS по умолчанию

Значения BIOS по умолчанию - те, которые установлены в качестве начальных для вашей системной платы и CHIPSET'а. Дают приемлемую возможность прохождения стартового теста. Как правило, являются неплохими начальными значениями перед точной настройкой вашей системы. Если вы допустили какую-либо ошибку и не знаете, какую именно, выберите этот пункт. Опция заменит ваши установки в BIOS на исходные, и вы сможете начать все сначала. От вас требуется точное знание конфигурации вашей системы. Эта опция *НЕ МЕНЯЕТ* ни системную дату, ни конфигурацию жесткого диска и флоппи-дисководов в стандартном CMOS SETUP, поэтому вы можете ожидать, что в большинстве случаев ваша система загрузится без проблем после выбора данной опции.

Защита информации в наше время играет огромную роль. Однако при помощи BIOS защитить компьютер можно только от несанкционированного доступа не очень искушенных пользователей. Вообще защита в BIOS – это, скорее, пережиток прошлого, чем реальная необходимость. Хотя, с другой стороны, защита на уровне BIOS может поставить некоторый барьер от неосознанного изменения внутренних настроек BIOS любопытными детьми и тем самым уберечь ваш компьютер от «самопроизвольного» разгона.

В арсенале BIOS имеется несколько возможностей защиты: защита паролем загрузки компьютера, защита паролем изменения настроек BIOS, защита загрузочного сектора жесткого диска, блокировки записи на дискету и блокировка чтения-записи жесткого диска.

В случае повреждения микросхемы CMOS RAM (или разряде батареи или аккумулятора) программа Setup имеет возможность воспользоваться некой информацией по умолчанию (BIOS Setup Default Values), которая хранится в таблице соответствующей микросхемы ROM BIOS. Кстати, на некоторых материнских платах питание микросхемы CMOS RAM может осуществляться как от внутреннего, так и от внешнего источника. Выбор определяется установкой соответствующей перемычки.

4. Ход лабораторной работы

1. Изучить теоретический материал, записав основные моменты лабораторной работы.
2. Записать версию BIOS. Выписать настройки BIOS на вашем ПК.

Таблица 1

Исследование памяти ПК

Base Memory:	
Extended Memory	
Other Memory:	
Total Memory:	

3. Найти и подключить в BIOS HDD посредством пункта IDE HDD Auto Detection. Записать его установки в таблицу.

Таблица 2

Установки HDD в BIOS

Число цилиндров (Cylinder)	
Число головок (Head)	
Число секторов (Sector)	
Число байт на сектор (Bytes per Sector)	
Объем дискового пространства	

4. Подобрать настройки вашего винчестера с помощью раздела Standard CMOS Setup на оптимальные величины, изменяя при этом вручную настройки табл. 2.

5. Настроить правильную работу ОЗУ.
6. Выполнить следующие действия:

– включить защиту от инфицирования вирусами;
– включить быстрое самотестирование при включении;
– установить различные последовательности загрузки (A, C; C, A);
– проверить пункт „Переименование дисководов гибких дисков”;
– включить и проверить пункт „Поиск дисковода при загрузке”;
– установить скорость ввода 30: 30 знак/с.
– выполнить пункт „Установить меры безопасности на вход в систему и на вход в BIOS”, проверить перезагрузкой компьютера;
– отключить копирование видеоBIOS в динамическое ОЗУ;
– установить все таймеры на максимум в режиме энергосбережения;
– выключить режим пониженного энергопотребления;
– в разделе PNP/PCI Configuration Setup сконфигурировать настройки шины PCI;
– установить ручной контроль за конфигурацией, выставить на IRQ3, 5,11, DMA1, 3,5 – Legacy ISA. Остальные установки PCI/ISA PnP;
– выставить определение PIO - режима Primary Master-диска; Mode 0,1,2,3,4 — для непосредственного определения PIO - режима. Проверить работоспособность;
– установить пароль супервизора/пользователя. Изменить пароль, удалить пароль, войти в BIOS и систему под установленным паролем супервизора и пользователя;
– сохранить изменения BIOS.

7. Сделать вывод о работе BIOS данного ПК. Записать.

5. Содержание отчета:

1. Титульный лист.
2. Цель работы.
3. Описание и назначение BIOS персонального компьютера.
4. Результаты настройки BIOS.
5. Выводы по работе.

6. Контрольные вопросы

1. Назначение BIOS.
2. Раздел Standard CMOS Setup BIOS. Основные настройки раздела.
3. Раздел BIOS Features Setup.
4. Раздел Chipset Features Setup BIOS.
5. Раздел Power Management Setup BIOS.
6. Раздел PNP/PCI Configuration Setup BIOS.
7. Раздел Integrated Peripherals BIOS.
8. Пароль супервизора/пользователя BIOS. Изменение, удаление, вход под паролем.

7. Литература

1. *Мюллер, С.* Модернизация и ремонт ПК = Upgrading PCs : пер. с англ. / С. Мюллер.– 17-е изд. – М. : Вильямс, 2007. – 1489 с. С 1470-1489. – ISBN 978-5-8459-1126-1.
2. *Леонтьев, В.П.* Новейшая энциклопедия персонального компьютера 2007 / В.П. Леонтьев. – М. : Олма Медиа Групп, 2007. – (Новейшая энциклопедия). – ISBN 978-5-373-00483-1.
3. *Крымов, Б.* Профессиональная диагностика компьютера : учеб. пособие / Б. Крымов. – М. : Триумф, 2006. – 268 с. – (Инструментальная книга). – ISBN 5-89392-142-9.
4. *Арсеньев, П.* BIOS и тонкая настройка ПК. Начали! / П. Арсеньев. – СПб. : Питер, 2009. – 158 с. – (Начали!). – ISBN 978-5-388-00202-0.

Лабораторная работа № 4. *РАСПРЕДЕЛЕНИЕ ПАМЯТИ И ПРЕРЫВАНИЙ ЭВМ*

1. **Цель работы**

Изучить структуру распределения памяти ЭВМ и процесс распределения прерываний, составить карту запроса на прерывание.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

Распределение системных ресурсов – один из важнейших моментов при загрузке компьютера. От того как система распределит преры-

вания и доступ к динамической памяти, сконфигурирует взаимодействие устройств (карт расширения) с шинами и так далее, может в значительной мере зависеть как производительность всей системы в целом, так и работоспособность отдельных устройств (рис. 8.8).

Рис. 8.8. Устройства ввода/вывода

При установке устройства Plug and Play Windows автоматически настраивает его, обеспечивая его правильную работу с другими установленными на компьютере устройствами. В ходе процесса настройки Windows назначает устанавливаемому устройству уникальный набор системных ресурсов. Эти ресурсы могут включать один или несколько следующих параметров:

- Номера строк запроса на прерывание (IRQ).
- Каналы прямого доступа к памяти (DMA).
- Адреса портов ввода/вывода (I/O).
- Диапазоны адресов памяти.

Адрес памяти – часть памяти компьютера, которая может быть выделена устройству или использоваться программой или операционной системой. Устройству обычно выделяется диапазон адресов.

Каждый ресурс, назначаемый устройству, должен быть уникальным. Это необходимо для правильной работы устройства. Для

устройств Plug and Play Windows автоматически проверяет правильность настройки ресурсов (рис. 8.9).

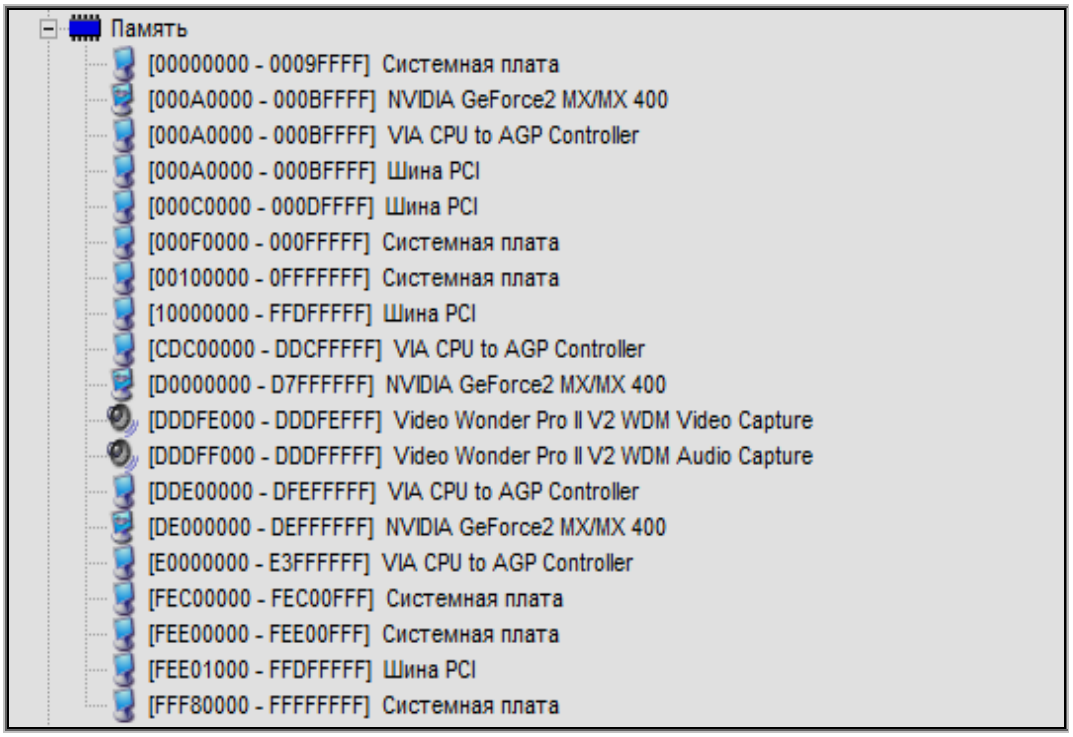


Рис. 8.9. Ресурсы ПК

Прерывание IRQ (Interrupt Request) – это запрос устройства на обработку данных процессором. При получении прерывания процессор приостанавливает свои операции, сохраняет текущее состояние и передает управление специальной программе (обработчику прерывания), содержащей команды для обработки ситуации, вызвавшей это прерывание. Каждому устройству как периферийному, так и внутреннему система присваивает определенное прерывание, используя которое устройство информирует процессор о необходимости обработки запроса от данного устройства.

Некоторые прерывания жестко закреплены за определенными устройствами, в то время как другие могут перераспределяться в зависимости от текущих требований. Всего в системе используется 16 прерываний: от IRQ0 до IRQ15. При этом четыре прерывания – IRQ0, IRQ1, IRQ8 и IRQ13 – зарезервированы системой (для системного таймера, клавиатуры, часов и математического сопроцессора) и не участвуют в распределении между остальными устройствами системы и картами расширения (рис. 8.10).

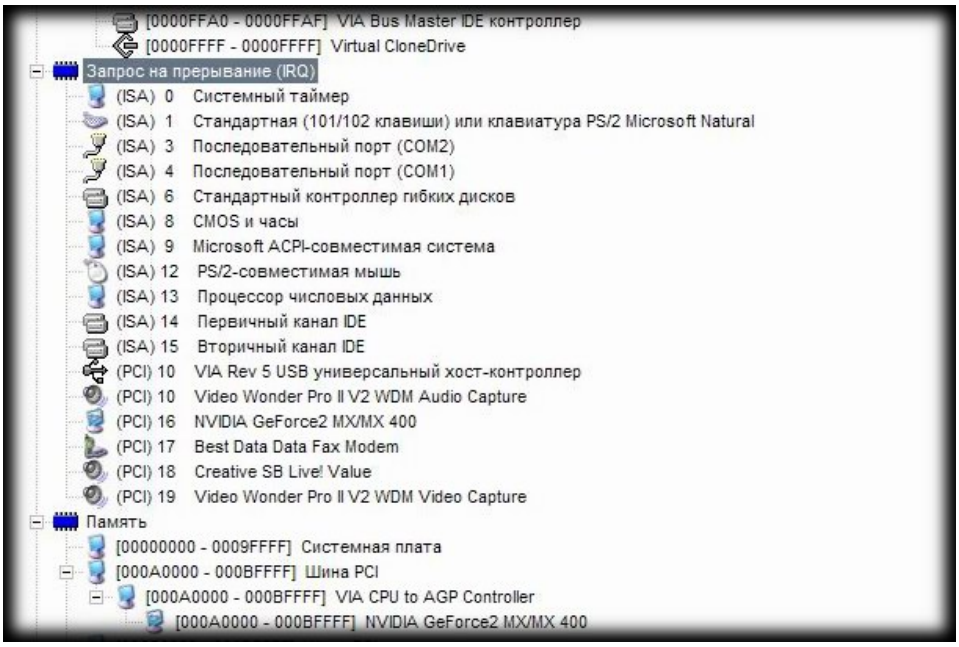


Рис. 8.10. Карта прерываний

Как происходит распределение прерываний? После включения компьютера при его тестировании выполняется присваивание прерываний системным устройствам, размещенным на материнской плате, затем ISA-устройствам и в конце – PCI-устройствам. Первыми распределяются прерывания для ISA-устройства, поскольку не все из таких устройств поддерживают технологию Plug and Play и могут требовать ручного (например, с помощью переключателей на материнской плате) указания номера прерывания. После распределения прерываний их конфигурация сохраняется в энергонезависимой CMOS-памяти. При последующих загрузках эта конфигурация автоматически вызывается из памяти, сравнивается с текущим состоянием системы и в этом случае, если не произошло никаких изменений, загружается. Если в системе произошли изменения, например замена или удаление карт расширения, процедура распределения прерываний выполняется заново.

Картам расширения, поддерживающим технологию Plug and Play, выделяются все оставшиеся прерывания, которые распределяются между устройствами автоматически. В тех случаях, когда карт расширения больше, чем свободных прерываний, на одном прерывании может оказаться несколько устройств. В некоторых операционных системах, например Windows 2000/XP, все прерывания перераспределяются еще раз после загрузки системы, и может сложиться си-

туация, когда без каких-либо видимых причин на одном прерывании одновременно окажутся несколько устройств, например звуковая плата, видеокарта и модем. В результате, если задействовать такие устройства параллельно, в работе системы могут происходить весьма ощутимые задержки, например в нашем случае – при работе модема (дозвоне) может «подвисать» звук. При этом если в Windows 98 можно было выполнить ручное конфигурирование и перераспределение прерываний, то в Widows 2000/XP по умолчанию этого сделать нельзя: данная операция становится доступной только после отключения менеджера энергосберегающего режима ACPI.

От правильного распределения прерываний может в значительной мере зависеть работоспособность компьютера. Поэтому при установке в слоты карт расширения необходимо помнить о следующем:

- Слоту AGP и первому слоту PCI присваивается один и тот же номер прерывания.
- Если PCI-слотов пять, то один номер прерывания разделяется между четвертым и пятым слотами.
- При установке сложного устройства, требующего сразу двух прерываний, следующий слот по возможности следует оставлять свободным.

В современных системах все большее развитие получают схемы, позволяющие подключать внешние периферийные устройства через USB-порты. Такие системы удобны, поскольку шина USB занимает одно прерывание и позволяет подключать в систему без дальнейшего расходования ресурсов все периферийные устройства, оборудованные соответствующими разъемами: мышь, клавиатуру, сканер, видеокамеру и т.д. Поэтому переход на USB-устройства кроме явного выигрыша в скорости обслуживания данного устройства (исключением, пожалуй, являются только высокоскоростные, имеющие IDE-контроллер: жесткие диски и CD – ROM-приводы), позволяет разгрузить системные требования, облегчив тем самым доступ к ним для других устройств.

Все параметры, позволяющие указать способ распределения аппаратных прерываний, находятся в разделе PNP/PCI Configuration BIOS.

Ручное выделение ресурсов необходимо в тех случаях, когда по-

сле автоматического распределения начинают возникать непонятные конфликты, не связанные с работоспособностью самого устройства. Самый простой пример – «подтормаживание» одного устройства при включении другого. Очень часто после перераспределения прерываний и (или) изменения местоположения устройства, т.е. изменения слота, система начинает работать нормально.

Нередко проблемы возникают при работе видеокарты – при выполнении рутинной офисной работы все нормально, а при загрузке игр, требующих использования 3D-ускорителя (при условии его наличия, правильности настройки драйверов и программного окружения), картинка начинает мерцать, изображение «ползет» и т.д.

Это может быть связано с тем, что видеокарта имеет общее прерывание с другими устройствами, и пока она работает в офисном режиме, система справляется с распределением запросов устройств. Но как только запускается игра, начинается обработка огромных массивов информации, которыми обменивается центральный процессор, системная память и видеопроцессор. В такой ситуации разделяемое использование прерывания ведет к снижению производительности всех участвующих в работе устройств.

Для выхода из создавшегося положения можно использовать параметр Assign IRQ For VGA (IRQ to PCI VGA, Allocated IRQ to PCI VGA). Этот параметр имеет два значения: Enabled (Yes) – разрешено, и Disabled (No) – запрещено. При разрешении система выделяет видеокарте отдельное прерывание, что положительно сказывается на производительности или при игре с программами, обрабатывающими видеопотоки. При этом нужно помнить, что, отдавая отдельное прерывание видеокарте, вы уменьшаете число прерываний для остальных PCI-устройств и соответственно ухудшаете их взаимодействие с системой. Все указывает на необходимость определить приоритеты при использовании компьютера: если первична офисная работа, то выделение отдельного прерывания нецелесообразно.

При работе со старыми PCI-видеокартами также можно использовать выделение индивидуального аппаратного прерывания, которое выполняется с помощью параметра Slot *n* IRQ for VGA, где *n* – это число имеющихся в компьютере PCI-слотов. Соответственно, тот слот, в котором находится видеокарта и которому необходимо выделить отдельное прерывание, должен получить значение Enable – разрешено, а все остальные – Disable – запрещено.

4. Ход лабораторной работы

1. Изучить теоретический материал, записав основные моменты лабораторной работы.

2. Составить карту запросов на прерывание и карту распределения памяти ЭВМ.

Для открытия карты прерываний необходимо щёлкнуть правой кнопкой мышки по ярлыку «Мой компьютер», открыть «Свойства», перейти во вкладку «Оборудование», нажать «Диспетчер устройств», в появившемся окне нажать вкладку «Вид» и открыть «Ресурсы по типу», затем необходимо раскрыть ветку «Запрос на прерывание IRQ».

Для открытия карты распределения памяти необходимо щёлкнуть правой кнопкой мышки по ярлыку «Мой компьютер», открыть «Свойства», перейти во вкладку «Оборудование», нажать «Диспетчер устройств», в появившемся окне нажать вкладку «Вид» и открыть «Ресурсы по типу», затем необходимо раскрыть ветку «Память».

Открытые карты прерываний и распределения памяти необходимо поместить в отчёт по лабораторной работе.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Описание и назначение прерываний и адресов памяти.
4. Карты прерываний и адресов памяти компьютера.
5. Выводы по работе.

6. Контрольные вопросы

1. Что такое прерывание?
2. Что такое адрес памяти?
3. Для чего необходимо знать распределение прерываний в ПК?
4. Что будет, если на одно прерывание установлено несколько устройств?
5. В каких операционных системах существует возможность ручного распределения прерываний?

7. Литература

1. *Мюллер, С.* Модернизация и ремонт ПК = Upgrading and Repairing PCs : пер. с англ. / С. Мюллер. – 17-е изд. – М. : Вильямс, 2007. – 1489 с. – ISBN 978-5-8459-1126-1.

2. *Ваулина, Е. Ю.* Мой компьютер : толковый слов. : свыше 3000 слов и устойчивых словосочетаний русского языка / Е.Ю. Ваулина. – М. : Эксмо, 2005. – 493 с. (Библиотека словарей). – ISBN 5-699-04753-0.

Лабораторная работа № 5. *ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ*

1. Цель работы

Изучить существующие способы оценки производительности вычислительных машин и получить базовые навыки сравнения их производительности.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

Тестовый пакет SiSoft Sandra Lite.

3. Краткие теоретические сведения

Основу для сравнения различных типов компьютеров между собой дают стандартные методики измерения производительности. В процессе развития вычислительной техники появилось несколько таких стандартных методик. Они позволяют разработчикам и пользователям осуществлять выбор между альтернативами на основе количественных показателей, что дает возможность постоянного прогресса в данной области.

Единицей измерения производительности компьютера является время: компьютер, выполняющий определённый объем работы за меньшее время, является более быстрым. Время выполнения любой программы измеряется в секундах. Часто производительность измеряется как скорость появления некоторого числа событий в секунду, так что меньшее время подразумевает большую производительность.

MIPS

Одной из альтернативных единиц измерения производительности процессора (по отношению к времени выполнения) является MIPS - (миллион целочисленных команд в секунду). Имеется несколько различных вариантов интерпретации определения MIPS.

В общем случае MIPS есть скорость операций с целыми числами в единицу времени, т.е. для любой данной программы MIPS есть просто отношение количества команд в программе к времени ее выполнения. Таким образом, производительность может быть определена как обратная к времени выполнения величина, причем более быстрые машины при этом будут иметь более высокий рейтинг MIPS.

Положительными сторонами MIPS является то, что эту характеристику легко понять, особенно покупателю, и что более быстрая машина характеризуется большим числом MIPS, что соответствует нашим интуитивным представлениям. Однако использование MIPS в качестве метрики для сравнения наталкивается на три проблемы. Во-первых, MIPS зависит от набора команд процессора, что затрудняет сравнение по MIPS компьютеров, имеющих разные системы команд. Во-вторых, MIPS даже на одном и том же компьютере меняется от программы к программе. В-третьих, MIPS может меняться по отношению к производительности в противоположенную сторону.

MFLOPS

Измерение производительности компьютеров при решении научно-технических задач, в которых существенно используется арифметика с плавающей точкой, всегда вызывало особый интерес. Именно для таких вычислений впервые встал вопрос об измерении производительности, а по достигнутым показателям часто делались выводы об общем уровне разработок компьютеров. Обычно для научно-технических задач производительность процессора оценивается в MFLOPS (миллионах чисел-результатов вычислений с плавающей точкой в секунду, или миллионах элементарных арифметических операций над числами с плавающей точкой, выполненных в секунду).

Как единица измерения, MFLOPS предназначена для оценки производительности только операций с плавающей точкой и поэтому не применима вне этой ограниченной области. Например, программы

компиляторов имеют рейтинг MFLOPS, близкий к нулю, вне зависимости от того, насколько быстра машина, поскольку компиляторы редко используют арифметику с плавающей точкой.

SPECint92, SPECfp92

Важность создания пакетов тестов, базирующихся на реальных прикладных программах широкого круга пользователей и обеспечивающих эффективную оценку производительности процессоров, была осознана большинством крупнейших производителей компьютерного оборудования, которые в 1988 году учредили бесприбыльную корпорацию SPEC (Standard Performance Evaluation Corporation). Основной целью этой организации является разработка и поддержка стандартизованного набора специально подобранных тестовых программ для оценки производительности новейших поколений высокопроизводительных компьютеров. Членом SPEC может стать любая организация, уплатившая вступительный взнос.

Основным результатом работы SPEC являются наборы тестов, которые разрабатываются SPEC с использованием кодов, поступающих из разных источников. SPEC работает над импортированием этих кодов на разные платформы, а также создает инструментальные средства для формирования из кодов, выбранных в качестве тестов, осмысленных рабочих нагрузок. Поэтому тесты SPEC отличаются от свободно распространяемых программ. Хотя они могут существовать под похожими или теми же самыми именами, время их выполнения в общем случае будет отличаться.

В настоящее время имеется два базовых набора тестов SPEC, ориентированных на интенсивные расчеты и измеряющих производительность процессора, системы памяти, а также эффективность генерации кода компилятором. Как правило, эти тесты ориентированы на операционную систему UNIX, но они также импортированы и на другие платформы. Процент времени, расходуемого на работу операционной системы и функции ввода/вывода, в общем случае ничтожно мал.

TPC-A, TPC-B, TPC-C

По мере расширения использования компьютеров при обработке транзакций в сфере бизнеса все более важной становится возмож-

ность справедливого сравнения систем между собой. С этой целью в 1988 году был создан Совет по оценке производительности обработки транзакций (TPC - Transaction Processing Performance Council), который представляет собой бесприбыльную организацию. Любая компания или организация может стать членом TPC после уплаты соответствующего взноса. На сегодня членами TPC являются практически все крупнейшие производители аппаратных платформ и программного обеспечения для автоматизации коммерческой деятельности. К настоящему времени TPC создал три тестовых пакета для обеспечения объективного сравнения различных систем обработки транзакций и планирует создать новые оценочные тесты.

Тесты TPC

TPC определяет и управляет форматом нескольких тестов для оценки производительности OLTP (On-Line Transaction Processing), включая тесты TPC-A, TPC-B и TPC-C. Как уже отмечалось, создание оценочного теста является ответственностью организации, выполняющей этот тест. TPC требует только, чтобы при создании оценочного теста выполнялись определенные условия. Хотя упомянутые тесты TPC не являются характерными тестами для оценки производительности баз данных, системы реляционных баз данных являются ключевыми компонентами любой системы обработки транзакций.

Следует отметить, что как и любой другой тест, ни один тест TPC не может измерить производительность системы, которая применима для всех возможных сред обработки транзакций, но эти тесты действительно могут помочь пользователю справедливо сравнивать похожие системы. Однако, когда пользователь делает покупку или планирует решение о покупке, он должен понимать, что никакой тест не может заменить его конкретную прикладную задачу.

Тест TPC-A

Выпущенный в ноябре 1989 года тест TCP-A предназначался для оценки производительности систем, работающих в среде интенсивно обновляемых баз данных, типичной для приложений интерактивной обработки данных (OLDP - on-line data processing). Такая среда характеризуется:

- множеством терминальных сессий в режиме on-line;

- значительным объемом ввода/вывода при работе с дисками;
- умеренным временем работы системы и приложений;
- целостностью транзакций.

Тест TPC-A определяет пропускную способность системы, измеряемую количеством транзакций в секунду (tps A), которые система может выполнить при работе с множеством терминалов. Хотя спецификация TPC-A не определяет точное количество терминалов, компании-поставщики систем должны увеличивать или уменьшать их количество в соответствии с нормой пропускной способности. Тест TPC-A может выполняться в локальных или региональных вычислительных сетях. В этом случае его результаты определяют либо "локальную" пропускную способность (TPC-A-local Throughput), либо "региональную" пропускную способность (TPC-A wide Throughput). Очевидно, эти два тестовых показателя нельзя непосредственно сравнивать. Спецификация теста TPC-A требует, чтобы все компании полностью раскрывали детали работы своего теста, свою конфигурацию системы и ее стоимость (с учетом пятилетнего срока обслуживания). Это позволяет определить нормализованную стоимость системы ($/tpsA).

Тест TPC-B

В августе 1990 года TPC одобрил TPC-B – интенсивный тест базы данных, характеризующийся следующими элементами:

- значительным объемом дискового ввода/вывода;
- умеренным временем работы системы и приложений;
- целостностью транзакций.

TPC-B измеряет пропускную способность системы в транзакциях в секунду (tpsB). Поскольку имеются существенные различия между двумя тестами –TPC-A и TPC-B – (в частности, в TPC-B не выполняется эмуляция терминалов и линий связи), их нельзя прямо сравнивать.

Тест TPC-C

Тестовый пакет TPC-C моделирует прикладную задачу обработки заказов, а также достаточно сложную систему OLTP, которая должна управлять приемом заказов, управлением учетом товаров и распространением товаров и услуг. Тест TPC-C осуществляет тести-

рование всех основных компонентов системы: терминалов, линий связи, ЦП, дискового ввода/вывода и базы данных.

TPC-C требует, чтобы выполнялись пять типов транзакций:

- новый заказ, вводимый с помощью сложной экранной формы;
- простое обновление базы данных, связанное с платежом;
- простое обновление базы данных, связанное с поставкой;
- справка о состоянии заказов;
- справка по учету товаров.

Обычно публикуются два результата. Один из них, tpm-C, представляет пиковую скорость выполнения транзакций (выражается в количестве транзакций в минуту). Второй результат, $/tpm-C, представляет собой нормализованную стоимость системы. Стоимость системы включает все аппаратные средства и программное обеспечение, используемые в тесте, плюс стоимость обслуживания в течение пяти лет.

AIM

Одной из независимых организаций, осуществляющей оценку производительности вычислительных систем, является частная компания AIM Technology, которая была основана в 1981 году. Компания разрабатывает и поставляет программное обеспечение для измерения производительности систем, а также оказывает услуги по тестированию систем конечным пользователям и поставщикам вычислительных систем и сетей, которые используют промышленные стандартные операционные системы.

Генератор тестовых пакетов представляет собой программную систему, которая обеспечивает одновременное выполнение множества программ. Он содержит большое число отдельных тестов, которые потребляют определенные ресурсы системы и тем самым акцентируют внимание на определенных компонентах, из которых складывается ее общая производительность. При каждом запуске генератора могут выполняться любые отдельные или все доступные тесты в любом порядке и при любом количестве проходов, позволяя тем самым создавать для системы практически любую необходимую рабочую нагрузку. Все это дает возможность тестовому пакету моделировать любой тип смеси при постоянной смене акцентов (для лучшего пред-

ставления реальной окружающей обстановки) и при обеспечении высокой степени конфигурирования.

Для оценки и сравнения систем в AIM Performance Report II используются следующие критерии:

1. Пиковая производительность (рейтинг производительности по AIM).

2. Максимальная пользовательская нагрузка.

3. Индекс производительности утилит.

4. Пропускная способность системы.

Рейтинг производительности по AIM - стандартная единица измерения пиковой производительности, установленная AIM Technology. Этот рейтинг определяет наивысший уровень производительности системы, который достигается при оптимальном использовании ЦП, операций с плавающей точкой и кэширования диска. Рейтинг вездесущей машины VAX 11/780 обычно составляет 1 AIM. В отчетах AIM представлен широкий ряд UNIX-систем, которые можно сравнивать по этому параметру.

Максимальная пользовательская нагрузка определяет “емкость” (capacity) системы, т.е. такую точку, начиная с которой производительность системы падает ниже приемлемого уровня для N-го пользователя (меньше чем одно задание в минуту на одного пользователя).

Индекс производительности утилит определяет количество пользовательских нагрузок пакета Milestone, которые данная система выполняет в течение одного часа. Набор тестов Milestone многократно выполняет выбранные утилиты UNIX в качестве основных и фоновых заданий при умеренных пользовательских нагрузках. Этот параметр показывает возможности системы по выполнению универсальных утилит UNIX.

Максимальная пропускная способность определяет пиковую производительность мультипрограммной системы, измеряемую количеством выполненных заданий в минуту. Приводящийся в отчете график пропускной способности системы показывает, как она работает при различных нагрузках.

4. Ход лабораторной работы

1. Изучить теоретический материал, записав основные положения работы.

2. Установить на ПК бесплатную версию пакета SiSoft Sandra Lite http://www.softportal.com/software-223-sisoftware-sandra-lite.html (рис. 8.11).

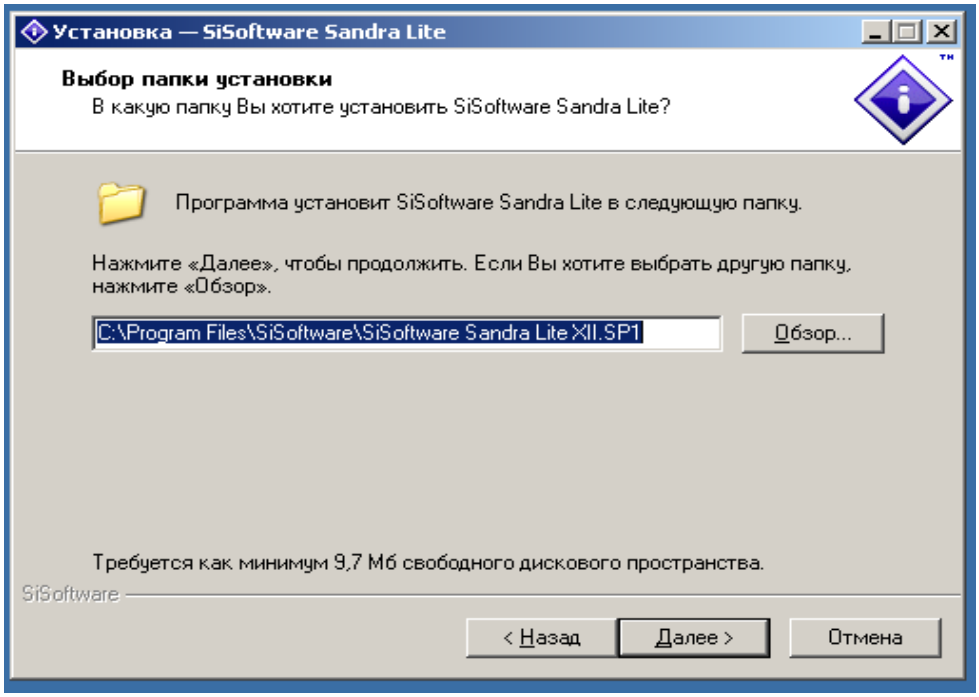


Рис. 8.11. Окно установки пакета SiSoft Sandra Lite

3. Настроить программный пакет в соответствии с требованиями операционной системы (рис. 8.12).

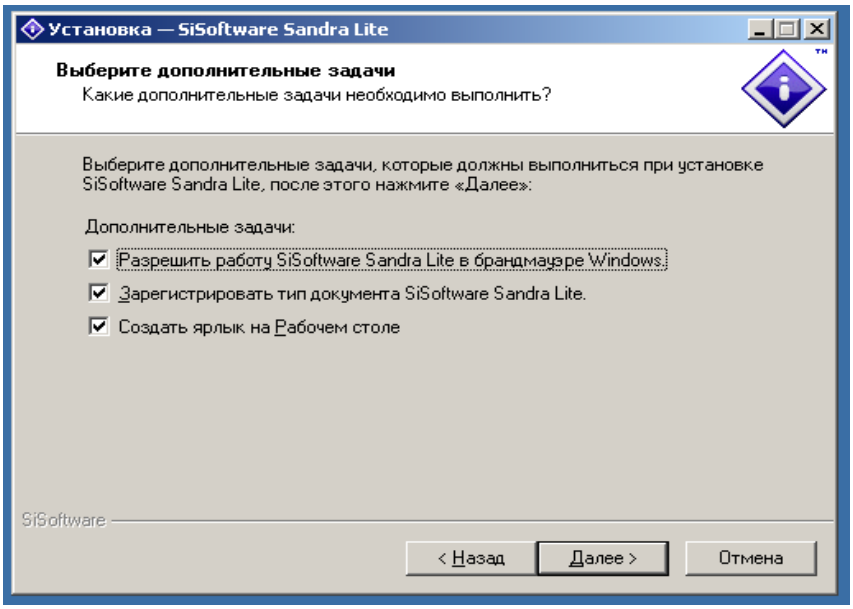


Рис. 8.12. Окно настройки дополнительных задач пакета SiSoft Sandra Lite

4. Получить сводную информацию о конфигурации Вашей системы (рис. 8.13).

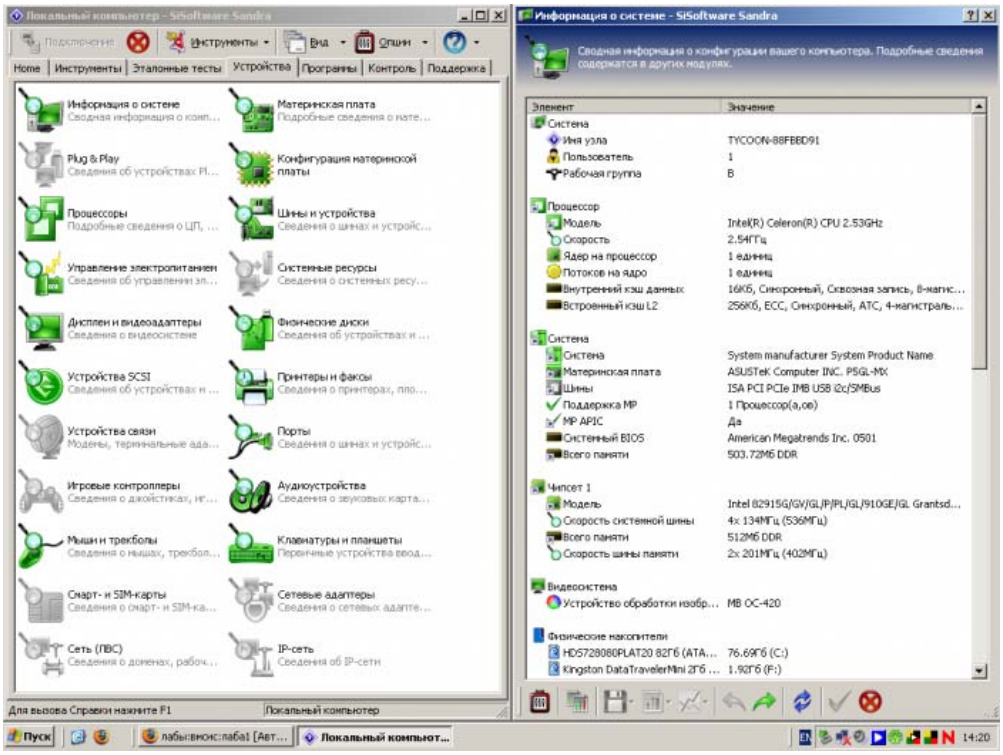


Рис. 8.13. Окно конфигурации системы

5. Провести тестирование производительности вычислительной системы по следующим пунктам:

- арифметический тест процессора;
- многоядерная эффективность;
- арифметика .net;
- файловые системы;
- съёмные диски;
- латентность памяти;
- пропускная способность сети;
- мультимедийный тест процессора;
- мультимедиа .net;
- физические диски;
- cd-rom и dvd;
- пропускная способность памяти;
- кэш и память;
- скорость интернет-соединения.

Получившиеся результаты необходимо представить в отчёте в качестве изображений с монитора ПК.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Обзор и анализ способов оценки производительности вычислительных машин.
4. Описание лабораторных устройств.
5. Результаты экспериментальных исследований.
6. Выводы по работе.

6. Контрольные вопросы

1. Назовите основные факторы, влияющие на производительность ВМ.
2. Какие существуют тесты для оценки производительности, в чем их различие?
3. Как связана тактовая частота микропроцессора и производительность ВМ?

7. Литература

1. MIPS // CITFORUM.RU Форум высоких технологий. 2011. URL: http://citforum.ru/hardware/app_kis/glava_7.shtml (дата обращения: 20.05.2012).
1. MIPS // CITFORUM.RU Форум высоких технологий. 2011. URL: http://citforum.ru/hardware/app_kis/glava_11.shtml (дата обращения: 20.05.2012).
2. MIPS // CITFORUM.RU Форум высоких технологий. 2011. URL: http://citforum.ru/hardware/app_kis/glava_8.shtml (дата обращения: 20.05.2012).

Лабораторная работа № 6. *СТРУКТУРА LAN*

1. Цель работы

Изучить структуру LAN.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

Под ЛВС понимают совместное подключение нескольких отдельных компьютерных рабочих мест (рабочих станций) к единому каналу передачи данных. Благодаря вычислительным сетям мы получили возможность одновременного использования программ и баз данных несколькими пользователями.

Понятие „локальная вычислительная сеть" – ЛВС – (англ. LAN - Lokal Area Network) относится к географически ограниченным (территориально или производственно) аппаратно-программным реализациям, в которых несколько компьютерных систем связанны друг с другом с помощью соответствующих средств коммуникаций. Благодаря такому соединению пользователь может взаимодействовать с другими рабочими станциями, подключенными к этой ЛВС.

В производственной практике ЛВС играют очень большую роль. Посредством ЛВС в систему объединяются персональные компьютеры, расположенные на многих удаленных рабочих местах, которые используют совместно оборудование, программные средства и информацию. Рабочие места сотрудников перестают быть изолированными и объединяются в единую систему. Рассмотрим преимущества, получаемые при сетевом объединении персональных компьютеров в виде внутрипроизводственной вычислительной сети.

Разделение ресурсов

Разделение ресурсов позволяет экономно использовать ресурсы, например, управлять периферийными устройствами, такими как лазерные печатающие устройства, со всех присоединенных рабочих станций.

Разделение данных

Разделение данных предоставляет возможность доступа и управления базами данных с периферийных рабочих мест, нуждающихся в информации.

Разделение программных средств

Разделение программных средств предоставляет возможность одновременного использования централизованных ранее установленных программных средств.

Разделение ресурсов процессора

При разделении ресурсов процессора возможно использование вычислительных мощностей для обработки данных другими системами, входящими в сеть. Предоставляемая возможность заключается в том, что на имеющиеся ресурсы не “набрасываются” моментально, а только лишь через специальный процессор, доступный каждой рабочей станции.

Многопользовательский режим

Многопользовательские свойства системы содействуют одновременному использованию централизованных прикладных программных средств, ранее установленных и управляемых, например, если пользователь системы работает с другим заданием, то текущая выполняемая работа отодвигается на задний план.

Все ЛВС работают в одном стандарте, принятом для компьютерных сетей, - в стандарте Open Systems Interconnection (OSI).

Топология типа „звезда”

Концепция топологии сети в виде звезды, в которой головная машина получает и обрабатывает все данные с периферийных устройств как активный узел обработки данных, пришла из области больших ЭВМ. Этот принцип применяется в системах передачи данных, например в электронной почте RELCOM. Вся информация между двумя периферийными рабочими местами проходит через центральный узел вычислительной сети (рис. 8.14).

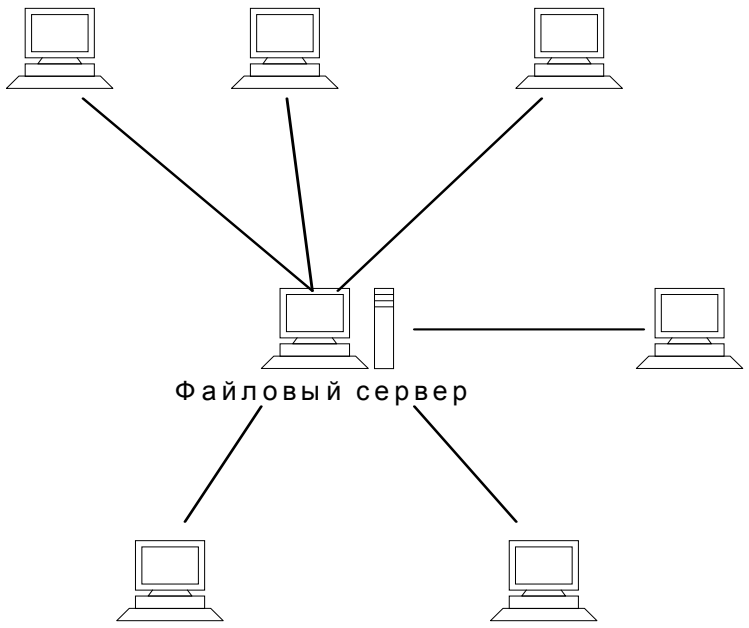


Рис. 8.14. Топология «звезда»

Пропускная способность сети определяется вычислительной мощностью узла и гарантируется для каждой рабочей станции. Коллизий (столкновений) данных не возникает.

Кабельное соединение довольно простое, так как каждая рабочая станция связана с узлом. Затраты на прокладку кабелей высокие, особенно когда центральный узел географически расположен не в центре топологии.

При расширении вычислительных сетей не могут быть использованы ранее выполненные кабельные связи: к новому рабочему месту необходимо прокладывать отдельный кабель из центра сети.

Топология в виде звезды является наиболее быстродействующей из всех топологий вычислительных сетей, поскольку передача данных между рабочими станциями проходит через центральный узел (при его хорошей производительности) по отдельным линиям, используемым только этими рабочими станциями. Частота запросов передачи информации от одной станции к другой невысокая по сравнению с достигаемой в других топологиях.

Производительность вычислительной сети в первую очередь зависит от мощности центрального файлового сервера. Он может быть узким местом вычислительной сети. В случае выхода из строя центрального узла нарушается работа всей сети.

Центральный узел управления - файловый сервер – может реализовать оптимальный механизм защиты против несанкционированного доступа к информации. Вся вычислительная сеть может управляться из ее центра.

Кольцевая топология

При кольцевой топологии сети рабочие станции связаны одна с другой по кругу, т.е. рабочая станция 1 с рабочей станцией 2, рабочая станция 3 – с рабочей станцией 4 и т.д. Последняя рабочая станция связана с первой. Коммуникационная связь замыкается в кольцо (рис. 8.15).

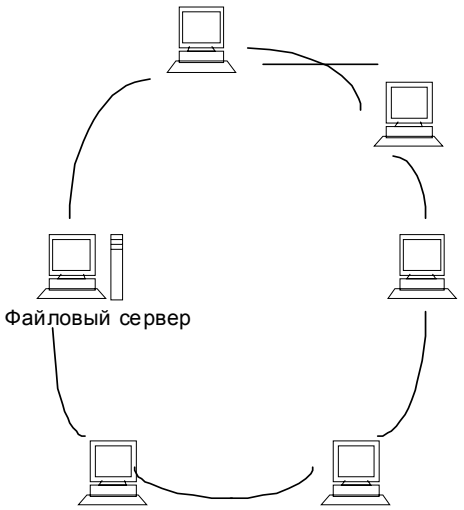


Рис. 8.15. Топология «кольцо»

Прокладка кабелей от одной рабочей станции до другой может быть довольно сложной и дорого-

стоящей, особенно если географически рабочие станции расположены далеко от кольца (например, в линию).

Сообщения циркулируют регулярно по кругу. Рабочая станция посылает по определенному конечному адресу информацию, предварительно получив из „кольца” запрос. Пересылка сообщений является очень эффективной, так как большинство сообщений можно отправлять “в дорогу” по кабельной системе одно за другим. Очень просто можно сделать кольцевой запрос на все станции. Продолжительность передачи информации увеличивается пропорционально количеству рабочих станций, входящих в вычислительную сеть.

Основная проблема при кольцевой топологии заключается в том, что каждая рабочая станция должна активно участвовать в пересылке информации, и в случае выхода из строя хотя бы одной из них вся сеть парализуется. Неисправности в кабельных соединениях локализуются легко.

Подключение новой рабочей станции требует краткосрочного выключения сети, так как во время установки кольцо должно быть разомкнуто. Ограничения на протяженность вычислительной сети не существует, так как оно в конечном счете определяется исключительно расстоянием между двумя рабочими станциями (рис. 8.16).

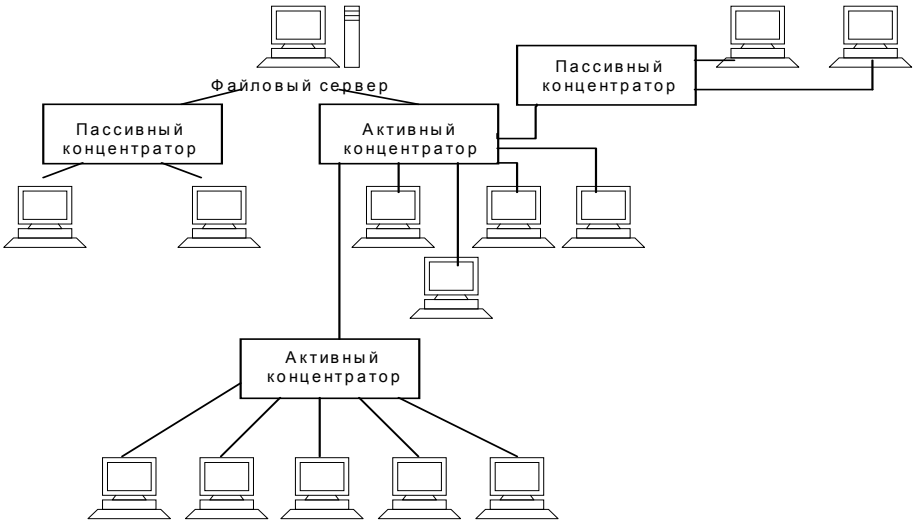


Рис. 8.16. Структура логической кольцевой цепи

Специальной формой кольцевой топологии является логическая кольцевая сеть. Физически она монтируется как соединение звездных топологий. Отдельные „звезды” включаются с помощью специальных

коммутаторов (англ. Hub -концентратор), которые по-русски также иногда называют “хаб”. В зависимости от числа рабочих станций и длины кабеля между рабочими станциями применяют активные или пассивные концентраторы. Активные концентраторы дополнительно содержат усилитель для подключения от 4 до 16 рабочих станций. Пассивный концентратор является исключительно разветвительным устройством (максимум на три рабочие станции). Управление отдельной рабочей станцией в логической кольцевой сети происходит так же, как и в обычной кольцевой сети. Каждой рабочей станции присваивается соответствующий ей адрес, по которому передается управление (от старшего к младшему и от самого младшего к самому старшему). Разрыв соединения происходит только для нижерасположенного (ближайшего) узла вычислительной сети, так что лишь в редких случаях может нарушаться работа всей сети.

Шинная топология

При шинной топологии среда передачи информации представляется в форме коммуникационного пути, доступного для всех рабочих станций, к которому они все должны быть подключены. Все рабочие станции могут непосредственно вступать в контакт с любой рабочей станцией, имеющейся в сети (рис. 8.17).

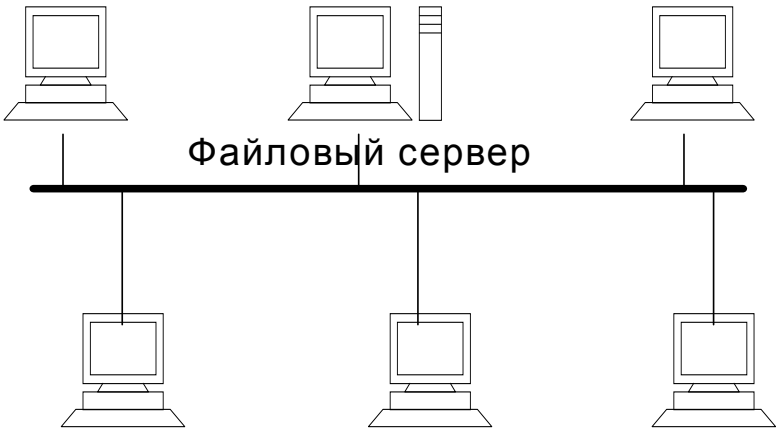


Рис. 8.17. Шинная топология

Рабочие станции в любое время без прерывания работы всей вычислительной сети могут быть подключены к ней или отключены. Функционирование вычислительной сети не зависит от состояния отдельной рабочей станции.

В стандартной ситуации для шинной сети Ethernet часто используют тонкий кабель или Cheapernet-кабель с тройниковым соединителем. Выключение и особенно подключение к такой сети требуют разрыва шины, что вызывает нарушение циркулирующего потока информации и зависание системы.

Новые технологии предлагают пассивные штепсельные коробки, через которые можно отключать и/или включать рабочие станции во время работы вычислительной сети.

Благодаря тому что рабочие станции можно включать без прерывания сетевых процессов и коммуникационной среды, очень легко прослушивать информацию, т.е. ответвлять информацию из коммуникационной среды.

В ЛВС с прямой (немодулируемой) передачей информации всегда может существовать только одна станция, передающая информацию. Для предотвращения коллизий в большинстве случаев применяется временной метод разделения, согласно которому для каждой подключенной рабочей станции в определенные моменты времени предоставляется исключительное право на использование канала передачи данных. Поэтому требования к пропускной способности вычислительной сети при повышенной нагрузке снижаются, например при вводе новых рабочих станций. Рабочие станции присоединяются к шине посредством устройств TAP (англ. Terminal Access Point - точка подключения терминала). TAP представляет собой специальный тип подсоединения к коаксиальному кабелю. Зонд игольчатой формы внедряется через наружную оболочку внешнего проводника и слой диэлектрика к внутреннему проводнику и присоединяется к нему.

В ЛВС с модулированной широкополосной передачей информации различные рабочие станции получают по мере надобности частоту, на которой эти рабочие станции могут отправлять и получать информацию. Пересылаемые данные модулируются на соответствующих несущих частотах, т.е. между средой передачи информации и рабочими станциями находятся соответственно модемы для модуляции и демодуляции. Техника широкополосных сообщений позволяет одновременно транспортировать в коммуникационной среде довольно большой объем информации. Для дальнейшего развития дискретной транспортировки данных не играет роли, какая первоначальная информация подана в модем (аналоговая или цифровая), так как она все равно в дальнейшем будет преобразована.

Древовидная структура ЛВС

Наряду с известными топологиями вычислительных сетей – „кольцо”, „звезда” и „шина” – на практике применяется и комбинированная, например, древовидна структура. Она образуется в основном в виде комбинаций вышеназванных топологий вычислительных сетей. Основание дерева вычислительной сети располагается в точке (корень), в которой собираются коммуникационные линии информации (рис. 8.18).

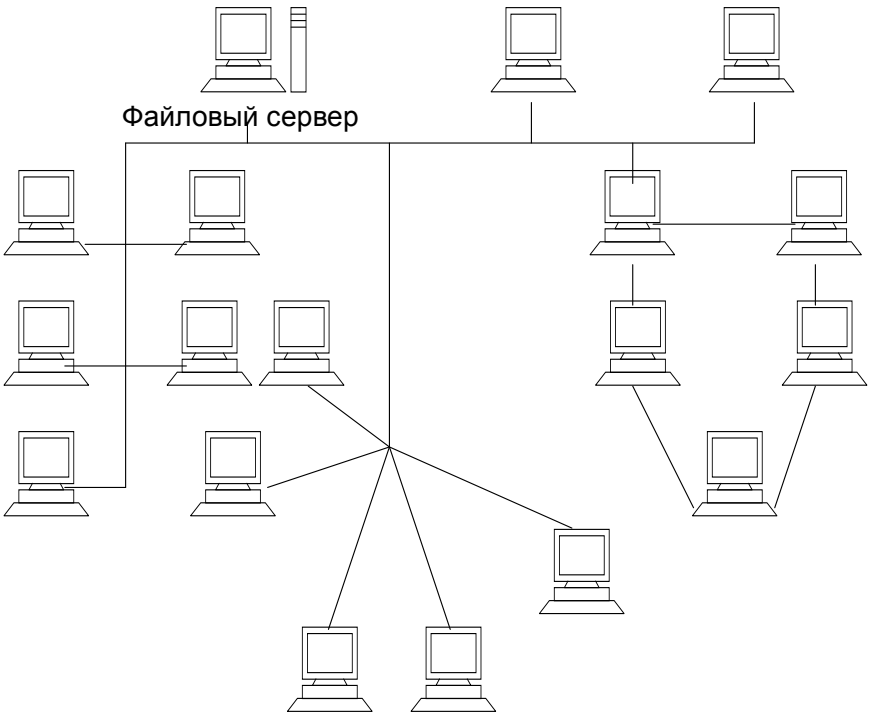


Рис. 8.18. Древовидная структура ЛВС

Вычислительные сети с древовидной структурой используются там, где невозможно непосредственное применение базовых сетевых структур в чистом виде. Для подключения большого числа рабочих станций соответственно адаптерным платам применяют сетевые усилители и/или коммутаторы. Коммутатор, обладающий одновременно и функциями усилителя, называют активным концентратором. На практике применяют две их разновидности, обеспечивающие подключение соответственно восьми или шестнадцати линий.

Устройство, к которому можно присоединить максимум три станции, называют пассивным концентратором. Его обычно используют как разветвитель. Он не нуждается в усилителе. Предпо-

сылкой для подключения пассивного концентратора является то, что максимальное возможное расстояние до рабочей станции не должно превышать нескольких десятков метров.

4. Ход лабораторной работы

1. Изучить теоретический материал, записав основные моменты лабораторной работы.

2. Заполнить таблицу с характеристиками топологий ЛВС.

Характеристики	***Топология***		
	„звезда”	***„кольцо”***	***„шина”***
Стоимость расширения			
Присоединение абонентов			
Защита от отказов			
Размеры системы			
Защищенность от прослушивания			
Стоимость подключения			
Поведение системы при высоких нагрузках			
Возможность работы в реальном режиме времени			
Разводка кабеля			
Обслуживание			

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Краткое описание существующих топологий ЛВС.
4. Выводы по работе.

6. Контрольные вопросы

1. Что такое ЛВС?
2. Какие преимущества возникают при сетевом объединении компьютеров?
3. Чем различаются описанные топологии ЛВС?
4. Какая топология является самой распостранённой?

7. Литература

1. *Олифер, В. Г.* Основы компьютерных сетей / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2009. – 305 с. – (Учебное пособие). – ISBN 978-5-49807-218-0.

2. *Фигурнов, В. Э.* IBM PC для пользователей / В. Э. Фигурнов. – Изд. 5-е, испр. и доп. – СПб. : Коруна : Информатика и компьютеры, 1994. – 352 с. – ISBN 5-87672-002-X.

3. *Крымов, Б.* Профессиональная диагностика компьютера : учеб. пособие / Б. Крымов. – М. : Триумф, 2006. – 268 с. – (Инструментальная книга). – ISBN 5-89392-142-9.

Лабораторная работа № 7. *ПОРТЫ ВВОДА-ВЫВОДА. УСТРОЙСТВА ПАРАЛЛЕЛЬНОГО И ПОСЛЕДОВАТЕЛЬНОГО ВВОДА-ВЫВОДА*

1. Цель работы

Изучить устройства параллельного и последовательного портов ввода/вывода.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

Порт ввода-вывода

Канал передачи данных между устройством и микропроцессором. Порт представляется в микропроцессоре как один или несколько адресов памяти, из которых можно прочитать или в которые можно записать данные.

Параллельный порт

Разъем ввода/вывода для подключения устройств параллельного интерфейса. Большинство принтеров подключаются к параллельному порту.

Последовательный порт

Порт компьютера для организации побайтной *асинхронной связи.* Последовательный порт называется также коммуникационным, или COM-портом.

Асинхронная связь

Форма передачи данных, в которой информация передается и принимается через нерегулярные интервалы времени, один символ за раз. Так как данные принимаются через нерегулярные интервалы времени, получающему модему должно быть передано сообщение, позволяющее ему определить, когда начинаются и заканчиваются биты данных символа. Для этого предназначены стартовый и стоповый биты.

Параллельный порт (LPT)

(25-контактный разъем). Предназначен для подключения принтера, сканера, а также внешних устройств для хранения и траспортировки информации (накопителей). До недавнего времени отличался сравнительно высокой скоростью передачи данных (около 2 Мбайт/с). Как правило, LPT-разъем на задней стенке компьютера единственный.

Последовательные порты (COM) (9-и 25-контактный разъем)

Отличаются меньшей скоростью (около 112 кбайт/с). Потому и выпадала на их долю поддержка всяческих «неспешных» устройств, например мыши или модема. Первоначально COM-портов на компьютере было четыре, однако со временем их осталось лишь два. Мышь предпочла последовательному порту свой собственный разъем PS/2, разделив его с клавиатурой, а на долю COM-порта осталась лишь поддержка медлительного модема. Со временем и модем эмигрирует к новому порту USB, тогда COM-порт окончательно и бесповоротно уйдет в прошлое.

Порт PS/2

В свое время мышь и клавиатура подключались к разным разъемам: мышь по соседству с модемом на COM-порте, а клавиатура имела свой собственный, ни на что не похожий разъем. PS/2-порт впервые появился на массовых материнских платах в 1998 году. Подключить к нему что-то кроме мыши и клавиатуры не получится.

Последовательный порт и интерфейс USB

Эту новинку, успешно дебютирующую в 2000 году, называли одной из самых значительных новаций десятилетия. Одним из главных плюсов USB является то, что на один USB-порт можно подключить 127 устройств (в отличие от старых портов: к каждому можно было подключить только одно устройство). Все USB-устройства могут подключаться к компьютеру «по цепочке» в том случае, если у каждого «звена» имеется свой USB-порт или USB-хаб на несколько портов одновременно. Единственное правило, которое следует соблюдать при работе с USB, – первыми в цепочке должны быть самые производительные устройства: принтер, сканер, колонки, накопители, а в самом конце медленные – клавиатура и мышь.

Еще одно важное качество USB – этот интерфейс позволяет подключать к компьютеру любые устройства без перезагрузки системы.

Скорость первой модификации USB (а именно к этому стандарту относятся все устройства, выпущенные до конца 2000 года) составляет около 12 Мбайт/с (на деле ряд подключенных к USB устройств работает с куда меньшей скоростью – до 1,5 Мбайт/с). Новая спецификация шины USB 2.0, принятая в апреле 2000 года, планировала увеличить скорость передачи данных до 60 Мбайт/с, однако новые устройства, поддерживающие такую скорость обмена, вышли на рынок только в конце года. USB 2.0 совместима с устройствами USB старого формата, но работать они будут с прежней скоростью.

Инфракрасный порт

Оптический порт, предназначенный для связи компьютера с другими компьютерами или устройствами посредством инфракрасного излучения, без кабелей. Инфракрасные порты применяются на некоторых переносных компьютерах, принтерах и камерах.

Порт и интерфейс FireWire (IEEE 1394)

Как ни была бы быстра и удобна шина USB, а все же существовали устройства, которым обеспечиваемой этим стандартом скорости было маловато (например, цифровая видеокамера). Именно под видеокамеры и был создан стандарт FireWire, обеспечивающий передачу данных со скоростью до 50 Мбайт/с. Сегодня контроллеры Fire-

Wire устанавливаются в материнскую плату дополнительно в виде отдельной платы для разъема PCI.

4. Ход лабораторной работы

Изучить теоретический материал, записав основные моменты лабораторной работы.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Краткое описание портов ввода-вывода.
4. Выводы по работе.

6. Контрольные вопросы

1. Дайте определение порта ввода-вывода.
2. Чем характеризуется асинхронная связь?
3. Какой интерфейс обеспечивает максимальную скорость передачи данных?
4. Какой интерфейс является самым распостранённым среди пользователей?

7. Литература

1. *Мюллер, С.* Модернизация и ремонт ПК = Upgrading and Repairing PCs : пер. с англ. / С. Мюллер. – 17-е изд. – М. : Вильямс, 2007. – 1489 с. – ISBN 978-5-8459-1126-1.
2. *Мерритт, М*. Безопасность беспроводных сетей = Wireless Securiti : пер. с англ. / М. Мерритт, Д. Поллино. – М. : Компания АйТи : ДМК Пресс, 2004. – 288 с. (Информационные технологии для инженеров). – ISBN 5-94074-248-3.
3. *Шашлов, С.* Азбука сисадмина. Энциклопедия IXBT.com / С. Шашлов. – СПб. : Питер, 2008. – 203 с. – ISBN 978-5-388-00223-5.

Лабораторная работа № 8. *РАБОТА С УТИЛИТОЙ PING*

1. Цель работы

Исследовать вероятностно-временные характеристики сети с использованием утилиты ping.

2. Приборы и материалы:

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

Утилита ping (Packet Internet Groper) является одним из главных средств, используемых для отладки сетей, и служит для принудительного вызова ответа конкретной машины. Она позволяет проверять работу программ TCP/IP на удаленных машинах, адреса устройств в локальной сети, адрес и маршрут для удаленного сетевого устройства. В выполнении команды ping участвуют система маршрутизации, схемы разрешения адресов и сетевые шлюзы. Это утилита низкого уровня, которая не требует наличия серверных процессов на зондируемой машине, поэтому успешный результат при прохождении запроса вовсе не означает, что выполняются какие-либо сервисные программы высокого уровня, а говорит о том, что сеть находится в рабочем состоянии, питание зондируемой машины включено и машина не отказала ("не висит").

Утилита ping имеется не только в UNIX, но и в большинстве реализаций TCP/IP для других операционных систем. В Windows утилита ping имеется в комплекте поставки, но представляет собой программу, выполняющуюся в сеансе DOS из командной строки.

Запросы утилиты ping передаются по протоколу ICMP (Internet Control Message Protocol). Получив такой запрос, программное обеспечение, реализующее протокол IP у адресата, немедленно посылает эхо-ответ. Эхо-запросы посылаются заданное количество раз (ключ -n) или по умолчанию до тех пор, пока пользователь не введет команду прерывания (Ctrl+C или Del), после чего выводятся статистические данные.

Поскольку с утилиты ping начинается хакерская атака, некоторые серверы в целях безопасности могут не посылать эхо-ответы (например, www.microsoft.com).

Формат команды: ping [-t][-a][-n][-l][-f][-i TTL][-v TOS] [-r][][имя машины][[-j списокУзлов]|[-k списокУзлов]][-w] (см. таблицу).

Параметры утилиты ping

Ключи	Функции
-t	Отправка пакетов на указанный узел до команд прерывания
-a	Определение адресов по именам узлов
-n	Число отправляемых запросов
-l	Размер буфера отправки
-f	Установка флага, запрещающего фрагментацию пакета
-i TTL	Задание времени жизни пакета (поле "Time To Live")
-v TOS	Задание типа службы (поле "Type Of Service")
-r	Запись маршрута для указанного числа переходов
-s	Штамп времени для указанного числа переходов
-j список узлов	Свободный выбор маршрута по списку узлов
-k список узлов	Жесткий выбор маршрута по списку узлов
-w интервал	Интервал ожидания каждого ответа в миллисекундах

Вызов утилиты Ping в командной строке Windows:

C:\Documents and Settings\<имя пользователя>ping www.mail.ru

Обмен пакетами с 207.227.119.10 по 32 байт:
Ответ от 207.227.119.10: число байт=32 время=196мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=198мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=193мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=195мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=199мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=196мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=192мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=197мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=197мс TTL=237
Время ожидания запроса истекло.
Ответ от 207.227.119.10: число байт=32 время=202мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=192мс TTL=237

Ответ от 207.227.119.10: число байт=32 время=191мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=193мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=200мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=196мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=196мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=199мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=196мс TTL=237
Ответ от 207.227.119.10: число байт=32 время=193мс TTL=237
Статистика Ping для 207.227.119.10: Пакетов: послано = 20, получено = 19, потеряно = 1 (5% потерь).
Приблизительное время передачи и приема:
наименьшее = 191 мс, наибольшее = 202 мс, среднее = 186 мс.
Максимальное значение TTL по умолчанию принимается равным 255 узлам.

Следовательно, чтобы определить количество узлов, через которые прошел пакет, надо от 255 отнять полученное значение TTL.

Практическое использование

- Можно узнать IP-адрес по доменному имени.
- Можно узнать, работает ли сервер. Например, системный администратор может узнать, „завис” ли только веб-сервер или на сервере глобальные проблемы.
- Можно узнать, есть ли связь с сервером. Например, проблемы с настройкой DNS-серверов на машине можно узнать, задав в ping сначала доменное имя, а потом IP-адрес.
- Также можно узнать качество канала, посмотрев, сколько ответов не пришло.

4. Ход лабораторной работы

1. Изучить утилиту ping.
2. Произвести трассировку узлов www.mail.ru, www.rome-guide.it, www.novol.pl, www.newslink.org или любых других по желанию, но не менее 7 узлов.
3. Представить графики статистической информации.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.

3. Обзор утилиты ping.
4. Выводы по работе.

6. Контрольные вопросы

1. Для чего применяется утилита ping?
2. Каков формат команды ping?
3. Что можно определить с помощью утилиты ping?

7. Литература

1. Вопросы о ping-е // CITFORUM.RU : Форум высоких технологий. 2011. URL: http://citforum.ru/internet/articles/ping/ (дата обращения: 20.05.2012).
2. Ping и Traceroute // CITFORUM.RU Форум высоких технологий. 2011. URL: http://citforum.ru/nets/semenov/4/45/ping_451.shtml (дата обращения: 20.05.2012).
3. *Фигурнов, В.Э.* IBM PC для пользователя / В.Э. Фигурнов. – Изд. 5-е, испр. и доп. – СПб. : Коруна : Информатика и компьютеры, 1994. – 352 с. – ISBN 5-87672-002-X.

Лабораторная работа № 9. *РАБОТА С УТИЛИТОЙ TRACEROUTE*

1. **Цель работы**

Исследовать топологии фрагментов Internet с использованием утилиты traceroute.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

Traceroute – это служебная компьютерная программа, предназначенная для определения маршрутов следования данных в сетях TCP/IP. Traceroute основана на протоколе ICMP.

Программа traceroute выполняет отправку данных указанному узлу сети, при этом отображая сведения о всех промежуточных

маршрутизаторах, через которые прошли данные на пути к целевому узлу. В случае проблем при доставке данных до какого-либо узла программа позволяет определить, на каком именно участке сети возникли неполадки. Здесь хочется отметить, что программа работает только в направлении от источника пакетов и является весьма грубым инструментом для выявления неполадок в сети. В силу особенностей работы протоколов маршрутизации в сети Интернет обратные маршруты часто не совпадают с прямыми, причем это справедливо для всех промежуточных узлов в трейсе. Поэтому ICMP-ответ от каждого промежуточного узла может идти своим собственным маршрутом, затеряться или прийти с большой задержкой, хотя в реальности с пакетами, которые адресованы конечному узлу, этого не происходит. Кроме того, на промежуточных маршрутизаторах часто стоит ограничение числа ответов ICMP в единицу времени, что приводит к появлению ложных потерь.

Traceroute входит в поставку большинства современных сетевых операционных систем. В системах Microsoft Windows эта программа носит название tracert, а в системах GNU/Linux, Cisco IOS и Mac OS — traceroute.

Для определения промежуточных маршрутизаторов traceroute отправляет серию (обычно три) пакетов данных целевому узлу, при этом каждый раз увеличивая на 1 значение поля TTL («время жизни»). Это поле обычно указывает максимальное количество маршрутизаторов, которое может быть пройдено пакетом. Первая серия пакетов отправляется с TTL, равным 1, и поэтому первый же маршрутизатор возвращает обратно сообщение ICMP, указывающее на невозможность доставки данных. Traceroute фиксирует адрес маршрутизатора, а также время между отправкой пакета и получением ответа (эти сведения выводятся на монитор компьютера). Затем traceroute повторяет отправку серии пакетов, но уже с TTL, равным 2, что позволяет первому маршрутизатору пропустить их дальше.

Процесс повторяется до тех пор, пока при определённом значении TTL пакет не достигнет целевого узла. При получении ответа от этого узла процесс трассировки считается завершённым.

На оконечном хосте IP-дейтаграмма с TTL = 1 не отбрасывается и не вызывает ICMP-сообщения типа „Срок истёк”, а должна быть отдана приложению. Достижение пункта назначения определяется сле-

дующим образом: отсылаемые traceroute дейтаграммы содержат UDP-пакет с таким номером UDP-порта адресата (превышающим 30 000), что он заведомо не используется на адресуемом хосте. В пункте назначения UDP-модуль, получая подобные дейтаграммы, возвращает ICMP-сообщения об ошибке «порт недоступен». Таким образом, чтобы узнать о завершении работы, программе traceroute достаточно обнаружить, что поступило ICMP-сообщение об ошибке этого типа.

Вызов утилиты traceroute в командной строке Windows:

C:\Documents and Settings\<имя пользователя>tracert www.mail.ru

Трассировка маршрута к www.mail.ru [91.198.174.2] с максимальным числом прыжков 30:

```
 1    1 ms   <1 ms   <1 ms  vpn4.kras.gldn [10.10.1.14]
 2     2 ms    <1 ms    <1 ms  C7604-BRAS4-FTTB.ranetka.ru [80.255.150.41]
 3    1 ms    1 ms    4 ms  C76-External.ranetka.ru [80.255.128.162]
 4    1 ms   <1 ms   <1 ms  pe-l.Krasnoyarsk.gldn.net [195.239.173.37]
 5   79 ms   79 ms   98 ms  cat01.Stockholm.gldn.net [194.186.157.62]
 6   131 ms   131 ms   132 ms  ams-ix.2ge-2-1.br1-knams.mail.org [195.69.145.176]
 7    131 ms    131 ms    131 ms  te-8-2.csw1-esams.mail.org [91.198.174.254]
 8  133 ms  134 ms  133 ms  mail.org [91.198.174.2]
```

Как обычно, имя машины может быть задано в символической или числовой формах. Выходная информация представляет собой список машин, начиная с первого шлюза и кончая пунктом назначения. Кроме того, показано полное время прохождения каждого шлюза.

Команда traceroute работает путем установки поля времени жизни (числа переходов) исходящего пакета таким образом, чтобы это время истекало до достижения пакетом пункта назначения. Когда время жизни истечет, текущий шлюз отправит сообщение об ошибке на машину-источник. Каждое приращение поля времени жизни позволяет пакету пройти на один шлюз дальше.

Команда traceroute посылает для каждого значения поля времени жизни три пакета. Если промежуточный шлюз распределяет трафик по нескольким маршрутам, то эти пакеты могут возвращаться разны-

ми машинами. В этом случае на печать выводятся они все. Некоторые системы не посылают уведомлений о пакетах, время жизни которых истекло, а некоторые посылают уведомления, которые поступают обратно на машину-источник только после того, как истекло время их ожидания командой traceroute. Эти шлюзы обозначаются рядом звездочек. Даже если конкретный шлюз определить нельзя, traceroute чаще всего сможет увидеть следующие за ним узлы маршрута.

4. Ход лабораторной работы

1. Изучить утилиту traceroute.
2. Произвести трассировку узлов www.mail.ru, www.rome-guide.it, www.novol.pl, www.newslink.org или любых других по желанию, но не менее 7 узлов.
3. Описать маршрут прохождения трассировки.
4. Предоставить графики времени прохождения шлюзов (по количеству узлов) с анализом узких мест сети.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Обзор утилиты traceroute.
4. Выводы по работе.

6. Контрольные вопросы

1. Для чего применяется утилита traceroute?
2. Каков формат команды traceroute?
3. Что можно определить с помощью утилиты traceroute?

7. Литература

1. Ping и Traceroute // CITFORUM.RU Форум высоких технологий. 2011. URL: http://citforum.ru/nets/semenov/4/45/ping_451.shtml (дата обращения: 20.05.2012).
2. *Фигурнов, В.Э.* IBM PC для пользователя / В.Э. Фигурнов. – Изд. 5-е, испр. и доп. – СПб. : Коруна : Информатика и компьютеры, 1994. – 352 с. – ISBN 5-87672-002-X.
3. *Кушнир, А.Н.* Новейшая энциклопедия компьютера / А.Н. Кушнир. – М. : Эксмо, 2008. – 975 с. – (Новейшая энциклопедия). – ISBN 978-5-699-24136-1.

Лабораторная работа № 10. *СПОСОБЫ СОЕДИНЕНИЯ ДВУХ КОМПЬЮТЕРОВ ДЛЯ СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ ФАЙЛОВ*

1. Цель работы

Изучить способы соединения двух компьютеров для совместного использования файлов.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

Компьютерные сети - это совокупность узлов, объединенных каналами передачи данных. Компьютерные сети также называются информационно-вычислительными сетями, вычислительными сетями, или сетями передачи данных.

Рассмотрим способы соединения двух компьютеров, которые используются, как правило, для создания временных сетей компьютер-компьютер с целью совместного использования файлов. В этом случае компьютеры соединяются непосредственно друг с другом без помощи дополнительного коммуникационного оборудования.

Для обмена данными между двумя компьютерами (например, настольными и переносными ПК) можно использовать различные способы их соединения, а именно:

- прямое соединение компьютеров через последовательные и параллельные порты (COM, USB, LPT, IrDA, Bluetooth);
- удаленное соединение через модемы;
- соединение в локальную сеть, используя сетевые карты и проводные линии связи;
- соединение в локальную сеть, используя встроенные беспроводные интерфейсы Wi-Fi.

Порт COM. Наиболее простой и доступный способ соединения двух компьютеров - это прямое соединение их через асинхронный последовательный порт COM. В этом случае для соединения используется только нуль-модемный кабель. Этот способ подключения самый доступный, так как все персональные компьютеры оснащены портами COM.

USB-порт. Способ соединение двух компьютеров через USB-порты. USB – это универсальный интерфейс (последовательный порт) для подключения 127 устройств. Для соединения двух компьютеров требуется только кабель usb link.

LPT-порт. Два компьютера могут быть подключены через параллельные LPT-порты, которые имеют восемь разрядов. В этом способе для соединения двух компьютеров используется только нуль-модемный кабель.

IrDA - инфракрасные порты, предназначены для беспроводного подключения карманных или блокнотных компьютеров к настольному компьютеру. Связь обеспечивается при условии прямой видимости, дальность передачи данных не более 1 м. Если в компьютере нет встроенного iRDA-адаптера, то он может быть выполнен в виде дополнительного внешнего устройства (USB/iRDA- адаптера), подключаемого через USB-порт. Этот способ предназначен для беспроводного подключения карманных или блокнотных ПК к настольному компьютеру.

Bluetooth ("блютус") - высокоскоростной микроволновый стандарт, позволяющий передавать данные на расстояниях до 10 метров. Если нет встроенного bluetooth-адаптера, то он может быть выполнен в виде дополнительного внешнего устройства (USB bluetooth-адаптера), подключаемого через USB-порт. Этот способ предназначен для беспроводного подключения карманных или блокнотных компьютеров к настольному компьютеру.

Связь с помощью модема. Модем – это устройство, которое преобразует цифровой сигнал компьютера в аналоговый частотный сигнал, передаваемый по аналоговым телефонным линиям. С помощью модема можно подключать два компьютера по телефонным линиям связи.

При соединении двух компьютеров в локальную сеть используют сетевые карты и проводные линии связи. В качестве линий связи могут быть применены коаксиальный кабель или кабель "витая пара".

Рассмотрим первый способ соединений компьютеров на основе стандарта 10Base-2. Для выполнения соединения двух компьютеров нужны две сетевые карты с разъемом UTP, тонкий коаксиальный кабель с волновым сопротивлением 50 Ом, два Т-коннектора и два терминатора. После подключения требуется настройка операционной системы.

Второй способ соединений компьютеров на основе стандарта 10Base-T. Для выполнения соединения двух компьютеров нужны две сетевые карты с разъемом RJ-45, кабель "витая пара" ("нуль-хабный" кабель) и две вилки RJ-45. В этом случае требуется специальная разводка кабеля, т.е. разводка проводников на разъемах кабеля должна быть нестандартной и соответствовать разводке "нуль-хабного" кабеля. Например, для четырехпроводного кабеля (две пары) разводка "нуль-хабного" кабель следующая: разводка 1-2-3-6 на одном разъеме должна соответствовать разводке 3-6-1-2 на другом. После подключения требуется настройка операционной системы.

Беспроводная связь Wi-Fi. Этот способ соединения двух компьютеров напрямую использует встроенные беспроводные интерфейсы Wi-Fi. Стандартом беспроводной связи для локальных сетей является технология Wi-Fi, которая обеспечивает соединение „точка-точка” (соединение Ad-Hoc), применямое для подключения двух ПК. В этом способе для соединения двух ПК требуется два адаптера Wi-Fi.

4. Ход лабораторной работы

Изучить теоретический материал, записав основные моменты лабораторной работы.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Краткое описание способов соединения двух компьютеров.
4. Выводы по работе.

6. Контрольные вопросы

1. Опишите способы соединения двух компьютеров.
2. Какие способы связи являются беспроводными?
3. Какой способ связи обеспечивает максимальную скорость передачи данных?

7. Литература

1. *Олифер, В. Г.* Основы компьютерных сетей / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2009. – 305 с. – (Учебное пособие). – ISBN 978-5-49807-218-0.

2. *Леонтьев, В.П.* «Новейшая энциклопедия персонального компьютера» / В.П. Леонтьев. – М: Олма Медиа Групп, 2007. – 734 с. – ISBN 978-5-373-00483-1.
3. *Максимов, Н. В.* Архитектура ЭВМ и вычислительных систем : учеб. для среднего проф. образования по специальности ″Информатика и вычислительная техника″ / Н.В. Максимов, Т.Л. Партыка, И.И. Попов. – М. : Форум, Инфа-М, 2007. – 511 с. – (Профессиональное образование). – ISBN 978-5-91134-105-3 (Форум). – ISBN 978-5-16-002970-2 (Инфа-М).

Лабораторная работа № 11. *ОБЖИМ КАБЕЛЯ UTP5/STP5 (ВИТАЯ ПАРА CAT5)*

1. Цель работы

Научиться обжимать кабель «витая пара» 5-й категории.

2. Приборы и материалы

Кабель «витая пара» 5-й категории, устройство для обжимки кабеля, коннекторы RJ-45.

3. Краткие теоретические сведения

Витая пара (англ. twisted pair) — вид кабеля связи, представляет собой одну или несколько пар изолированных проводников, скрученных между собой (с небольшим числом витков на единицу длины), покрытых пластиковой оболочкой. Свивание проводников производится с целью повышения связи проводников одной пары (электромагнитная помеха одинаково влияет на оба провода пары) и последующего уменьшения электромагнитных помех от внешних источников, а также взаимных наводок при передаче дифференциальных сигналов. Для снижения связи отдельных пар кабеля (периодического сближения проводников различных пар) в кабелях UTP категории 5 и выше провода пары свиваются с различным шагом. Витая пара – один из компонентов современных структурированных кабельных систем. Используется в телекоммуникациях и компьютерных сетях в качестве сетевого носителя во многих технологиях, таких как Ethernet, ARCNet и Token ring. В настоящее время благодаря своей

дешевизне и лёгкости в установке является самым распространённым решением для построения локальных сетей.

Кабель подключается к сетевым устройствам при помощи соединителя 8P8C (RJ45 или RJ-45), немного большего, чем телефонный соединитель RJ11.

В зависимости от наличия защиты — электрически заземлённой медной оплетки или алюминиевой фольги вокруг скрученных пар – определяют разновидности данной технологии.

Незащищенная витая пара

Неэкранированная витая пара (UTP — Unscreened twisted pair) — экранирование полностью отсутствует;

Фольгированная витая пара (FTP — Foiled twisted pair) — также известна как S/UTP [1] присутствует один общий внешний экран;

Фольгированная экранированная витая пара (SFTP — Shielded Foiled twisted pair) — отличается от FTP наличием дополнительного внешнего экрана из медной оплетки;

Виды защищенной витой пары:

Стандартная (STP — Shielded twisted pair) присутствет экран для каждой пары;

Экранированная витая пара (S/STP — Screened shielded twisted pair) отличается от STP наличием дополнительного общего внешнего экрана.

Экранирование обеспечивает лучшую защиту от электромагнитных наводок как внешних, так и внутренних, и т. д. Экран по всей длине соединен с неизолированным дренажным проводом, который объединяет экран в случае разделения на секции при излишнем изгибе или растяжении кабеля. В зависимости от структуры проводников кабель применяется одно- и многожильный. В первом случае каждый провод состоит из одной медной жилы, а во втором — из нескольких.

Одножильный кабель не предполагает прямых контактов с подключаемой периферией. То есть, как правило, его применяют для прокладки в коробах, стенах и так далее с последующим оконечиванием розетками. Связано это с тем, что медные жилы довольно толстые и при частых изгибах быстро ломаются. Однако для «врезания» в разъемы панелей розеток такие жилы подходят как нельзя лучше.

В свою очередь, многожильный кабель плохо переносит «врезание» в разъёмы панелей розеток (тонкие жилы разрезаются), но заме-

чательно ведет себя при изгибах и скручиваниях. Кроме того, многожильный провод обладает большим затуханием сигнала. Поэтому многожильный кабель используют в основном для изготовления патчкордов (PatchCord), соединяющих периферию с розетками.

Конструкция кабеля

Кабель обычно состоит из четырёх пар. Проводники в парах изготовлены из монолитной медной проволоки толщиной 0,5 – 0,65 мм. Кроме метрической, применяется система AWG, в которой эти величины составляют 24 или 22 соответственно. Толщина изоляции около 0,2 мм, материал обычно поливинилхлорид (английское сокращение PVC), для более качественных образцов 5-й категории – полипропилен (PP), полиэтилен (PE). Особенно высококачественные кабели имеют изоляцию из вспененного (ячеистого) полиэтилена, который обеспечивает низкие диэлектрические потери, или тефлона, обеспечивающего уникальный рабочий диапазон температур.

Также внутри кабеля встречается так называемая «разрывная нить» (обычно капрон), которая используется для облегчения разделки внешней оболочки: при вытягивании она делает на оболочке продольный разрез, который открывает доступ к кабельному сердечнику, гарантированно не повреждая изоляцию проводников.

Внешняя оболочка имеет толщину 0,5 - 0,6 мм и обычно изготавливается из привычного поливинилхлорида с добавлением мела, который повышает хрупкость. Это необходимо для точного облома по месту надреза лезвием отрезного инструмента. Кроме этого начинают применяться так называемые «молодые полимеры», которые не поддерживают горения и не выделяют при нагреве галогенов (такие кабели маркируются как LSZH – Low Smoke Zero Halogen и обычно имеют яркую окраску внешней оболочки).

Самый распространенный цвет оболочки – серый. Оранжевая окраска, как правило, указывает на негорючий материал оболочки, который позволяет прокладывать линии в закрытых областях. В общем случае цвета не обозначают особых свойств, но их применение позволяет легко отличать коммуникации с разным функциональным назначением как при монтаже, так и обслуживании.

Отдельно нужно отметить маркировку. Кроме данных о производителе и типе кабеля она обязательно включает в себя метровые или футовые метки.

Форма внешней оболочки также может быть различна. Чаще других применяется самая простая – круглая. Только для прокладки под половым покрытием по очевидной причине используется плоский кабель.

Кабели для наружной прокладки обязательно имеют влагостойкую оболочку из полиэтилена, которая наносится (как правило) вторым слоем поверх обычной, поливинилхлоридной. Кроме этого возможны заполнение пустот в кабеле водоотталкивающим гелем и бронирование с помощью гофрированной ленты или стальной проволоки.

Категории кабеля

Существует несколько категорий кабеля „витая пара", которые нумеруются от CAT1 до CAT7 и определяют эффективный пропускаемый частотный диапазон. Кабель более высокой категории обычно содержит больше пар проводов и каждая пара имеет больше витков на единицу длины. Категории неэкранированной витой пары описываются в стандарте EIA/TIA 568 (Американский стандарт проводки в коммерческих зданиях).

CAT1 (полоса частот 0.1 МГц) – телефонный кабель, всего одна пара (в России применялся кабель без скруток — «лапша», у него характеристики не хуже, но больше влияние помех). В США использовался ранее только в «скрученном» виде. Применяется только для передачи голоса или данных при помощи модема.

CAT2 (полоса частот 1 МГц) – старый тип кабеля, 2 пары проводников, поддерживал передачу данных на скоростях до 4 Мбит/с, использовался в сетях token ring и ARCNet. Сейчас иногда встречается в телефонных сетях.

CAT3 (полоса частот 16 МГц) – 4-парный кабель, использовался при построении локальных сетей 10BASE-T и token ring, поддерживает скорость передачи данных до 10 или 100 Мбит/с по технологии 100BASE-T4. В отличие от предыдущих двух отвечает требованиям стандарта IEEE 802.3. Также до сих пор встречается в телефонных сетях.

CAT4 (полоса частот 20 МГц). Кабель состоит из 4 скрученных пар, использовался в сетях token ring, 10BASE-T, 100BASE-T4, скорость передачи данных не превышает 16 Мбит/с по одной паре, сейчас не используется.

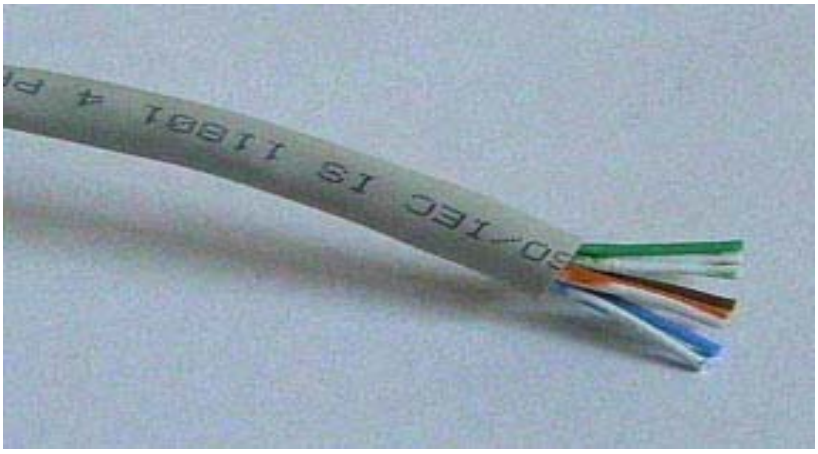


Рис. 8.19. Кабель из 4 неэкранированных витых пар

CAT5 (полоса частот 100 МГц) – 4-парный кабель, это и есть то, что обычно называют кабель «витая пара» (рис. 8.19). Благодаря высокой скорости передачи до 100 Мбит/с при использовании 2 пар и до 1000 Мбит/с при использовании 4 пар, является самым распространённым сетевым носителем, использующимся в компьютерных сетях до сих пор. При прокладке новых сетей пользуются несколько усовершенствованным кабелем CAT5e (полоса частот 125 МГц), который лучше пропускает высокочастотные сигналы. Ограничение на длину кабеля между устройствами (компьютер-свитч, свитч-компьютер, свитч-свитч) 100 м. Ограничение хаб-хаб 5 м.

CAT6 (полоса частот 250 МГц) применяется в сетях Fast Ethernet и Gigabit Ethernet, состоит из 4 пар проводников и способен передавать данные на скорости до 1000 Мбит/с. Добавлен в стандарт в июне 2002 года. Существует категория CAT6a, в которой увеличена частота пропускаемого сигнала до 500 МГц. По данным IEEE, в 70 % установленных сетей в 2004 году применялся кабель категории CAT6.

CAT7. Спецификация на данный тип кабеля пока не утверждена, скорость передачи данных до 100 Гбит/с, частота пропускаемого сигнала до 600–700 МГц. Кабель этой категории экранирован. Седьмая категория витой пары не UTP, а S/FTP (Screened Fully shielded Twisted Pair). Благодаря двойному экрану длина кабеля может превышать 100 м.

Схемы обжима витой пары

Для обжима витой пары UTP используются разъемы стандарта RJ-45 (рис. 8.20), которые в зависимости от вида кабеля «витой пары» бывают:

- экранированными или неэкранированными;
- конструктивно выполненными со вставками или без вставок. Вставки выполняют роль направляющих для проводников «витой пары», упрощающих заправку проводников в корпус разъема;
- для одножильных или многожильных «витых пар».

Для обжимки «витых пар» используют специальный инструмент, который имеет три рабочие области и соответственно выполняет три функции:

1. Ближе всего к рукояткам устройства располагается область, в которой установлен нож для обрезания проводников «витой пары». Также в этой области есть специальная выемка для снятия внешней изоляции с круглого кабеля (рис. 8.21).

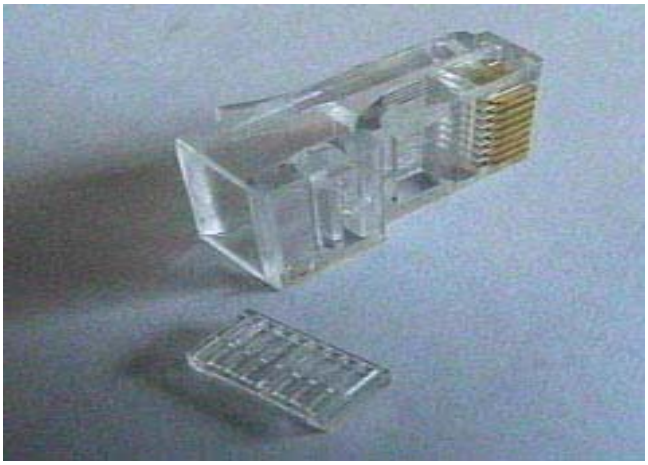


Рис. 8.20. Разъем RJ-45 для «витой пары» + вставка

2. В центре находится гнездо для обжима разъема RJ-45.

3. В верхней части устройства, область для зачистки наружной изоляции витой пары UTP (внутренняя изоляция проводников не зачищается, а прорезается контактами разъема).

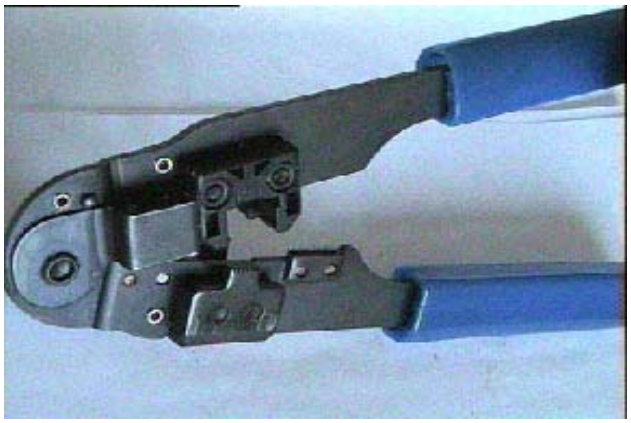


Рис. 8.21. Устройство для зачистки и обжима сетевого кабеля

Существует две схемы обжимки кабеля: прямой кабель и перекрёстный (кроссовер) кабель (рис. 8.22).

а)

Рис. 8.22. Схемы обжимки кабеля: а – прямой кабель (см. также с. 243)

б)

Рис. 8.22. Схемы обжимки кабеля: б – перекрёстный (кросс-овер) кабель (окончание)

Первая схема используется для соединения компьютера со свитчем/хабом, вторая для соединения двух компьютеров напрямую.

4. Ход лабораторной работы

1. Изучить теоретический материал, записав основные моменты лабораторной работы.

2. Обжать кабель «витая пара».

3. Вначале проводят зачистку наружной изоляции кабеля. При зачистке плоского кабеля его упирают в специальный выступ на устройстве, расположенный в области зачистки, чтобы получить глубину зачистки под стандартный разъем, зажимают кабель и рывком производят зачистку. Немного более сложным выглядит процесс зачистки круглых кабелей витых пар. Наружную изоляцию круглого кабеля лучше только слегка надрезать, осторожно поворачивая его в области зачистки, а затем снять кусочек изоляции по кольцевому надрезу вручную. На многих обжимных устройствах есть специальная область для снятия внешней изоляции с круглого кабеля.

4. После зачистки разводят провода сетевого кабеля в одной плоскости в определенном порядке, выравнивают длину всех проводов и еще раз ровно подрезают (рис. 8.23).

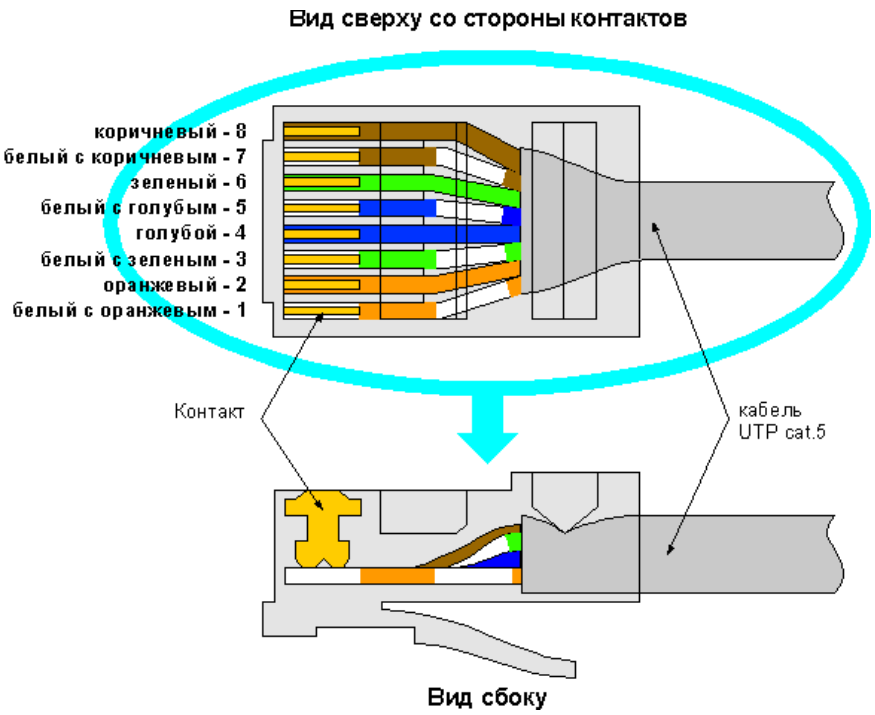


Рис. 8.23. Схема разводки кабеля

Затем производят заправку проводников в разъем и опрессовку. Рекомендуется по возможности, использовать разъемы без вставки, так как процесс заправки проводников в корпус такого разъема выполняется проще:

а) Если конструктивно разъем выполнен без вставки, то проводники аккуратно заправляются в его корпус до упора в торец разъема. Затем вставляют разъем в гнездо обжимного устройства и надавливают до тех пор пока устройство полностью не закроется.

б) Если в конструкцию разъема входит вставка, то сначала на проводники «витой пары» надевается вставка. Вставка имеет форму крышки спичечного коробка, на одной из поверхностей которого имеются прорези по количеству проводников в витой паре. Вставку надевают на проводники таким образом, чтобы прорези были обращены к корпусу разъема. После насаживания вставки проводники витой пары еще раз подрезают и выравнивают срез с краем вставки. Для закрепления вставки в этом положении полезно у ее противоположного конца обжать витую пару пальцами, чтобы вставка не смещалась. Затем вставку с проводниками вставляют в корпус разъема до тех пор, пока она не упрется в торец разъема, и обжимают разъем также, как в случае разъема без вставки.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Краткое описание кабеля «витая пара».
4. Схемы обжима кабеля «витая пара».
5. Выводы по работе.

6. Контрольные вопросы

1. Строение кабеля «витая пара».
2. Чем отличаются кабели «витая пара» различных категорий?
3. Каковы ограничения на применение «витой пары»?
4. Чем различаются схемы соединения „прямой кабель” и „перекрёстный кабель”?

7. Литература

1. *Олифер, В. Г.* Основы компьютерных сетей / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2009. – 305 с. – (Учебное пособие). – ISBN 978-5-49807-218-0.

2. Обжим сетевого кабеля // OBZHIMKABELJA.RU : рук. по монтажу ЛВС. 2009. URL: http://obzhimkabelja/ru/ (дата обращения: 15.02.2012).

Лабораторная работа № 12. *СОЕДИНЕНИЕ ДВУХ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ С ПОМОЩЬЮ ВИТОЙ ПАРЫ*

1. Цель работы

Изучить соединение двух персональных компьютеров с помощью витой пары.

2. Приборы и материалы

2 ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3, кабель «витая пара» 5-й категории, устройство для обжимки кабеля, коннекторы RJ-45.

3. Краткие теоретические сведения

Для построения компьютерных сетей применяются линии связи, использующие различную физическую среду. В качестве физической

среды в коммуникациях используются металлы (в основном медь), сверхпрозрачное стекло (кварц) или пластик и эфир. Физическая среда передачи данных может представлять собой кабель "витая пара", коаксиальные кабель, волоконно-оптический кабель и окружающее пространство.

Линии связи, или линии передачи, данных - это промежуточная аппаратура и физическая среда, по которой передаются информационные сигналы (данные).

В одной линии связи можно образовать несколько каналов связи (виртуальных или логических каналов), например путем частотного или временного разделения каналов. Канал связи - это средство односторонней передачи данных. Если линия связи монопольно используется каналом связи, то в этом случае линию связи называют каналом связи.

Канал передачи данных - это средства двухстороннего обмена данными, которые включают в себя линии связи и аппаратуру передачи (приема) данных. Каналы передачи данных связывают между собой источники информации и приемники информации.

В качестве линий связи могут быть применены коаксиальный кабель или кабель "витая пара". Рассмотрим первый способ соединений компьютеров на основе стандарта 10Base-2. Для выполнения соединения двух компьютеров нужны две сетевые карты с разъемом UTP, тонкий коаксиальный кабель с волновым сопротивление 50 Ом, два Т-коннектора и два терминатора. После подключения требуется настройка операционной системы.

Второй способ – соединение компьютеров на основе стандарта 10Base-T. Для выполнения соединения двух компьютеров нужны две сетевые карты с разъемом RJ-45, кабель "витая пара" ("нуль-хабный" кабель) и две вилки RJ-45. В этом случае требуется специальная разводка кабеля, т.е. разводка проводников на разъемах кабеля должна быть нестандартной и соответствовать разводке "нуль-хабного" кабеля. Например, для четырехпроводного кабеля (две пары) разводка "нуль-хабного" кабель следующая: разводка 1-2-3-6 на одном разъеме должна соответствовать разводке 3-6-1-2 на другом разъеме. После подключения требуется настройка операционной системы.

4. Ход лабораторной работы

1. Изучить теоретический материал, записав основные моменты лабораторной работы.

2. Обжать кабель «витая пара» по схеме «перекрёстный кабель».

3. Присвоить имена компьютерам.

1. На ПК1 щелкните правой кнопкой мыши по значку «Мой компьютер» на рабочем столе и выберите «Свойства» (рис. 8.24).

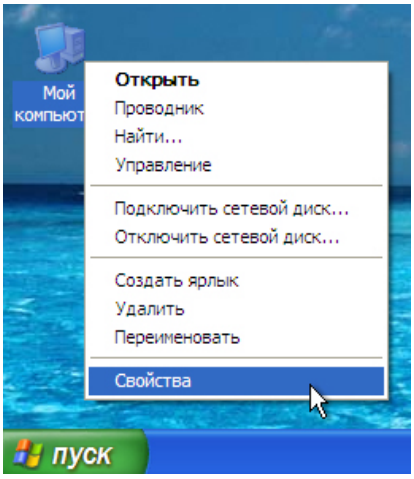


Рис. 8.24. Свойства ПК

2. В открывшемся окне выберите вкладку «Имя компьютера» и нажмите кнопку «Изменить…» (рис. 8.25).

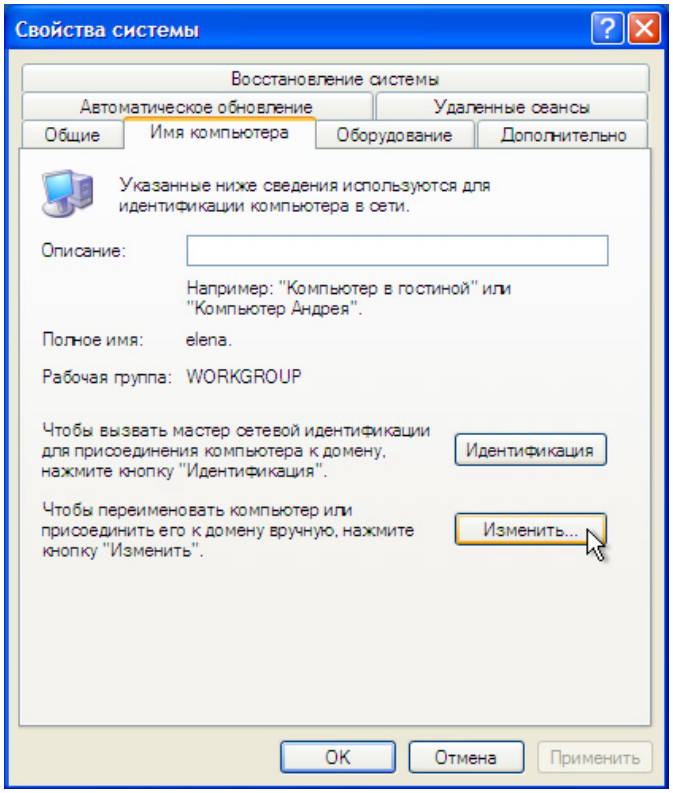


Рис. 8.25. Изменение имени ПК

3. Введите имя компьютера на английском языке – PC1 и имя рабочей группы – WORKGROUP. Имя рабочей группы может быть уже указано (по умолчанию оно одинаково для всех компьютеров под ОС Windows). В таком случае просто проверьте, чтобы оно было WORKGROUP (рис. 8.26).

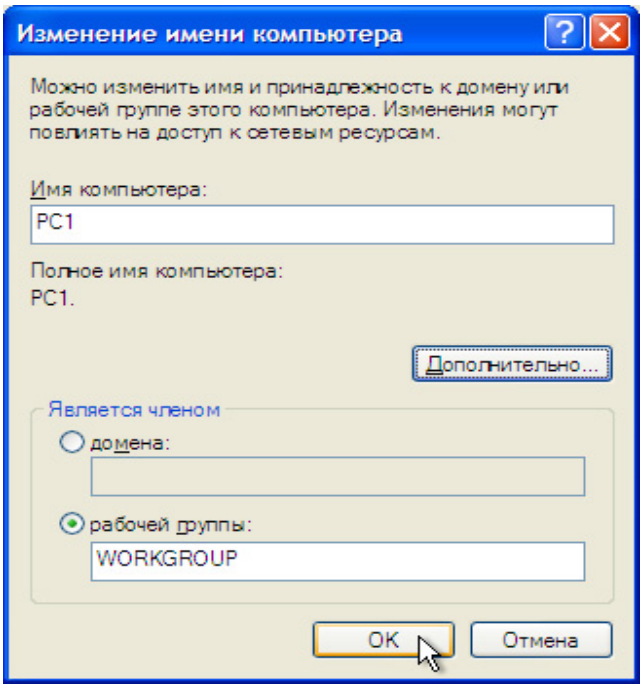


Рис. 8.26. Изменение имени ПК

4. Нажмите кнопку «ОК» в этом окне и в следующем. Затем перезагрузите компьютер, чтобы изменения вступили в силу. То же самое проделайте со вторым компьютером (ПК2), только имя ему присвойте PC2. Рабочая группа на обоих компьютерах должна быть одинаковой - WORKGROUP. По окончании настройки, второй компьютер также нужно перезагрузить.

5. Присвоить адреса компьютерам

1. На ПК1 нажмите «Пуск» - «Настройка» и дважды щелкните мышью по пункту «Сетевые подключения» (рис. 8.27).

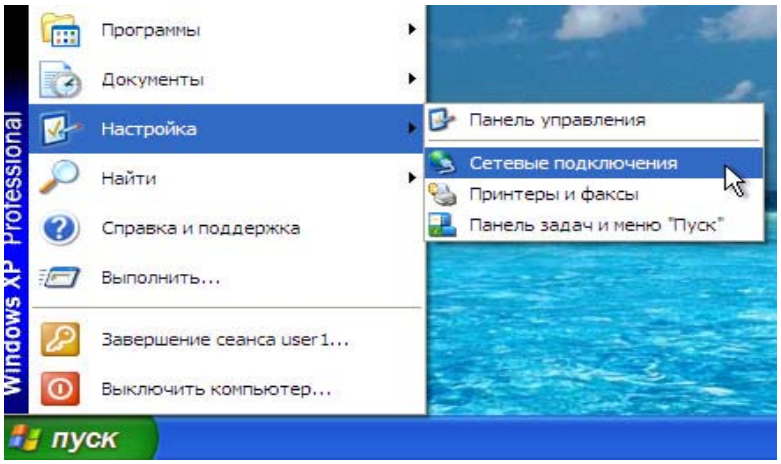


Рис. 8.27. Сетевые подключения

2. Щелкните правой кнопкой мыши «Подключение по локальной сети» и нажмите «Свойства» (рис. 8.28).

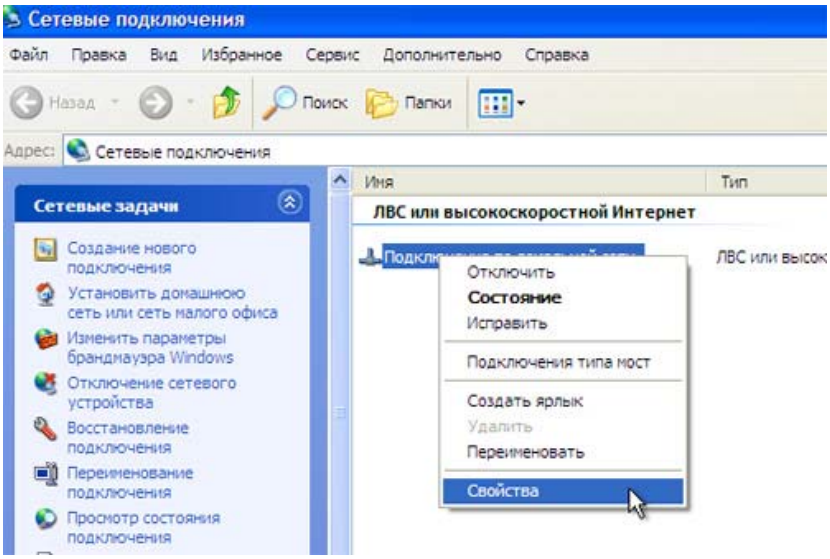


Рис. 8.28. Свойства подключения по локальной сети

3. В открывшемся окне выберите «Протокол Интернета (TCP/IP)» и нажмите кнопку «Свойства» (рис. 8.29).

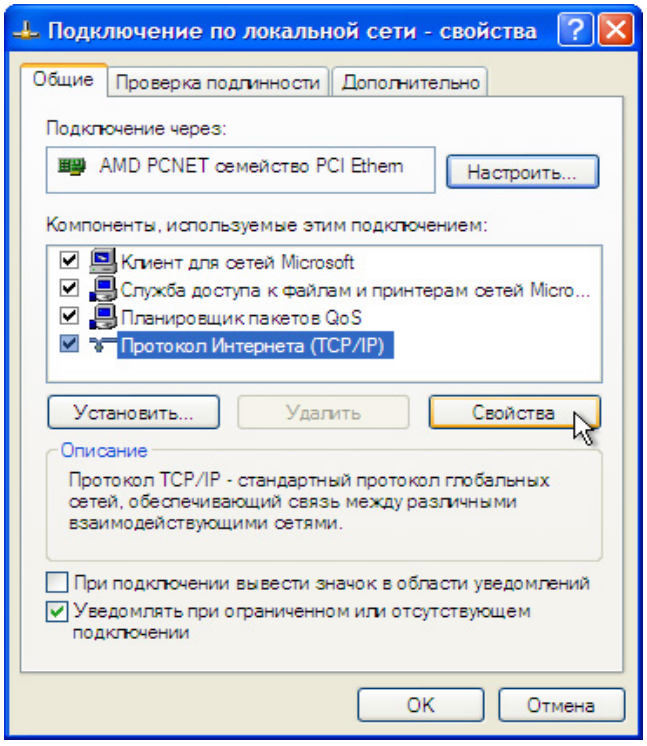


Рис. 8.29. Свойства протокола TCP\IP

4. Отметьте пункт «Использовать следующий IP-адрес». В поле «IP-адрес» введите адрес вашего компьютера – 192.168.0.100. Щелкните мышью по полю «Маска подсети» - там появится соответствующая адресу компьютера величина (рис. 8.30).

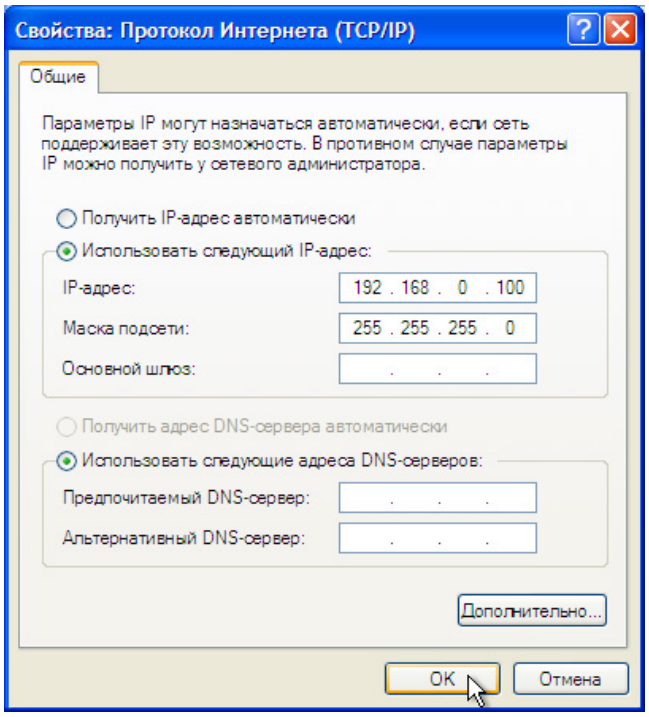


Рис. 8.30. Присвоение IP-адреса

5. Нажмите кнопку «ОК» в этом окне и «Закрыть» в следующем. Подождите несколько секунд, пока настройки вступят в силу. Закройте окно сетевых подключений.

Таким образом вы присвоили адрес одному компьютеру под управлением ОС Windows XP – ПК1. Теперь сядьте за другой компьютер (ПК2) и проделайте всё то же самое, только вместо 192.168.0.100 присвойте ему адрес 192.168.0.6

4.5. Проверить связь в сети

1. На ПК2 откройте меню «Пуск» и выберите «Выполнить». В поле введите cmd (на английском) и нажмите «ОК» (рис. 8.31).

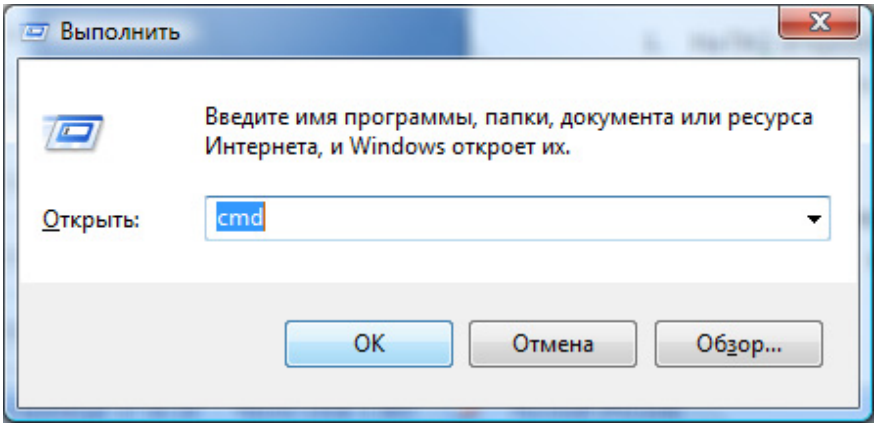


Рис. 8.31. Вызов командной строки

2. Откроется командный интерпретатор Windows. Здесь можно набирать различные команды, все команды вводятся только на английском языке. Нас пока интересует единственная команда – команда проверки связи с другим компьютером. Она называется ping (пинг). Сейчас мы будем пинговать (от «пинг», означает «проверять связь с…») ПК1 с адресом 192.168.0.100.

Введите в командном интерпретаторе ping 192.168.0.100 (на человеческом языке это означает «проверить связь с компьютером, имеющим адрес 192.168.0.100») и нажмите на клавиатуре Enter. Процесс пошел, мы видим, как ПК2 отправляет пакеты, а ПК1 отвечает на них (рис. 8.32).

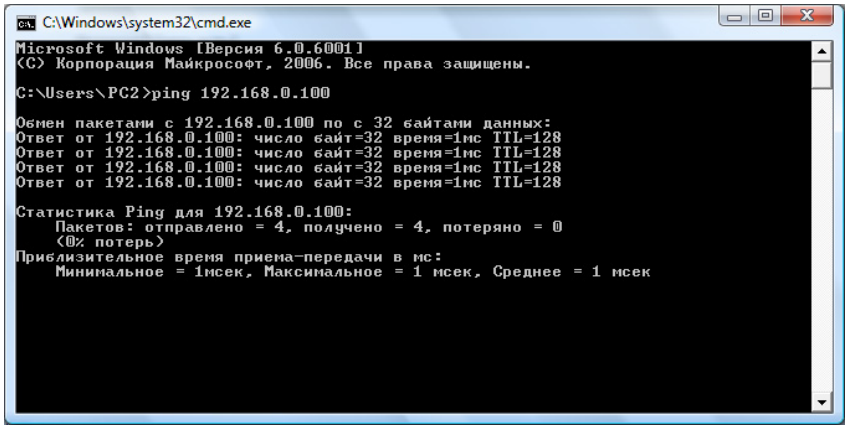


Рис. 8.32. Проверка связи между ПК

Необходимо теперь сесть за ПК1 и пропинговать ПК2. Также откройте командный интерпретатор и используйте команду ping, но вот адрес компьютера будет уже 192.168.0.6, т.е. в командном интерпретаторе нужно будет набрать ping 192.168.0.6.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Краткое описание способа соединения двух компьютеров с помощью витой пары.
4. Выводы по работе.

6. Контрольные вопросы

1. Опишите способ соединения двух компьютеров с помощью витой пары.

2. Чем различаются схемы обжимки кабеля?
3. Для чего необходимо «пинговать» ПК?
4. Для чего необходимо присваивать имена и адреса компьютерам?

7. Литература

1. *Олифер, В. Г.* Основы компьютерных сетей / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2009. – 350 с. – (Учебное пособие). – ISBN 978-5-49807-218-0.
2. Обжим сетевого кабеля // OBZHIMKABELJA.RU: рук. по монтажу ЛВС. 2009. URL: http://obzhimkabelja.ru/ (дата обращения: 15.02.2012).
3. *Кушнир, А. Н.* Новейшая энциклопедия компьютера / А. Н. Кушнир.– М. : Эксмо, 2008. – 975 с. – (Новейшая энциклопедия). – ISBN 978-5-699-24136-1.

Лабораторная работа № 13. *МОНТАЖ И НАСТРОЙКА БЕСПРОВОДНОЙ СЕТИ WI-FI*

1. **Цель работы**

Исследовать топологии фрагментов Internet с использованием утилиты traceroute.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3.

3. Краткие теоретические сведения

В современном мире все большее применение находят беспроводные сети Wi-Fi, позволяющие давать клиентам доступ к ресурсам сетей, например к Internet, с ноутбука или персонального компьютера, используя в качестве среды передачи данных радиоканал, что не требует наличия специальных проводных соединений клиентов с сетью, обеспечивая таким образом их мобильность.

Wi-Fi (англ. Wireless Fidelity — «беспроводная точность») — торговая марка Wi-Fi Alliance для беспроводных сетей на базе стандарта IEEE 802.11.

Схема Wi-Fi сети содержит не менее одной точки доступа (так называемый режим infrastructure) и не менее одного клиента. Также возможно подключение двух клиентов в режиме точка-точка, когда точка доступа не используется, а клиенты соединяются посредством сетевых адаптеров «напрямую». Точка доступа передаёт свой идентификатор сети (SSID) с помощью специальных сигнальных пакетов на скорости 0.1 Мбит/с каждые 100 мс. Поэтому 0.1 Мбит/с – наименьшая скорость передачи данных для Wi-Fi. Зная SSID сети, клиент может выяснить, возможно ли подключение к данной точке доступа. При попадании в зону действия двух точек доступа с идентичными SSID приёмник может выбирать между ними на основании данных об уровне сигнала. Стандарт Wi-Fi даёт клиенту полную свободу при выборе критериев для соединения (рис. 8.33).

Рис. 8.33. Схема сети Wi-Fi

Преимущества Wi-Fi:

* Позволяет развернуть сеть без прокладки кабеля, может уменьшить стоимость развёртывания и расширения сети. Места, где нельзя проложить кабель, например вне помещений и в зданиях, имеющих историческую ценность, могут обслуживаться беспроводными сетями.

* Wi-Fi-устройства широко распространены на рынке, а устройства разных производителей могут взаимодействовать на базовом уровне сервисов.

* Wi-Fi-сети поддерживают роуминг, поэтому клиентская станция может перемещаться в пространстве, переходя от одной точки доступа к другой.

* Wi-Fi – это набор глобальных стандартов. В отличие от сотовых телефонов Wi-Fi-оборудование может работать в разных странах по всему миру.

Недостатки Wi-Fi:

* Частотный диапазон и эксплуатационные ограничения в различных странах неодинаковы; во многих европейских странах разрешены два дополнительных канала, которые запрещены в США; в Японии есть ещё один канал в верхней части диапазона, а другие страны, например Испания, запрещают использование низкочастотных каналов. Более того, некоторые страны, например Италия, требуют регистрации всех Wi-Fi-сетей, работающих вне помещений, или требуют регистрации Wi-Fi-оператора.

* Довольно высокое по сравнению с другими стандартами потребление энергии, что уменьшает время жизни батарей и повышает температуру устройства.

* Самый популярный стандарт шифрования WEP может быть относительно легко взломан даже при правильной конфигурации (из-за слабой стойкости ключа). Несмотря на то что новые устройства поддерживают более совершенный протокол шифрования данных WPA, многие старые точки доступа не поддерживают его и требуют замены. Принятие стандарта IEEE 802.11i (WPA2) в июне 2004 года сделало доступной более безопасную схему, которая доступна в новом оборудовании. Обе схемы требуют более стойкий пароль, чем те, которые обычно назначаются пользователями. Многие организации используют дополнительное шифрование (например VPN) для защиты от вторжения.

* Wi-Fi имеют ограниченный радиус действия. Типичный домашний Wi-Fi-маршрутизатор стандарта 802.11b или 802.11g имеет радиус действия 45 м в помещении и 90 м снаружи. Микроволновка или зеркало, расположенные между устройствами Wi-Fi, ослабляют уровень сигнала. Расстояние зависит также от частоты.

* Наложение сигналов закрытой или использующей шифрование точки доступа и открытой точки доступа, работающих на одном или соседних каналах, может помешать доступу к открытой точке доступа. Эта проблема может возникнуть при большой плотности точек доступа, например в больших многоквартирных домах, где многие жильцы ставят свои точки доступа Wi-Fi.

* Неполная совместимость между устройствами разных производителей или неполное соответствие стандарту может привести к ограничению возможностей соединения или уменьшению скорости.

* Во время дождя работает плохо.

4. Ход лабораторной работы

1. Монтаж сети:

- Подключите адаптер Wi-Fi к USB-порту ноутбука № 2.
- Включите ноутбуки. После загрузки операционной системы на ноутбуках на обоих адаптерах должны загореться сигнальные лампочки, свидетельствующие об установке радиообмена между адаптерами и точкой доступа.
- Сеть собрана, теперь ее необходимо настроить.

2. Настройка сети

Настройка сети заключается в установке протоколов ноутбука клиента, которые необходимы для его работы, а также включении и настройки DHCP-сервера, который находится в точке.

На ноутбуке-сервере протокол TCP/IP уже установлен, нам осталось установить и настроить этот протокол на ноутбуке-клиенте. Все пункты настройки должны выполняться в той последовательности, в которой они указаны. Не нарушайте последовательность настройки.

На ноутбуке № 2 выполните следующие действия:

3. Щелкните правой клавишей мыши на значке «Мое сетевое окружение», выберите в меню «Свойства». Откроется список сетевых подключений (рис. 8.34).

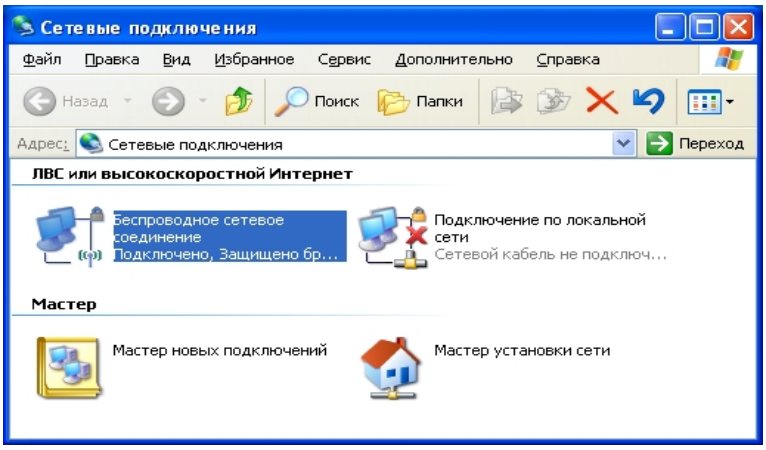


Рис. 8.34. Сетевые подключения

2. Выберите в списке «Беспроводное сетевое соединение», щелкните по нему правой клавишей мыши и выберите пункт «Свойства»). Откроется окно свойств соединения (рис. 8.35).

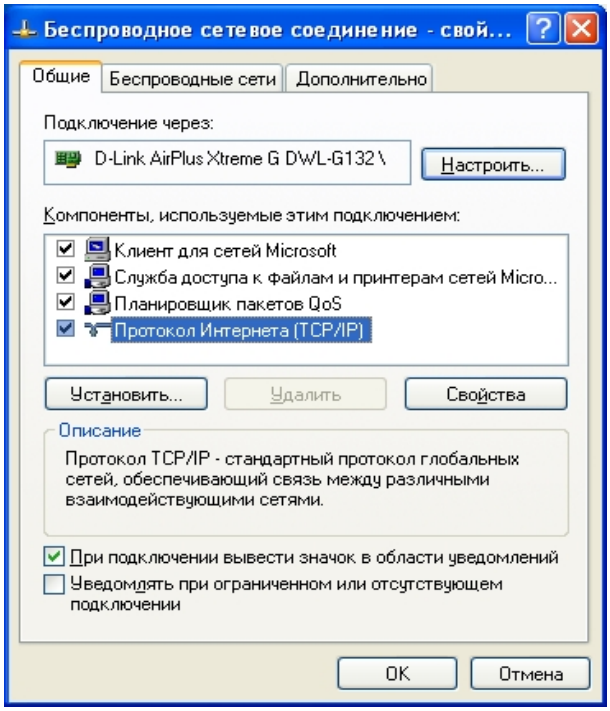


Рис. 8.35. Свойства соединения

3. В появившемся окне выберите «Протокол Интернета (TCP/IP)», нажмите «Свойства». Откроется окно настроек протокола. Активируйте флажок «Использовать следующий IP-адрес». Введите в поля IP-адрес и Маска подсети адреса установок (рис. 8.36).

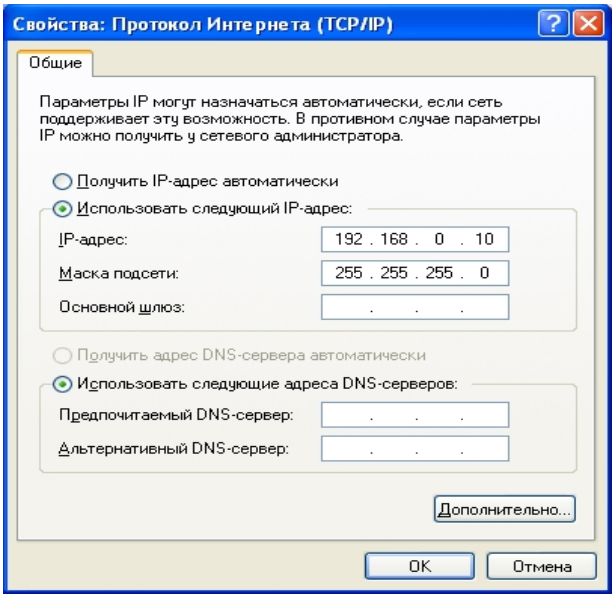


Рис. 8.36. Установка адреса

192.168.0.10 – это IP-адрес компьютера в сети. 255.255.255.0 – маска подсети. Это специальный параметр, который вместе с адресом однозначно определяет сеть, в которой находится компьютер.

После ввода настроек нажмите «ОК», окно «Свойства: Протокол Интернета (TCP/IP)» закроется. В окне «Беспроводное сетевое соединение» нажмите «ОК».

Для ноутбука прописан статический IP-адрес, это означает, что мы присвоили ноутбуку выделенный постоянный IP-адрес и прочие настройки, которые можно менять и назначать только вручную. Статический IP-адрес нам необходим для того, чтобы подключиться к точке доступа Wi-Fi и чтобы другие компьютеры в сети могли с ним связываться.

Для того чтобы начала функционировать сеть Wi-Fi, необходимо настроить точку доступа.

3. Настройка точки доступа Wi-Fi и DHCP-сервера

Загрузите обозреватель Internet Explorer. Введите в его адресной строке адрес: http://192.168.0.50/. Это IP-адрес точки доступа Wi-Fi. По этому адресу расположена система ее конфигурации. Вход в систему конфигурации защищен логином и паролем, и на экране появится окно для ввода этих данных. Введите Пользователь – admin, Пароль – 12345678 и нажмите кнопку «OK».

Откроется главная страница системы конфигурации точки доступа Wi-Fi:

- Нажмите на кнопку «Advanced». Откроется страница расширенных настроек точки доступа.
- Нажмите на кнопку «DHCP Server». Откроется страница для изменения настроек DHCP-сервера.

Установите следующие параметры DHCP либо измените существующие, если они не совпадают с указанными:

- Function Enable / Disable – Enabled
- IP Assigned From – 192.168.0.51
- The Range Of Pool (1-255) – 200
- SubMask – 255.255.255.0
- lease Time (60 – 31536000 sec) – 10000000
- Status – ON

Сохраните сделанные настройки. Точка доступа Wi-Fi уйдет на перезагрузку, которая занимает примерно полминуты.

Выполненные выше настройки обеспечивают следующие функции:

- Function Enable / Disable – Включает (Enabled) или отключает (Disabled) DHCP-сервер.
- IP Assigned From – задает начальный IP-адрес, с которого начинается диапазон IP-адресов, выделяемых динамически пользователям (пользователи, которые подключаются временно).
- The Range of Pool – задает конец диапазона IP-адресов, конечное значение последней цифры IP-адреса.
- Таким образом в нашем примере мы задали диапазон IP-адресов от 192.168.0.51 до 192.168.0.200 включительно.
- SubMask – маска подсети. Это специальный параметр, который вместе с адресом однозначно определяет сеть, в которой находится компьютер.

- Lease Time – время «жизни» выделенных пользователю сетевых настроек. При динамической адресации настройки пользователя существуют определенное время, после чего сбрасываются и программное обеспечение пользователя запрашивает новые настройки. Здесь задается время существования выделенных пользователю настроек (в секундах).
- Status – специальный параметр, он ставится в значение ON, если в сети используется совместно динамическая и статическая адресации. В нашем случае этот параметр установлен в ON, поскольку на ноутбуке клиента прописан статический постоянный адрес.

4. Проверка работы беспроводной сети.

После того как сеть настроена, нужно проверить ее работу и убедиться, что компьютеры могут обмениваться данными между собой. В сети могут существовать самые разные службы и сервисы, каждый из которых выполняет свои задачи. В сети, которую мы настроили, работают две службы: локальный WEB-сервер, предназначенный для размещения HTML-страниц в сети, и Сеть Microsoft, посредством которой производятся обмен файлами и совместная работа с клиентами.

Сначала проверим работу WEB-сервера. WEB-сервер установлен на ноутбуке-сервере. Для того чтобы проверить работу WEB-сервера, запустите на ноутбуке № 2 (компьютер Клиент) обозреватель Интернета Internet Explorer и в его адресной строке введите http://192.168.0.3/wifi/

Если страница загрузится, действуйте в соответствии с указаниями, написанными на этой странице.

Статическая IP-адресация имеет следующие недостатки:

- Для того чтобы узнать все настройки сети, необходимо обратиться к администратору сети, который должен индивидуально выделить для каждого клиента свой уникальный IP-адрес. Это неудобно как для клиента, так и для администратора.
- При подключении к какой-либо другой беспроводной сети настройки компьютера клиента приходится снова изменять под новую сеть, узнавая их у администратора.
- Если случайно ваши настройки совпадут с настройками другого клиента, вы не сможете подключиться к сети [4].

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Описание настройки сети Wi-FI.
4. Выводы по работе.

6. Контрольные вопросы

1. Каковы достоинства сети Wi-Fi?
2. Каковы недостатки сети WiFi?
3. Дайте характеристики сети Wi-Fi.
4. Опишите недостатки статической IP-адресации.

7. Литература

1. Монтаж и настройка беспроводной сети Wi-Fi // RUDOCS. EXDAT.COM : рук. по монтажу беспроводной сети Wi-Fi. 2011. URL: http://rudocs.exdat.com/docs/index-43009.html (дата обращения: 20.02.2012).
2. *Леонтьев, В.П.* «Новейшая энциклопедия персонального компьютера» / В.П. Леонтьев. – М: Олма Медиа Групп, 2007. – 734 с. – ISBN 978-5-373-00483-1.
3. *Мюллер, С.* Модернизация и ремонт ПК = Upgrading and Repairing PCs : пер. с англ. / С. Мюллер. – 17-е изд. – М. : Вильямс, 2007. – 1489 с. – ISBN 978-5-8459-1126-1.

Лабораторная работа № 14. *СОЗДАНИЕ МАКЕТА САЙТА С ФРЕЙМОВОЙ СТРУКТУРОЙ*

1. Цель работы

Создать макет веб-сайта с фреймовой структурой.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3, MS FrontPage 2007.

3. Краткие теоретические сведения

HTML (от англ. HyperText Markup Language — «язык разметки

гипертекста») — стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создается при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузером и отображается в виде документа в удобной для человека форме.

HTML является приложением («частным случаем») SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879. XHTML же является приложением XML.

Текстовые документы, содержащие код на языке HTML (такие документы традиционно имеют расширение .html или .htm), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые браузерами или интернет-обозревателями, обычно предоставляют пользователю удобный интерфейс для запроса веб-страниц, их просмотра (и вывода на иные внешние устройства) и при необходимости отправки введённых пользователем данных на сервер. Наиболее популярными на сегодняшний день браузерами являются Internet Explorer, Mozilla Firefox, Safari, Google Chrome и Opera.

HTML — теговый язык разметки документов. Любой документ на языке HTML представляет собой набор элементов, причём начало и конец каждого элемента обозначаются специальными пометками – тегами. Элементы могут быть пустыми, то есть не содержащими никакого текста и других данных (например, тег перевода строки
). В этом случае обычно не указывается закрывающий тег. Кроме того, элементы могут иметь атрибуты, определяющие какие-либо их свойства (например, размер шрифта для элемента font). Атрибуты указываются в открывающем теге.

Фрейм (frame) — это отдельный, законченный HTML-документ, который вместе с другими HTML-документами может быть отображён в окне web-браузера.

Фреймы по своей сути очень похожи на ячейки таблицы, однако более универсальны. Фреймы разбивают web-страницу на отдельные мини-кадры, расположенные на одном экране, которые являются независимыми друг от друга. Каждое окно может иметь собственный адрес. При нажатии на любую из ссылок, расположенных в одном фрейме, вы можете рассматривать страницы, показанные в другом окне.

Фреймы довольно часто использовались для навигации по web-сайту. При этом навигационная страница располагается в одном окне, а страницы с текстом - в другом.

В современном WWW использование фреймов не рекомендовано. Главным образом это связано, как ни странно, с алгоритмами работы поисковых машин, которые могут привести пользователя к html-документу, являющемуся согласно задумке лишь одним из фреймов того, что автору сайта хотелось бы представить. Данный недостаток фреймов успешно устраняется средствами JavaScript.

4. Ход лабораторной работы

1. Создайте папку сайта с именем **site_l.**
2. Откройте редактор Microsoft FrontPage.
3. Создайте сайт на основе папки **site_l:**

   а) выполните команду меню **Файл, Создать, Страница или Веб-узел;**

   б) в панели **Создание веб-страницы или узла** в разделе **Создание** выберите **Пустой веб-узел** (рис. 8.37);

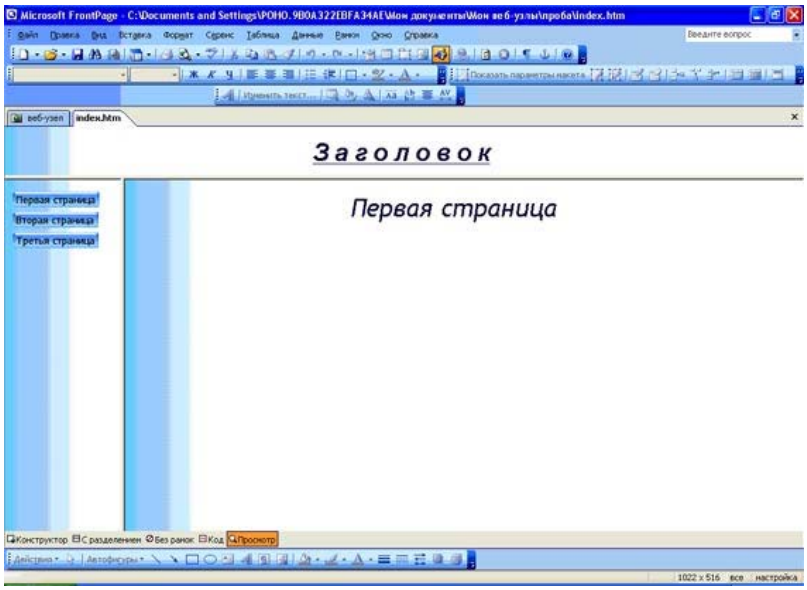


Рис. 8.37. Образец главной страницы сайта

в) в окне **Шаблоны веб-узлов** в разделе **Параметры** укажите расположение нового веб-узла (используя экранную кнопку **Обзор,** выделите папку **site_l)** (рис. 8.38, 8.39);

г) щелкните на кнопках **Открыть, ОК.**

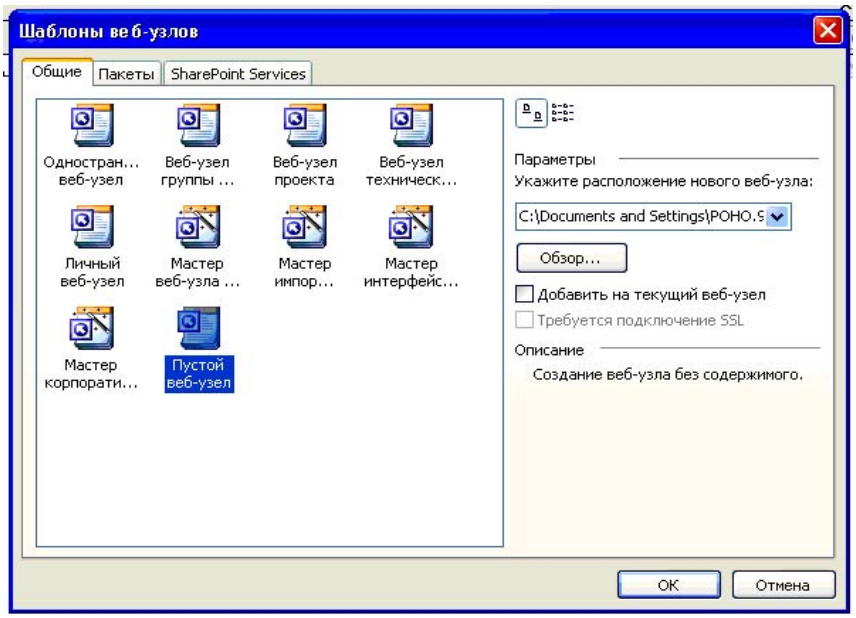


Рис .8.38. Вид окна Шаблоны веб-узлов

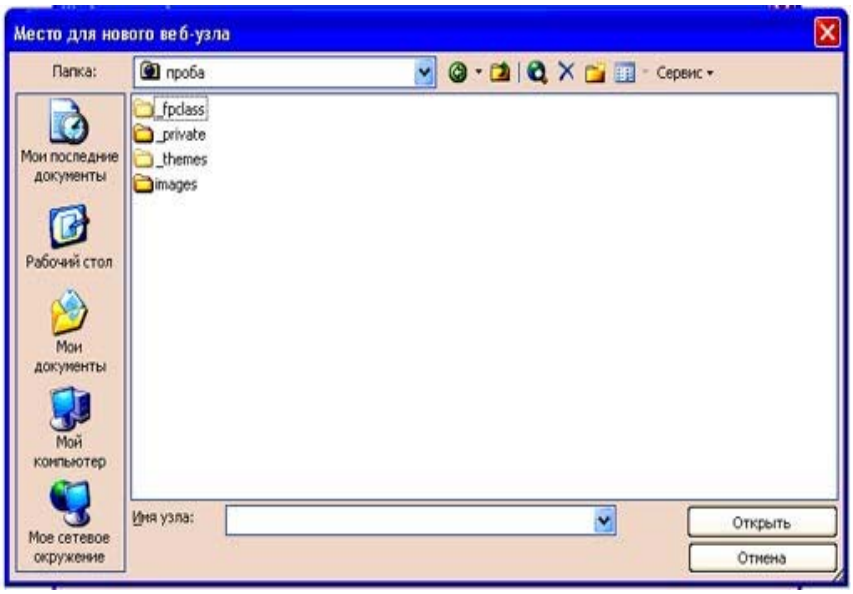


Рис. 8.39. Выбор папки для создания веб-узла

4. Выполните настройку кодировки языка сайта: меню **Сервис, Параметры узла.**

В окне **Параметры узла** (рис. 8.40**)** перейдите на вкладку **Язык** и выберите следующие установки:

- язык сообщений сервера: **русский;**
- применяемый по умолчанию набор знаков для страниц - **кириллица;**
- игнорировать раскладку при выборе кодировки новых страниц.

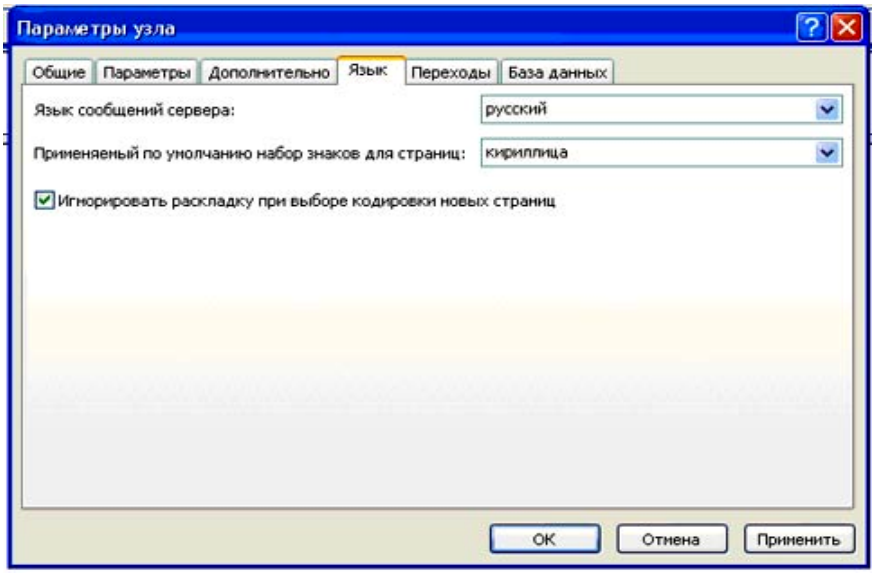


Рис. 8.40. Настройка кодировки языка сайта в окне Параметры узла

5. Создайте страницы сайта с именами: *zagolov.htm, menu.htm, l.htm, 2.htm, 3.htm.* (Информационное наполнение страниц в данной лабораторной работе опускается и оставляется на последующую доработку созданного макета сайта). С этой целью выполните следующие операции:

а) создайте новую страницу, щелкнув на кнопке **Новая страница;**

б) щелкнув правой кнопкой в области поля страницы, выберите в контекстное меню команду **Свойства страницы** (рис. 8.41);

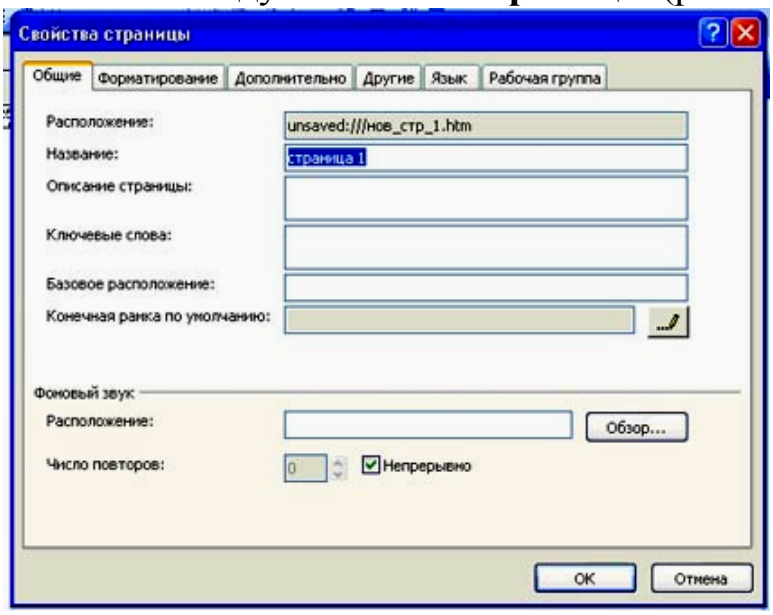


Рис. 8.41. Задание параметров страницы в окне Свойства страницы

в) в окне **Свойства** страницы на вкладке **Общие** введите текст заголовка страницы, перейдите на вкладку **Язык** и выберите следующие установки (рис. 8.42).

• пометить текущий документ, указав: **русский;**

• сохранить документ, используя: **кириллица;**

• повторить загрузку текущего документа, используя: **кириллица;**

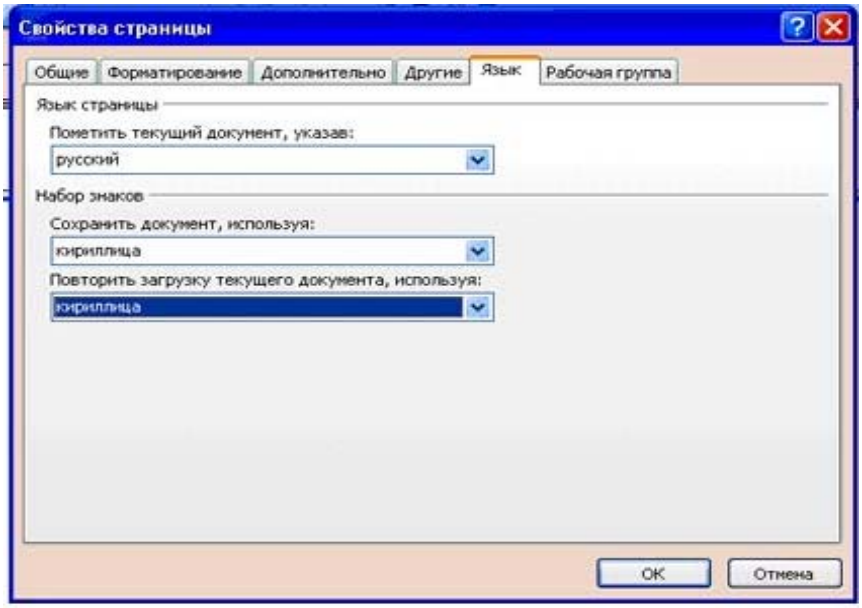


Рис. 8.42. Установка языка страницы и набора знаков

г) примените к страницам *zagolov.htm, menu.htm* тему для графического оформления: меню **Формат, Тема, Перетекание** (рис. 8.43);

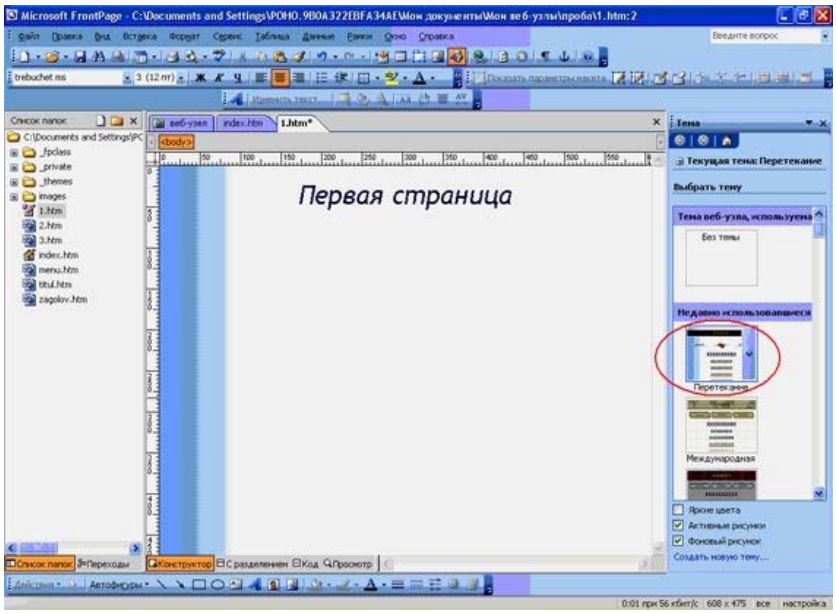


Рис. 8.43. Выбор темы страницы

д) введите в область страницы соответствующий текстовый заголовок;

е) сохраните страницу в папке сайта: меню **Файл, Сохранить как,** введите имя страницы;

ж) аналогично оформите и остальные веб-страницы (рис. 8.44).

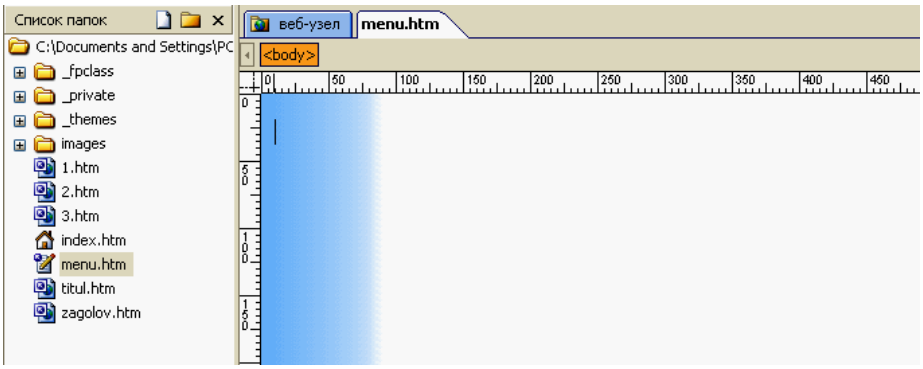


Рис. 8.44. Образцы заготовок страниц

6. Создайте главную страницу сайта с фреймовой структурой:

а) выполните команду меню **Файл, Создать, Страница или Веб-узел**;

б) в панели **Создание веб-страницы или узла** в разделе **Создание с помощью шаблона** выберите **Шаблоны страниц** (рис. 8.45);

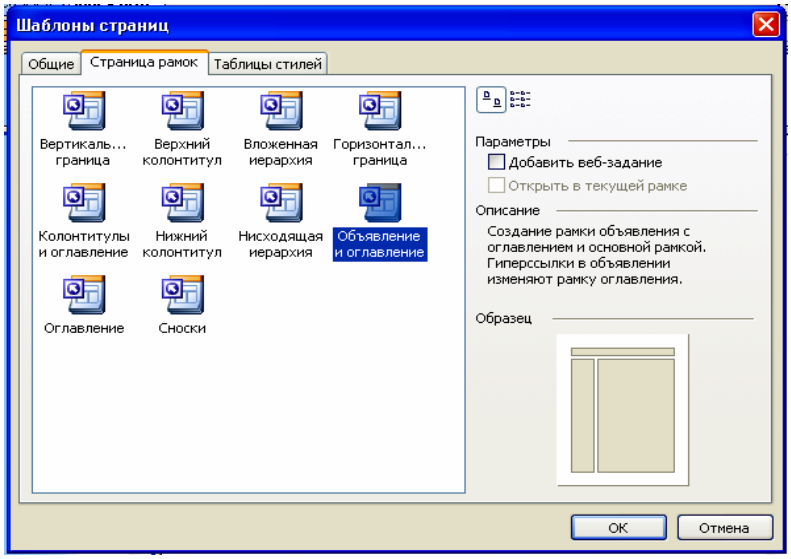


Рис. 8.45. Окно Шаблоны страниц

в) в открывшемся окне Шаблоны страниц перейдите на вкладку Страница рамок и выберите тип фреймовой структуры Объявление и оглавление;

г) в окне отображения макета страницы (рис. 8.46) щелкните на кнопке Задать начальную страницу в заголовочном фрейме; в списке файлов сайта выделите страницу *zagolov.htm* и щелкните на кнопке ОК (рис. 8.47);

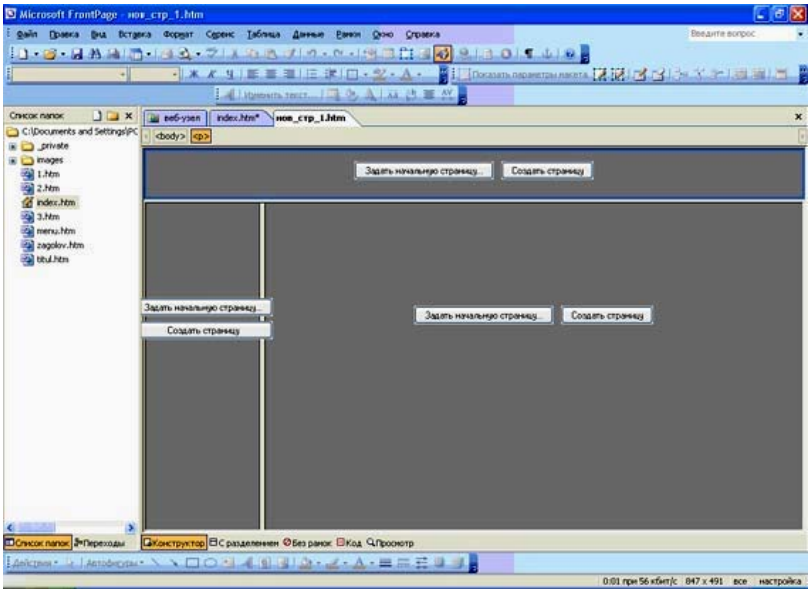


Рис. 8.46. Созданный редактором макет страницы

Рис. 8.47. Задание страницы, отображаемой во фрейме

д) в левом фрейме щелкните на кнопке Задать начальную страницу, в списке файлов сайта выделите страницу *menu.htm* и щелкните на кнопке ОК. Страница *menu.htm,* открытая в левом фрейме, пока не содержит кнопки перехода на другие веб-страницы;

е) в правом фрейме щелкните на кнопке Задать начальную страницу, в списке файлов сайта выделите страницу *l.htm* и щелкните на кнопке ОК.

Результат отражен на рис. 8.48;

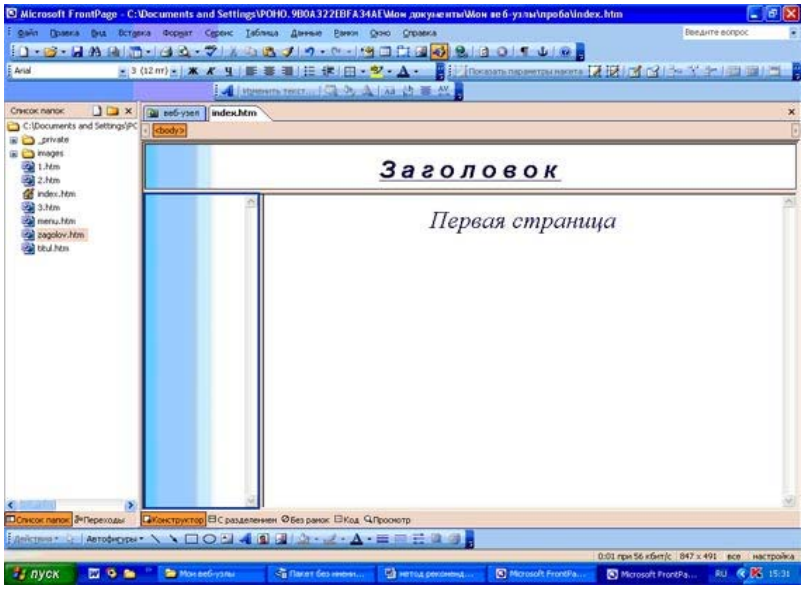


Рис. 8.48. Образец страницы index.htm

ж) сохраните главную страницу сайта: меню Файл, Сохранить как, введите имя страницы *index.htm.*

7. Если Вы хотите изменить вид границ фреймов, то, щелкая правой кнопкой по отдельным фреймам и выбирая в контекстном меню команду Свойства рамки, можно задать то или иное отображение границ фрейма на экране (рис. 8.49).

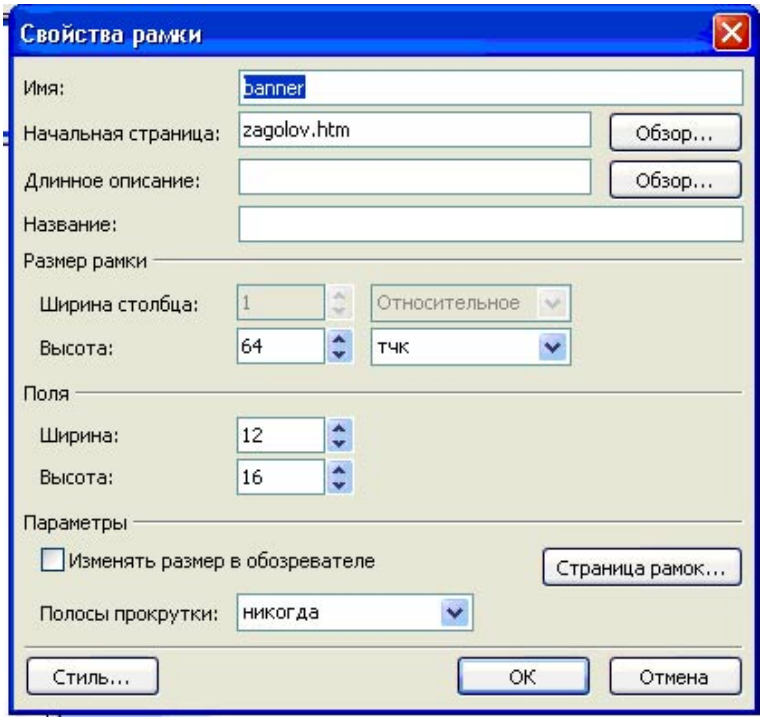


Рис. 8.49. Задание параметров фрейма в окне Свойства рамки

8. Продолжите работу над страницей *menu.htm* из страницы *index.htm*. Разместите кнопки с динамическими эффектами, осуществляющие навигацию по страницам сайта:

а) щелкните в области левого фрейма;

б) выполните команду меню Вставка, Веб-компонент, Динамические эффекты, Интерактивная кнопка (рис. 8.50), щелкните на кнопке Готово (или команда Вставка, Меняющаяся кнопка);

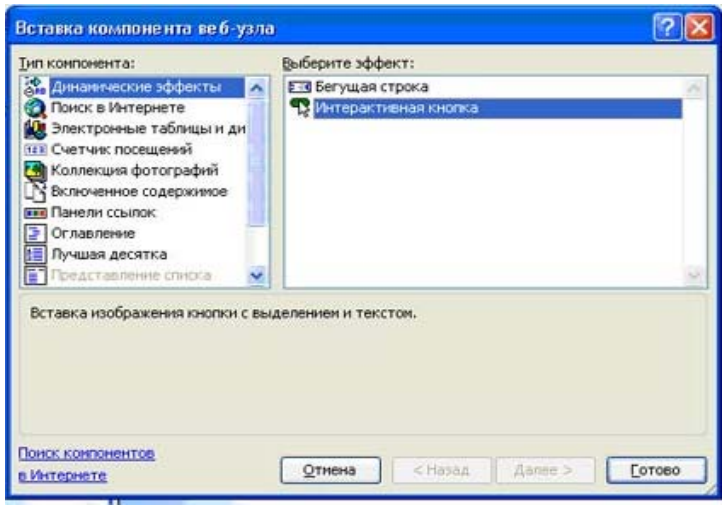


Рис. 8.50. Вставка компонента веб-узла

в) в окне **Свойства меняющейся кнопки** выполните следующие действия:

• введите текст надписи на кнопке (используя кнопку **Шрифт,** можно изменить стиль надписи) (рис. 8.51);

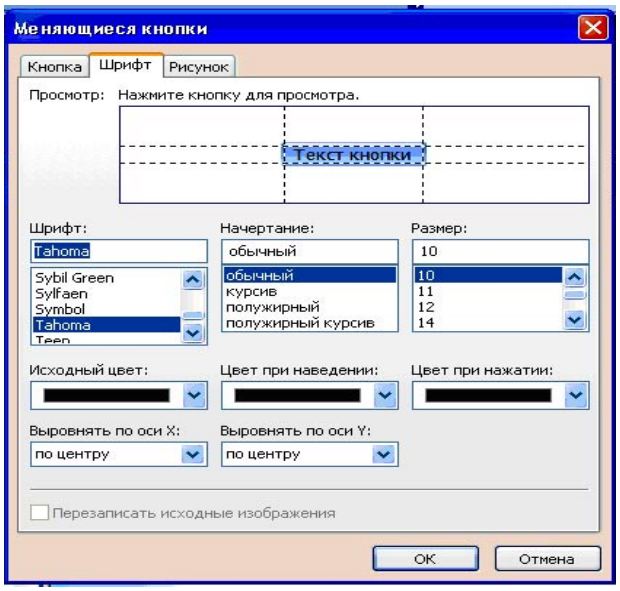


Рис. 8.51. Задание параметров меняющейся кнопки

• укажите файл перехода и фрейм, в котором будет открыт указанный файл, т.е. установите текстовый курсор в поле Ссылка на:, щелкните на кнопке **Обзор** и в окне **Выбор гиперссылки для меняющейся кнопки** выделите имя страницы перехода в списке файлов страниц (рис. 8.52);

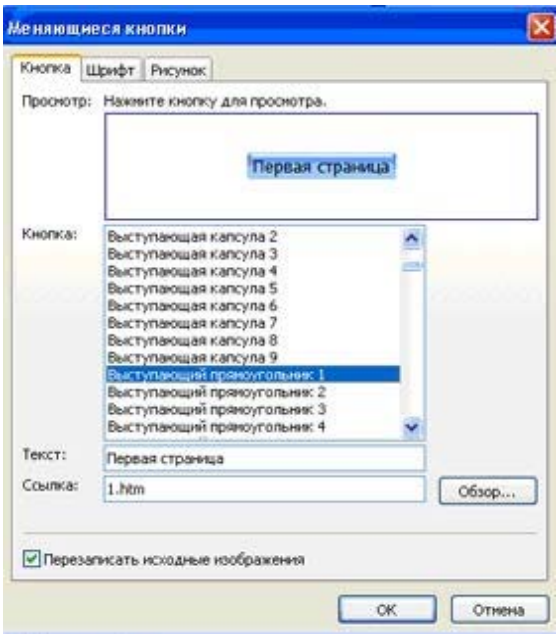


Рис. 8.52. Выбор гиперссылки для меняющейся кнопки

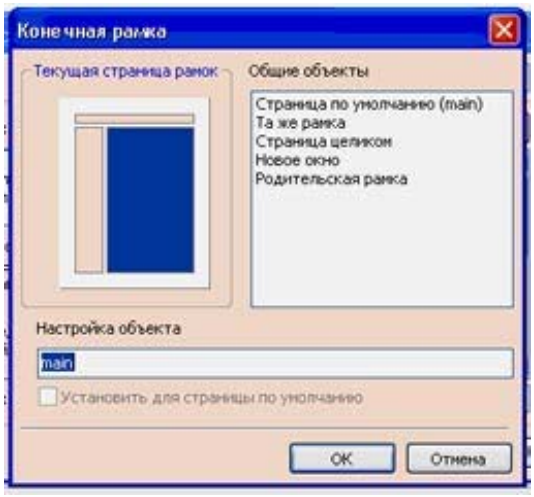


Рис. 8.53. Указание фрейма для открытия файла перехода

• в окне **Выбор гиперссылки для меняющейся кнопки** щелкните на кнопке **Выбор рамки** и укажите фрейм, в котором будет открываться страница (рис. 8.53); для этого выделите соответствующий фрейм щелчком мыши в разделе **Текущая страница рамок** и подтвердите выбор щелчками на экранных кнопках ОК;

• сохраните страницу *index.htm* на диске. Страница *menu.htm,* открытая в левом фрейме, содержит кнопки переходов по страницам сайта, которые будут открываться в правом фрейме (рис. 8.54).

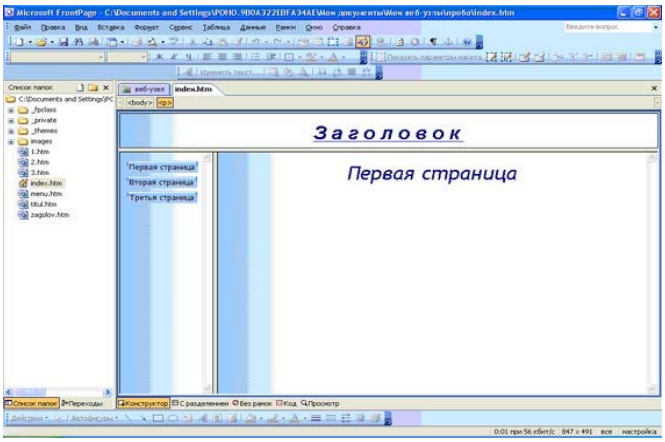


Рис. 8.54. Образец страницы index.htm

9. Просмотрите сайт в браузере, щелкнув на кнопке  **Просмотр в браузере** [4].

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Краткое описание языка HTML, порядок создания макета сайта.
4. Выводы по работе.

6. Контрольные вопросы

1. Дайте определение понятия HTML.
2. Какие существуют программы для создания сайтов кроме MS FrontPage?
3. Для чего используются фреймы?

7. Литература

1. Создания сайта в редакторе MS FrontPage 2003 // IVRONO. NAROD.RU: Отдел образования администрации Ивнянского района. 2005. URL: http://invrono.narod.ru/Metodish/ sait.doc (дата обращения: 12.02.2012).
2. *Омельченко, Л.Н.* Самоучитель Microsoft FrontPage 2002 / Л.Н. Омельченко, А. Ф. Федоров. – СПб. : БХВ-Петербург, 2001. – 564 с. – (Самоучитель). – ISBN 5-94157-103-8.
3. *Березин, С.В.* Ваш выход в Интернет. Секреты эффективной и безопасной работы / С.В. Березин. – СПб. : БХВ-Петербург, 2004. – 590 с. – (Самоучитель). – ISBN 5-94157-332-4.

Лабораторная работа № 15. *ПРИМЕНЕНИЕ ПАНЕЛИ ГИПЕРССЫЛОК ДЛЯ ОСУЩЕСТВЛЕНИЯ НАВИГАЦИИ ПО СТРАНИЦАМ САЙТА*

1. Цель работы

Применить в созданном макете сайта гиперссылки.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3, MS FrontPage 2007.

3. Краткие теоретические сведения

Гиперссылка (англ. hyperlink) в компьютерной терминологии — часть гипертекстового документа, ссылающаяся на другой элемент (команда, текст, заголовок, примечание, изображение) в самом документе, на другой объект (файл, директория, приложение), расположенный на локальном компьютере или в компьютерной сети, либо на элементы этого объекта.

Гиперссылка может быть добавлена к любому элементу гипертекстового документа и обычно выделяется графически. В HTML-документах текстовые ссылки по умолчанию выделяются синим цветом, при наведении на них курсором мыши в окне браузера изменяются, например, меняют цвет или выделяются подчеркиванием. При навигации в браузере с помощью клавиатуры текстовые и графические ссылки выделяются прямоугольной пунктирной рамочкой. Посещенная ранее ссылка обычно выделяется цветом, отличным от цвета непосещенной ссылки.

«Битой» ссылкой называют такую гиперссылку, которая ссылается на отсутствующий по каким-либо причинам объект, например, если документ или файл удалены или перемещены администратором ресурса, на котором они были расположены, или если сам ресурс недоступен. Обычно в таком случае на странице появляется сообщение с кодом ошибки, но это происходит не всегда.

Гиперссылка — фрагмент HTML-документа и его базовый элемент:

- указывающий на другой файл, который может быть расположен в Интернет;
- содержащий полный путь (URL) к этому файлу.

Гиперссылка для пользователя – графическое изображение или текст на сайте в письме электронной почты или в каком-либо электронном документе, устанавливающие связь и позволяющие переходить к другим объектам Интернет.

Гипертéкст — термин, введённый Тедом Нельсоном в 1965 году для обозначения «текста, ветвящегося или выполняющего действия по запросу». Обычно гипертекст представляется набором текстов, содержащих узлы перехода между ними, которые позволяют избирать читаемые сведения или последовательность чтения. Общеизвестным и ярко выраженным примером гипертекста служат веб-страницы — документы HTML (язык разметки гипертекста), размещённые в Сети. В более широком понимании термина гипертекстом является любая повесть, словарь или энциклопедия, где встречаются отсылки к другим частям данного текста, имеющие отношение к данному термину. В компьютерной терминологии гипертекст – текст, сформированный с помощью языка разметки, потенциально содержащий в себе гиперссылки.

4. Ход лабораторной работы

1. Создайте папку сайта с именем site_2.
2. Откройте редактор Microsoft FrontPage.
3. Создайте сайт на основе папки site_2).

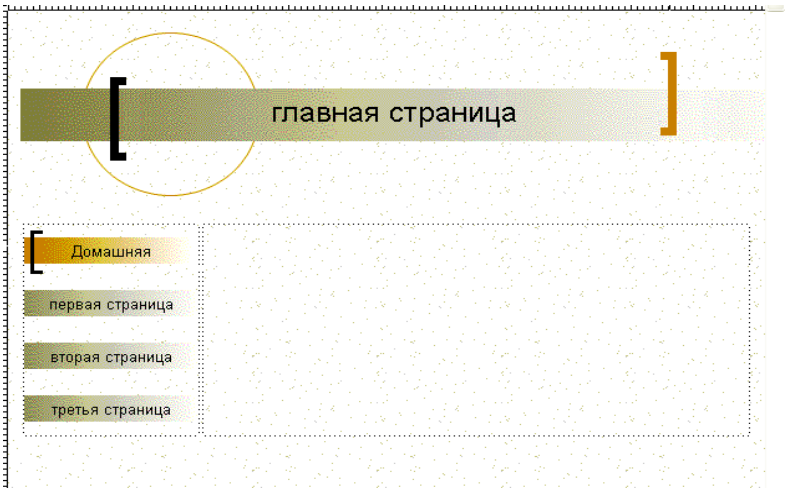


Рис. 8.55. Образец главной страницы сайта

4. Выполните настройку кодировки языка.
5. Создайте страницы сайта с именами: *index.htm, l.htm, 2.htm, 3.htm* (см. лабораторную работу 1, п. 5) (рис. 8.55).Примените к страницам другую тему для графического оформления (Формат, Тема, Romanesque).

6. Создайте иерархическую диаграмму переходов по страницам сайта:

а) на панели Представления щелкните на кнопке Переходы (рис. 8.56);

б) отобразите список папок, выполнив команду меню Вид, Список папок;

в) создайте иерархическую диаграмму переходов по страницам сайта методом перетаскивания веб-страницы в соответствующую позицию диаграммы (транспортируйте с помощью мыши пиктограмму веб-страницы из списка папок в поле диаграммы);

7. Поместите на каждой странице объявление (баннер):

а) отобразите страницу в режиме Страница;

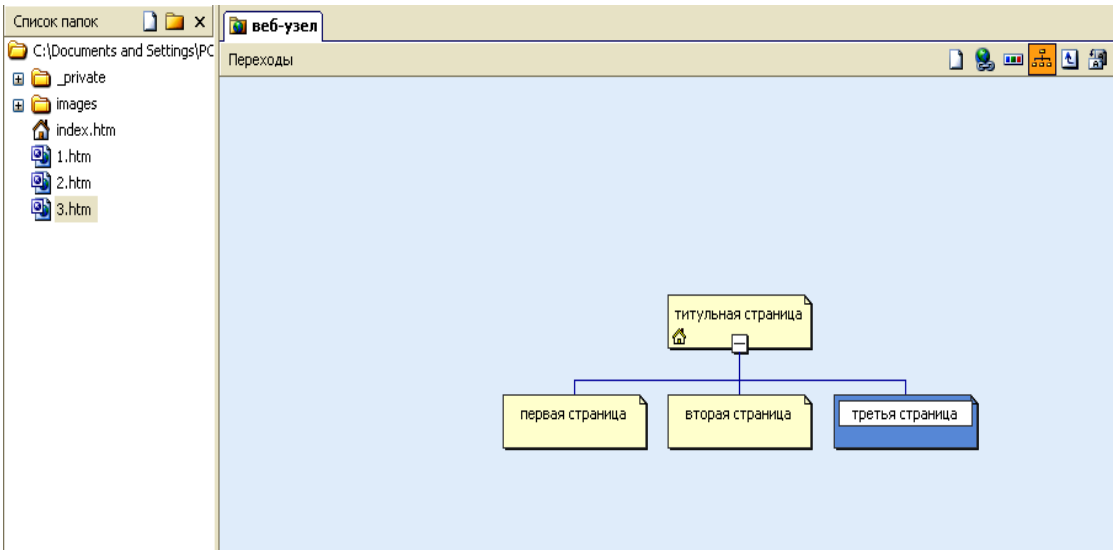


Рис. 8.56. Образец иерархической диаграммы переходов

б) выполните команду меню **Вставка, Объявление на странице;**

в) в окне **Свойства объявления на странице** (рис. 8.57) включите переключатель Рисунок, щелкните на кнопке ОК;

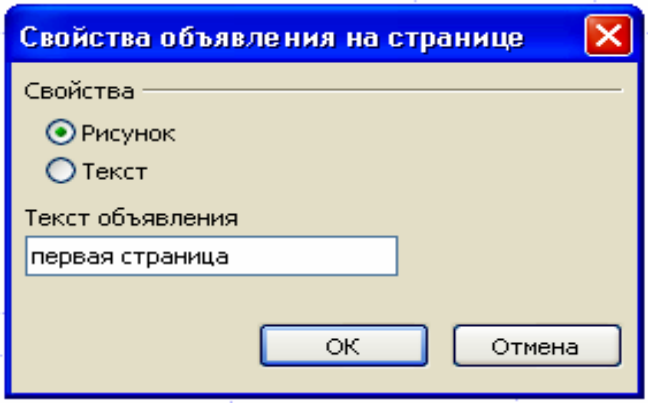


Рис. 8.57. Задание параметров объявления

г) в текстовое поле **Текст объявления** введите соответствующий текст объявления и щелкните на кнопке ОК;

д) выполните выравнивание объявления по центру страницы с помощью соответствующей кнопки на панели инструментов **Форматирование;**

е) сохраните страницу в папке сайта, выполнив команду меню **Файл, Сохранить.**

Внимание!

Для правильного отображения объявлений (баннеров) необходимо, чтобы:

- страница была включена в иерархическую диаграмму сайта;
- к странице была применена тема (иначе объявление будет отображаться в виде обычного текста).

Если текст объявления отображается неверно, следует выполнить следующие действия:

- щелкнуть правой кнопкой мыши на объявлении;
- в контекстном меню выбрать команду **Свойства объявления;**
- проверить текст;
- проверить, включена ли опция **Рисунок;**
- проверить, включена ли текущая страница в иерархическую диаграмму сайта.

8. Создайте на главной странице сайта *index.htm* под объявлением пустую таблицу из одной строки и двух столбцов для жесткого размещения панели навигации и информационного содержания страницы:

а) выполните команду меню **Таблица, Вставить, Таблица;** в окне **Вставка таблицы** укажите **количество строк: 1, количество столбцов: 2.**

б) определите следующие параметры таблицы (рис. 8.58): в группе Положение установите Выравнивание: по центру, в группе Границы установите Размер: О, для того чтобы границы таблицы были невидимыми;

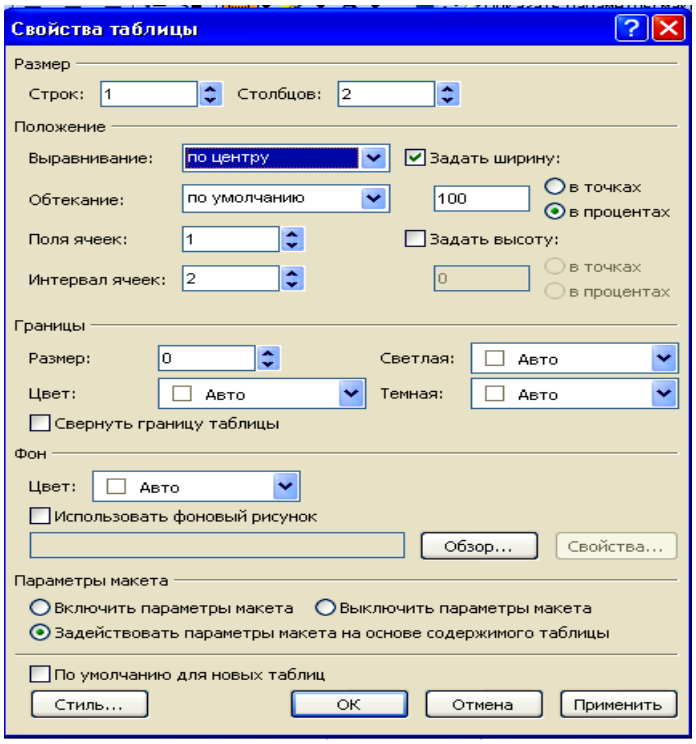


Рис. 8.58. Задание параметров таблицы

в) выполните выравнивание содержимого ячеек таблицы по левому и верхнему краям:

- выделите ячейки таблицы;
- щелкните правой кнопкой мыши на выделенной области и в контекстном меню выберите команду Свойства ячейки, в появившемся окне Свойства ячейки в списке Выровнять по горизонтали укажите по левому краю, в списке Выровнять по вертикали укажите по верхнему краю (рис. 8.59).

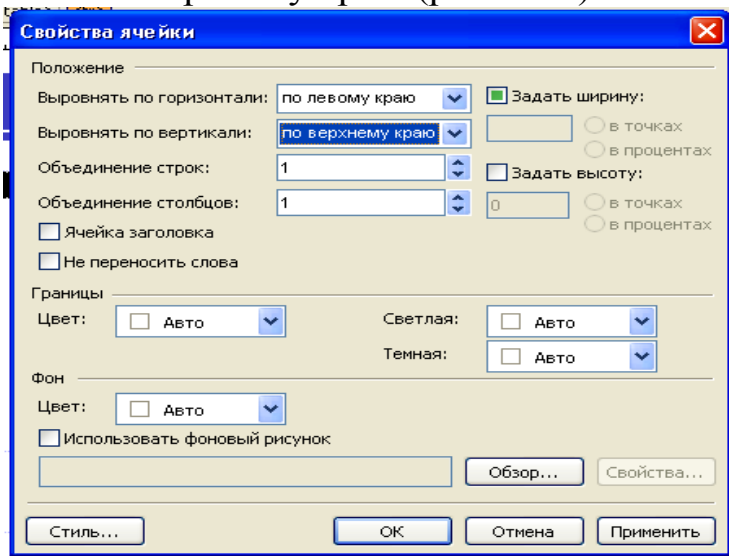


Рис. 8.59. Задание параметров ячеек таблицы в окне Свойства ячейки

9. Добавьте в левую ячейку таблицы панель ссылок (панели навигации работают при наличии на сервере FP Server Extensions!):

а) щелкните в левой ячейке таблицы;

б) выберите в меню Вставка, Веб-компонент, Панели ссылок;

в) в окне Вставка компонента веб-узла (рис. 8.60) выберите тип панели: Панель, основанная на структуре переходов, щелкните на кнопке Далее;

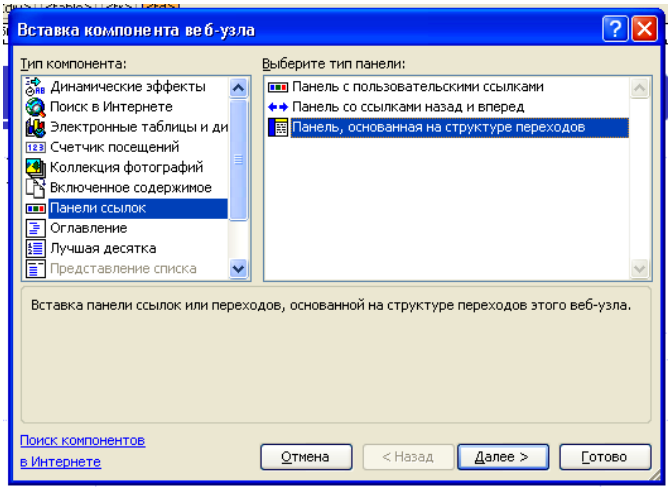


Рис. 8.60. Вставка панели переходов

г) в следующем окне диалога укажите стиль панели Использовать тему страницы (рис. 8,61), щелкните на кнопке Далее;

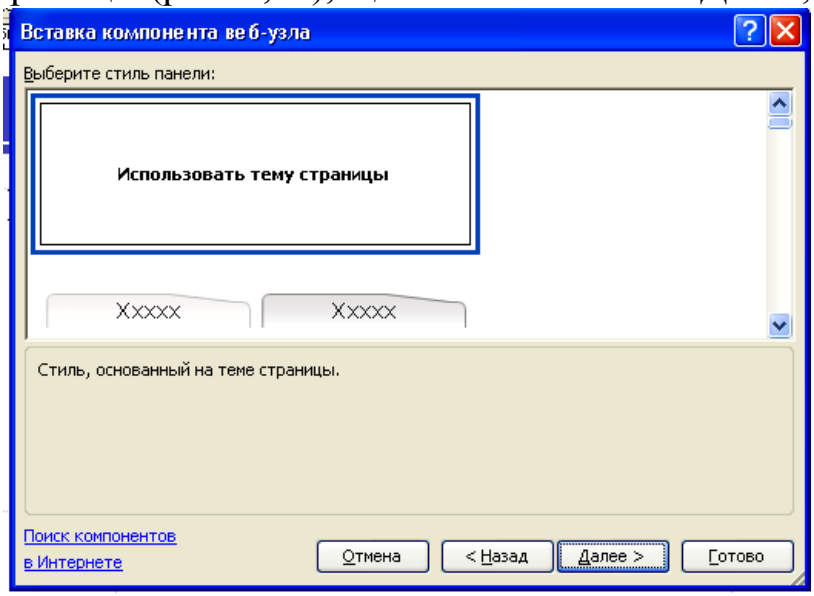


Рис. 8.61. Выбор стиля панели переходов в окне Вставка компонента веб-узла

д) выберите ориентацию панели **вертикальная** в диалоговом окне **Вставка компонента веб-узла,** щелкните на кнопке **Готово;**

е) в окне **Свойства панели ссылок** (рис. 8.62) **на** вкладке **Об-**

щие задайте режим связи между страницами сайта: включите переключатель **Дочерние страницы домашней,** установите флажок **домашняя страница;**

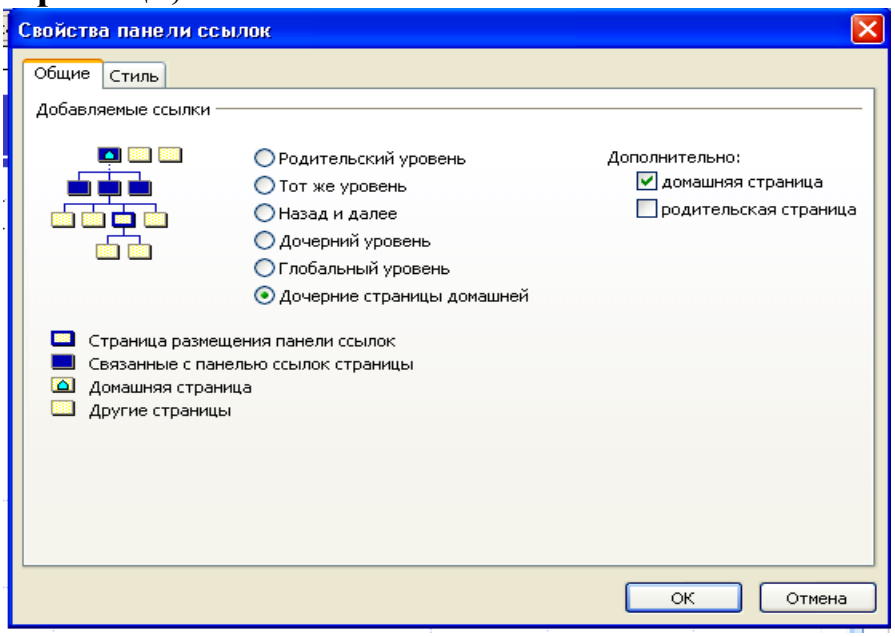


Рис. 8. 62. Задание режима связи между страницами сайта в окне Свойства панели ссылок

10. Выровняйте ширину левой ячейки таблицы по ширине панели навигации перетаскиванием правой границы ячейки.

11. Сохраните страницу под именем *index.htm* на диске.

12. Скопируйте созданную на странице *index.htm* таблицу с размещенной в левой ячейке панелью навигации на все страницы сайта.

Результатом должны быть следующие веб-страницы (рис. 8.63):

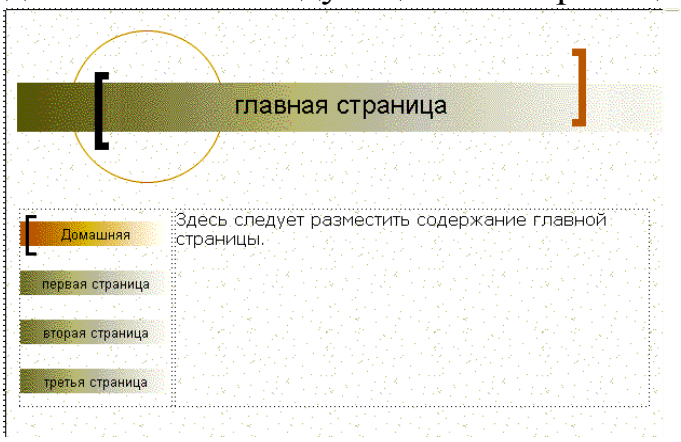


Рис. 8.63. Образец страниц сайта

13. Просмотрите сайт в браузере, щелкнув на кнопке **Просмотр в браузере** основного меню программы FrontPage [5].

14. Заполните шаблоны веб-страниц, входящих в веб-узел «Личный веб-узел»:

- Добро пожаловать!
- О себе
- Избранное
- Обратная связь
- Увлечения
- Коллекция фотографий.

15. Сохраните созданный сайт.

5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Краткое описание гиперссылок и гипертекста, порядок создания сайта.
4. Выводы по работе.

6. Контрольные вопросы

1. Дайте определение понятия «гиперссылка».
2. Для чего используются гиперссылки?
3. Какая гиперссылка называется «битой»?

7. Литература

1. Создание сайта в редакторе MS FrontPage 2003 // IVRONO. NAROD.RU: Отдел образования администрации Ивнянского района. 2005. URL: http://invrono.narod.ru/Metodish/ sait.doc (дата обращения: 12.02.2012).
2. *Омельченко, Л.Н.* Самоучитель Microsoft FrontPage 2002 / Л.Н. Омельченко, А. Ф. Федоров. – СПб. : БХВ-Петербург, 2001. – 564 с. – (Самоучитель). – ISBN 5-94157-103-8.
3. *Березин, С.В.* Ваш выход в Интернет. Секреты эффективной и безопасной работы / С.В. Березин. – СПб. : БХВ-Петербург, 2004. – 590 с. – (Самоучитель). – ISBN 5-94157-332-4.

Лабораторная работа № 16. *ПУБЛИКАЦИЯ ВЕБ-УЗЛА*

2. Цель работы

Опубликовать созданный веб-узел на хостинге narod.ru.

2. Приборы и материалы

ПК на базе Intel Core 2 Duo 2,3 ГГц, ОЗУ 2048 Mb, HDD Seagate 80Gb 7200 rpm, ОС Windows XP SP3, MS FrontPage 2007.

3. Краткие теоретические сведения

Веб-узел – это группа файлов, связанных между собой гиперссылками, которые позволяют переходить с одной страницы на другую с помощью щелчка кнопкой мыши. Обычно веб-узел состоит из нескольких веб-страниц, которые могут содержать рисунки, гиперссылки и более сложные элементы, такие как формы и базы данных.

Публикация веб-узла в общем случае представляет собой копирование всех файлов, составляющих веб-узел, в определенное место назначения.

Публикация узла обычно осуществляется, когда требуется: а) предоставить общий доступ для просмотра узла, в этом случае его копируют, как правило, на веб-сервер; б) создание резервной копии узла, в этом случае его копируют в определенную папку на диске.

Публикация веб-узла на веб-сервере связана с передачей большого количества файлов. Кроме того, очень часто возникает необходимость изменять файлы, удалять старые и добавлять новые. Поэтому в приложении FrontPage существует возможность выбора публикации всех или только измененных файлов веб-узла, удаление уже опубликованных файлов и публикация только отмеченных с помощью мыши файлов.

Процедура публикации состоит из нескольких шагов, которые последовательно описаны ниже.

Заходим на сайт narod.yandex.ru и в правой верхней части страницы обнаруживаем окно «Вход», где требуется вести Логин и Пароль. Это те самые логин и пароль, которые вы получили при регистрации почтового ящика. Если почтовый ящик у вас отсутствует или вы забыли его реквизиты, нужно пройти регистрацию (рис.8. 64).

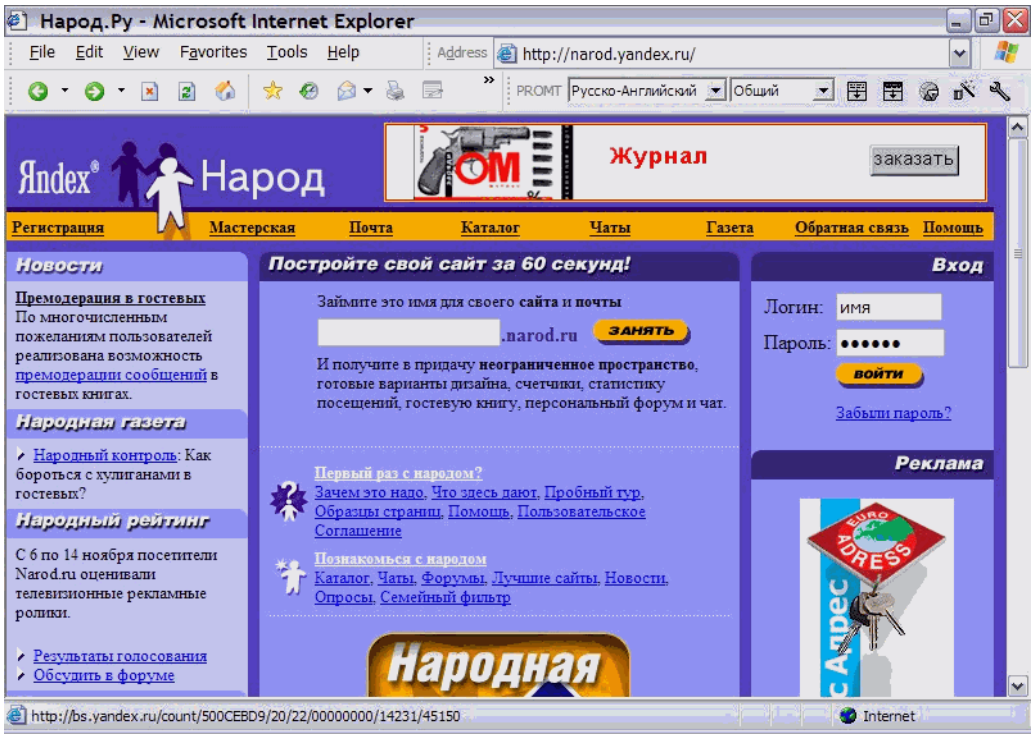


Рис. 8.64. Хостинг narod.ru

Если логин и пароль уже были, имя своего сайта вам будет предложено выбрать после входа.

Веб-сайт, размещаемый на сервере бесплатного хостинга www.narod.ru, имеет адрес <ваше_имя>.narod.ru.

narod.nu — проект компании «Яндекс». Каждый пользователь сайта «Народ» получает неограниченное пространство для построения своего сайта и почтовый ящик, а также персональный чат, форум, поиск, гостевую книгу, счётчики и статистику посещений. Страницы можно строить непосредственно на сайте с использованием шаблонов разнообразного дизайна или загружать по протоколу FTP.

Для размещения готовых страниц предоставляется двухуровневый тематический каталог, в некоторых разделах которого («Знакомства», «Бизнес и финансы», «Работа и карьера») возможен поиск страниц по параметрам. Народные сайты близкой тематики могут объединяться в сообщества.

Размещение сайта на „Народе” бесплатное. Правила предоставления услуг описаны в Пользовательском соглашении. Пользователи „Народа” несут ответственность за содержание размещаемых ими страниц.

Сайт открылся 4 февраля 2000 г. По состоянию на 24.11.2002 на Narod.Ru создано 642494 сайта.

После завершения процедуры регистрации начинается второй этап публикации сайта. Обычно каждый бесплатный веб-хостинг имеет в своей структуре некоторый файловый менеджер, который позволяет загружать подготовленные вами файлы на сервер. Недостатком файловых менеджеров является то, что вы можете за один раз опубликовать лишь несколько файлов, и папки на сервере нужно создавать вручную, а затем заполнять созданную папку файлами. Для небольших сайтов это вполне терпимо, но если у вас их несколько сотен, то занятие покажется весьма утомительным. На веб-хостинге narod.ru упоминавшийся выше файловый менеджер называется мастерской.

После регистрации непосредственно попадаете в „Мастерскую" и можете непосредственно приступить к загрузке файлов своего веб-сайта, подготовленного на предыдущих занятиях. Ниже показан интерфейс файлового менеджера на веб-хостинге narod.ru (рис. 8.65).

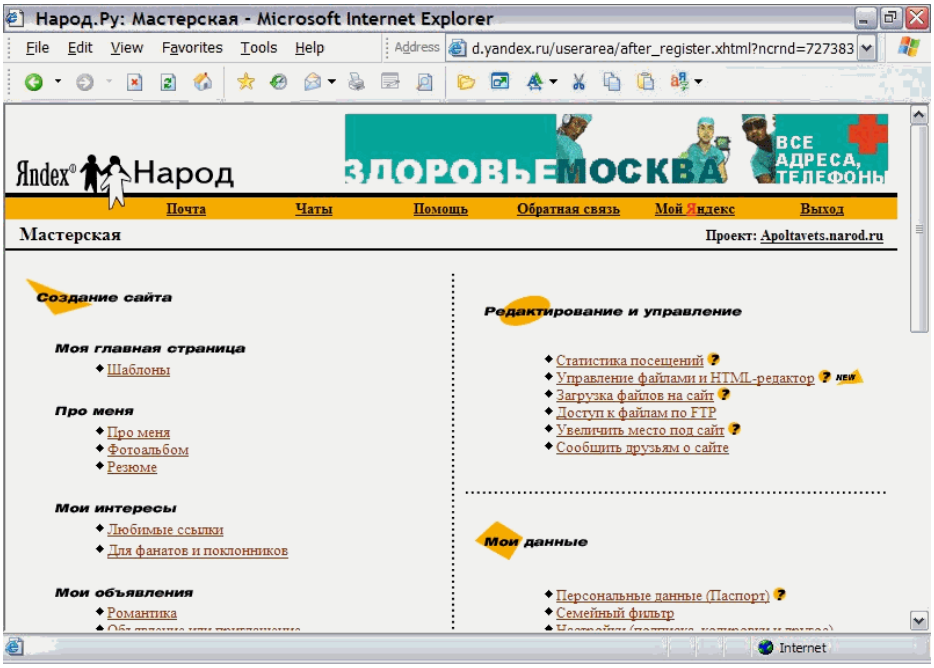


Рис. 8.65. Файловый менеджер на веб-хостинге narod.ru.

В „Мастерской" вы можете: а) создавать свой сайт, используя стандартные шаблоны; б) управлять файлами своего веб-сайта; в) просматривать статистику посещений своего сайта, и т.д.

На веб-сервере в сети Internet FrontPage XP имеет встроенные возможности публикации веб-сайтов с использованием HTTP-протокола и FTP (File Transfer Protocol – протокол пересылки файлов) протокола.

HTTP-протокол может быть использован только для публикации на серверах, поддерживающих серверные расширения FP (иначе говоря, на сервере, где вы собираетесь опубликовать сайт должно быть установлено дополнительное программное обеспечение, которое обеспечивает полномасштабное функционирование FP). Число таких серверов быстро растет, и их список можно получить по адресу www.microsoftwpp.com/wpp.search/intlwpp.htm или найти такие веб-серверы, используя поиск.

К сожалению, бесплатные веб-серверы не поддерживают серверные расширения FP, поэтому остается одна возможность для публикации сервера с помощью FP – использовать протокол FTP.

FrontPage XP имеет встроенную поддержку этого протокола, и поэтому веб-сайт можно опубликовать непосредственно из программы, воспользовавшись опциями Файл/Опубликовать веб-узел.

Итак, если веб-сайт (веб-узел) изготовлен, тщательно просмотрен и готов к публикации, переходим к следующему шагу. Перед этим проверьте имена файлов и папок, кодировку html- файлов (файлы, размещаемые на „Народе", должны быть в кодировке Windows-1251. Кроме того, не надо использовать meta-тег charset).

Имена загружаемых файлов не должны содержать недопустимые символы, например символы «пробел», «тире», русские буквы и т.д. Допустимые символы при копировании файлов – латинские буквы, цифры, точки и подчеркивания.

Необходимо проверить суммарный объем файлов. Большинство бесплатных веб-хостингов имеют ограничение объема 5 Мб. На веб-хостинге narod.ru ограничения не существует. При регистрации выделяемое под сайт пространство равно 100 Мб. При заполнении этого пространства информацией добавляется еще 100 Мб (для этого нужно всего лишь в «Мастерской» в разделе «Редактирование и управление» зайти в щелкнуть по кнопке «Увеличить место под сайт»).

4. Ход лабораторной работы

1. В меню Файл выберите команду Опубликовать веб-узел.

2. В появившемся диалоговом окне Место публикации введите расположение FTP-сервера (в нашем случае - ftp.narod.ru) и нажмите кнопку ОК (рис. 8.66).

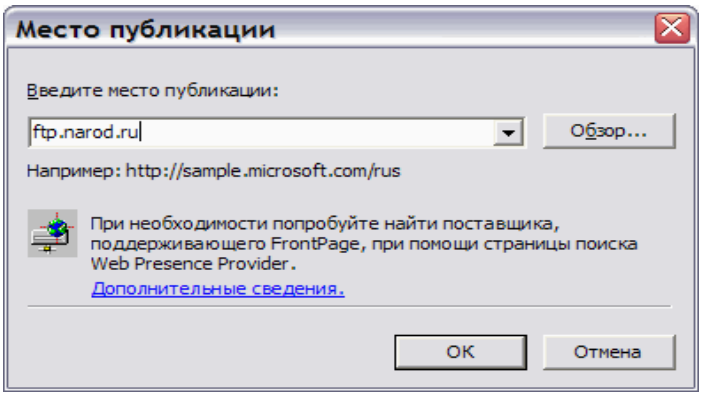


Рис. 8.66. Ввод адреса FTP-сервера

Если место публикации веб-узла было задано ранее, диалоговое окно Место публикации отображено не будет.

3. Появится окно для указания имени входа и пароля – это те самые Имя и Пароль, которые вы вводили при регистрации. Введите их (рис. 8.67).

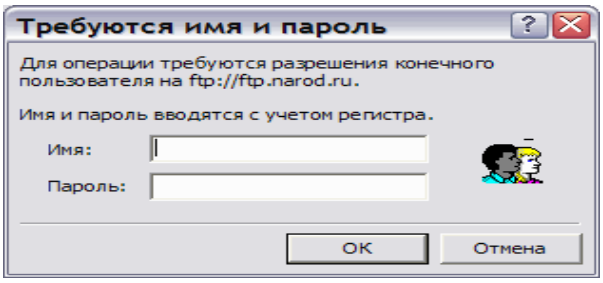


Рис. 8.67. Ввод пароля

4. Появится диалоговое окно Публикация веб-узла. Отметьте страницы, подлежащие публикации. Нажмите кнопку Опубликовать (рис. 8.68).

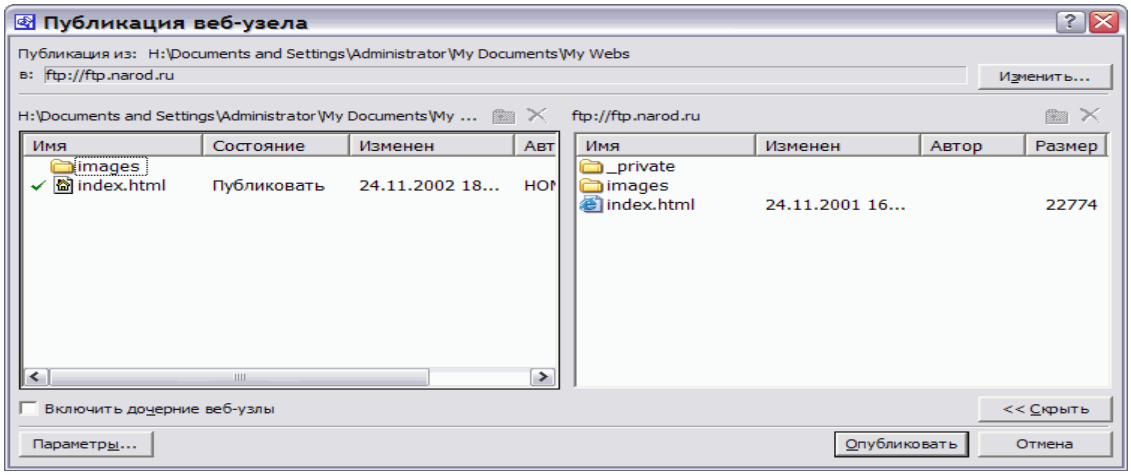


Рис. 8.68. Публикация веб-узла

В диалоговом окне Публикация веб-узла нажмите кнопку Параметры, расположенную в нижнем левом углу.

Откройте вкладку **Публикация** и выполните одно или несколько следующих действий (рис. 8.69).

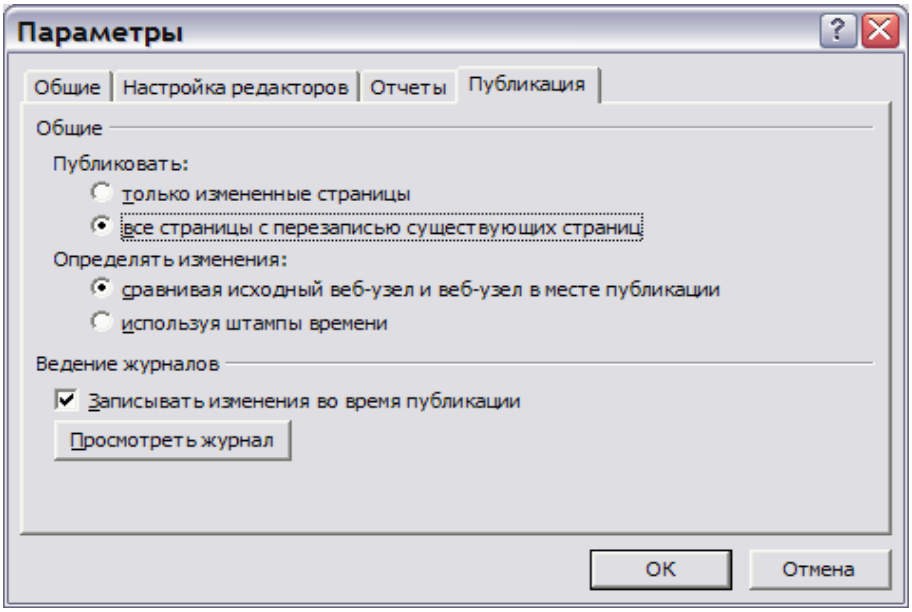


Рис. 8.69. Вкладка «Публикация»

В группе **Публиковать** определите, следует ли публиковать все страницы или только измененные. В группе **Определять** изменения задайте способ определения различий. Если требуется создать файл протокола изменений, внесенных при публикации, установите соответствующий флажок. Нажмите кнопку OK. Веб-узел будет опубликован на заданном FTP-сервере.

В случае успешного выполнения вы сразу сможете просмотреть изменения на веб-сайте (не забывайте нажать кнопку «Обновить» у браузера) [5].

6. Контрольные вопросы

1. Дайте определение понятия «хостинг».
2. Дайте определение понятия «веб-узел».
3. Какие характеристики имеет хостинг www.narod.ru?
4. Для чего необходимо использовать FTP?

7. Литература

1. Создание сайта в редакторе MS FrontPage 2003 // IVRONO. NAROD.RU: Отдел образования администрации Ивнянского района. 2005. URL: http://invrono.narod.ru/Metodish/ sait.doc (дата обращения: 12.02.2012).

2. *Омельченко, Л.Н.* Самоучитель Microsoft FrontPage 2002 / Л.Н. Омельченко, А. Ф. Федоров. – СПб. : БХВ-Петербург, 2001. – 564 с. – (Самоучитель). – ISBN 5-94157-103-8.
3. *Березин, С.В.* Ваш выход в Интернет. Секреты эффективной и безопасной работы / С.В. Березин. – СПб. : БХВ-Петербург, 2004. – 590 с. – (Самоучитель). – ISBN 5-94157-332-4.

Глава 9

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА

1. ЦЕЛЬ И ЗАДАЧИ ВЫПОЛНЕНИЯ КУРСОВОГО ПРОЕКТА

Цель выполнения курсового проекта заключается в получении практических навыков самостоятельной инженерной работы при решении комплекса задач по разработке вычислительных систем обработки и преобразования информации для технических систем различного назначения, а также в закреплении теоретических и практических знаний, полученных студентами в процессе изучения курса «Вычислительные машины, системы и сети». Достигнуть эту цель помогут знания из курсов "Электроника систем управления", "Интегральная электроника и цифровая схемотехника", "Основы программирования" и основного курса "Вычислительные машины, системы и сети".

В процессе разработки вычислительных систем студенты получают навыки конструирования систем передачи данных, алгоритмов управления и осваивают принципы программирования на языке ассемблер для управления микропроцессорными системами реального времени. В задачи проектирования входят разработка структуры вычислительной системы, подключение стандартных устройств ввода-вывода, стыковка различного рода устройств преобразования информации с каналом вычислительной системы, разработка прикладных программ, отладка и выполнение разработанных программ. Студенты получают навыки проектирования распределенных систем управления, многомашинных комплексов с использованием стандартных коммуникационных портов на основе стандартных или разработанных протоколов обмена информацией.

Курсовой проект выполняется индивидуально каждым студентом. Для этой цели предлагается достаточное количество тем. В ряде случаев студент самостоятельно определяет недостающие параметры, необходимые для успешного решения поставленной в работе задачи.

Часть работ носит реальный, практический характер, т.е. выполняется работа по тематике хоздоговорной и госбюджетной НИР кафедры.

Процесс проектирования выполняется в соответствии с графиком работ и включает в себя несколько этапов: получение задания, разработка общей структуры системы, топологии связей и обобщенного алгоритма работы, разработка систем согласования устройств преобразования с каналом вычислительной системы, разработка схем электрических принципиальных и составления блок-схемы алгоритма работы всего комплекса, разработка программного обеспечения.

Дополнительная информация приобретается студентом в сетях Internet и им подобных. Необходимую информацию также можно приобрести на университетском и кафедральном серверах, в электронной библиотеке.

Каждый этап контролируется преподавателем. Взаимодействие преподавателя со студентами осуществляется на практических занятиях и индивидуальных консультациях.

Последний этап - защита курсовой работы (проекта), осуществляемая в присутствии комиссии из трех человек, преподавателей кафедры. При подготовке материала рекомендаций использовались источники литературы, приведенные в библиографическом списке.

2. СОДЕРЖАНИЕ КУРСОВОГО ПРОЕКТА И ВАРИАНТЫ ЗАДАНИЙ

Содержание проекта

В содержание курсового проекта входят несколько разделов, основными из которых являются четыре.

В первом разделе приводится выбор топологии сети. В нем указываются расстояния между устройствами, тип используемых устройств, способы их соединения.

Второй раздел посвящен разработке полной функциональной схемы проектируемой системы, в состав которой входят и контроллеры со всеми необходимыми устройствами, обеспечивающими нормальный режим работы в сети.

Третий раздел связан с разработкой электрической принципиальной схемы всей системы с учетом особенностей систем коммуникаций.

В четвертом разделе приводится протокол работы сети.

В качестве малых вычислительных систем для реализации курсового проекта используются структуры, разработанные в курсовом проекте по микропроцессорным системам, и исходные данные, приведенные ниже, или разрабатывается система по приведенным вариантам задания. В последнем случае вариант комбинируется из нескольких разделов задания, позволяющих реализовать полноценную вычислительную систему для управления объектом, т.е. включающую в себя набор отдельных микропроцессорных устройств, объединенных для решения определенных задач.

В графической части проекта вычерчиваются топология сети, структурная схема вычислительной системы, электрическая принципиальная схема вычислительной системы, блок-схема алгоритма (протокол) работы вычислительной системы в сетевой конфигурации.

Во всех курсовых проектах проектируется подсистема связи разработанной вычислительной системы с ЭВМ верхнего уровня с использованием стандартного протокола для интерфейса RS232C на программном и аппаратном уровнях или с использованием USB-интерфейса. Возможны и другие варианты интерфейсов.

Структурная (функциональная) схема вычислительной системы должна давать наглядное представление о связях, реализуемых в вычислительной системе. Степень декомпозиции системы определяет наглядность и простоту восприятия процессов, происходящих в системе. Каждый блок отвечает конкретному выполняемому действию и процессу, в нем происходящему. Структура вычислительной части представляется не одним блоком микроЭВМ, а набором функциональных блоков и подробными связями между ними.

Варианты заданий для проектирования

Задание состоит из двух частей. Первая часть посвящена разработке локальной вычислительной системы на базе МЭВМ.

Варианты заданий приведены в табл. 9.1 - 9.9.

Таблица 9.1

Устройства измерения перемещений

№ п/п	Диапазон выходного сигнала, мм/град	Дискретность, мкм	Быстродействие, макс	Преобразователь	Вид перемещения
1	-1000 0 +1000	5	100	ФИ	Угловое
2	-500 0 +500	1	50	ВТ	Линейное
3	0 +10000	1	5	С	Угловое

Таблица 9.2

Устройства измерения скоростей перемещений

№ п/п	Максимальная скорость, м/мин, град/с	Диапазон	Быстродействие, мкс	Тип первичного преобразователя	Вид перемещения
1	-10 0 +10	5000	10	ФИ	Угловое
2	-5 0 +5	1000	5	ВТ	Линейное
3	0 +10	10000	0.5	С	Угловое
4	-1 0 +1	50000	0.1	Р	Линейное

Таблица 9.3

Устройства измерения тока

№ п/п	Диапазон выходного сигнала, А	Дискрет крет-ность, мА	Быстродействие, мкс	Преобразователь	Гальваническая развязка
1	-100 0 +100	500	10	ПЭ	Есть
2	-50 0 +50	100	5	Шунт	Есть
3	0 +1000	10	0.5	МДМ	Нет
4	-1 0 +1	5	0.1	Резистор	Есть

Таблица 9.4

У*стройства измерения напряжения*

№ п/п	Диапазон выходного сигнала, В	Дискретность, В	Быстродействие, мкс	Число каналов	УВХ
1	-10 0 +10	0.5	10	Восемь	Нет
2	-5 0 +5	0.05	5	Четыре	Есть
3	-1000 0 +1000	0.1	0.5	Два	Нет
4	-1 0 +1	0.005	0.1	Четыре	Есть

Таблица 9.5

Регулятор положения

№ п/п	Диапазон выходного сигнала, мм	Дискрет кретность, мкм	Быстродействие, мкс	Преобразователь	Вид перемещения	Регулятор
1	-999 0 +999	1	10	ФИ	Угловое	П
2	-500 0 +500	0.5	5	ВТ	Линейное	ПИ
3	-10 0 +10	0.1	0.5	С	Угловое	П
4	-1 0 +1	0.05	0.1	И	Линейное	ПИ

Таблица 9.6

Регулятор скорости

№ п/п	Максимальная скорость, м/мин, град/с	Диапазон	Быстродействие, мкс	Преобразователь	Вид перемещения	Тип регулятора
1	-10 0 +10	5000	10	ФИ	Угловое	П
2	-5 0 +5	1000	5	ВТ	Линейное	ПИ
3	-2 0 +2	10000	0.5	С	Угловое	ПИД

Таблица 9.7

Регулятор тока

№ п/п	Диапазон выходного сигнала, А	Дискретность, мА	Быстродействие, мкс	Преобразователь	Регулятор
1	-100 0 +100	500	10	ПЭ	ПД
2	-50 0 +50	100	5	Шунт	ПИ
3	0 +1000	10	0.5	МДМ	ПИД

Таблица 9.8

Тиристорный регулятор напряжения с импульсно-фазовым управлением

№ п/п	Диапазон выходного сигнала, В	Дискретность, град	Быстродействие, Гц	Схема включения
1	0 200	0.5	100	Однофазная
2	0 50	0.1	50	Двухфазная
3	0 100	0.01	150	Трехфазная

Таблица 9.9

Транзисторный регулятор напряжения с широтно-импульсной модуляцией

№ п/п	Диапазон выходного сигнала, В	Дискрет крет-ность, В	Быст-родей-ствие, Гц	Схема включения транзисторов
1	0 200	0.5	1000	Мостовая, трехфазная
2	0 50	0.1	1500	Мостовая, однофазная
3	0 100	0.01	5000	Полумостовая, трехфазная
4	0 60	0.005	100	Полумостовая, однофазная

Примечание. В таблицах приняты сокращения: ВТ- вращающийся трансформатор; ФИ – фотоимпульсный; С – сельсин; Р – резольвер; ПЭ - пьезоэлектрический; И – индуктивный; ЭМ – электромагнитный; МС – магнитострикционный; Т – тензометрический; ПР – пьезорезестивный.

Во второй части разрабатывается распределенная система на базе сетевых технологий, объединяющих разработанные МЭВМ. Удаление от управляющей ПЭВМ для всех вариантов одинаковое и равно 1000 м.

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЭТАПОВ ПРОЕКТИРОВАНИЯ

Общие указания

Прежде чем перейти к реализации устройства, необходимо провести предварительные расчеты, исходя из диапазона изменений параметра и заданной дискретности, определить формат слова. Формат полученного слова является основанием при расчете числа байт, необходимых для реализации протокола. Параметр быстродействия определит частоту передачи информации по линиям связи (табл. 9.10). Время работы алгоритма не должно быть больше заданного быстродействия.

Таблица 9.10

Параметры объекта

№ п/п	Число ОЭВМ в группе	Число групп	Скорость передачи в канала, бит/с	Удаление в группе, м	Удаление между группами, м
1	5	30	960	1	1000
2	6	29	1200	2	2000
3	7	28	2400	3	3000
4	8	27	4800	4	4000
5	9	26	9600	5	5000
6	10	25	19200	6	6000
7	11	24	38400	7	7000
8	12	23	76800	8	8000
9	13	22	153600	9	9000
10	14	21	307200	10	1000
11	15	20	960	11	1100
12	16	19	1200	12	1200
13	17	18	2400	13	1300
14	18	17	4800	14	1400
15	19	16	9600	15	1500
16	20	15	19200	16	1600
17	21	14	38400	17	1700
18	22	13	76800	18	1800
19	23	12	153600	19	1900
20	24	11	307200	20	2000
21	25	10	9600	21	2100
22	26	9	19200	22	2200
23	27	8	38400	23	2300
24	28	7	76800	24	2400
25	29	6	153600	25	2500

При заданных перемещении и величине дискреты вычисляют число бит для задания минимальной и максимальной величины перемещения. Например, заданная величина перемещения равна 500 мм, а дискретность перемещения – 0,001 мм, тогда число дискрет равно 500000, и для задания этой величины необходимо 19 двоичных разрядов, т.е. более двух байт. Аналогично рассчитываются и остальные параметры. Далее просчитывают число бит для задания команд

управления. Такие команды, как включение, выключение, реверс и тому подобные можно задавать любым числом бит, но необходимо помнить о скорости передачи информации. В процессе работы системы учитываются и сигналы, информирующие о текущем состоянии объекта, например сигналы ограничения перемещения. Для них также отводится определенное число бит.

Обмен данными может осуществляться либо последовательно, либо параллельно, либо смешанным образом. Зачастую использование стандартных интерфейсов по тем или иным причинам не удовлетворяет потребностям разработчиков. Среди этих причин можно выделить как экономические, так и технические. Главной же причиной является избыточность стандартных средств в интерфейсах, которые разрабатывались с учетом универсальности в предметной области.

При разработке проблемно-ориентированных систем многие функции стандартных интерфейсов не используются, а в ряде случаев являются сдерживающими при разрешении, например, задач быстродействия, удобства обслуживания, габаритных размеров и т.д. Однако при последовательном обмене информацией использование стандартного протокола обмена обычно не вызывает препятствий. Действительно, для организации последовательного интерфейса необходимо преобразовывать передаваемые данные из параллельного формата в последовательный. Для сопровождения информации в передаваемом сигнале добавляются специальные служебные символы. К ним относятся стартовый и стоповый биты и бит контроля. В посылке эти символы распределяются так, как это показано на рис. 9.1, *а*. На приемной стороне производится обратное преобразование и происходит распознавание служебных символов.

Передача информации может выполняться синхронно или асинхронно. В синхронном режиме данные передаются массивами слов, а синхронизация достигается использованием нескольких байт, специально для этого выделенных. Если обмен осуществляется с несколькими устройствами, то в передаваемой информации должны содержаться адресные посылки (рис. 9.1, *б*).

Общая процедура приема (передачи) данных сводится к анализу посылки и выделению стартового бита, дешифрации адреса устройства приема, приему, упаковке передаваемого слова и анализу ошибок в переданных данных.

Временные интервалы определяются в зависимости от количества передаваемой информации, а также способом физической реализации канала связи. Последовательная передача данных осуществляется для удаленных объектов, включая и иерархические системы.

При параллельном обмене информацией число линий связей определяется разрядностью шин данных, адресов и управления. Протокол обмена должен поддерживать процедуры по вводу и выводу информации используемого микропроцессора или МЭВМ. Скорость передачи достаточно велика и составляет сотни наносекунд. Однако использование параллельной связи для удаленных объектов экономически нецелесообразно.

В табл. 9.11 приведены основные технические характеристики некоторых широко распространенных стандартных интерфейсов. В этой таблице не приводятся последовательные интерфейсы, так как число линий связи для них одинаково, а отличие заключается только в скорости передачи информации по каналу. Последнее определяется физической реализацией канала связи и расстоянием от приемника до источника.

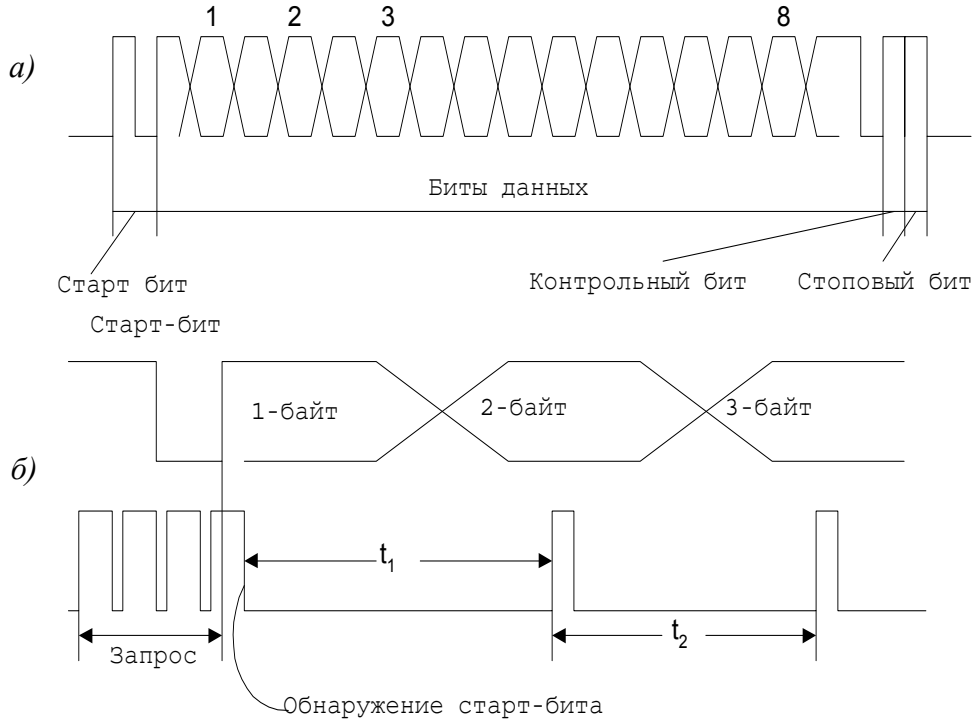


Рис. 9.1. Распределение информации в передаваемых данных

Для расчетов пропускной способности канала, скорости или времени передачи информации поступим следующим образом. В зависимости от типов каналов передачи информации требуется рассчитать их производительность, в общем объеме времени которых учитывается и время передачи информации от объекта к ЭВМ.

Оценку времени передачи информации по каналу связи можно провести по формуле

$$t_2 = \frac{[(I + \log_2 a + \log_2 b + F)d - Qht_1]m}{Qh},$$

где a - число используемых адресов; I - разрядность информационной шины; b - используемое число команд; F - эквивалентное число команд для реализации аппаратных функций; t_1- общее время на реализацию всех процессов в системе; d - учет времени при использовании канала ПДП; Q - добротность канала; h - общее число шин интерфейса; m - число слов для реализации обмена.

Таблица 9.11

Характеристики наиболее употребительных интерфейсов

Интерфейс	Число шин			$t_{пер}/f$ одного бита	Число команд	Общее число шин
	Адрес	Данные	Управление			
SC 13	8	9/18		1МГц	20	
IES 625-1	8	8	8	1МГц		
ISI	16	18	4	3МГц		
Q-bus	16	16				
ИРПР	16	16	21			64
MYC	8/16	16	11	0.5МГц		
IEEE (P-796)	20	16	22		28	86
AMS-bus	16/20	16	21	0.4мкс		
И41	20	16				96
Eurobus	34	34	12			
VME-bus	31	16/32	41	0.3мкс		96
COMPEX	24	24/29	46	0.6мкс		86
ESSS	16/24	18	23	0.25мкс		96
Futurebus	32	32	16	400нс		96

Используя данные, приведенные в табл. 9.11, и учитывая, что при параллельном обмене максимальная разрядность устройств ввода-вывода не превышает шестнадцати разрядов, а адресная шина – восьми, и число линий управления 5, произведем расчет параметров. Среднее число команд *b* на реализацию протокола обмена определим по алгоритму, представленному на рис. 9.2. Он включает в себя формирование адреса передаваемых данных, загрузку данных в передающее устройство и передачу в канал. На приемной стороне эти процедуры аналогичны. Поэтому окончательный результат должен быть удвоен. Среднее число команд *b* при вводе-выводе составляет примерно 15 – 20.

Расчет добротности выполним по следующей методике. Добротность канала (пропускную способность) можно определить на основе анализа энтропии до и после эксперимента. Количество информации, получаемой в результате эксперимента, равно: $I(X,U) = H(X) - H(X/U) = H(U) - H(U/X)$.

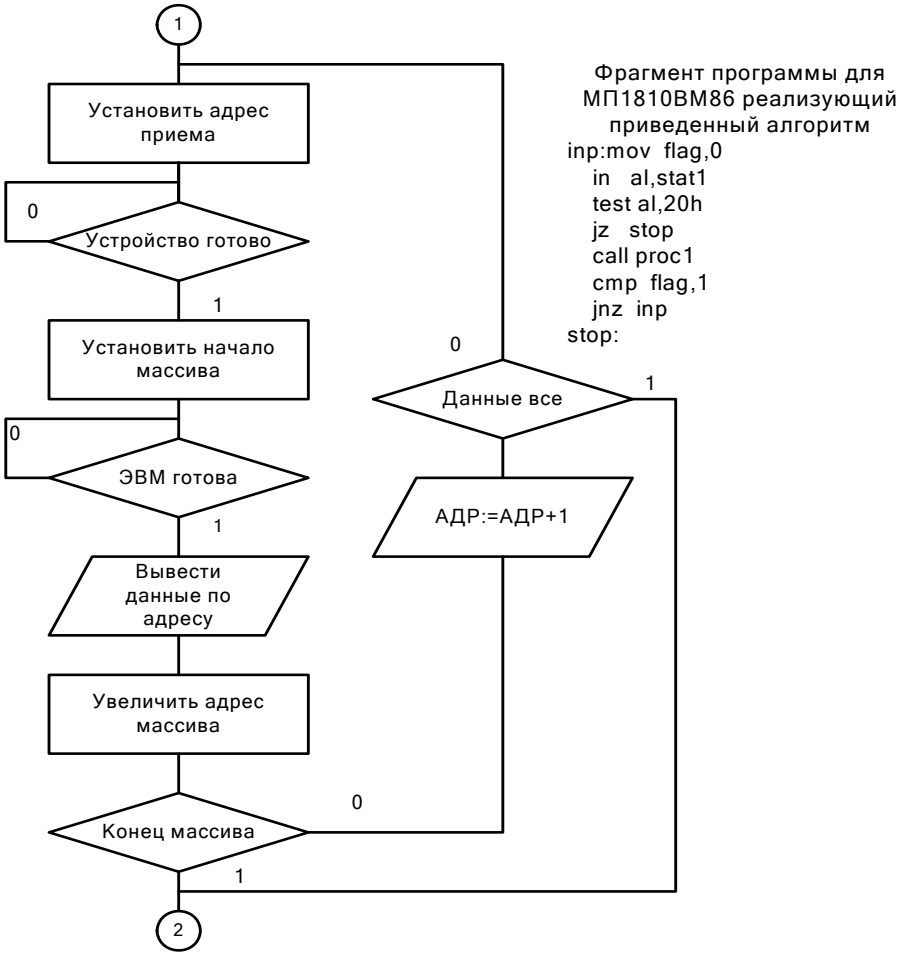


Рис. 9.2. К оценке числа команд ввода-вывода

Если предположить, что устройство свободно от погрешностей, а квантование сигнала по уровню носит идеальный характер и, кроме того, известен динамический диапазон и шаг (ступень) квантования, то число уровней определяется как: $N = L/\Delta = a^{I(X/U)}$, где a - основание систем счисления.

При равномерном распределении $I(X,U) = \log_a(L/\Delta) - H(X/U)$ и тогда $N = La^{H(XU)}/\Delta$.

В этом случае скорость получения информации можно определить: $V = I(X,U)/T$, а пропускную способность как $C = \max_{\{f(x)\}} V$.

При нормально-распределенных аддитивной помехи U и сигнала X с мощностями Pn и Pc и заданной граничной частоте сигнала $H(X) = 0.5\log 2\pi e P_c$, $H(U) = 0.5\log 2\pi e P_n$, $H(X,U) = 0.5\log 2\pi e(Pn+Pc)$ и пропускная способность $C = n/T[H(X) - H(X/U)]$.

Таким образом, пропускная способность канала определяется из соотношения $C = f_{гр} \log(1+Pc/Pn)$, а максимальное количество получаемой информации из соотношения $I_{\max}(X,U) = CT = T f_{гр} \log(1+Pc/Pn)$.

Считая, что установившаяся ошибка в канале передачи информации не превосходит Δ, определим добротность канала как

$$Q = \frac{V}{\Delta} = \frac{C}{\Delta} = \frac{f_{гр} T \log(1 + P_C / P_n)}{\Delta T} = \frac{f_{гр} \log(1 + P_C / P_n)}{\Delta}.$$

Скорость передачи информации можно определить по более простой формуле, если не учитывать помехи: $V = 0.1 f_{гр} \log(L/\Delta)$.

Соответственно пропускная способность и добротность $C = 0.1 f_{\max} \log(L_{\max}/\Delta)$, $Q = (0.1 f_{гр} \log(L/\Delta))/\Delta$.

Еще более простую оценку добротности можно связать с быстродействием используемых МЭВМ или микропроцессора. В том случае если передача осуществляется по последовательному каналу, добротность определяют как $Q = 0{,}1f$ (1/с), а по параллельному – $Q = (1{,}1...1{,}5)f$ (1/с). В последнем соотношении меньшему коэффициенту соответствует меньшая по разрядности шина данных. Эти соотношения целесообразно использовать только на стадии эскизного проектирования для ориентировочных оценок.

После определения всех составляющих приведем расчеты и построим зависимости времени передачи информации от добротности канала $t_1 = f(Q)$ при параллельной и последовательной передаче.

Пример

При шаге квантования $\Delta = 10^{-3}$, граничной частоте 1000 Гц и динамическом диапазоне 10 В скорость, пропускная способность и добротность соответственно составляют V = 60,0, C =100,0[бит/с], Q = =60000,0 (1/с) (рис. 9.3).

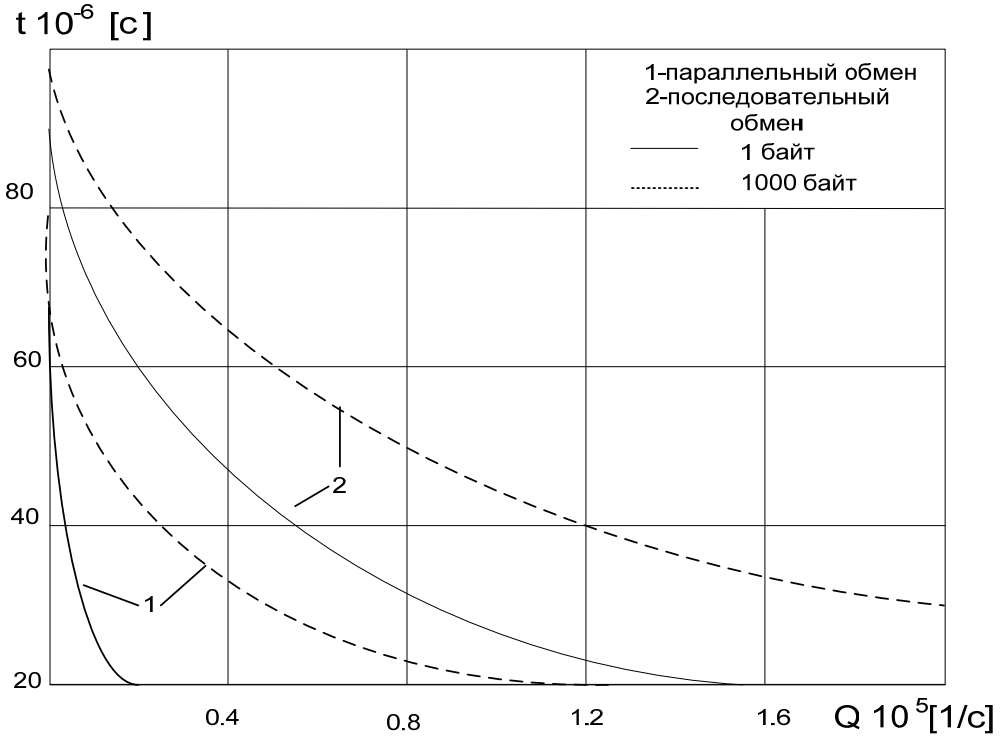


Рис. 9.3. Определение быстродействия выполнения алгоритмов

Для указанного типа микропроцессорной системы выбирается конкретный представитель с его схемотехническим решением и встроенным устройством для связи с каналами передачи данных. Его параметры также должны соответствовать заданным характеристикам. Приводятся структурная и полная электрическая принципиальная схемы с указанием номиналов элементов, их типа и т.д. Для сетевой версии необходимо выбрать топологию и физическую реализацию среды передачи данных, концентраторы, хабы, разветвители и т.д.

Процесс разработки структурной схемы сопровождается предварительными расчетами. При необходимости использования графиков, диаграмм и тому подобного последние приводятся по тексту с

указанием первоисточника. При выборе тех или иных технических решений не должно быть голословных заявлений типа "этот лучше или нам подходит". Должно быть четкое утверждение, например, "так как разрядность этого устройства на один бит больше, а величина выходного сигнала равна 5В, то....".

Принципиальная электрическая схема вычислительной системы разрабатывается на основе структурной схемы. Выбор элементов схемы производится в соответствии с решаемой задачей. Каждый элемент должен иметь таблицу характеристик по уровням, быстродействию, технологии изготовления и т.д. Выбор конкретного элемента должен быть обоснован как расчетами, так и описанием работы. Следует указать, почему отдано предпочтение этому (выбранному) элементу. Если элемент представлен СБИС, то для него необходимо привести особенности программирования, подключения тех или иных входов/выходов, сопроводить соответствующей таблицей кодировок и т.п. Указывается нагрузочная способность выводов микросхем. Резисторы, конденсаторы и прочие элементы выбираются с указанием конкретных номиналов, типа, точности и условий работы. Приводятся расчеты по выбору мощности элемента, номинала и т.д.

Блок-схема алгоритма работы вычислительной системы должна отражать последовательность выполняемых действий в работе от начальной загрузки до завершения работы. Каждый элемент блок-схемы несет смысловую нагрузку, отражающую не только выполняемую функцию или процедуру в системе, но и метод описания задания для использования конкретных операторов языка программирования. При этом алгоритм не должен сводиться к простому перечислению выполняемых команд. Блок-схема алгоритма должна быть подробно описана в тексте. Указываются символические имена, расшифровываются все переменные, даются пояснения для блоков, отражающих укрупненную функцию.

Определяется формат передаваемого пакета, способ его реализации. Приводятся структуры пакетов. В каждом из них указывается назначение бит. Обеспечивается контроль достоверности передаваемых данных, команд и служебной информации. Рассчитывается число байт, передаваемых в кадре, исходя из особенностей работы МЭВМ, управляющей исполнительным устройством. Логическая организация сети представляется блок-схемой алгоритма (протокола) работы.

Перед началом написания программы на языке ассемблер необходимо провести распределение памяти, составить списки меток и придать им конкретные физические адреса, установить адрес начала программы пользователя. При наличии программного обеспечения и макросредств для написания программ последнее выполняется в автоматическом режиме. Исходный текст программ представляется в напечатанном виде с учетом синтаксиса и требований языка программирования. Поле „Комментарий” должно быть заполнено текстовым разъяснением выполняемых функций.

Более подробные рекомендации по выполнению разделов проекта приведены ниже.

2. Выбор элементной базы вычислительной системы

Однокристальные ЭВМ (в дальнейшем МК) семейств MSC48, MSC51, MSC96 и других обладают значительными функционально-логическими возможностями и представляют собой эффективное средство для управления (автоматизации на основе применения средств и методов обработки данных и цифрового управления) разнообразными объектами и процессами. Наибольший эффект от их применения достигается в локальных устройствах автоматики.

Анализ основных признаков показывает, что перечисленные МК целесообразно использовать как на этапе опытно-конструкторской разработки и отладки систем, так и в изделиях, выпускаемых малыми и большими сериями. Микроконтроллеры, в которых нет резидентной памяти программ, используют, как правило, не в конечных изделиях, а в автономных отладочных устройствах и многофункциональных программируемых контроллерах, где в качестве памяти программ и данных используются внешние БИС и имеются средства загрузки программ.

Выбор того или иного МК связан с содержанием задания в части постановки задачи и в части параметров, определяющих характеристики проектируемого устройства. На базе микроконтроллеров можно строить и сетевые устройства, такие как коммутаторы, распределители, концентраторы, например так, как это показано на рис. 9.4.

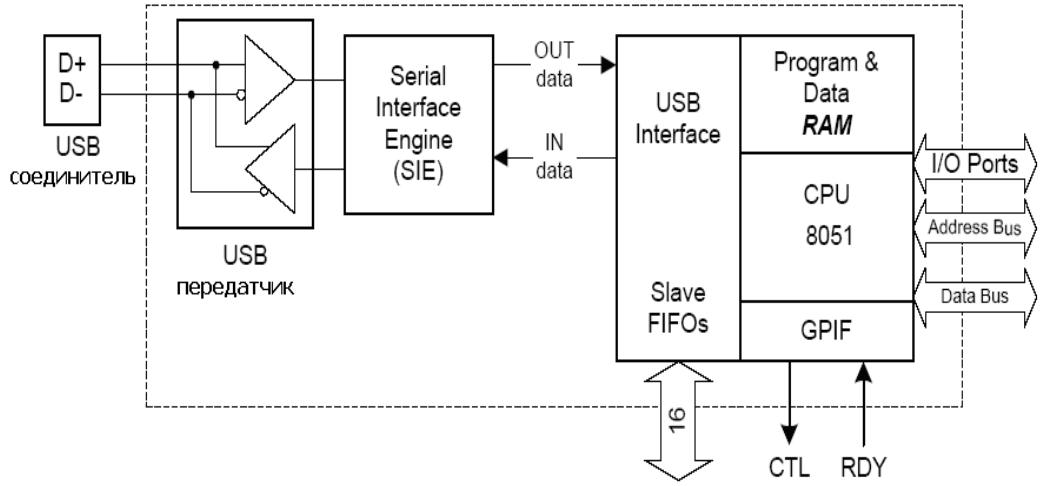


Рис. 9.4. Микроконтроллер со встроенным портом

Основные характеристики контроллера 80C51XA (табл. 9.12):

- Максимальная тактовая частота 30 МГц .
- Время выполнения команды пересылки из регистра в регистр 100 нс (3 такта) при 30 МГц.
- Размер шины адреса 20 - 24 бита, шины данных 8 или 16 бит (определяется пользователем).
- CPU полностью статическое, 16 бит.
- 32 вектора прерываний: 31 маскируемое прерывание и NMI.
- Аппаратная поддержка мультизадачного программного обеспечения.
- Расширенная по отношению к 80C51 система команд.
- Регистры 8; 16 бит для работы с арифметическими и логическими операндами.
- Указатель стека (SP): 16 бит.
- Аппаратная реализация умножения 16·16 (12 тактов), деления 32:16 (24 такта).
- Четыре (девять для XA-S3) восьмибитных порта с четырьмя программируемыми конфигурациями выводов.
- 64 регистра специальных функций.
- Режим пониженного потребления.
- Широкий диапазон напряжений питания: от 3.0 до 5.5V.
- Минимальная конфигурация: внешний кварцевый резонатор.
- Режимы адресации: регистровая, косвенная, косвенная со смещением, прямая, непосредственная, битовая, адресация регистров специальных функций.

Таблица 9.12

Однокристальный контроллер 80C51XA

Тип	Память		Особенности архитектуры			
	(EP) ROM	RAM	A/D	UART	Timers	Speed MHz
80C51XA-G1	8К	512		2	3+WD	25
80C51XA-G2	16К	512		2	3+WD	25
80C51XA-G3	32К	512		2	3+WD	30
80C51XA-C3	32К	1024		1+Net	3+WD	25
80C51XA-S3	32К	1024	8-10	2+I^2C	4+WD+PCA	20

Примечание. С3 поддержка CAM (1 Mbit/s), детектор напряжения питания. S3 – два аналоговых компаратора, шесть восьмибитных PWM.

Сведения о некоторых узлах МЭВМ семейства MCS-96

EPA (*EVENT PROCESSOR ARRAY*) Этот узел пришел на смену *HSIO*, начиная с кристалла 8XC196KR. *EPA* имеет более простую архитектуру, чем *HSIO*, обладая при этом лучшей разрешающей способностью. В *HSIO*, все входные каналы имеют общую память (7-уровневое FIFO), в которой запоминаются временные отметки, соответствующие событиям на входах. То же касается выходных линий *HSIO* - все они имеют общую память (8 ячеек), в которую процессор записывает команды. Поэтому за один такт *HSIO* может обработать только один входной и один выходной канал. В *EPA* каждый канал имеет свой собственный буфер, а выдача и прием сигналов производятся одновременно по всем каналам. Поэтому разрешающая способность *EPA* выше, чем у *HSIO*, в 4 раза. Кроме того, *EPA* - более гибкий узел: каждый его канал может служить и входом, и выходом, тогда как *HSIO* имеет 4 выходные, 2 входные и 2 двунаправленные линии.

CODE RAM – это дополнительное ОЗУ, в котором можно размещать исполняемый код. Этот код будет выполняться очень быстро, так как *Code RAM* имеет 16-разрядный интерфейс с нулевым циклом ожидания. *Code RAM* может принести существенную пользу в задачах, где требуется максимально быстрое выполнение только небольших фрагментов кода, позволяя при этом использовать сравнительно медленное и дешевое 8-битное ПЗУ для хранения остальной части

программы. Конечно, эту память можно использовать и для размещения данных или стека.

PTS (*PERIPHERAL TRANSACTION SERVER*) Этот узел предназначен для аппаратной обработки прерываний. Он содержит набор встроенных алгоритмов, исходные данные для которых должны быть размещены программой пользователя во встроенном ОЗУ кристалла. Алгоритмы PTS охватывают в основном пересылки данных. Прерывания, обслуживаемые PTS, отрабатываются быстрее, чем те, которые обслуживаются обычным способом. Однако программировать PTS непросто, а отлаживать еще сложнее. Поэтому мы не рекомендуем использовать PTS без крайней необходимости. В новейшем кристалле 4-го поколения, 80296SA, PTS нет.

ГЕНЕРАТОР СИГНАЛОВ *CHIP SELECT* (*CHIP SELECT UNIT*). Этот узел появился у кристалла 8XC196NP и имеется у 80C196NU и 80296SA. Он позволяет существенно упростить аппаратуру, необходимую для подключения внешней памяти к процессору, и тем самым удешевить систему.

МК может генерировать до 6 сигналов выборки (*Chip Select*) с независимо устанавливаемыми циклами ожидания и шириной шины. Кроме того, кристаллы с *Chip Select Unit* имеют демультиплексированную шину, что позволяет отказаться от внешних регистров-защелок и использовать медленную и дешевую память, сохранив при этом быстродействие системы. Краткая техническая характеристика МК 80C196KB (табл. 9.13).

ЦПУ. На частоте 16 МГц ЦПУ выполняет 2 млн оп/с при выполнении элементарных операций над знаковыми/беззнаковыми данными длиной 1 или 2 байта. Для этих чисел имеются также и операции умножения и деления (быстродействие: 580 тыс. умножений/с, 330 тыс. делений/с).

ПАМЯТЬ И ВНЕШНЯЯ ШИНА. ЦПУ имеет одно адресное пространство размером 64К байт, в котором находятся регистры общего назначения (232 байта), регистры специального назначения, встроенная программная память (если имеется), внешняя память для программы и данных. У версии со встроенным ПЗУ (87C196KB), ПЗУ имеет объем 8 кбайт и оснащено защитой от несанкционированного доступа.

Контроллер памяти работает с 8- и 16-разрядной внешней шиной, причем ширина шины может динамически переключаться, можно вводить циклы ожидания.

ПРЕРЫВАНИЯ. 28 источников запросов, 16 векторов и 16 приоритетов.

ТАЙМЕРЫ. Два 16-разрядных таймера TIMER1 и TIMER2 обеспечивают синхронизацию работы устройства ввода-вывода импульсных сигналов (*HSIO, High Speed In/Out unit*) с реальным временем и внешними событиями. TIMER1 синхронизируется изнутри, тогда как TIMER2 – снаружи.

Таблица 9.13

Некоторые характеристики контроллеров семейства MSC96

Кристалл	Частота, МГц/ MIPS	Память	Каналы АЦП	Количество линий ввода-вывода	HSIO/ EPA*	ШИМ
8X96BH	12/1	64K	8	48	HSIO	1
8XC196KB	16/2	64K	8	48	HSIO	1
8XC198 8	16/2	64K	4	48	HSIO	1
XC196KC	20/2.5	64K	8	48	HSIO	3
8XC196KD	20/2.5	64K	8	48	HSIO10	3
8XC196KR	16/2	64K	8	56	EPA6	-
8XC196JR	16/2	64K	6	41	EPA10	-
8XC196KT	16/2	64K	8	56	EPA	-
8XC196JT/J	16/2	64K	6	41	6 EPA	-
S 8XC196JV	20/2.5	64K	6	41	6 EPA	-
8XC196MC	16/2	64K	13	53	4 EPA	-
8XC196MD	16/2	64K	14	64	6 EPA	-
8XC196MH	16/2	64K	8	50	6 EPA	-
8XC196CA	20/2.5	64K	6	44	6 EPA	-
8XC196NT	20/2.5	1M	4	56	10 EPA	-
8XC196CB	20/2.5	16M	8	56	10 EPA	-
8XC196NP	25/3	1M	Нет	32	4 EPA	3
80C196NU	50/6	1M	Нет	32	4 EPA	3
80296SA	50/16	6M	Нет	32	4 EPA	3

* HSIO/EPA- высокоскоростной канал ввода/вывода.

ЦИФРОВЫЕ ПОРТЫ. Имеется шесть 8-разрядных портов ввода/вывода цифровых сигналов.

ИМПУЛЬСНЫЙ ВВОД И ВЫВОД (*HSIO*). Одно из самых мощных встроенных устройств 80C196 - устройство генерации импульс-

ных сигналов (*HSO Unit*). Его функция - выполнять различные действия в заранее запрограммированные моменты времени с минимальным контролем со стороны ЦПУ. От ЦПУ требуется только указать, что сделать, и в какой момент времени (время отсчитывается по так называемому ссылочному таймеру - TIMER1 или TIMER2). Помимо генерации сигналов *HSO* одновременно может выполнять функции 4 дополнительных таймеров.

Устройство ввода импульсных сигналов (HSI Unit) фиксирует моменты времени, в которые произошли какие-либо внешние события, например переход из 0 в 1. *HSI* имеет 4 входа, а *HSO* - 6.

АЦП. Встроенный 10-разрядный АЦП имеет 8 входов, диапазон входного напряжения 0 - 5 В. На частоте 16 МГц время преобразования 19,5 мкс. Имеется схема выборки/хранения и отдельные входы опорного напряжения и аналоговой земли.

ГЕНЕРАТОР ШИМ-СИГНАЛА. Генератор ШИМ имеет один выход. Диапазон изменения скважности импульсов 256 градаций. Период импульсов может быть равен 256 или 512 тактам (31, 25 или 15, 625 кГц соответственно для частоты 16 МГц).

ПОСЛЕДОВАТЕЛЬНЫЙ ПОРТ. На ОЭВМ имеется последовательный универсальный синхронно-асинхронный дуплексный порт связи (SIO, Serial In/Out). Максимальная скорость обмена (на частоте 16 МГц) в асинхронном режиме - 1 Мбод, в синхронном - 4 Мбод.

Рекомендации по выбору цифроаналогового преобразователя. Преобразование информации из цифровой формы в аналоговую осуществляется путем подключения БИС ЦАП к одному из портов МК. Выдача аналогового управляющего воздействия в этом случае сводится к одной команде, например OUTL P1, A. При этом на выходе ЦАП появится напряжение (ток), пропорциональное двоичному коду, загруженному в порт 1.

Некоторые объекты управления могут требовать непрерывного управляющего воздействия сложной формы. Для реализации такого воздействия в МК используются цифровые методы интегрирования: на каждом интервале времени Δt непрерывная функция заменяется ее средним дискретным значением. Таким образом, управляющее воздействие становится ступенчатым (рис. 9.5.) и программа его формирования может быть реализована с использованием процедур выдачи кода и временной задержки заданной длительности.

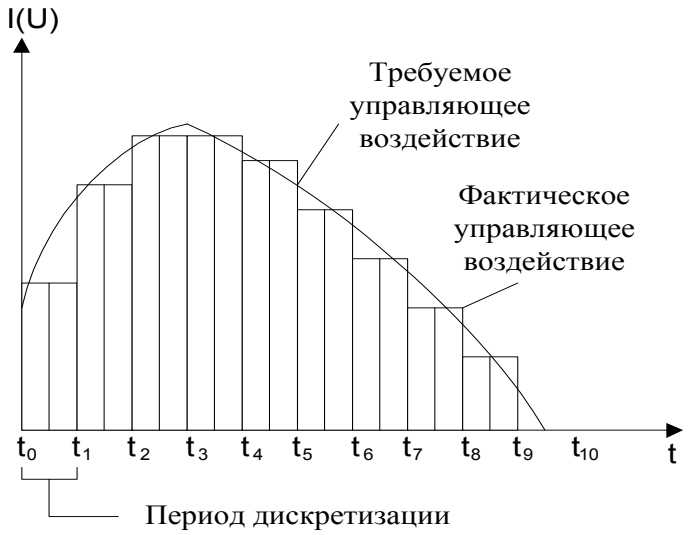


Рис. 9.5. Дискретизация непрерывной функции

Если предположить, что для управления некоторым объектом требуется сформировать управляющее воздействие с периодом дискретизации 100 мкс, то прежде всего в ПП следует сформировать таблицу (TABL) двоичных эквивалентов дискретных значений функции для каждого из периодов дискретизации. Если допустить, что длина TABL в байтах хранится в ее первом байте, а выводимые из таблицы двоичные коды не требуют масштабирования, то программа формирования управляющего воздействия будет иметь вид:

```
FANCNV:
MOV R3,◊LOW TABL    ФОРМИРОВАНИЕ АДРЕСА
MOV      A,R3              ;ПЕРВОГО БАЙТА ТАБЛИЦЫ TABL
MOVP3 A,@A                 ;ЧТЕНИЕ ДЛИНЫ ТАБЛИЦЫ
                           ;ИЗ ПАМЯТИ ПРОГРАММ
MOV R2,A                   ;(R2)<ДЛИНА ТАБЛИЦЫ TABL
LOOP:      INCR3           ;ПРОДВИЖЕНИЕ УКАЗАТЕЛЯ TABL
MOV      A,R3
MOVP3 A,@A                 ;ЧТЕНИЕ КОДА ИЗ TABL
OUTL P1,A                  ;ВЫВОД КОДА НА ЦАП
CALL  ELAY                 ;ЗАДЕРЖКА НА ДЛИНУ ПЕРИОДА
DJNZ R2,LOOP               ;ДЕКРЕМЕНТ R2, И ЦИКЛ, ЕСЛИ НЕ НУЛЬ
EXIT:                      ;ВЫХОД ИЗ ПРОЦЕДУРЫ
```

Рекомендации по применению аналого-цифрового преобразователя. Преобразование аналогового сигнала от датчика в цифровой код, принимаемый и обрабатываемый в МК, можно осуществить несколькими способами:

1) аппаратурным на основе БИС АЦП, подключаемой к порту МК. В этом случае МК только инициирует АЦП и через заданные периоды дискретизации считывает из него цифровой код. Данный способ характеризуется самым высоким быстродействием, но требует использования БИС АЦП, что далеко не во всех применениях МК является оправданным;

2) аппаратурно-программным на основе БИС ЦАП и программы взвешивания бит (последовательных приближений, побитного уравновешивания). Данный способ характеризуется хорошим быстродействием и требует использования относительно простых и дешевых микросхем ЦАП и операционного усилителя;

3) программно-аппаратурным на основе метода двойного интегрирования. Это самый дешевый, но и наиболее медленный способ; может обеспечить достижение очень высокой точности преобразования. Из дополнительного оборудования требуются два операционных усилителя и аналоговый мультиплексор на два входа;

4) аппаратурно-программным на основе использования преобразователя "напряжение-частота" и программы измерения периода сигнала.

Наибольший практический интерес представляют способы 2-й и 3-й, так как их использование обеспечивает получение высоких технико-экономических характеристик МК-систем относительно простыми средствами.

Метод последовательных приближений. Из дополнительной аппаратуры в МК используются интегральные ЦАП на восемь входов и сравнивающий операционный усилитель (компаратор), включенные так, как показано на рис. 9.6.

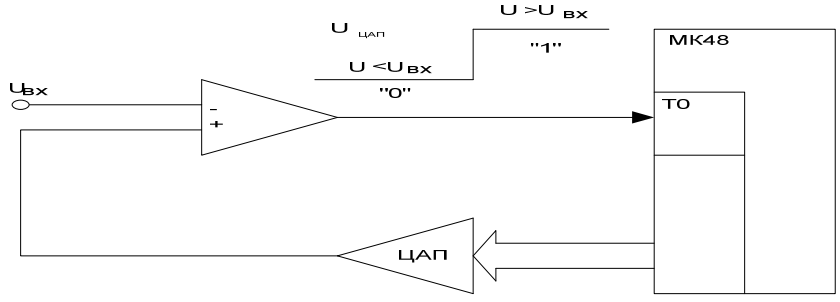


Рис. 9.6. Схема аналого-цифрового преобразования

На выходе компаратора формируется сигнал 0, если $U_{ЦАП}<U_{ВХ}$, и сигнал 1, если $U_{ЦАП}>U_{ВХ}$. Микроконтроллер через порт P1, работающий в режиме вывода, передает двоичные коды в ЦАП, выход которого подсоединяется к входу «плюс» компаратора. Выход компаратора (двоичный сигнал) заводится на вход тестирования T0.

Программа аналого-цифрового преобразования работает следующим образом: МК выдает через порт 1 байт данных, преобразуемый ЦАП в аналоговый сигнал $U_{ЦАП}$ и сравниваемый с входным аналоговым сигналом $U_{ВХ}$, а затем анализирует результат сравнения. В зависимости от значения сигнала на входе T0 МК или оставляет старший бит выводимого байта в 1, если $U_{ЦАП}>U_{ВХ}$, или сбрасывает его в 0, если $U_{ЦАП}<U_{ВХ}$. Затем аналогичным образом в порядке убывания весовых значений проверяется каждый бит выводимого байта:

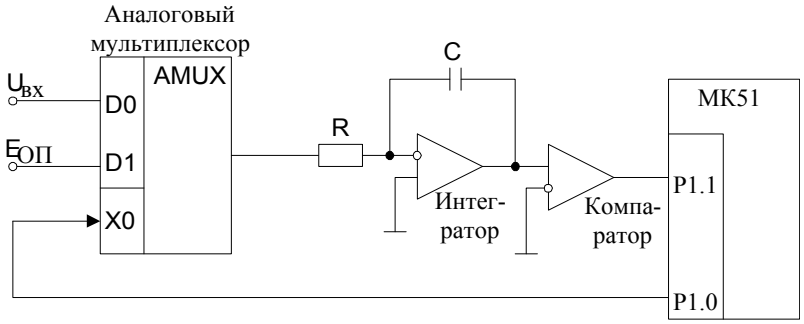
```
                                    ;ВЕРСИЯ ДЛЯ МК48
;R4 - РЕГИСТР ЦИФРОВОГО ЭКВИВАЛЕНТА
;R3 - РЕГИСТР БЕГУЩЕЙ ЕДИНИЦЫ ДЛЯ УКАЗАНИЯ
;ТЕКУЩЕГО ВЗВЕШИВАЕМОГО БИТА
CONVRT :MOV     R5, ◊08H  ;ЗАГРУЗКА СЧЕТЧИКА ЦИКЛОВ
        MOV     R3, ◊1
        MOV     R4, ◊0
LOOP :  MOV     A,R3      ;СДВИГ (R3) ВПРАВО НА 1 РАЗРЯД
        RR      A
        MOV     R3,A
        ORL     A,R4      ;ВЗВЕШИВАНИЕ ТЕКУЩЕГО РАЗРЯДА
        OUTL    P1,A      ;ВЫДАЧА ЦИФРОВОГО
                          ;ЭКИВАЛЕНТА НА ЦАП
        JTO     ENO       ;ЕСЛИ T0=1, ТО БАЙТ БОЛЬШЕ
                          ;ДВОИЧНОГО ЭКВИВАЛЕНТА UВХ, И
                          ;СОХРАНЯЕТСЯ СТАРОЕ ЗНАЧЕНИЕ R4
                          ;ЕСЛИ T0=0, ТО УСТАНОВЛЕННЫЙ БИТ
        MOV     R4,A      ;ЗАПОМИНАЕТСЯ В РЕГИСТРЕ R4
        DJNZ    R5,LOOP   ;ДЕКРЕМЕНТ R5 И, ЕСЛИ НЕ НУЛЬ, ТО К
                          АНАЛИЗУ СЛЕДУЮЩЕГО (J-1)-ГО БИТА
        END
```

Метод двойного интегрирования. Схема подключения к МК дополнительной аппаратуры показана на рис. 9.7. Первоначально на вход интегратора подается положительное напряжение $E_{ОП}$. При этом

на выходе интегратора через некоторое время установится отрицательный уровень, а на выходе компаратора будет сформирован сигнал 0. Процесс преобразования состоит из двух этапов. Сначала производится интегрирование входного аналогового сигнала в течение строго определенного времени Т1. Отсчет интервала Т1 производится от момента t_0 перехода напряжения на выходе интегратора через нуль. Входной преобразуемый сигнал (для данной схемы) должен быть отрицательного напряжения. Затем в момент времени t_1 на вход интегратора подается опорное напряжение $E_{ОП}$ (противоположной полярности) и измеряется время интегрирования до 2х256 мкс.

Построение подсистемы аналогового ввода с использованием монолитных АЦП. При построении управляющих и измерительных систем можно использовать специализированные микросхемы ЦАП и АЦП. Микросхемы этих устройств должны подключаться к шинам микроЭВМ. В справочных материалах оговариваются не только параметры микросхем, но и описывается их структура. Принципиальным является наличие или отсутствие в структуре АЦП встроенного регистра, с помощью которого можно обеспечить подключение его к шинам МК. Если регистр во внутренней структуре АЦП отсутствует, то необходимо между выводами микросхемы и МК установить внешний регистр.

а)



б)

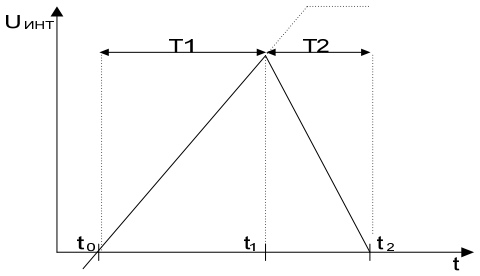


Рис. 9.7. Схема (а)и временная диаграмма (б) работы АЦП

Вход тактовой частоты можно использовать как управляющий. При этом он должен быть подключен к управляемому генератору. Схему генератора приводят на чертеже и показывают связи как с АЦП, так и с МК. Необходимо обратить внимание на усилители А1-А4. К ним могут быть предъявлены особые требования. Тип этих усилителей выбираем из справочника и обеспечиваем полную схему включения. Аналогичные требования предъявляются и ко всем остальным микросхемам.

Выбор АЦП. Чтобы правильно выбрать АЦП для конкретного применения, нужно знать обусловленные этим применением требования к его рабочим параметрам – разрешению, времени преобразования, допустимой погрешности и т. д. Эти требования определяются проектируемыми техническими характеристиками разрабатываемой системы сбора данных. Наиболее важными являются следующие характеристики:

1. Число аналоговых каналов.

2. Производительность. Учитывается как производительность всей системы, так и максимальная производительность отдельных каналов.

3. Расположение измерительных преобразователей (вблизи или в удалении от выходного терминала).

4. Точность преобразования.

5. Окружение. В частности, важно знать уровень электрических помех и диапазон изменения окружающей температуры.

6. Стоимость системы. Разработка системы обычно начинается с выбора ее конфигурации. Затем выясняются требования к рабочим характеристикам каждого компонента системы (рис. 9.8).

Точность АЦП. Требования к точности преобразователя вытекают из соответствующей технической характеристики разрабатываемой системы сбора данных с учетом погрешностей, вносимых всеми другими компонентами этой системы. Распространенная ошибка – выбор АЦП с разрешением, удовлетворяющим этому требованию по точности, поскольку фактическая точность преобразователя хуже того значения, на которое указывает разрешение, в силу наличия различных погрешностей преобразователя. Список вкладов основных погрешностей, называемый бюджетом погрешностей, помогает рассчитать реальную точность преобразователя.

Время преобразования. Требуемое от АЦП число преобразований, выполняемых за одну секунду, определяется проектируемой производительностью системы сбора данных, числом каналов и выбранной конфигурацией системы.

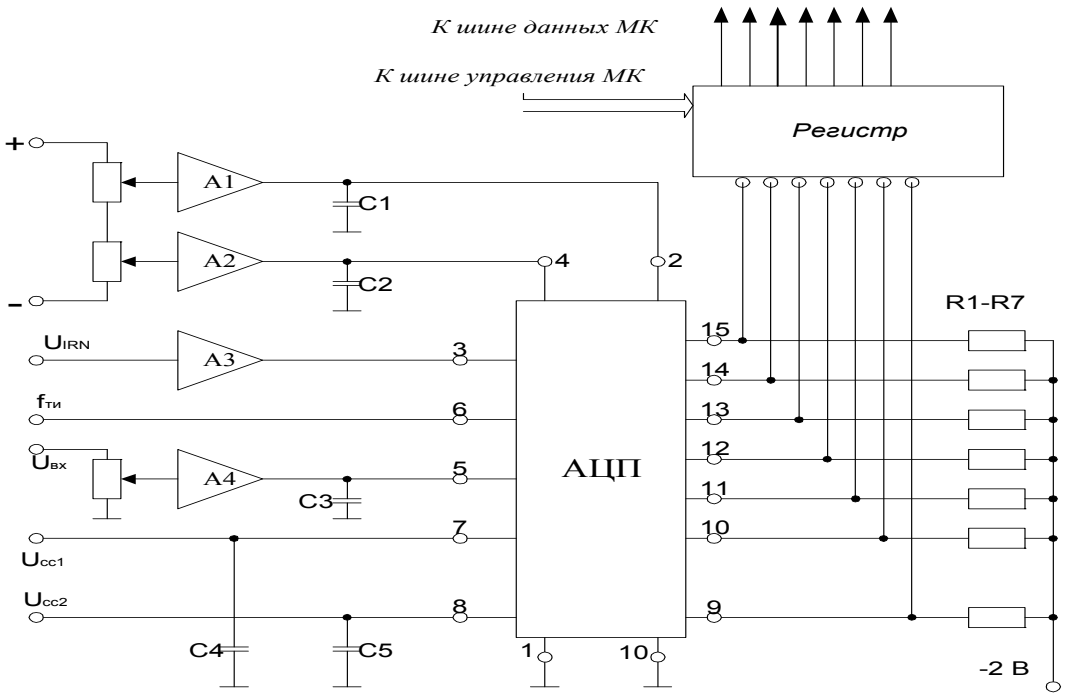


Рис 9.8. Подключение АЦП 1107ПВ3 к МК

Частота дискретизации по одному каналу равна производительности АЦП только в том случае, когда для каждого канала используется отдельный АЦП. Список всех временных задержек, связанных с одним преобразованием, называется временным бюджетом. Производительность АЦП рассчитывается, исходя из этого временного бюджета.

Тип АЦП. Для выбора типа АЦП обычно достаточно информации об используемой конфигурации системы, требуемом разрешении АЦП и времени преобразования. Например, для обеспечения среднего или высокого быстродействия следует выбрать АЦП последовательного приближения. Если одновременно требуется также высокое разрешение, то придется, по-видимому, применить АЦП, выполненный по гибридной технологии. При высоком разрешении, но низком быстродействии более подходящим будет двухтактный интегрирующий АЦП, которому следует отдать предпочтение и в тех случаях, когда нужно обеспечить высокую помехоустойчивость или ослабить

наводки с частотой 60 Гц. Аналогично в системах дистанционного сбора данных лучше всего использовать АЦП на основе преобразования напряжения в частоту, тогда как в сверхбыстродействующих системах сбора данных вне конкуренции будет АЦП параллельного преобразования.

Другие факторы. Выяснив, какой тип АЦП нам нужен, мы должны затем выбрать среди АЦП данного типа устройство, удовлетворяющее всем другим нашим требованиям. Например, диапазон температур, в котором предполагается использовать АЦП, определяет эксплуатационный класс выбираемого устройства: должно ли оно относиться к классу коммерческих устройств (диапазон рабочих температур 0 – 70 °С) или предназначается для промышленных (-25 – +85 °С) или военных (-55 – +125 °С) применений. Нужно проверить также входной диапазон устройства, его совместимость с биполярными входными сигналами, форму представления выходных цифровых данных (последовательная или параллельная) и, если это необходимо, возможность реализации интерфейса с микропроцессорами.

Рекомендации по использованию АЦП. Соблюдение некоторых простых правил при использовании АЦП будет гарантией того, что мы получим от него наибольшую отдачу.

Используйте полный входной диапазон АЦП. Если входной сигнал изменяется только от 1 до 3,5 В при использовании АЦП с входным диапазоном 0 – 5 В, погрешность преобразователя фактически удваивается. Чтобы предотвратить это неоправданное ухудшение рабочих характеристик преобразователя, используйте предварительное масштабирование сигнала для обеспечения максимально возможного соответствия диапазона его изменения и входного диапазона АЦП.

Используйте хорошие источники опорного сигнала. Температурный и временной дрейфы опорного сигнала проявляются как погрешность усиления и поэтому должны удерживаться на минимальном уровне. Прецизионный интегральный источник опорного сигнала – хороший выбор для большинства применений.

Обращайте внимание на скорость изменения входного сигнала. Изменения входного сигнала в течение времени преобразования приводят к погрешности усиления в АЦП последовательного приближения. Если характер изменения входного сигнала непредсказуем,

используйте УВХ. Модели УВХ общего назначения довольно дешевы. Используйте высококачественные полипропиленовые или полистирольные конденсаторы в качестве запоминающих конденсаторов в УВХ.

Применяйте отдельные общие провода для цифровых и аналоговых схем. Цифровые сигналы создают большие выбросы тока на общих проводах. Общие провода аналоговых и цифровых компонентов схемы должны быть отдельными и должны соединяться только в одной общей точке.

Добивайтесь минимизации помех и не забывайте о нагрузочных характеристиках схем. Стремитесь к уменьшению погрешностей входного аналогового сигнала, вызываемых земляными контурами, синфазными наводками и другими помехами, с помощью технических приемов, описанных в гл. 2. Вводите адекватное шунтирование (танталовый конденсатор емкостью 10 мкФ – для пульсации и керамический конденсатор емкостью 10 или 100 нФ – для импульсных помех) каждой ТТЛ ИС в цифровой части схемы. Не нагружайте управляющие линии более чем двумя ТТЛ БИС или используйте буферные схемы, которые имеются на выходе большинства АЦП, но может потребоваться дополнительная буферизация, если выходные линии, по которым передаются данные, имеют достаточно большую длину или если к выходу АЦП подключено несколько других устройств.

3. Выбор топологии сети

Первый шаг – выбор топологии связей проектируемой системы. Конфигурация, или топология, ЛВС определяет взаимное размещение станций сети и способ соединения между ними. Существуют следующие топологии ЛВС: шинная, кольцевая, звездообразная, петлевая, древовидная, гибридная и полносвязанная.

При выборе той или иной топологии надо руководствоваться соображениями минимизации аппаратных средств и простоты логического управления. Так, для шинной топологии среда передачи информации представляется в форме коммуникационного пути, доступного для всех рабочих станций, к которому они все должны быть подключены. Все рабочие станции могут непосредственно вступать в контакт с любой рабочей станцией, имеющейся в сети. Рабочие стан-

ции в любое время без прерывания работы всей вычислительной сети могут быть подключены к ней или отключены. Функционирование вычислительной сети не зависит от состояния отдельной рабочей станции.

В ЛВС кольцевой конфигурации сигналы передаются по кольцу в большинстве случаев только в одном направлении, т.е. от рабочей станции 1 к рабочей станции 2, от рабочей станции 3 к рабочей станции 4 и т.д. Последняя рабочая станция связана с первой. Коммуникационная связь замыкается в кольцо. Каждая станция непосредственно подсоединяется только к двум соседним углам и "прослушивает" передачу любой другой станции. Кольцо состоит из нескольких приемопередатчиков, соединенных физической средой. В кольцевой ЛВС может отсутствовать центральный управляющий узел и все станции имеют равные права доступа к физической среде. Однако во многих таких ЛВС одна из станций выполняет функции активного монитора, осуществляя инициацию, тестирование кольца, обнаружение и удаление искаженных или дублированных пакетов.

Специальной формой кольцевой топологии является логическая кольцевая сеть. Физически она монтируется как соединение звездных топологий. Отдельные „звезды” включаются с помощью специальных коммутаторов (Hub – концентратор). В зависимости от числа рабочих станций и длины кабеля между рабочими станциями применяют активные или пассивные концентраторы. Активные концентраторы дополнительно содержат усилитель для подключения от 4 до 16 рабочих станций. Пассивный концентратор является исключительно разветвляющим устройством (максимум на три рабочие станции). Управление отдельной рабочей станцией в логической кольцевой сети происходит так же, как и в обычной кольцевой сети. Каждой рабочей станции присваивается соответствующий ей адрес, по которому передается управление (от старшего к младшему и от самого младшего к самому старшему). Разрыв соединения происходит только для нижерасположенного (ближайшего) узла вычислительной сети, так что лишь в редких случаях может нарушаться работа всей сети.

Концепция топологии сети в виде звезды пришла из области больших ЭВМ, в которой головная машина получает и обрабатывает все данные с периферийных устройств как активный узел обработки данных. Такая конфигурация характерна также и для обычных вычис-

лительных сетей с терминальными устройствами или систем телеобработки данных. В ЛВС в центре „звезды” находится либо пассивный коммутатор, либо активное устройство, которое последовательно опрашивает станции и предоставляет им права на обмен данными.

Внимание. *Центральный узел такой ЛВС действует обычно не только как коммутатор, но и как преобразователь скоростей и протоколов.*

Наряду с известными топологиями вычислительных сетей – «кольцо», «звезда» и «шина» – на практике применяется и комбинированная, например древовидная, структура. Она образуется в основном в виде комбинаций вышеназванных топологий вычислительных сетей. Основание дерева вычислительной сети (корень) располагается в точке, в которой собираются коммуникационные линии информации (ветви дерева).

Вычислительные сети с древовидной структурой используют там, где невозможно непосредственное применение базовых сетевых структур в чистом виде. Для подключения большого числа рабочих станций соответственно адаптерным платам применяют сетевые усилители и/или коммутаторы. Коммутатор, обладающий одновременно и функциями усилителя, называют активным концентратором.

4. **Общая характеристика каналов связи**

Следующий шаг – выбор типа физической среды. Физически среда передачи данных представляет собой проводник (проводники), по которому передается информация с использованием различных видов кабелей (медная витая пара, коаксиальные, волоконно-оптические), а также эфир (радио-, микроволновые, инфракрасные каналы). В табл. 9.14 приведены основные параметры наиболее распространенных видов физической среды. *Симметричный* кабель состоит из оболочки (с экраном или без), внутри которой содержится одна пара проводников или несколько. Симметричные кабели используют для телефонной связи и при подключениях телексных терминалов. В ЛВС симметричные кабели применяются, как правило, в режиме передачи немодулированных сигналов, причем две пары проводников или более отводятся для передачи сигналов оповещения о предстоящей передаче данных.

Таблица 9.14

Сравнительные характеристики физических сред ЛВС

Характеристика	Тип физической среды		
Скорость передачи, Мбит/с	До 10	До 140	До 560
Дальность передачи, км	0,01 – 0,1	До 2,5	До 200
Число узлов в сети	10 – 100	До 100	2
Сложность соединения	Низкая	Средняя	Очень высокая
Возможность ответвления	Плохая	Средняя	Плохая
Возможность передачи различных видов информации	Низкая	Ограниченная	Очень хорошая
Помехозащищенность	Средняя	Высокая	Очень высокая

До недавнего времени одним из недостатков симметричных однопарных кабелей являлась низкая скорость передачи (до 1 Мбит/с). В настоящее время на таких кабелях достигнута скорость 10 Мбит/с и ожидается, что она может быть увеличена еще почти на порядок.

Симметричные многопарные кабели применяются в целом ряде ЛВС. Отдельные провода кабеля могут использоваться для разных целей (передачи данных, сигналов идентификации, индикации состояний и др.). Передача данных по нескольким параллельным линиям умножает пропускную способность всего кабеля (сотни мегабит в секунду) при сравнительно малой скорости передачи сигналов по одному проводу (десятки мегабит в секунду). Низкая скорость передачи сигналов снимает проблемы отражения сигналов и согласования импедансов, характерные для высокоскоростных линий, упрощает и удешевляет интерфейсные схемы, хотя и увеличивает их необходимое число.

Основными недостатками симметричных кабелей остаются простота несанкционированного доступа и чувствительность к электромагнитным помехам (при отсутствии экрана). Поэтому симметричные кабели применяются главным образом в кольцевых сетях, где используются повторители, имеется возможность стыковать различные типы кабелей и вставлять в критическом месте нечувствительную к помехам секцию кабеля.

Существует много разновидностей *коаксиального* кабеля с разными характеристиками. Коаксиальные кабели отличаются широкой полосой пропускания, обладают меньшим затуханием, более высокой устойчивостью к наводкам и т.д. Высококачественный кабель обладает большой жесткостью, и его трудно монтировать. Кроме того, электрические характеристики коаксиального кабеля (например, его волновое сопротивление в рабочем диапазоне частот составляет 50 – 75 Ом/м) делают его неудобным для многих целей. Но он очень удобен для передачи высокочастотных сигналов при сохранении относительной устойчивости к электрическим наводкам, а также для передачи модулированных и немодулированных сигналов. Кабель отличается надежностью, простотой конструкции и умеренной массой. В сетях кабельного телевидения используется коаксиальный кабель с полосой пропускания более 300 МГц, обеспечивающий передачу на большие расстояния. В режиме передачи немодулированных сигналов коаксиальный кабель позволяет передавать информацию со скоростью 10 Мбит/с. Эти свойства коаксиального кабеля обусловили его использование в качестве физической среды большинства ЛВС.

Волоконно-оптический кабель (световод) основан на использовании в качестве проводящей среды сверхпрозрачного стекловолокна. Теоретический предел пропускной способности световода определяется сотнями гигабит в секунду, а на практике уже достигнута скорость 2,41 Гбит/с [5].

Помимо высокой скорости передачи к достоинствам световода следует отнести его высокую помехозащищенность, защищенность от несанкционированного доступа и небольшую массу. К его недостаткам относятся высокая стоимость и сложность подключения новых станций из-за неразвитости технологии волоконно-оптических разветвителей, по световодам нельзя передавать электрическую энергию для повторителей и других устройств. Подключение ответвителей вызывает значительное ослабление сигналов. Сигналы могут передаваться по кабелю только в одном направлении. Световоды применяются в ЛВС кольцевой и звездообразной конфигурации. Они наиболее подходят для взаимосвязей больших ЭВМ, где требуются высокие скорости передачи. В то же время их применение для работы при средних и низких скоростях не считается целесообразным.

Новым типом физической среды для ЛВС является эфир, в котором могут быть организованы радиоканалы, инфракрасные и микроволновые каналы.

Радиоканал наиболее пригоден для обслуживания мобильных станций. В стационарных ЛВС радиоканалы используются редко из-за экранированности зданий и узкой полосы доступных радиочастот.

Инфракрасный канал обеспечивает высокие скорости передачи (несколько мегабит в секунду) на расстояние прямой видимости. В отличие от радиоканалов он нечувствителен к электромагнитным помехам и не занимает полосы частот радио- или видеосигналов. К недостаткам инфракрасного канала следует отнести небольшую дальность передачи.

Микроволновый канал обеспечивает еще более высокие скорости (до 20 Гбит/с) на расстояния 15 - 20 км (при прямой видимости).

Внимание. *Выбирая физическую среду, следует обеспечить заданные параметры, учитывающие такие особенности, как расстояние, скорость передачи данных, параметры самой среды. Необходимо привести таблицу сопоставительных параметров для альтернативных вариантов и доказать, что выбираемая среда отвечает заданию на проектирование.*

5. Коммуникационное оборудование

Наличие коммуникационного оборудования продиктовано особенностями как физической среды, так и организации системы связей. Наиболее употребительны повторители сигналов и коммутаторы с различной архитектурой, определяемой выполняемыми функциями.

Репитеры (повторители). При распространении сигнала вдоль линии связи его форма искажается: «заваливаются» фронты, уменьшается амплитуда, появляются «наложения» из-за влияния внешних помех, отражений и т.д. Для восстановления формы сигнала используются ретрансляторы, в простейшем случае обычно это логические элементы ТТЛ с повышенной нагрузочной способностью и с гистерезисной передаточной характеристикой вход-выход. Пример выполнения схем трансляции сигнала представлен на рис. 9.9. В качестве ретранслятора могут выступать коммутатор, хаб и другие активные устройства, используемые для реализации системы связей. В этом случае перечисленные устройства выполняют функции ретранслятора. При разработке функциональной и электрической принципиаль-

ной схем необходимо помнить, что информация по каналам связи передается в двух направлениях по одним и тем же линиям независимо от типа физической среды.

Внимание. *Проблема построения системы ретрансляторов, функционально эквивалентных обычному проводу, не так проста, как это может показаться на первый взгляд.*

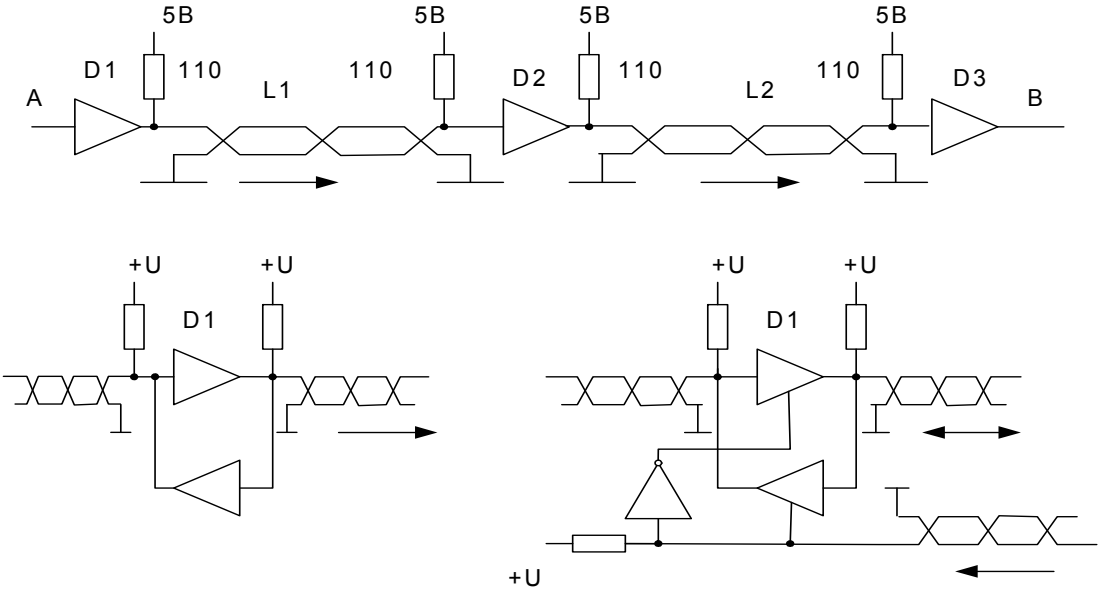


Рис. 9.9. Пример реализации повторителя

Коммутаторы. Несмотря на то что в коммутаторах работают известные и хорошо отработанные алгоритмы прозрачных мостов и мостов с маршрутизацией от источника, существует большое разнообразие моделей коммутаторов. Они отличаются как внутренней организацией, так и набором выполняемых дополнительных функций, таких как трансляция протоколов, поддержка алгоритма покрывающего дерева, образование виртуальных логических сетей и ряда других.

В настоящее время коммутаторы используют в качестве базовой одну из трех схем, на которой строится такой узел обмена, а именно:

- коммутационную матрицу;
- разделяемую многовходовую память;
- общую шину.

Часто эти три способа взаимодействия комбинируются в одном коммутаторе. При выборе коммутатора приводится его функциональная схема и описание работы (рис. 9.10).

Внимание. *При разработке электрической принципиальной схемы следует учесть, что передача информации осуществляется в двух направлениях, при этом число устройств, претендующих на передачу информации по одним и тем же линиям, может быть значительным.*

Оптические приемопередатчики. Разрабатываемая ВОСП должна обеспечить передачу электрического сигнала без искажений или с их допустимыми уровнями. К основным искаженим, которые могут возникнуть в аналоговой ВОСП, относятся нелинейные и линейные искажения.

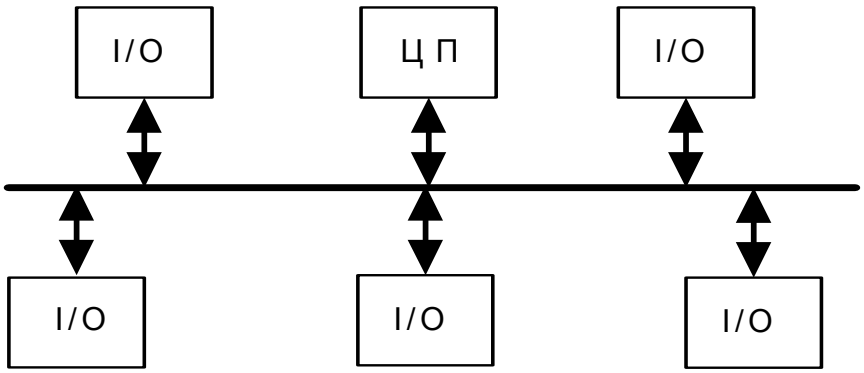


Рис. 9.10. Коммутатор на процессоре общего назначения

Нелинейные искажения в наших условиях приводят к ухудшению отношения сигнал/шум, т. е. к ухудшению чувствительности, а также к появлению ложных сигналов приема.

Линейные искажения приводят также к ухудшению отношения сигнал/шум. Наиболее опасными искажениями являются нелинейные, которыми и будет определяться динамический диапазон ВОСП, особенно интермодуляционные искажения, создающие помехи с частотами (mfi + nfj). Поэтому выбор структуры, схематических решений составляющих узлов будет направляться на обеспечение минимизации собственных шумов и нелинейных искажений.

Для передачи информации по оптическому волокну необходимо изменение параметров оптической несущей в зависимости от изменений исходного сигнала. Этот процесс называется модуляцией.

Наиболее простым с точки зрения реализации видом модуляции является прямая модуляция оптической несущей по интенсивности на основе полупроводникового источника излучения. Здесь исходный

сигнал через усилитель подаётся на базу транзистора, в коллектор которого включен излучатель. Устройство смещения позволяет выбрать рабочую точку на ваттамперной характеристике излучателя.

Оптический передатчик. Структурная схема оптического передатчика прямой модуляции, приведенная на рис. 9.11, является оптимальной, так как наиболее рационально реализует все функциональные возможности и достоинства выбранного вида модуляции. Преобразователь кода ПК преобразует стыковой код в код, используемый в линии, после чего сигнал поступает на модулятор. Схема оптического модулятора исполняется в виде передающего оптического модуля (ПОМ), который помимо модулятора содержит схемы стабилизации мощности и частоты излучения полупроводникового лазера или светоизлучающего диода.

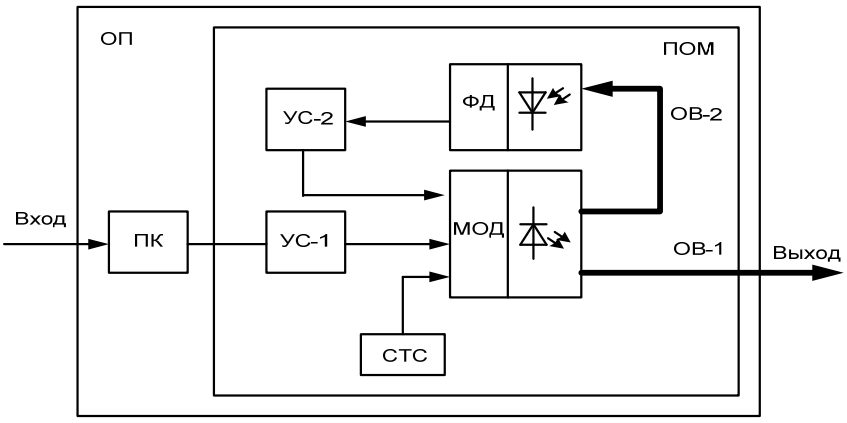


Рис. 9.11. Структурная схема оптического передатчика

Здесь модулирующий сигнал через дифференциальный усилитель УС-1 поступает в прямой модулятор с излучателем (МОД). Модулированный оптический сигнал излучается в основное волокно ОВ-1. Для контроля мощности излучаемого оптического сигнала используется фотодиод (ФД), на который через вспомогательное волокно ОВ-2 подается часть излучаемого оптического сигнала. Напряжение на выходе фотодиода, отображающее все изменения оптической мощности излучателя, усиливается усилителем УС-2 и подается на инвертирующий вход усилителя УС-1. Таким образом создается петля отрицательной обратной связи, охватывающая излучатель. Благодаря введению ООС обеспечивается стабилизация рабочей точки излучателя. При повышении температуры энергетическая характеристика лазерного диода смещается и при отключенных цепях стабилизации мощности уровень оптической мощности при передаче «0» (Р0) и при пе-

редаче «1» (Р1) уменьшаются, разность тока смещения $I_б$ и порогового тока $I_п$ увеличивается, а разность Р1 - Р0 уменьшается. После времени установления переходных процессов в цепях стабилизации устанавливаются новые значения $I_б$ и $I_п$ и восстанавливаются прежние значения Р1 - Р0 и Рср. Для уменьшения температурной зависимости порогового тока в передающем оптическом модуле имеется схема термостабилизации (СТС), поддерживающая мощность излучения передающего оптического модуля постоянной при изменении температуры от номинального значения.

Оптический приемник. Структурная схема оптического приемника (ОПр) показана на рис. 9.12. Приемник содержит фотодетектор (ФД) для преобразования оптического сигнала в электрический, малошумящий усилитель (УС) для усиления полученного электрического сигнала до номинального уровня.

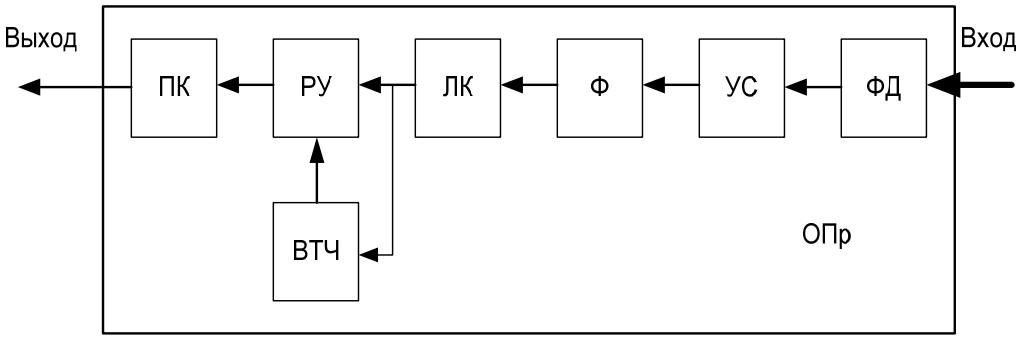


Рис. 9.12. Структурная схема оптического приемника

Усиленный сигнал через фильтр (Ф), формирующий частотную характеристику приемника, обеспечивающую квазиоптимальный прием, поступает в устройство линейной коррекции (ЛК). В линейной коррекции компенсируются частотные искажения электрической цепи на стыке фотодиода и первого транзистора усилителя. После преобразований сигнал поступает на вход решающего устройства (РУ), где под действием тактовых импульсов, поступающих от устройства выделения тактовой частоты (ВТЧ), формируется решение о принятом символе. На выходе оптического приёмника имеется преобразователь кода (ПК), преобразующий линейный код в стыковой.

6. Логическая организация сети

Пожалуй, самый сложный момент в организации сетевого обмена информации связан с логической организацией сети, под которой будем понимать процедуры переноса информации от источника при-

емнику. Эта процедура носит название „протокол”. Данная процедура предполагает несколько видов передаваемой информации. Первый вид – фактические данные, необходимые для задания работы устройств (скорость, положение, ускорение и т.д.). Второй вид – команды, обеспечивающие режимы работы оборудования (пуск, стоп, реверс, пошаговый режим, режим разгона и торможения и т.д.). Третий – служебная информация, сопровождающая процедуру переноса информации (начало и конец кадра, контрольная сумма, квитирование, число передаваемых байт, адрес источника и приемника, адрес коммутатора и т.п.). Еще один вид – это информация о нештатных ситуациях в системе (сбой информации при передаче, авария устройства, отсутствие ответа о подтверждении приема и т.д.).

Логически передача данных между конечной точкой и ПО производится с помощью выделения канала и обмена данными по этому каналу, а с точки зрения представленных уровней передача данных выглядит следующим образом (рис. 9.13).

Рассмотрим в качестве примера фрагменты протокола с использованием шины USB:

1. Клиентское ПО посылает IPR-запросы на уровень системного драйвера USB.

2. Системный драйвер USB разбивает запросы на транзакции по следующим правилам:

- выполнение запроса считается законченным, когда успешно завершены все транзакции, его составляющие;
- все подробности отработки транзакций (такие как ожидание готовности, повтор транзакции при ошибке, неготовность приемника и т.д.) до клиентского ПО не доводятся;
- ПО может только запустить запрос и ожидать или выполнения запроса, или выхода по тайм-ауту;
- устройство может сигнализировать о серьезных ошибках, что приводит к аварийному завершению запроса, о чем уведомляется источник запроса.

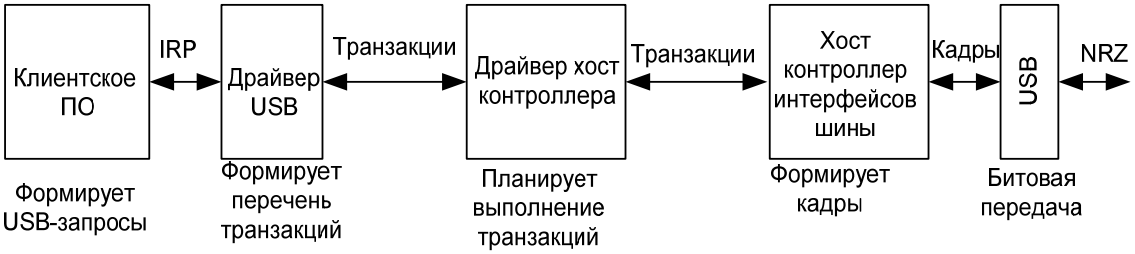


Рис 9.13. Уровни передачи данных

3. Драйвер контроллера хоста принимает от системного драйвера USB перечень транзакций и выполняет следующие действия:

- планирует исполнение полученных транзакций, добавляя их к списку транзакций;
- извлекает из списка очередную транзакцию и передает её уровню хост-контроллера интерфейса шины USB;
- отслеживает состояние каждой транзакции вплоть до ее завершения.

4. Хост-контроллер интерфейса шины USB формирует кадры.

5. Кадры передаются последовательной передачей бит по методу, называемому NRZI *with bit stuffing* (Non Return to Zero Invert, метод невозврата к нулю с инвертированием единиц).

Таким образом, можно сформировать следующую *упрощенную* схему (рис. 9.14).

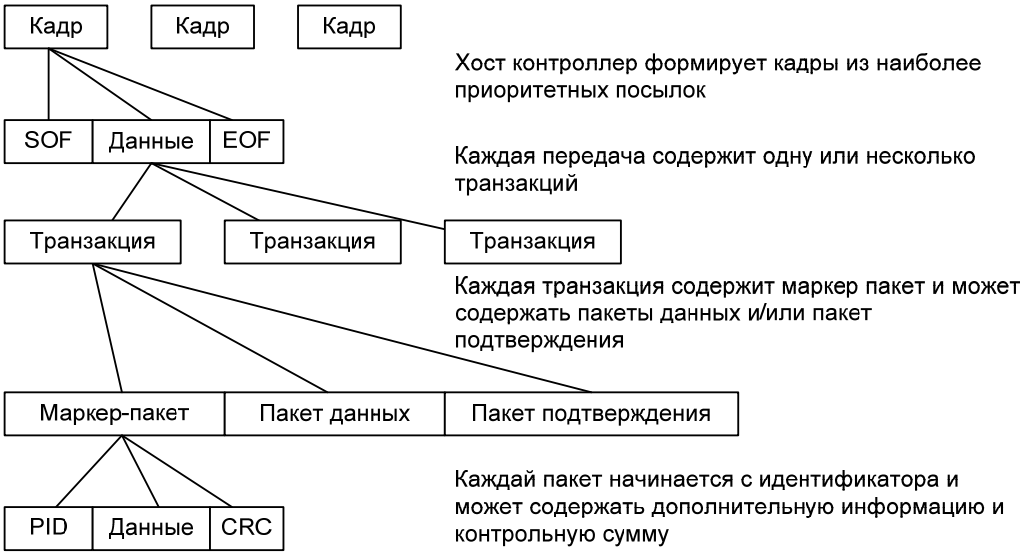


Рис. 9.14. Общая схема составляющих USB-протокола

Характеристика USB-протокола:

❒ каждый кадр состоит из наиболее приоритетных посылок, состав которых формирует драйвер контроллера хоста;

❒ каждая передача состоит из одной или нескольких транзакций;

❒ каждая транзакция состоит из пакетов;

❒ каждый пакет состоит из идентификатора пакета, данных (если они есть) и контрольной суммы.

Любой обмен по шине USB инициируется хостом. Он организует обмены с устройствами согласно своему плану распределения ре-

сурсов. Контроллер циклически (с периодом 1,0 + 0,0005 мс) формирует *кадры* (frames), в которые укладываются все запланированные передачи (рис. 9.15). Каждый кадр начинается с посылки маркер-пакета SOF (Start Of Frame», начало кадра), который является синхронизирующим сигналом для всех устройств, включая хабы. В конце каждого кадра выделяется интервал времени EOF (End Of Frame, конец кадра), во время которого хабы запрещают передачу по направлению к контроллеру. Если хаб обнаружит, что с какого-то порта в это время ведется передача данных, то он отключит этот порт.

В режиме высокоскоростной передачи пакеты SOF передаются в начале каждого *микрокадра* (период 125±0,0625 мкс).

Рис. 9.15. Поток кадров USB

Хост планирует загрузку кадров так, чтобы в них всегда находилось место для наиболее приоритетных передач, а свободное место кадров заполняется низкоприоритетными передачами больших объемов данных. Спецификация USB позволяет занимать под периодические транзакции (изохронные и прерывания) до 90 % пропускной способности шины. Каждый кадр имеет свой номер. Хост оперирует 32-битным счетчиком, но в маркере SOF передает только младшие 11 бит. Номер кадра циклически увеличивается во время EOF.

Для изохронной передачи важна синхронизация устройств и контроллера. Есть три варианта синхронизации:

- ❒ синхронизация от внутреннего генератора устройства с маркерами SOF;
- ❒ подстройка частоты кадров под частоту устройства;
- ❒ согласование скорости передачи (приема) устройства с частотой кадров.

В каждом кадре может быть выполнено несколько транзакций, их допустимое число зависит от скорости, длины поля данных каждой из них, а также от задержек, вносимых кабелями, хабами и USB-устройствами.

Все транзакции кадров должны быть завершены до момента времени EOF. Частота генерации кадров может немного варьироваться с помощью специального регистра хоста, что позволяет подстраивать частоту для изохронных передач. Подстройка частоты кадров контроллера возможна под частоту внутренней синхронизации только одного USB-устройства.

Канал (pipe) – это логическое соединение между конечной точкой USB-устройства и ПО хоста (рис. 9.16).

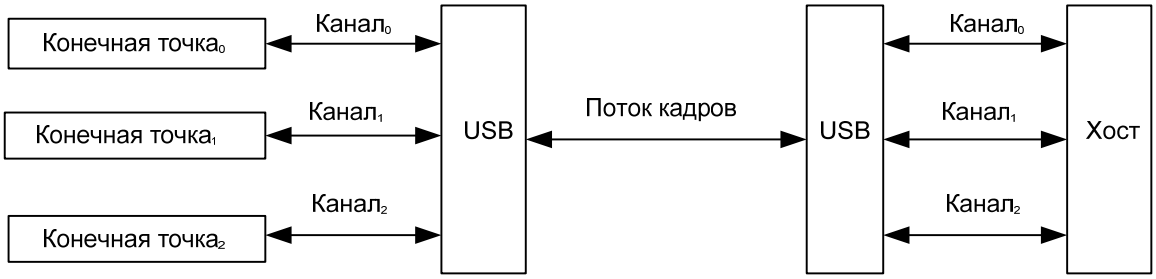


Рис. 9.16. Каналы USB

Существуют две модели каналов:

❒ *потоковый канал (*или просто *поток,* streaming *pipe)* – это канал для передачи данных, структура которых определяется клиентским ПО. Потоки используются для передачи массивов данных, передачи данных по прерываниям и изохронной передачи данных. Поток всегда однонаправленный. Один и тот же номер конечной точки может использоваться для двух разных потоковых каналов – ввода и вывода. Передачи данных в потоковых каналах подчиняются следующим *правилам синхронизации:*

• запросы клиентских драйверов для разных каналов, поставленные в определенном порядке относительно друг друга, могут выполняться в другом порядке;

• запросы для одного канала будут исполняться строго в порядке их поступления;

• если во время выполнения какого-либо запроса происходит серьезная ошибка (STALL), поток останавливается.

Канал сообщений (message pipe или control pipe) – это канал для передачи данных, структура которых определяется спецификацией USB. Каналы этого типа двунаправленные, применяются для передачи управляющих посылок. Каналы сообщений строго синхронизированы - спецификация USB запрещает одновременную обработку нескольких запросов: нельзя начинать передачу нового сообщения, пока

не завершена обработка предыдущего. В случае возникновения ошибки передача сообщения может быть прервана хостом, после чего хост может начать передачу нового сообщения.

Основными характеристиками каналов являются:

- ❒ полоса пропускания канала;
- ❒ используемый каналом тип передачи данных;
- ❒ характеристики, соответствующие конечной точке: направление передачи данных и максимальный размер пакета.

Полоса пропускания шины делится между всеми установленными каналами. Выделенная полоса закрепляется за каналом, и если установление нового канала требует такой полосы, которая не вписывается в уже существующее распределение, запрос на выделение канала отвергается. Архитектура USB предусматривает внутреннюю буферизацию всех USB-устройств, причем чем большая полоса пропускания требуется, тем больше должен быть буфер. Шина должна обеспечивать обмен с такой скоростью, чтобы задержка данных в устройстве, вызванная буферизацией, не превышала нескольких миллисекунд.

Канал сообщений, связанный с нулевой конечной точкой, называется *основным каналом сообщений* (default control pipe или control pipe 0). Владельцем этого канала является USBD, и он используется для конфигурирования USB-устройства. Основной канал сообщений поддерживает только управляющие передачи. Остальные каналы (они называются *клиентскими каналами,* client pipe) создаются в процессе конфигурирования. Их владельцами являются драйверы USB-устройств. По клиентским каналам могут передаваться как потоки, так и сообщения с помощью любых типов передач.

Набор клиентских, каналов, с которыми работает драйвер USB-устройства, называется *интерфейсом устройства,* или *связкой клиентских каналов* (pipe 's bundle).

Информация по каналу передается в виде *пакетов* (packet, рис. 9.17). Каждый пакет начинается с *поля синхронизации* SYNC (SYNChroniuzation), за которым следует *идентификатор пакета* PID (Packet Identifier), значения которого приведены в табл. 9.15. Поле check представляет собой побитовую инверсию PID.

SYNC PID Check Данные пакета EOP

Рис. 9.17. Структура пакета

Таблица 9.15

Список кодов PID

Обозначение	Код PID	Источник	Описание
Идентификаторы маркер-пакетов (Token Packet)			
OUT	0001b	Хост	Маркер транзакции вывода, передает адрес и номер конечной точки при передаче от хоста к функции
IN	1001b	То же	Маркер транзакции ввода, передает адрес и номер конечной точки при передаче от функции к хосту
SOF	0101b	„	Маркер начала кадра, содержит номер кадра
SETUP	1101b	„	Маркер транзакции управления: передает адрес и номер конечной точки при передаче команды от хоста к функции
Идентификаторы пакетов данных (Data Packet)			
Data0	0011b	Хост, USB-устройство	Пакеты данных с четным и нечетным PID, чередуются для точной идентификации подтверждений
Data1	1011b	То же	То же
Data2	0111b	„	Дополнительные типы пакетов данных, используемые в транзакциях с широкополосными изохронными точками (в USB .2.0 для HS)
MData	1111b	”	То же
Идентификаторы пакетов подтверждений (Handshake)			
ACK	0010b	Хост, USB-устройство	Подтверждение безошибочного приема пакета
NAK	1010b	USB-устройство	Приемник не сумел принять или передатчик не сумел передать данные. Может использоваться для управления потоком данных ("ответ на запрос не готов"). В транзакциях прерываний является признаком отсутствия необслуживаемых прерываний

Окончание табл. 9.15

Обозначение	Код PID	Источник	Описание
STALL	1110b	USB-устройство	Произошел сбой в конечной точке или запрос не поддерживается, требуется вмешательство хоста
NYET	0110b	USB-устройство	Подтверждение безошибочного приема, но указание на отсутствие места для приема следующего пакета максимального размера (USB 2.0)
Идентификаторы специальных пакетов (Special Packet)			
PRE	1100b	Хост	Специальный маркер, сообщающий, что следующий пакет будет передаваться в режиме LS (разрешает трансляцию данных на низкоскоростной порт хаба)
ERR		USB-устройство, хаб	Сигнализация ошибки в расщепленной транзакции (USB 2.0)
SPLIT (SS/CS)	1000b	Хост	Маркер расщепленной транзакции (USB 2.0) В зависимости от назначения обозначается как SS (маркер запуска) и CS (маркер завершения), назначение определяется битом SC в теле маркера
PING	0100b	Хост	Пробный маркер высокоскоростного управления потоком (USB 2.0)
RESERV	0000b		Зарезервированный PID

Из табл. 9.15 видно, что два младших бита идентификатора определяют группу, к которой он принадлежит:

- 00 — специальный пакет;
- 01 — маркер-пакет;
- 10 — пакет-подтверждение;
- 11 — пакет данных.

Структура данных пакета зависит от группы, к которой он относится.

Формат маркер-пакетов IN, OUT, SETUP и PING

Поле данных пакетов типа IN, OUT. SETUP и PING содержит следующие поля (полный формат пакета показан на рис. 9.18):

- (7) адрес функции;
- (4) адрес конечной точки;
- (5) циклический контрольный код.

Маркер транзакции отмечает начало очередной транзакции на шине USB и позволяет адресовать до 127 функций USB (нулевой адрес используется для конфигурирования) и по 16 конечных точек в каждой функции. Поле CRC вычисляется по полям Func и EndP.

SYNC PID (4) Check (4) Func (7) EndP (4) CRC(5) EOP

Рис. 9.18. Формат пакета типа IN, OUT, SETUP и PING

Формат пакета SOF. Поле данных пакета SOF содержит (полный формат пакета показан на рис. 9.19.): (11) номер кадра; (5) циклический контрольный код.

Пакет SOF используется для отметки начала кадра. Хотя хост оперирует 32-битным счетчиком кадров, в маркере SOF передаются только младшие 11 бит. Контрольная сумма вычисляется по 11 битам поля Frame.

SYNC PID(4) Check(4) Frame(11) CRC(5) EOP

Рис. 9.19. Формат пакета типа SOF

Формат пакета данных

В поле данных пакетов типа Data0, Datal, Data2 и MData может содержаться от 0 до 1023 байт данных, за которыми следует 16-разрядный циклический контрольный код, вычисленный по полю Data. Пакет данных всегда должен посылать целое число байт. Для LS режима максимальный размер пакета равен 8, для FS – 1023, а для HS – 1024 байтам. Полный формат пакета данных показан на рис. 9.20.

SYNC PID(4) Check(4) Data(0.8192) CRC(16) EOP

Рис. 9.20. Формат пакетов данных

4. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСОВ ВНЕШНЕЙ СВЯЗИ

Для объединения вычислительных устройств с ЭВМ используются встроенные в нее интерфейсы. К ним относятся последовательный порт RS-232, USB 2.0, CAN-bus шина и другие. Особенность проектируемой системы такова, что управление процессами идет по инициативе ЭВМ, а при заданном удалении единственным средством является использование экономичных по числу проводников систем связи. Для RS-232, USB 2.0, CAN-bus имеются ограничения по удалению от ЭВМ. Избежать этого можно, если воспользоваться интерфейсами RS-485 или разновидностью RS-422. Тогда надо обеспечить переход от RS-232, USB 2.0, CAN-bus к RS-485 или RS-422, разработать схему преобразования одного протокола в другой. Отметим особенности построения двух вышеназванных интерфейсов.

1. СТАНДАРТ RS-232C

Стандартный интерфейс RS-232 был первоначально разработан для сопряжения терминалов или оконечного оборудования данных (ООД) с модемом (модулятором/демодулятором) или аппаратурой передачи данных (АПД). В настоящее время этот интерфейс используется для сопряжения практически любого устройства с персональными компьютерами IBM PC, а также с аналогичными компьютерами других типов.

В табл. 9.16 приведены функции отдельных выводов соединителя DB25, предусмотренного в последовательном связном адаптере компьютера IBM PC.

Таблица 9.16

Функции выводов соединителя DB25

Номер вывода	Направление передачи	Функция вывода
1	–	Защитное заземление
2	Вывод	Передаваемые данные (— TXD)
3	Ввод	Принимаемые данные (—RXD)
4	Вывод	Запрос передатчика (RTS)
5	Ввод	Сброс передатчика (CTS)
6	Ввод	Готовность модема (DSR)
7	–	Сигнальная земля (SG)

Окончание табл. 9.16

8	Вывод	Обнаружение несущей сигнала (DCD)
20	Вывод	Готовность терминала (DTR)
22	Ввод	Указатель вызовов (RI)
24	–	Не задействован

Вывод 1. Защитное заземление (PG) соединяет между собой корпуса полупроводниковых приборов с целью предотвращения накопления статического заряда. Этот вывод присоединяется к экрану кабеля связного адаптера IBM.

Вывод 2. Передаваемые данные (-TXD); для случая ООД через этот вывод сигнал передается в последовательную линию данных. В АПД вывод 2 служит для приема данных (-RXD).

Вывод 3. Принимаемые данные (-RXD); для случая ООД через этот вывод сигнал принимается с последовательной линии. В АПД вывод 3 служит для передачи данных (-TXD).

Вывод 4. Запрос передатчика (RTS) возбуждается передатчиком, когда последний должен передать данные по линии. Вывод 4 должен сохранять активное состояние до конца передачи.

Вывод 5. Сброс передатчика (CTS) используется приемником для информирования передатчика относительно того, готов ли приемник к приему передаваемых данных. Вывод 5 также должен находиться в активном состоянии на протяжении всей передачи. Если сигнал CTS переходит в неактивное состояние в середине передачи, то передатчик прекращает передачу и выдается сообщение об ошибке.

Вывод 6. Готовность модема (DSR) задает режим модема. Когда модем включен, этот вывод обычно находится в активном состоянии. В случае модемов с фиксированной коммутацией, которые способны работать в режиме передачи либо речи, либо данных, на выводе 6 устанавливается низкий (активный) уровень при работе в режиме передачи данных и высокий – при передаче речи.

Вывод 7. Сигнальное заземление (SG) является заземлением цепей сигнала, передаваемого на вывод -TXD.

Вывод 8. Указатель принятого по линии сигнала обычно называют признаком обнаружения несущей сигнала (DCD), или указателем несущей (CD). Этот сигнал используется модемом для информирования передатчика о том, что каналом передачи можно пользоваться, и обычно активизируется в тех случаях, когда уже выдан сигнал «запрос передатчика».

Вывод 20. Готовность терминала (DTR) – сигнал от терминала, указывающий, что последний находится в режиме взаимодействия с системой и, следовательно, связь возможна. Если этот сигнал установлен в 1, то терминал находится в автономном (или локальном) режиме и связь с ним как с внешним устройством невозможна.

Вывод 22. Указатель вызовов (RI) активизируется модемом, когда последний обнаруживает поступивший по телефонной линии вызов. Этот сигнал используется устройствами, которые могут автоматически отвечать на поступающие вызовы.

Выводы 9, 11, 18 и 25 соединителя асинхронного адаптера зарезервированы для интерфейса типа «токовая петля» и не используются.

Внимание. *Описание этого интерфейса – одно из немногих мест, где соглашение относительно представления логических 1 и 0 электрическими сигналами не совпадает с их стандартным представлением.*

Например, в ТТЛ-логике «1» определяется уровнем напряжения от +2,4 до +5 В, а «0» — от 0 до 0,8 В. Схемы на ТТЛ-элементах не имеют достаточного запаса помехоустойчивости для работы на длинные кабели в среде с высокой интенсивностью шумов.

В случае стандартного интерфейса RS-232C уровень напряжения в пределах от +5 до +15 В считается высоким и представляет логический «0». Уровень, лежащий в пределах от -5 до -15 В, считается низким и соответствует логической «1». Логическая «1» – это истинное состояние сигнала, а логический «0» – ложное состояние вне зависимости от используемого уровня напряжения. При указании уровней подразумевалось, что схема формирования нагружена надлежащим образом. Уровни, соответствующие ненагруженному формирователю, могут меняться в пределах ±25 В. Приемник воспринимает сигналы с напряжениями от +3 до +25 В и от -3 до –25 В. Широкие диапазоны изменения напряжения сигнальных уровней и неопределенная область в пределах ±3 В выбраны с целью минимизации электрических шумов при работе с длинными кабелями. Такой выбор обеспечивает надежную работу при расстоянии между терминалом и модемом порядка 15 м.

Сигнал со знаком «+» в начале его обозначения или без знака (например, +RTS или RTS) имеет активный высокий уровень. Сигнал

со знаком « − » в начале обозначения (например, −TxD) имеет активный низкий уровень. Так, сигнал +RTS имеет уровень от +5 до +15 В (т.е. высокий уровень), когда он активен, а сигнал -TxD характеризуется уровнем от -5 до -15 В (т.е. низким), когда он находится в состоянии логическая «I».

Все сигналы квитирования установления связи представляются активными нулями (положительными напряжениями). Когда вход квитирования не используется, он должен быть связан с точкой, имеющей статическое напряжение, которое находится в диапазоне логического нуля (т.е. от +3 до +25 В). Каждое отдельное устройство, как правило, не работает со всеми сигналами, выведенными на контакты соединителя. Так, соединитель RS-232 асинхронного адаптера компьютера IBM PC имеет и контакты со стандартными уровнями напряжения и контакты, связанные с нестандартными сигнальными цепями передатчика и приемника, позволяющими реализовать «токовую петлю». Очевидно, что для обеспечения сопряжения следует соединять только линии, согласующиеся по уровню сигналов.

2. Стандарт USB

Разработчики шина USB (Universal Serial Bus, универсальная последовательная шина) ориентировались на создание интерфейса, обладающего следующими свойствами:

- передавать данные со скоростью до 12 Мбит/с (480 Мбит/с для USB 2.0);
- гибкость протокола смешанной передачи изохронных данных и асинхронных сообщений;
- интеграция с выпускаемыми устройствами;
- обеспечение стандартного интерфейса;
- создание новых классов устройств, расширяющих ПК.

Обычная архитектура USB подразумевает подключение одного или нескольких USB-устройств к компьютеру (рис. 9.22), который в такой конфигурации является главным управляющим устройством и называется *хостом.* Подключение USB-устройств к хосту производится с помощью *кабелей.* Для соединения компьютера и USB-устройства используется *хаб.*

Физическая архитектура USB определяется следующими правилами (рис. 9.23):

- устройства подключаются к хосту;
- физическое соединение устройств между собой осуществляется по топологии многоярусной „звезды”, вершиной которой является корневой хаб;
- центром каждой „звезды” является хаб;
- каждый кабельный сегмент соединяет между собой две точки: хост с хабом или функцией (см. далее), хаб с функцией или другим хабом;
- к каждому порту хаба может подключаться периферийное USB-устройство или другой хаб, при этом допускается до 5 уровней каскадирования хабов, не считая корневого.

Логическая архитектура выгладит как обычная „звезда”, центром которой является прикладное ПО, а вершинами — набор *конечных точек.*

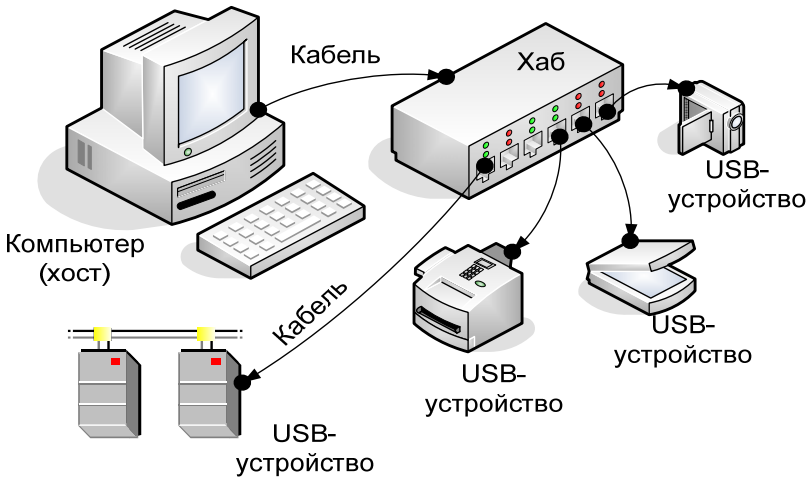


Рис. 9.22. Архитектура USB

Шина USB состоит из следующих элементов:

- х*ост-контроллер* (host controller) — это главный контроллер, который входит в состав системного блока компьютера и управляет работой всех устройств на шине USB. Для краткости мы будем писать просто хост. На шине USB допускается наличие только одного хоста. Системный блок персонального компьютера содержит один или несколько хостов, каждый из которых управляет отдельной шиной USB;
- *устройство* (device) может представлять собой хаб, функцию или их комбинацию (compound device);
- *порт* (port) — точка подключения;
- *хаб* (hub другое название — *концентратор)*— устройство,

которое обеспечивает дополнительные порты на шине USB. Другими словами, хаб преобразует один порт *(восходящий порт,* upstream ports) во множество портов *(нисходящие порты,* downstream ports). Архитектура допускает соединение нескольких хабов (не более 5). Хаб распознает подключение и отключение устройств к портам и может управлять подачей питания на порты. Каждый из портов может быть разрешен или запрещен и сконфигурирован на полную или ограниченную скорость обмена. Хаб обеспечивает изоляцию сегментов с низкой скоростью от высокоскоростных. Хаб может ограничивать ток, потребляемый каждым портом.

Существуют асинхронный и блочный методы передачи данных. Блок передаваемых данных называется *USB-фреймом, или USB-кадром,* и передается за фиксированный временной интервал. Оперирование командами и блоками данных реализуется при помощи логической абстракции, называемой *каналом.* Внешнее устройство также делится на логические абстракции, называемые *конечными точками.* Таким образом, канал является логической связкой между хостом и конечной точкой внешнего устройства. Канал можно сравнить с открытым файлом.

Для передачи команд (и данных, входящих в состав команд) используется канал по умолчанию, а для передачи данных открываются либо *потоковые каналы*, либо *каналы сообщений.*

Внимание. *Все операции по передаче данных по шине USB инициируются хостом. Периферийные USB-устройства сами начать обмен данными не могут. Они могут только реагировать на команды хоста*.

Для шины USB настоящего механизма прерываний (как, например, для последовательного порта) не существует. Вместо этого хост опрашивает подключенные устройства на предмет наличия данных о прерывании. Опрос происходит в фиксированные интервалы времени, обычно каждые 1 – 32 мс. Устройству разрешается посылать до 64 байт данных.

С точки зрения драйвера, возможности работы с прерываниями фактически определяются хостом, который и обеспечивает поддержку физической реализации USB-интерфейса.

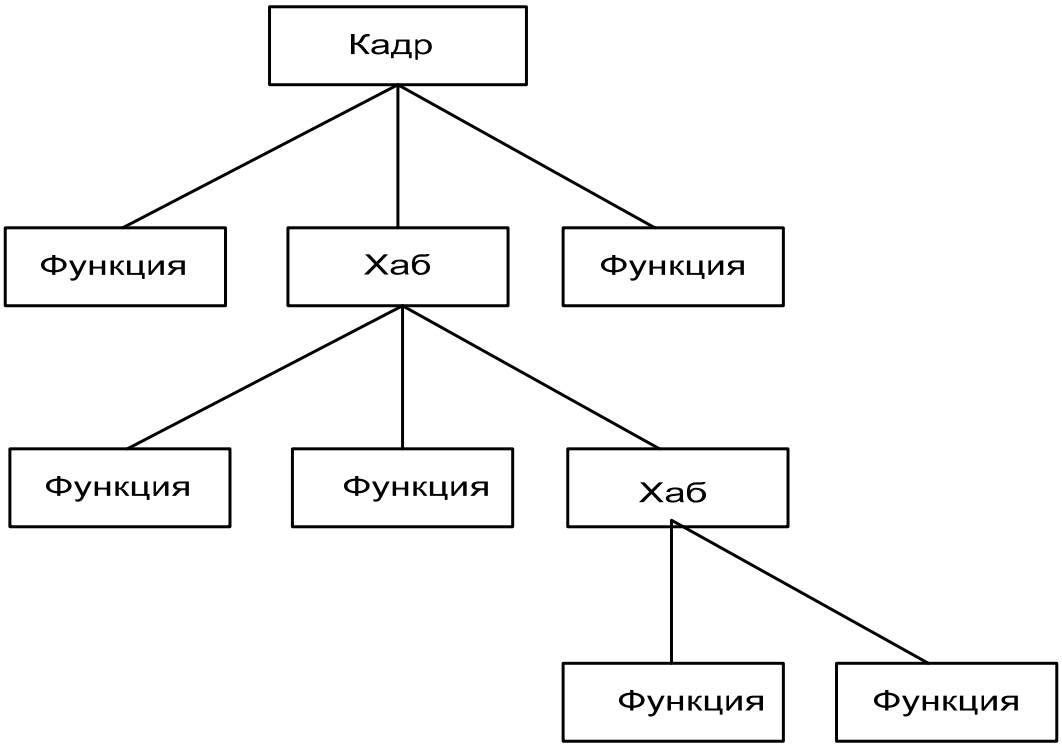


Рис. 9.23. Архитектура соединения хабов

Пропускная способность шины USB, соответствующей спецификации 1.1, составляет 12 Мбит/с (т.е. 1,5 Мбайт/с). Спецификация 2.0 определяет шину с пропускной способностью 400 Мбайт/с. Полоса пропускания делится между всеми устройствами, подключенными к шине. Шина USB имеет три режима передачи данных: низкоскоростной (LS, Low-speed); полноскоростной (FS, Full-speed); высокоскоростной (HS, High-speed, только для USB 2.0).

Спецификация USB определяет три *логических уровня* с определенными правилами взаимодействия. USB-устройство содержит интерфейсную, логическую и функциональную части. Хост тоже делится на три части – интерфейсную, системную и ПО. Каждая часть отвечает только за определенный круг задач. Логическое и реальное взаимодействие между ними показано на рис. 9.24.

Таким образом, операция обмена данными между прикладной программой и шиной USB выполняется путем передачи буферов памяти через следующие уровни: уровень клиентского ПО в хосте и уровень хост-контроллера интерфейса шины USB (HCD, Host Controller Driver).

Уровень системного драйвера USB необходим дня управления ресурсами USB. Он отвечает за назначение логических адресов каждому физическому USB-устройству и планирование транзакций.

Логическое USB-устройство представляет собой набор независимых конечных точек, с которыми клиентское ПО обменивается информацией. Каждому логическому USB-устройству (как функции, так и хабу) назначается свой адрес (1 – 127), уникальный на данной шине USB. Каждая конечная точка логического USB-устройства идентифицируется своим номером (0 – 15) и направлением передачи (IN-передача к хосту, OUT - от хоста).

Транзакции на шине USB – это последовательность обмена пакетами между хостом и ПУ, в ходе которой может быть передан или принят один пакет данных. Когда клиентское ПО передает IRP системному драйверу USB, то он преобразует их в одну или несколько транзакций шины и затем передает получившийся перечень транзакций драйверу контроллера хоста.

Системный драйвер USB состоит из *драйвера USB* и *драйвера контроллера хоста.* Драйвер контроллера хоста принимает от системного драйвера перечень транзакций, планирует исполнение полученных транзакций, добавляя их к списку транзакций, извлекает из списка очередную транзакцию, передает её на уровень хост-контроллера интерфейса шины USB и отслеживает состояние каждой транзакции вплоть до ее завершения.

При выполнении всех связанных с командным пакетом транзакций системный уровень уведомляет об этом клиентский уровень.

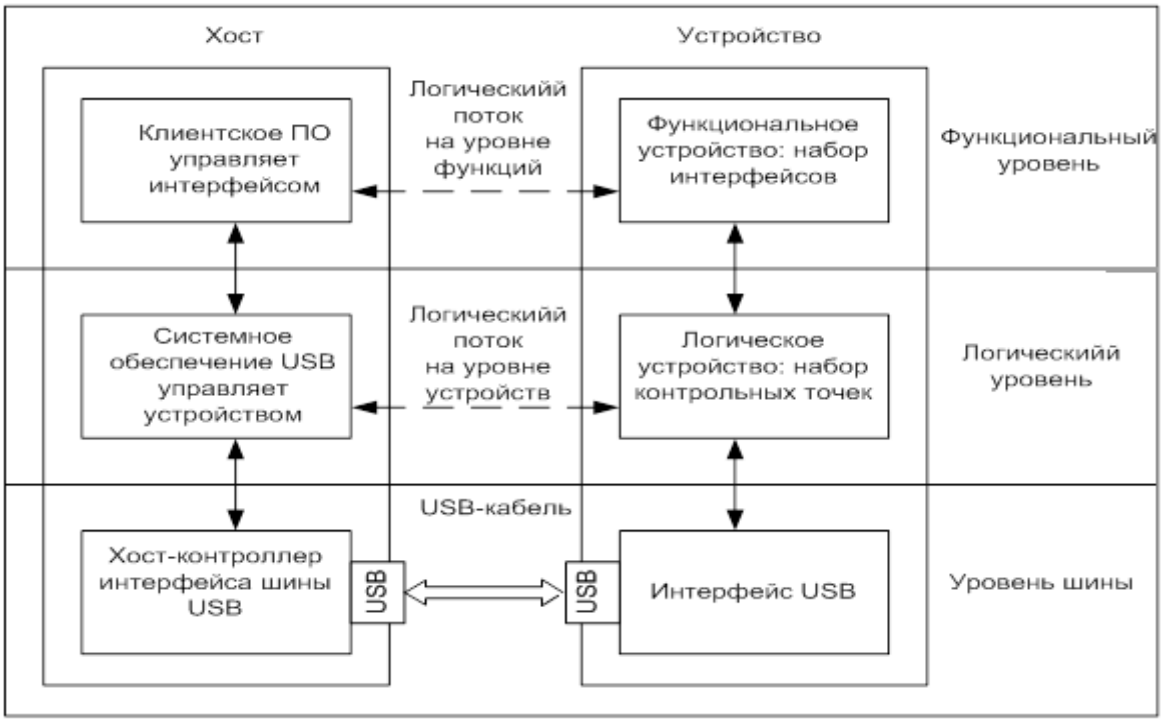


Рис. 9.24. Взаимодействие компонентов USB

Уровень хост-контроллера интерфейса шины USB получает отдельные транзакции от драйвера контроллера хоста (в составе уровня системного обеспечения USB) и преобразует их в соответствующую последовательность операций шины. В результате этого USB-пакеты передаются вдоль всей физической иерархии хабов (на рис. 9.24 мы изобразили последовательность хабов как одну логическую линию, но физически это может быть как один USB-кабель, так и последовательность хабов) до периферийного USB-устройства.

Нижний уровень периферийного USB-устройства называется уровнем интерфейса шины USB. Он взаимодействует с интерфейсным уровнем шины USB на стороне хоста и передает пакеты данных от хоста в формате, определяемом спецификацией USB. Затем он передает пакеты вверх – уровню логического USB-устройства.

Средний уровень периферийного USB-устройства называется уровнем логического USB-устройства. Каждое логическое USB-устройство представляется набором своих точек. Эти точки являются источниками и приемниками всех коммуникационных потоков между хостом и периферийными USB-устройствами.

Самый верхний уровень периферийного USB-устройства называется функциональным уровнем. Этот уровень соответствует уровню клиентского обеспечения хоста. С точки зрения клиентского уровня, нижележащие уровни нужны для организации между ним и конечными точками прямых *каналов данных,* которые идут вплоть до функционального уровня периферийного USB-устройства. А с точки зрения нашей схемы, функциональный уровень получает данные, посылаемые клиентским уровнем хоста из конечных точек каналов данных нижележащего уровня логического USB-устройства и посылает данные клиентскому уровню хоста, направляя их в конечные точки каналов данных нижележащего уровня логического USB-устройства.

Спецификация шины определяет четыре различных типа передачи (transfer type) данных для конечных точек:

- *управляющие передачи* (control transfers) используются хостом для конфигурирования устройства во время подключения, для управления устройством и получения статусной информации в процессе работы. Протокол обеспечивает гарантированную доставку таких посылок. Длина поля данных управляющей посылки не может

превышать 64 байта на полной скорости и 8 байтов на низкой. Для таких посылок хост гарантированно выделяет 10 % полосы пропускания;

- *передачи массивов данных* (bulk data transfers) применяются при необходимости обеспечения гарантированной доставки данных от хоста к функции или от функции к хосту, но время доставки неограниченно. Такая передача занимает всю доступную полосу пропускания шины. Пакеты имеют поле данных размером 8, 16, 32 или 64 байт. Приоритет у таких передач самый низкий, они могут приостанавливаться при большой загрузке шины. Допускаются только на полной скорости передачи. Такие посылки используются, например, принтерами или сканерами;
- *передачи по прерываниям* (interrupt transfers) используются в том случае, когда требуется передавать одиночные пакеты данных небольшого размера. Каждый пакет требуется передать за ограниченное время. Операции передачи носят спонтанный характер и должны обслуживаться не медленнее, чем того требует устройство. Поле данных может содержать до 64 байтов при передаче на полной скорости и до 8 байтов на низкой. Предел времени обслуживания устанавливается в диапазоне 1 – 255 мс для полной скорости и 10 – 255 мс для низкой. Такие передачи используются в устройствах ввода, таких как мышь и клавиатура;
- *изохронные передачи* (isochronous transfers) применяются для обмена данными в "реальном времени", когда на каждом временном интервале требуется передавать строго определенное количество данных, но доставка информации негарантирована (передача данных ведется без повторения при сбоях, допускается потеря пакетов). Такие передачи занимают предварительно согласованную часть пропускной способности шины и имеют заданную задержку доставки.

Внимание. *Все операции по передаче данных инициируются только хостом независимо от того, принимает ли он данные или пересылает в периферийное USB-устройство. Все невыполненные операции хранятся в виде четырех списков по типам передач. Списки постоянно обновляются новыми запросами. Планирование операции по передаче информации в соответствии с упорядоченными в виде списков запросами выполняется хостом с интервалом в один кадр.*

Обслуживание запросов выполняется по следующим правилам.

❒ наивысший приоритет имеют изохронные передачи;

❒ после отработки всех изохронных передач система переходит к обслуживанию передач прерываний:

❒ в последнюю очередь обслуживаются запросы на передачу массивов данных;

❒ по истечении 90 % указанного интервала хост автоматически переходит к обслуживанию запросов на передачу управляющих команд, независимо от того, успел ли он полностью обслужить другие три списка или нет.

5. ОФОРМЛЕНИЕ И ЗАЩИТА КУРСОВОГО ПРОЕКТА

1. Пояснительная записка

Объем пояснительной записки должен давать полное представление о предмете разработки, способе работы, особенностях функционирования и состовлять не менее 30 листов машинописного текста (без рисунков, схем и графиков). Оформление пояснительной записки должно соответствовать требованиям ГОСТ на оформление конструкторской документации. Текст набирается на компьютере в редакторе Word 7 для Windows ****. Тип шрифта Times New Roman Cyr, размер шрифта 14-й, междустрочный интервал 1.5.

Описываемая вычислительная система должна содержать все необходимые блоки, маршрутизаторы, роутеры, репитеры схемы дешифрации и так далее и описание их работы в проектируемой системе. В записке приводятся подробная блок-схема алгоритма работы устройства и текст программы, карта памяти, распределение системных адресов устройств. Приводится описание работы на уровне функциональной схемы преобразователя информации (первичного преобразователя) и описание его схемотехнического решения.

Все разделы записки должны быть представлены в объеме, позволяющем понять принцип работы проектируемого устройства, его составных частей, обоснование выбранного первичного преобразователя и сопоставительные характеристики, выбор вычислительного устройства с указанием типа МЭВМ, ее основными характеристиками

и сопоставительного анализа, на примере нескольких, не менее 5 - 6, МЭВМ. При этом необходимо особо подчеркнуть, почему выбранный микропроцессор в наиболее полной степени удовлетворяет условиям разработки. В ходе разработки алгоритма функционирования устройства следует подробно описать всю последовательность действий, выполняемую МЭВМ для достижения поставленной цели. Блок-схема алгоритма выполняется с соблюдением ГОСТ. Каждый элемент блок-схемы, изображение которого наносится методами, регламентируемыми ГОСТ, должен сопровождаться текстовым описанием.

Программа должна содержать определение всех ячеек памяти, регистров и тому подобного, используемых в ходе ее выполнения. Листинг программы должен начинаться с заголовка, а каждый программный блок должен иметь подробный комментарий, который не должен сводиться к простому наименованию команды, а должен описывать выполняемую функцию.

Первый лист пояснительной записки представляет задание на проектирование. Задание должно быть подписано руководителем проектирования, студентом и утверждено заведующим кафедрой.

На следующем листе помещают аннотацию.

Аннотация. Аннотация представляет собой краткую характеристику содержания курсового проекта (работы), излагающую его разделы и указывающую их назначение, адресат, научную или практическую ценность. Из текста аннотации должно явно следовать, о чем идет речь в представляемом проекте. Текст аннотации должен подготавливать рецензента или читателя к восприятию содержания пояснительной записки. Как правило, аннотация содержит пять-семь предложений. Число строк может быть и больше, но их увеличение не должно сводиться к простому перечислению выполненных работ. В конце аннотации, последней строкой, указывается число страниц основного текста, а также количество использованных при написании записки источников литературы, приведенных в тексте рисунков, построенных таблиц и приложений.

Запись этой строки выглядит следующим образом:

С. 105. Табл. 9. Ил. 10. Библиогр. 50. Прил. 5.

Текст аннотации должен продемонстрировать умение студента кратко и содержательно излагать существо сложных процессов и явлений.

Следующий лист записки - содержание. В нем представляются все разделы, имеющие заголовки и подзаголовки.

Далее разделы представляют в следующей последовательности.

Введение. Это вступительный раздел, содержащий предварительные сведения и основные положения, рассматриваемые в проекте. Задача студента - на основе различных документов, постановлений, решений, концепций показать актуальность, необходимость решения поставленной задачи, ее практическую ценность для промышленности и предполагаемые технико-экономические характеристики.

Условно введение можно разделить на три части. Первая часть характеризует, например, постановления правительства о текущих задачах в определенной отрасли промышленности. Во второй части делается акцент на разрабатываемую тему проекта и показывается связь с ранее указанными нормативными документами. Отмечается необходимость разработки темы именно в направлении рассмотрения предлагаемой концепции, которая позволит решить ту или иную проблему. В третьей части приводятся ожидаемые технико-экономические характеристики. Во введении можно приводить графики и таблицы, наглядно демонстрирующие повышение качественных и количественных характеристик в результате принятых технических и экономических решений. Общий рекомендуемый объем введения должен составлять примерно от 1 до 3 страниц.

Обзор литературы и патентные исследования

Аналитический обзор. Аналитический обзор (исследования) – это текст, содержащий синтезированную информацию сводного характера по какому-либо вопросу или ряду вопросов, извлеченную из некоторого множества специально отобранных для этой цели первичных документов. Обзоры различаются по предмету анализа, цели составления, назначению, видам используемых первоисточников, широте тематики, наличию сопоставлений и прогнозов, уникальному назначению в документальной системе, характеру оформления и др.

Аналитическая часть содержит анализ и его результаты, обобщение и оценку систематизированных сведений о состоянии рассматриваемых и нерешенных вопросов, использованные методы и средства исследования, состояние исследований и разработок, достигнутый научно-технический уровень, организационно-экономическую ситуацию, тенденции развития.

Патентные исследования. Патентная информация как источник научно-технической информации обладает оперативностью (как правило, предшествует публикации других информационных материалов), достоверностью (данные проверяются государственной патентной экспертизой), полнотой сведений (излагается суть открытий или изобретений, используется сквозная нумерация патентных документов).

Поэтому правильное использование патентной информации дает возможность осуществлять новые разработки на уровне лучших мировых образцов с учетом имеющихся решений и основных тенденций развития техники.

В связи с этим перед началом разработки темы необходимо предварительно провести патентные исследования. Это комплекс работ, включающий поиск, отбор, анализ и целенаправленное использование патентной информации (патентной документации и литературы). Под патентной документацией понимается публикация официальными органами различных стран сведений об открытиях, изобретениях, промышленных образцах, полезных моделях, товарных знаках.

Источниками информации, используемыми в процессе патентных исследований, являются бюллетени патентных ведомств стран мира, описания изобретений, реферативная информация по изобретениям, публикации о внедренных изобретениях, рекламные материалы, отчеты о НИР, командировках, информация по отраслям народного хозяйства, а также отчеты о патентных исследованиях.

Рекомендуемый объем раздела 5 - 7 страниц.

Технологическая часть. В этом разделе основная задача – анализ технологического процесса с целью реализации основной задачи проекта. Если считать, что объектом проектирования является технологический процесс управления, выполняющий конкретные функции, то необходимо в первую очередь рассмотреть последовательность действий по достижению главной цели.

Пусть целью проектирования является создание распределенной системы управления. Технологический процесс в общем случае представляется некоторой последовательностью операций (обработка, транспортировка, сборка, ориентация, преобразование информации или электрического сигнала из одного вида в другой и т.п.). В этом случае необходимо описать последовательность действий по выполнению операций для преобразования исходного объекта в законченное изделие или, иными словами, построить алгоритм работы системы.

Описание работы оборудования составляют, как правило, в виде текста с характеристикой работы отдельных цепей, состоящих из соответствующих элементов, и соединяющих их линий связи. Обычно для сложных схем с большим числом отдельных линий связи или большим числом элементов в них такие описания весьма многословны, что затрудняет восприятие и понимание процесса работы каждой цепи и всего оборудования в целом.

Рекомендуемый объем раздела 5 - 7 страниц.

Конструкторская часть. Включает в себя описание структурной схемы вычислительной системы (расчеты всех устройств), электрической принципиальной схемы вычислительной системы (выбор и обоснование применяемых электрических элементов), блок-схемы алгоритма работы разработанного устройства, программу на языке ассемблер для выбранного микропроцессора, конструкции устройства. Приводится обоснование выбора топологии сети, физической среды, устройств, реализующих передачу информации с заданными характеристиками. Полное описание логического уровня представляется в виде пакетов передаваемой информации и их оформления. Каждое поле пакета подробно описывается в тексте и выносится на чертеж.

Описывается размерность и приводится расчет числа бит для каждого байта передаваемой информации с учетом особенностей функционирования сети.

Примерный объем – 15 страниц без учета страниц с рисунками.

Заключение. В заключении необходимо привести конкретные результаты, полученные в процессе проектирования. Объем примерно одна страница.

Список использованной литературы. При выполнении проекта необходимо использованную литературу упомянуть в тексте, указывая в квадратных скобках номер по порядку ссылок, приводимых в списке. Список неограничен, но если литература приведена, то на нее обязательно должна быть ссылка в тексте. Неуказанная по тексту литература в списке не приводится. Список оформляется в соответствии с требованиями стандарта.

Приложения. В приложении приводится дополнительный материал, необходимый для полного раскрытия темы, но не являющийся основополагающим. Объем приложения определяется студентом, но не должен повторять известные материалы без упоминания о нем в основном тексте.

2. Графический материал

Чертежи выполняются на ватмане (базовый формат А1) в соответствии с требованиями ГОСТ. Плотность заполнения поля чертежа не менее 80 %. Для защиты представляется дискета с чертежами и распечатка их в формате А4. Могут использоваться графические редакторы AUTOCAD2004 или VISIO2000. Рекомендуемый редактор – „Компас”.

При оформлении чертежей необходимо соблюдение действующих стандартов при обозначении элементов на схемах электрических принципиальных и функциональных. Для электрических принципиальных схем обязательно нанесение на чертежах перечня элементов.

3. Защита проекта

Защита проекта проводится в последнюю неделю семестра. Процедура защиты включает в себя доклад по теме разработки перед аудиторией и комиссией, ответы на вопросы присутствующих на защите студентов и преподавателей, проставление оценки комиссией. В состав комиссии входят не менее трех преподавателей кафедры, ведущих родственные дисциплины.

ПРИЛОЖЕНИЕ

ПРИМЕР РАЗРАБОТКИ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

РАСПРЕДЕЛЕННАЯ ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА ИЗМЕРЕНИЯ СКОРОСТИ

(приводится в сокращенном виде)

ЗАДАНИЕ

Необходимо разработать:

* структуру вычислительной системы;
* схему подключения стандартных устройств ввода-вывода;
* состыковать датчик скорости с каналом вычислительной системы;
* прикладную программу на языке ассемблер;
* схему электрическую принципиальную вычислительного устройства.

СОДЕРЖАНИЕ

1. АННОТАЦИЯ

В курсовом проекте разработана структурная схема системы измерения скорости для привода. Разработана электрическая принципиальная схема, блок-схема алгоритма работы устройства и программа на языке примененной МЭВМ.

Ил. 3, Табл. 1, Библиогр. 12.

2. ВВЕДЕНИЕ

Производство микропроцессоров и МЭВМ - новая и наиболее быстроразвивающаяся отрасль промышленности. Её влияние на развитие всех отраслей народного хозяйства страны становится всё более заметным.

Технология производства микропроцессоров позволяет уже сегодня иметь на кристалле более 100 тыс. транзисторов и строить функционально-развитые вычислительные устройства. Такой подход предопределил появление МЭВМ. Оборудование с МЭВМ такого типа потребляет мало энергии, имеет малые габаритные размеры, повышенную надёжность, ремонтопригодность, удобство и простоту обслуживания.

Возможности микропроцессоров определили две крупные области их применения: первая – встраивание микропроцессоров в станки, двигатели, роботы; вторая – использование их для управления взаимосвязанными технологическими комплексами, гибкими переналаживаемыми производствами, автоматизированными предприятиями.

3. ОБЗОР ЛИТЕРАТУРЫ И ПАТЕНТНЫЕ ИССЛЕДОВАНИЯ

Измерение скорости (частоты вращения) основано на использовании небольших электрических машин - тахогенераторов и импульсных датчиков. Напряжение тахогенератора (ТГ) постоянного тока от контактных щеток подается к измерительным приборам и регуляторам систем автоматического управления электроприводами.

Для упразднения коллектора и щеток, а также в целях получения специальных сигналов разработаны ТГ синхронные, асинхронные и индукционные. Магнитный поток ТГ всех видов создается при помощи независимых обмоток возбуждения или постоянных магнитов. На базе малогабаритных однофазных и трехфазных синхронных ТГ с помощью разных схем выпрямления построены бесконтактные ТГ постоянного тока. Разработаны схемы и конструкции машин, позволяющие осуществлять реверс. Для этой цели синхронные ТГ комплектуются сельсинами, применяются фазочувствительные выпрямительные схемы.

Наряду с ТГ для измерения скорости все чаще применяются им-

пульсные датчики (ИМД). Основное распространение находят индукционные и фотоимпульсные датчики.

В схемах автоматизации технологических линий применяются комбинированные устройства контроля скорости, включающие два электромагнитных или импульсных датчика и более, устройства контроля ускорения, переключатели диапазонов, преобразователи сигналов в дискретную или аналоговую форму. Импульсы импульсных датчиков (ИД) создаются под влиянием пульсирующего или знакопеременного магнитного потока. У фотоимпульсных датчиков (ФД) импульсы создаются при помощи дисков с прорезями и отверстиями. Преобразователи частотных сигналов, поступающих от ИД и ФД, в зависимости от структуры системы контроля скорости содержат следующие узлы: электронный усилитель, преобразователь частоты в ток, генератор опорной частоты.

Большинство ТГ и ИМД выпускаются как автономные устройства в отдельных корпусах. Конструктивно наиболее простыми являются сборные датчики с жестким креплением всех деталей. Широкое применение в системах контроля скорости нашли подвесные “плавающие” конструкции. Подвесные конструкции имеют особые свойства компенсации вибраций, благодаря которым уменьшаются угловые погрешности сигнала.

Для единичных или периодических изменений частоты вращения применяют переносные устройства – тахометры различных принципов действия и конструкций. В тахометрах новых видов используются электромашинные датчики и элементы электронной техники. Известны также стробоскопические и индукционные системы.

Выбор отдельных видов и исполнений датчиков скорости определяется назначением устройства, условиями конструктивного сочетания датчика с другими элементами измерительного комплекта, а также техническими требованиями в отношении точности, чувствительности, диапазона измерения и условий работы.

4. ОПИСАНИЕ СТРУКТУРНОЙ СХЕМЫ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Данная микропроцессорная система реализует измерение скорости путем подсчета импульсов, поступающих за определенный интервал времени. Она состоит из микропроцессора МП К580ВМ80, интер-

вального таймера, связанного с МП шиной данных, тактового генератора, блока памяти, шины адреса (рис. П1).

Интервальный таймер подсчитывает число импульсов, поступающих с вала двигателя за интервал времени между тактовыми импульсами, поступающими с тактового генератора. С каждым новым тактовым импульсом счетчик обнуляется. Данные с интервального таймера по шине данных поступают в МП. Затем поступают данные для расчетов скорости, а потом преобразуются в параллельный код, который помещается в ячейку с адресом, номер которой МП выдает на шину адреса. Чтение значения кода скорости осуществляется на лету, без остановки счета интервального таймера. Значение кода скорости затем передается либо в блок индикации, либо в систему управления приводом.

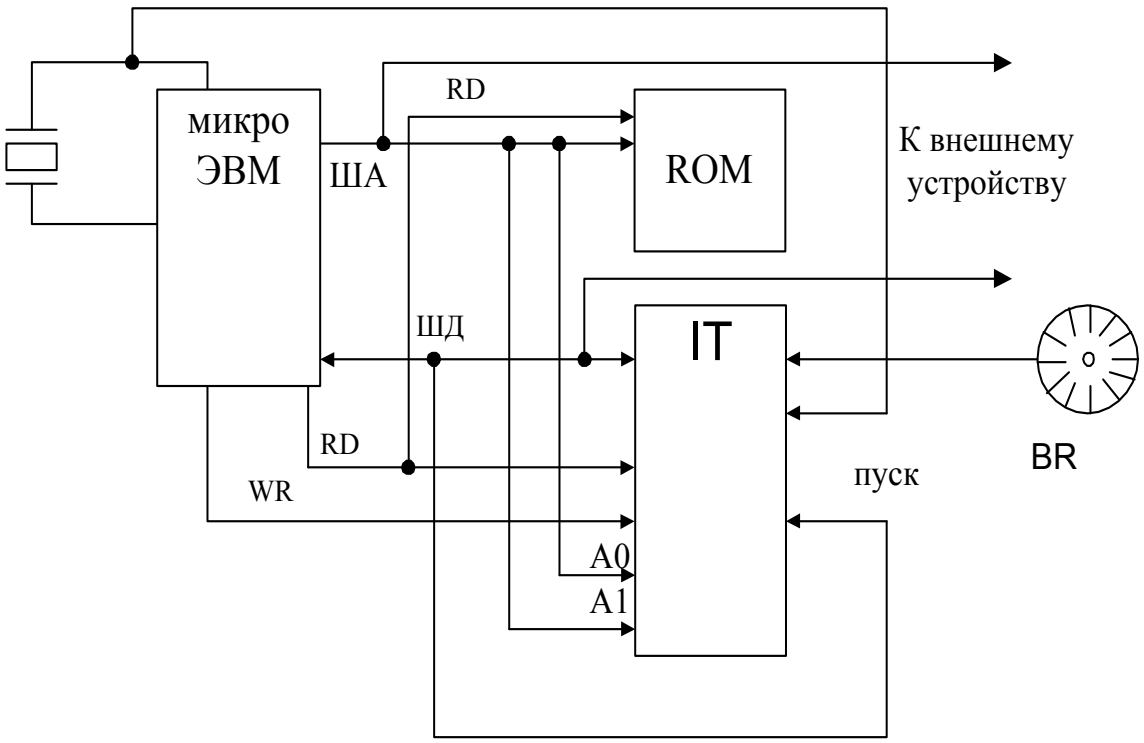


Рис. П1. Структурная схема системы измерения скорости

5. ОПИСАНИЕ ЭЛЕКТРИЧЕСКОЙ ПРИНЦИПИАЛЬНОЙ СХЕМЫ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Все операции, реализуемые в системе, инициируются процессорным модулем, который представляет собой одноплатную МЭВМ,

выполненную на основе МЭВМ К1816ВЕ35. Обмен данными в МЭВМ осуществляется через квазидвунаправленные порты ввода/вывода информации Р1 и Р2. Кроме того, в МЭВМ, существует порт ввода/вывода данных BUS. Порт Р1 через буферное устройство (BF2) выведен на внешнюю шину. К выводам этого буферного устройства подключаются соответствующие линии пульта управления микроконтроллером. Назначение разрядов этой шины представлены в таблице.

Контакт	Разряды внешней шины	Обозначение
А2	Данные разряд 0	D0
Б3	Данные разряд 1	D1
А3	Данные разряд 2	D2
Б4	Данные разряд 3	D3
А5	Данные разряд 4	D4
Б5	Данные разряд 5	D5
А6	Данные разряд 6	D6
Б6	Данные разряд 7	D7
А1,Б1	Источник питания 0	U1
А15,Б16	Источник питания 0	0V

Выходные шины через нагрузочные резисторы подключены к источнику питания +5V и выведены на внешний разъем. Два остальных буферных устройства могут быть использованы для расширения. Шинный формирователь BF1 используется для буферирования шин управления, предназначенных для организации канала. Выходные сигналы шинного формирователя BF1 распределяются следующим образом. Сигналы RD, WR, ALE, T0 подключаются к магистрали канала и образуют общую шину. Сигналы PME и выходные сигналы порта Р2 (Р2.3 и Р2.4) используются для формирования сигналов управления внешней памяти, обеспечивая выбор одного из банков памяти (рис. П2).

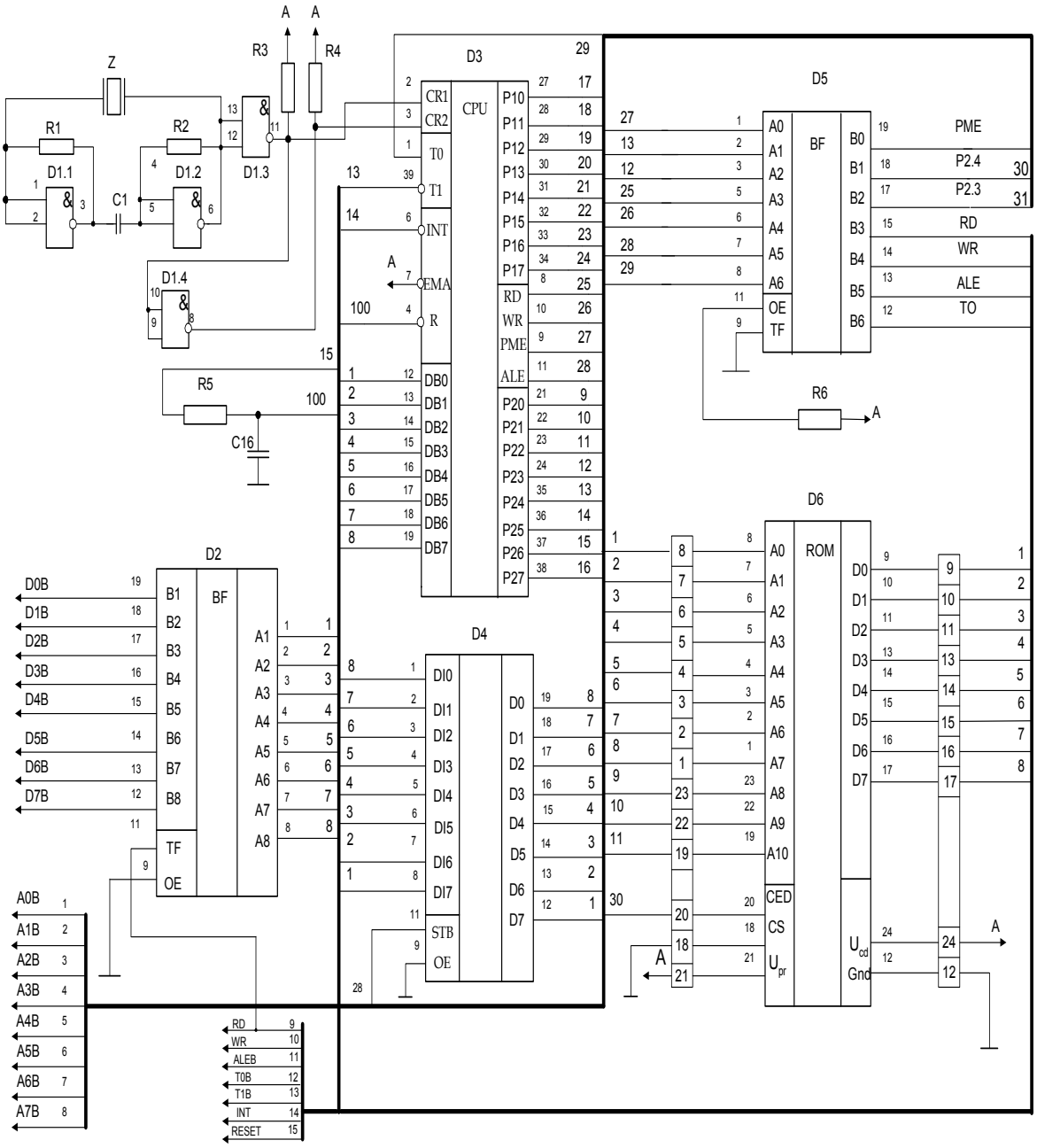


Рис. П2. Электрическая принципиальная схема МЭВМ

Объем памяти каждой микросхемы составляет 2 К. Поэтому для адресации используются все шины порта DB и разряды 0, 1, 2 порта P2. Для фиксации младших разрядов адресной части используется буферный регистр RG. Выводимые из ПЗУ команды поступают на вход шины DB. Связь с общей шиной по каналу передачи данных осуществляется через буферный формирователь BF3.

6. ОПИСАНИЕ БЛОК-СХЕМЫ АЛГОРИТМА РАБОТЫ СПРОЕКТИРОВАННОГО УСТРОЙСТВА

Загружается текущее значение скорости вращения двигателя $n_т$, затем анализируется равенство $n_т$ и заданного значения скорости вращения $n_з$ (рис. П3). Если они равны, то угол отпирания $\alpha_р$ сохраняет свое предыдущее значение и выводится на внешнее устройство. Если равенство не выполняется, то рассчитывается новое значение $\alpha_р$, которое впоследствии выводится на внешнее устройство. Затем производится генерация задержки, и по истечении заданного времени снимаются данные и вводятся в МП. По окончании измерения происходит опрос триггера готовности. Цикл вновь повторяется.

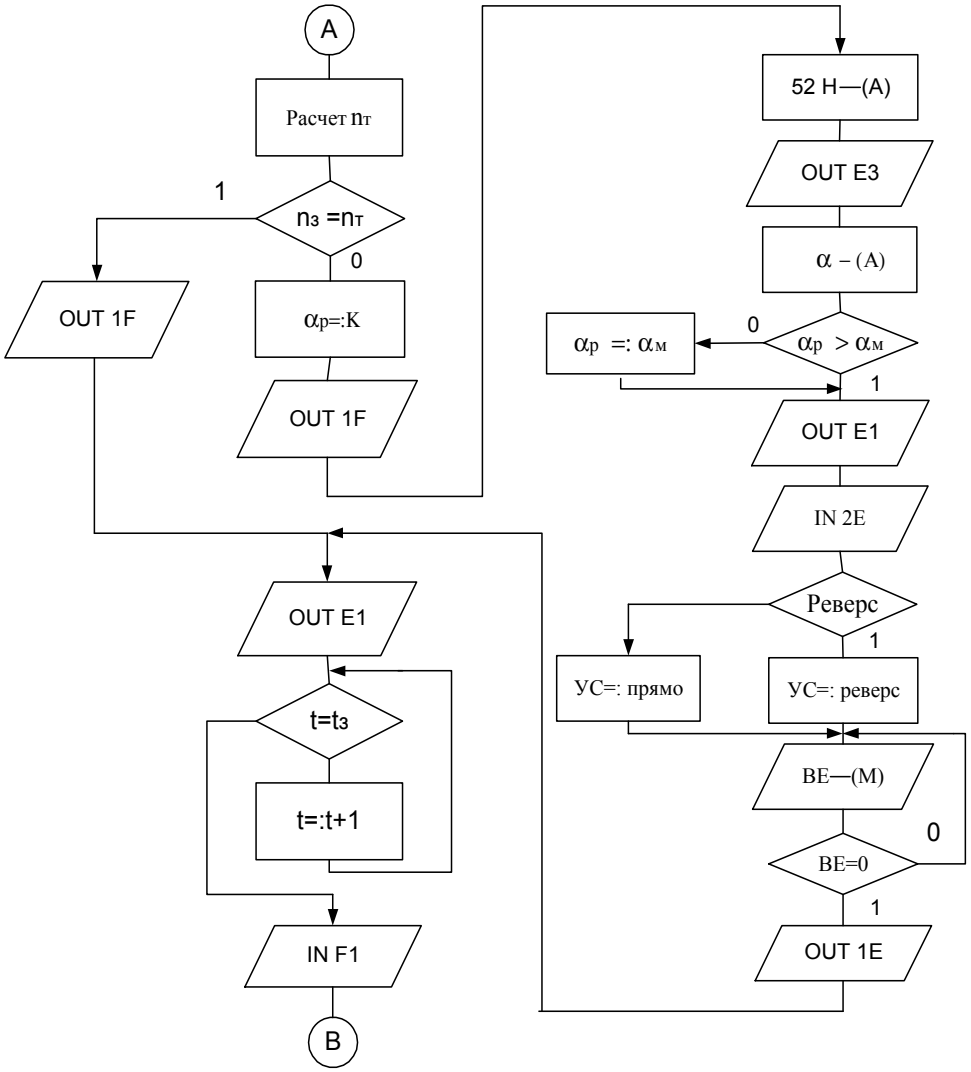


Рис. П3. Алгоритм работы устройства

7. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

Распределение адресов

Исполнительные устройства	Шестнадцатеричный адрес
• Интервальный таймер	
канал 0	ED
канал1	E1
канал 2	E2
управляющее слово	E3
• Регистр задания режима	1E
• Триггер датчика тока	2E
Область памяти:	
• Константы	3000-3100
• Промежуточные данные	3200-3300
• Основная память для "рабочих" программ	2100-2FFF

Программа работы измерителя скорости

Адрес	Объектный код	Метка	Операция	Комментарий
20F1	3E	–	MVI A,$n_т$	Загрузить $n_т$
20F3	3A0031	–	LDA 3100	$n_з$
20F6	BE	–	CMP M	Сравнить
20F7	DA213E	–	JC AA	
20FA	C3212B	–	JMP KK	
20FD	3EK	–	MVI A,K	Загрузить К
20FF	D31F	–	OUT1F	Вывести в 1F
2100	3E52	SI	MVI A,52	Загрузить управ-ляющее слово
2102	03E3	–	OUT E3	
2104	3A0030	–	LDA3000	
2107	210032	–	LXIH,3200	
210A		–	CMP M	Сравнить ALFP>ALFM
210B	DA1021	–	JC BE	
210E	3E18	–	MVI A,18	
2110	D3E1	RR	OUT E1	

Окончание

Адрес	Объектный код	Метка	Операция	Комментарий
2112	DB1E	–	IN1E	Опрос триггера режима
2114	E608	–	ANI08	Прежний режим
2116	SA1F21	–	JZ AR	–
2119	0617	–	MVI B,17	Загрузить управляющее слово
211B	00	–	NOP	–
211C	C32121	–	JMP BR	–
211F	062A	AR	MVI B,2A	Загрузить реверс
2121	DB2E	BR	IN2E	Опрос датчика тока
2123	E620	–	ANI20	Ток равен нулю
2125	C2121	–	JNZ BR	–
2128	78	–	MOV A,B	Загрузить регистр
2129	D31E	–	OUT1E	Вывести управляющее слово
212B	D31E	KK	OUT	–
212D	3A0231	BB	LDA3102	Загрузить t
2130	3A0131	–	LDA3101	Загрузить t_3
2134		BE	CMP M	Сравнить $t = t_3$
2135	DA3C21	–	JC AK	–
2138	3A0331	–	LDA3103	Загрузить t+1
213C	C32D21	–	JMP BB	–
213F	DBF1	AK	IN F1	–
2141	D31F	AA	OUT1F	Вывести в порт1F

8. КОНСТРУКЦИЯ УСТРОЙСТВА

Процессорный модуль представляет собой одноплатную микроЭВМ, выполненную на основе однокристальной микроЭВМ (ОЭВМ) К1816ВЕ35. Плата оснащена разъемами, один из которых предназначен для подключения к интерфейсу "Общая шина", а с другой - для подключения периферийных устройств. Конструктивно контроллер выполнен в виде каркаса, задняя стенка которого представляет собой печатные проводники, в которых включены разъемы. В каркас контроллера устанавливаются платы, число которых определяется чис-

лом линий входа и выхода. С правой стороны в каркас установлен блок питания, на лицевой панели которого находится светодиод для контроля включенного состояния и разъем для подключения переменного напряжения. Модули устанавливаются в контроллер по направляющим. Каркас контроллера открытого типа и при установке в станцию управления крепится на болтах.

9. ЗАКЛЮЧЕНИЕ

В данной работе была спроектирована микропроцессорная система, осуществляющая измерение скорости в приводах путём подсчёта импульсов, поступающих с датчика за определённый интервал времени. Были разработаны структурная и электрическая принципиальная схемы вычислительной системы, реализующие процедуру управления, блок-схема алгоритма работы разработанного устройства, а также была составлена программа на языке ассемблер.

10. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. *Григорьев, В. Л.* Программное обеспечение микропроцессорных систем. – М. : Энергоатомиздат, 1983. – 208 с.
2. МикроЭВМ / под ред. А. Дирксена. – М. : Энергоиздт, 1982. – 328 с.
3. *Острем, К. Виттенмарк, Б.* Системы управления с ЭВМ : пер. с англ. – М. : Мир, 1987. – 480 с.

Библиографический список

1. *Веселов, О. В.* Локальные сети малых вычислительных систем : учеб. пособие (для вузов) / О. В. Веселов. – Владимир: Изд-во Владим. гос. ун-та, 2005. – 211 с. – ISBN 5-89368-631-4.

2. *Столлингс, В.* Операционные системы. Внутреннее устройство и принципы проектирования = Operating Systems. Internals and Design Principles : пер. с англ. / В. Столлингс. – 4-е изд. – М. : Вильямс, 2002 . – 843 с. – ISBN 5-8459-0310-6.

3. *Олифер, В. Г.* Компьютерные сети. Принципы, технологии, протоколы : учеб. пособие / В. Г. Олифер, Н. А. Олифер. – 3-е изд. – СПб. : Питер, 2008. – 957 с. (Учебник для вузов). – С. 922 - 957 . – ISBN 978-5-469-00504-9.

4. *Мюллер, С.* Модернизация и ремонт ПК = Upgrading and Repairing PCs : пер. с англ. / С. Мюллер. – 17-е изд. – М. : Вильямс, 2007. – 1489 с. – ISBN 978-5-8459-1126-1.

5. *Пирогов, В. Ю.* Ассемблер на примерах / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2005. – 416 с. – ISBN 5-94157-745-1.

6. *Безуглов, Д. А.* Цифровые устройства и микропроцессоры : учеб. пособие для вузов по направлению 210300 (654200) "Радиотехника" / Д. А. Безуглов, И. В. Калиенко. – Ростов н/Д : Феникс, 2006. – 469 с. – (Высшее образование). – ISBN 5-222-08211-3.

Оглавление

Учебное издание

ВЕСЕЛОВ Олег Вениаминович
БАКУТОВ Александр Владимирович

МАЛЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Учебное пособие

Подписано в печать 12.12.12 .
Формат 60×84/16. Усл. печ. л. 21,04. Тираж 100 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.