С. В. БЕЛУГИНА

АРХИТЕКТУРА КОМПЬЮТЕРНЫХ СИСТЕМ

КУРС ЛЕКЦИЙ

Учебное пособие

САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР
2020

УДК 004
ББК 32.81я73

Б 43 **Белугина С. В.** Архитектура компьютерных систем. Курс лекций : учебное пособие / С. В. Белугина. — Санкт-Петербург : Лань, 2020. — 160 с. : ил. — (Учебники для вузов. Специальная литература). — Текст : непосредственный.

ISBN 978-5-8114-4489-2

Учебная дисциплина «Архитектура компьютерных систем» рассчитана на 74 аудиторных часа и предназначена для студентов 2 курса специальности «Программирование в компьютерных системах».

Целью изучения дисциплины является формирование у студентов знаний о представлении информации в вычислительных системах, об архитектуре и принципах работы ЭВМ и её основных логических блоков, организации вычислительных систем. Содержание и структура пособия соответствуют требованиям Государственного образовательного стандарта среднего профессионального образования.

Издание поможет систематизировать знания, полученные на лекциях и практических занятиях, выполнить внеаудиторную самостоятельную работу, подготовиться к текущему и промежуточному контролю.

Пособие адресовано студентам средних образовательных учреждений, а также всем интересующимся данной тематикой.

УДК 004
ББК 32.81я73

Обложка
П. И. ПОЛЯКОВА

© Издательство «Лань», 2020
© С. В. Белугина, 2020
© Издательство «Лань»,
художественное оформление, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Архитектура компьютерных систем — это одна из базовых, фундаментальных основ в процессе профессиональной подготовки будущего техника-программиста.

Курс лекций дисциплины «Архитектура компьютерных систем» соответствует содержанию ФГОС для специальности 09.02.03 «Программирование в компьютерных системах». Данная дисциплина является общепрофессиональной. Целью изучения дисциплины является формирование у студентов знаний о представлении информации в вычислительных системах, об архитектуре и принципах работы ЭВМ и её основных логических блоков, организации вычислительных систем.

Преподавание дисциплины должно иметь практическую направленность и проводиться в тесной взаимосвязи с общепрофессиональными дисциплинами «Операционные системы», «Технические средства информатизации». Полученные по дисциплине знания позволят студентам выполнять задания учебной практики.

Основные разделы курса:

Раздел 1. Представление информации в вычислительных системах.

Раздел 2. Архитектура и принципы работы основных логических блоков вычислительных систем.

Раздел 3. Вычислительные системы.

На самостоятельное изучение отдельных тем дисциплины студентами отводится 48% учебного времени. Назначение самостоятельной работы — способствовать совершенствованию практических умений по разработке и тестированию программных продуктов, систематизации и обобщению теоретических знаний и получению первоначального опыта творческой деятельности программиста.

По окончании изучения дисциплины у студентов должны сформироваться следующие общие и профессиональные компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 6. Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

ПК 1.1. Выполнять разработку спецификаций отдельных компонент.

ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК 1.5. Осуществлять оптимизацию программного кода модуля.

ПК 2.3. Решать вопросы администрирования базы данных.

ПК 2.4. Реализовывать методы и технологии защиты информации в базах данных.

ПК 3.1. Анализировать проектную и техническую документацию на уровне взаимодействия компонент программного обеспечения.

ПК 3.2. Выполнять интеграцию модулей в программную систему.

ПК 3.4. Осуществлять разработку тестовых наборов и тестовых сценариев.

В результате освоения дисциплины обучающийся должен уметь:

– получать информацию о параметрах компьютерной системы;

– подключать дополнительное оборудование и настраивать связь между элементами компьютерной системы;

– производить инсталляцию и настройку программного обеспечения компьютерных систем.

В результате освоения дисциплины обучающийся должен знать:

– базовые понятия и основные принципы построения архитектур вычислительных систем;

– типы вычислительных систем и их архитектурные особенности;

– организацию и принцип работы основных логических блоков компьютерных систем;

– процессы обработки информации на всех уровнях компьютерных архитектур;

– основные компоненты программного обеспечения компьютерных систем;

– основные принципы управления ресурсами и организации доступа к этим ресурсам.

Лекции охватывают все содержание дисциплины. Могут использоваться студентами на уроках, при подготовке к практическим и лабораторным работам, в самостоятельной работе.

Обращение к студенту

Уважаемый студент!

На вашем пути изучения профессиональных дисциплин появилась важная дисциплина: «Архитектура компьютерных систем». Вы держите в руках учебное пособие, которое поможет вам организовать процесс собственного образования по данной дисциплине. Как вы можете догадаться уже из названия предмета, — это одна из базовых, фундаментальных основ в процессе профессиональной подготовки будущего техника-программиста.

Общеизвестна роль вычислительной техники в различных сферах человеческой деятельности. Особенно популярны персональные компьютеры (ПК), обладающие высокой производительностью и не требующие от пользователя глубокого знания процессов, происходящих в компьютере во время вычислений. Без преувеличения можно сказать, что появление ПК в середине 1970-х годов и бурное их распространение в наше время открыли новую эру в массовом использовании вычислительной техники людьми всех рангов и профессий. Человечество заметно переходит от экономики, основанной на тяжёлой промышленности, к экономике с компьютеризированной технологией, средствами связи и услугами.

При изучении данной дисциплины вы приобретете знания о представлении информации в вычислительных системах, об архитектуре и принципах работы ЭВМ и её основных логических блоков, организации вычислительных систем.

Для закрепления теоретических знаний и приобретения необходимых практических умений учебной дисциплины будут проводиться лабораторные и практические работы.

Желаем удачи!

Тема 1. Организация и принцип работы основных логических блоков компьютерных систем

Лекция 1. Системы счисления

План:

1. Системы счисления.
2. Правила перевода чисел из одной системы счисления в другую.

1. Системы счисления

Системы счисления — совокупность приемов и правил изображения чисел цифровыми знаками.

Самый простой и очевидный пример — система счисления, где количество обозначается I (палочкой/единицей): 1 = I; 2 = II; 5 = IIII; 10 = IIIIIIII.

В зависимости от способа изображения чисел системы счисления делятся на:

- непозиционные;
- позиционные.

Непозиционная система счисления — это система, в которой значение символа не зависит от его положения в числе. Пример: римская система счисления, в которой цифры обозначаются различными знаками: 1 — I, 3 — III, 5 — V, 10 — L, 60 — LX, 90 — XC, 100 — C, 1000 — M.

VII = 5 + 1 + 1 = 7; VI = 5 + 1 = 6; IV = 5 – 1 = 4.

Позиционная система счисления — это система, в которой значение символа зависит от его места (позиции) в ряду цифр, изображающих число. Например, 5643: 3 – единицы, 4 — десятки и т. д. Данные системы более удобны для вычислительных процессов. Позиционная система характеризуется основанием — P. Основание (базис) позиционной системы счисления — количество знаков (символов), используемых для изображения числа в разрядах данной системы счисления.

P = 2: $\{0, 1\}_2$ — двоичная система счисления;

P = 8: $\{0, 1, 2, \ldots, 7\}_8$ — восьмеричная система счисления;

P = 16: $\{0, 1, 2, 3, 4, \ldots, 9, A, B, C, D, E, F\}_{16}$ — шестнадцатеричная система счисления.

Наиболее естественный способ представления числа в компьютерной системе заключается в использовании строки битов, называемой двоичным числом — числом в двоичной системе счисления (символ также может быть представлен строкой битов или символа).

Основание системы счисления — количество (P) различных цифр, используемых для изображения числа в позиционной системе счисления. Значения цифр лежат в пределах от 0 до P – 1. В общем случае запись любого числа N в системе счисления с основанием P будет представлять собой ряд (многочлен) вида

$$\mathbf{N = a_{m-1}\cdot P^{m-1} + a_{m-2}\cdot P^{m-2} + \ldots + a_k\cdot P^k + \ldots + a_1\cdot P^1 + a_0\cdot P^0 + \ldots + a_{-1}\cdot P^{-1} + a_{-2}\cdot P^{-2} + \ldots + a_{-s}\cdot P^s} \quad (1)$$

Нижние индексы определяют местоположение цифры в числе (разряд):

- положительные значения индексов — для целой части числа (m разрядов);
- отрицательные значения — для дробной (s разрядов).

Максимальное целое число, которое может быть представлено в m разрядах:

$$\mathbf{N_{max} = P^m - 1.}$$

Минимальное значащее, не равное 0, число, которое можно записать в s разрядах дробной части:

$$\mathbf{N_{min} = P^{-s}.}$$

Имея в целой части числа m разрядов, а в дробной — s, можно записать P^{m+s} разных чисел.

Двоичная система счисления имеет основание P = 2 и использует для представления информации две цифры: 0 и 1.

В аппаратной основе компьютера лежат двухпозиционные элементы, которые могут находиться только в двух состояниях: одно из них обозначается 0, а другое — 1. Поэтому основной системой счисления, применяемой в компьютерной технике, является двоичная система. С целью сокращения разрядов для записи числа при выводе на экран компьютера используют системы с основанием, являющимся целой степени числа 2: восьмеричную и шестнадцатеричную системы счисления. Для представления одной цифры восьмеричной системы счисления используется три двоичных разряда (триада), шестнадцатеричной — четыре двоичных разряда (тетрада) (табл. 1).

2. Правила перевода чисел из одной системы счисления в другую

Существуют правила перевода чисел из одной системы счисления в другую, основанные в том числе и на выражении (1).

Например, двоичное число 101110,101 равно десятичному числу 46,625:

$$101110{,}101_2 = 1\cdot2^5 + 0\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} = 46{,}625_{10}.$$

Практически перевод из двоичной системы в десятичную можно легко выполнить, надписав над каждым разрядом соответствующий ему вес и сложив затем произведения значений соответствующих цифр на их веса.

Например, двоичное число 01000001_2 равно 65_{10}. Действительно, $64\cdot1 + 1\cdot1 = 65$.

вес	128	64	32	16	8	4	2	1
цифра	0	1	0	0	0	0	0	1

Таким образом, для перевода числа из позиционной системы счисления с любым основанием в десятичную систему счисления можно воспользоваться выражением (1).

Обратный перевод из десятичной системы счисления в систему счисления с другим основанием непосредственно по (1) затруднителен, поскольку все арифметические действия, предусмотренные этой формулой, следует выполнять в той системе счисления, в которую число переводится. Обратный перевод выполняется значительно проще, если предварительно преобразовать отдельно целую и дробную части выражения (1) к виду

$$\mathbf{N_{цел} = ((\ldots (a_{m-1}\cdot P + a_{m-2})\cdot P + \ldots + a_2)\cdot P + a_1)\cdot P + a_0;}$$

$$\mathbf{N_{др} = P^{-1}\cdot (a_{-1} + P^{-1}\cdot (a_{-2} + P^{-1}\cdot (a_{-3} + \ldots + P^{-1}\cdot (a_{-s+1} + P^{-1}\cdot a_{-s})\ldots))).}$$

Алгоритм перевода числа из десятичной системы счисления в систему счисления с основанием P, основанный на этих выражениях, позволяет оперировать числами в той системе счисления, из которой число переводится, и может быть сформулирован следующим образом.

Прямой перевод — это перевод из любой системы счисления в десятеричную.

Для позиционной системы счисления с общим основанием справедливо равенство

$$\mathbf{A(p) = a_n p^n + a_{n-1} p^{n-1} + a_{n-2} p^{n-2} + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + \ldots + a_{-n} p^{-n}},$$

это представление используется для перевода из любой системы счисления в десятичную (прямой перевод).

Примеры:

1) $1992_{10} = 1 \cdot 10^3 + 9 \cdot 10^2 + 9 \cdot 10^1 + 2 \cdot 10^0$;

2) $1010{,}1_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1}$;

3) $1AD{,}B_{16} = 1 \cdot 16^2 + 10 \cdot 16^1 + 13 \cdot 16^0 + 11 \cdot 16^{-1}$.

Обратный перевод — это перевод из десятеричной системы счисления в любую другую.

1. Перевод целых чисел.

Исходное число последовательно делят на основание системы, записывая при этом остатки от деления. Результат — это полученные остатки, записанные в обратном порядке.

Например, $47_{10} = 101111_2$; $47_{10} = 2F_{16}$.

2. Перевод правильных дробей.

Дробную часть числа последовательно умножают на основание системы до тех пор, пока она не превратится в 0, или пока не появится период.

Пример: $0{,}14_{10} = 0{,}10755_8$.

0	14
1	12
0	96
7	68
5	64
5	12

3. Перевод неправильных дробей.

При переводе смешанного числа следует переводить его целую и дробную части отдельно.

3.1. Для перевода целой части числа ее, а затем целые части получающихся частных от деления следует последовательно делить на основание P до тех пор, пока очередная целая часть частного не окажется равной 0. Остатки от деления, записанные последовательно справа налево, образуют целую часть числа в системе счисления с основанием P.

3.2. Для перевода дробной части числа ее, а затем дробные части получающихся произведений следует последовательно умножать на основание P до тех пор, пока очередная дробная часть произведения не окажется равной 0, или не будет достигнута нужная точность дроби. Целые части произведений,

записанные после запятой последовательно слева направо, образуют дробную часть числа в системе счисления с основанием Р.

Пусть требуется перевести смешанное число из десятичной в двоичную систему счисления на примере числа 46,625.

1. Переводим целую часть числа:

46:2 = 23 (остаток 0).

23:2 = 11 (остаток 1).

11:2 = 5 (остаток 1).

5:2 = 2 (остаток 1).

2:2 = 1 (остаток 0).

1:2 = 0 (остаток 1).

Записываем остатки последовательно справа налево — 101110, т. е. $46_{10} = 101110_2$.

2. Переводим дробную часть числа:

$0{,}625 \times 2 = 1{,}250$.

$0{,}250 \times 2 = 0{,}500$.

$0{,}500 \times 2 = 1{,}000$ (дробная часть равна 0 = > стоп).

Записываем целые части получающихся произведений после запятой последовательно слева направо — 0,101, т. е. $0{,}625_{10} = 0{,}101_2$.

Окончательно: $46{,}625_{10} = 101110{,}101_2$.

С учетом того, что на входах и выходах электронных устройств часть чисел требуется выражать в десятеричной системе счисления, был разработан ряд кодов: 8-4-2-1 (двоично-десятичный), 7-4-2-1, 2-4-2-1.

Кроме двоичной и десятичной при работе с компьютером часто используются также *двоично-десятичная* и *шестнадцатеричная системы счисления* (табл. 1).

Шестнадцатеричная система счисления часто используется при программировании. Перевод чисел из шестнадцатеричной системы счисления в двоичную систему весьма прост — он выполняется поразрядно.

Для изображения цифр больше 9 в шестнадцатеричной системе счисления применяются буквы: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Например, шестнадцатеричное число F17B в двоичной системе выглядит так: 1111000101111011, а в десятичной — 61 819.

Двоично-десятичная система счисления получила большое распространение в современных компьютерах ввиду легкости перевода в десятичную систему и обратно. Она используется там, где основное внимание уделяется не простоте технического построения машины, а удобству работы пользователя. В этой системе счисления все десятичные цифры отдельно кодируются четырьмя двоичными цифрами и в таком виде записываются последовательно друг за другом.

Двоично-десятичная система не экономична с точки зрения реализации технического построения машины (примерно на 20% увеличивается требуемое оборудование), но очень удобна при подготовке задач и при программировании. В двоично-десятичной системе счисления основанием системы

счисления является число 10, но каждая десятичная цифра (0, 1, ..., 9) кодируется двоичными цифрами.

Таблица 1. Перевод цифр из двоичной системы счисления в восьмеричную и десятичную и наоборот

Триада	Восьмеричная цифра	Тетрада	Шестнадцатеричная цифра	Десятичное число	Двоично-десятичная запись
000	0	0000	0	0	0000-0000
001	1	0001	1	1	0000-0001
010	2	0010	2	2	0000-0010
011	3	0011	3	3	0000-0011
100	4	0100	4	4	0000-0100
101	5	0101	5	5	0000-0101
110	6	0110	6	6	0000-0110
111	7	0111	7	7	0000-0111
		1000	8	8	0000-1000
		1001	9	9	0000-1001
		1010	A	10	0001-0000
		1011	B	11	0001-0001
		1100	C	12	0001-0010
		1101	D	13	0001-0011
		1110	E	14	0001-0100
		1111	F	15	0001-0101

Перевод целого числа из *p*-ичной системы счисления в десятичную осуществляется путем представления числа в виде степенного ряда с основанием той системы, из которой число переводится, т. е. число записывается в развернутой форме. Затем подсчитывается значение суммы, причем все арифметические действия осуществляются в десятичной системе.

Пример 1.

а) Перевести $10101101 \rightarrow X_{10}$.

$10101101_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 173_{10}$.

Ответ: $1010110_2 = 1173_{10}$.

б) Перевести $703_8 \rightarrow X_{10}$.

$703_8 = 7 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0 = 451_{10}$.

Ответ: $703_8 = 451_{10}$.

в) Перевести $B2E_{16} \rightarrow X_{10}$.

$B2E_{16} = 11 \cdot 16^2 + 2 \cdot 16^1 + 14 \cdot 16^0 = 2862_{10}$.

Ответ: $B2E_{16} = 2862_{10}$.

Перевод правильной конечной *p*-ичной дроби в десятичную систему счисления осуществляется аналогично переводу целого числа через развернутую форму представления числа.

Пример 2.

а) Перевести $0{,}1101_2 \rightarrow X_{10}$.

$0{,}1101_2 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = 0{,}8125_{10}$.

Ответ: $0{,}1101_2 = 0{,}8125_{10}$.

б) Перевести $0{,}04_8 \rightarrow X_{10}$.

$0{,}04_8 = 0 \cdot 8^{-1} + 4 \cdot 8^{-2} = 0{,}0625_{10}$.

Ответ: $0{,}04_8 = 0{,}0625_{10}$.

в) Перевести $0{,}C4_{16} \rightarrow X_{10}$.

$0{,}C4_{16} = 12 \cdot 16^{-1} + 4 \cdot 16^{-2} = 0{,}765625_{10}$.

Ответ: $0{,}C4_{16} = 0{,}465625_{10}$.

При переводе неправильной конечной *p*-ичной дроби в десятичную систему счисления необходимо перевести как целую, так и дробную части с помощью развернутой формы представления чисел.

Пример 3.

Перевести $1001101{,}1101_2 \rightarrow X_{10}$.

$1001101{,}1101_2 = 2^6 + 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2} + 2^{-4} = 77{,}8125_{10}$.

Ответ: $1001101{,}1101_2 = 77{,}8125_{10}$.

Перевод целого числа из десятичной системы счисления в *p*-ичную осуществляется последовательным целочисленным делением десятичного числа на основание той системы, в которую оно переводится, до тех пор, пока не получится частное меньшее этого основания. Число в новой системе счисления записывается в виде остатков от деления в обратном порядке, начиная с последнего частного от деления.

Пример 4.

а) Перевести $181_{10} \rightarrow X_8$.

181	8	
176	22	8
5	16	**2**
	6	

Ответ: $181_{10} = 265_8$.

б) Перевести $622_{10} \rightarrow X_{16}$.

622	16	
48	38	16
142	32	**2**
128	**6**	
4		

Результат $622_{10} = 26E_{16}$.

Перевод правильной конечной дроби из десятичной системы счисления в *p*-ичную осуществляется последовательным умножением на основание той системы, в которую она переводится, до тех пор, пока дробная часть произведения не станет равной нулю, или не выделится период. При этом умножаются только дробные части. Дробь в новой системе счисления записывается в виде последовательности целых частей произведений, начиная с первого.

Пример 5.

а) Перевести $0{,}3125_{10} \rightarrow X_8$.

0	3125 × 8
2	5000 × 8
4	0000

Ответ: $0{,}3125_{10} = 0{,}24_8$.

б) Перевести $0{,}65_{10} \rightarrow X_2$.

0	65× 2
1	3 × 2
0	6 × 2
1	2 × 2
0	4 × 2
0	8 × 2
1	6 × 2
	. . .

Ответ: $0{,}65_{10} \approx 0{,}10(1001)_2$.

При переводе неправильной конечной десятичной дроби в *p*-ичную систему счисления необходимо отдельно перевести целую часть и отдельно дробную, а затем их соединить.

Пример 6.

Перевести $23{,}125_{10} \rightarrow X_2$.

а) Переведем целую часть:

23	2			
22	11	2		
1	10	5	2	
	1	4	2	2
		1	2	**1**
			0	

б) Переведем дробную часть:

0	125× 2
0	25 × 2
0	5 × 2
1	0

Таким образом, $23_{10} = 10111_2$; $0{,}125_{10} = 0{,}001_2$.

Ответ: $23{,}125_{10} = 10111{,}001_2$.

Необходимо отметить, что целые числа остаются целыми, а правильные дроби — правильными в любой системе счисления.

Для **перевода восьмеричного или шестнадцатеричного числа в двоичную систему счисления** достаточно заменить каждую цифру этого числа соот-

ветствующим трехразрядным двоичным числом (триадой) или четырехразрядным двоичным числом (тетрадой) и отбросить незначащие нули в старших и младших разрядах.

Для **перевода из двоичной в восьмеричную или шестнадцатеричную систему счисления** поступают следующим образом: двигаясь от точки разделения целой и дробной части числа влево и вправо, разбивают двоичное число на группы по три или четыре разряда, дополняют при необходимости нулями крайние левую и правую группы. Затем триаду или тетраду заменяют соответствующей восьмеричной или шестнадцатеричной цифрой.

Контрольные вопросы:

1. Что называется, системой счисления?
2. На какие два типа можно разделить все системы счисления?
3. Что называется основанием системы счисления?
4. Для чего используется перевод чисел из одной системы счисления в другую?

Лекция 2. Форматы и коды чисел в ЭВМ

План:

1. Форматы и коды чисел в ЭВМ.
2. Представление чисел в ЭВМ.

1. Форматы и коды чисел в ЭВМ

В качестве единицы информации Клод Шеннон предложил принять один бит (*англ*. bit — binary digit — двоичная цифра). **Бит** в теории информации — количество информации, необходимое для различения двух равновероятных сообщений («орел — решка», «чет — нечет» и т. п.).

В вычислительной технике битом называют наименьшую «порцию» памяти компьютера, необходимую для хранения одного из знаков 0 и 1, используемых для машинного представления данных команд. За единицу информации можно было бы выбрать количество информации, необходимое для различения, например, десяти равновероятных сообщений. Это будет не двоичная (бит), а десятичная (дит) единица информации. Поскольку бит — слишком мелкая единица измерения, на практике чаще применяется более крупная единица — байт, равная восьми битам. В частности, восемь бит требуется для того, чтобы закодировать любой из 256 символов основного компьютерного кода ASCII ($256 = 2^8$).

Используются также более крупные производные единицы информации:

- килобайт (Кбайт) = 1024 байт = 2^{10} байт;
- мегабайт (Мбайт) = 1024 Кбайт = 2^{20} байт;
- гигабайт (Гбайт) = 1024 Мбайт = 2^{30} байт.

В последнее время в связи с увеличением объемов обрабатываемой информации входят в употребление такие производные единицы, как:

- терабайт (Тбайт) = 1024 Гбайт – 2^{40} байт;
- петабайт (Пбайт) = 1024 Тбайт = 2^{50} байт;
- экзобайт = 10^{18} Мбайт и пр.

Для описания скорости передачи данных можно использовать термин бод. Число бод равно количеству значащих изменений сигнала (потенциала, фазы, частоты), происходящих в секунду. Первоначально бод использовался в телеграфии. Для двоичных сигналов нередко принимают, что бод равен биту в секунду, например 1200 бод = 1200 бит/с. Однако единого мнения о правильности использования этого термина нет, особенно при высоких скоростях, где число бит в секунду не совпадает с числом бод.

Код (code) — совокупность знаков, символов и правил представления информации.

В частности, можно различать двоичный и троичный коды. Алфавит первого ограничен двумя символами (0, 1), а второго — тремя символами (–1, 0, +1). Сигналы, реализующие коды, обладают одной из следующих характеристик:

- униполярный код (значения сигнала равны 0, +1 либо 0, –1);
- полярный код (значения сигнала равны –1, +1);
- биполярный код (значения равны –1, 0, +1).

Биполярные коды часто используются в каналах передачи данных (рис. 1). Здесь единицы представляются чередующимися положительными и отрицательными импульсами. Отсутствие импульсов определяет состояние «нуль». Биполярное кодирование обеспечивает обнаружение одиночной ошибки. Так, если вместо нуля появится единица, либо единица ошибочно сменится на нуль, то это легко обнаруживается. В обоих случаях нарушается чередование полярности импульсов.

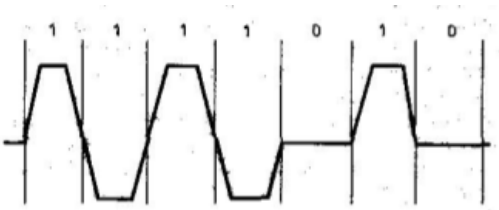


Рисунок 1. Биполярное кодирование

Рассмотрим методы дискретного представления информации, или кодирования (которые, надо сказать, появились задолго до вычислительных машин). Первым широко известным примером является Азбука Морзе (АМ), в которой буквы латиницы (или кириллицы) и цифры кодируются сочетаниями из «точек» и «тире». Кодируемые (обозначаемые) элементы входного алфавита обычно называют символами.

Символом (служит условным знаком какого-нибудь понятия, явления), как правило, являются цифра, буква, знак пунктуации или иероглиф естественного языка, знак препинания, знак пробела, специальный знак, символ операции. Кроме этого, учитываются управляющие («непечатные») символы.

Кодирующие (обозначающие) элементы выходного алфавита называются знаками; количество различных знаков в выходном алфавите назовем значно-

стью (-арностью, -ичностью), количество знаков в кодирующей последовательности для одного символа — разрядностью кода; последовательным кодом является такой, в котором знаки следуют один за другим во времени (например, радио- или оптические сигналы либо передача по двум проводам, 2-жильному кабелю), параллельным — тот, в котором знаки передаются одновременно (например, по четырем проводам, 4-жильному кабелю), образуя символ (т. е. символ передается в один прием, в один момент времени).

Поскольку знаки передаются последовательно (электрические импульсы, звуковые или оптические сигналы разной длины, соответствующие «точкам» и «тире»), АМ есть последовательный код (можно представить себе некоторое табло, на котором одновременно вспыхивали бы сочетания лампочек, образующих точки и тире, представляющие передаваемый символ, но автору не приходилось слышать о подобных абсурдных приспособлениях). Первые опыты телеграфной и радиосвязи осуществлялись именно посредством АМ, причем приемное устройство записывало импульсы переменной длины в виде «точек» и «тире» на движущуюся телеграфную ленту, однако уже в начале XX в. был осуществлен переход на 5-разрядный (5-битовый) телеграфный код.

Наиболее известные коды, которые использовались первоначально для связи, кодирования данных, а затем для представления информации в ЭВМ:

– код Бодо — 5-разрядный код, бывший в прошлом европейским стандартом для телеграфной связи (другое название — IA-1 — International Alphabet #1);

– М-2 (российское обозначение) или IA-2 (международное обозначение) — телеграфный код, предложенный Международным консультативным комитетом по телефонии и телеграфии (МККТТ) и заменивший код Бодо;

– ASCII (American Standard Code for Information Interchange) — стандартный 7-битовый код для передачи данных, поддерживает 128 символов, включающих заглавные и строчные символы латиницы, цифры, специальные значки и управляющие символы. Этот код, к которому были добавлены некоторые национальные символы (10 бинарных комбинаций), был принят Международной организацией по стандартизации (ISO) как стандарт ISO-7;

– EBCDIC (Expanded Binary Coded Decimal Interchange Code) — 8-разрядный код, предложенный фирмой IBM для машин серий IBM/360-375 (внутреннее представление данных в памяти), а затем распространившийся и на системы других производителей;

– ASCII-8 — 8-разрядный код, принятый для внутреннего и внешнего представления данных в вычислительных системах. Включает стандартную часть (128 символов) и национальную (128 символов);

– код Холлерита, предложенный для ПК (1913 г.), затем использовавшийся для кодирования информации перед вводом в ЭВМ с перфокарт.

Таблица 2. Разрядность некоторых наиболее известных кодов

Код	Разрядность
IA-2 (M-2, MKKTT-2)	5
Baudot (Бодо)	5
ISO-7 (IA-5, ASCII-7, USASCII, ANSI X3.4)	7
EBCDIC	8
ASCII-8	8
Hollerith (перфокарты Холлерита)	12

Одним из «последних слов» в процессе развития систем символьного кодирования является универсальный код UNICODE (UNIversal CODE) — стандарт 16-разрядного кодирования символов. Стандарт UNICODE разработан техническим комитетом, в который вошли представители ряда ведущих фирм. Он определяет идентификацию различных символов: букв, иероглифов, цифр и т. д. Код может использоваться вместо 7-, 8-битовых, в том числе и ASCII. Поскольку в 16-разрядном UNICODE можно закодировать 65 536 символов вместо 128 в ASCII, то отпадает необходимость в создании модификаций таблиц кодов. Это существенно упрощает обработку текстовых файлов, хотя и несколько увеличивает их размеры.

Таблица 3. Некоторые кодовые таблицы

Наименование кодовой страницы	Интерпретация кодовой страницы
Latin-1	Международный стандарт (ISO-8859-1) для интерпретации 2-й половины (128-256) кода ASCII, таблица предназначена для латиницы
Latin-8	Международный стандарт (ISO-8859-8) для иврита
Latin-C	Международный стандарт (ISO-8859) для кириллицы
CP-437	Стандарт IBM для интерпретации 2-й половины (128–256) кода ASCII, таблица предназначена для греческого алфавита
CP-850	Стандарт IBM для восточноевропейских алфавитов
CP-852	Стандарт IBM для греческого алфавита
CP-862	Стандарт IBM для иврита
CP-866	Стандарт IBM для русской кириллицы

UNICODE охватывает 28 000 букв, знаков, слогов, иероглифов национальных языков мира, 30 000 мест в UNICODE зарезервировано. Использование этого резерва дает возможность пользователям вводить математические или технические символы, а также создавать свои собственные символы.

Единая стандартизация языковых форматов наводит порядок в международном кодировании алфавитов различных языков. Здесь учтено также то, что в таких языках, как иврит и арабский, текст пишется справа налево.

При передаче данных часто используются избыточные коды, т. е. такие, которые за счет усложнения структуры позволяют повысить надежность передачи данных. К ним в первую очередь относятся коды с обнаружением ошибок. Чаще всего это циклические избыточные коды. Простая разновид-

ность такого кода — код с контролем по четности. Широко используется для обнаружения ошибок в блоках данных также код контроля циклической избыточности CRC. Он определяется на основе содержимого блока данных перед его передачей, включается в одно из полей блока, а затем повторно вычисляется после передачи. Несовпадение результатов свидетельствует об ошибке в передаваемом содержимом.

Таблица 4. Фрагменты некоторых кодовых таблиц

Символ	IA-2	Бодо	ISO-7	EBCOC	ASCII-8	Холлерт
A	03	10	41	C1	A1	900
B	19	06	42	C2	A2	880
C	0E	16	43	C3	A3	840
D	09	1E	44	C4	A4	820
a			61	81	E1	
b			62	82	E2	
c			63	83	E3	
d			64	84	E4	
. (точка)	1C	05	2E	4B	4E	842
, (запятая)	0C	09	2C	6B	4C	242
: (двоеточие)	1E		3B	5E	5B	40A
? (вопрос)	10	0D	3F	6F	5F	206

Важное значение имеют коды с исправлением ошибок. Использование этих кодов позволяет с большой вероятностью не только обнаруживать, но и исправлять возникшие при передаче ошибки Хемминга (код Хемминга, позволяющий исправлять одиночные ошибки, появляющиеся в блоках данных).

2. Представление чисел в ЭВМ

Как известно, в ЭВМ применяется двоичная система счисления. Может быть доказано, что при этом на построение ЭВМ тратится меньшее количество базовых аппаратных элементов — «вентилей». Точнее, оптимальным основанием системы счисления по критерию «минимум аппаратных расходов» является основание натурального логарифма e ≈ 2,72.

Однако по ряду очевидных причин для ЭВМ принято P = 2. Достаточно вспомнить, что одна из первых ЭВМ ENIAK содержала 17 468 электронных ламп, имела размеры около 6 м в высоту и 30 м в длину. Обилие применяемых вакуумных ламп, габаритные размеры машины отчасти объяснялись тем, что она работала с десятичными числами.

В ЭВМ применяются две формы представления чисел:

– естественная форма, или форма с фиксированной запятой (точкой), — ФЗ (ФТ);

– нормальная форма, или форма с плавающей запятой (точкой), — ПЗ (ПТ).

Фиксированная запятая (точка). В форме представления «фиксированной запятой (точкой)» числа изображаются в виде последовательности цифр с постоянным для всех чисел положением запятой, отделяющей целую часть от дробной.

Например, пусть числа представлены в десятичной системе счисления и имеют пять разрядов в целой части числа (до запятой) и в дробной части (после запятой). Числа, записанные в такую разрядную сетку, имеют вид:

+00721.35500.

+00000.00328.

–10301.20260.

Эта форма наиболее проста, естественна, но имеет небольшой диапазон представления чисел и поэтому чаще всего неприемлема при вычислениях.

Диапазон значащих чисел N в системе счисления с основанием P при наличии m разрядов в целой части и s разрядов в дробной части числа (без учета знака числа) будет таким: $P^{-z} \le N \le P^m - P^{-z}$.

Например, при P = 2, m = 10 и s = 6 числа изменяются в диапазоне $0{,}015 < N < 1024$. Если в результате операции получится число, выходящее за допустимые пределы, произойдет переполнение разрядной сетки, и дальнейшие вычисления теряют смысл. В современных компьютерах естественная форма представления используется как вспомогательная и только для целых чисел.

В памяти ЭВМ числа с фиксированной точкой хранятся в трех форматах:

а) полуслово — это обычно 16 бит, или 2 байта;

б) слово — 32 бита, или 4 байта;

в) двойное слово — 64 бита, или 8 байтов.

Отрицательные числа с ФТ записываются в разрядную сетку в дополнительных кодах, которые образуются прибавлением единицы к младшему разряду обратного кода. Обратный код получается заменой единиц на нули, а нулей на единицы в прямом двоичном коде.

Плавающая запятая (точка). В форме представления с плавающей запятой (точкой) число изображается в виде двух групп цифр:

– мантисса;

– порядок.

При этом абсолютная величина мантиссы должна быть меньше 1, а порядок должен быть целым числом. В общем виде число в форме с плавающей запятой может быть представлено так: $\mathbf{N = \pm M \times P^{\pm r}}$, где M — мантисса числа ($|M| < 1$); r — порядок числа (целое число); P — основание системы счисления.

Например, приведенные ранее числа в нормальной форме запишутся следующим образом: $+0{,}721355 \times 10^3$; $+ 0{,}328 \times 10^{-3}$; $-0{,}103012026 \times 10^5$.

Нормальная форма представления обеспечивает большой диапазон отображения чисел и является основной в современных компьютерах. Так, диапазон значащих чисел в системе счисления с основанием P при наличии m разрядов у мантиссы и s разрядов у порядка (без учета знаковых разрядов порядка и мантиссы) будет $P^{-m} \times P^{-(p^S-1)} \le N \le (1-P^{-m}) \times P^{(P^S-1)}$.

Например, при P = 2, m = 22 и s = 10 диапазон чисел простирается примерно от 10^{-300} до 10^{300}. Для сравнения: количество секунд, которые прошли с момента образования планет Солнечной системы, составляет около 10^{18}.

Следует заметить, что все числа с плавающей запятой хранятся в машине в так называемом нормализованном виде.

Нормализованным называют такое число, старший разряд мантиссы которого больше нуля. У нормализованных двоичных чисел, следовательно, $0{,}5 < |M| < 1$.

Нормализованные, т. е. приведенные к правильной дроби, числа:

$$10{,}35_{10} = 0{,}1035_{10} \times 10^{+2};\ 0{,}00007245_8 = 0{,}7245_8 \times 8^{-4};$$
$$5C{,}9B_{16} = 0{,}F5C9B_{16} \times 16^{+3}.$$

В памяти ЭВМ числа с ПТ хранятся в двух форматах:

– слово — 32 бита, или 4 байта;

– двойное слово — 64 бита, или 8 байт.

Разрядная сетка для чисел с ПТ имеет следующую структуру:

– нулевой разряд — это знак числа (0 — «минус», 1 — «плюс»);

– с 1 по 7 разряд записывается порядок в прямом двоичном коде, пустые разряды заполняются нулями. В первом разряде указывается знак порядка (1 — «плюс» или 0 — «минус»);

– с 8 по 31 (63) указывается мантисса, слева направо без нуля целых в прямом двоичном коде и для отрицательных чисел и пустые разряды заполняются нулями.

Знак числа обычно кодируется двоичной цифрой, при этом:

− код 0 означает знак + (плюс);

− код 1 — знак – (минус).

Для алгебраического представления чисел, т. е. для представления чисел с учетом их знака, в вычислительных машинах используются специальные коды:

– прямой код числа;

– обратный код;

– дополнительный код.

При этом два последних кода позволяют заменить неудобную для компьютера операцию вычитания на операцию сложения с отрицательным числом. Дополнительный код обеспечивает более быстрое выполнение операций, поэтому в компьютере чаще всего применяется именно он.

Прямой код числа N обозначим $[N]_{пр}$.

Пусть $N = a_1, a_2, a_3, \ldots, a_m$, тогда:

при $N > 0$, $[N]_{пр} = 0, a_1, a_2, a_3, \ldots, a_m$;

при $N < 0$, $[N]_{пр} = 1, a_1, a_2, a_3, \ldots, a_m$;

при $N = 0$ имеет место неоднозначность $[0]_{пр} = 0, 0\ldots = 1, 0\ldots$

Если при сложении в ЭВМ оба слагаемых имеют одинаковый знак, то операция сложения выполняется обычным путем. Если при сложении слагаемые имеют разные знаки, то сначала необходимо выявить большее по абсолютной величине число, из него произвести вычитание меньшего по абсолютной величине числа и разности присвоить знак большего числа.

Выполнение операций умножения и деления в прямом коде выполняется обычным образом, но знак результата определяется по совпадению или несовпадению знаков участвовавших в операции чисел.

Операцию вычитания в этом коде нельзя заменить операцией сложения с отрицательным числом, поэтому возникают сложности, связанные с займом значений из старших разрядов уменьшаемого числа. В связи с этим прямой код в ЭВМ почти не применяется.

Обратный код числа N обозначим $[N]_{обр}$.

Пусть $N = a_1, a_2, a_3, \ldots, a_m$ и а обозначает инверсию а, т. е. если а = 1, то а = 0, и наоборот. Тогда:

при $N > 0$, $[N]_{обр} = 0, a_1, a_2, a_3, \ldots, a_m$;

при $N < 0$, $[N]_{обр} = 1, a_1, a_2, a_3, \ldots, a_m$;

при N = 0 имеет место неоднозначность $[0]_{обр} = 0{,}00\ldots0 = 1{,}11\ldots1$.

Для того чтобы получить обратный код отрицательного числа, необходимо все цифры этого числа инвертировать, т. е. в знаковом разряде поставить 1, во всех значащих разрядах нули заменить единицами, а единицы нулями.

Например,

для $N = 1011$ $[N]_{обр} = 0{,}1011$;

для $N = –1011$ $[N]_{обр} = 1{,}0100$.

Дополнительный код числа N обозначим $[N]_{доп}$.

Пусть, как и выше, $N = a_1, a_2, a_3, \ldots, a_m$ и а обозначает величину, обратную а (инверсию а), т. е. если а = 1, то а = 0, и наоборот. Тогда:

при $N \geq 0$, $[N]_{доп} = 0, a_1, a_2, a_3, \ldots, a_m$;

при $N \leq 0$, $[N]_{доп} = 1, a_1, a_2, a_3, \ldots, a_m + 0{,}00\ldots1$.

Для того чтобы получить дополнительный код отрицательного числа, необходимо все его цифры инвертировать (в знаковом разряде поставить единицу, во всех значащих разрядах нули заменить единицами, а единицы — нулями) и затем к младшему разряду прибавить единицу. В случае возникновения переноса из первого после запятой разряда в знаковый разряд к числу следует прибавить единицу в младший разряд.

Например,

для $N = 1011$, $[N]_{доп} = 0{,}1011$; для $N = –1100$, $[N]_{доп} = 1{,}0100$;

для $N = –0000$, $[N]_{доп} = 10{,}0000 = 0{,}0000$ (1 исчезает). Неоднозначность в изображении 0 нет.

Эмпирическое правило: для получения дополнительного кода отрицательного числа необходимо все символы этого числа инвертировать, кроме последней (младшей) единицы и тех нулей, которые за ней следуют.

Контрольные вопросы:

1. Какие 2 формы представления чисел применяются в ЭВМ?
2. В каких 3 форматах хранятся в памяти ЭВМ числа с фиксированной точкой?
3. Какие специальные коды используются для алгебраического представления чисел?

Лекция 3. Арифметические основы ЭВМ

План:

1. Элементы двоичной арифметики.
2. Алгебраическое представление двоичных чисел.

1. Элементы двоичной арифметики

Рассмотрим, как выполняются арифметические действия в двоичной системе. Для этого проведем анализ таблиц сложения и умножения в двоичной системе:

$0 + 0 = 0$, $0 \times 0 = 0$, $0 + 1 = 1$, $0 \times 1 = 0$, $1 + 1 = 10$.

Арифметические операции

Для описания принципов работы ЭВМ в подавляющем большинстве случаев используется двоичная система счисления, в которой поступление электронного сигнала означает логическую 1, а отсутствие электронного сигнала — логический 0. Рассмотрим арифметические операции над двоичными числами.

Правила двоичной арифметики:

Сложение: $0 + 0 = 0$ $\quad 1 + 0 = 1$

$0 + 1 = 1$ $\quad 1 + 1 = \boxed{1}0$

перенос единицы в старший разряд

Вычитание: $0 - 0 = 0$ $\quad 1 - 1 = 0$

$1 - 0 = 1$ $\quad \boxed{1}0 - 1 = 1$

заем единицы в старшем разряде

Умножение: $0 \cdot 0 = 0$ $\quad 1 \cdot 0 = 0$

$0 \cdot 1 = 0$ $\quad 1 \cdot 1 = 1$

При сложении и вычитании чисел со знаком, если при этом бит старшего разряда равен 0, то число положительное, а если равен 1, то результат — отрицательное число в дополнительном коде. Если требуется найти абсолютное значение результата, последний надо представить в обратном коде, а затем прибавить к нему единицу

2. Алгебраическое представление двоичных чисел

Пример 1. Вычислить $58 - 23 = 35$. Для этого:

а) определим дополнительный код числа $23_{10} = 00010111_2$, к обратному коду данного числа 11101000 прибавим единицу 00000001 и получим 11101001 — это и есть дополнительный код числа 23;

б) вычислим разность:

+ 00111010	число 58_{10}
11101001	дополнительный код 23_{10}
$1\boxed{0}0100011$	разность 35, т. е. число положительное, следовательно, первую единицу отбрасываем.

Пример 2. Вычислим $26 - 34 = -8$.

а) определим дополнительный код числа $34_{10} = 00100010_2$, т. е. к обратному коду 1011101 прибавим 00000001 и получим 11011110 — это дополнительный код числа 34;

б) вычислим разность:

+ 00011010	число 26_{10}
11011110	дополнительный код 34_{10}
11111000	разность в форме дополнения, т. к. в старшем разряде 1 (число отрицательное);

в) определим абсолютное значение разности:

11111000	дополнительный код разности
00000111	обратный код
+ 00000001	единица, добавляемая к обратному коду
00001000	абсолютное значение разности (8).

Следует обратить внимание на аналогию в правилах выполнения арифметических действий в двоичной и десятичных системах счисления: если при сложении двух двоичных чисел (точнее, представленных в двоичной системе счисления) сумма цифр окажется больше единицы, то возникает перенос в старший разряд; если уменьшаемая цифра меньше вычитаемой, то нужно сделать «заем» единицы в старшем разряде.

Анализируя примеры умножения в двоичной системе счисления, необходимо обратить внимание на одну важную особенность выполнения этой операции в данной системе. Так как очередная цифра множителя может быть только 1 или 0, то промежуточное произведение равно либо множимому, либо 0. Таким образом, операция умножения в двоичной системе фактически не производится: в качестве промежуточного произведения записывается либо множимое, либо 0, а затем промежуточные произведения суммируются. Иначе говоря, операция умножения заменяется последовательным сложением.

Как уже известно, дополнительный код используется для вычитания чисел в компьютерах и позволяет эту операцию свести к сложению чисел.

Правила выполнения вычитания с дополнительным числом следующие. Чтобы вычесть число A из числа B, достаточно сложить B с дополнительным числом к A и отбросить перенос в соседний старший разряд. Например, чтобы вычесть 623 из 842, достаточно сложить 842 с 377; отбросив перенос, получим 219 (842 – 623 = 219).

Таким образом, важнейшее преимущество двоичной арифметики заключается в том, что она позволяет все арифметические действия свести к одному — сложению, а это значительно упрощает устройство процессора ЭВМ. Изложенные здесь основные принципы положены в основу функционирования элементов и узлов ЭВМ (триггер, сумматор, полусумматор).

Контрольные вопросы:

1. Как выполняются арифметические действия в двоичной системе?
2. Какие правила используются при выполнении арифметических действий в двоичной системе счисления?

Лекция 4. Представление информации в ЭВМ

План:

1. Определение и классификация информации.
2. Измерение количества информации.
3. Типы и структуры данных.
4. Двоичное кодирование мультимедиа информации.
5. Двоичное кодирование звуковой информации.
6. Сжатие информации.
7. Кодирование видеоинформации.

1. Определение и классификация информации

Понятие «информация» является таким же фундаментальным, как понятия «материя», «энергия» и другие философские категории. Это атрибут, свойство сложных систем, связанное с их развитием и самоорганизацией. Известно большое количество различных определений информации, отличие информации от данных, знаний и пр. Мы здесь ограничимся только рассмотрением некоторых практически важных понятий и определений.

В настоящее время наука пытается найти общие свойства и закономерности, присущие многогранному понятию «информация», но пока это понятие во многом остается интуитивным и получает различные смысловые наполнения в различных отраслях человеческой деятельности:

- в обиходе информацией называют любые данные или факты, которые кого-либо интересуют. Например, сообщение о каких-либо событиях, о чьей-либо деятельности и т. п. «Информировать» в этом смысле означает «сообщить нечто, неизвестное раньше»;
- в технике под информацией понимают сообщения, передаваемые в форме знаков или сигналов;
- в кибернетике под информацией понимают ту часть знаний, которая используется для ориентирования, активного действия, управления, т. е. в целях сохранения, совершенствования, развития системы.

Информация может классифицироваться, например, по следующим основаниям:

а) признаки, отражающие структуру данных и форму представления информации (табл. 5);

б) содержание предметной области применения (табл. 6).

Исторически первой технологической формой получения, передачи, хранения информации являлось аналоговое (непрерывное) представление звукового, оптического, электрического или другого сигнала (сообщения). Магнитная, аудио- и видеозапись, фотографирование, запись на шеллачные или виниловые грампластинки, проводное и радиовещание — основные способы хранения и передачи информации в аналоговой форме.

Дискретный сигнал — сигнал, имеющий конечное, обычно небольшое, число значений.

Таблица 5. Некоторые классы информации (по структуре и форме)

Основание для классификации	**Классы информации**			
По уровням сложности	Сигнал	Сообщение, документ	Информационный массив	Информационный ресурс
По типу сигнала	Аналоговая (непрерывная)	Цифровая (дискретная)		
По уровням доступа и организации	Данные в регистровой памяти	Данные в оперативной памяти	Файлы данных на внешних устройствах	Базы данных
По способам кодирования и представления (данные, файлы и БД)	Цифровая (вычислительные данные, двоичные)	Символьная (алфавитно-цифровая, строчная)	Графическая	
По организации данных (файлы и БД)	Табличная	Текстовая	Графическая	

Таблица 6. Классификация информации по содержанию

Тип информации	**Содержание**	**Поставщик содержания**
Биржевая и финансовая	Индексы рынка, котировки, цены, обзоры	Биржи, банки, службы финансовой информации
Экономическая и демографическая	Первичная и вторичная; национальная, региональная статистика	Переписи, опросы, аналитические исследования
Коммерческая	Данные о предприятиях, товарах, услугах	Аналитические службы
Деловые новости	Состояние рынка, события в области экономики	Службы фильтрации, агентства новостей
Научно-техническая	Фундаментальные, прикладные науки	Центры НТИ, издательства, библиотеки
Правовая	Нормативно-правовые акты	Законодательные органы, Минюст РФ
Медицинская	Медучреждения, болезни, лекарства, яды	Информационные центры, библиотеки, госпитали
Потребительская и развлекательная	Образование, музыка, музеи, библиотеки, кино	Справочные службы, учреждения
Бытовая	Погода, туризм, справочники	Информационные службы

Практически всегда дискретный сигнал имеет два либо три значения. Нередко его называют также цифровым сигналом.

В цифровых системах используются двоичные сигналы, имеющие значения (+ , –). Вместе с тем при передаче данных в большинстве случаев применяются троичные сигналы со значениями: (+), (0), (–). Здесь «единица» представляется отсутствием потенциала в канале, тогда как «нуль» характеризуется положительным либо отрицательным импульсом. При этом полярность импульсов, представляющих «нули», должна чередоваться, т. е. за положительным (+) импульсом должен следовать отрицательный (–), и наоборот. В форме троичного сигнала осуществляется не только кодирование передаваемых данных, но также обеспечиваются синхронизация работы канала и проверка целостности данных.

Дискретные сигналы по сравнению с аналоговыми имеют ряд преимуществ: помехоустойчивость, легкость восстановления формы, простота аппаратуры передачи.

Более чем тридцатилетнее развитие теории и практики ЭВМ приводит к вытеснению (в том числе и на бытовом уровне) аналоговых устройств и сигналов цифровыми. Наиболее популярным примером является, несомненно, аудиокомпакт-диск (digital audio CD).

В этом случае звуковой сигнал сначала преобразуется в дискретную аппроксимацию («многоуровневый ступенчатый сигнал»), при этом происходит квантование во времени, которое заключается в измерении в дискретные промежутки времени необходимого параметра аналогового сигнала.

Кроме этого, осуществляется квантование по амплитуде сигнала. Элемент разбиения этого сигнала именуют квантом. Поэтому говорят, что квантование заключается в делении на кванты. При квантовании аналогового сигнала происходит округление его мгновенных значений до некоторой заданной фиксированной величины, называемой уровнем. Расстояние между соседними уровнями именуется шагом. Из-за округления квантование всегда связано с определенным искажением сигнала. Уменьшение искажения требует увеличения числа уровней квантования и уменьшения шага квантования.

При квантовании по амплитуде каждая ступенька представляется последовательностью бинарных двухуровневых цифровых сигналов. Принятый в настоящее время стандарт CD использует так называемый 16-разрядный звук с частотой сканирования 44 кГц.

2. Измерение количества информации

Термин «информация» имеет корень «form» (форма), что разумно трактовать как «информирование — придание формы, вывод из состояния неопределенности, бесформенности», поэтому логично подходить к определению понятия «количество информации», исходя из того, что информацию, содержащуюся в сообщении, можно трактовать в смысле ее новизны или, иначе, уменьшения неопределенности знаний «приемника информации» об объекте.

Американский инженер Р. Хартли в 1928 г. рассматривал процесс получения информации как выбор одного сообщения из конечного заданного множества из N равновероятных сообщений, а количество информации I, содержащееся в выбранном сообщений, определял как двоичный логарифм N: $\mathbf{I = \log_2 N.}$

Допустим, нужно угадать одно число из набора чисел от единицы до ста. По формуле Хартли можно вычислить, какое количество информации для этого требуется: $\mathbf{I = \log_2 100 \approx 6{,}644.}$ Таким образом, сообщение о верно угаданном числе содержит количество информации, приблизительно равное 6,644 единицы информации.

Другие примеры равновероятных сообщений: при бросании монеты — «выпала решка», «выпал орел»; на странице книги — «количество букв четное», «количество букв нечетное».

Определим теперь, являются ли равновероятными сообщения «первой выйдет из дверей здания женщина» и «первым выйдет из дверей здания мужчина». Однозначно ответить на этот вопрос нельзя. Все зависит от того, о каком именно здании идет речь. Если это, например, станция метро, то вероятность выйти из дверей первым одинакова для мужчины и женщины, а если это военная казарма, то для мужчины эта вероятность значительно выше, чем для женщины.

Для задач такого рода американский ученый К. Шеннон предложил в 1948 г. другую формулу определения количества информации, учитывающую возможную неодинаковую вероятность сообщений в наборе.

Формула Шеннона:

$$I = -(p_1\log_2 p_1 + p_2\log_2 p_2 + \ldots + p_N\log_2 p_N) = -\sum_{i=1}^{N} p_i\log_2 p_i ,$$

где p_i — вероятность того, что именно i-е сообщение выделено в наборе из N сообщений.

Очевидно, что если вероятности p_1, …, p_N равны, то каждая из них равна $\frac{1}{N}$, и формула Шеннона превращается в формулу Хартли.

Помимо двух рассмотренных подходов к определению количества информации существуют и другие. Важно помнить, что любые теоретические результаты применимы лишь к определенному кругу случаев, очерченному первоначальными допущениями.

3. Типы и структуры данных

Типы данных (табл. 7). Классификация информационных единиц, обрабатываемых на ЭВМ, включает следующие аспекты:

- типы данных, или совокупность соглашений о программно-аппаратурной форме представления и обработки, а также ввода, контроля и вывода элементарных данных;
- структуры данных — способы композиции простых данных в агрегаты и операции над ними;
- форматы файлов — представление информации на уровне взаимодействия операционной системы с прикладными программами.

Ранние языки программирования (ЯП), а точнее — системы программирования (СП) — Фортран, Алгол, будучи ориентированы исключительно на вычисления, не содержали развитых систем типов и структур данных.

В ЯП Алгол символьные величины и переменные вообще не предусматривались, в некоторых реализациях строки (символы в апострофах) могли встречаться только в операторах печати данных.

Типы числовых данных Алгола: INTEGER (целое число), REAL (действительное) — различаются диапазонами изменения, внутренними представлениями и применяемыми командами процессора ЭВМ (соответственно арифметика с фиксированной и плавающей точкой). Нечисловые данные представлены типом BOOLEAN — логические, имеющие диапазон значений {true, false}.

Позже появившиеся ЯП (СП) Кобол, PL/1, Паскаль вводят новые типы данных:

- символьные (цифры, буквы, знаки препинания и пр.);
- числовые символьные для вывода;
- числовые двоичные для вычислений;
- числовые десятичные (цифры 0–9) для вывода и вычислений.

Разновидности числовых данных здесь соответствуют внутреннему представлению и машинным (или эмулируемым) командам обработки. Кроме того, вводятся числа двойного формата (два машинных слова), для обработки которых также необходимо наличие в процессоре (или эмуляция) команд обработки чисел двойной длины (точности).

Уместно привести пример представления числовой информации в различных перечисленных формах. Пусть задано число $135_{10} = 207_8 = 87_{16} = 100000111_2$, тогда:

- внутренняя стандартная форма представления (тип BINARY для обработки в двоичной арифметике) — сохраняется (100000111_2). Объем — 1 байт, или 8 двоичных разрядов;
- внутренняя форма двоично-десятичного представления (тип DECIMAL, каждый разряд десятичного числа представляется двоично-десятичной, в 4 бита, комбинацией). Представление 135 есть $001\ 011\ 101_2$. Объем — 2,5 байта, 12 двоичных разрядов;
- символьное представление (тип ALPHABETIC, для вывода) — каждый разряд представляется байтом в соответствии с кодом ASCII. Представление 135 есть $00110001\ 00110011\ 00110101_2$. Объем — 3 байта.

Появление систем управления базами данных и систем программирования для разработки ИС приводит к появлению других типов данных:

- дата и время;
- бинарные (BLOB — Binary Large Object) и текстовые объекты без внутренней структуры (интерпретация возлагается на прикладные программы).

Понятие типа данных ассоциируется также с допустимыми значениями переменной и операциями над ними, например данные типа время (ЧЧ:ММ:СС) или дата (ГГ/ММ/ДД) предполагают определенные диапазоны значений каждого из разрядов, а также машинные или эмулируемые операции.

Таблица 7. Типы и структуры данных в некоторых системах программирования

Данные		Система — язык программирования, СУБД, ИПС						
		Алгол	Кобол	PL/1	Fox-Base/ Clipper	Adabas/ Natural	Oracle/ SQL	STAIRS IRBIS, ISIS
Тип данных	Целое короткое (2 байта)	—	—	—	—	—	Small-int	—
	Целое норм. (4 байта)	Inte-ger	Compu-tational	Int	N(x)	N(x)	Int	—
	Целое длинное (8 байт)	—	—	Double	—	—	—	—
	Действительное норм. (4 байт)	Real	Compu-tational	Float	N(x.y)	N(x.y)	Float Real	—
	Действительное двойное (8 байт)	—	—	—	—	—	Float Double	—
	Двоичное	—	—	Binary	—	B(x)	—	—
	Десятичное упакованное (2 цифры на байт)	—	PIC(9)	Deci-mal	—	P(x)	—	—
	Десятичное распакованное (1 цифра на байт)	—	PIC(X)	—	N(x)	U(x)	—	—
	Логическое	Bool-ean	—	+	Logical	—	—	—
	Символьное	—	PIC(A)	Char	C(x)	A(x)	Char	+
	Длинный текстовый или бинарный объект (BLOB)	—	—	—	Memo	—	Var-Grafic Var-Char	—
	Дата	—	—	—	Date	—	Date	—
	Время	—	—	—	—	—	Time	—
Структуры данных	Массивы	Array	—	Dim	Dimen-tion	VAR(n)	—	—
	Записи (структуры)	—	+	+	+	+	+	—
	Множественные (векторные) поля записи	—	—	—	—	MU	—	+
	Групповые поля записи	—	+	+	—	GR	—	+
	Повторяющиеся группы в записи	—	—	—	—	PE	—	—
	Текстовые поля (параграфы, предложения, слова)	—	—	—	—	—	—	+

Структуры данных. В Алголе были определены два типа структур: элементарные данные и массивы (векторы, матрицы, тензоры, состоящие из арифметических или логических переменных). Основным нововведением, появившимся первоначально в Коболе (затем PL/1, Паскаль и пр.), являются агрегаты данных (структуры, записи), представляющие собой именованные комплексы переменных разного типа, описывающих некоторый объект или образующих некоторый достаточно сложный документ.

Рассмотренные выше экзотические типы данных (комплексные), очевидно, занимают промежуточное положение между элементарными переменными и массивами (структурами).

Термин «запись» подразумевает наличие множества аналогичных по структуре агрегатов, образующих файл (картотеку), содержащих по совокупности однородных объектов элементы данных поля, среди которых выделяются элементарные и групповые (агрегатные).

Появление СУБД и АИПС приводит к появлению новых разновидностей структур:

- множественных полей данных;
- периодических групповых полей;
- текстовых объектов (документов), имеющих иерархическую структуру (документ, сегмент, предложение, слово).

4. Двоичное кодирование мультимедиа информации

С 1980-х гг. бурно развивается технология обработки на компьютере графической информации. Компьютерная графика широко используется в компьютерном моделировании в научных исследованиях, компьютерных тренажерах, компьютерной анимации, деловой графике, играх и т. д.

В последнее время в связи с резким ростом аппаратных возможностей персональных компьютеров пользователи получили возможность обрабатывать видеоинформацию.

Графическая информация на экране дисплея представляется в виде изображения, которое формируется из точек (пикселей). В современных компьютерах разрешающая способность (количество точек на экране дисплея), а также количество цветов зависят от видеоадаптера и могут меняться программно.

Цветные изображения могут иметь различные режимы: 16 цветов, 256 цветов, 65 536 цветов (high color), 16 777 216 цветов (true color) — табл. 8. Очевидно, что количество бит на точку (пиксель), например режима true color, равно: $I = \log_2 65536 = 16$ бит = 2 байта.

Таблица 8. Характеристики различных стандартов представления графики

Разрешение	16 цветов	256 цветов	65 536 цветов	16 777 216 цветов
640×480	150 Кбайт	300 Кбайт	600 Кбайт	900 Кбайт
800×600	234,4 Кбайт	468,8 Кбайт	937,5 Кбайт	1,4 Мбайт
1024×768	384 Кбайт	768 Кбайт	1,5 Мбайт	2,25 Мбайт
1280x1024	640 Кбайт	1,25 Мбайт	2,5 Мбайт	3,75 Мбайт

Наиболее распространенной разрешающей способностью экрана является разрешение 800 на 600 точек, т. е. 480 000 точек.

Рассчитаем необходимый для режима true color объем видеопамяти:

V = 2 байта × 480 000 = 960 000 байт = 937,5 Кбайт.

Аналогично рассчитывается объем видеопамяти, необходимый для хранения битовой карты изображений при других видеорежимах.

В видеопамяти компьютера хранится битовый план (bit map), являющийся двоичным кодом изображения, отсюда она считывается (не реже 50 раз в секунду) и отображается на экране.

5. Двоичное кодирование звуковой информации

С начала 1990-х гг. персональные компьютеры получают широкие возможности для работы со звуковой информацией. Каждый компьютер, имеющий звуковую плату, может сохранять звук в виде файлов и воспроизводить его. С помощью специальных программных средств (редакторов аудиофайлов) открываются широкие возможности по созданию, редактированию и прослушиванию звуковых файлов. В дальнейшем создаются программы распознавания речи, и появляется возможность голосового управления компьютером.

При двоичном кодировании аналогового звукового сигнала непрерывный сигнал дискретизируется (оцифровывается), т. е. заменяется серией отдельных выборок. Качество двоичного кодирования зависит от двух параметров: количества распознаваемых дискретных уровней сигнала и количества выборок в секунду.

Различные звуковые карты могут обеспечить как 8-, так и 16-битные выборки. При замене непрерывного звукового сигнала дискретным представлением в виде ступенек 8-битные карты позволяют закодировать 256 различных уровней дискретизации звукового сигнала, соответственно, 16-битные — 65 536 уровней. Частота дискретизации аналогового звукового сигнала (количество выборок в секунду) также может принимать различные значения (5,5, 11, 22 и 44 кГц). Таким образом, качество звука в дискретной форме может быть очень плохим (качество радиотрансляции) при 8 битах и 5,5 кГц и весьма высоким (качество аудио-CD) при 16 битах и 44 кГц.

Можно оценить объем моноаудиофайла с длительностью звучания 1 с при среднем качестве звука (16 бит, 22 кГц). Для этого 16 бит на одну выборку необходимо умножить на 22 000 выборок в секунду, что дает в результате 43 Кбайта.

6. Сжатие информации

Объем обрабатываемой и передаваемой информации быстро растет. Это связано с выполнением все более прикладных процессов, появлением новых информационных служб, использованием изображений и звука. Сжатие данных (data compression) — процесс, обеспечивающий уменьшение объема данных. Сжатие позволяет резко уменьшить объем памяти, необходимый для хранения данных, сократить (до приемлемых размеров) время их передачи. Особенно эффективно сжатие изображений. Сжатие данных может осуществляться как программным, так и аппаратным или комбинированным методом. Сжатие текстов связано с более компактным расположением байтов, кодирующих символы. Определенные результаты дает статистическое кодирование, в котором наиболее часто встречающиеся символы получают коды наименьшей длины. Здесь также используется счетчик повторений пробелов. Что же касается звука и изображений, то объем представляющей их информации зависит от выбранного шага квантования и числа разрядов аналого-дискретного преобразования. В принципе, здесь используются те же методы сжатия, что и при обработке текстов. Если сжатие текстов происходит без потери информации, то сжатие звука и изображения почти всегда приводит к ее некоторой потере. Сжатие широко используется при архивировании данных.

Сжатие изображений (images compression) — процесс минимизации данных, определяющих изображение. Минимизация количества информации, представляющей изображение или видеофильм, прежде всего осуществляется при выборе шага квантования и разрядности кодов. При этом, естественно, происходит определенная (допустимая) потеря информации. Затем происходит сжатие изображения, представленного дискретным сигналом.

Сжатие изображения осуществляется в несколько этапов:

- изображение делится на блоки пикселов, каждый из которых подвергается обработке, устраняющей избыточность;
- осуществляется кодирование с переменной длиной кодов, что исключает длинные цепочки нулей и единиц в последовательностях битов;
- дополнительное сжатие движущегося изображения за счет сравнения каждого изображения с предыдущим, чтобы сохранять только изменившуюся его часть.

Допускается потеря той информации, которая в решении поставленной задачи считается несущественной. Например, можно при обработке изображений удалить из аналогового сигнала частоты, которые находятся вне спектра, воспринимаемого глазом человека (до 10 000 цветов, 250 оттенков серого цвета). Нередко допускается игнорирование цвета каждого второго пиксела, либо группа пикселов заменяется одним со средним значением цвета. Осуществляется также групповое кодирование. Его сущность заключается в кодировании групп одинаковых пикселов (например, небо без облаков на картине).

Размер файла сжатого дискретного неподвижного изображения зависит от четырех параметров: площади изображения, квадрата разрешения, числа бит, необходимых для представления пиксела, и коэффициента сжатия. В видеофильме к этому еще добавляется число образующих его неподвижных изображений. Выбор коэффициентов сжатия — компромисс между пропускной способностью системы (скоростью переноса файлов) и качеством восстанавливаемого изображения. Чем выше коэффициент сжатия, тем ниже это качество. При этом следует иметь в виду, что при очень высокой разрешающей способности и большом коэффициенте сжатия можно получить изображение с низкой разрешающей способностью. Поэтому выбор указанных параметров обосновывается технико-экономическим анализом и алгоритмом сжатия. Что касается качества изображения, то оно зависит от конкретной поставленной задачи.

В зависимости от скорости сжатия изображений выполняемые процессы подразделяются на два класса, к первому относится сжатие неподвижных изображений, которое может выполняться в фоновом режиме, с любой возможной скоростью. Второй класс образуют алгоритмы сжатия движущихся изображений, которые должны выполняться в реальном времени по мере получения данных. Существует немало технологий сжатия/восстановления изображений. Наиболее популярная из них предложена Объединенной группой экспертов в области фотографии (JPEG) и позволяет сократить размер графического файла в 10–20 раз. Благодаря специальным процессорам и алгоритмам удается также сжимать видеосюжеты.

7. Кодирование видеоинформации

В связи с большим объемом информации, содержащейся в видеопотоке (до 6 Мбайт/с), для записи информации в ЭВМ обычно применяют сжатое кодирование потока данных на входе с использованием алгоритмов семейства MPEG/JPEG (табл. 9).

Таблица 9. Характеристики представления видеоинформации в различных форматах

№	Формат	Тип данных (размер изображения)	Длительность записи (CD/DVD), мин
1	VCD	288×384	63
2	S-VCD	480×576	32
3	DVD	576×720	59
4	VHS	288×384	—
5	S-VHS	540×720	—
6	Internet High Speed	193×144	—

Стандарт MPEG (Motion Picture Expert Group) включает несколько компонент: системного потока, описывающего структуру смешанного аудио- и видеопотока, а также MPEG-video и MPEG-audio.

В случае MPEG-video сжатие достигается за счет четырех факторов:

1. Использование составляющих YUV вместо обычных RGB (красный, зеленый, синий).

Вместо элементарных цветов кодируются яркость (luminance, Y) и цветность (chrominance, U & V), причем цветность «прорежена» по вертикали и горизонтали в два раза по сравнению с яркостью (децимация). При этом вместо сильно коррелированных сигналов RGB получаются практически некоррелированные YUV, и за счет децимации достигается двукратное сжатие.

2. Дискретно-косинусное преобразование с последующим квантованием.

При этом квадраты пикселей (8×8) подвергаются двухмерному дискретно-косинусному преобразованию (DCT), которое родственно преобразованию Фурье, различие заключается в наборе базисных функций (в преобразовании Фурье это синусы и косинусы, в DCT — косинусы). Это преобразование переводит пространственное представление сигнала в частотное. Результат преобразования подвергается квантованию, т. е. огрублению точности, при этом коэффициент квантования для более высоких пространственных частот выбирается более высоким, чем для низких, с учетом особенностей восприятия. При этом высокие пространственные частоты передаются с меньшей точностью, чем низкие частоты. При квантовании многие пространственные частоты не кодируются и не передаются.

3. Устранение временной избыточности с компенсацией движения.

Это означает, что для ликвидации избыточности, заключающейся в большой корреляции между соседними кадрами, передается разность между ними. Кадры видеопотока разбиваются на несколько типов — Intro (I), которые кодируются полностью, Predicted (P), для которых кодируется различие с предыдущим I- или P-кадром, и Bidirectional (B), для которых в качестве

опорных (reference) используются I- и/или P-кадры, между которыми он находится. Обычно I-кадры следуют 1 или 2 раза в секунду, и между двумя опорными кадрами лежит 2–4 B-кадра. Типичная последовательность кадров имеет вид: IBBPBBPBBPBBIBBP. В общем случае вид последовательности выбирается кодеком (кодирующее устройство или программа) и может зависеть или нет от содержания кадров. Поскольку изображение на соседних кадрах обычно сдвинуто, применяется компенсация движения, т. е. кодируется отклонение («разность») от некоторого сдвинутого опорного изображения. Кодирование выполняется макроблоками (16×16 яркость, 8×8 цветность), для каждого макроблока определяется свой вектор движения.

4. Квазиоптимальное кодирование.

Коэффициенты, полученные после DCT, векторы движения и все остальное кодируются кодами переменной длины. Это кодирование называют квазиоптимальным, поскольку кодовая таблица не строится заново для каждого конкретного случая, а выбрана при разработке стандарта на основе анализа типичных видеопоследовательностей.

MPEG-1 проектировался из расчета на поток 1,5 Мбит/с при 30 кадрах размером 352×240 в секунду, хотя он не ограничен этим и допускает существенно больший поток при произвольном размере кадра.

MPEG-2 проектировался с учетом опыта использования MPEG-1 и ориентируется на вещание, так как содержит средства для маскирования ошибок.

В случае MPEG-audio исходный сигнал подвергается многоканальной фильтрации. Далее амплитуды сигналов в каждой полосе сравниваются для нахождения полос, подлежащих кодированию с эффекта маскирования слабого сигнала сильным. Далее амплитуда сигнала в полосе квантуется и кодируется. При записи на Video CD скорость потока звука составляет 32 Кбайт/с.

Контрольные вопросы:

1. Что такое информация?
2. Как измеряется количество информации?
3. Какие аспекты включает классификация информационных единиц, обрабатываемых на ЭВМ?
4. Что такое кодирование?
5. Как происходит сжатие изображения?

Лекция 5. Логические основы ЭВМ

План:

1. Логические операции и базовые элементы компьютера.
2. Вентили.
3. Свойства операций.

1. Логические операции и базовые элементы компьютера

Начало исследованиям в области формальной логики было положено работами Аристотеля в IV в. до нашей эры. Однако математические подходы к этим вопросам впервые были указаны Дж. Булем. В честь него алгебру высказывания называют булевой алгеброй, а логические значения — булевыми.

Основу математической логики составляет алгебра высказываний. Это освобождает матлогику от неопределенности в толковании логических выражений, показывающих связь между отдельными суждениями и понятиями. Алгебра логики используется при построении основных узлов ЭВМ (дешифратор, сумматор, шифратор). Алгебра логики оперирует с высказываниями. Под высказыванием понимают повествовательное предложение, относительно которого можно утверждать, истинно оно или ложно. Например, выражение «Расстояние от Москвы до Киева больше, чем от Москвы до Тулы» истинно, а выражение «5 < 2» — ложно. Высказывания принято обозначать буквами латинского алфавита: A, B, C, ..., X, Y и т. д. Если высказывание C истинно, то пишут C = 1 (C = t, true), а если оно ложно, то C = 0 (C = f, false).

В алгебре высказываний над высказываниями можно производить определенные логические операции, в результате которых подаются новые высказывания. Истинность полученных высказываний зависит от истинности исходных высказываний и использованных для их преобразования логических операций.

Для образования новых высказываний наиболее часто используются логические операции, выражаемые словами «не», «и», «или».

Логический элемент компьютера — это часть электронной схемы, которая реализует элементарную логическую функцию. Логическими элементами компьютеров являются электронные схемы И, ИЛИ, НЕ, И — НЕ, ИЛИ — НЕ и другие (называемые обычно вентилями), а также триггер.

Может быть доказано, что с помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у вентилей бывает от двух до восьми входов и один или два выхода.

На структурных схемах ЭВМ каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая именно электронная схема в нем реализована. Работу логических элементов описывают с помощью таблиц истинности.

Логические операции. Рассмотрим логические операции и соответствующие им элементы логических схем.

Конъюнкция. Соединение двух (или нескольких) высказываний в одно с помощью союза И (AND) называется операцией логического умножения, или конъюнкцией. Эту операцию принято обозначать знаками ^, & или знаком умножения ×. Сложное высказывание A & B истинно только в том случае, когда истинны оба входящих в него высказывания (табл. 10)

Таблица 10. Таблица истинности конъюнкции

A	B	A & B
false	false	false
false	true	false
true	false	false
true	true	true

Логическая схема И реализует конъюнкцию двух или более логических значений. Условное обозначение на структурных диаграммах схемы И с двумя входами представлено на рис. 2*а*.

Единица на выходе схемы И будет тогда и только тогда, когда на всех входах будут единицы. Когда хотя бы на одном входе будет нуль, на выходе также будет нуль. Связь между выходом z этой схемы и входами x и y описывается соотношением: z = x & y (читается как «x И y»). Операция конъюнкции на структурных схемах обозначается знаком &.

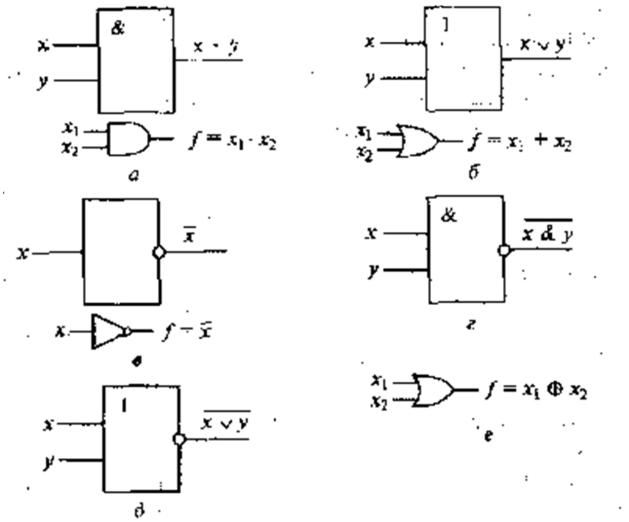


Рисунок 2. Схемные логические элементы вычислительных машин

Дизъюнкция. Объединение двух (или нескольких) высказываний с помощью союза ИЛИ (OR) называется операцией логического сложения, или дизъюнкцией. Эту операцию обозначают знаками |, ∨ или знаком сложения +. Сложное высказывание A ∨ B истинно, если истинно хотя бы одно из входящих в него высказываний (табл. 11).

Таблица 11. Таблица истинности для логической суммы высказываний

A	B	A ∨ B	A XOR B
false	false	false	false
false	true	true	true
true	false	true	true
true	true	true	false

В последнем столбце табл. 11 размещены результаты модифицированной операции ИЛИ — ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR). Отличается от обычного ИЛИ последней строкой (рис. 2*в*, *г*).

Схема ИЛИ реализует дизъюнкцию двух или более логических значений. Когда хотя бы на одном входе схемы ИЛИ будет единица, на ее выходе также будет единица. Условное обозначение на структурных схемах схемы ИЛИ с двумя входами представлено на рис. 2*д*. Знак «1» на схеме — от классического обозначения дизъюнкции как «≥1» (т. е. значение дизъюнкции равно единице, если сумма значений операндов больше или равна 1). Связь между выходом z этой схемы и входами x и y описывается соотношением z = x ∨ y (читается как «x ИЛИ y»).

Инверсия. Присоединение частицы НЕ (NOT) к некоторому высказыванию называется операцией отрицания (инверсии) и обозначается $\overline{A}$. Если высказывание А истинно, то В ложно, и наоборот (табл. 12).

Таблица 12. Таблица истинности отрицания

A	$\overline{A}$
false	true
true	false

Схема НЕ (инвертор) реализует операцию отрицания. Связь между входом x этой схемы и выходом z можно записать соотношением $z = \overline{x}$, где $\overline{x}$ читается как «НЕ x» или «ИНВЕРСИЯ x». Если на входе схемы «0», то на выходе «1», и наоборот. Условное обозначение на структурных схемах инвертора — на рис. 2*в*.

2. Вентили

Кроме схемных элементов, соответствующих перечисленным логическим операторам, в состав логических схем входят комбинированные связки, именуемые вентилями.

Схема И — НЕ состоит из элемента И и инвертора и осуществляет отрицание результата схемы И (табл. 13). Связь между выходом z и входами x и y схемы записывают как x & y, или «ИНВЕРСИЯ x И y». Условное обозначение на структурных схемах схемы И — НЕ с двумя входами представлено на рис. 2*г*.

Таблица 13 — Таблица истинности схемы И — НЕ

X	**Y**	**X & Y**
false	false	true
false	true	true
true	false	true
true	true	false

Схема ИЛИ — НЕ состоит из элемента ИЛИ и инвертора и осуществляет отрицание результата схемы ИЛИ (табл. 14). Связь между выходом z и входами x и y схемы записывают как $x \vee y$, или «ИНВЕРСИЯ x ИЛИ y». Условное обозначение на структурных схемах схемы ИЛИ — НЕ с двумя входами представлено на рис. 2*д*.

Таблица 14. Таблица истинности схемы ИЛИ — НЕ

X	**Y**	**X ∨ Y**
false	false	true
false	true	false
true	false	false
true	true	false

Схема «ИСКЛЮЧАЮЩЕЕ ИЛИ» (рис. 2*е*) соответствует «сложению по модулю два». Кроме операций И, ИЛИ, НЕ в алгебре высказываний существует ряд других операций. Например, операция эквиваленции (эквивалентности) A ~ B (или A = B, A EQV B) (табл. 15).

Таблица 15. Таблица истинности операции эквивалентности

A	B	A ~ B
false	false	true
false	true	false
true	false	false
true	true	true

Другим примером может служить логическая операция импликации или логического следования (A → B, A IMP B), иначе говоря, «ЕСЛИ A, то B» (табл. 16).

Таблица 16. Таблица истинности импликации

A	B	A → B
false	false	true
false	true	true
true	false	false
true	true	true

Высказывания, образованные с помощью логических операций, называются сложными. Истинность сложных высказываний можно установить, используя таблицы истинности. Например, истинность сложного высказывания A & B определяется по табл. 17.

Таблица 17. Таблица истинности высказывания A & B

A	B	$\overline{A}$	B	$\overline{A}$ & B
false	false	true	true	true
false	true	true	false	false
true	false	false	true	false
true	true	false	false	false

Высказывания, у которых таблицы истинности совпадают, называются равносильными. Для обозначения равносильных высказываний используют знак = (A = B). Рассмотрим сложное высказывание (A & B)|(B & B) — табл. 18.

Таблица 18. Таблица истинности выражения (A & B)|(B & B)

A	$\overline{A}$	B	$\overline{B}$	A & B	B & B	(A & B)|(B & B)
false	false	true	true	false	true	true
false	true	true	false	false	false	false
true	false	false	true	false	false	false
true	true	false	false	true	false	true

Если сравнить эту таблицу с таблицей истинности операции эквивалентности высказываний A и B, то можно увидеть, что высказывания (A & B)|(B & B) и A ~ B тождественны, т. е. A ~ B = (A & B)|(B & B)

В алгебре высказываний можно проводить тождественные преобразования, заменяя одни высказывания равносильными им другими высказываниями.

3. Свойства операций

Исходя из определений дизъюнкции, конъюнкции и отрицания, устанавливаются свойства этих операций и взаимные распределительные свойства.

Коммутативность (переставновочность):

$A \wedge B = B \wedge A$, $A \vee B = B \vee A$

Закон идемпотентности:

$A \,\&\, A = A$, $A \vee A = A$.

Двойное отрицание: $\overline{\overline{A}} = A$.

Сочетательные (ассоциативные) законы:

$A \vee (B \vee C) = (A \vee B) \vee C = A \vee B \vee C$, $A \wedge (B \wedge C) = (A \wedge B) \wedge C = A \wedge B \wedge C$.

Распределительные (дистрибутивные) законы:

$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$, $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$.

Поглощение:

$x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$.

Склеивание:

$(x \wedge y) \vee (\overline{x} \wedge y) = y$, $(x \vee \overline{x}) \wedge (y \vee y) = y$.

Операция переменной с ее инверсией:

$x \vee \overline{x} = 1$, $x \wedge \overline{x} = 0$.

Операция с константами:

$x \vee 0 = x$, $x \vee 1 = 1$, $x \wedge 1 = x$, $x \wedge 0 = 0$.

Законы Де Моргана:

1) $\overline{A \,\&\, B} = \overline{A} \vee \overline{B}$;

2) $\overline{A \vee B} = \overline{A} \,\&\, \overline{B}$.

Высказывания, образованные с помощью нескольких операций логического сложения, умножения и отрицания, называются сложными. Истинность всякого сложного высказывания устанавливается с помощью таблиц истинности. Сложные высказывания, истинные для любых значений истинности входящих в них простых высказываний, называются тождественно-истинными. Наоборот, тождественно-ложными являются формулы, принимающие значение (false) для любых значений входящих в него простых высказываний. На рис. 3*а–ж* приведены иллюстрации к основным логическим операциям и их композициям (так называемые диаграммы Эйлера — Венна).

Логическое значение null. В некоторых языках программирования (Visual Basic и пр.) для расширения применимости логических выражений на те случаи, когда значения одного или нескольких логических аргументов неизвестны или не определены, вводится значение null (в дополнение к false и true), как правило, такое значение присваивается компилятором логической переменной по умолчанию. С учетом значения null таблицы истинности основных логических операций приобретают следующий вид (табл. 19, 20).

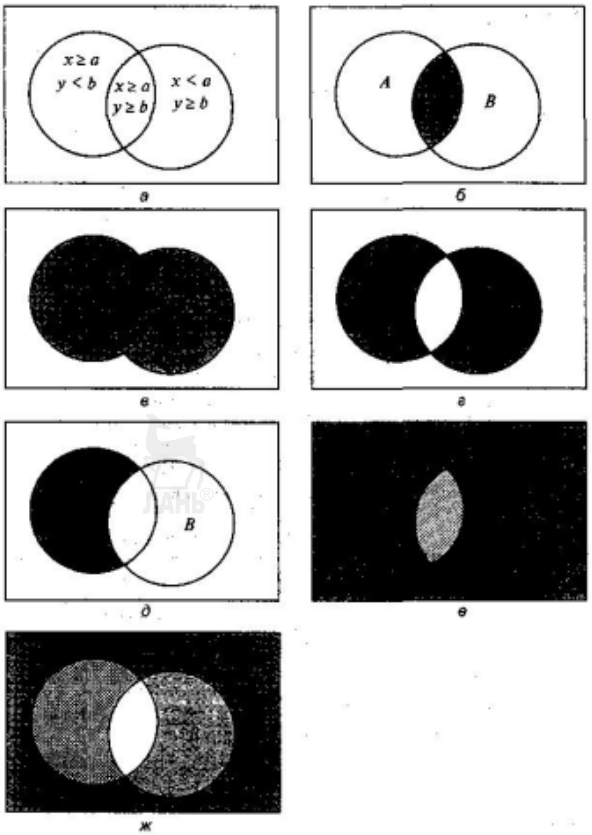


Рисунок 3. Некоторые примеры диаграмм Эйлера — Венна: а — диаграмма Эйлера — Венна, иллюстрирующая расположение областей истинности высказываний A и B; б — конъюнкция высказываний A и B (AND); в — дизъюнкция высказываний A и B (OR); г — исключающая дизъюнкция (XOR); д — разность высказываний (A – B); е — иллюстрация к законам де Моргана (дополнение пересечению высказываний); ж — иллюстрация к законам де Моргана (объединение дополнений).

Таблица 19. Одноместная (унарная) операция отрицания с учетом значения null

A	$\overline{A}$
false	true
true	false
null	null

Таблица 20. Некоторые двухместные (бинарные) операции с учетом значения null

A	**B**	**A & B**	**A ∨ B**	**A→ B**	**A ~ B**	**A XOR B**
false	false	false	false	true	true	false
false	true	false	true	true	false	true
true	false	false	true	false	false	true
true	true	true	true	true	true	false

Продолжение табл. 20

A	B	A & B	A ∨ B	A→ B	A ~ B	A XOR B
false	null	false	false	true	null	null
true	null	null	true	null	null	null
null	false	false	null	null	null	null
null	true	null	true	true	null	null
null	null	null	null	null	null	null

Побитовые операции. В некоторых современных ЯП включены операции побитового сравнения содержимого машинных слов (которые могут содержать числовые, строчные и др. данные), при этом каждый бит результата образуется в соответствии с табл. 21. Унарная операция отрицания (NOT) в данном случае реализует очевидную замену «l» на «0» и наоборот.

Таблица 21. Операнды и результаты некоторых операций побитового сравнения

x	y	x & y	x ∨ y	x IMP y	x EQV y	x XOR y
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	0	1
1	1	1	1	1	1	0

Контрольные вопросы:

1. Назовите основные логические операции и приведите их таблицы истинности.
2. Что такое логическое выражение?
3. Каков порядок выполнения операций при вычислении значения логического выражения?

Лекция 6. Логические элементы и узлы ЭВМ

План

1. Структурные единицы ЭВМ.
2. Типовые функциональные узлы.
3. Цифровые автоматы (триггеры, регистры, счетчики).
4. Узлы ЭВМ.

1. Структурные единицы ЭВМ

В ЭВМ различают структурные единицы:

1. Элементы — обработка одной единицы электронных сигналов (битов информации).
2. Узлы — одновременная обработка группы электронных сигналов (информационных слов).
3. Блоки — некоторая последовательность в обработке информационных слов.
4. Устройства — выполнение отдельных машинных операций и их последовательности.

2. Типовые функциональные узлы

Типовые функциональные узлы ЭВМ:

– мультиплексор — комбинационное логическое устройство, предназначенное для управляемой передачи данных от нескольких источников информации в один выход;

– демультиплексор — комбинационное логическое устройство, предназначенное для управляемой передачи данных от одного источника информации в несколько выходных каналов;

– шифратор и дешифратор — преобразование из десятичной системы счисления в двоичную и наоборот;

– цифровой компаратор — комбинационное логическое устройство для сравнения чисел в двоичном коде;

– АЛУ — арифметико-логическое устройство;

– сумматор.

3. Цифровые автоматы (триггеры, регистры, счетчики)

Триггер. Данное устройство — это электронная схема, широко применяемая в регистрах компьютера для запоминания одного разряда двоичного кода. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое — двоичному нулю.

Термин триггер происходит от английского слова trigger — защелка, спусковой крючок. Для обозначения этой схемы в английском языке чаще употребляется термин flip-flop, что в переводе означает «хлопанье». Это звукоподражательное название электронной схемы указывает на ее способность мгновенно переходить («перебрасываться») из одного состояния в другое и обратно.

Самый распространенный тип триггера — так называемый RS-триггер (S и R соответственно от английских et — установка, и reset — сброс). Условное обозначение RS-триггера приводится на рис. 5*а*. Он имеет два симметричных входа S и R и два симметричных выхода Q и $\overline{Q}$, причем выходной сигнал Q является логическим отрицанием сигнала $\overline{Q}$. На каждый из двух входов S и R могут подаваться входные сигналы в виде кратковременных импульсов (⎍). Наличие импульса на входе считается единицей, а его отсутствие — нулем.

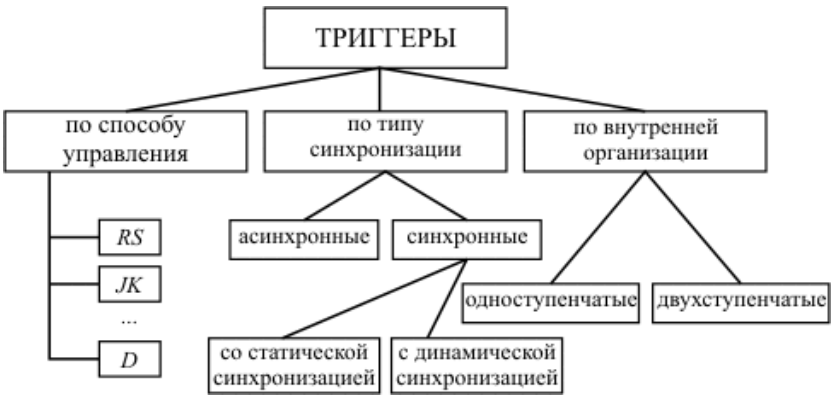


Рисунок 4. Классификация триггеров

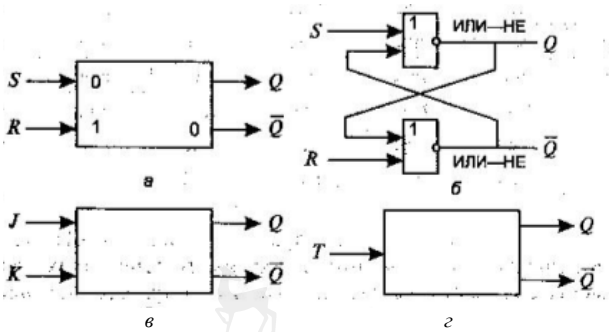


Рисунок 5. Варианты триггерных цепей:
а — RS-триггер; б — его реализация; в — JK-триггер; г — Т-триггер.

Таблица 22. Таблица истинности для RS-триггера

S	R	Q	$\bar{Q}$
0	0	Без изменений	
0	1	1	0
1	0	0	1
1	1	Не определено (запрещено)	

Поскольку триггер может запомнить только один разряд двоичного кода, для запоминания байта нужно 8 триггеров, для запоминания килобайта соответственно $8 \times 2^{10} = 8192$ триггера. Современные микросхемы памяти содержат миллионы триггеров.

Кроме RS-триггеров известны также JK- и Т-триггеры. JK-триггер содержит схемные дополнения, которые снимают неопределенность состояния при подаче ⎍⎍ на оба входа. Теперь при этом происходит «переброс» схемы в противоположное состояние (Q и $\bar{Q}$ меняются местами — «0» переходит в «1» и наоборот).

Таблица 23. Таблица истинности для JK-триггера

J	K	Q	$\bar{Q}$
0	0	Без изменений	
0	1	1	0
1	0	0	1
1	1	Переброс состояния	

Т-триггер имеет единственный вход (Т), при подаче ⎍⎍ на который осуществляется «переброс» схемы (табл. 24).

Таблица 24. Иллюстрация к действию Т-триггера

T	Q	$\bar{Q}$
0	Без изменений	
1	Переброс состояния	

Т-триггеры могут использоваться для создания двоичных счетчиков (например, счетчик адреса команд, обеспечивающий последовательную выборку слов из оперативной памяти).

Полусумматор. Вспомним, что при сложении двоичных чисел образуется сумма в данном разряде, при этом возможен перенос в старший разряд. Обозначим слагаемые (A, B), перенос (P), сумму (S) и рассмотрим соответствующую данной операции табл. 25.

Таблица 25. Таблица сложения одноразрядных двоичных чисел с учетом переноса в старший разряд

Слагаемые		**Перенос**	**Сумма**
A	B	P	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Из этой таблицы очевидно, что перенос можно реализовать с помощью операции логического умножения P = A & B.

Получим теперь формулу для вычисления суммы. Значения суммы более всего совпадают с результатом операции логического сложения (кроме случая, когда на вход подаются две единицы, а на выходе должен получиться нуль).

Нужный результат достигается, если результат логического сложения умножить на инвертированный перенос. Таким образом, для определения суммы можно применить выражение $S = A \vee B \,\&\, A \,\&\, B$. Теперь, на основе полученных логических выражений, можно построить из базовых логических элементов схему полусумматора.

Из логической формулы для суммы очевидно, что на выходе должен стоять элемент логического умножения И, который имеет два входа. На один из входов подается результат логического сложения исходных величин, т. е. на него должен подаваться сигнал с элемента логического сложения ИЛИ.

На второй вход требуется подать результат инвертированного логического умножения исходных сигналов A & B, т. е. на второй вход подается сигнал с элемента НЕ, на вход которого поступает сигнал с элемента логического умножения И.

Данная схема называется полусумматором, так как реализует суммирование одноразрядных двоичных чисел без учета переноса из младшего разряда.

Полный одноразрядный сумматор. Полный одноразрядный сумматор должен иметь три входа, a_i, b_i — слагаемые и p_{i-1} — перенос из предыдущего разряда и два выхода, сумма S_i и перенос P_i. Порядок функционирования схемы иллюстрирует табл. 26.

Таблица 26. Таблица сложения для полного одноразрядного сумматора

a_i	b_i	p_{i-1}	P_i	S_i
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

Идея построения полного сумматора точно такая же, как и полусумматора. Перенос реализуется с помощью формулы для его получения

$P_i = (a_i \,\&\, b_i) \vee (a_i \,\&\, p_{i-1}) \vee (b_i \,\&\, p_{i-1})$.

Логическое выражение для вычисления суммы в полном сумматоре принимает следующий вид:

$S_i = (a_i \vee b_i \vee p_{i-1}) \,\&\, p_{i-1} \vee (a_i \,\&\, b_i \,\&\, p_{i-1})$.

Укрупненная схема, соответствующая полному сумматору, приведена на рис. 6.

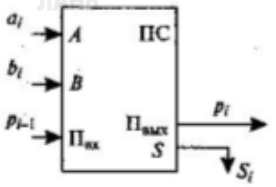


Рисунок 6. Укрупненная схема одноразрядного сумматора

3. Узлы ЭВМ

Узлами ЭВМ являются стандартизованные наборы логических элементов, из которых, как из «кирпичиков», набираются схемы входящих в состав микропроцессоров, блоков памяти, контроллеров внешних устройств и пр.

Узлы ЭВМ разделяются на:

– комбинационные, или узлы, выходные сигналы которых определяются только сигналом на входе, действующим в настоящий момент времени (например, дешифратор). Выходной сигнал дешифратора зависит только от двоичного кода, поданного на вход в настоящий момент времени. Комбинационные узлы называют также автоматами без памяти;

– последовательностные (автоматы с памятью) — это узлы, выходной сигнал которых зависит не только от комбинации входных сигналов, действующих в настоящий момент времени, но и от предыдущего состояния узла (счетчик);

– программируемые узлы функционируют в зависимости от того, какая программа в них записана. Например, программируемая логическая матрица (ПЛМ), которая в зависимости от встроенной («прожженной») в ней программы может выполнять функции сумматора, дешифратора, ПЗУ.

Сумматоры. Многоразрядный сумматор процессора состоит из полных одноразрядных сумматоров (рис. 6). На каждый ставится одноразрядный сумматор, причем выход (перенос) сумматора младшего разряда подключен к входу сумматора старшего разряда.

Например, схема вычисления суммы $S = (s_3\ s_2\ s_1\ s_0)$ двух двоичных трехразрядных чисел $A = (a_2\ a_1\ a_0)$ и $B = (b_2\ b_1\ b_0)$ может иметь вид, приведенный на рис. 7.

Сумматор может быть построен в двух вариантах:

- комбинационная схема (последовательный сумматор);
- последовательностная схема (накапливающий сумматор).

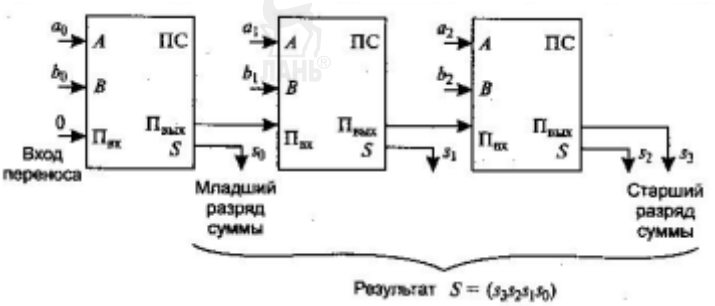


Рисунок 7. Фрагмент (3 разряда) схемы многоразрядного сумматора

Последовательный сумматор (рис. 8) осуществляет суммирование слагаемых и цифр переноса поразрядно начиная с младшего разряда. Основой его схемы является одноразрядный сумматор. Суммирование производится в одноразрядном сумматоре SM.

Цифры i-го разряда слагаемого и цифра переноса из младшего разряда передаются на вход сумматора одновременно с приходом тактового импульса. Регистры 1 и 2 используются для приема и хранения цифр i-го разряда слагаемых. В триггере ТТ хранится цифра переноса из младшего разряда. Регистр 3 принимает и хранит i-ю цифру суммы.

С приходом тактового импульса из регистров 1, 2 и триггера ТТ разряды слагаемых и цифра переноса поступают на одноразрядный сумматор. Одновременно регистр 3 освобождается для приема цифры суммы.

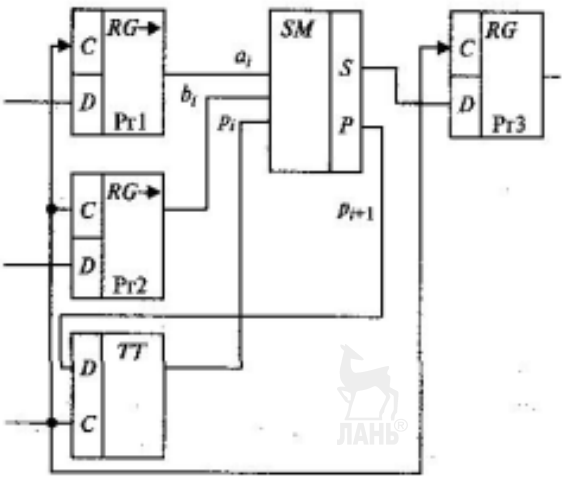


Рисунок 8. Последовательный сумматор

В параллельном сумматоре все разряды операндов суммируются одновременно, но быстродействие снижается за счет времени передачи цифры переноса из младшего разряда (рис. 9*а*).

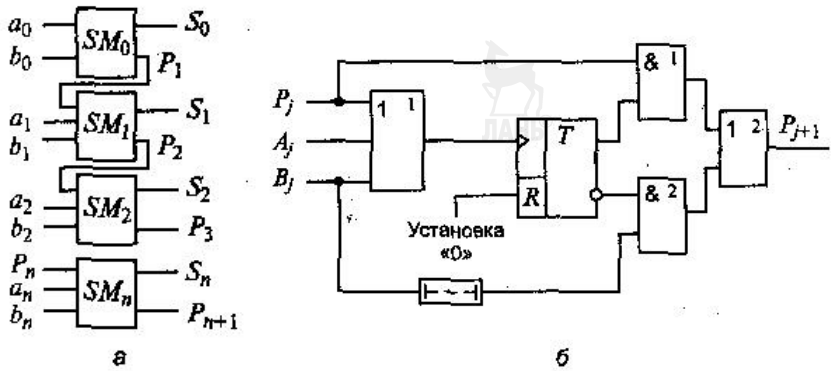


Рисунок 9. Схемы параллельного (а) и накапливающего (б) сумматоров

Накапливающий сумматор является автоматом с памятью, т. е. слагаемые могут приходить поочередно в произвольные моменты времени и запоминаться в линиях задержки или триггерах, запоминающий сумматор применяется в асинхронных устройствах, в которых слагаемые не привязаны к тактам тактового генератора (рис. 9*б*).

С приходом слагаемого $a_i = 1$ элемент ИЛИ и триггер устанавливаются в «1». Если $b_i = 1$ и приходит через какое-то время после a_i, то оно запоминается в линии задержки и одновременно b_i «опрокидывает» триггер в «0». На инверсном выходе триггера устанавливается «1», следовательно, на вторую схему И подаются две единицы, следовательно, на выходе второй схемы ИЛИ формируется цифра переноса в старший разряд, равная «1». Если $P_i = 0$, то цифра суммы, которая снимается с прямого выхода триггера, равна «0». Если $P_i = 1$, то сумма $S_i = 1$.

Дешифраторы. Схемы предназначаются для преобразования двоичного кода (X) на входе в управляющий сигнал (Z) на одном из выходов. Если входов n, то выходных шин должно быть $N = 2^3$ (табл. 27, n = 3, N = 8).

Дешифраторы могут быть линейными и многокаскадными. У линейных дешифраторов все переменные X_1, X_2, X_3 подаются на вход одновременно (рис. 10*а*).

Таблица 27. Пример таблицы состояний дешифратора

X_1	X_2	X_3	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Они обладают более высоким быстродействием, но более трех переменных одновременно подать нельзя, поэтому чаше применяются многокаскадные дешифраторы, у которых количество элементов в каждом следующем разряде больше, чем в предыдущем. На вход первого каскада подается один слог, на вход следующего каскада — второй слог и результаты конъюнкций, проведенных в первом каскаде.

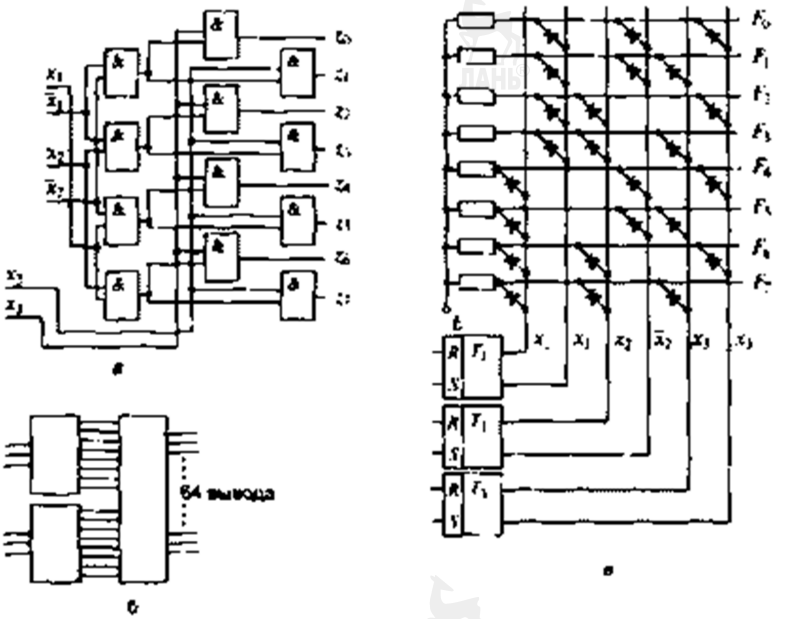


Рисунок 10. Линейный дешифратор (а), диодная матрица (б), многокаскадный дешифратор (в)

Простейший линейный дешифратор можно построить на диодной матрице (рис. 10*б*). В этой схеме используется отрицательная логика. При подаче «1» на анод (коллектор) диода он закрывается. Если закрыты все три диода, подсоединенные к одной горизонтальной линии, то на этой линии появляется потенциал –Е, соответствующий уровню «1».

Многокаскадный дешифратор можно организовать так, как это изображено на рис. 10*в*. Два линейных дешифратора обрабатывают по два слова. В последнем каскаде образуются конъюнкции выходного сигнала первого каскада. Многокаскадные дешифраторы дают меньшим быстродействием.

Контрольные вопросы:

1. Какие структурные единицы различают в ЭВМ?
2. Перечислите функциональные узлы ЭВМ.
3. Что такое триггер?
4. Что такое узел ЭВМ? Перечислите их.

Тема 2. Базовые понятия и основные принципы построения архитектур вычислительных систем

Лекция 7. Базовые представления об архитектуре ЭВМ

План:

1. Базовые представления об архитектуре ЭВМ.
2. Принципы (архитектура) фон Неймана.
3. Логические узлы (агрегаты) ЭВМ, простейшие типы архитектур.

1. Базовые представления об архитектуре ЭВМ

Определим основные объекты рассмотрения:

– обычные вычислительные машины;

– вычислительные комплексы (системы), в том числе многопроцессорные машины;

– суперкомпьютеры;

– вычислительные сети.

Три последних относятся к вычислительным системам.

Архитектурой компьютера считается его представление на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т. д. Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера: *процессора, оперативного запоминающего устройства (ОЗУ, ОП), внешних ЗУ и периферийных устройств.*

Общность архитектуры разных компьютеров обеспечивает их совместимость с точки зрения пользователя.

Структура компьютера — это совокупность его функциональных элементов и связей между ними. Элементами могут быть самые различные устройства — от основных логических узлов компьютера до простейших схем. Структура компьютера графически представляется в виде структурных схем, с помощью которых можно дать описание компьютера на любом уровне детализации.

2. Принципы (архитектура) фон Неймана

В основу построения большинства компьютеров положены следующие общие принципы, сформулированные в 1945 г. американским ученым Джоном фон Нейманом.

1. Принцип программного управления. Из него следует, что программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.

Выборка программы из памяти осуществляется с помощью *счетчика команд.* Этот регистр процессора последовательно увеличивает хранимый в нем адрес очередной команды на длину команды. Так как команды программы расположены в памяти друг за другом, то тем самым организуется выборка цепочки команд из последовательно расположенных ячеек памяти.

Если после выполнения команды следует перейти не к следующей, а к какой-то другой, используются команды *условного* или *безусловного переходов* (*ветвления*), которые заносят в счетчик команд номер ячейки памяти, содержа-

щей следующую команду. Выборка команд из памяти прекращается после достижения и выполнения команды «стоп».

Таким образом, процессор исполняет программу автоматически, без вмешательства человека.

2. Принцип однородности памяти. Программы и данные хранятся в одной и той же памяти. Поэтому компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда. Над командами можно выполнять такие же действия, как и над данными. Это открывает целый ряд возможностей. Например, программа в процессе своего выполнения также может подвергаться переработке, что позволяет задавать в самой программе правила получения некоторых ее частей (так в программе организуется выполнение циклов и подпрограмм). Более того, команды одной программы могут быть получены как результаты исполнения другой программы. На этом принципе основаны методы *трансляции* — перевода текста программы с языка программирования высокого уровня на язык конкретной машины.

3. Принцип адресности. Структурно основная память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти, так чтобы к запомненным в них значениям можно было впоследствии обращаться или менять их в процессе выполнения программ с использованием присвоенных имен.

Компьютеры, построенные на этих принципах, относятся к типу фон-неймановских. Существуют и другие классы компьютеров, принципиально отличающиеся от фон-неймановских. Здесь, например, может не выполняться принцип программного управления, т. е. они могут работать без *счетчика (регистра адреса) команд,* указывающего на выполняемую команду программы. Для обращения к какой-либо переменной, хранящейся в памяти, этим компьютерам не обязательно давать ей имя. Такие компьютеры называются не-фон-неймановскими.

3. Логические узлы (агрегаты) ЭВМ, простейшие типы архитектур

Центральное устройство. ЦУ представляет основную компоненту ЭВМ и, в свою очередь, включает ЦП — центральный процессор (central processing unit — CPU) и ОП — оперативную (главную) память (main storage, core storage, random access memory — RAM).

Процессор непосредственно реализует операции обработки информации и управления вычислительным процессом, осуществляя выборку машинных команд и данных из оперативной памяти и запись в ОП, включение и отключение ВУ. Основными блоками процессора являются:

– устройство управления (УУ) с интерфейсом процессора (системой сопряжения и связи процессора с другими узлами машины);

– арифметико-логическое устройство (АЛУ);

– процессорная память (внутренний кэш).

Оперативная память предназначена для временного хранения данных и программ в процессе выполнения вычислительных и логических операций.

ЦУ описывается следующими характеристиками:

– длина машинного слова (разрядность, адресность);

− система команд;
− объем ОП;
− быстродействие (тактовая частота процессора, цикл записи/считывания ОП).

Внешние устройства (ВУ). ВУ обеспечивают эффективное взаимодействие компьютера с окружающей средой — пользователями, объектами управления, другими машинами. ВУ разделяются на следующие группы:
− интерактивные устройства (ввода-вывода);
− устройства хранения (массовые накопители);
− устройства массового ввода информации;
− устройства массового вывода информации.

В *специализированных* управляющих ЭВМ (технологические процессы, связь, ракеты и пр.) внешними устройствами ввода являются датчики (температуры, давления, расстояния и пр.), вывода — манипуляторы (гидро-, пневмо-, сервоприводы рулей, вентилей и др.).

В *универсальных* ЭВМ (человеко-машинная обработка информации) в качестве ВУ выступают терминалы, принтеры и др. устройства.

Каналы связи (внутримашинный интерфейс) служат для сопряжения центральных узлов машины с ее внешними устройствами.

Однотипные ЦУ и устройства хранения данных могут использоваться в различных типах машин. Известны примеры того, как фирмы, начавшие свою деятельность с производства управляющих машин, совершенствуя свою продукцию, перешли к выпуску систем, которые в зависимости от конфигурации ВУ могут исполнять роль как универсальных, так и управляющих машин (Hewlett-Packard и Digital Equipment Corporation).

Если абстрагироваться от подробностей, то основные классические типы архитектур — это *«звезда», иерархическая, магистральная* (схематически — рис. 11, подробнее — рис. 12, 13, 14).

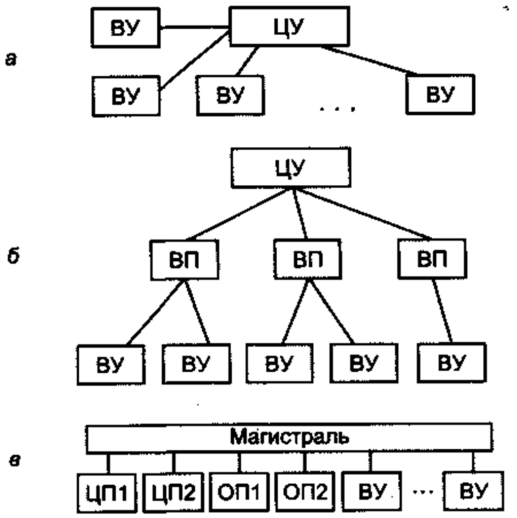


Рисунок 11. Основные классы архитектур ЭВМ

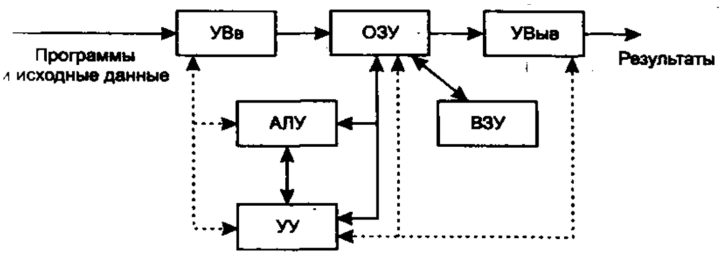


Рисунок 12. Структурная схема ЭВМ 1-го и 2-го поколения (архитектура фон Неймана), «звезда»

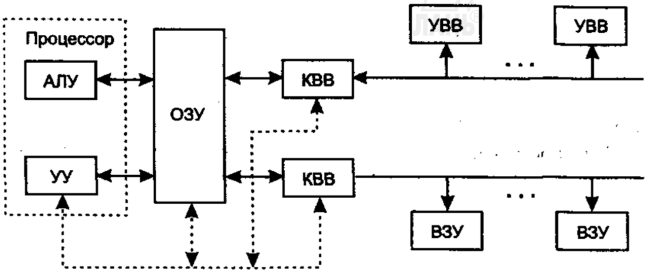


Рисунок 13. Структурная схема ЭВМ 3-го поколения (иерархическая)

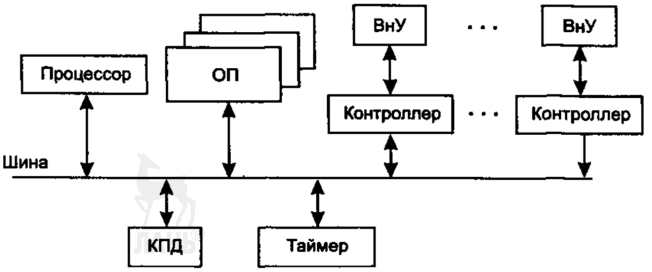


Рисунок 14. Структурная схема ПЭВМ

Архитектура «звезда». Здесь ЦУ соединено непосредственно с ВУ и управляет их работой (ранние модели машин).

Классическая архитектура (фон Неймана) — одно арифметико-логическое устройство (АЛУ), через которое проходит поток данных, и одно устройство управления (УУ), через которое проходит поток команд — программа. Это однопроцессорный компьютер.

Вычислительная машина включает пять базовых компонент и состоит из следующих типов устройств:

− центральный процессор (ЦП), включающий АЛУ и УУ;

− запоминающие устройства — память, в том числе оперативная (ОП) и внешние ЗУ;

− устройства ввода и устройства вывода информации — внешние (периферийные) устройства (ВУ).

Иерархическая архитектура — ЦУ соединено с периферийными процессорами (вспомогательными процессорами, каналами и пр.), управляющими,

в свою очередь, контроллерами, к которым подключены группы ВУ (системы IBM 360-375).

Магистральная структура (общая шина — unibas) — процессор (процессоры) и блоки памяти (ОП) взаимодействуют между собой и с ВУ (контроллерами ВУ) через внутренний канал, общий для всех устройств (машины DEC, ПЭВМ IBM PC-совместимые).

К этому типу архитектуры относится также архитектура персонального компьютера: функциональные блоки здесь связаны между собой общей шиной, называемой также системной магистралью.

Физически *магистраль* представляет собой многопроводную линию с гнездами для подключения электронных схем. Совокупность проводов магистрали разделяется на отдельные группы: шину адреса, шину данных и шину управления.

Периферийные устройства (принтер и др.) подключаются к аппаратуре компьютера через специальные *контроллеры* — устройства управления периферийными устройствами.

Контроллер — устройство, которое связывает периферийное оборудование или каналы связи с центральным процессором, освобождая процессор от непосредственного управления функционированием данного оборудования.

Мы хотим обратить внимание читателя на тот факт, что все перечисленные архитектурные элементы ЭВМ базируются на следующих схемных элементах и базовых узлах:

− *память* обычно использует возможности и свойства триггера или его аналогов;

− *счетчик* (регистр) адреса команд, очевидно, есть схемный узел «счетчик»;

− *сумматор* — или полный сумматор, или полусумматор;

− *дешифратор* (например, команд) тоже здесь присутствует.

Контрольные вопросы:

1. Что такое архитектура компьютера?
2. Что такое структура компьютера?
3. Какие общие принципы лежат в основе построения большинства компьютеров?
4. Какие пять базовых компонент включает вычислительная машина?

Лекция 8. Процессор, структура и функционирование

План:

1. Реализация принципов фон Неймана.
2. Абстрактное центральное устройство.
3. Микропроцессор.
4. Технологии повышения производительности процессоров.
5. Матричные и векторные процессоры.
6. Динамическое исполнение.
7. Процессор Pentium.
8. Классы процессоров.

1. Реализация принципов фон Неймана

В большинстве машин реализованы принципы фон Неймана в следующем виде:

− *оперативная память (ОП)* организована как совокупность *машинных слов (МС) фиксированной длины или разрядности* (имеется в виду количество двоичных единиц или бит, содержащихся в каждом МС). Например, ранние ПЭВМ имели разрядность 8, затем появились 16-разрядные, а затем — 32- и 64-разрядные машины. В свое время существовали также 45-разрядные (М-20, М-220), 35-разрядные (Минск-22, Минск-32) и др. машины;

− ОП образует единое адресное пространство, адреса МС возрастают от младших к старшим;

− в ОП размещаются как данные, так и программы, причем в области данных одно слово, как правило, соответствует одному числу, а в области программы — одной команде (машинной инструкции — минимальному и неделимому элементу программы);

− команды выполняются в *естественной последовательности* (по возрастанию адресов в ОП), пока не встретится *команда управления* (условного/безусловного перехода, или ветвления — branch), в результате которой естественная последовательность нарушится;

− ЦП может произвольно обращаться к любым адресам в ОП для выборки и/или записи в МС чисел или команд.

2. Абстрактное центральное устройство

Перечислим основные понятия и рассмотрим структуру и функции абстрактного центрального устройства ЭВМ (1–2-е поколения ЭВМ) (рис. 15), АЛУ которого предназначено для обработки целых чисел и битовых строк.

Команда (instruction) — описание операции, которую необходимо выполнить. Каждая команда начинается с *кода операции (КОП),* содержит необходимые *адреса,* характеризуется форматом, который определяет структуру команды, ее организацию, код, длину, метод расположения адресов. Длина различных команд может быть как одинаковой, так и разной.

Команды подразделяются на арифметические, логические, ввода-вывода, передачи данных. Каждая команда выполняется в компьютере за один либо несколько *тактов.*

Последовательность взаимосвязанных команд именуется *макрокомандой.* Использование макрокоманд упрощает программирование и обеспечивает механизм вставки добавлений в программы.

Цикл процессора — период времени, за который осуществляется выполнение команды исходной программы в машинном виде; состоит из нескольких *тактов.*

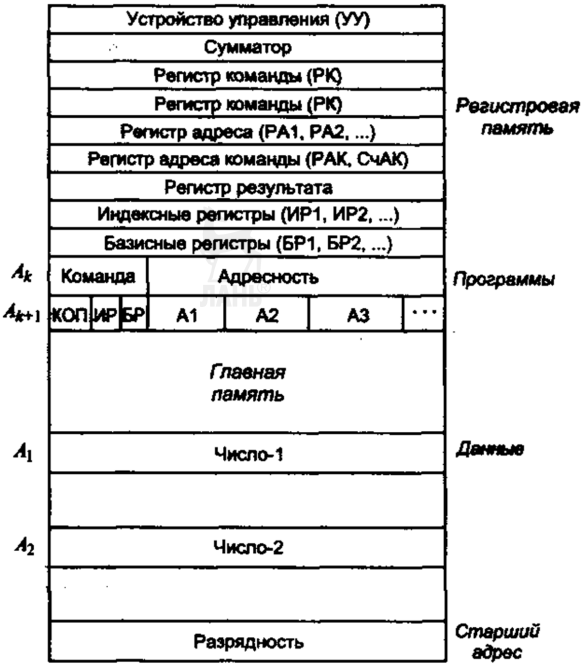


Рисунок 15. Структура абстрактного центрального устройства ЭВМ

Такт работы процессора — промежуток времени между соседними импульсами *генератора тактовых импульсов,* частота которых есть *тактовая частота процессора.* Эта частота является одной из основных характеристик компьютера и во многом определяет скорость его работы, поскольку каждая операция в вычислительной машине выполняется за определенное количество тактов. Выполнение *короткой команды* (арифметика с фиксированной точкой, логические операции), о которой речь здесь и пойдет, обычно занимает пять тактов:

- выборка команды;
- расшифровка кода операции (декодирование);
- генерация адреса и выборка данных из памяти;
- выполнение операции;
- запись результата в память.

Процедура, соответствующая такту, реализуется определенной логической цепью (схемой) процессора, обычно именуемой *микропрограммой.*

3. Микропроцессор

Микропроцессор (МП) — это выращенный по специальной технологии кристалл кремния («камень»), который содержит множество отдельных элементов — транзисторов.

Архитектура процессора — это логическая организация, определяющая возможности аппаратной и программной реализации функций в ЭВМ.

МП состоит из трех основных блоков: АЛУ, блока внутренних регистров и УУ.

Для передачи данных между этими блоками используется внутренняя шина данных.

Перечень функций АЛУ зависит от типа МП. Функции АЛУ определяют архитектуру МП в целом.

Регистры — устройства, предназначенные для временного хранения данных ограниченного размера. Важной характеристикой регистра является высокая скорость приема и выдачи данных. Регистр состоит из разрядов, в которые можно быстро записывать, запоминать и считывать слово, команду, двоичное число и т. д. Обычно регистр имеет ту же разрядность, что и машинное слово. Регистр, накапливающий данные, именуют *аккумулятором.*

Регистр, обладающий способностью перемещать содержимое своих разрядов, называют *сдвиговым.* В этих регистрах за один такт хранимое слово поразрядно сдвигается на одну позицию. Сдвиговые регистры используются при обработке данных, кодировании и декодировании.

Некоторые регистры служат *счетчиками.* Счетчик является устройством, которое на своих выходах выдает (в двоичной форме) сумму числа импульсов, подаваемых на его единственный вход. Максимальное число импульсов, которое счетчик может подсчитать, называется его *емкостью.*

Регистры общего назначения (РОН) (General Purpose Registers) — общее название для регистров, которые временно содержат данные, передаваемые или принимаемые из памяти.

Регистр команды (РК) (Instruction Register — IR) служит для размещения текущей команды, которая находится в нем в течение текущего цикла процессора.

Регистр (РАК), *счетчик* (СчАК) *адреса команды* (Program Counter — PC) — регистр, содержащий адрес текущей команды.

Регистр адреса (*числа*) (РА (Ч)) — содержит адрес одного из операндов выполняемой команды (регистров может быть несколько).

Регистр числа (РЧ) — содержит операнд выполняемой команды, РЧ также несколько.

Регистр результата (РР) — предназначается для хранения результата выполнения команды.

Сумматор — регистр, осуществляющий операции сложения (логического и арифметического двоичного) чисел или битовых строк, представленных в *прямом или обратном коде* (иногда РЧ и РР включают в состав сумматора).

Существуют и другие регистры, не отмеченные на рис. 15, например регистр состояния — Status Register (SR). Типичным содержанием SR является информация о результатах завершения команды (ноль, переполнение, деление на ноль, перенос и пр.). УУ использует информацию из SR для исполнения условных переходов (например, «в случае переполнения перейти по адресу 4170»).

Цикл выполнения команды может выглядеть следующим образом.

1. В соответствии с содержимым СчАК (адрес очередной команды) УУ извлекает из ОП очередную команду и помещает ее в РК.

Некоторые команды УУ обрабатывает самостоятельно, без привлечения АЛУ (например, по команде «перейти по адресу 2478» величина 2478 сразу заносится в СЧАК, и процессор переходит к выполнению следующей команды).

Типичная команда содержит:

− код операции (КОП) — характеризующий тип выполняемого действия (сложение, вычитание и пр. чисел; сравнение строк; передача управления, обращение к ВУ и пр.);

− номера индексного (ИР) и базисного (БР) регистров (в некоторых машинах — адреса слов, ячеек ОП, в которых размещена соответствующая информация);

− адреса операндов Al, А2 и т. д., участвующих в выполнении команды (чисел, строк, других команд программы).

2. Осуществляется расшифровка (декодирование) команды.

3. Адреса Al, А2 и пр. помещаются в регистры адреса.

4. Если в команде указаны ИР или БР, то их содержимое используется для модификации РА — фактически выбираются числа или команды, смещенные в ту или иную сторону по отношению к адресу, указанному в команде.

При этом ИР используются для текущего изменения адреса, связанного с работой программы (например, при обработке массива чисел). БР используется для глобального смещения программы или данных в ОП.

5. По значениям РА осуществляются чтение чисел (строк) и помещение их в РЧ.

6. Выполнение операции (арифметической, логической и пр.) и помещение результата в РР.

7. Запись результата по одному из адресов (если необходимо).

8. Увеличение содержимого СчАК на единицу (переход к следующей команде).

Очевидно, что за счет увеличения числа регистров возможно *распараллеливание, перекрытие* операций. Например, при считывании команды СчАК можно автоматически увеличить на 1, подготовив выборку следующей команды. После расшифровки текущей команды РК освобождается, и в него может быть помещена следующая команда программы. При выполнении операции возможна расшифровка следующей команды и т. д. Все это является предпосылкой построения так называемых *конвейерных структур (pipeline).* Однако все это хорошо только при последовательном (естественном) порядке выполнения команд. Появление переходов (особенно по не определенному заранее условию) нарушает эту картину. Поэтому современные процессоры пытаются предсказывать переходы в программе *(branch prediction).*

4. Технологии повышения производительности процессоров

Конвейерная обработка команд (pipelining). Суперскаляризация

Рассмотрим процесс выполнения процессором команды для *коротких* (с фиксированной запятой или логических) операций. Обработка команды, или цикл процессора, может быть разделена на несколько основных этапов, которые можно назвать *микрокомандами,* которых известно пять основных типов.

Каждая операция требует для своего выполнения времени, равного такту генератора процессора (tick of the internal clock). Отметим, что к длинным операциям (плавающая точка) это не имеет отношения — там другая арифметика.

Очевидно, что при тактовой частоте в 100 МГц быстродействие составит 20 миллионов операций в секунду.

Все этапы команды задействуются только один раз и всегда в одном и том же порядке: одна за другой. Это, в частности, означает, что если первая микрокоманда выполнила свою работу и передала результаты второй, то для выполнения текущей команды она больше не понадобится и, следовательно, может приступить к выполнению следующей команды.

Конвейеризация осуществляет многопоточную параллельную обработку команд, так что в каждый момент одна из команд считывается, другая декодируется и т. д., и всего в обработке одновременно находится пять команд. Таким образом, на выходе конвейера на каждом такте процессора появляется результат обработки одной команды (одна команда в один такт). Первая инструкция может считаться выполненной, когда завершат работу все пять микрокоманд.

Такая технология обработки команд носит название *конвейерной (pipeline)* обработки. Каждая часть устройства называется *ступенью конвейера,* а общее число ступеней — *длиной линии конвейера.*

С ростом числа линий конвейера и увеличением числа ступеней на линии (табл. 28) увеличивается пропускная способность процессора при неизменной тактовой частоте. Процессоры с несколькими линиями конвейера получили название *суперскалярных.* Pentium — первый суперскалярный процессор Intel. Здесь две линии, что позволяет ему при одинаковых частотах быть вдвое производительней 180486, выполняя сразу две инструкции за такт.

Во многих вычислительных системах, наряду с *конвейером команд,* используются *конвейеры данных.*

Таблица 28. Характеристики конвейеров процессоров Intel

Процессор	**180486**	**Pentium**	**Pentium Pro**	**Pentium MMX**	**Pentium II**	**Pentium III**	**Pentium IV**
Число линий	1	2	3	2	3	3	3
Длина линии	5	5	14	6	14	20	31 (Hyper-Pipeline)

Сочетание этих двух конвейеров дает возможность достичь очень высокой производительности на определенных классах задач, особенно если используется несколько различных конвейерных процессоров, способных работать одновременно и независимо друг от друга.

Одной из наиболее высокопроизводительных вычислительных конвейерных систем считается CRAY. В этой системе конвейерный принцип обработки используется в максимальной степени. Имеется как конвейер команд, так и конвейер арифметических и логических операций. В системе широко применяется совмещенная обработка информации несколькими устройствами. Максимальная пиковая производительность процессора может составлять 12 Гфлопс.

5. Матричные и векторные процессоры

В отличие от скалярных и даже суперскалярных процессоров данные устройства манипулируют массивами данных и предназначены для обработки изображений, матриц и массивов данных. Частным случаем векторного процес-

сора является процессор изображений, который предназначен для обработки сигналов, поступающих от датчиков-формирователей изображения.

Матричный процессор имеет архитектуру, рассчитанную на обработку числовых массивов. Архитектура процессора включает в себя матрицу процессорных элементов, например 64×64, работающих одновременно. Постпроцессор предназначен для реализации некоторых специальных функций, например управления базой данных.

Векторный процессор обеспечивает параллельное выполнение операции над массивами данных, векторами. Он характеризуется специальной архитектурой, построенной на группе параллельно работающих процессорных элементов.

Векторная обработка увеличивает производительность процессора за счет того, что обработка целого набора данных (вектора) производится одной командой. Векторные компьютеры манипулируют массивами сходных данных подобно тому, как скалярные машины обрабатывают отдельные элементы таких массивов. В этом случае каждый элемент вектора надо рассматривать как отдельный.

При работе в векторном режиме векторные процессоры обрабатывают данные практически параллельно, что делает их в несколько раз более быстрыми, чем при работе в скалярном режиме. Максимальная скорость передачи данных в векторном формате может составлять 64 Гбайт/с, что на два порядка быстрее, чем в скалярных машинах.

В настоящее время созданы однокристальные векторно-конвейерные процессоры, такие как SX-6. Основными компонентами микропроцессора являются скалярный процессор и восемь идентичных векторных устройств, суммарная производительность которых составляет 64 Гфлопс. Примерами систем подобного типа являются, например, процессоры фирм NEC и Hitachi.

6. Динамическое исполнение

Это совокупность технологий обработки данных в процессоре, обеспечивающая более эффективную работу процессора за счет манипулирования данными, а не простого исполнения списка инструкций. Динамическое исполнение представляет собой комбинацию трех методов обработки данных:

- множественное предсказание ветвлений;
- анализ потока данных;
- спекулятивное (по предположению) исполнение.

Впервые реализовано в процессоре Pentium Pro.

Множественное предсказание ветвлений. Предсказывается прохождение программы по нескольким ветвям: процессор может предвидеть разделение потока инструкций, используя алгоритм множественного предсказания ветвлений. С большой точностью (более 90%) он предсказывает, в какой области памяти можно найти следующие инструкции. Это оказывается возможным, поскольку в процессе исполнения инструкции процессор просматривает программу на несколько шагов вперед. Этот метод позволяет увеличить загруженность процессора.

Хотя BTB (Branch Target Buffer — буфер предсказания переходов) и не может правильно предсказать абсолютно все переходы, но большинство предсказаний оказываюся точными, что обеспечивает значительное повышение производительности. Например, программный цикл, состоящий из пересылки, сравнения, сложения и перехода в 80486 DX выполняется за 6 тактов синхронизации, а в Pentium за 2 (команды пересылки и сложения, а также сравнения и перехода сочетаются, и предсказывается переход).

Анализ потока данных. Анализируется и составляется график исполнения инструкций в оптимальной последовательности, независимо от порядка их следования в тексте программы. Используя анализ потока данных, процессор просматривает декодированные инструкции и определяет, готовы ли они к непосредственному исполнению или зависят от результата других инструкций. Далее процессор определяет оптимальную последовательность выполнения и исполняет инструкции наиболее эффективным образом.

Спекулятивное выполнение. Повышает скорость выполнения, просматривая программу вперед и исполняя те инструкции, которые необходимы. Процессор выполняет инструкции (до пяти инструкций одновременно) по мере их поступления в оптимизированной последовательности (спекулятивно). Поскольку выполнение инструкций происходит на основе предсказания ветвлений, результаты сохраняются как «спекулятивные». На конечном этапе порядок инструкций восстанавливается и переводится в обычное машинное состояние.

Процессоры уровня IA-64 имеют мощные вычислительные ресурсы, включая 128 регистров целых чисел, 128 регистров действительных чисел, 64 предикационных регистра, а также ряд специальных регистров. Набор команд оптимизирован для решения задач криптографии, обработки видеосигналов и других процессов, встречающихся в серверах и рабочих станциях.

Предикация — центральный метод планирования параллельной обработки. Компилятор транслирует операторы исходного кода, содержащие ветвление (условный переход), в совокупность блоков машинных команд, идущих друг за другом. Обычный процессор, в зависимости от исхода условия, исполняет один из этих базовых блоков, пропуская все другие. Более развитые процессоры пытаются прогнозировать исход операции перехода и заранее *(спекулятивно, по предположению)* выполняют один из блоков, теряя время при ошибке прогнозирования.

Базовые блоки обычно малы (2–3 команды), и ветвление встречается в среднем через каждые шесть операторов языка программирования. Поэтому выигрыш оказывается небольшим.

Когда компилятор уровня IA-64 обнаруживает оператор ветвления в исходной программе, он анализирует все возможные ветви (блоки) и помечает их метками или *предикатами (predicate).* После этого он определяет, какие из них могут быть выполнены параллельно (из соседних, независимых, ветвей).

Затем компилятор группирует машинные коды в 128-битовые *связки (bundles),* по 3 команды в каждой. В *описания связок (template)* заносится информация о том, какие команды могут исполняться параллельно (независимо). Например, если компилятор находит 16 команд, которые не имеют взаимной

связи, он укладывает их в 6 независимых связок (по 3 в первых 5 и одна в 6-й) и помечает их в описании.

В процессе выполнения программы ЦП просматривает описания связок, выбирает команды, которые взаимно независимы, и распределяет их на параллельную обработку. Если ЦП обнаруживает оператор ветвления, он не пытается предсказать переход, а начинает выполнять все возможные ветви программы.

Таким образом, могут быть обработаны все ветви программы, но без записи полученного результата. В определенный момент процессор наконец «узнает» о реальном исходе условного оператора, записывает в память результат «правильной ветви» и отменяет остальные результаты.

Опережающее чтение (по предположению) разделяет загрузку данных в регистры и их реальное использование, избегая ситуации, когда процессору приходится ожидать прихода данных, чтобы начать их обработку.

Прежде всего, компилятор анализирует программу, определяя команды, которые требуют приема данных из оперативной памяти. Там, где это возможно, он вставляет команды опережающего чтения и парную команду контроля опережающего чтения (speculative check). В то же время компилятор переставляет команды таким образом, чтобы ЦП мог их обрабатывать параллельно.

В процессе работы ЦП встречает команду опережающего чтения и пытается выбрать данные из памяти. Может оказаться, что они еще не готовы (результат работы блока команд, который еще не выполнился). Обычный процессор в этой ситуации выдает сообщение об ошибке, однако система уровня IA-64 откладывает «сигнал тревоги» до момента прихода процесса в точку «команда проверки опережающего чтения». Если к этому моменту все предшествующие подпроцессы завершены и данные считаны, то обработка продолжается, в противном случае вырабатывается сигнал прерывания.

7. Процессор Pentium

Здесь реализуется разделение времени на аппаратном уровне: физически процессор разбивается на два логических процессора, каждый из которых использует ресурсы чипа — ядро, кэш-память, шины, исполнительное устройство. Ядро процессора выполняет два процесса одновременно.

Специалисты Intel оценивают повышение эффективности в 30% при использовании на HT-процессорах многопрограммных ОС и обычных прикладных программ.

Процессор Pentium состоит из следующих блоков (рис. 16).

Ядро (Core). Основное исполнительное устройство.

Производительность МП при тактовой частоте 66 МГц составляет около 112 млн инструкций в секунду (Mips). Пятикратное повышение (по сравнению с 80486 DX) достигалось благодаря двум конвейерам, позволяющим выполнить одновременно несколько инструкций. Это два параллельных 5-ступенчатых конвейера обработки целых чисел, которые позволяют читать, интерпретировать, исполнять две команды одновременно.

Целочисленные команды могут выполняться за один такт синхронизации. Эти конвейеры неодинаковы: U-конвейер выполняет любую команду системы команд семейства 86; V-конвейер выполняет только «простые» команды, т. е.

команды, которые полностью встроены в схемы МП и не требуют микропрограммного управления (microcode) при выполнении (это команды, допускающие спаривание с другими командами: регистр — регистр, память — регистр, регистр — память, переходы, вызовы, арифметико-логические операции).

Предсказатель переходов (Branch Predictor). Пытается угадать направление ветвления программы и заранее загрузить информацию в блоки предвыборки и декодирования команд.

Буфер адреса переходов (Branch Target Buffer, BTB). Обеспечивает динамическое предсказание переходов. Он улучшает выполнение команд путем запоминания состоявшихся переходов (256 последних переходов) и с опережением выполняет наиболее вероятный переход при выборке команды ветвления. Если предсказание верно, то эффективность увеличивается. Если нет, то конвейер приходится сбрасывать полностью. Согласно данным Intel, вероятность правильного предсказания переходов в процессорах Pentium, Pentium MMX составляет 75–80%, а для Pentium Pro, Pentium II — 90%.

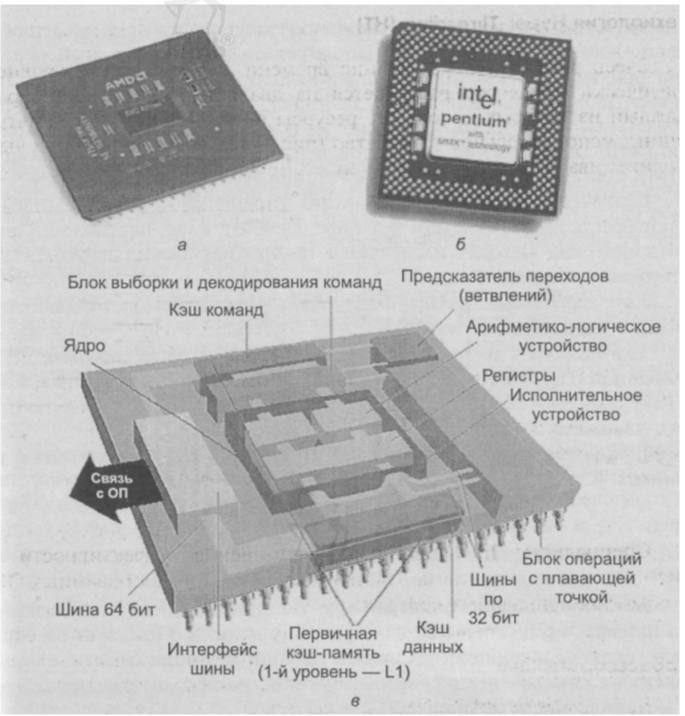


Рисунок 16. Микропроцессоры AMD (а), Intel Pentium MMX (б), основные компоненты процессора Pentium (в)

Статические методы предсказания упрощены — они предписывают всегда выполнять или нет определенные виды переходов. При динамических методах исследуется поведение команд перехода за предшествующий период.

Блок плавающей точки (Floating Point Unit). Выполняет обработку чисел с плавающей точкой.

Кэш-память 1-го уровня (Level I cache, L1). Процессор имеет два банка памяти по 8 Кбайт: 1-й — для команд, 2-й — для данных, которые обладают бóльшим быстродействием, чем более емкая внешняя кэш-память (L2 cache).

Интерфейс шины (Bus Interface). Передает в ЦП поток команд и данных, а также передает данные из ЦП.

8. Классы процессоров

В зависимости от набора и порядка выполнения команд процессоры подразделяются на четыре класса, отражающих также последовательность развития ЭВМ. Ранее других появились процессоры CISC. Затем с целью повышения быстродействия процессоров были разработаны процессоры RISC, которые характеризуются сокращенным набором быстро выполняемых команд. Ряд редко встречающихся команд процессора CISC выполняется последовательностями команд процессора RISC. Позже появилась концепция процессоров MISC, использующая минимальный набор длинных команд. Вслед за ними возникли процессоры VL1W, работающие со сверхдлинными командами.

CISC (Complex Instruction Set Computer) есть традиционная архитектура, в которой ЦП использует микропрограммы для выполнения исчерпывающего набора команд. Они могут иметь различную длину, методы адресации и требуют сложных электронных цепей для декодирования и исполнения. В течение долгих лет производители компьютеров разрабатывали и воплощали в изделиях все более сложные и полные системы команд. Однако анализ работы процессоров показал, что в течение примерно 80% времени выполняется лишь 20% большого набора команд. Поэтому была поставлена задача оптимизации выполнения небольшого числа часто используемых команд.

В 1974 г. John Cocke (IBM Research) решил испробовать подход, который мог бы существенно уменьшить количество машинных команд в ЦП. В середине 1970-х гг. это привело многих производителей компьютеров к пересмотру своих позиций и к разработке ЦП с весьма ограниченным набором команд.

RISC (Redused Instuction Set Computer) — процессор, функционирующий с сокращенным набором команд. Так, в процессоре CISC для выполнения одной команды необходимо в большинстве случаев 10 и более тактов. Что же касается процессоров RISC, то они близки к тому, чтобы выполнять по одной команде в каждом такте. Следует также иметь в виду, что благодаря своей простоте процессоры RISC не патентуются. Это также способствует их быстрой разработке и широкому производству. Между тем в сокращенный набор RISC вошли только наиболее часто используемые команды.

Первый процессор RISC был создан корпорацией IBM в 1979 г. и имел шифр IBM 801. В настоящее время процессоры RISC получили широкое распространение. Современные процессоры RISC характеризуются следующим:

− упрощенный набор команд, имеющих одинаковую длину;

− большинство команд выполняются за один такт процессора;

− отсутствуют макрокоманды, усложняющие структуру процессора и уменьшающие скорость его работы;

− взаимодействие с оперативной памятью ограничивается операциями пересылки данных;

− резко уменьшено число способов адресации памяти (не используется косвенная адресация);

− используется конвейер команд, позволяющий обрабатывать несколько из них одновременно;

− применяется высокоскоростная память.

Новый подход к архитектуре процессора значительно сократил площадь, требуемую для него на кристалле интегральной схемы. Это позволило резко увеличить число регистров. В современном процессоре RISC уже используется более 100 регистров. В результате процессор на 20–30% реже обращается к оперативной памяти, что также повысило скорость обработки данных. Упростилась топология процессора, выполняемого в виде одной интегральной схемы, сократились сроки ее разработки, она стала дешевле.

Начиная с процессора Pentium корпорация Intel начала внедрять элементы RISC-технологий в свои изделия.

Процессор MISC — процессор, работающий с минимальным набором длинных команд.

Увеличение разрядности процессоров привело к идее укладки нескольких команд в одно слово (связку, bound) размером 128 бит. Оперируя с одним словом, процессор получил возможность обрабатывать сразу несколько команд. Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных.

Процессор MISC, как и процессор RISC, характеризуется небольшим набором чаще всего встречающихся команд. Вместе с этим принцип команд VLIW обеспечивает выполнение группы команд за один цикл работы процессора. Порядок выполнения команд распределяется таким образом, чтобы в максимальной степени загрузить маршруты, по которым проходят потоки данных. Таким образом, архитектура MISC объединила вместе суперскалярную (многопоточную) и VLIW концепции. Компоненты процессора просты и работают с высокими скоростями.

Процессор VLIW — процессор, работающий с системой команд сверхбольшой разрядности.

Идея технологии VLIW заключается в том, что создается специальный компилятор планирования, который перед выполнением прикладной программы проводит ее анализ и по множеству ветвей последовательности операций определяет группу команд, которые могут выполняться параллельно. Каждая такая группа образует одну сверхдлинную команду. Это позволяет решать две важные задачи: во-первых, в течение одного такта выполнять группу коротких («обычных») команд и, во-вторых, упростить структуру процессора. Этим технология VLIW отличается от суперскалярности. В последнем случае отбор групп одновременно выполняемых команд происходит непосредственно в ходе выполнения прикладной программы (а не заранее), из-за чего усложняется структура процессора и замедляется скорость его работы.

Технология VLIW появилась в результате работ, проведенных корпорациями HP и Intel.

Контрольные вопросы:

1. Дайте определение процессора.
2. Назовите основные функции центрального процессора.
3. Назовите характеристики центрального процессора.
4. Каково назначение кэш-памяти?
5. Какую функцию выполняет конвейер микрокоманд?
6. Что представляют собой функциональные устройства?
7. Для чего предназначены регистры?
8. Что определяет внешнюю производительность процессора?
9. Как подразделяются процессоры в зависимости от набора и порядка выполнения команд?

Лекция 9. Устройство управления. Структура команд процессора

План:

1. Устройство управления.
2. Структура команд процессора.

1. Устройство управления

Процессор условно можно разделить на два основных блока: операционный и управляющий. Для реализации любой команды необходимо на соответствующие управляющие входы любого устройства компьютера подать определенным образом распределенную во времени последовательность управляющих сигналов. Часть цифрового вычислительного устройства, предназначенная для выработки этой последовательности, называется устройством управления.

Любое действие, выполняемое в операционном блоке, описывается некоторой микропрограммой и реализуется за один или несколько тактов. Элементарная функциональная операция, выполняемая за один тактовый интервал и приводимая в действие управляющим сигналом, называется микрооперацией [7]. Например, в спроектированном АЛУ для умножения чисел в первом такте выполняются следующие микрооперации: TX = 0, TY = 0, RGX = |X|, RGY = |Y|, RGZ = 0. Совокупность микроопераций, выполняемых в одном такте, называется микрокомандой (МК). Если все такты должны иметь одну и ту же длину, а именно это имеет место при работе компьютера, то она устанавливается по самой продолжительной микрооперации. Микрокоманды, предназначенные для выполнения некоторой функционально законченной последовательности действий, образуют микропрограмму. Например, микропрограмму образует набор микрокоманд для выполнения команды умножения.

Устройство управления (УУ) предназначено для выработки управляющих сигналов, под воздействием которых происходят преобразование информации в арифметико-логическом устройстве, а также операции по записи и чтению информации в/из запоминающего устройства.

Устройства управления делятся на:

- УУ с жесткой, или схемной, логикой;
- УУ с программируемой логикой (микропрограммные УУ).

В устройствах управления первого типа для каждой команды, задаваемой кодом операции, строится набор комбинационных схем, которые в нужных тактах вырабатывают необходимые управляющие сигналы.

В микропрограммных УУ каждой команде ставится в соответствие совокупность хранимых в специальной памяти слов — микрокоманд. Каждая из микрокоманд содержит информацию о микрооперациях, подлежащих выполнению в данном такте, и указание, какое слово должно быть выбрано из памяти в следующем такте.

2. Структура команд процессора

Инструкция записывается на отдельной строке и включает до четырех полей, из которых необязательные выделены [ ]:

[метка:]	мнемоника_команды	[операнд(ы)]	[;комментарий]

Метка или символический адрес содержит до 31 символа из букв цифр и знаков ? @ . _ $. Причем цифра не должна стоять первой, а точка, если есть, должна быть первой.

Мнемоника — сокращенное обозначение кода операции (КОП) команды, например мнемоника ADD обозначает сложение (addition).

Операндами могут быть явно или неявно задаваемые двоичные наборы, над которыми производятся операции. Операнды приводятся в одной из четырех систем счисления и должны оканчиваться символом b (B), o (O), d (D) или h (H) для 2, 8, 10 или 16-й СС. К шестнадцатеричному числу добавляется слева ноль, если оно начинается с буквы.

Основные команды ЭВМ классифицируются вкратце следующим образом: по функциям (выполняемым операциям), направлению приема-передачи информации, адресности.

Классы команд

1. Команды обработки данных, в том числе (01 — первый операнд, 02 — второй).

1.1. Короткие операции (один такт).

1.1.1. Логические:

− логическое сложение (для каждого бита 01 и 02 осуществляется операция ИЛИ);

− логическое умножение (для каждого бита определяется операция И);

− инверсия (в 01 все единицы заменяются на нули, и наоборот);

− сравнение логическое (если 01 = 02, то некий флаг или регистр устанавливается в «1», иначе в «0»).

1.1.2. Арифметические:

− сложение операндов;

− вычитание (сложение в обратном коде);

− сравнение арифметическое (если 01 > 02, или 01 = 02, или 01 < 02, то некий флаг или регистр устанавливается в 1, иначе — в 0).

1.2. Длинные операции (несколько тактов):

− сложение/вычитание с фиксированной точкой;

− умножение/деление с фиксированной точкой.

2. Операции управления:

− безусловный переход (ветвление, branch);

− условный переход (по условию, результатам вычислений (conditional branch)).

3. Операции обращения к внешним устройствам (требование на запись или считывание информации).

Естественно, могут существовать и другие операции — *десятичная арифметика,* обработка *символьной* информации, работа с числами *половинной* (полуслово — например, 16 бит) или *двойной* (двойное слово — например, 64 бит) длины.

Кроме того, команды различаются по типу выборки и пересылок данных:

− регистр — регистр (01 и 02 размещаются в регистрах АЛУ);

− память — регистр (регистр — память) — один из операндов размещается в ОП;

− память — память (01 и 02 размещены в ОП).

Далее известны одно-, двух- и трехадресные машины (системы команд). Очевидна связь таких параметров ЦУ, как длина адресного пространства, адресность, разрядность. Увеличение разрядности позволяет увеличить адресность команды и длину адреса (т. е. объем памяти, доступной данной команде). Увеличение адресности, в свою очередь, приводит к повышению быстродействия обработки (за счет снижения числа требуемых команд).

В трехадресной машине, например, сложение двух чисел требует одной команды (извлечь число по А1, число по А2, сложить и записать результат по А3). В двухадресной необходимы две команды (первая — извлечь число по А1 и поместить в РЧ (или сумматор), вторая — извлечь число по А1, сложить с содержимым РЧ и результат записать по А2). Легко видеть, что одноадресная машина потребует три команды. Поэтому неудивительно, что основная тенденция в развитии ЦУ ЭВМ состоит в увеличении разрядности.

Типовая структура трехадресной команды

КОП	А1	А2	А3

где А2 и А3 — адреса ячеек (регистров), где расположены соответственно первое и второе числа, участвующие в операции; А1 — адрес ячейки (регистра), куда следует поместить число, полученное в результате выполнения операции.

Типовая структура двухадресной команды

КОП	А1	А2

где А1 — это обычно адрес ячейки (регистра), где хранится первое из чисел, участвующих в операции, и куда после завершения операции должен быть записан результат операции; А2 — обычно адрес ячейки (регистра), где хранится второе участвующее в операции число.

Типовая структура одноадресной команды

КОП	A1

где A1 в зависимости от модификации команды может обозначать либо адрес ячейки (регистра), в которой хранится одно из чисел, участвующих в операции, либо адрес ячейки (регистра), куда следует поместить число — результат операции.

Безадресная команда содержит только код операции, а информация для нее должна быть заранее помещена в определенные регистры машины.

Наибольшее применение нашли *двухадресные системы* команд.

Таким образом, программирование в машинных адресах требует знания системы команд конкретной ЭВМ и их адресности. При этом реализация даже довольно несложных вычислений требует разложения их на простые операции, что значительно увеличивает общий объем программы и затрудняет ее чтение и отладку.

В качестве примера рассмотрим последовательность реализации вычисления по формуле $y = (a + b)^2 – c/d$.

План последовательности машинных операций, выполнение которой приведет к нужному результату, в данном случае следующий:

rl = a + b; — операция сложения;
r2 = rl * rl; — операция умножения;
r3 = c/d; — операция деления;
y = r2 – r3; — операция вычитания;
Стоп — завершение обработки.

Здесь количество переменных, необходимых для хранения промежуточных результатов, связано с адресностью системы команд и с тем, разрешено или нет в процессе вычислений изменять значения исходных данных.

Контрольные вопросы:

1. Для чего предназначено устройство управления процессора?
2. Как делятся УУ?
3. Что такое мнемоника?
4. Как классифицируют основные команды ЭВМ?

Лекция 10. АЛУ: назначение и классификация

План:

1. Назначение АЛУ.
2. Классификация АЛУ.

1. Назначение АЛУ

Arithmetic and Logical Unit (ALU) — компонента процессора, выполняющая арифметические и логические операции над данными.

АЛУ реализует важную часть процесса обработки данных. Она заключается в выполнении набора простых операций. 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, ...).

АЛУ состоит из регистров, сумматора с соответствующими логическими схемами и блока управления выполняемым процессом.

Устройство работает в соответствии с сообщаемыми ему именами (кодами) операций, которые при пересылке данных нужно выполнить над переменными, помещаемыми в регистры.

2. Классификация АЛУ

АЛУ классифицируются следующим образом:

1. По способу действий над операндами:

− последовательного действия;

− параллельного действия.

В последовательных АЛУ действия над операндами производятся последовательно разряд за разрядом начиная с младшего. В параллельных АЛУ все разряды операндов обрабатываются одновременно.

2. По виду обрабатываемых чисел. АЛУ могут производить операции над двоичными числами с фиксированной или плавающей запятой и над двоично-десятичными числами. В последнем случае каждая десятичная цифра записывается четырьмя разрядами двоичного кода:

$2003_{10} = 0010\ \ 0000\ \ 0000\ \ 0011_2$.

АЛУ при действии над двоично-десятичными числами должны содержать схему десятичной коррекции. Схема десятичной коррекции преобразует полученный результат таким образом, чтобы каждый двоично-десятичный разряд не содержал цифру больше 9.

При записи числа с фиксированной запятой запятая фиксируется после младшего разряда, если число целое, и перед старшим, если число меньше 1.

При записи чисел с плавающей запятой выделяется целая часть, которая называется мантиссой, и показатель степени, который характеризует положение запятой.

3. По организации действий над операндами:

− блочные;

− многофункциональные.

В блочных АЛУ отдельные блоки предназначены для действий над двоично-десятичными числами, отдельно для действий над числами с фиксированной запятой, отдельно с плавающей запятой.

В многофункциональных АЛУ одни и те же блоки обрабатывают числа с фиксированной запятой, плавающей запятой и двоично-десятичные числа (рис. 17).

Клапаны К1 и К2 объединяют сумматоры 1, 2 и 3 для действий над числами с фиксированной запятой.

Для действий над числами с плавающей запятой клапан К2 объединяет сумматоры 2 и 3 для обработки мантисс, а клапан К1 отсоединяет первый сумматор от второго. Сумматор 1 обрабатывает порядки.

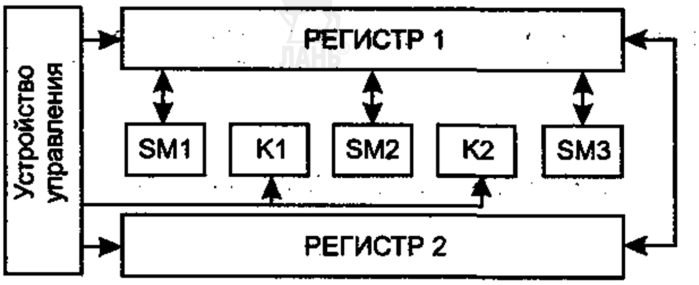


Рисунок 17. Многофункциональное АЛУ

4. По структуре:

− с непосредственными связями;

− многосвязные.

В многосвязных АЛУ входы и выходы регистров приемников и источников информации подсоединяются к одной шине. Распределение входных и выходных сигналов происходит под действием управляющих сигналов.

В АЛУ с непосредственной связью вход регистра приемника связан с выходом регистра источника операндов и регистра, в котором происходит обработка (рис. 18).

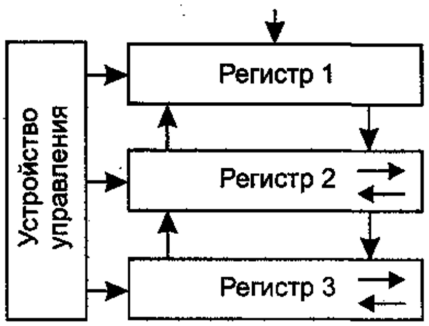


Рисунок 18. АЛУ с непосредственной связью

Например, в этой схеме суммирование происходит так: операнды подаются в регистр 1. Регистр 2 является накапливающим сумматором или автоматом с памятью. Он суммирует слагаемые, поступающие в разные моменты времени и передает результат в регистр 3.

Умножение в этом АЛУ происходит так: множимое помещают в регистр 4, множитель — в регистр 1. Регистры 2 и 3 являются, кроме того, сдвигающими регистрами. В зависимости от содержимого разряда множителя, множимое сдвигается на один разряд, если множитель содержит 1, и на два, если множитель содержит 0. Эти частные произведения суммируются в регистре 2.

Контрольные вопросы:

1. Что такое АЛУ?
2. Как АЛУ классифицируются?

Лекция 11. Материнские платы

План:

1. Архитектура материнских плат.
2. Основные характеристики материнской платы.
3. Форм-факторы материнских плат.

1. Архитектура материнских плат

Важнейшим узлом компьютера является системная плата (system board). В литературе и в повседневной жизни можно встретить такие названия: материнская плата (mother board), основная или главная плата (main board) и др.

Системная плата — это микросхема, содержащая набор компонент электроники, с помощью которых осуществляется взаимодействие узлов (устройств) компьютера. Системная плата покрыта сетью медных проводников-дорожек, по которым подается электропитание и осуществляется передача данных между узлами (устройствами) компьютера.

Системная плата (*англ.* motherboard, MB — матери́нская пла́та, также используется название mainboard — главная плата; на компьютерном жаргоне — мама, мать, материнка) — 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).

Системная плата является основной компонентой компьютера. Приоритет системной платы не случаен, так как она:

– во-первых, обеспечивает связь между компонентами;

– во-вторых, отвечает за функционирование устройств, осуществляя передачу питания для всех элементов;

– в-третьих, она контролирует состояния важнейших узлов.

Таким образом, материнская плата является своеобразным фундаментом для будущей системы, задавая ей основные характеристики, возможности наращивания мощности и даже сроки морального износа. Основными вопросами при выборе материнской платы являются:

- архитектура системной платы;
- основные разъемы и поддерживаемые интерфейсы;
- технологии, которыми она «напичкана».

Архитектура системной платы определяется разработчиком (производителем) материнской платы. При её конструировании используется понятие форм-фактора (form factor).

Форм-фактор представляет собой физические параметры платы и определяет тип корпуса, в котором она может быть установлена. Форм-факторы системных плат могут быть стандартными (т. е. взаимозаменяемыми) или нестандартными. Нестандартные форм-факторы, к сожалению, являются препятствием для модернизации компьютера, поэтому от их использования лучше отказаться.

Форм-фактор, или типоразмер системной платы, определяет ее габариты, параметры электропитания, расположение монтажных элементов (отверстий, клипсов), размещение разъемов различных интерфейсов и т. д.

Форм-фактор (техника) — стандарт технического изделия, описывающий некоторую совокупность его технических параметров.

Существуют множество форм-факторов, среди них наиболее известными являются:

устаревшие:

- AT (полноразмерная, Baby-AT);
- LPX (mini-);

современные:

- NLX;
- ATX (micro-, flex-, mini-);
- BTX (micro-, pico) ;
- WTX;
- CEB;
- ITX (mini-, nano-, pico);

прочие:

- независимые конструкции (разработки компаний Compaq, Packard Bell, HewlettPackard, портативные/мобильные системы и т. д.).

2. Основные характеристики материнской платы

1. Поддерживаемые процессоры. Каждый процессор характеризуется определенным набором параметров. Важнейшими являются тактовые частоты — внутренние и внешние, напряжение питания — одно или несколько, величины напряжений и т. д. Процессоры имеют определенные конструктивные отличия, тесно связанные с особенностями их внутренней структуры. Обычно для идентификации процессора достаточны следующие данные: фирма-изготовитель процессора; тип процессора, например, Pentium, Pentium II/III, AMD Athlon и т. д.; разъем подключения (Socket 7, Slot 1, Socket 370 и т. д.); внешняя и внутренняя частота.

2. Чипсет. В настоящее время на материнских платах используются самые разные чипсеты, которые влияют как на производительность материнской платы и ее функциональные возможности, так и на стоимость платы, а в конечном счете, на цену компьютера. Чипсет обеспечивает связь между основными узлами, расположенными на материнской плате, в первую очередь между процессором и памятью. Поэтому от чипсета также зависит производительность компьютера в целом, т. к. если чипсет медленно работает с памятью, то и система работает медленнее, нежели система с тем же процессором и памятью, но другим, более быстро работающим с памятью чипсетом. Но функция связывания всех компонентов в единую систему — не единственная функция чипсета. Кроме того, современный чипсет содержит целый ряд основных, базовых контроллеров различных устройств, подключаемых к материнской плате.

3. Системные шины и частотные параметры. С помощью существующих перемычек па плате или средствами BIOS можно установить необходимые тактовые частоты процессора: внешнюю — для процессора и его шины (FSB), внутреннюю — для процессора и кэш-памяти L1, L2. Например, одни платы поддерживают 50, 60, 66 МГц, вторые — 66, 100 МГц, третьи осуществляют поддержку 100, 133 МГц. Некоторые позволяют устанавливать не только стан-

дартные частоты — это обычно 60, 66, 100, 133 МГц, соответствующие рекомендованным режимам, но и дополнительные частоты, позволяющие устанавливать форсированные режимы (overclocking).

4. Объем и тип внешней кэш-памяти (L2) для процессоров с разъемом Socket7. От объема и типа кэш-памяти, как известно, зависит общая производительность ПК. Большинство материнских плат имеют объем кэш-памяти 512 Кбайт с возможностью расширения до 1 Мбайт или даже до 2 Мбайт.

5. Объем, тип и количество разъемов оперативной памяти. Большинство современных материнских плат позволяют установить как минимум память до 256 Мбайт DIMM SDRAM, а некоторые — до 1 Гбайт и даже до 1,5 Гбайт.

6. Контроллеры и адаптеры. Современные материнские платы уже включают в себя контроллеры жестких и гибких дисков, а некоторые из плат еще и аудио- и видеоадаптеры. С одной стороны, это обеспечивает компактность ПК и полное отсутствие каких-либо конфликтов между устройствами. С другой стороны — сложнее выполнить модернизацию.

7. Количество и типы разъемов (AGP, PCI, ISA, AMR) для плат контроллеров. Определяют количество и стандарт (AGP, PCI, ISA, AMR) подключения контроллеров, которые могут быть установлены в разъемы (слоты) материнской платы. Это определяет функциональные возможности ПК. Необходимое число и типы слотов зависят как от уже существующих на плате интегрированных контроллеров, так и от решаемых на ПК задач.

3. Форм-факторы материнских плат

Материнские платы классифицируются по так называемому форм-фактору (Form Factor). Форм-фактор определяет не только размеры материнских плат, но и ряд специфических характеристик, определяющих их функциональные и эксплуатационные свойства. При этом каждый формат требует соответствующего корпуса и блока питания. На сегодняшний день существует четыре преобладающих типоразмера материнских плат — AT, ATX, LPX и NLX.

3.1. Форм-фактор AT

Форм-фактор AT делится на две отличающиеся по размеру модификации — AT и Baby AT. Размер полноразмерной AT платы достигает до 12" в ширину, а это значит, что такая плата вряд ли поместится в большинство сегодняшних корпусов. Монтажу такой платы наверняка будут мешать отсек для дисководов и жестких дисков и блок питания. Кроме того, расположение компонентов платы на большом расстоянии друг от друга может вызывать некоторые проблемы при работе на больших тактовых частотах. Поэтому после материнских плат для процессора 386 такой размер уже не встречается.

Таким образом, единственные материнские платы, выполненные в форм-факторе AT, доступные в широкой продаже, — это платы, соответствующие формату Baby AT. Размер платы Baby AT 8,5" в ширину и 13" в длину. Почти все имеют последовательные и параллельные порты, присоединяемые к материнской плате через соединительные планки. Они также имеют один разъем клавиатуры типа DIN5, впаянный на плату в задней части. Гнездо под процессор устанавливается на передней стороне платы. Слоты SIMM и DIMM

находятся в различных местах, хотя почти всегда они расположены в верхней части материнской платы.

Сегодня этот формат плавно сходит со сцены. Часть фирм еще выпускает некоторые свои модели в двух вариантах — Baby AT и более новом — ATX, но это происходит все реже и реже. Не говоря уже просто об удобстве работы — так, чаще всего на Baby AT платах все коннекторы собраны в одном месте, в результате чего либо кабели от коммуникационных портов тянутся практически через всю материнскую плату к задней части корпуса, либо от портов IDE и FDD — к передней. Гнезда для модулей памяти, заезжающие чуть ли не под блок питания, при ограниченности свободы действий внутри весьма небольшого пространства MiniTower — это, мягко говоря, неудобно. Вдобавок неудачно решен вопрос с охлаждением — воздух не поступает напрямую к самой нуждающейся в охлаждении части системы — процессору.

Достоинства:

– хорошо организованная стандартизация, широко развернутое производство позволяют использовать большой выбор корпусов и блоков питания;

– простой и дешевый в производстве дизайн.

Недостатки:

– неэффективное охлаждение современных компонентов, может возникнуть потребность в дополнительных вентиляторах;

– расположение процессора может вызвать проблемы с установкой плат расширения большой длины, например полноразмерных — 330×120 мм.

– разъемы ввода-вывода подключаются к материнской плате посредством большого числа соответствующих кабелей, имеющих несколько стандартов распайки.

3.2. Форм-фактор LPX

Еще до появления ATX первым результатом попыток снизить стоимость ПК стал форм-фактор LPX. Он предназначался для использования в маленьких корпусах для построения дешевых ПК. Задача была решена путем довольно новаторского предложения — введения стойки. Вместо того, чтобы вставлять карты расширения непосредственно в материнскую плату, в этом варианте они помещаются в подключаемую к плате вертикальную стойку параллельно материнской плате. Это позволило заметно уменьшить высоту корпуса, поскольку обычно именно высота карт расширения влияет на этот параметр. Расплатой за компактность стало максимальное количество подключаемых карт — 2–3 штуки. Еще одно нововведение, начавшее широко применяться именно на платах LPX, — это интегрированный на материнскую плату видеочип. Размер корпуса для LPX составляет 9×13", для Mini LPX — 8×10". Разумеется, этот форм-фактор не был предназначен для широкой замены Baby AT в массовом ПК: его предназначение — дешевые системы. Затем появился форм-фактор NLX, который начал вытеснять LPX.

3.3. Форм-фактор ATX

Неудивительно, что форм-фактор ATX во всех его модификациях стал популярным. Спецификация ATX, предложенная Intel еще в 1995 г., нацелена как раз на исправление всех тех недостатков, что выявились со временем у форм-

фактора AT. А решение, по сути, было очень простым — повернуть Baby AT плату на 90 градусов и внести соответствующие поправки в конструкцию. К тому моменту у Intel уже был опыт работы в этой области — форм-фактор LPX. В ATX как раз воплотились лучшие стороны и Baby AT, и LPX: от Baby AT была взята расширяемость, а от LPX — высокая интеграция компонентов.

Контрольные вопросы:

1. Что называют системной платой, и почему ее называют материнской?
2. Какие типы разъемов нашли наибольшее распространение?
3. Какие современные интерфейсы используются в материнских платах?
4. Что называют чипсетами? Их назначение.

Лекция 12. Интерфейсы: понятие, классификация

План:

1. Структурная схема интерфейса.
2. Классификация интерфейсов.

1. Структурная схема интерфейса

Под стандартным интерфейсом понимается совокупность унифицированных аппаратных, программных и конструктивных средств, необходимых для реализации взаимодействия различных функциональных элементов в автоматических системах сбора и обработки информации при условиях, предписанных стандартом и направленных на обеспечение информационной, электрической и конструктивной совместимости указанных элементов (ГОСТ 23633-79, ISO, IEEE). Иногда вместо термина «интерфейс» используется понятие «стык» — место соединения устройств передачи сигналов данных, входящих в системы передачи данных. Структурная схема интерфейса показана на рис. 19, где ИБ — интерфейсный блок.

Интерфейс предназначен для унификации внутрисистемных и межсистемных связей и устройств сопряжения с целью эффективной реализации существующих и перспективных элементов ЭВС.

Рисунок 19. Структурная схема интерфейса

Одна из основных функций интерфейса — это обеспечение им формационной совместимости между элементами системы, которая заключается в согласованности взаимодействий функциональных элементов в соответствии с совокупностью логических условий. Логические условия определяют структуру и состав унифицированного набора шин, набор процедур по реализации взаимодействия

и последовательность их выполнения для различных режимов функционирования, способ кодирования и форматы данных, команд, адресной информации и информации состояния, временные соотношения между управляющими сигналами, ограничения на их форму и взаимодействие.

Условия информационной совместимости определяют объем и сложность схемотехнического оборудования и программного обеспечения, а также основные технико-экономические показатели — пропускную способность, надежность работы интерфейса и объем аппаратных затрат на устройства сопряжения.

Составными физическими элементами связей интерфейса являются электрические цепи, называемые линиями интерфейса. Часть линий, сгруппированных по функциональному назначению, называется шиной, а вся совокупность линий — *магистралью.*

В системе шин интерфейсов условно можно выделить две *магистрали: информационную* и *управления (каналом)*; по *информационной* передаются коды данных, адресов, команд и состояний устройств. Аналогичные наименования присваиваются соответствующим шинам интерфейса.

К наиболее распространенным командам относятся ЧТЕНИЕ, ЗАПИСЬ, КОНЕЦ ПЕРЕДАЧИ, ЗАПУСК. Коды состояния описывают состояния сопрягаемых устройств или формируются в ответ на действия команд. Широко используются такие коды состояний, как ЗАНЯТОСТЬ УСТРОЙСТВА, НАЛИЧИЕ ОШИБКИ, ГОТОВНОСТЬ УСТРОЙСТВА (к приему или передаче информации) и др.

Магистраль управления информационным каналом делится на ряд шин. Шина управления обменом состоит из линий синхронизации передачи информации. В зависимости от принятого принципа обмена (асинхронного, синхронного) число линий может изменяться от одной до трех.

Асинхронная передача происходит при условии подтверждении принимающим информацию устройством готовности к приему и завершается подтверждением о приеме данных. При синхронной передаче темп выдачи и приема данных задается регулярной последовательностью сигналов. Линии шины управления обменом выполняются, как правило, двунаправленными.

Шина передачи управления выполняет операции приоритетного занятия (захвата) магистрали информационного канала. Состав и конфигурация линий этой шины зависят от структуры управления интерфейсом. Различают децентрализованную и централизованную структуры управления. В интерфейсах, предназначенных для объединения только двух устройств (соединение типа «щечка — точка»), эта шина отсутствует.

Шина прерывания применяется в основном в машинных интерфейсах мини- и микро-ЭВМ и программно-модульных систем. Основная ее функция — идентификация устройства, запрашивающего сеанс обмена информацией. Идентификация состоит в определении контроллером (процессором) исходной информации об опрашиваемом устройстве. В качестве информации об устройстве используется адрес источника прерывания текущей программы либо адрес программы обслуживания прерывания (вектор прерывания).

Шина специальных управляющих сигналов включает в себя линии, предназначенные для обеспечения работоспособности и повышения надежности устройства интерфейса. К этим линиям относятся линии питания, контроля источника питания, тактовых импульсов, защиты памяти, общего сброса, контроля информации и т. п.

В соответствии с ГОСТ 26.016-81 структуры связей интерфейсов подразделяют на магистральную, радиальную, цепочечную и смешанную (комбинированную).

Для обеспечения информационной совместимости интерфейс реализует ряд функций:

- селекцию (выбор) информационного канала;
- синхронизацию обмена информацией;
- координацию взаимодействия;
- буферное хранение информации;
- преобразование формы представления информации.

Первые три функции выполняет канал управления, четвертую и пятую — информационный канал. Рассмотрим особенности реализации указанных функций в типовых интерфейсах, используемых в ВС и локальных сетях массового применения.

Управление операциями селекции выполняется централизовано и децентрализованно. При централизованном управлении возможны несколько вариантов реализации селекции. На рис. 20*а* приведена схема временной селекции магистрали на основе генератора временных интервалов контроллера. Магистраль предоставляется каждому устройству через равные промежутки времени, определяемые скоростью работы генератора тактов, а момент занятия магистрали определяется счетчиками, синхронно работающими в каждом из подключаемых устройств. Такое решение обеспечивает правило приоритетного обслуживания «первым пришел — последним обслуживается».

На рис. 20*б* приведена схема пространственной селекции на основе последовательного адресного сканирования источников запроса.

Выбор источника запроса начинается по общему сигналу запроса и выполняется последовательно кодовой адресацией всех подключаемых устройств в соответствии с принятым правилом обслуживания. При обнаружении источника запроса устанавливается сигнал ЗАНЯТО, и дальнейшая выдача контроллером адресов прекращается. По окончании обслуживания данного запроса возобновляется поиск следующего источника. Достоинство этого варианта — гибкость в реализации правил обслуживания, недостаток — низкое быстродействие. Этот вариант широко применяется в стандартных интерфейсах (ГОСТ 26.003-80).

Схема последовательной (цепочечной) селекции показана на рис. 20*в*. Такая селекция наиболее распространена в машинных интерфейсах как наиболее простая и достаточно быстродействующая. Поиск источника запроса начинается по сигналу ЗАПРОС. Идентификация наиболее приоритетного устройства выполняется сигналом ПОДТВЕРЖДЕНИЕ, который последовательно проходит через все устройства. Приоритетным в данном случае будет устройство, наиболее близко расположенное к контроллеру. При поступлении сигнала ПОДТВЕРЖДЕНИЕ в устройство-источник запроса дальнейшее его прохожде-

ние блокируется, и устройством выставляется сигнал ЗАНЯТО. Отличие схемы селекции по выделенным линиям (рис. 20*г*) заключается в том, что общие линии ЗАПРОС и ПОДТВЕРЖДЕНИЕ заменяются системой радиальных линий. Максимальное время занятия информационной магистрали для этой схемы будет меньше, чем для цепочечной структуры, так как сигналы по шинам запроса и подтверждения могут передаваться параллельно. Этот вариант обладает также гибкостью установления дисциплины облуживания, поскольку контроллер с помощью масок может устанавливать произвольный приоритет и порядок опроса. Однако это достигается за счет существенного увеличения числа линий и усложнения аппаратуры. При децентрализованном управлении также имеются варианты реализации селекции (рис. 21). В схемах децентрализованной пространственной селекции наличествуют замкнутые линии запроса и подтверждения. Вариант, показанный на рис. 21*а*, отличается исключением линии ЗАНЯТО и замыканием общей линии ЗАПРОС с линией ПОДТВЕРЖДЕНИЕ. Необходимым условием установления запроса любым устройством является отсутствие входного сигнала подтверждения, при выдаче запроса этот сигнал «дизъюнктивно» формируется на линии и трансформируется в сигнал ПОДТВЕРЖДЕНИЕ, который будет проходить до устройства, выставившего запрос и находящегося наиболее близко по отношению к участку замыкания.

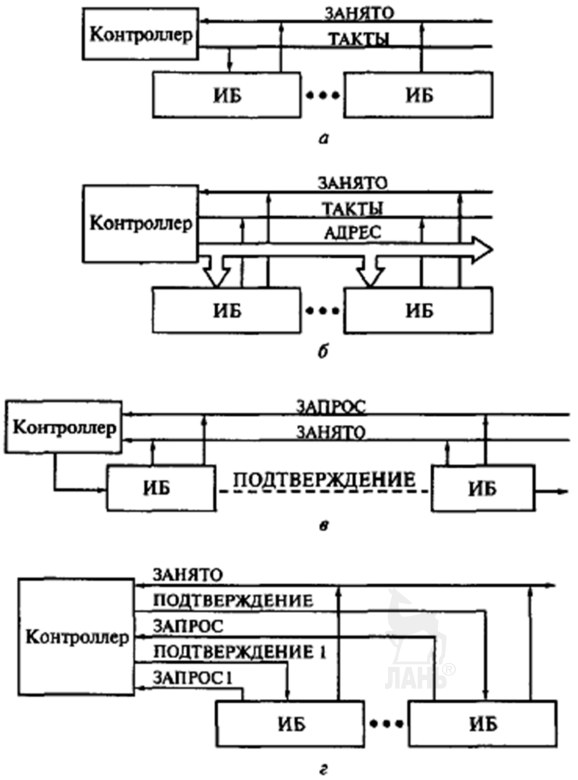


Рисунок 20. Схемы селекции магистрали централизованной структуры

Вариант децентрализованной кольцевой структуры показан на рис. 21*б*. Здесь используется одна линия, определяющая состояние занятости информационного канала по циркуляции в линии маркерного импульса или серии импульсов. Устройство, запрашивающее шину, не пропускает маркер к следующему устройству, и, таким образом, циркуляция импульсов прекращается. Эта структура широко распространена в интерфейсах локальных сетей. Достоинство кольцевой структуры — использование малого количества оборудования и линий связи, основной недостаток — низкая помехоустойчивость.

Способ параллельного адресного сравнения или децентрализованного кодового управления (ДКУ) является одним из перспективных способов селекции для магистральных систем сопряжения (рис. 21*в*). Сущность алгоритма ДКУ заключается в параллельном выделении приоритетного кода запроса с помощью поразрядного сравнения кодов приоритета в асинхронном режиме одновременно во всех устройствах интерфейса, выставивших запрос.

Процессы передачи массива могут быть детерминированными и стохастическими. К детерминированным относятся процессы передачи массива слов фиксированной длины (от одного до нескольких тысяч слов за сеанс связи), к стохастическим — переменной длины. При детерминированных процессах используется синхронный способ сигнализации окончания процесса взаимодействия, при стохастических процессах — асинхронный. Синхронный способ используется редко, причем в основном в интерфейсах, где фиксированная длина массива слов изменяется от 1 до 256 слов. Основное преимущество синхронного способа сигнализации — отсутствие в системе шин линии окончания сеанса связи.

Асинхронный способ сигнализации при передаче массива слов наиболее распространен. Сигналы синхронизации могут выдаваться в произвольный момент времени отправителем (передатчиком) по информационной шине или специально выделенным линиям управляющего канала.

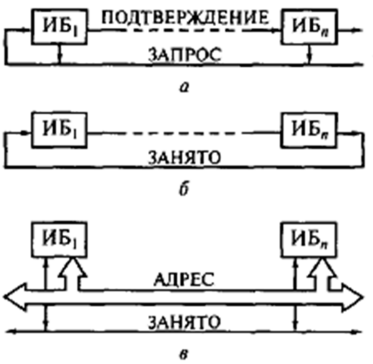


Рисунок 21. Схемы селекции магистрали децентрализованной структуры

2. Классификация интерфейсов

Интерфейсы можно разделить на следующие основные классы (рис. 22):

- машинные или системные;
- периферийного оборудования;

- мультиплексорных систем;
- распределенных вычислительных систем;
- локальных сетей.

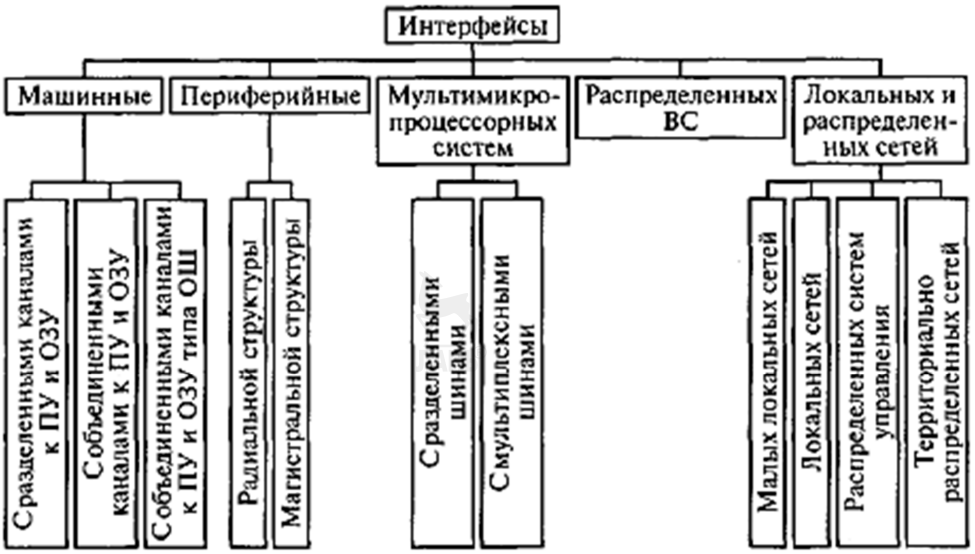


Рисунок 22. Классификация интерфейсов

Машинные интерфейсы предназначены для организации связи между составными компонентами ЭВМ и систем. В свою очередь, они разделяются на три группы. Первая — интерфейсы ввода-вывода в (из) ЭВМ с разделенными информационными каналами к УВВ и к ОЗУ. Вторая группа — интерфейсы ЭВМ с объединенным информационным каналом к УВВ и ОЗУ типа «общая шина» (ОШ). Некоторые классы персональных ЭВМ, мини- и микро-ЭВМ имеют такую архитектуру. К третьей группе относятся интерфейсы одноплатных ЭВМ с объединенным информационным каналом к УВВ и ОЗУ типа ОШ, ориентированные на внутриплатное и внутрисхемное применение. Такие интерфейсы предназначены для организации сопряжения между составными компонентами микропроцессорных комплектов БИС и составных функциональных узлов СБИС мини-ЭВМ (ОШ) и микро-ЭВМ (Q-шина или GBUS).

Контрольные вопросы:

1. Что такое интерфейс?
2. Как классифицируют интерфейсы?
3. Какие магистрали выделяют в системе шин интерфейсов?

Тема 3. Типы вычислительных систем и их архитектурные особенности

Лекция 13. Типы вычислительных систем и их архитектурные особенности

План:

1. Вычислительная система.
2. Общая классификация вычислительных систем.
3. Классификация систем параллельной обработки данных по М. Флинну.
4. Примеры вычислительных систем.

1. Вычислительная система

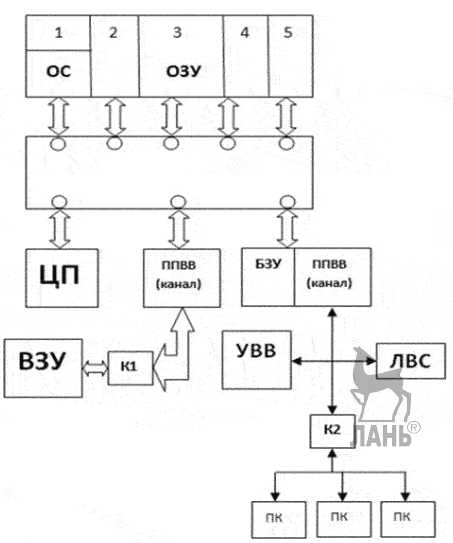
Вид деятельности, в котором задействован человек, как правило, обусловлен решением каких-либо задач. Вычислительные системы, так же как и человек, выполняют или решают задачи в разных областях. В то же время есть люди, которые работают в нескольких трудовых областях одновременно, либо группа людей работает над решением одной и той же задачи. Так, и ВС могут решать одну задачу, две задачи, либо группа объединенных ВС решает одну и ту же задачу. В связи с этим представляется возможным дать классификацию ВС.

Система — это совокупность элементов, которые находятся между собой в определенных отношениях и связях, и которые образуют определенную целостность, единство какого-либо явления или предмета исследования (от *греч.* system — соединенная из частей). Основой этого направления явились труды Л. фон Бертоланфи в области биологии (1937).

Итак, можно сформулировать определение, *вычислительная система* — это совокупность взаимосвязанных и взаимодействующих компьютеров (процессоров), периферийного оборудования и программного обеспечения, предназначенных для подготовки и решения задач пользователя.

Иногда под ВС понимают совокупность технических средств ЭВМ, в которую входит не менее двух процессоров, связанных общностью управления и использования общесистемных ресурсов (память, периферийные устройства, программное обеспечение и т. п.) (рис. 23).

В верней части блок-схемы показано ОЗУ, разделенное на отдельные блоки. Каждый блок имеет свое местное устройство управления оперативной памяти. Обмен между ОЗУ и быстрой ВЗУ происходит сравнительно большими объемами информации достаточно быстро и оперативно. Программирование упрощается, так как пользователю дают весь объем памяти (ОЗУ + ВЗУ) для прямой поячеечной адресации. Такие адреса носят названия виртуальных. Физическими адресами данных являются только адреса ОЗУ. В первом блоке ОЗУ размещена резидентная часть ОС (часть блоков ОС, которые часто используются). Под системой коммутации располагаются процессоры: центральный процессор (ЦП) и периферийные процессоры ввода-вывода (ППВВ), выполняющие функции обмена между ОЗУ и периферийными устройствами. Простейшие ППВВ, по терминологии для отечественных ЭВМ, также называют каналами.



ОС – операционная система
ОЗУ – оперативно запоминающее устройство
ЦП – центральный процессор
ППВВ – периферийные процессоры ввода-вывода
БЗУ – быстродействующее запоминающее устройство
ВЗУ – внешнее запоминающее устройство
УВВ – устройства ввода-вывода
ЛВС – локально вычислительная сеть
К1, К2 – контроллеры
ПК – др. персональные компьютеры

Рисунок 23. Общая схема ВС

Левый канал подключен к ВЗУ. Поскольку скорость передачи данных высокая и, следовательно, время обмена короткое, канал поддерживает связь ОЗУ с выбранным ВЗУ все время, пока не закончится обмен. Поэтому этот канал называется селекторным, т. е. канал выбирает устройство на все время обмена. К правому каналу подключены более медленные периферийные устройства ввода-вывода (УВВ). Чтобы полностью использовать пропускную способность канала, его снабжают своим быстродействующим ЗУ и системой переключения с одного УВВ на другое. Этот канал работает в 2 такта: во-первых, он накапливает данные из УВВ в ячейках памяти, закрепленных за этим УВВ, переключаясь, по мере готовности передать или принять данные, с одного УВВ на другое, а во-вторых, обменивается более крупными порциями данных между своим буферным ЗУ и ОЗУ. Этот тип канала за способность к быстрому переключению получил название мультиплексного.

2. Общая классификация вычислительных систем

Общая классификация вычислительных систем:

<u>*По назначению:*</u>

- *универсальные* предназначаются для решения широкого класса задач (от математических расчетов до обработки мультимедиа), т. е. такие ВС должны обслуживать программные приложения, разработанные для самых разных и далеко отстоящих друг от друга направлений научных исследований;
- *специализированные* ориентированы на решение узкого класса задач.

<u>*По типу:*</u>

- *многопроцессорные*. В качестве общего ресурса они имеют общую оперативную память. Параллельная работа процессоров и использование общей оперативной памяти обеспечиваются под управлением общей

операционной системы. Это позволяет в случае отказа одного из процессоров, перераспределить нагрузку между оставшимися процессорами (рис. 24);

- *многомашинные*.

Возможны два варианта:

1) обе машины решают одну и ту же задачу и периодически сверяют результаты решения;
2) обе машины работают параллельно, но обрабатывают собственные потоки заданий (рис. 25).

Основной недостаток многомашинной ВС — достаточно в каждой ЭВМ выйти из строя по одному устройству (даже разных типов), как вся система становится неработоспособной.

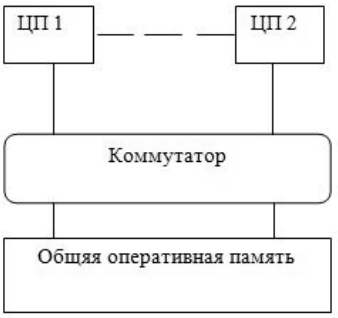


Рисунок 24. Многопроцессорная ВС

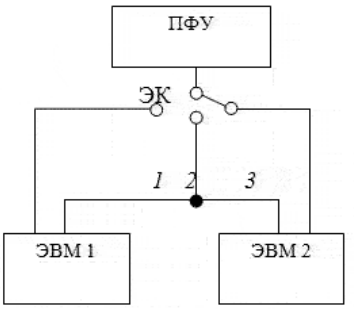


Рисунок 25. Многомашинная ВС

<u>*По характеру устройств:*</u>

- *однородные системы* содержат несколько однотипных ЭВМ (или процессоров). Основной недостаток однородных ВС — неполная загруженность отдельных ЭВМ (процессоров) во время ее работы. В связи с этим недостатком применяются неоднородные ВС;
- *неоднородные.* Неоднородные системы содержат разнотипные ЭВМ (или процессоры).

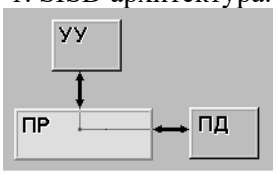
<u>*По управлению:*</u>

- *централизованные*. Функции управления сосредоточены в главной ЭВМ (процессор). Ее задачами являются распределение нагрузки между элементами, выделение ресурсов, контроль состояния ресурсов, координация взаимодействия;
- *децентрализованные*. Функции управления распределены между ее элементами, т. е. каждый процессор или ЭВМ действуют автономно, решая свои задачи;
- *смешанные*. Совмещаются процедуры централизованного и децентрализованного управления. Т. е. ВС разбивается на группы взаимодействующих ЭВМ (или процессоров), где в каждой группе осуществляется централизованное управление, а между группами — децентрализованное.

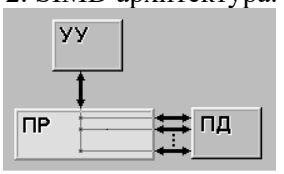
3. Классификация систем параллельной обработки данных по М. Флинну

Классификация базируется на понятиях двух потоков: команд и данных. На основе числа этих потоков выделяется четыре класса архитектур:

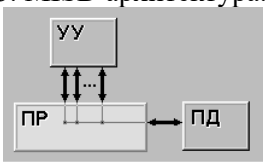
1. SISD архитектура.



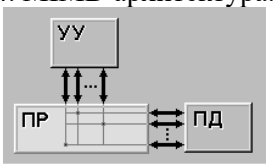
2. SIMD архитектура.



3. MISD архитектура.



4. MIMD архитектура.



Обозначения: УУ — управляющее устройство (организует поток команд), ПР — процессор, ПД — поток данных.

4. Примеры вычислительных систем

Появление любого нового направления в вычислительной технике определяется требованиями компьютерного рынка. Поэтому у разработчиков компьютеров нет одной единственной цели. Большая универсальная вычислительная машина (мейнфрейм) или суперкомпьютер стоят дорого. Для достижения поставленных целей при проектировании высокопроизводительных конструкций приходится игнорировать стоимостные характеристики.

Суперкомпьютеры фирмы Cray Research и высокопроизводительные мейнфреймы компании IBM относятся именно к этой категории компьютеров. Другим крайним примером может служить конструкция, где производительность принесена в жертву для достижения низкой стоимости. К этому направлению относятся персональные компьютеры различных клонов IBM PC. Между этими двумя крайними направлениями находятся конструкции, основанные на отношении стоимость/производительность, в которых разработчики находят баланс между стоимостными параметрами и производительностью. Типичными примерами такого рода компьютеров являются мини-компьютеры и рабочие станции.

Многопроцессорная ВС с общим кэшем и общей памятью

Общий вид такой многопроцессорной ВС представлен на рис. 26.

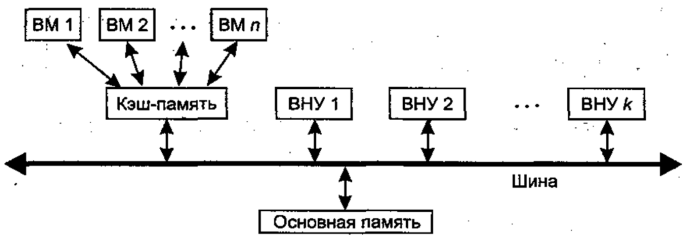


Рисунок 26. Многопроцессорная ВС с общим кэшем и общей памятью

Здесь ВМ — вычислительный модуль (процессор), всего n штук, ВНУ — внешние устройства, всего k штук. В качестве основной памяти может использоваться память с горизонтальным расслоением или же для повышения надежности — блочно-модульная память.

Основными проблемами в данной архитектуре являются следующие: скорость работы кэш-памяти, блокировка при совместном доступе нескольких процессоров на одновременную запись или запись/чтение данных по одному и тому же адресу. Кэш-память должна работать с тактовой частотой, в n раз большей тактовой частоты процессора, так как она должна обеспечивать данными процессоры с частотой один раз в такт. В противном случае каждый процессор должен быть блокирован до момента получения (или записи) им требуемых данных, что снижает производительность многопроцессорной ВС.

Многопроцессорная ВС с раздельными кэшами и общей памятью

Данная модель мультипроцессорной системы наиболее распространена в настоящее время. Структурная схема многопроцессорной ВС с раздельными кэшами и общей памятью представлена на рис. 27.

Каждый вычислительный модуль (ВМ) имеет собственную локальную кэш-память, имеется общая разделяемая основная память, все ВМ подсоединены к основной памяти посредством шины. К шине подключены также внешние устройства. Все действия с использованием транзакций шины, производимые ВМ и внешними устройствами, с копиями строк, как в каждой кэш-памяти, так и в основной памяти, доступны для отслеживания всем ВМ. Это является следствием того, что в каждый момент на шине передает только один, а воспринимают все абоненты, подключенные к шине.

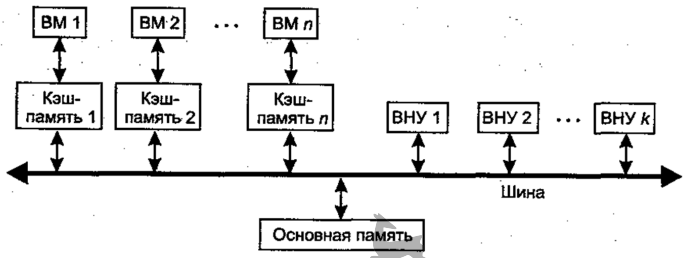


Рисунок 27. Многопроцессорная ВС с раздельными кэшами и общей памятью

По сравнению с многопроцессорными ВС с общим кэшем и общей памятью в данной системе наряду с проблемой пропускной способности шины возникает еще и проблема когерентности кэш-памяти.

В такой вычислительной системе кэши могут содержать как разделяемые, так и частные данные. Частные данные — это данные, которые используются одним процессором, в то время как разделяемые данные используются многими процессорами, по существу обеспечивая обмен между ними. Когда кэшируется элемент частных данных, их значение переносится в кэш для сокращения среднего времени доступа, а также требуемой полосы пропускания. Поскольку никакой другой процессор не использует эти данные, этот процесс идентичен процессу для однопроцессорной машины с кэш-памятью. Если кэшируются разделяемые данные, то разделяемое значение реплицируется и может содер-

жаться в нескольких кэшах. Кроме сокращения задержки доступа и требуемой полосы пропускания такая репликация данных способствует также общему сокращению количества обменов. Таким образом, кэширование разделяемых данных вызывает проблему когерентности кэш-памяти.

Коммуникационные среды

В самом общем смысле архитектуру компьютера можно определить как способ соединения процессоров между собой, с памятью и с внешними устройствами. Реализация этого соединения может идти различными путями. Конкретная реализация соединений такого рода называется *коммуникационной средой компьютера.* Одна из самых простых реализаций — это использование *общей шины,* к которой подключаются как процессоры, так и память. Сама шина состоит из определенного числа линий связи, необходимых для передачи адресов, данных и управляющих сигналов между процессором и памятью. Этот способ реализован в SMP-системах. Основным недостатком таких систем является плохая масштабируемость. Увеличение, даже незначительное, числа устройств на шине вызывает заметные задержки при обмене с памятью и катастрофическое падение производительности системы в целом. Необходимы другие подходы для построения коммуникационной среды, и одним из них является разделение памяти на независимые модули и обеспечение возможности доступа разных процессоров к различным модулям одновременно посредством использования различного рода коммутаторов.

При этом возможны различные конфигурации получающихся систем связи. Так, в компьютерах семейства Cray T3D/T3E все процессоры были объединены специальными высокоскоростными каналами в трехмерный тор, в котором каждый вычислительный узел имел непосредственные связи с шестью соседями. В компьютерах IBM SP/2 взаимодействие процессоров происходит через иерархическую систему коммутаторов, также обеспечивающую возможность соединения каждого процессора с любым другим. Эти оригинальные уникальные решения значительно увеличивают цену компьютеров.

Существенно более простым и более дешевым оказалось использование системы связи на базе Ethernet, разработанной фирмой Xerox. Первоначально использовалась обычная 10-мегабитная сеть, затем стали применять Fast Ethernet, а в последнее время — Gigabit Ethernet. Однако для Fast Ethernet характерна большая латентность (задержка в передаче данных), оцениваемая в 160–180 мкс, а Gigabit Ethernet отличается высокой стоимостью. Поэтому при создании многопроцессорных вычислительных систем часто предпочтение отдается технологиям SCI, Myrinet или Raceway.

Примеры построения коммуникационных сред на основе масштабируемого когерентного интерфейса SCI

SCI принят как стандарт в 1992 г. Предназначен для достижения высоких скоростей передачи с малым временем задержки, при этом обеспечивая масштабируемую архитектуру, позволяющую строить системы, состоящие из множества блоков. Представляет собой комбинацию шины и локальной сети, обеспечивает реализацию когерентности кэш-памяти, размещаемой в узле SCI, посредством механизма распределенных директорий, который улучшает производительность, скрывая затраты на доступ к удаленным данным в модели с рас-

пределенной разделяемой памятью. Производительность передачи данных обычно находится в пределах от 200 до 1000 Мбайт/с на расстояниях десятков метров с использованием электрических кабелей и километров с использованием оптоволокна. SCI уменьшает время межузловых коммуникаций по сравнению с традиционными схемами передачи данных в сетях путем устранения обращений к программным уровням — операционной системе и библиотекам времени выполнения; коммуникации представляются как часть простой операции загрузки данных процессором.

Обычно обращение к данным, физически расположенным в памяти другого вычислительного узла и не находящимся в кэше, приводит к формированию запроса на удаленный узел для получения необходимых данных, которые в течение нескольких микросекунд доставляются в локальный кэш, и выполнение программы продолжается. Старый подход требовал формирования пакетов на программном уровне с последующей передачей их аппаратному обеспечению. Точно так же происходил и прием, в результате чего задержки были в сотни раз больше, чем у SCI. Однако для совместимости SCI имеет возможность переносить пакеты других протоколов. Другое преимущество SCI — использование простых протоколов типа RISC, которые обеспечивают большую пропускную способность. Узлы с адаптерами SCI могут использовать для соединения коммутаторы или же соединяться в кольцо. Обычно каждый узел оказывается включенным в два кольца (рис. 28).

Данная технология оптимизирована для работы с динамическим трафиком, однако может быть менее эффективна при работе с большими блоками данных. Протокол SCI достаточно сложен, он содержит большие возможности по управлению трафиком, но использование этих возможностей предполагает наличие развитого программного обеспечения. На коммуникационной технологии SCI основана система связи гиперузлов CTI (Convex Torroidal Interconnect) в системах HP/Convex Exemplar X-class, кроме того, на ней построены кластерные системы SCALI Computer, системы семейства hpcLine компании Siemens, а также cc-NUMA-сервера Data General и Sequent.

Традиционная область применения SCI — это коммуникационные среды многопроцессорных систем. На основе этой технологии построены, в частности, компьютеры серии hpcLine от Siemens или модульные серверы NUMA-Q от IBM, ранее известные как Sequent.

Модульные SCI-коммутаторы Dolphin позволяют потребителям строить масштабируемые, кластерные решения класса предприятия на платформах Windows NT/2000/XP, Linux, Solaris, VxWorks, LynuxWorks и NetWare с использованием стандартизованного оборудования и программного обеспечения.

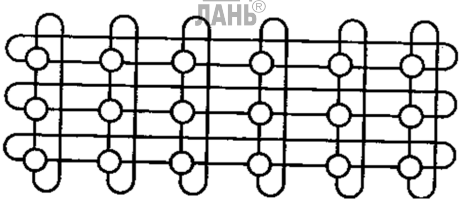


Рисунок 28. Матрица узлов кластера на основе сети SCI

Таблица 29. Основные характеристики SCI (Scalable Coherent Interface)

Производители оборудования	Dolphin Interconnect Solutions и др.
Показатели производительности	Для продуктов Dolphin: пиковая пропускная способность 667 Мбайт/с. Аппаратная латентность — 1,46 мкс
Программная поддержка	Драйверы для Linux, Windows NT, Solaris. ScaMPI — реализация MPI компании Scali Computer для систем на базе SCI. SISCI API — интерфейс программирования нижнего уровня
Комментарии	SCI (ANSI/IEEE 1596-1992) — хорошо стандартизированная технология. Кроме стандартной сетевой среды, SCI поддерживает построение систем с разделяемой памятью и с когерентностью кэшей. На коммуникационной технологии SCI основаны кластерные системы компании SCALI Computer, системы семейства hpcLine компании Siemens, а также cc-NUMA-сервера Data General и Sequent. Технология SCI использовалась для связи гиперузлов в системах HP/Convex Exemplar X-class

Коммуникационная среда Myrinet

Сетевую технологию Myrinet представляет компания Myricom, которая впервые предложила свою коммуникационную технологию в 1994 г., а на сегодня имеет уже более 1000 инсталляций по всему миру. Технология Myrinet основана на использовании многопортовых коммутаторов при ограниченных несколькими метрами длинах связей узлов с портами коммутатора. Узлы в Myrinet соединяются друг с другом через коммутатор (до 16 портов). Myrinet может одновременно передавать несколько пакетов, каждый из которых идет со скоростью, близкой к 2 Гбит/с. В отличие от некоммутированных сетей Ethernet и FDDI, которые разделяют общую среду передачи, совокупная пропускная способность сети Myrinet возрастает с увеличением количества машин. На сегодняшний день Myrinet чаще всего используют как локальную сеть (LAN) сравнительно небольшого физического размера, связывая вместе компьютеры внутри комнаты или здания. Из-за своей высокой скорости, малого времени задержки, прямой коммутации и умеренной стоимости, Myrinet особенно популярен для объединения компьютеров в кластеры. Myrinet также используется как системная сеть (System Area Network, SAN), которая может объединять компьютеры в кластер внутри стойки с той же производительностью, но с более низкой стоимостью, чем Myrinet LAN. Пакеты Myrinet могут иметь любую длину. Таким образом, они могут включать в себя другие типы пакетов, включая IP-пакеты. Соединение вычислительных узлов с адаптерами Myrinet в сеть происходит с помощью коммутаторов, которые имеют сейчас 4, 8, 12 или 16 портов. В коммутаторах используется передача пакетов путем установления соединения на время передачи, для маршрутизации сообщений используется алгоритм прокладки пути (wormhole, «червоточина»). Коммутаторы, как и сетевые адаптеры, построены на специализированных микропроцессорах LANai фирмы Myricom.

На физическом уровне каналы ввода-вывода (линки) Myrinet состоят из девяти линий: 8 битов предназначены для передачи информации, интерпретируемой в зависимости от состояния девятого бита, который используется как байт данных или управляющий символ; при этом на каждом линке обеспечиваются управление потоком и контроль ошибок. Среда Myrinet выгодно отли-

чается от многих других сред передачи, в частности SCI, простотой концепции и аппаратной реализацией протоколов. Она содержит ограниченный набор средств управления трафиком, использующих приливно-отливный буфер, управляющие символы и таймерные интервалы. Myrinet является открытым стандартом, компания Myricom предлагает широкий выбор сетевого оборудования по сравнительно невысоким ценам. Технология Myrinet дает большие возможности масштабирования сети и в настоящее время широко используется при построении высокопроизводительных вычислительных кластеров.

Коммуникационная среда Raceway

Коммуникационная среда Raceway обеспечивает пропускную способность на уровне 1 Гбайт/с; среда передачи создается с помощью коммутатора фирмы Cypress и соответствующих сетевых адаптеров. Коммутатор имеет шесть портов, пропускная способность каждого составляет 160 Мбайт/с. Порт состоит из 32 сигнальных линий данных и пяти управляющих линий. При начале транзакции среда Raceway предварительно устанавливает соединение, задержка в коммутаторе при установлении соединения составляет примерно 125 нс. Структуры вычислительных систем, создаваемых с помощью Raceway, аналогичны тем, которые применяются в случае использования сети Myrinet или коммутаторов и адаптеров SCI. Разница заключается в количестве портов коммутаторов, форматах передаваемых пакетов и в протоколах. Коммуникационная среда Raceway принята в качестве стандарта (ANSI/VINA 5-1994).

Коммуникационные среды на базе транспьютероподобных процессоров

Транспьютер — это микроэлектронный прибор, объединяющий на одном кристалле микропроцессор, быструю память, интерфейс внешней памяти и линки, предназначенные для подключения аналогичных приборов. Прибор спроектирован таким образом, чтобы максимально облегчить построение параллельных вычислительных систем. При соединении транспьютерных элементов между собой требуется минимальное число дополнительных интегральных схем. Связь между транспьютерами осуществляется путем непосредственного соединения линка одного прибора с линком другого. Это позволяет создавать сети с различными топологиями с большим числом элементов.

Транспьютер представляет собой микропроцессор, в состав которого входят:

1) ЦПУ с сокращенным набором команд (RISC);

2) 64-разрядный сопроцессор (FPU) плавающей арифметики с высокой пиковой производительностью, работающий параллельно с ЦПУ;

3) внутрикристальное ОЗУ;

4) 32-разрядная шина памяти;

5) четыре последовательные двунаправленные линии связи (*link*), обеспечивающие взаимодействие транспьютера с внешним миром, работающие параллельно;

6) таймер;

7) системные управляющие сигналы *«инициализация», «анализ», «ошибка»,* управляющие загрузкой и анализом состояния транспьютера, сигнализирующие об ошибках;

8) интерфейс внешних событий (*event*), обеспечивающий асинхронную связь внутреннего процесса и внешнего события.

Транспьютеры размещаются на транспьютерных модулях (TRAM или TPAM) — дочерних платах, содержащих транспьютер, ОЗУ, переключатели для выбора режимов и интерфейс, включающий гнезда/штекеры питания, четыре линии связи, линии внешних событий и системных управляющих сигналов. В зависимости от состава TPAM может иметь разные физические размеры, которые стандартизованы и пронумерованы. Так, наименьший по размеру TPAM имеет номер 1, следующий — 2 и т. д.

TPAMы размещаются на объединительных платах, которые либо непосредственно включаются в некоторый компьютер, либо соединенные вместе составляют сетевой компьютер. Изначально транспьютеры производила фирма Inmos. В настоящее время ряд зарубежных фирм пошел по пути создания транспьютероподобных микропроцессоров, имеющих гораздо большую вычислительную мощность, чем транспьютер фирмы Inmos (например, фирма Texas Instruments выпустила сигнальный процессор TMS320C40 с производительностью 50 Мфлопс).

Возникновение высокопроизводительных параллельных вычислительных систем на базе транспьютеров и транспьютероподобных микропроцессоров в свое время потребовало создания новых эффективных операционных систем.

Коммутаторы для многопроцессорных вычислительных систем

Коммуникационные среды вычислительных систем (ВС) состоят из адаптеров вычислительных модулей (ВМ) и коммутаторов, обеспечивающих соединения между ними.

Используются как простые коммутаторы, так и составные, компонуемые из набора простых. Простые коммутаторы могут соединять лишь малое число ВМ в силу физических ограничений, однако обеспечивают при этом минимальную задержку при установлении соединения. Составные коммутаторы, обычно строящиеся из простых в виде многокаскадных схем с помощью линий «точка — точка», преодолевают ограничение на малое количество соединений, однако увеличивают и задержки.

Простые коммутаторы

Известно два основных типа простых коммутаторов:

- с временным разделением;
- с пространственным разделением.

Простые коммутаторы с временным разделением. Простые коммутаторы с временным разделением используются в системах SMP, например Power Challenge от SGI. Простые коммутаторы с временным разделением называются также шинами или шинными структурами. Все устройства подключаются к общей информационной магистрали, используемой для передачи информации между ними. Обычно шина является пассивным элементом, управление передачами осуществляются передающими и принимающими устройствами. Достоинства: простота управления и высокое быстродействие. Недостатки: малое количество входов и выходов.

Передающее устройство сначала получает доступ к шине, далее пытается установить контакт с устройством-адресатом и определить его способность к приему данных. Принимающее устройство распознает свой адрес на шине и отвечает на запрос передающего. Далее передающее устройство сообщает, какие

действия должно произвести принимающее устройство в ходе взаимодействия. После этого происходит передача данных.

Так как шина является общим ресурсом, за доступ к которому соревнуются подключенные к ней устройства, то необходимы методы управления предоставлением доступа устройств к шине. Возможно использование центрального устройства для управления доступом к шине, однако это уменьшает масштабируемость и гибкость системы.

В последнее время применяются шины следующих стандартов:

– ISA — Industry Standard Architecture;
– EISA — Extended ISA;
– VESA — Video Electronics Standards Association;
– PCI — Peripheral Computer Interconnect;
– VME — Versabus Module Europe;
– I2C — Inter Integrated Circuit;
– AGP — Accelerated Graphic Port.

Шины используются также в *мезонинной технологии,* где на большой плате устанавливается один или несколько шинных разъемов для установки меньших плат, так называемых *мезонинов.*

Простые коммутаторы с пространственным разделением. Простые коммутаторы с пространственным разделением (Gigaplane) используются, например, в семействе Sun Ultra Enterprise.

Простые коммутаторы с пространственным разделением позволяют одновременно соединять любой вход с любым одним выходом (ординарные) или несколькими выходами (неординарные). Такие коммутаторы представляют собой совокупность мультиплексоров, количество которых соответствует количеству выходов коммутатора, при этом каждый вход коммутатора должен быть заведен на все мультиплексоры. Структура этих коммутаторов показана на рис. 29. Достоинства: возможность одновременного контакта со всеми устройствами; минимальная задержка. Недостатки: высокая сложность порядка $n \times m$, где n — количество входов, m — количество выходов; сложность обеспечения надежности.

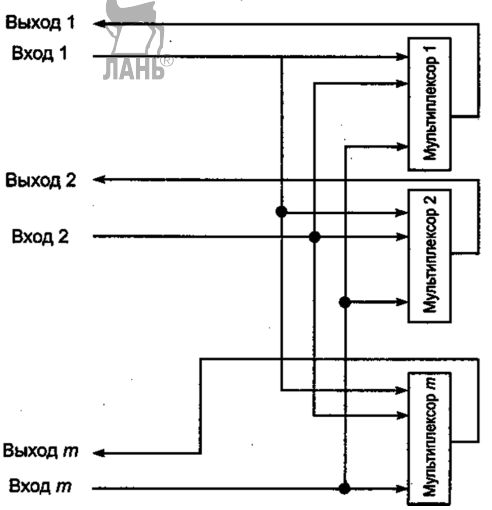


Рисунок 29. Простой коммутатор с пространственным разделением

Составные коммутаторы

Простые коммутаторы имеют ограничения на число входов и выходов, а также могут требовать большого количества оборудования при увеличении этого числа (в случае пространственных коммутаторов). Поэтому для построения коммутаторов с большим количеством входов и выходов используют совокупность простых коммутаторов, объединенных с помощью линий «точка — точка».

Составные коммутаторы имеют задержку, пропорциональную количеству простых коммутаторов, через которые проходит сигнал от входа до выхода, т. е. числу каскадов. Однако объем оборудования составного коммутатора меньше, чем простого с тем же количеством входов и выходов.

Чаще всего составные коммутаторы строятся из прямоугольных коммутаторов 2×2 с двумя входами и выходами. Они имеют два состояния: прямое пропускание входов на соответствующие выходы и перекрестное пропускание. Коммутатор 2×2 состоит из собственно блока коммутации данных и блока управления. Блок управления в зависимости от поступающих на него управляющих сигналов определяет, какой тип соединения следует осуществить в блоке коммутации: прямой или перекрестный. При этом если оба входа хотят соединиться с одним выходом, то коммутатор разрешает конфликт и связывает с данным выходом только один вход, а запрос на соединение со стороны второго блокируется или отвергается.

Коммутатор Клоза (рис. 30). Коммутатор Клоза может быть построен в качестве альтернативы для прямоугольного коммутатора с (m×d) входами и (m×d) выходами. Он формируется из трех каскадов коммутаторов: т коммутаторов (d×d) во входном каскаде, m коммутаторов (d×d) в выходном и d промежуточных коммутаторов (m×n).

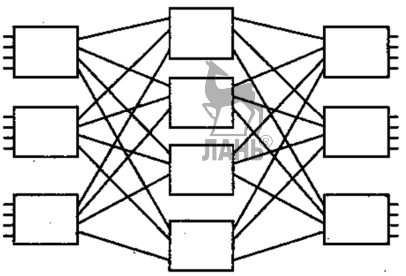


Рисунок 30. Коммутатор Клоза 3×4

Данный тип составных коммутаторов позволяет соединять любой вход с любым выходом, однако при установленных соединениях добавление нового соединения может потребовать разрыва и переустановления всех соединений.

Баньян-сети

Коммутаторы этого типа строятся на базе прямоугольных коммутаторов таким образом, что существует только один путь от каждого входа к каждому выходу.

Структура баньяновой сети, выполненная в виде узла на 16 входов и выходов, состоит из простых коммутирующих элементов, соединённых друг с другом (рис. 31). Через последовательности этих элементов передаются блоки данных. Изображенная структура имеет четыре каскада (1–4) коммутирующих элементов. Каждый передаваемый блок данных имеет в заголовке адрес, разрядность которого равна числу элементов баньяновой сети.

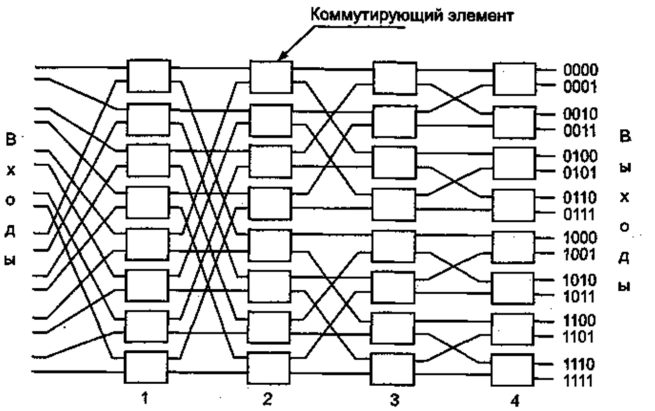


Рисунок 31. Баньян-сеть

Распределенные составные коммутаторы

В распределенных вычислительных системах ресурсы разделяются между задачами, каждая из которых исполняется на своем подмножестве процессоров. В связи с этим возникает понятие близости процессоров, которая является важной для активно взаимодействующих процессоров. Обычно близость процессоров выражается в различной каскадности соединений, различных расстояниях между ними.

Один из вариантов создания составных коммутаторов заключается в объединении прямоугольных коммутаторов таким образом, что один вход и один выход каждого составляющего коммутатора служат входом и выходом составного коммутатора. К каждому внутреннему коммутатору подсоединяются процессор и память, образуя вычислительный модуль с v каналами для соединения с другими вычислительными модулями. Свободные v выходов и v входов каждого вычислительного модуля соединяются линиями «точка — точка» с входами и выходами других коммутаторов, образуя граф межмодульных связей.

Наиболее эффективным графом межмодульных связей с точки зрения организации обмена данными между вычислительными модулями является *полный граф.* В этом случае между каждой парой вычислительных модулей существует прямое соединение. При этом возможны одновременные соединения между произвольными вычислительными модулями.

Однако обычно создать полный граф межмодульных связей невозможно по различным причинам. Обмен данными приходится производить через цепочки транзитных модулей. Из-за этого увеличиваются задержки и ограничивается возможность установления одновременных соединений. Таким образом,

эффективный граф межмодульных связей должен минимизировать время межмодульных обменов и максимизировать количество одновременно активизированных соединений. Кроме того, на выбор графа межмодульных связей влияет учет отказов и восстановлений вычислительных модулей и линий связи.

Матричный коммутатор состоит из множества одинаковых коммутирующих элементов (рис. 32). На два входа каждого из них подаются блоки данных, которые в зависимости от адресов назначения необходимо передать на один из выходов элемента. Число коммуникационных элементов как минимум равно произведению числа входов на число выходов коммутатора. Поэтому, матричные коммутаторы применяются в интегральной коммутации, число входов/выходов в которых невелико.

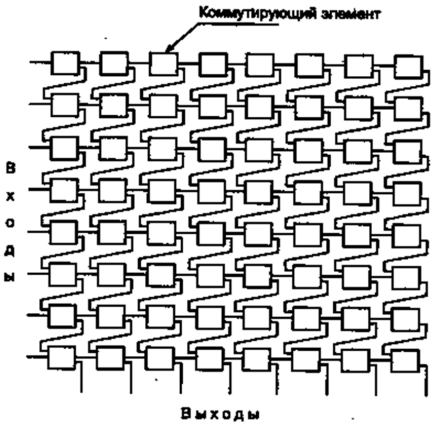


Рисунок 32. Матричный коммутатор

Кластерные и массивно-параллельные системы различных производителей

Развитие сетевых технологий привело к появлению недорогих, но эффективных коммуникационных решений. Это и предопределило появление кластерных вычислительных систем, фактически являющихся одним из направлений развития компьютеров с массовым параллелизмом. Классические суперкомпьютеры, использующие специализированные процессоры таких фирм, как, например, Cray, NEC (векторно-параллельные или массивно-параллельные), обычно недешевы, поэтому и стоимость подобных систем не сравнима со стоимостью систем, находящихся в массовом производстве.

Вычислительные системы, создаваемые из массово выпускаемых компонентов, стали притягательной альтернативой традиционным суперкомпьютерным системам. При выполнении многих прикладных задач такие ВС, даже с небольшим или средним (до 128–256) числом вычислительных модулей, показывают производительность, не уступающую или даже превосходящую производительность традиционных суперкомпьютеров как с распределенной, так и с разделяемой памятью. Наряду с этим, эти ВС обладают рядом преимуществ, среди которых: более низкая стоимость, короткий цикл разработки и возможность оперативно использовать наиболее эффективные вычислительные и ком-

муникационные компоненты из имеющихся на рынке во время создания системы. Поэтому неудивительно, что ведущие фирмы-разработчики высокопроизводительной техники приступили к созданию кластерных систем.

Отечественные суперкомпьютеры семейства МВС

Создание и использование суперкомпьютеров в мире относится к факторам стратегического значения и входит в первую десятку приоритетов «ключевых» технологий развитых стран. В настоящее время в США эксплуатируются суперкомпьютеры с производительностью более триллиона операций в секунду, что дает возможность ускорения работ в области фундаментальных исследований, решения прикладных задач народнохозяйственного и оборонного комплекса. США, Япония и страны Западной Европы резко ограничивают возможность приобретения Россией мощных вычислительных систем. Кроме того, зарубежные вычислительные системы такого уровня непомерно дороги, и их эксплуатационное освоение затруднено.

Проблема создания конкретных суперкомпьютеров требуемого уровня мощности для наиболее крупных российских вычислительных центров решается на основе сбалансированного целенаправленного сочетания закупок новейших комплектующих изделий, создания на этой основе отечественных суперкомпьютерных систем, их интеграции в информационно-вычислительные сети и необходимых усилий в области применения, т. е. в разработке прикладных программ и методов математического моделирования. Эта концепция реализована в мультипроцессорной вычислительной системе МВС-100, которая создана в кооперации научно-исследовательских институтов Российской академии наук и промышленности (головные организации НИИ «Квант» РАСУ и ИПМ РАН). Система построена на основе микропроцессоров с быстродействием порядка 100 миллионов операций в секунду, межпроцессорное взаимодействие осуществляется с помощью транспьютеров. Установки МВС-100 с суммарной производительностью более 50 миллиардов операций в секунду в течение нескольких лет успешно эксплуатируются в вычислительных центрах РАН (в Москве, Екатеринбурге, Новосибирске, Владивостоке) и в отраслевых ВЦ. С их помощью решены сложные прикладные задачи качественно нового уровня из области аэродинамики для самолетостроения и создания реактивных двигателей, ядерной физики, управления динамическими системами, распознавания изображений при навигации движущихся объектов, сейсмогеологоразведки, нефтедобычи, метеорологии, биоинженерии и др. Показана возможность эффективного распараллеливания вычислений и обработки данных.

В сложившейся кооперации проведены работы по созданию системы нового поколения МВС-1000 на микропроцессорах Alpha с технологическими нормами 0,35 мкм и быстродействием до 1–2 млрд оп./с. В течение 1998 г. на эксплуатируемых установках этого типа отрабатывалось программное обеспечение и решен ряд новых сложных реальных вычислительных задач. К настоящему времени введена в действие система с производительностью 200 млрд оп./с для Межведомственного суперкомпьютерного центра (Миннауки, Минобразования, РАН, РФФИ); предполагается дальнейшее наращивание мощности. Освоен режим телекоммуникационного доступа к МВС, в том числе по Internet, с обеспе-

чением требований защиты информации. Предстоят дальнейшие разноплановые работы по техническому и математическому освоению созданных систем, развитию их программного обеспечения. Эти работы входят в соответствующие целевые научно-технические программы федерального и ведомственного уровня.

Массово-параллельные масштабируемые системы МВС предназначены для решения прикладных задач, требующих большого объема вычислений и обработки данных. Суперкомпьютерная установка системы МВС представляет собой мультипроцессорный массив, объединенный с внешней дисковой памятью и устройствами ввода-вывода информации под общим управлением персонального компьютера или рабочей станции.

МВС-1000 — система 3-го поколения, основана на использовании микропроцессоров Alpha 21164 (разработка фирмы DEC-Compaq; выпускается также заводами фирм Intel и Samsung) с производительностью до 1–2 млрд операций в секунду и присоединенной оперативной памятью объемом 0,1–2 Гбайт.

Мультипроцессорный массив системы с блоками вторичного электросилового питания и вентиляцией располагается в стойках размером 550×650×2200 мм промышленного стандарта; вес заполненной стойки — 220 кг, потребляемая мощность до 4 кВт.

В основном исполнении системы межпроцессорный обмен структурно аналогичен используемому в системе МВС-100 и реализуется в двух модификациях: на базе «транспьютероподобного» связного микропроцессора TMS320C44 (фирма Texas Instruments), имеющего четыре канала с пропускной способностью каждого 20 Мбайт/с, либо на базе связного микропроцессора SHARC ADSP 21060 (фирма Analog Devices), имеющего шесть внешних каналов с пропускной способностью каждого 40 Мбайт/с.

Исполнение МВС-1000К отличается использованием для межпроцессорного обмена коммутационной сети Myrinet (фирма Myricom, США) с пропускной способностью канала в дуплексном режиме 2×160 Мбайт/с. Кроме того, предусмотрено подключение к каждому процессору памяти на жестком диске с объемом 2–9 Гбайт.

В стандартной стойке располагается до 64 процессоров системы МВС-1000 или 24 процессора системы МВС-1000К. Предусмотрены средства системного объединения стоек для установок с большим числом процессоров.

В программном обеспечении МВС используются:

− языки FORTRAN и C (C++), дополнительные средства описания параллельных процессов;

− программные средства PVM и MPI (общепринятые для систем параллельной обработки);

− средства реализации многопользовательских режимов и удаленного доступа.

Контрольные вопросы:

1. Чем отличаются многомашинные ВС от многопроцессорных ВС?
2. Разновидности многопроцессорных ВС.

3. Что такое кластеры, и какими преимуществами они обладают?
4. Объясните понятие «масштабируемость».
5. Объясните понятие «транспьютер».
6. Что такое коммутационные среды? Приведите примеры коммутаторов.
7. Что такое вычислительные системы и каковы их разновидности?
8. Назовите основные классы и подклассы вычислительных машин.
9. Дайте общую характеристику и определите область использования супер-ЭВМ и мэйнфреймов.

Лекция 14. Поток данных и поток команд

План:
1. Потоки команд и потоки данных.
2. Машины потоков данных.

1. Потоки команд и потоки данных

Общепринята удачная классификация ВС, которую предложил в 1970 г. Г. Флин (США). Основным определяющим архитектурным параметром он выбрал взаимодействие потока команд и потока данных (операндов и результатов).

В ЭВМ классической архитектуры ведется последовательная обработка команд и данных. Команды поступают одна за другой (за исключением точек ветвления программы), и для них из ОЗУ или регистров так же последовательно поступают операнды. Одной команде (операции) соответствует один необходимый ей набор операндов (как правило, два для бинарных операций). Этот тип архитектуры — «один поток команд — один поток данных», ОКОД (SISD — «Single Instruction, Single Data») условно изображен на рис. 33.

Рисунок 33. ВС типа ОКОД (SISD)

Тип ОКМД («один поток команд — много потоков данных» (SIMD — «Single Instruction — Multiple Data»)) — охватывает ВС, в которых одной командой обрабатываются набор данных, множество данных, вектор и вырабатывается множество результатов. Это векторные и матричные системы, в которых по одной команде выполняется одна и та же операция над всеми элементами массива — вектора или матрицы, распределенными между процессорными (обрабатывающими) элементами ПЭ или процессорами. Принцип обработки показан на рис. 34.

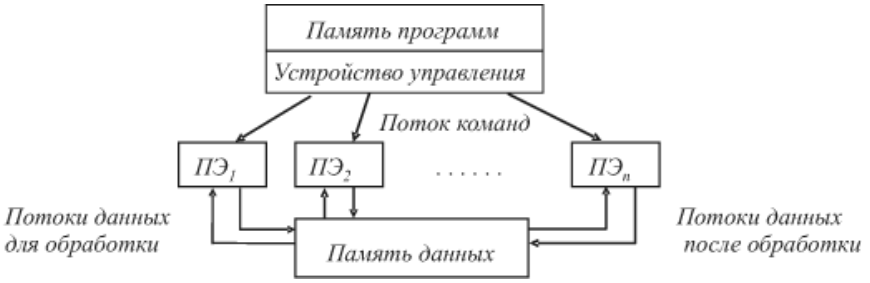


Рисунок 34. ВС типа ОКМД (SIMD)

Отечественные векторные ВС (ПС-2000, ПС-2100) допускают организацию матричной обработки. Классический пример матричной архитектуры — ILLIAC-1V (США).

К типу МКОД («много потоков команд — один поток данных» (MISD — «Multiple Instruction — Single Data»)) принято относить векторный конвейер (обычно в составе ВС, чтобы подчеркнуть основной используемый принцип вычислений), например в составе ВС Crey-1, «Электроника ССБИС». На векторном конвейере производится последовательная обработка одного потока данных многими обрабатывающими устройствами (ступенями, станциями) конвейера.

К такому же типу относится ВС, реализующая макроконвейер (ВС «Украина»). В ней задача, решаемая циклически, «разрезается» на последовательные этапы, закрепляемые за отдельными процессорами. Запускается конвейер многократного выполнения цикла, составляющего задачу. Принцип обработки показан на рис. 35.

Тип МКМД («много потоков команд — много потоков данных» (MIMD — «Multiple Instruction — Multiple Data»)) соответствует более полному и независимому распараллеливанию. К этому типу относятся, например, все многопроцессорные вычислительные комплексы (МВК) семейства «Эльбрус».

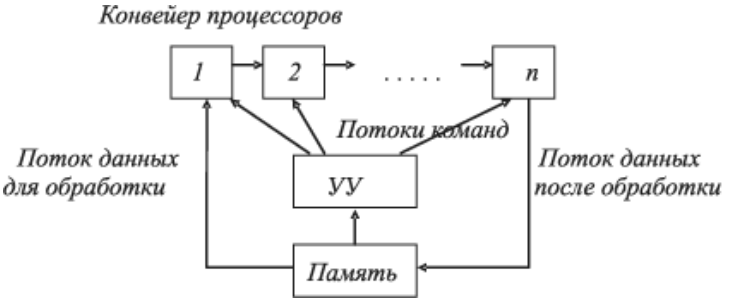


Рисунок 35. ВС типа МКОД (MISD)

2. Машины потоков данных

Появление машин, управляемых потоком данных (машины потоков данных), обусловлено тремя основными причинами: потребностью в существенном увеличении вычислительной мощности, серьезными недостатками прин-

ципов построения современных языков программирования и наличием «узких мест» в физической структуре традиционных машин.

Первая причина — отсутствие достаточной вычислительной мощности — является значительным препятствием для применения машин при решении важных проблем. В качестве примеров, подтверждающих значительность проблемы, приведем следующие факты. Новому поколению машин автоматического перевода с иностранного языка на другой понадобятся процессоры, способные выполнять 2–3 млрд операций в секунду. Для составления местного кратковременного прогноза погоды (на сутки) по результатам наблюдений за состоянием атмосферы потребуется скорость порядка 100 млрд операций в секунду. Моделирование действия подъемной силы на крыло самолета и турбулентности воздушного потока возможно при наличии вычислительных систем, обеспечивающих выполнение 1 млрд операций в секунду над числами с плавающей точкой. Эффективность будущих военных разработок зависит от бортовых вычислительных машин самолетов, кораблей и космических аппаратов, производительность которых должна быть по крайней мере выше, чем у современных аналогичных машин. Традиционным решением задачи увеличения мощности системы (после того как производительность единственного процессора доведена до максимального значения) является использование нескольких процессоров. Однако такой подход не является удовлетворительным в силу действия следующих обстоятельств.

Во-первых, возникают проблемы программирования, обусловленные необходимостью для программиста «подгонять» структуру данных и программ под жесткую структуру многопроцессорной или распределенной вычислительной системы.

Во-вторых, в традиционных мультипроцессорных системах в режиме разделения памяти между процессорами возможны наложения обращений к памяти — так называемая *интерференция обращений к памяти.* Поскольку каждый отдельный процессор стремится захватить определенную часть памяти, подключение новых процессоров к памяти, уже разделяемой другими процессорами, может дать только ограниченную выгоду. Интерференция обращений к памяти может быть уменьшена, если каждый процессор снабдить локальным кэш-буфером (кэш-памятью). Однако при этом возникает другая проблема: некоторая ячейка памяти может оказаться привязанной к нескольким кэш-буферам, т. е. операция записи одного процессора может повлечь за собой искажение информации в кэш-буферах других процессоров. Таким образом, хотя мультипроцессорные системы имеют определенные достоинства, повышая степень готовности системы для решения различных задач, они все же уступают однопроцессорным аналогам по такому критерию, как отношение стоимости к производительности.

Второй причиной появления машин потоков данных являются недостатки, присущие самой природе современных языков программирования, и осознание того факта, что эти языки отражают в своей структуре основополагающие принципы архитектуры машин фон Неймана. Упомянутая организация ЭВМ предполагает использование пассивной памяти, процессора, выполняющего операции по изменению содержимого памяти, и устройства управления, воз-

действующего на процессор с помощью потока следующих одна за другой команд. Понятие «переменная» в языке программирования адекватно понятию «область пассивной памяти» в машине, понятие «передача управления» в языке (реализуемая, например, операторами GO TO, IF, DO, CALL) отображает устройство управления и счетчик команд в машине фон Неймана. Подобным же образом понятие «присваивание» является отображением определенной операции процессора (изменения содержимого области памяти машины).

Использование потока данных для управления машиной позволило сделать выводы:

− основные характеристики традиционных машин являются искусственными, не соответствуют естественному порядку реализации алгоритмов решения задач и усложняют процесс программирования;

− современные языки программирования являются продуктом развития представлений не «снаружи внутрь» (т. е. не исходя из точки зрения программиста), а «изнутри наружу» (под сильным влиянием организации первых машин с запоминаемой программой).

Третьей причиной появления машин потоков данных является осознание того факта, что тракт, соединяющий процессор с памятью, является тем самым «узким местом», которое в основном и ограничивает эффективность системы. В современных вычислительных системах назначение программы заключается в изменении содержимого памяти, предоставляемой для данных этой программы. Достигается это путям «проталкивания» одиночных команд и данных через узкую магистраль, связывающую память и процессор.

Машины потоков данных производят наиболее сильное впечатление тем, что принципы их проектирования не базируются на основных свойствах и характеристиках традиционных машин и языков программирования. В архитектуре машин потоков данных отсутствует понятие «пассивная память данных», а в языке потоков данных нет понятия «переменная»: данные перемещаются из команды в команду по мере выполнения программы. Кроме того, в данном случае не используются понятия «передача управления», «счетчик команд» и «ветвление вычислительного процесса». Вместо этого команды (операторы) управляются данными. Считается, что команда готова к выполнению (т. е. ее выполнение разрешено), если данные присутствуют в каждом из ее входных портов и отсутствуют в выходном порте. Выполнение команды приводит к исчезновению данных в ее входных портах и появлению результата в выходном порте. Программа представляет собой направленный граф, образованный соединенными между собой командами: выходной порт одной команды соединен с входным портом другой команды. Таким образом, порядок выполнения команды определяется не счетчиком команд, а движением потока данных в командах.

Контрольные вопросы:

1. Как реализуется принцип обработки в ВС типа ОКОД (SISD)?
2. Как реализуется принцип обработки в ВС типа ОКМД (SIMD)?
3. Как реализуется принцип обработки в ВС типа МКОД (MISD)?
4. Как реализуется принцип обработки в ВС типа МКМД (MIMD)?

Тема 4. Процессы обработки информации на всех уровнях компьютерных архитектур

Лекция 15. Алгоритмы маршрутизации. Методы передачи данных

План:

1. Алгоритмы маршрутизации.
2. Методы передачи данных.

1. Алгоритмы маршрутизации

Алгоритмы маршрутизации определяют путь передачи данных от процессора — источника сообщения до процессора, к которому сообщение должно быть доставлено.

При разработке алгоритмов маршрутизации часто преследуют одну или несколько из перечисленных ниже целей:

1. Оптимальность. Она характеризует способность алгоритма маршрутизации выбирать «наилучший» маршрут.

2. Простота и низкие непроизводительные затраты. Другими словами, алгоритм маршрутизации должен эффективно обеспечивать свои функциональные возможности, с минимальными затратами программного обеспечения и коэффициентом использования.

3. Живучесть и стабильность. Другими словами, они должны четко функционировать в случае неординарных или непредвиденных обстоятельств, таких как отказы аппаратуры, условия высокой нагрузки и некорректные реализации.

4. Быстрая сходимость. Сходимость — это процесс соглашения между всеми маршрутизаторами по оптимальным маршрутам.

5. Гибкость. Другими словами, алгоритмы маршрутизации должны быстро и точно адаптироваться к разнообразным обстоятельствам в сети.

Алгоритмы маршрутизации могут быть классифицированы по типам:

1. Статические или динамические.

Алгоритмы, использующие статические маршруты, просты для разработки и хорошо работают в окружениях, где трафик сети относительно предсказуем, а схема сети относительно проста.

Динамические алгоритмы маршрутизации подстраиваются к изменяющимся обстоятельствам сети в масштабе реального времени. Они выполняют это путем анализа поступающих сообщений об обновлении маршрутизации.

2. Одномаршрутные или многомаршрутные.

Некоторые сложные протоколы маршрутизации обеспечивают множество маршрутов к одному и тому же пункту назначения. Такие многомаршрутные алгоритмы делают возможной мультиплексную передачу трафика по многочисленным линиям; одномаршрутные алгоритмы не могут делать этого. Преимущества многомаршрутных алгоритмов очевидны — они могут обеспечить значительно бóльшую пропускную способность и надежность.

3. Одноуровневые или иерархические.

В одноуровневой системе маршрутизации все роутеры равны по отношению друг к другу. В иерархической системе маршрутизации некоторые марш-

рутизаторы формируют то, что составляет основу (*backbone* — базу) маршрутизации. Основным преимуществом иерархической маршрутизации является то, что она имитирует организацию большинства компаний и, следовательно, очень хорошо поддерживает их схемы трафика.

4. С интеллектом в главной вычислительной машине или в маршрутизаторе.

В первой системе интеллект маршрутизации находится в главной вычислительной машине. Во второй системе интеллектом маршрутизации наделены роутеры.

5. Внутридоменные и междоменные.

Некоторые алгоритмы маршрутизации действуют только в пределах доменов; другие — как в пределах доменов, так и между ними. Природа этих двух типов алгоритмов различная. Поэтому понятно, что оптимальный алгоритм внутридоменной маршрутизации не обязательно будет оптимальным алгоритмом междоменной маршрутизации.

6. Алгоритмы состояния канала или вектора расстояний.

Алгоритмы состояния канала (известные также как алгоритмы «первоочередности наикратчайшего маршрута») направляют потоки маршрутной информации во все узлы объединенной сети. Однако каждый роутер посылает только ту часть маршрутной таблицы, которая описывает состояние его собственных каналов. Алгоритмы вектора расстояния (известные также как алгоритмы Белмана — Форда) требуют от каждого маршрутизатора посылки всей или части своей маршрутной таблицы, но только своим соседям. Алгоритмы состояния каналов фактически направляют небольшие корректировки по всем направлениям, в то время как алгоритмы вектора расстояний отсылают более крупные корректировки только в соседние роутеры.

2. Методы передачи данных

При обмене данными между узлами сети используются три метода передачи данных:

- симплексная (однонаправленная) передача (телевидение, радио);
- полудуплексная (прием и передача информации осуществляются поочередно);
- дуплексная (двунаправленная), каждая станция одновременно передает и принимает данные.

Для передачи данных в сетях наиболее часто применяется последовательная передача. Широко используются следующие методы последовательной передачи: асинхронная и синхронная.

При асинхронной передаче каждый символ передается отдельной посылкой (рис. 36). Стартовые биты предупреждают приемник о начале передачи. Затем передается символ. Для определения достоверности передачи используется бит четности (бит четности = 1, если количество единиц в символе нечетно, и 0 в противном случае). Последний бит («стопбит») сигнализирует об окончании передачи.

Преимущества: несложная отработанная система; недорогое (по сравнению с синхронным) интерфейсное оборудование.

Недостатки асинхронной передачи: третья часть пропускной способности теряется на передачу служебных битов (старт/стоповых и бита четности); невысокая скорость передачи по сравнению с синхронной; при множественной ошибке с помощью бита четности невозможно определить достоверность полученной информации.

Асинхронная передача используется в системах, где обмен данными происходит время от времени и не требуется высокая скорость передачи данных. Некоторые системы используют бит четности как символьный бит, а контроль информации выполняется на уровне протоколов обмена данными.

При использовании синхронного метода данные передаются блоками. Для синхронизации работы приемника и передатчика в начале блока передаются биты синхронизации. Затем передаются данные, код обнаружения ошибки и символ окончания передачи. При синхронной передаче данные могут передаваться и как символы, и как поток битов. В качестве кода обнаружения ошибки обычно используется циклический избыточный код обнаружения ошибок (CRC). Он вычисляется по содержимому поля данных и позволяет однозначно определить достоверность принятой информации.

Рисунок 36. Асинхронная и синхронная передача данных

Преимущества синхронного метода передачи информации: высокая эффективность передачи данных; высокие скорости передачи данных; надежный встроенный механизм обнаружения ошибок.

Недостатки: интерфейсное оборудование более сложное и, соответственно, более дорогое.

Контрольные вопросы:

1. Какие цели преследуют при разработке алгоритмов маршрутизации?
2. Как делятся алгоритмы маршрутизации?
3. Какие методы используются при обмене данными между узлами сети?

Лекция 16. Организация памяти в ЭВМ

План:

1. Память ЭВМ.
2. Иерархия ЗУ ЭВМ.
3. Запоминающие элементы.

1. Память ЭВМ

Память ЭВМ — функциональная часть ЭВМ, предназначенная для записи, хранения и выдачи данных. В соответствии с этим различают три режима работы памяти: записи, хранения и считывания. Запись в запоминающее устройство (ЗУ) или считывания из него информации иначе называются обращением к ЗУ. Быстродействие памяти определяется продолжительностью операции обращения к ЗУ. Время обращения t_0 при записи информации складывается из времени поиска операнда $t_п$, стирания ранее записанной информации $t_{ст}$ (при необходимости) и записи нового числа $t_{зп}$:

$$t_о = t_п + t_{ст} + t_{зп}.$$

При считывании информации время цикла обращения складывается из времени поиска, времени считывания и восстановления считанных кодов:

$$t_о = t_п + t_{ст} + t_{восст}.$$

Запоминающие устройства разделяют:

- по использованию — внешние и внутренние (или оперативные);
- по назначению — сверхоперативные, оперативные, постоянные, буферные и внешние;
- по физическим принципам действия — полупроводниковые, магнитные и оптические;
- по способу хранения информации — статические и динамические;
- по способу доступа к заданной ячейке (для адресных ЗУ) — с последовательным, циклическим и произвольным доступом;
- по характеру обращения – с адресным обращением (или адресной выборкой) и с ассоциативным обращением (ассоциативной выборкой).

Для достижения в ЭВМ одновременно и большой информационной емкости, и высокого быстродействия используется принцип иерархического построения памяти. Техническая реализация иерархических структур обеспечивает большую емкость памяти и малое время обращения, что позволяет решать на ЭВМ сложные задачи, требующие хранения большого количества данных. При иерархическом принципе построения структуры ЗУ логическая организация потоков информации должна быть такой, чтобы все информационное поле ЭВМ или ВС представлялось в виде внутреннего абстрактного ЗУ с единым адресным пространством. Это абстрактное ЗУ называют виртуальным (кажущимся) ЗУ. Адресацию его ячеек осуществляют посредством абстрактных математических адресов с использованием страничных таблиц.

2. Иерархия ЗУ ЭВМ

Запоминающие устройства ЭВМ образуют иерархию, представленную на рис. 37.

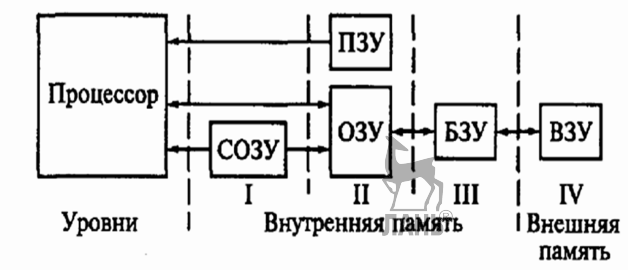


Рисунок 37. Схема иерархии ЗУ ЭВМ

Сверхоперативные ЗУ (СОЗУ) или местная память. Они имеют быстродействие, соизмеримое с быстродействием процессора. Емкость СОЗУ обычно составляет от нескольких десятков до нескольких тысяч слов, а период обращения — десятые или сотые доли микросекунды. Они предназначаются для хранения ряда чисел, используемых некоторой текущей последовательностью команд программы. Сверхоперативные ЗУ применяют в том случае, если быстродействие процессора ограничивается быстродействием ОЗУ.

Оперативные ЗУ (ОЗУ) или основная память предназначены для хранения данных и программ текущих вычислений, а также программ, к которым следует быстро перейти, если в ходе вычислительного процесса возникло прерывание. ОЗУ современных ЭВМ имеют емкость от нескольких тысяч до сотен тысяч слов и период обращения от долей до нескольких микросекунд. ОЗУ может быть связано с процессором как непосредственными связями, так и через СОЗУ. В качестве элементов памяти ОЗУ и СОЗУ используются полупроводниковые элементы, тонкие магнитные пленки, ферритовые сердечники и др.

Постоянные ЗУ (ПЗУ) или односторонние ЗУ предназначены только для хранения и считывания информации, которая не изменяется в процессе вычислений, например постоянно используемые программы, различные константы, таблицы функций, микропрограммы и т. п. В ПЗУ информация записывается один раз при изготовлении, а при работе только считывается. Такие ПЗУ имеют меньшую аппаратную сложность, чем ОЗУ.

Буферные ЗУ (БЗУ). Эти ЗУ используются для промежуточного хранения данных при обмене между устройствами с разным быстродействием, например между ОЗУ и внешним ЗУ. По емкости и быстродействию БЗУ занимают промежуточное место между ОЗУ и внешним ЗУ. Они могут быть построены на полупроводниковых элементах, ферритовых сердечниках и на магнитных дисках.

Внешние ЗУ (ВЗУ). Они предназначены для хранения больших массивов информации. Объем данных, которые могут одновременно храниться в таких ЗУ, обычно превышает сотни миллионов слов, однако период обращения к ним составляет от нескольких миллисекунд до нескольких десятков секунд. Данные, хранящиеся в ВЗУ, непосредственно не используются в вычислительном процессе, а передаются из ВЗУ в ОЗУ. В качестве ВЗУ чаще всего ис-

пользуют накопители информации на магнитных лентах (НМЛ), на гибких и жестких магнитных дисках (НГМД, НЖМД), на микросхемах на основе материалов, содержащих цилиндрические магнитные домены (ЦМД), а также на оптических дисках.

3. Запоминающие элементы

Основой полупроводникового запоминающего элемента (ЗЭ) служит бистабильная схема триггера, состоящая из инверторов с перекрестными связями, с цепями управления для записи и считывания информации. Достоинством таких ЗЭ является единство технологического процесса изготовления как самих ЗЭ, так и элементов электронного обрамления — регистров, дешифраторов и др.

Разнообразие типов ЗЭ объясняет большое количество способов объединения ЗЭ в большие интегральные схемы (БИС), способов объединения БИС в накопитель.

Классификацию ЗЭ можно провести по трем типам признаков:

− физико-технологическим;
− схемотехническим;
− системотехническим.

Физико-технологические признаки:

– выполненные на биполярных транзисторах;
– выполненные на МДП-транзисторах.

Схемотехнические признаки классификации:

− число транзисторов в ЗЭ;
− количество и функции адресных и разрядных шин выборки;
− характер нагрузочных сопротивлений в триггере (линейная, нелинейная, переключаемая нагрузка);
− тип связи ЗЭ с разрядными шинами (непосредственная, через ключевые элементы).

Системотехнических признаков три:

− принцип хранения (статический, динамический и квазистатический);
− принцип считывания (с разрушением и без разрушения информации);
− форма и вид считанного из ЗЭ сигнала (импульсный ток или импульсное напряжение).

Контрольные вопросы:

1. Что такое память ЭВМ?
2. Перечислите виды памяти.
3. По каким признакам классифицируются полупроводниковые запоминающие устройства?
4. Как определяется информационная емкость ЗУ?
5. Перечислите основные параметры ЗУ.

Лекция 17. Оперативная память. Виды адресации

План:

1. Основные характеристики элементов (микросхем) памяти.
2. Структура внутренней памяти.
3. Виды адресации.

1. Основные характеристики элементов (микросхем) памяти

Элементы ОП выполнены в виде модулей, так что при желании можно сравнительно легко заменить их или установить дополнительные и тем самым увеличить объем ОП компьютера. Характеристики ОП определяют быстродействие всей системы. Основная задача ОП — предоставлять по требованию МП необходимую информацию, т. е. данные должны быть доступны для обработки. Недостаток ОП состоит в том, что она временная.

Основные характеристики элементов (микросхем) памяти:

- тип;
- емкость;
- разрядность;
- быстродействие;
- временная диаграмма.

Емкость и разрядность

Структура микросхемы, которая имеет одну линию ввода-вывода, позволяет считать (записать) только один бит данных. Для повышения скорости обмена данными между МП и памятью были разработаны микросхемы, имеющие 4, 8 и 16 линий ввода-вывода. Подобные микросхемы имеют 4, 8, или 16 одинаковых матриц ячеек памяти. Таким образом, при поступлении на входы микросхемы адреса производится одновременное чтение (запись) всех ячеек, находящихся по данному адресу, но в различных матрицах. В этом случае одновременно считывается (записывается) сразу несколько бит информации. Например, если микросхема имеет 8 линий ввода-вывода (8 матриц), то МП может считывать/записывать информацию побайтно.

Количество линий ввода-вывода определяет разрядность шины ввода-вывода микросхем.

Количество бит информации, которое хранится в ячейках каждой матрицы, называется глубиной адресного пространства.

Общая емкость микросхемы равна произведению:

глубина адресного пространства × количество линий ввода-вывода.

Например, 1 Мб × 4 = 4 Мб.

Быстродействие

Производительность микросхемы динамической памяти характеризуется временем выполнения элементарных действий между двумя операциями чтения либо записи данных. Последовательность этих операций называют рабочим циклом (или циклом обращения). Он включает указание адреса данных (RAS, CAS), чтение (запись).

Время, необходимое для чтения (записи) данных, хранящихся по случайному адресу называется временем доступа (40–60 нс). В реальных условиях обращение к памяти чаще происходит не по случайному адресу, поэтому цикл короче. На материнскую плату можно установить элементы памяти различных форм, которые могут иметь различное время доступа (не должно различаться более чем на 10 нс во избежание серьезных проблем).

ЛАНЬ®

Временная диаграмма

Между МП и элементами памяти не должно быть временного рассогласования, которое может возникнуть из-за различного быстродействия этих компонентов.

Временная диаграмма показывает зависимость тактовой частоты системной шины от типа памяти. Она характеризует количество тактов, которые необходимы МП для выполнения четырех последовательных операций считывания данных.

2. Структура внутренней памяти

Основные структурные единицы памяти ПК: *бит*, *байт*, *машинное слово*.

Бит. Все данные и программы, хранящиеся в памяти ПК, имеют вид двоичного кода. Один символ из двоичного кода несет 1 бит информации. Ячейка памяти, хранящая один двоичный знак, называется «бит». В одном бите хранится один бит информации.

Битовая структура памяти определяет первое свойство памяти — дискретность.

Байт. Восемь расположенных подряд битов образуют байт. В одном байте памяти хранится один байт информации. Во внутренней памяти ПК все байты пронумерованы. Нумерация начинается от нуля. Порядковый номер байта называется его *адресом*. В ПК адреса обозначаются двоичным кодом. Используется также шестнадцатеричная форма обозначения адреса.

Машинное слово. Наибольшую последовательность бит, которую процессор может обрабатывать как единое целое, называют *машинным словом*. Длина машинного слова может быть разной — 8, 16, 32 бита и т. д. Адрес машинного слова в памяти ПК равен адресу младшего байта, входящего в это слово.

Занесение информации в память, а также извлечение ее из памяти производятся по адресам. Это свойство называется адресуемостью.

Пример 1. ПК имеет ОП 2 Кб. Указать адрес последнего байта ОП (десятичный, шестнадцатеричный, двоичный).

Решение:

Объем ОП составляет 2 Кб = 1024 × 2 б = 2048 б. Десятичный адрес (номер) последнего байта равен 2047, так как нумерация начинается с нуля.

Ответ: $2047_{10} = 7FF_{16} = 011111111111_2$.

Пример 2. Объем ОП компьютера равен 1 Мб, а адрес последнего машинного слова — 1048574. Чему равен размер машинного слова?

Решение:

1 Мб = 1024 Кб = 104857 б. Так как нумерация байтов начинается с нуля, значит адрес последнего байта будет равен 1048575. Таким образом, последнее машинное слово включает в себя 2 байта с номерами 1048574 и 1048575.

Ответ: 2 байта.

3. Виды адресации

В адресном ОЗУ каждый элемент памяти имеет адрес, соответствующий его пространственному расположению в запоминающей среде. Поэтому обращение к определенному элементу производится в соответствии с кодом его адреса. В ЗУ после приема кода осуществляется его дешифрация, после чего следует выборка из элемента конкретной группы битов или слов.

В ассоциативном ОЗУ поиск данных происходит по конкретному содержимому, независимо от его адреса. Такой поиск информации идет с использованием определенных признаков, например ключевых слов, которые связаны с искомыми данными. Ассоциативные устройства, хотя и являются более сложными, обеспечивают более быстрый поиск и выбор хранимых данных.

Рассмотрим адресные ОЗУ. Команды, исполняемые ЭВМ при выполнении программы, равно как и числовые и символьные операнды, хранятся в памяти компьютера. Память состоит из многих миллионов ячеек, в каждой из которых содержится один бит информации (значения 0 или 1). Биты редко обрабатываются поодиночке, а, как правило, группами фиксированного размера. Для этого память организуется таким образом, что группы по n бит могут записываться и считываться за одну операцию. Группа бит называется *словом*, а значение n — длиной слова.

Обычно длина машинного слова компьютеров составляет от 16 до 64 бит. Если длина слова равна 32 битам, в одном слове может храниться 32-разрядное число в дополнительном коде или четыре символа ASCII, занимающих 8 бит каждый.

Восемь идущих подряд битов являются байтом. Для представления машинной команды требуется одно или более слов.

Для доступа к памяти с целью записи или чтения отдельных элементов информации, будь то слова или байты, необходимы имена или адреса, определяющие их расположение в памяти. В качестве адресов традиционно используются числа из диапазона от 0 до $2^k - 1$ со значением k, достаточным для адресации всей памяти компьютера. Все 2^k адресов составляют адресное пространство компьютера. Следовательно, память состоит из 2^k адресуемых элементов. Например, использование 24-разрядных (как в процессоре 80286) адресов позволяет адресовать 2^{24} (16 777 216) элементов памяти. Обычно это количество адресуемых элементов обозначается как 16 Мбайт (1 Мбайт = 2^{20} = 1 048 576 байт, адресное пространство 8086 и 80186)

Адреса назначаются байтам памяти. Именно так адресуется память большинства современных компьютеров. Память, в которой каждый байт имеет отдельный адрес, называется памятью с байтовой адресацией. Последовательные байты имеют адреса 0, 1, 2 и т. д. Таким образом, при использовании слов длиною 32 бита последовательные слова имеют адреса 1, 4, 8, ..., и каждое слово состоит из 4 байт.

Прямой и обратный порядок байтов. Существует два способа адресации байтов в словах:

− в прямом порядке (рис. 38*а*);

− в обратном порядке (рис. 38*б*).

Обратным порядком байтов *(big-endian)* называется система адресации, при которой байты адресуются слева направо, так что самый старший байт слова (расположенный с левого края) имеет наименьший адрес.

Прямым порядком байтов *(little-endian)* называется противоположная система адресации, при которой байты адресуются справа налево, так что наименьший адрес имеет самый младший байт слова (расположенный с правого края). Слова «старший» и «младший» определяют вес бита, т. е. степень двойки, соответствующей данному биту, когда слово представляет число.

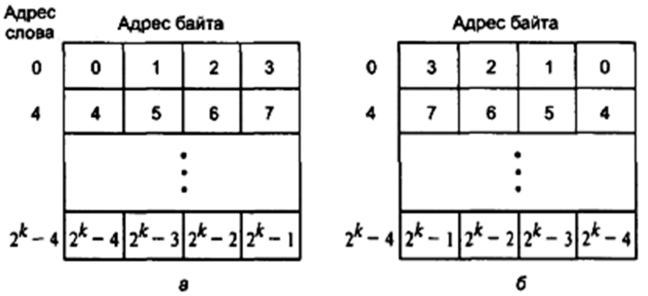


Рисунок 38. Способы адресации байтов в ОЗУ

Наряду с порядком байтов в слове важно также определить порядок битов в байте. Наиболее естественный порядок битов для кодирования числовых данных (непосредственно соответствующий их разрядам) «слева направо»: b_{32}, ..., b_1, b_0. Однако существуют компьютеры, для которых характерен обратный порядок битов.

Расположение слов в памяти. В случае 32-разрядных слов их естественные границы располагаются по адресам 0, 4, 8 и т. д. При этом считается, что слова *выровнены по адресам* в памяти. Если говорить в общем, слова считаются выровненными в памяти в том случае, если адрес начала каждого слова кратен количеству байтов в нем. По практическим причинам, связанным с манипулированием двоично-кодированными адресами, количество байтов в слове обычно является степенью двойки. Поэтому, если длина слова равна 16 битам (2 байтам), выровненные слова начинаются по байтовым адресам 0, 2, 4, ..., а если она равна 64 битам (2^3, т. е. 8 байтам), то выровненные слова начинаются по байтовым адресам 0, 8, 16, ...

Не существует причины, по которой слова не могли бы начинаться с произвольных адресов. Такие слова называются *невыровненными*. Как правило, слова выравниваются по адресам памяти, но иногда этот принцип нарушается.

Доступ к числам, символам и символьным строкам. Обычно число занимает целое слово, поэтому, для того чтобы обратиться к нему, нужно указать адрес слова, по которому оно хранится. Точно так же доступ к отдельно хранящемуся в памяти символу осуществляется по адресу содержащего его байта. Во многих приложениях необходимо обрабатывать строки символов переменной длины. Для доступа к такой строке нужно указать адрес байта, в котором хранится ее первый символ. Последовательные символы строки содержатся в последовательных байтах. Существует два способа определения длины строки. Первый из них заключается в использовании специального управляющего символа, обозначающего конец строки и являющегося ее последним символом. Второй способ состоит в использовании отдельного слова памяти или регистра процессора, содержащего число, которое определяет длину строки в байтах.

Контрольные вопросы:

1. Перечислите основные структурные единицы памяти.
2. Какие существуют виды адресации?
3. Как организуется адресация байтов в словах?

Лекция 18. Основы программирования процессора

План:

1. Взаимодействие оперативной памяти и процессора.
2. Директивы определения данных.
3. Регистры процессора.
4. Инструкции Ассемблера.

1. Взаимодействие оперативной памяти и процессора

В оперативной памяти хранится выполняемая программа вместе с принадлежащими ей данными; процессор выполняет вычисления и другие действия, описанные в программе. Операционная система, загрузив программу и при необходимости настроив ее для выполнения в той области памяти, куда она попала, сообщает процессору начальный адрес загруженной программы и инициирует процесс ее выполнения. Процессор считывает из памяти первую команду программы, находит в памяти или в своих регистрах данные, необходимые для ее выполнения (если команда требует данных) и, выполнив требуемую операцию, возвращает в память или, возможно, оставляет в регистрах результат своей работы (рис. 39).

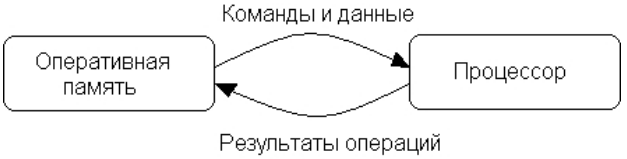


Рисунок 39. Взаимодействие оперативной памяти и процессора

Выполнив первую команду, процессор переходит к следующей и так дальше до конца программы. Завершив программу, процессор не будет знать, что ему дальше делать, поэтому любая программа должна завершаться командами, передающими управление операционной системе компьютера. Оперативная память компьютера представляет собой электронное устройство, состоящее из большого числа двоичных запоминающих элементов, а также схем управления ими. Все байты оперативной памяти нумеруются, начиная с нуля. Нужные байты отыскиваются в памяти по их номерам, выполняющим функции адресов. Некоторые данные (например, коды символов) требуют для своего хранения одного байта; для других данных этого места не хватает, и под них в памяти выделяется 2, 4, 8 или еще большее число байтов. Обычно пары байтов называют словами, а четверки — двойными словами (рис. 40), хотя иногда термином «слово» обозначают любую порцию машинной информации.

При обсуждении содержимого многобайтового данного приходится ссылаться на составляющие его байты; эти байты условно нумеруются от нуля и располагаются в порядке возрастания номера справа налево, так что слева оказываются байты с большими номерами, а справа — байты с меньшими номерами. Крайний слева байт принято называть старшим, а крайний справа — младшим. Такой порядок расположения байтов связан с привычной для нас формой записи чисел: в многоразрядном числе слева указываются старшие разряды, а справа — младшие. Следующее число, если его написать за предыдущим, опять

начнется со старшего разряда и закончится младшим. Однако в памяти компьютера данные располагаются в более естественном порядке непрерывного возрастания номеров байтов, и, таким образом, каждое слово или двойное слово в памяти начинается с его младшего байта и заканчивается старшим (рис. 41).

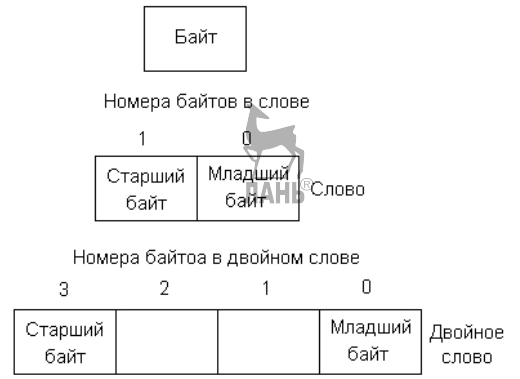


Рисунок 40. Байт, слово и двойное слово

Рисунок 41. Нумерация байтов в многобайтовых данных

Для записи иных данных, символов или дробных чисел предусматриваются правила кодирования, т. е. представления в виде последовательности битов той или иной длины. Так, действительное число одинарной точности занимает в памяти двойное слово (32 бита), в котором 23 бита отводятся под мантиссу, 8 бит под порядок и еще один бит под знак числа. Программы, работающие с такого рода данными, должны, естественно, знать правила их записи и руководствоваться ими при обработке и представлении этих данных.

В языке ассемблера шестнадцатеричные числа, чтобы отличать их от десятичных, завершаются буквой h (или H). Таким образом, 100 — это десятичное число, а l00h — шестнадцатеричное (равное 256). Поскольку одна шестнадцатеричная цифра требует для записи ее в память компьютера четырех двоичных разрядов, то содержимое байта описывается двумя шестнадцатеричными цифрами (от 00h до FFh, или от 0 до 255), а содержимое слова — четырьмя (от 0000h до FFFFh, или от 0 до 65535).

Помимо ячеек оперативной памяти, для хранения данных используются еще запоминающие ячейки, расположенные в процессоре и называемые *регистрами.* Достоинство регистров заключается в их высоком быстродействии, гораздо большем, чем у оперативной памяти, а недостаток в том, что их очень

мало — всего около десятка. Поэтому регистры используются лишь для кратковременного хранения данных.

В режиме МП 86 все регистры процессора имеют длину 16 разрядов, или 1 слово.

За каждым регистром закреплено определенное имя (например, AX или DS), по которому к нему можно обращаться в программе.

Для того, чтобы с помощью 16-разрядных чисел адресовать любой байт памяти, в МП 86 предусмотрена сегментная адресация памяти, реализуемая с помощью сегментных регистров процессора. Суть сегментной адресации заключается в следующем. Обращение к памяти осуществляется исключительно с помощью сегментов — логических образований, накладываемых на те или иные участки физической памяти. Исполнительный адрес любой ячейки памяти вычисляется процессором путем сложения начального адреса сегмента, в котором располагается эта ячейка, со смещением к ней (в байтах) от начала сегмента (рис. 42). Это смещение иногда называют относительным адресом.

Рисунок 42. Образование физического адреса из сегментного адреса и смещения

Начальный адрес сегмента без четырех младших битов, т. е. деленный на 16, помещается в один из сегментных регистров и называется сегментным адресом. Сам же начальный адрес хранится в специальном внутреннем регистре процессора, называемом теневым регистром. Для каждого сегментного регистра имеется свой теневой регистр; начальный адрес сегмента загружается в него процессором в тот момент, когда программа заносит в соответствующий сегментный регистр новое значение сегментного адреса.

Процедура умножения сегментного адреса на 16 (10h) является принципиальной особенностью реального режима, ограничивающей диапазон адресов, доступных в реальном режиме, величиной 1 Мбайт. Действительно, максимальное значение сегментного адреса составляет FFFFh, или 64 Кб – 1, из чего следует, что максимальное значение начального адреса сегмента в памяти равно FFFF0h, или 1 Мбайт – 16. Если, однако, учесть, что к начальному адресу сегмента можно добавить любое смещение в диапазоне от 0 до FFFFh, то адрес последнего адресуемого байта окажется равен 10FFEFh, что соответствует величине 1 Мбайт + 64 Кбайт – 17.

Диапазон адресов, формируемых процессором, называют адресным пространством процессора; как мы видим, в реальном режиме он немного превышает 1 Мбайт. Заметим еще, что для описания адреса в пределах 1 Мбайт требуются 20 двоичных разрядов, или 5 шестнадцатеричных. Процессор 8086 имел как раз 20 адресных линий и не мог, следовательно, выйти за пределы 1 Мбайт; современным 32-разрядным процессорам, если они работают в реальном режиме, доступно несколько большее (почти на 64 Кбайт) адресное пространство. Если же процессор работает в защищенном режиме (с использованием 32-разрядных регистров), то его адресное пространство увеличивается до 232 = 4 Гбайт.

Адрес — это порядковый номер ячейки памяти, т. е. неотрицательное целое число, поэтому в общем случае адреса представляются так же, как и числа без знака. Под адресом в i80x86 понимается 16-битовое смещение (*offset*) — адрес ячейки, отсчитанный от начала сегмента (области) памяти, которому принадлежит эта ячейка. В этом случае под адрес отводится слово памяти, причем адрес записывается в «перевернутом» виде (как и числа-слова вообще).

В другом случае под «адресом» понимается 20-битовый абсолютный адрес некоторой ячейки памяти. В силу ряда причин в i80x86 такой адрес задается не как 20-битовое число, а как пара «сегмент:смещение», где «сегмент» (*segment*) — это первые 16 битов начального адреса сегмента памяти, которому принадлежит ячейка, а «смещение» (*offset*) — 16-битовый адрес этой ячейки, отсчитанный от начала данного сегмента памяти (величина 16 × сегмент + смещение дает абсолютный адрес ячейки).

Такая пара записывается в виде двойного слова, причем (как и для чисел) в «перевернутом» виде: в первом слове размещается смешение, а во втором — сегмент, причем каждое из этих слов в, свою очередь, представлено в «перевернутом» виде.

Например, пара I234h:5678h будет записана так:

78	56	34	12
Смещение		Сегмент	

2. Директивы определения данных

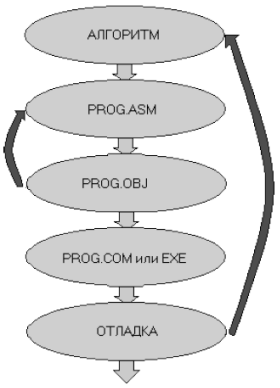
Язык программирования, наиболее полно учитывающий особенности «родного» микропроцессора и содержащий мнемонические обозначения машинных команд, называется Ассемблером. Программа, написанная на Ассемблере, называется исходной программой.

Разработка программы на Ассемблере состоит из следующих этапов:

1. Составление алгоритма в виде блок-схемы или структурного описания.

2. Написание текста исходной программы PROG.ASM с помощью редактора текстов. Имя PROG может быть произвольным, а расширение ASM — обязательно.

3. Перевод (трансляция или ассемблирование) исходной программы в машинные коды с помощью транслятора TASM.EXE. На этом этапе получается промежуточный продукт PROG.OBJ (объектный код). Выявленные при этом синтаксические и орфографи-ческие ошибки исправляются повтором пп. 2 и 3.



4. Преобразование с помощью программы TLINK.EXE объектного кода PROG.OBJ в выполнимый код PROG.EXE или PROG.COM. Эта операция называется компоновкой или связыванием. Для других семейств МП и МК исполнимые программы могут иметь другие расширения — BIN, HEX и др.

5. Выполнение программы и ее отладка начиная с п. 1, если встретились логические ошибки или ошибки периода выполнения (run time errors).

Текст программы на Ассемблере содержит следующие элементы/опе-рации:

а) команды (инструкции);

б) директивы (псевдооператоры);

в) операторы;

г) предопределенные имена.

Действия, обусловленные операциями, перечисленными в пп. б, в, г, выполняются на этапе трансляции, т. е. являются командами Ассемблеру. Операции, называемые командами или инструкциями, выполняются во время работы программы, т. е. являются командами микропроцессору.

Директивы или *команды ассемблера* — это предложения программы, которыми ее автор сообщает какую-то информацию Ассемблеру или просит что-то сделать дополнительно, помимо перевода символьных команд на машинный язык.

Для того чтобы в программе на Ассемблере зарезервировать ячейки памяти под константы и переменные, необходимо воспользоваться директивами определения данных — с названиями db (описывает данные размером в байт), dw (размером в слово) и dd (размером в двойное слово). В простейшем случае в директиве db, dw или dd описывается одна константа, которой дается имя для последующих ссылок на нее. По этой директиве ассемблер формирует машинное представление константы (в частности, если надо, «переворачивает» ее) и записывает в очередную ячейку памяти. Адрес этой ячейки становится значением имени: все вхождения имени в программу Ассемблер будет заменять на этот адрес.

Имена, указанные в директивах db, dw и dd, называются именами переменных (в отличие от меток — имен команд).

В Ассемблере числа записываются в нормальном (неперевернутом) виде в системах счисления с основанием 10, 16, 8 или 2. Десятичные числа записываются как обычно, за шестнадцатеричным числом ставится буква h (если число начинается с «цифры» A, B, ..., F, то вначале обязателен 0), за восьмеричным числом — буква q или o, за двоичным числом — буква b.

Примеры:

```
A DB 162        ; описать константу-байт 162 и дать ей имя A
B DB 0A2h       ; такая же константа, но с именем B
C DW -1         ; константа-слово -1 с именем C
D DW OFFFFh ; такая же константа-слово, но с именем D
E DD -1         ; -1 как двойное слово
```

Константы-символы описываются в директиве db двояко: указывается либо код символа (целое от 0 до 255), либо сам символ в кавычках (одинарных или двойных); в последнем случае Ассемблер сам заменит символ на его код.

По любой из директив db, dw и dd можно описать переменную, т. е. отвести ячейку, не дав ей начального значения. Имя, указанное в директиве, считается именующим первую из констант.

3. Регистры процессора

Регистры общего назначения (AX, BX, CX, DX) используются во всех арифметических и логических командах и конструктивно устроены следующим образом: каждый из этих регистров состоит из двух байтовых регистров, обозначаемых AH, AL и т. д. (H — старший, L — младший).

Все регистры имеют размер слова (16 бит), за каждым из них закреплено определенное имя (Ax, SP и т. п.). По назначению и использованию регистры можно разбить на следующие группы:

- регистры общего назначения (AX, BX, CX, OX, BP, SI, DI, SP);
- сегментные регистры (CS, DS, SS, ES);
- счетчик команд (IP);
- регистр флагов (Flags).

Регистры общего назначения можно использовать во всех арифметических и логических командах. В то же время каждый из них имеет определенную специализацию. При умножении байтов (слов, 16 бит) первый сомножитель обязан находиться в регистре AL (AX), результатом же умножения является слово (двойное слово, 32 бита), которое заносится в регистр AX (AX и DX), тем самым сохраняются все цифры произведения.

При делении байтов (слов) первый операнд (делимое) должен быть словом (двойным словом) и обязан находиться в регистре AX (DX и AX). Результатом деления являются две величины размером в байт (слово) — неполное частное (div) и остаток от деления (mod); неполное частное записывается в регистр AL (AX), а остаток — в результате AH (DX).

Таблица 30. Регистры МП

Регистр	Аббревиатура	Старший полурегистр	Младший полурегистр
Аккумулятор	AX	AH	AL
База	BX	BH	BL
Счетчик	CX	CH	CL
Данные	DX	DH	DL
Разряды	15-0	15-8	7-0

Шестнадцатеричные числа записываются с буквой h на конце, двоичные числа — с буквой b.

Например,

число 130 = 0082h хранится в памяти

82	00

а в регистре

AX	00	82
	AH	AL

В группу регистров данных включаются четыре регистра AX, BX, CX и DX. Программист может использовать их по своему усмотрению для временного хранения любых объектов (данных или адресов) и выполнения над ними требуемых операций. При этом регистры допускают независимое обращение к старшим (AH, BH, CH и DH) и младшим (AL, BL, CL и DL) половинам. Так, команда mov BL, AH пересылает старший байт регистра AX в младший байт регистра BX, не затрагивая при этом вторых байтов этих регистров. Заметьте, что сначала указывается операнд-приемник, а после запятой — операнд-источник, т. е. команда как бы выполняется справа налево.

Рисунок 43. Регистры процессора

Во многих случаях регистры данных вполне эквивалентны, однако предпочтительнее пользоваться регистром AX, поскольку многие команды занимают в памяти меньше места и выполняются быстрее, если их операндом является регистр AX (или его половина AL). С другой стороны, ряд команд использует определенные регистры неявным образом. Так, все команды циклов используют регистр CX в качестве счетчика числа повторений; в командах умножения и деления регистры AX и DX выступают в качестве неявных операндов; операции ввода-вывода можно осуществлять только через регистры AX или AL и т. д.

Индексные регистры SI и DI так же, как и регистры данных, могут использоваться произвольным образом. Однако их основное назначение — хранить индексы, или смещения относительно некоторой базы (т. е. начала массива), при выборке операндов из памяти. Адрес базы при этом может находиться в базовых регистрах BX или BP. Специально предусмотренные команды работы со строками используют регистры SI и DI в качестве неявных указателей в обрабатываемых строках.

Регистр BP служит указателем базы при работе с данными в стековых структурах, но может использоваться и произвольным образом в большинстве

арифметических и логических операций. Последний из группы регистров-указателей, указатель стека SP, стоит особняком от других в том отношении, что используется исключительно как указатель вершины стека.

Регистры SI, DI, BP и SP, в отличие от регистров данных, не допускают побайтовую адресацию. Четыре сегментных регистра CS, DS, ES и SS являются важнейшим элементом архитектуры процессора, обеспечивая адресацию 20-разрядного адресного пространства с помощью 16-разрядных операндов.

Указатель команд IP «следит» за ходом выполнения программы, указывая в каждый момент относительный адрес команды, следующей за исполняемой.

При вводе исходного текста программы с клавиатуры можно использовать как прописные, так и строчные буквы; транслятор воспринимает, например, строки MOV AX, DATA и mov ax, data одинаково. Предложения языка ассемблера могут содержать комментарии, которые отделяются от предложения языка знаком точки с запятой (;). При необходимости комментарий может занимать целую строку (начинающуюся со знака «;»). Текст программы заканчивается директивой ассемблера end, завершающей трансляцию. В качестве операнда этой директивы указывается точка входа в программу; в нашем случае это метка begin.

4. Инструкции Ассемблера

В тексте программы встречаются ключевые слова двух типов: команды процессора (mov, int) и директивы транслятора (в данном случае термины «транслятор» и «ассемблер» являются синонимами, обозначая программу, преобразующую исходный текст, написанный на языке ассемблера, в коды, которые будут при выполнении программы восприниматься процессором). Директивы служат для передачи транслятору служебной информации, которой он пользуется в процессе трансляции программы. Однако в состав выполнимой программы, состоящей из машинных кодов, эти строки не попадут, так как процессору, выполняющему программу, они не нужны.

Инструкции при трансляции преобразуются в команды процессора, которые исполняются после загрузки в память загрузочного модуля программы, имеющего расширение .com или .exe.

Директивы управляют процессом ассемблированием — преобразованием текста исходной программы в коды объектного модуля (расширение .obj). Ассемблер интерпретирует и обрабатывает операторы один за другим.

Общий формат оператора ассемблер:

[Метка:] Код_операции [Операнд1 [, Операнд2]] [; комментарии],

где элементы, указанные в квадратных скобках, могут отсутствовать.

Метка — это идентификатор, присваиваемый первому байту того оператора, в котором она появляется. Код_операции — это мнемоническое обозначение соответствующих команд процессора. Операнды оператора ассемблера описываются выражениями. Выражения конструируются на основе операций над числовыми и текстовыми константами, метками и идентификаторами переменных с использованием знаков операций и некоторых зарезервированных слов.

ОПЕРАТОРЫ

1. () — скобки, определяют порядок вычислений.

2. [] — например, [BX] означает содержимое ячейки памяти с адресом в регистре BX. Признак косвенной или прямой адресации.

mov al, [bx] mov al, [my_mem1].

3. + , -, *, / — операторы сложения, вычитания, умножения и деления.

mov ax, (2 * 3 + 8/2) - 2; в регистр ax будет помещено число 8.

4. MOD — деление по модулю. Дает остаток.

5. SHL, SHR — сдвиг операнда влево, вправо.

mov si, 01010101b SHR 3; в регистр SI будет загружено число 0Ah (00001010).

6. NOT — побитовая инверсия.

7. AND, OR, XOR — операции «И», «ИЛИ», «ИСКЛ.ИЛИ».

mov dl, (10d OR 5d) XOR 7d; (dl) будет равно 8.

8. : — переназначение сегмента.

mov dl, [es:bx]; поместить в dl байт данных из сегмента es и отстоящий от его начала на (bx) байтов (смещение).

9. OFFSET — оператор получения смещения адреса относительно начала сегмента (т. е. количества байтов от начала сегмента до идентификатора адреса).

mov bx, OFFSET table; table — символическое имя, OFFSET table — адрес.

ДИРЕКТИВЫ (ПСЕВДООПЕРАТОРЫ)

1. : — определяет близкую метку (в пределах сегмента).

```
jmp lbl .... ....
lbl: ....
```

2. = — присваивает символическому имени значение выражения (допускается переопределение).

videoram = 0B800h; videoram = 0B000h; значение переопределено.

3. .CODE — указывает начало кодового сегмента, то есть сегмента, где располагаются коды программы.

4. .DATA — определяет начало сегмента данных в программе.

5. DB, DW ... — директивы, резервирующие один или несколько байтов: DB, или одно или несколько слов: DW. fibs, rus ... Array — произвольные имена.

```
....
.DATA
fibs DB 1,1,2,3,5,8,13
rus DB 'Турбо Ассемблер'
buf DB 80 DUP(0); резервируется 80 байтов, каждый обнуляется

int DW 65535; в двух байтах располагается число FFFFh.
Array DW 100 DUP (0); резервируется 100 слов
```

6. END — обозначает конец программы.

```
....
.CODE
MyPROG:....; точка входа (начало программы).
....; команды программы.
....
END MyPROG; MyPROG — произвольное имя.
```

7. ENDM — окончание блока или макроопределения.

8. ENDP — обозначает конец подпрограммы.

9. EQU — присваивает символическому имени или строке значение выражения (в отличие от «=» переопределение не допускается).

```
BlkSize EQU 512
BufBlks EQU 4
BufSize EQU BlkSize * BufBlks
```

10. LABEL — определяет метку соответствующего типа.

```
....
.DATA
m_byte LABEL BYTE; метка m_byte типа BYTE позволяет теперь
m_word DW 0; иметь доступ отдельно к каждому байту данных
.CODE;m_word типа WORD
....
mov [m_word],0204h
add [m_byte],'0'; теперь в m_word хранится код
add [m_byte + 1],'0';3234h,ASCII код '0' равен 30h
```

11. LOCAL — определяет метки внутри макроопределений как локальные и в каждом макрорасширении вместо них ассемблер вставляет уникальные метки: ??XXXX, где XXXX = (0000...FFFF)h. Почему «XXXX»? Потому что никому не должно прийти в голову начинать символическое имя с ??, и транслятор смело может генерировать метки, не боясь совпадений.

12. MACRO — задает макроопределение.

```
XchgM MACRO a,b; a,b — параметры макро (ячейки памяти)
mov ax,b ; данное макрооопределение позволяет делать
mov bx,a ; обмен данными между ячейками памяти, в
mov a,ax ; отличие от команды xchg ;
mov b,bx ; нельзя записать mov a,b;
ENDM
```

Вызов этого макроса производится командой: XchgM m,n. Во время трансляции каждый макровызов заменяется макроопределением — макроподстановка.

13. .MODEL — определяет размер памяти под данные и код программы.

.MODEL tiny; под программу, данные и стек отводится один общий сегмент (64 Kb).

14. PROC — определяет начало подпрограммы.

```
Print PROC; Print — произвольное имя
; здесь команды подпрограммы
```

ret; обязательная команда возврата
ENDP
....
call Print ; вызов подпрграммы.

15. .STACK — задает размер стека.

.STACK 200h; выделяет 512 байтов для стека.

16. .RADIX base — определяет систему счисления по умолчанию, где base — основание системы счисления: 2, 8, 10, 16.

.RADIX 8
oct = 77; oct равно 63d.

17. **;** — начало комментария.

Краткая справка

1. Запуск среды Ассемблер: C:\ASSEMBLER\QC\BIN\QC.EXE
2. Сохранение файла под именем *.asm командой [File-Save as…].
3. При компиляции и отладке программы на Ассемблере следует правильно установить режим работы QC: Options — Make — Compiler flags — Memory model — Small.

Options — Make — Linker flags — Generate COM-file.

4. Вызов регистровой панели: View\windows\registers.
5. При пошаговой отладке программы и подпрограмм контролировать содержимое регистров можно в окне Registers, которое включается с помощью цепочки команд: View — Windows — Register.

При выполнении работы можно использовать регистры AX, BX, CX, DX и регистровый и непосредственный методы адресации. Для задания констант используется директива EQU.

Контрольные вопросы:

1. Дайте определение алгоритма и программы решения задачи.
2. Что такое регистр процессора?
3. Какие регистры микропроцессорной памяти используются для адресации данных, команд программы, стековой памяти?
4. Назовите основные компоненты языка ассемблер, приведите структуру команд.
5. Приведите структуру ассемблерной программы и дайте краткую характеристику основных структурных фрагментов этой программы.
6. Каково назначение отладчика программ? Назовите основные его возможности.

Тема 5. Основные компоненты программного обеспечения компьютерных систем

Лекция 19. Программное обеспечение (ПО) вычислительных систем

План:

1. Классы программных продуктов.
2. Состав и назначение инструментария технологии программирования.
3. Пакеты прикладных программ.

1. Классы программных продуктов

Программные продукты можно классифицировать по различным признакам. Рассмотрим классификацию, в которой основополагающим признаком является сфера (область) использования программных продуктов:

- аппаратная часть автономных компьютеров и сетей ЭВМ;
- функциональные задачи различных предметных областей;
- технология разработки программ.

Рисунок 44. Классы программных продуктов

Для поддержки информационной технологии в этих областях выделяют соответственно три класса программных продуктов:

- системное программное обеспечение;
- пакеты прикладных программ;
- инструментарий технологии программирования.

Системное программное обеспечение (System Software) — совокупность программ и программных комплексов для обеспечения работы компьютера и сетей ЭВМ.

Пакет прикладных программ (application program package) — комплекс взаимосвязанных программ для решения задач определенного класса конкретной предметной области.

Инструментарий технологии программирования — совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов.

Базовое программное обеспечение (base software) — минимальное количество программных средств, обеспечивающих работу ПК.

Сервисное программное обеспечение — программы и программные комплексы, которые расширяют возможности базового программного обеспечения и организуют более удобную среду работы пользователя.

В базовое программное обеспечение входят:

- операционная система;
- операционные оболочки (текстовые и графические);
- сетевая операционная система.

Рисунок 45. Структура системного программного обеспечения

Прикладные программы — программы, которые непосредственно обеспечивают выполнение необходимых пользовательских работ: редактирование текстов, рисование картинок, обработка информации массивов.

Наиболее часто используются следующие типы прикладных программ:

1. Редакторы текстов и издательские системы предоставляют возможность подготавливать документы на компьютере.

2. Табличные процессоры обеспечивают работу с большими таблицами чисел. При работе с табличным процессором на экран выводится прямоугольная таблица, в клетках которой могут находиться числа, пояснительные тексты и формулы для расчета значений в клетке по имеющимся данным.

3. Системы управления базами данных (СУБД) позволяют управлять большими информационными массивами. Наиболее простые системы этого вида позволяют обрабатывать на компьютере один массив информации, например персональную картотеку. Они обеспечивают ввод, поиск, сортировку записей, составление отчетов и т. д., все действия в них осуществляются с помощью меню и других диалоговых средств.

4. Графические редакторы позволяют создавать и редактировать картинки на экране компьютера. Как правило, пользователю представляются возможности: рисование линий, кривых, раскраски областей экрана, создание надписей различными шрифтами.

5. Бухгалтерские программы предназначены для ведения бухгалтерского учета, подготовки финансовой отчетности и финансового анализа деятельности предприятий. Из-за несовместимости отечественного бухгалтерского учета с зарубежным бухгалтерским учетом в нашей стране используются почти ис-

ключительно отечественные бухгалтерские программы (начисление заработной платы, учет материалов, товаров и т. д.).

6. Досуговые программы.

2. Состав и назначение инструментария технологии программирования

В настоящее время бурно развивается направление, связанное с технологией создания программных продуктов. Это обусловлено переходом на промышленную технологию производства программ, стремлением к сокращению сроков, трудовых и материальных затрат на производство и эксплуатацию программ, обеспечению гарантированного уровня их качества. Это направление часто называют программотехникой. *Программотехника* (software en-gineering) — технология разработки, отладки, верификации и внедрения программного обеспечения. *Инструментарий технологии программирования* — программные продукты поддержки (обеспечения) технологии программирования.

В рамках этих направлений сформировались следующие группы программных продуктов:

1) средства для создания приложений, включающие:
   - локальные средства, обеспечивающие выполнение отдельных работ по созданию программ;
   - интегрированные среды разработчиков программ, обеспечивающие выполнение комплекса взаимосвязанных работ по созданию программ;
2) CASE-технология, представляющая методы анализа, проектирования и создания программных систем и предназначенная для автоматизации процессов разработки и реализации информационных систем.

Рисунок 46. Классификация инструментария технологии программирования

Прикладное ПО служит программным инструментарием решения функциональных задач и является самым многочисленным классом ПО. В данный класс входят программные продукты, выполняющие обработку информации различных предметных областей. Таким образом, прикладное ПО — комплекс взаимосвязанных программ для решения задач определенного класса предметной области.

Инструментарий технологии программирования обеспечивает процесс разработки программ и включает специализированное ПО, которое является инструментальным средством разработки. ПО данного класса поддерживает все технологические этапы процесса проектирования, программирования, отладки и тестирования создаваемых программ. Пользователями данного ПО являются системные и прикладные программисты.

Инструментарий технологии программирования

Инструментарий технологии программирования — это программные продукты, предназначенные для поддержки технологии программирования.

Средства для создания приложений — совокупность языков и систем программирования, инструментальные среды пользователя, а также различные программные компоненты для отладки и поддержки создаваемых программ.

Язык программирования — это формализованный язык для описания алгоритма решения задач на компьютере. Языки программирования можно условно разделить на следующие классы:

− машинные языки — это языки, воспринимаемые аппаратной частью компьютера (машинные коды);

− машинно-ориентированные языки, отражающие структуру конкретного типа компьютера (ассемблер);

− процедурно-ориентированные языки — это языки, в которых имеется возможность описания программы как совокупности процедур или подпрограмм (Си, Паскаль и др.);

− проблемно-ориентированные языки, предназначенные для решения задач определенного класса (ЛИСП, ПРОЛОГ).

Другой классификацией языков является их деление на языки, ориентированные на реализацию основ структурного программирования, основанного на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей, и объектно-ориентированные языки, поддерживающие понятие объектов, их свойств и методов обработки.

Системы программирования включают:

− компилятор (транслятор);

− интегрированную среду разработки программ (не всегда);

− отладчик;

− средства оптимизации кода программ;

− набор библиотек;

− редактор связей;

− сервисные средства (утилиты) (для работы с библиотеками, текстовыми и двоичными файлами);

− справочные системы;

− систему поддержки и управления продуктами программного комплекса.

Компилятор транслирует всю программу без ее выполнения. Трансляторы (интерпретаторы) выполняют пооперационную обработку и выполнение программы.

Отладчики (debugger) — специальные программы, предназначенные для трассировки и анализа выполнения других программ. Трассировка — это обеспечение выполнения в пооператорном варианте.

Инструментальная среда пользователя — это специальные средства, встроенные в пакеты прикладных программ, такие как:

- библиотека функций, процедур, объектов и методов обработки;
- макрокоманды;
- клавишные макросы;
- языковые макросы;
- конструкторы экранных форм и объектов;
- генераторы приложений;
- языки запросов высокого уровня;
- конструкторы меню и др.

Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы.

CASE-технология (CASE — Computer-Aided System Engineering) — программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем.

Средства CASE-технологий делятся на:

- встроенные в систему реализации — все решения по проектированию и реализации привязки к выбранной СУБД;
- независимые от системы реализации — решения по проектированию ориентированы на унификацию (определение) начальных этапов жизненного цикла программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации.

Основное достоинство CASE-технологии — это поддержка коллективной работы над проектом за счет возможности работы в локальной сети разработчиков, экспорта (импорта) любых фрагментов проекта, организованного управления проектами.

В некоторых CASE-системах поддерживается кодогенерация программ — создание каркаса программ и создание полного продукта.

Примеры программных продуктов для создания приложений: Visual C++, Delphi, Visual Basic и т. д.

3. Пакеты прикладных программ

Классификация пакетов прикладных программ (ППП) приведена на рис. 47.

Проблемно-ориентированные ППП. Для некоторых предметных областей возможна типизация функций управления, структуры данных и алгоритмов обработки. Это вызвало разработку значительного типа ППП одинакового функционального назначения:

- автоматизированный бухгалтерский учет;
- финансовая деятельность;
- управление персоналом;

− управление производством;
− банковские информационные системы и т. п.

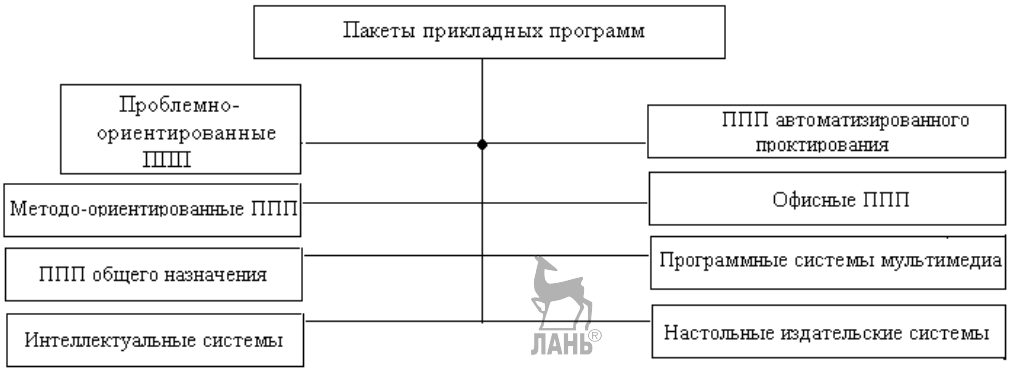


Рисунок 47. Классификация пакетов прикладных программ

Перечислим основные тенденции развития ППП:

− создание программных комплексов в виде автоматизированных рабочих мест (АРМ) управленческого персонала;

− создание интегрированных систем управления предметной областью на базе вычислительных сетей, объединяющих АРМы;

− организация данных больших информационных систем в виде распределенной БД на сети ЭВМ;

− наличие простых языков средств конечного пользователя и др.

ППП автоматизированного проектирования предназначены для поддержки работы конструкторов и технологов, связанной с работкой чертежей, схем, графическим моделированием и конструированием. Отличительной особенностью этого класса ППП являются высокие требования к аппаратному обеспечению, наличие библиотек встроенных функций, объектов, интерфейсов с графическими системами и БД (AutoCAD).

К *ППП общего назначения* относятся следующие программные средства:

1. Системы управления базами данных (СУБД), обеспечивающие организацию и хранение локальных БД на автономно работающих компьютерах либо централизованное хранение БД на файл-сервере и сетевой доступ к ним.

2. Серверы БД — это ПО, предназначенное для создания и использования при работе в сети интегрированных БД в архитектуре клиент — сервер. Многопользовательские СУБД в сетевом варианте обработки информации хранят данные на файл-сервере, специально выделенном компьютере, но сама обработка ведется на рабочих станциях. Серверы БД в отличие от этой большей части обработки данных (хранение, поиск, извлечение и передачи данных клиенту) выполняют самостоятельно, одновременно обеспечивая данными большое число пользователей сети. Общим для различных видов серверов БД является использование реляционного языка SQL (Structured Query Language) для реализации запросов к данным.

3. Генераторы отчетов (серверы отчетов) обеспечивают реализацию запросов и формирование отчетов в печатном или экранном виде в условиях сети с архитектурой клиент — сервер. Сервер отчетов подключается к серверу БД, используя драйверы сервиса БД (Crystal Reports, Profit for Windows).

4. Текстовые процессоры предназначены для работы с текстовыми документами. Развитием данного направления являются издательские системы (Microsoft Word).

5. Табличные процессоры являются удобной средой для вычислений конечным пользователем, содержат средства деловой графики, средства специализированной обработки (Microsoft Excel).

6. Средства презентационной графики — специализированные программы, предназначенные для создания изображений и их показа на экране, подготовка слайд-фильмов, мультфильмов и их проектирования (Microsoft PowerPoint).

7. Интегрированные пакеты — набор нескольких программных продуктов, функционально дополняющих друг друга, поддерживающих единые информационные технологии, реализованные на единой операционной и вычислительной платформе (Microsoft Office). Компоненты интегрированных пакетов могут работать изолированно друг от друга, имеют общий интерфейс благодаря чему их легче осваивать.

Методо-ориентированные ППП. Данный класс охватывает программные продукты, обеспечивающие независимо от предметной области и функции информационных систем математические, статистические и другие методы решения задач. Наиболее распространены методы математического программирования, решения дифференциальных уравнений, имитационного моделирования, исследования операций (Storm, SYSTAT, SAS и др.).

Офисные ППП. Данный класс охватывает программы, обеспечивающие ориентационное управление деятельностью офиса:

− органайзеры (планировщики) — ПО для планирования рабочего времени, составления протоколов встреч, расписаний, ведения записей и телефонной книжки. В состав входят: калькулятор, записная книжка, часы-календарь и т. п.;

− программы-переводчики, средства проверки орфографии, распознавания текста (Tiger — система распознавания русского языка, Stylus Lingvo Office, содержащий Fine Reader, Stylus for Windows — переводчик на указанный язык, корректор орфографии Lingvo Corrector и резидентный словарь Lingvo);

− коммуникационные пакеты предназначены для организации взаимодействия пользователей с удаленными абонентами или информационными ресурсами сети;

− браузеры, средства создания WWW-страниц;

− средства электронной почты (Pegasys Mail).

Настольные издательские системы. Данный класс ПО включает программы (PageMaker, CorelDraw, PhotoShop for Windows и т. д.), обеспечивающие информационную технологию компьютерной издательской деятельности:

− форматирование и редактирование текстов;
− автоматическую разбивку текста на страницы;
− компьютерную верстку печатной страницы;
− монтирование графики;
− подготовку иллюстраций и т. п.

Программные средства мультимедиа. Основное значение данных программных средств — создание и использование аудио- и видеоинформации для расширения информационного пространства пользователя (различные БД произведений искусства, библиотеки звуковых записей и т. д.).

Системы искусственного интеллекта:

− программы-оболочки для создания экспертных систем путем наполнения баз знаний и правил логического вывода;

− готовые экспертные системы для принятия решений в рамках определенных предметных областей;

− системы анализа и распознавания речи, текста и т. п.

Примеры систем искусственного интеллекта: FTDE, MYSIN, Guru и др.

Контрольные вопросы:

1. Как делятся программные продукты?
2. Что такое CASE-технология?
3. Какой состав имеет инструментарий технологии программирования?
4. Перечислите основные типы прикладных программ.

Лекция 20. Основные компоненты ПО компьютерных систем на базе ОС Windows

План:

1. Классификация операционной системы (ОС).
2. Компоненты ОС.
3. Операционная среда.

1. Классификация операционной системы (ОС)

Операционная система (ОС) — система программ, предназначенная для управления ресурсами ЭВМ и процессами, которые используют эти ресурсы, а также для обеспечения пользовательского интерфейса.

Под ресурсом ЭВМ понимается любой логический или аппаратный компонент ЭВМ, а под процессом — последовательность действий, предписанных программой.

Любая ОС решает две наиболее важные задачи:

1) предоставление пользователю удобного интерфейса (пользовательский интерфейс) для управления аппаратными средствами и программным обеспечением ЭВМ;

2) повышение эффективности использования компьютера путем рационального управления его ресурсами (программный и аппаратно-программный интерфейсы).

Первая задача обусловлена сложностью управления компьютером на уровне машинного языка, особенно это касается ввода-вывода. Например, организация чтения блока данных с гибкого диска включает последовательное выполнение ряда команд с заданием таких параметров, как номер блока на диске, номер сектора на дорожке и др. Наличие же ОС сводит эту задачу к копированию логического объекта — файла — с одного носителя на другой в интуитивно понятном пользователю интерфейсе (например, Проводник ОС Windows). Аналогично операционная система, управляя другими аппаратными средствами, образуя на их основе в некотором смысле «виртуальную» машину, с которой взаимодействует пользователь.

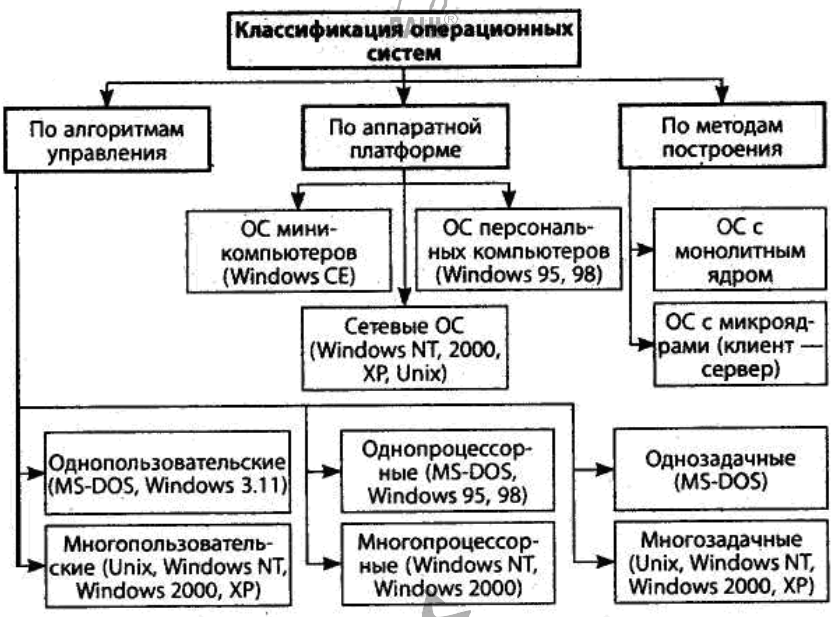


Рисунок 48. Классификация операционных систем

Наряду с этим ОС управляет всеми частями сложной системы компьютера, распределяя ресурсы компьютера (время процессора, память и данные) между конкурирующими процессами, обеспечивая при этом максимальную эффективность функционирования компьютера.

Операционные системы различают (рис. 48) по особенностям реализации внутренних алгоритмов управления основными ресурсами компьютера, особенностям использованных методов проектирования, типам аппаратных платформ, сферам применения и др.

Обобщенная характеристика современной ОС для ПК — сетевая, многопользовательская, многозадачная и даже многопроцессорная. Всем этим критериям соответствуют наиболее широко используемые ОС семейства Windows, выпускаемые компанией Microsoft для ПК начиная с 2003 г. Включение сетевого ядра в ОС персональных компьютеров в первую очередь связано с развитием и массовым использованием глобальной сети Internet. Однозадачные ОС выпол-

няют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером. Однозадачные ОС включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем.

Многозадачные ОС, кроме вышеперечисленных функций, управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства.

Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей. Появление многопользовательских ОС в первую очередь связано с решением проблемы разграничения прав доступа пользователей к аппаратным, программным ресурсам ПК, а также данным.

2. Компоненты ОС

Большинство ОС используют монолитное ядро, которое компонуется как одна программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую. Альтернативой являются ОС, построенные на базе микроядра, работающего в привилегированном режиме и выполняющего только минимум функций по управлению аппаратурой, в то время как функции ОС более высокого уровня выполняют специализированные компоненты ОС — серверы, работающие в пользовательском режиме. Такие ОС реализуют «клиент-серверную» модель взаимодействия прикладной программы и операционной системы, в которой все обращения пользовательской программы (клиента) к операционной системе обрабатываются специальной программой (сервером).

При таком построении ОС работает более медленно, так как часто выполняются переходы между привилегированным режимом и пользовательским, однако, система получается более гибкой — ее функции можно наращивать, модифицировать или сужать, добавляя, модифицируя или исключая серверы пользовательского режима.

Все операционные системы способны обеспечивать как пакетный, так и диалоговый режимы работы с пользователем. В пакетном режиме операционная система автоматически исполняет заданную последовательность команд. В диалоговом режиме ОС находится в ожидании команды пользователя и, получив ее, приступает к исполнению, а исполнив, возвращает отклик и ждет очередной команды. Диалоговый режим работы основан на использовании прерываний процессора. Опираясь на эти аппаратные прерывания, операционная система создает свой комплекс системных прерываний. Способность операционной системы прервать текущую работу и отреагировать на события, вызванные пользователем с помощью управляющих устройств, и определяет сущность диалогового режима работы.

С точки зрения пользователя, компетенция которого ограничивается грамотным применением имеющегося программного обеспечения, основными функциями операционных систем являются:

- обеспечение автоматического запуска;

- формирование интерфейса пользователя;
- организация и обслуживание файловой системы.

Автоматический запуск дисковых ОС обеспечивается записью на этапе инсталляции (установки) ОС программного кода в специальной (системной) области диска. Обращение к этому коду осуществляют программы BIOS, которые по окончании своей работы дают команду на загрузку и исполнение содержимого системного диска.

Интерфейсы пользователя, предоставляемые ОС пользователю, делятся на терминальные и графические. Терминальный интерфейс реализован в неграфических ОС, которые поддерживают интерфейс командной строки (MS-DOS). Основным устройством управления в данном случае является клавиатура. Управляющие команды вводят в поле командной строки, где их можно редактировать. Исполнение команды начинается после ее подтверждения, например, нажатием клавиши ENTER.

Графические операционные системы реализуют более сложный тип интерфейса, в котором в качестве устройства управления, кроме клавиатуры, можно использовать мышь или другое устройство позиционирования. Работа с графической ОС основана на взаимодействии активных и пассивных экранных элементов управления.

Первой наиболее полноценной графической ОС была ОС Windows 95, представленная компанией Microsoft в 1995 г.

Организация и обслуживание файловой системы являются одними из важнейших функций ОС, обеспечивающими упорядоченное хранение данных на магнитных и оптических носителях, а также доступ к этим данным.

Сетевое программное обеспечение является составной частью системного ПО и предназначено для управления общими ресурсами в распределенных вычислительных системах. Общими ресурсами, как правило, являются сетевые накопители на магнитных и оптических дисках, принтеры, сканеры и др. аппаратные средства. Кроме этого, к общим ресурсам относятся программы и данные.

К сетевому ПО относят ОС, поддерживающие работу компьютера в сетевых конфигурациях (так называемые сетевые ОС), а также отдельные сетевые программы (пакеты), используемые совместно с обычными — несетевыми ОС.

Сетевая операционная система составляет основу любой вычислительной сети. В узком смысле сетевая ОС — это операционная система отдельного компьютера, обеспечивающая ему возможность работать в сети. Сетевая ОС отдельного ПК включает несколько частей (рис. 49):

− средства управления локальными ресурсами компьютера реализуют функции ПК в локальном (изолированном от других ПК) режиме;

− средства предоставления собственных ресурсов и услуг в общее пользование (серверная часть сетевой ОС) обеспечивают обработку запросов удаленного доступа к собственной файловой системе и базе данных, управление очередями запросов удаленных пользователей к своим периферийным устройствам и т. д.;

− средства запроса доступа к удаленным ресурсам и услугам и их использование (клиентская часть сетевой ОС) формируют и перенаправляют в сеть запросы к удаленным ресурсам от приложений и пользователей;

− коммуникационные средства ОС реализуют обмен сообщениями в сети, обеспечивая адресацию сообщений и выбор маршрута передачи сообщений по сети.

В зависимости от функций, возлагаемых на конкретный компьютер, в его операционной системе может отсутствовать либо клиентская, либо серверная часть.

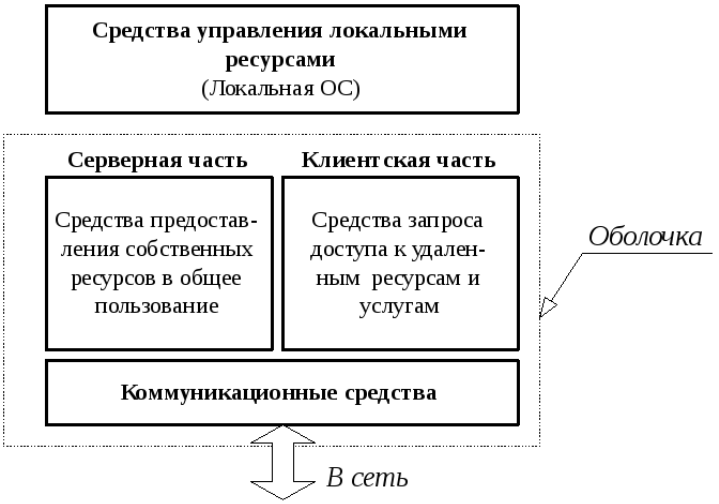


Рисунок 49. Структура сетевой операционной системы

Сервисные программы включают операционные среды, оболочки операционных систем и утилиты, которые предназначены для расширения возможностей операционной системы, изменения ее пользовательского и программного интерфейса, а также для предоставления дополнительных услуг по управлению ресурсами компьютера.

3. Операционная среда

Операционная среда — система, изменяющая и дополняющая как пользовательский, так и программный интерфейс. Операционная среда создает для пользователя и прикладных программ иллюзию работы в полноценной операционной системе, поскольку может полностью изменить интерфейс пользователя, часто ее называют операционной системой. Появление операционной среды обычно означает, что используемая операционная система не полностью удовлетворяет требованиям практики. Примерами операционных сред являются Windows 3.11 и Windows 3.11 for Work Groups (для рабочих групп), расширившая возможности ОС MS-DOS.

Оболочка операционной системы, в отличие от операционной среды, модифицирует только пользовательский интерфейс, предоставляя пользователю

качественно новый интерфейс по сравнению с реализуемым операционной системой.

Утилиты дополняют интерфейс пользователя и используются в тех случаях, когда программного обеспечения, представленного в ОС, недостаточно для обслуживания пользователем компьютера. Они обеспечивают реализацию следующих действий:

– обслуживание магнитных дисков (форматирование, дефрагментация, проверка поверхности и др.);

– обслуживание файлов и каталогов (поиск и восстановление удаленных файлов);

– предоставление информации о ресурсах компьютера;

– шифрование информации;

– защита от компьютерных вирусов;

– архивация файлов и др.

Существуют отдельные утилиты, используемые для решения одного из перечисленных действий, и многофункциональные утилиты.

Контрольные вопросы:

1. Охарактеризуйте место операционной системы в программном обеспечении компьютеров, компьютерных систем и сетей.
2. В чем заключается основное назначение операционной системы?
3. Перечислите основные функции операционной системы.
4. Дайте определение понятия компьютерные ресурсы.
5. Дайте определение понятия архитектура операционных систем.
6. Перечислите классификационные признаки операционной системы.
7. Охарактеризуйте виды интерфейсов операционных систем.
8. Что такое операционная среда?

Тема 6. Основные принципы управления ресурсами и организации доступа к этим ресурсам

Лекция 21. Архитектура компьютерных сетей

План:

1. Типовой состав оборудования локальной сети.
2. Среда передачи данных внутри сети.
3. Сетевые адаптеры.
4. Физическая структуризация локальной сети. Повторители и концентраторы.
5. Логическая структуризация сети. Мосты и коммутаторы.
6. Маршрутизаторы.
7. Топологии компьютерной сети.
8. Классификация ВС.

1. Типовой состав оборудования локальной сети

Фрагмент вычислительной сети (рис. 50) включает основные типы коммуникационного оборудования, применяемого сегодня для образования локальных сетей и соединения их через глобальные связи друг с другом. Для построения локальных связей между компьютерами используются различные виды кабельных систем, сетевые адаптеры, концентраторы-повторители, мосты, коммутаторы и маршрутизаторы. Для подключения локальных сетей к глобальным связям используются специальные выходы (WAN-порты) мостов и маршрутизаторов, а также аппаратура передачи данных по длинным линиям — модемы (при работе по аналоговым линиям) или же устройства подключения к цифровым каналам (ТА — терминальные адаптеры сетей ISDN, устройства обслуживания цифровых выделенных каналов типа CSU/DSU и т. п.).

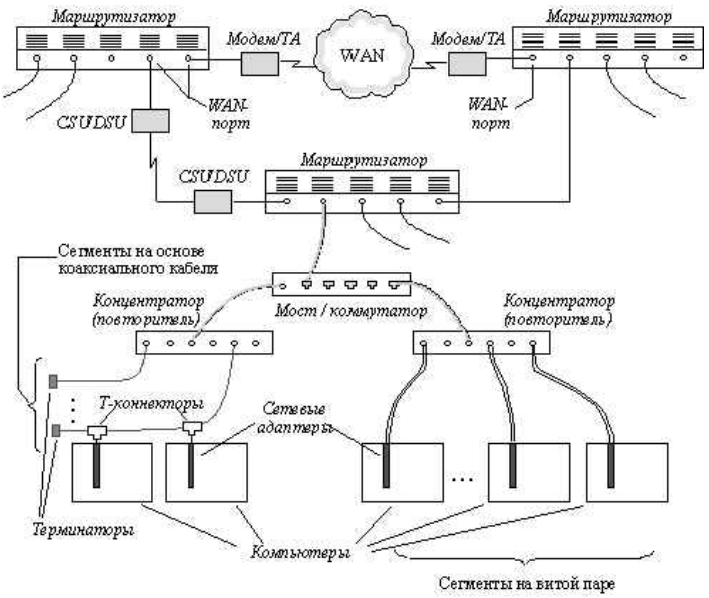


Рисунок 50. Фрагмент сети

Архитектура включает следующие компоненты:

- топология — структура связей элементов в сети;
- протоколы — правила взаимодействия функциональных элементов сети;
- интерфейсы — средства сопряжения функциональных элементов-узлов и программных модулей;
- технические средства — устройства, которые обеспечивают объединения компьютеров в единую сеть (сетевые адаптеры, концентраторы, кабели и т. д.);
- сетевые программные средства — управляют работой сети и предоставляют пользовательский интерфейс (сетевые ОС, вспомогательные служебные программы).

2. Среда передачи данных внутри сети

В настоящее время используются различные виды сред для передачи данных внутри локальной сети. Их условная классификация показана на рис. 51.

Для построения локальных связей в вычислительных сетях используются различные виды *кабелей — коаксиальный* кабель, кабель на основе *экранированной* и *неэкранированной витой пары* и *оптоволоконный* кабель. Наиболее популярным видом среды передачи данных на небольшие расстояния (до 100 м) становится неэкранированная витая пара, которая включена практически во все современные стандарты и технологии локальных сетей и обеспечивает пропускную способность до 100 Мб/с (на кабелях категории 5). Оптоволоконный кабель широко применяется как для построения локальных связей, так и для образования магистралей глобальных сетей. Оптоволоконный кабель может обеспечить очень высокую пропускную способность канала (до нескольких Гб/с) и передачу на значительные расстояния (до нескольких десятков километров без промежуточного усиления сигнала).

Рисунок 51. Среда передачи данных

В качестве среды передачи данных в вычислительных сетях используются также электромагнитные волны различных частот — КВ, УКВ, СВЧ. Однако пока в локальных сетях радиосвязь используется только в тех случаях, когда оказывается невозможной прокладка кабеля, например в зданиях, являющихся памятниками архитектуры. Для построения глобальных каналов этот вид среды передачи данных используется шире — на нем построены спутниковые каналы

связи и наземные радиорелейные каналы, работающие в зонах прямой видимости в СВЧ-диапазонах.

Структурированная кабельная система (Structured Cabling System, *SCS*) — это набор коммутационных элементов (кабелей, разъемов, коннекторов, кроссовых панелей и шкафов), а также методика их совместного использования, которая позволяет создавать регулярные, легко расширяемые структуры связей в вычислительных сетях.

3. Сетевые адаптеры

Сетевой адаптер (Network Interface Card, *NIC*) — это периферийное устройство компьютера, непосредственно взаимодействующее со средой передачи данных, которая прямо или через другое коммуникационное оборудование связывает его с другими компьютерами. Это устройство решает задачи надежного обмена двоичными данными, представленными соответствующими электромагнитными сигналами, по внешним линиям связи. Как и любой контроллер компьютера, сетевой адаптер работает под управлением драйвера операционной системы, и распределение функций между сетевым адаптером и драйвером может изменяться от реализации к реализации.

В первых локальных сетях сетевой адаптер с сегментом коаксиального кабеля представлял собой весь спектр коммуникационного оборудования, с помощью которого организовывалось взаимодействие компьютеров. Сетевой адаптер компьютера-отправителя непосредственно по кабелю взаимодействовал с сетевым адаптером компьютера-получателя. В большинстве современных стандартов для локальных сетей предполагается, что между сетевыми адаптерами взаимодействующих компьютеров устанавливается специальное коммуникационное устройство (концентратор, мост, коммутатор или маршрутизатор), которое берет на себя некоторые функции по управлению потоком данных.

Сетевые адаптеры различаются по типу и разрядности используемой в компьютере внутренней шины данных — ISA, EISA, PCI, MCA.

Сетевые адаптеры различаются также по типу принятой в сети сетевой технологии — Ethernet, Token Ring, FDDI и т. п. Как правило, конкретная модель сетевого адаптера работает по определенной сетевой технологии (например, Ethernet). В связи с тем, что для каждой технологии сейчас имеется возможность использования различных сред передачи данных, сетевой адаптер может поддерживать как одну, так и одновременно несколько сред. В случае, когда сетевой адаптер поддерживает только одну среду передачи данных, а необходимо использовать другую, применяются трансиверы и конверторы.

Трансивер (приемопередатчик, transmitter + receiver) — это часть сетевого адаптера, его оконечное устройство, выходящее на кабель. В первом стандарте Ethernet, работающем на толстом коаксиале, трансивер располагался непосредственно на кабеле и связывался с остальной частью адаптера, располагавшейся внутри компьютера, с помощью интерфейса AUI (Attachment Unit Interface). В других вариантах Ethernet'а оказалось удобным выпускать сетевые адаптеры (да и другие коммуникационные устройства) с портом AUI, к которому можно присоединить трансивер для требуемой среды.

4. Физическая структуризация локальной сети. Повторители и концентраторы

Для построения простейшей односегментной сети достаточно иметь сетевые адаптеры и кабель подходящего типа. Но даже в этом простом случае часто используются дополнительные устройства — *повторители сигналов*, позволяющие преодолеть ограничения на максимальную длину кабельного сегмента.

Основная функция *повторителя* (*repeater*), как это следует из его названия, — повторение сигналов, поступающих на один из его портов, на всех остальных портах (Ethernet) или на следующем в логическом кольце порте (Token Ring, FDDI) синхронно с сигналами-оригиналами. Повторитель улучшает электрические характеристики сигналов и их синхронность, и за счет этого появляется возможность увеличивать общую длину кабеля между самыми удаленными в сети станциями.

Многопортовый повторитель часто называют *концентратором* (*hub*, *concentrator*), что отражает тот факт, что данное устройство реализует не только функцию повторения сигналов, но и концентрирует в одном центральном устройстве функции объединения компьютеров в сеть. Практически во всех современных сетевых стандартах концентратор является необходимым элементом сети, соединяющим отдельные компьютеры в сеть. Отрезки кабеля, соединяющие два компьютера или какие-либо два других сетевых устройства, называются физическими сегментами. Таким образом, концентраторы и повторители, которые используются для добавления новых физических сегментов, являются средством физической структуризации сети.

Концентраторы образуют из отдельных физических отрезков кабеля общую среду передачи данных — логический сегмент (рис. 52).

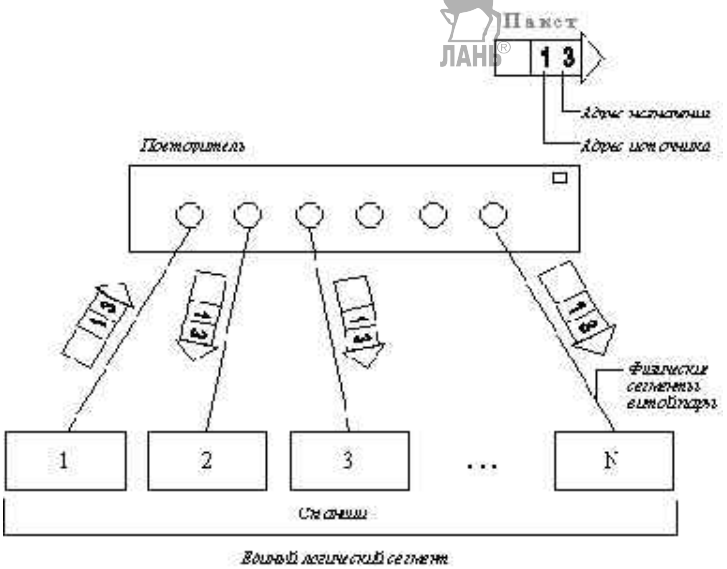


Рисунок 52. Повторитель Ethernet синхронно повторяет биты кадра на всех портах

Логический сегмент также называют доменом коллизий, поскольку при попытке одновременной передачи данных любых двух компьютеров этого сегмента, хотя бы и принадлежащих разным физическим сегментам, возникает блокировка передающей среды. Следует особо подчеркнуть, что какую бы сложную структуру ни образовывали концентраторы, например путем иерархического соединения (рис. 53), все компьютеры, подключенные к ним, образуют единый логический сегмент, в котором любая пара взаимодействующих компьютеров полностью блокирует возможность обмена данными для других компьютеров.

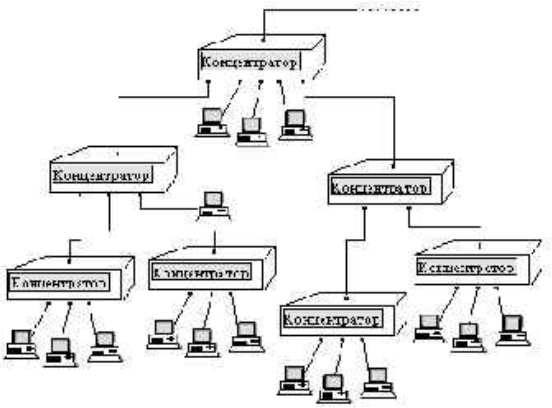


Рисунок 53. Логический сегмент, построенный с использованием концентраторов

Появление устройств, централизующих соединения между отдельными сетевыми устройствами, потенциально позволяет улучшить управляемость сети и ее эксплуатационные характеристики (модифицируемость, ремонтопригодность и т. п.).

5. Логическая структуризация сети. Мосты и коммутаторы

Несмотря на появление новых дополнительных возможностей основной функцией концентраторов остается передача пакетов по общей разделяемой среде. Коллективное использование многими компьютерами общей кабельной системы в режиме разделения времени приводит к существенному снижению производительности сети при интенсивном трафике. Общая среда перестает справляться с потоком передаваемых кадров, и в сети возникает очередь компьютеров, ожидающих доступа. Это явление характерно для всех технологий, использующих разделяемые среды передачи данных, независимо от используемых алгоритмов доступа.

Поэтому сети, построенные на основе концентраторов, не могут расширяться в требуемых пределах — при определенном количестве компьютеров в сети или при появлении новых приложений всегда происходит насыщение передающей среды, и задержки в ее работе становятся недопустимыми. Эта проблема может быть решена путем логической структуризации сети с помощью *мостов, коммутаторов и маршрутизаторов*.

Мост (*bridge*), а также его быстродействующий функциональный аналог — *коммутатор* (*switching hub*) делят общую среду передачи данных на ло-

гические сегменты. Логический сегмент образуется путем объединения нескольких физических сегментов (отрезков кабеля) с помощью одного или нескольких концентраторов. Каждый логический сегмент подключается к отдельному порту моста/коммутатора (рис. 54). При поступлении кадра на какой-либо из портов мост/коммутатор повторяет этот кадр, но не на всех портах, как это делает концентратор, а только на том порту, к которому подключен сегмент, содержащий компьютер-адресат.

Разница между мостом и коммутатором состоит в том, что мост в каждый момент времени может осуществлять передачу кадров только между одной парой портов, а коммутатор одновременно поддерживает потоки данных между всеми своими портами. Другими словами, мост передает кадры последовательно, а коммутатор параллельно. Следует отметить, что в последнее время локальные мосты полностью вытеснены коммутаторами. Мосты используются только для связи локальных сетей с глобальными, т. е. как средства удаленного доступа, поскольку в этом случае необходимость в параллельной передаче между несколькими парами портов просто не возникает.

При работе коммутатора среда передачи данных каждого логического сегмента остается общей только для тех компьютеров, которые подключены к этому сегменту непосредственно. Коммутатор осуществляет связь сред передачи данных различных логических сегментов. Он передает кадры между логическими сегментами только при необходимости, т. е. только тогда, когда взаимодействующие компьютеры находятся в разных сегментах.

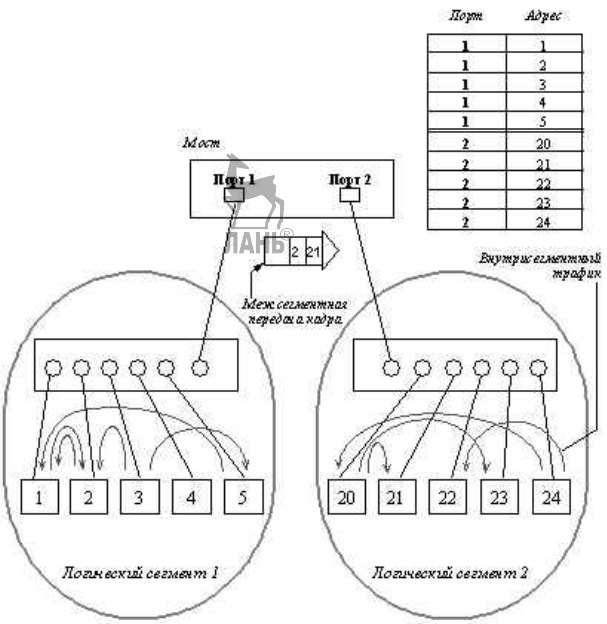


Рисунок 54. Разделение сети на логические сегменты

Деление сети на логические сегменты улучшает производительность сети, если в сети имеются группы компьютеров, преимущественно обменивающиеся информацией между собой.

Коммутаторы принимают решение о том, на какой порт нужно передать кадр, анализируя адрес назначения, помещенный в кадре, а также на основании информации о принадлежности того или иного компьютера определенному сегменту, подключенному к одному из портов коммутатора, т. е. на основании информации о конфигурации сети. Для того, чтобы собрать и обработать информацию о конфигурации подключенных к нему сегментов, коммутатор должен пройти стадию «обучения», т. е. самостоятельно проделать некоторую предварительную работу по изучению проходящего через него трафика. Определение принадлежности компьютеров сегментам возможно за счет наличия в кадре не только адреса назначения, но и адреса источника, сгенерировавшего пакет. Используя информацию об адресе источника, коммутатор устанавливает соответствие между номерами портов и адресами компьютеров. В процессе изучения сети мост/коммутатор просто передает появляющиеся на входах его портов кадры на все остальные порты, работая некоторое время повторителем. После того, как мост/коммутатор узнает о принадлежности адресов сегментам, он начинает передавать кадры между портами только в случае межсегментной передачи. Если, уже после завершения обучения, на входе коммутатора вдруг появится кадр с неизвестным адресом назначения, то этот кадр будет повторен на всех портах.

Мосты/коммутаторы, работающие описанным способом, обычно называются прозрачными (transparent), поскольку появление таких мостов/коммутаторов в сети совершенно незаметно для ее конечных узлов. Это позволяет не изменять их программное обеспечение при переходе от простых конфигураций, использующих только концентраторы, к более сложным, сегментированным.

Существует и другой класс мостов/коммутаторов, передающих кадры между сегментами на основе полной информации о межсегментном маршруте. Эту информацию записывает в кадр станция-источник кадра, поэтому говорят, что такие устройства реализуют алгоритм маршрутизации от источника (source routing). При использовании мостов/коммутаторов с маршрутизацией от источника конечные узлы должны быть в курсе деления сети на сегменты и сетевые адаптеры, в этом случае должны в своем программном обеспечении иметь компонент, занимающийся выбором маршрута кадров.

6. Маршрутизаторы

Маршрутизатор (*router*) позволяет организовывать в сети избыточные связи, образующие петли. Он справляется с этой задачей за счет того, что принимает решение о передаче пакетов на основании более полной информации о графе связей в сети, чем мост или коммутатор. Маршрутизатор имеет в своем распоряжении базу топологической информации, которая говорит ему, например, о том, между какими подсетями общей сети имеются связи и в каком состоянии (работоспособном или нет) они находятся. Имея такую карту сети, маршрутизатор может выбрать один из нескольких возможных маршрутов доставки пакета адресату. В данном случае под маршрутом понимают последовательность прохождения пакетом маршрутизаторов. Например, на рис. 55 для связи станций L2 сети LAN1 и L1 сети LAN6 имеется два маршрута: M1-M5-M7 и M1-M6-M7.

В отличие от моста/коммутатора, который не знает, как связаны сегменты друг с другом за пределами его портов, маршрутизатор видит всю картину связей подсетей друг с другом, поэтому он может выбрать правильный маршрут и

при наличии нескольких альтернативных маршрутов. Решение о выборе того или иного маршрута принимается каждым маршрутизатором, через который проходит сообщение.

Для того чтобы составить карту связей в сети, маршрутизаторы обмениваются специальными служебными сообщениями, в которых содержится информация о тех связях между подсетями, о которых они знают (эти подсети подключены к ним непосредственно, или же они узнали эту информацию от других маршрутизаторов).

Маршрутизаторы не только объединяют сети, но и надежно защищают их друг от друга. Причем эта изоляция осуществляется гораздо проще и надежнее, чем с помощью мостов/коммутаторов. Например, при поступлении кадра с неправильным адресом мост/коммутатор обязан повторить его на всех своих портах, что делает сеть незащищенной от некорректно работающего узла. Маршрутизатор же в таком случае просто отказывается передавать «неправильный» пакет дальше, изолируя дефектный узел от остальной сети.

Кроме того, маршрутизатор предоставляет администратору удобные средства фильтрации потока сообщений за счет того, что сам распознает многие поля служебной информации в пакете и позволяет их именовать понятным администратору образом. Нужно заметить, что некоторые мосты/коммутаторы также способны выполнять функции гибкой фильтрации, но задавать условия фильтрации администратор сети должен сам в двоичном формате, что достаточно сложно.

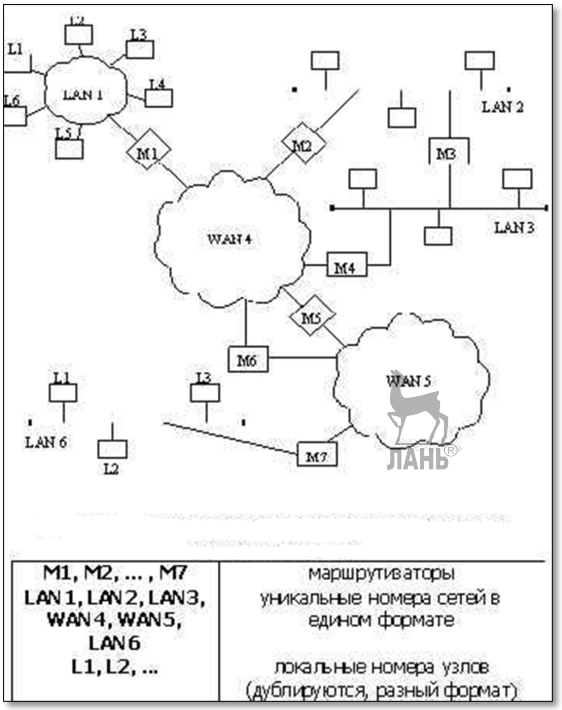


Рисунок 55. Структура интерсети, построенной на основе маршрутизаторов

7. Топологии компьютерной сети

Под топологией (компоновкой, конфигурацией, структурой) компьютерной сети обычно понимаются физическое расположение компьютеров сети один относительно одного и способ соединения их линиями связи. Важно отметить, что понятие топологии относится в первую очередь к локальным сетям, в которых структуру связей можно легко проследить. В глобальных сетях структура связей обычно спрятана от пользователей. Топология определяет требования к оборудованию, тип используемого кабеля, возможные и наиболее удобные методы управления обменом, надежность работы, возможности расширения сети.

Существуют три основные топологии сети:

1. Сетевая топология «шина» (bus), при которой все компьютеры параллельно подключаются к одной линии связи, и информация от каждого компьютера одновременно передается всем другим компьютерам (рис. 56).

2. Сетевая топология «звезда» (star), при которой к одному центральному компьютеру присоединяются другие периферийные компьютеры, причем каждый из них использует свою отдельную линию связи (рис. 57).

3. Сетевая топология «кольцо» (ring), при которой каждый компьютер передает информацию всегда только одному компьютеру, следующему в цепочке, а получает информацию только от предыдущего компьютера в цепочке, и эта цепочка замкнута в «кольцо» (рис. 58).

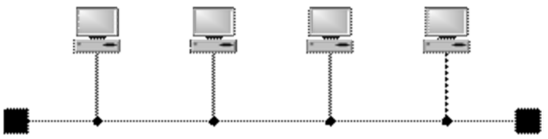


Рисунок 56. Сетевая топология «шина»

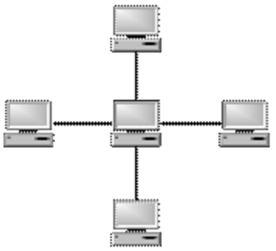


Рисунок 57. Сетевая топология «звезда»

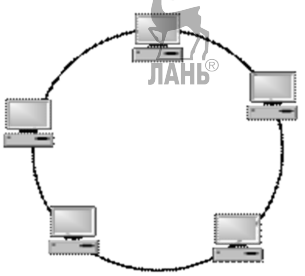


Рисунок 58. Сетевая топология «кольцо»

На практике нередко используют и комбинации базовой топологии, но большинство сетей ориентированы именно на эти три. Рассмотрим теперь коротко особенности перечисленной сетевой топологии.

Топология «шина» (или, как ее еще называют, «общая шина») самой своей структурой допускает идентичность сетевого оборудования компьютеров, а также равноправие всех абонентов. При таком соединении компьютеры могут передавать только по очереди, потому что линия связи единственная. В противном случае переданная информация будет искажаться в результате наложения (конфликта, коллизии). В топологии «шина» отсутствует центральный абонент, через которого передается вся информация, которая увеличивает ее надежность (ведь при отказе любого центра перестает функционировать вся управляемая этим центром система). Добавление новых абонентов в шину достаточно простое и обычно возможно даже во время работы сети. В большинстве случаев при использовании шины нужно минимальное количество соединительного кабеля по сравнению с другой топологией. Правда, нужно учесть, что к каждому компьютеру (кроме двух крайних) подходит два кабеля, что не всегда удобно. Потому что разрешение возможных конфликтов в этом случае ложится на сетевое оборудование каждого отдельного абонента, аппаратура сетевого адаптера при топологии «шина» выходит сложнее, чем при другой топологии. Однако благодаря широкому распространению сетей с топологией «шина» (Ethernet, Arcnet) стоимость сетевого оборудования выходит не слишком высокой.

Шине не страшны отказы отдельных компьютеров, потому что все другие компьютеры сети могут нормально продолжать обмен.

Топология «звезда» — это топология с явно выделенным центром, к которому подключаются все другие абоненты. Весь обмен информацией идет исключительно через центральный компьютер, на который ложится очень большая нагрузка, потому ничем другим, кроме сети, он заниматься не может. Понятно, что сетевое оборудование центрального абонента должно быть существенно более сложным, чем оборудование периферийных абонентов. О равноправии абонентов в этом случае говорить не придется. Как правило, именно центральный компьютер является самым мощным, и именно на него возлагают все функции по управлению обменом. Никакие конфликты в сети с топологией «звезда» в принципе невозможны, потому что управление полностью централизовано. Если говорить о стойкости звезды к отказам компьютеров, то выход из строя периферийного компьютера никак не отражается на функционировании части сети, которая осталась, зато любой отказ центрального компьютера делает сеть полностью неработоспособной. Поэтому должны приниматься специальные мероприятия по повышению надежности центрального компьютера и его сетевой аппаратуры. Обрыв любого кабеля или короткое замыкание в нем при топологии «звезда» нарушает обмен только с одним компьютером, а все другие компьютеры могут нормально продолжать работу. Звезда, показанная на рис. 58, называется активной, или настоящей «звездой». Существует также топология, внешне похожая на звезду, которая называется пассивной звездой (рис. 59). Сейчас она распространена намного больше, чем активная звезда. Достаточно сказать, что она используется в самой популярной на сегодняшний день сети Ethernet.

Рисунок 59. Топология «пассивная звезда»

В центре сети с данной топологией содержится не компьютер, а концентратор, или хаб (hub), что выполняет ту же функцию, что и репитер. Он возобновляет сигналы, которые поступают, и пересылает их в другие линии связи. Хотя схема прокладки кабелей подобна настоящей, или активной, звезде, фактически мы имеем дело с шинной топологией, потому что информация от каждого компьютера одновременно передается ко всем другим компьютерам, а центрального абонента не существует. Естественно, пассивная звезда выходит дороже обычной шины, потому что в этом случае обязательно нужен еще и концентратор. Однако она предоставляет целый ряд дополнительных возможностей, связанных с преимуществами звезды. Именно поэтому в последнее время пассивная звезда все больше вытесняет настоящую звезду, которая считается малоперспективной топологией.

Топология «кольцо» — это топология, в которой каждый компьютер соединен линиями связи только с двумя другими: от одного он только получает информацию, а другому только передает. На каждой линии связи, как и в случае звезды, работают только один передатчик и один приемник. Это позволяет отказаться от применения внешних терминаторов. Особенность кольца заключается в том, что каждый компьютер ретранслирует (возобновляет) сигнал, т. е. выступает в роли репитера, потому затухание сигнала во всем кольце не имеет никакого значения, важно только затухание между соседними компьютерами кольца. Четко выделенного центра в этом случае нет, все компьютеры могут быть одинаковыми. Однако достаточно часто в кольце выделяется специальный абонент, который управляет обменом или контролирует обмен. Понятно, что наличие такого управляющего абонента снижает надежность сети, потому что выход его из строя сразу же парализует весь обмен.

Недостатком кольца (в сравнении со звездой) можно считать то, что к каждому компьютеру сети необходимо подвести два кабеля.

Иногда топология «кольцо» выполняется на основе двух кольцевых линий связи, которые передают информацию в противоположных направлениях. Цель подобного решения – увеличение (в идеале вдвое) скорости передачи информации. К тому же при повреждении одного из кабелей сеть может работать с другим кабелем (правда, предельная скорость уменьшится).

Кроме трех рассмотренных основных, базовых топологий нередко применяется также сетевая топология «**дерево» (tree),** которую можно рассматривать как комбинацию нескольких звезд. Как и в случае звезды, дерево может быть активным, или настоящим (рис. 60), и пассивным (рис. 61). При активном дереве в центрах объединения нескольких линий связи находятся центральные компьютеры, а при пассивном — концентраторы (хабы).

Рисунок 60. Топология «активное дерево»

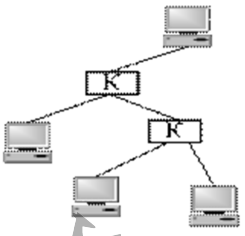


Рисунок 61. Топология «пассивное дерево». К — концентраторы

8. Классификация ВС

Концепция вычислительных сетей является логическим результатом эволюции компьютерной технологии. По мере эволюции вычислительных систем сформировались следующие разновидности архитектуры компьютерных сетей:

– одноранговая архитектура;

– классическая архитектура «клиент — сервер»;

– архитектура «клиент — сервер» на основе Web-технологии.

Правильно выбранная архитектура компьютерной сети позволяет достигнуть выдвинутых требований по общей производительности, надежности защиты сетевых ресурсов, гибкости настройки сети, а также минимизации денежных затрат на ее построение и администрирование.

Одноранговая сеть — это сеть, в которой отсутствует выделенный сервер, а клиентские компьютеры могут использовать ресурсы друг друга. В одноранговой сети все компьютеры равноправны: нет иерархии среди компьютеров и нет выделенного сервера. Каждый компьютер функционирует и как клиент, и как сервер, нет отдельного компьютера, ответственного за администрирование всей сети. Все пользователи самостоятельно решают, что на своем компьютере можно сделать общедоступным по сети.

Централизованно управлять защитой в одноранговой сети сложно, так как каждый пользователь устанавливает ее самостоятельно, да и «общие» ресурсы могут находиться на всех компьютерах, а не только на центральном сервере. Такая ситуация представляет серьезную угрозу для всей сети.

Явные недостатки, свойственные одноранговой архитектуре, и развитие инструментальных средств привели к появлению вычислительных систем с архитектурой «клиент — сервер».

Клиент-серверная технология — это стиль работы приложений, где клиентский процесс запрашивает обслуживание у процесса сервера. Сервер — это

программа, предоставляющая доступ к каким-либо услугам, например к электронной почте, файлам, ftp, Web или данным (в качестве сервера баз данных). Клиент — это приложение, которое соединяется с сервером, чтобы воспользоваться предоставляемыми им услугами.

Контрольные вопросы:

1. Что такое компьютерная сеть?
2. Что необходимо для создания компьютерных сетей?
3. Какие сети называются одноранговыми?
4. Каковы основные компоненты локальной сети?
5. Какие вы знаете топологии сетей?
6. Какие существуют виды кабелей для объединения компьютеров в сеть?
7. Что такое концентратор?
8. Что такое маршрутизатор?

ЛАНЬ®

Лекция 22. Основные принципы управления ресурсами и организации доступа к этим ресурсам

План:

1. Системные основы управления ресурсами.
2. ОС как средство управления ресурсами.
3. Управление процессами.

1. Системные основы управления ресурсами

В системе управления ресурсами выделяют следующие, общие для всех ресурсов, компоненты (подсистемы):

- интерфейс прикладного программирования;
- пользовательский интерфейс;
- защита данных;
- администрирование.

Интерфейс прикладного программирования

Обращение прикладного программиста к системным функциям происходит в двух случаях:

- когда для выполнения тех или иных действий прикладному программисту не хватает полномочий, и ему требуется статус операционной системы (в большинстве современных операционных систем это любые действия, связанные с управлением аппаратурой);
- когда использование системных функций помогает упростить написание приложения (это использование стандартных, часто используемых функций, таких как обработка строк, построение изображений и т. д.).

Здесь следует отметить, что использование системных функций может осуществляться косвенно. Так, например, в C++ Builder имеется много встроенных в Builder функций, которые сами используют системные функции. Такой вариант удобнее тем, что адаптированный вариант функции обычно более удобен, чем чисто системные вызовы.

ЛАНЬ®

Возможности операционной системы доступны программисту в виде набора функций, называющегося *интерфейсом прикладного программирования* (Application Programming Interface, API).

Для прикладных программистов все особенности конкретной операционной системы представляются особенностями ее API. Именно поэтому операционные системы с различной внутренней организацией, но с одинаковым набором функций API представляются прикладному программисту одной системой. Это упрощает стандартизацию операционных систем и улучшает переносимость приложений между различными системами.

Приложения обращаются к функциям API с помощью системных вызовов:

1. Операционной системе передается запрос, состоящий из идентификатора функции и данных. Этот запрос может помещаться в стек, в регистры или в область памяти.

2. Затем управление передается операционной системе, которая выполняет требуемую функцию и помещает ее результаты в стек, в регистры или в область памяти.

3. Если функция выполнена с ошибкой, результат включает индикацию ошибок.

4. Управление возвращается прикладной программе, которая забирает сохраненные системой результаты выполнения функции и анализирует ошибки, если они возникли.

Способ реализации системных вызовов зависит от реализации конкретной операционной системы, от аппаратной платформы и от языка, на котором написано приложение.

Пользовательский интерфейс

Помимо удобного интерфейса для прикладного программиста, операционная система должна быть удобной и для людей, работающих с ней. Это могут быть конечные пользователи, администраторы, программисты.

В ранних операционных системах пакетного режима все функции пользовательского интерфейса сводились к *командной строке*, которая к тому же часто вводилась с перфокарты.

Современные операционные системы поддерживают развитые функции пользовательского интерфейса. Они предназначены для интерактивной работы за терминалами двух типов:

- алфавитно-цифровыми;
- графическими.

При работе за *алфавитно-цифровым терминалом* пользователь имеет в своем распоряжении систему команд, которая отражает функциональные возможности конкретной операционной системы. Обычно командный язык операционной системы позволяет:

- запускать и останавливать приложения;
- выполнять различные операции с файлами и каталогами;
- получать информацию о состоянии операционной системы (активные процессы, объем свободной оперативной и дисковой памяти, загрузка процессора и т. д.);

− администрировать систему.

Команды могут вводиться не только с терминала в интерактивном режиме, но и из командного (пакетного) файла.

При использовании *графического интерфейса* ввод команд упрощается — для выполнения многих команд достаточно выполнить некоторые действия мышью (хотя следует отметить, что для некоторых пользователей проще ввести командную строку, чем «кликать» мышью).

Защита данных и администрирование

Безопасность данных в операционной системе определяется:

− средствами, направленными на защиту от сбоев и отказов аппаратуры и от ошибок программного обеспечения;

− средствами защиты от несанкционированного доступа к данным.

В последнем случае принято различать защиту от ошибочного и от злонамеренного доступа.

Первым уровнем защиты данных от несанкционированного доступа является *процедура входа в систему*. Операционная система должна убедиться, что в систему пытается войти пользователь, вход которого разрешен администратором.

Функции защиты операционной системы непосредственно связаны с *функциями администрирования*:

− определение прав пользователей при обращении их к различным ресурсам системы (файлам, каталогам, принтерам, сканерам и т. д.);

− ограничение возможностей пользователей при выполнении некоторых системно важных действий (завершение работы, установка системного времени, завершение чужих процессов, создание новых учетных записей, изменять права доступа к ресурсам и т. д.);

− исключение некоторых возможностей пользовательского интерфейса, например скрытие некоторых пунктов системного меню.

Важным средством защиты данных являются функции *аудита операционной системы*. Они заключаются в фиксации всех событий, важных для безопасности системы. Фиксируются такие события, как удачные и неудачные попытки входа в систему, операции доступа к выделенным каталогам и файлам, принтерам и другим устройствам.

Список событий, которые необходимо фиксировать, определяет администратор системы.

Для повышения отказоустойчивости операционной системы во многих системах используются *функции резервирования*. Для этого система может поддерживать несколько копий данных на различных носителях, избыточное количество принтеров и других устройств ввода-вывода. При отказе одного из таких устройств система должна быстро и прозрачно для пользователя выполнить переконфигурацию и продолжить работу с уцелевшими устройствами.

Особым случаем является использование нескольких процессоров, когда система продолжает работу при отказе одного или нескольких процессоров, хотя и с меньшей производительностью. Следует отметить, что многие мультипроцессорные операционные системы прекращают работу при отказе одного из процессоров.

Для повышения отказоустойчивости системы используются также специальные утилиты, с помощью которых администратор может регулярно выполнять операции резервного копирования для обеспечения быстрого восстановления важных для системы данных.

2. ОС как средство управления ресурсами

Операционная система не только предоставляет удобный *интерфейс* пользователям и программистам к аппаратным ресурсам компьютера. Она является также механизмом, распределяющим эти ресурсы.

К числу **основных ресурсов**, управляемых средствами операционной системы, можно отнести следующие:

– процессы;

– память;

– таймеры;

– наборы данных;

– иные (накопители на магнитных дисках и лентах, другие внешние накопители, принтеры, сетевые устройства и др.).

Процесс (задача) представляет собой базовое понятие для большинства современных операционных систем и часто определяется как программа в стадии выполнения. Полагаем, необходимо четкое разграничение понятий «программа» и «процесс».

Программа — это статический объект, представляющий собой файл с кодами и данными.

Процесс — это динамический объект, который возникает в оперативной памяти после того, как пользователь или операционная система запустят программу на выполнение. В этом случае создается новая единица вычислительной работы. Это, конечно, очень грубое представление, но оно достаточно для решения задачи распределения ресурсов.

Цель задачи управления ресурсами — наиболее эффективное управление ресурсами и их использование. Например, мультипрограммная операционная система организует одновременное выполнение нескольких процессов на одном компьютере, поочередно переключая процессор с одного процесса на другой, исключая простои процессора из-за обращения процессов к вводу-выводу. Задачей операционной системы в этом случае является также отслеживание и разрешение конфликтов, возникающих при обращении нескольких процессов к одному и тому же ресурсу (устройству ввода-вывода, данным).

Критерий эффективности, в соответствии с которым операционная система осуществляет управление ресурсами, зависит от назначения ОС. В одних системах важна пропускная способность вычислительной системы, в других — время реакции. Управление ресурсами организуют в соответствии с выбранными критериями эффективности.

В задачу управления любыми ресурсами входит четыре типа подзадач:

− планирование ресурса — определение, какому процессу, когда и в каком количестве (если ресурс может выделяться частями) следует выделить данный ресурс;

− удовлетворение запроса на ресурс;

− отслеживание состояния и учет использования ресурса — поддержание оперативной информации о том, занят или свободен ресурс, и какая доля ресурса уже распределена;

− разрешение конфликтов между запросами ресурсов.

Алгоритмы, реализованные в каждой операционной системе для реализации этих подзадач, определяют облик операционной системы в целом и отдельные ее характеристики — производительность, область применения, пользовательский интерфейс и т. д.

Сложность задачи организации эффективного использования ресурсов несколькими процессами во многом определяется случайным характером возникновения запросов на потребление ресурсов. В мультипрограммной системе образуются очереди заявок к разделяемым ресурсам компьютера (процессору, странице памяти, принтеру, диску и т. д.) от одновременно выполняемых программ.

Управление ресурсами составляет важнейшую часть функций любой операционной системы, особенно мультипрограммной. Большинство функций управления ресурсами выполняются непосредственно операционной системой и прикладному программисту обычно недоступны.

Функции операционной системы обычно группируются либо в соответствии с типами ресурсов, которыми управляет операционная система, либо в соответствии со специфическими задачами, применимыми ко всем ресурсам. Иногда такие группы функций называют ***подсистемами***.

Наиболее важными подсистемами являются подсистемы управления:

− процессами;

− памятью;

− файлами;

− внешними устройствами.

3. Управление процессами

Одной из важнейших подсистем операционной системы является подсистема управления процессами. Эта подсистема самым непосредственным образом влияет на работоспособность вычислительной системы.

Для каждого вновь создаваемого процесса операционная система генерирует системные структуры, которые содержат для каждого процесса данные о потребностях и о фактически выделенных ресурсах. С этой точки зрения процесс можно определить как некоторую заявку на потребление системных ресурсов.

Для выполнения процесса необходимы:

− область оперативной памяти, в которой будут размещены коды и данные процесса;

− необходимое количество процессорного времени.

Дополнительно процессу может понадобиться доступ к таким ресурсам, как файлы и устройства ввода-вывода.

Часто в информационные структуры процесса включаются вспомогательные данные:

– историю пребывания процесса в системе (относительные доли вычислений и ввода-вывода);

– его текущее состояние (активное или заблокированное);

– уровень привилегированности процесса (значение приоритета).

Эти дополнительные данные могут учитываться системой при принятии решения о предоставлении ресурсов процессу.

В мультипрограммной операционной системе обычно одновременно существует несколько процессов.

Процессы, порождаемые по инициативе пользователей и их приложений, называют *пользовательскими*.

Системные процессы обычно порождаются операционной системой для выполнения своих функций.

Различные процессы часто одновременно претендуют на одни и те же ресурсы, поэтому важнейшей задачей операционной системы является поддержание **очередей заявок процессов на ресурсы** (процессор, принтер, последовательный порт и т. д.).

Не менее важной задачей операционной системы является **защита ресурсов**, выделенных одному процессу, от притязаний других. Наиболее тщательно должны защищаться области оперативной памяти, в которых хранятся коды и данные процесса. Совокупность всех областей памяти, выделенных процессу, называется его *адресным пространством*. Когда говорят, что каждый процесс работает в своем адресном пространстве, имеется в виду защита адресных пространств, осуществляемая операционной системой. Защиты требуют и другие ресурсы, такие как файлы, внешние устройства и др. Одной из сторон функции защиты адресных пространств процессов является организация совместного их использования, например разрешение доступа к некоторой области памяти нескольким процессам для передачи данных друг другу.

За время существования процесса его выполнение может быть многократно прервано и продолжено. Для возобновления выполнения процесса необходимо восстановить его операционную среду. Состояние операционной среды характеризуется:

− состоянием регистров и программного счетчика процессора;

− режимом работы процессора,

− указателями на открытые файлы,

− информацией о незавершенных операциях ввода-вывода,

− кодами ошибок, выполняемых процессом системных вызовов и т. д.

Эта информация обычно называется **контекстом** процесса. При этом говорят, что при смене процесса происходит переключение контекста.

Операционная система занимается также **синхронизацией процессов**. Это позволяет процессу приостанавливать свое выполнение до наступления какого-то конкретного события в системе, например завершения операции ввода-вывода.

В операционной системе обычно нет однозначного соответствия между процессами и программами. Так, один и тот же программный файл может породить несколько параллельно выполняемых процессов. Процесс же в ходе своего выполнения может сменить программный файл и начать выполнять другую программу.

Часто при реализации сложных программных комплексов их работа организуется в виде нескольких параллельно выполняемых процессов, которые периодически взаимодействуют друг с другом и обмениваются данными. Из соображений защиты операционная система не позволяет процессам читать или писать данные в памяти других процессов. Для оперативного взаимодействия процессов операционная система должна предоставлять особые средства — средства межпроцессного взаимодействия.

Подсистема управления процессами:

− планирует выполнение процессов, то есть распределяет время между несколькими одновременно существующими в системе процессами;
− создает и уничтожает процессы;
− обеспечивает процессы необходимыми системными ресурсами;
− обеспечивает синхронизацию процессов;
− обеспечивает межпроцессное взаимодействие.

ЛАНЬ®

Контрольные вопросы

1. Что входит в системные основы управления ресурсами?
2. Что такое интерфейс прикладного программирования?
3. Перечислите основные ресурсы, управляемые средствами ОС.
4. Что такое процесс?
5. Что необходимо для управления процессами?

ЛАНЬ®

ЗАКЛЮЧЕНИЕ

В современном мире ведущее место в жизни общества занимают персональные компьютеры, сфера применения которых безгранична. Персональный компьютер представляет собой наиболее развитый вид микропроцессорных систем. На основе персональных компьютеров можно строить самые сложные контрольно-измерительные, управляющие, вычислительные и информационные системы. Имеющиеся в персональном компьютере аппаратные и программные средства делают его универсальным инструментом для самых разных задач.

В случае вычислительных и информационных систем персональный компьютер не нуждается в подключении нестандартной аппаратуры, все сводится к подбору или написанию необходимого программного обеспечения. В случае же контрольно-измерительных и управляющих систем персональный компьютер оснащается набором инструментов для сопряжения с внешними устройствами и соответствующими программными средствами.

В одно и то же время один и тот же компьютер может решать самые разнообразные задачи. Например, в системе управления технологическими процессами или научными установками он может математически моделировать происходящие процессы, выдавать в реальном времени управляющие сигналы, принимать в реальном времени ответные сигналы, накапливать информацию, обрабатывать ее, обмениваться информацией с другими компьютерами и т. д.

Появление новых поколений ЭВМ обусловлено расширением сферы их применения, требующей более производительной, дешевой и надежной вычислительной техники.

Проблема создания эффективных систем параллельного программирования, ориентированных на высокоуровневое распараллеливание алгоритмов вычислений и обработки данных, представляется достаточно сложной и предполагает дифференцированный подход с учетом сложности распараллеливания и необходимости синхронизации процессов во времени.

Наряду с развитием архитектурных и системотехнических решений ведутся работы по совершенствованию технологий производства интегральных схем и по созданию принципиально новых элементных баз, основанных на оптоэлектронных и оптических принципах.

В плане создания принципиально новых архитектур вычислительных средств большое внимание уделяется проектам нейрокомпьютеров, базирующихся на понятии нейронной сети, моделирующей основные свойства реальных нейронов. В случае применения био- или оптоэлементов могут быть созданы соответственно биологические или оптические нейрокомпьютеры. Многие исследователи считают, что в следующем веке нейрокомпьютеры в значительной степени вытеснят современные ЭВМ, используемые для решения трудноформализуемых задач. Последние достижения в микроэлектронике и разработка элементной базы на основе биотехнологий дают возможность прогнозировать создание биокомпьютеров.

Важным направлением развития вычислительных средств пятого и последующих поколений является интеллектуализация ЭВМ, связанная с наделением ее элементами интеллекта, интеллектуализацией интерфейса с пользователем и др. Работа в данном направлении, затрагивая в первую очередь программное обеспечение, потребует и создания ЭВМ определенной архитектуры, используемых в системах управления базами знаний, — компьютеров баз знаний, а также других подклассов ЭВМ. При этом ЭВМ должна обладать способностью к обучению, производить ассоциативную обработку информации и вести интеллектуальный диалог при решении конкретных задач.

Для того чтобы грамотно и полноценно использовать персональный компьютер в составе любых систем, надо иметь представление о его архитектуре, об основных принципах построения, об устройствах, входящих в его состав, наконец, о внешних интерфейсах.

Данное учебное пособие создано для подготовки специалистов в сфере компьютерных систем в практической деятельности.

ЛАНЬ®

СПИСОК ЛИТЕРАТУРЫ

Основные источники:

1. *Богданов, А. В*. Архитектуры и топологии многопроцессорных вычислительных систем [Электронный ресурс]: учебник / А. В. Богданов, В. В. Корхов, Е. Н. Станкова. — М. : НОУ «ИНТУИТ», 2016. — 135 с.
2. *Гребешков, А. Ю*. Вычислительная техника, сети и телекоммуникации [Электронный ресурс]: учебное пособие для вузов. — М. : Горячая линия — Телеком, 2015. — 190 с.
3. *Китаев, Ю. В*. Основы микропроцессорной техники [Электронный ресурс]: учебное пособие. — СПб. : Университет ИТМО, 2016. — 51 с.
4. *Сидоров, В. Д*. Аппаратное обеспечение ЭВМ [Электронный ресурс] : учебник / В. Д. Сидоров, Н. В. Струмпэ. — М. : Академия, 2014. — 336 с.

Дополнительные источники:

1. *Богомазова, Г. Н*. Установка и обслуживание программного обеспечения персональных компьютеров, серверов, периферийных устройств и оборудования [Электронный ресурс] : учебник. — М. : Академия, 2015. — 256 с.
2. *Максимов, Н. В*. Архитектура ЭВМ и вычислительных систем: учебник / Н. В. Максимов, Т. Л. Партыка, И. И. Попов. — 5-е изд., перераб. и доп. — М.: ФОРУМ : ИНФРА-М, 2013. — 512 с. : ил. — (Профессиональное образование).
3. *Таненбаум, Э*. Архитектура компьютера / Э. Таненбаум, Т. Остин. — 6-е изд. — СПб. : Питер, 2013. — 816 с.

Интернет-ресурсы:

1. Интернет-университет информационных технологий. Основы микропроцессорной техники [Электронный ресурс]. — Режим доступа : intuit.ru\mpbasics\hardware\mpbasics, свободный. — Загл. с экрана.
2. Интернет-университет информационных технологий. Архитектура и организация ЭВМ [Электронный ресурс]. — Режим доступа: www.intuit.ru/ Архитектура ЭВМ, свободный. — Загл. с экрана.
3. Организация ЭВМ и систем [Электронный ресурс]. — Режим доступа: http://paralichka85.px6.ru/, свободный. — Загл. с экрана.

Светлана Викторовна БЕЛУГИНА

АРХИТЕКТУРА КОМПЬЮТЕРНЫХ СИСТЕМ

КУРС ЛЕКЦИЙ

Учебное пособие

Зав. редакцией
литературы по информационным технологиям
и системам связи *О. Е. Гайнутдинова*
Ответственный редактор *С. В. Макаров*
Корректор *Е. А. Романова*
Выпускающий *С. В. Орловский*

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.10.953.П.1028
от 14.04.2016 г., выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lanbook.ru; www.lanbook.com
196105, Санкт-Петербург, пр. Юрия Гагарина, д.1, лит. А.
Тел.: (812) 336-25-09, 412-92-72.
Бесплатный звонок по России: 8-800-700-40-71

Подписано в печать 25.11.20.
Бумага офсетная. Гарнитура Школьная. Формат 70×100 $^{1}/_{16}$.
Печать офсетная. Усл. п. л. 13,00. Тираж 100 экз.

Заказ № 049-20.

Отпечатано в полном соответствии
с качеством предоставленного оригинал-макета
в АО «Т8 Издательские технологии»
109316, г. Москва, Волгоградский пр., д. 42, к. 5.