Частное профессиональное образовательное учреждение
«СЕВЕРО-КАВКАЗСКИЙ КОЛЛЕДЖ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ»
(ЧПОУ «СККИТ»)

АРХИТЕКТУРА АППАРАТНЫХ СРЕДСТВ

КУРС ЛЕКЦИЙ

09.02.06 Сетевое и системное администрирование

Пятигорск

СОДЕРЖАНИЕ

Введение
Тема 1.1 Классы вычислительных машин

За единицу количества информации принимается такое количество информации, которое содержит сообщение, уменьшающее неопределенность в два раза. Такая единица названа "бит".

Для информации существуют свои единицы измерения информации. Если рассматривать сообщения информации как последовательность знаков, то их можно представлять битами, а измерять в байтах, килобайтах, мегабайтах, гигабайтах, терабайтах и петабайтах.

Давайте разберемся с этим, ведь нам придется измерять объем памяти и быстродействие компьютера.

Бит

Единицей измерения количества информации является бит – это наименьшая (элементарная) единица.

1бит – это количество информации, содержащейся в сообщении, которое вдвое уменьшает неопределенность знаний о чем-либо.

Байт

Байт – основная единица измерения количества информации.

Байтом называется последовательность из 8 битов.

Байт – довольно мелкая единица измерения информации. Например, 1 символ – это 1 байт.

Производные единицы измерения количества информации

1 байт=8 битов

1 килобайт (Кб)=1024 байта =210 байтов

1 мегабайт (Мб)=1024 килобайта =210 килобайтов=220 байтов

1 гигабайт (Гб)=1024 мегабайта =210 мегабайтов=230 байтов

1 терабайт (Гб)=1024 гигабайта =210 гигабайтов=240 байтов

Запомните, приставка КИЛО в информатике – это не 1000, а 1024, то есть 210 .

Методы измерения количества информации

Итак, количество информации в 1 бит вдвое уменьшает неопределенность знаний. Связь же между количеством возможных событий N и количеством информации I определяется формулой Хартли:

$N=2i$.

Алфавитный подход к измерению количества информации

При этом подходе отвлекаются от содержания (смысла) информации и рассматривают ее как последовательность знаков определенной знаковой системы. Набор символов языка, т.е. его алфавит можно рассматривать как различные возможные события. Тогда, если считать, что появление символов в сообщении равновероятно, по формуле Хартли можно рассчитать, какое количество информации несет в себе каждый символ:

$I=\log 2N$.

Вероятностный подход к измерению количества информации

Этот подход применяют, когда возможные события имеют различные вероятности реализации. В этом случае количество информации определяют по формуле Шеннона:

$$I = -\sum_{i=1}^{N} p_i \log_2 p_i$$, где

I – количество информации,

N – количество возможных событий,

Pi – вероятность i-го события.

6. Кодирование информации различных видов

1.6.1. КОДИРОВАНИЕ ЧИСЕЛ.

Используя n бит, можно записывать двоичные коды чисел от 0 до 2n-1, всего 2n чисел.

1) Кодирование положительных чисел: Для записи положительных чисел в байте заданное число слева дополняют нулями до восьми цифр. Эти нули называют незначимыми.

Например: записать в байте число 1310 = 11012

Результат: 00001101

2) Кодирование отрицательных чисел:Наибольшее положительное число, которое можно записать в байт, - это 127, поэтому для записи отрицательных чисел используют числа с 128-го по 255-е. В этом случае, чтобы записать отрицательное число, к нему добавляют 256, и полученное число записывают в ячейку.

1.6.2. КОДИРОВАНИЕ ТЕКСТА.

Соответствие между набором букв и числами называется кодировкой символа. Как правило, код символа хранится в одном байте, поэтому коды символов могут принимать значение от 0 до 255. Такие кодировки называют однобайтными. Они позволяют использовать 256 символов. Таблица кодов символов называется ASCII (American StandardCodeforInformationInterchange- Американский стандартный код для обмена информацией). Таблица ASCII-кодов состоит из двух частей:

Коды от 0 до 127 одинаковы для всех IBM-PC совместимых компьютеров и содержат:

коды управляющих символов;

коды цифр, арифметических операций, знаков препинания;

некоторые специальные символы;

коды больших и маленьких латинских букв.

Вторая часть таблицы (коды от 128 до 255) бывает различной в различных компьютерах. Она содержит:

коды букв национального алфавита;

коды некоторых математическихсимволов;

коды символов псевдографики.

В настоящее время все большее распространение приобретает двухбайтная кодировка Unicode. В ней коды символов могут принимать значение от 0 до 65535.

1.6.3. КОДИРОВАНИЕ ЦВЕТОВОЙ ИНФОРМАЦИИ.

Одним байтом можно закодировать 256 различных цветов. Это достаточно для рисованных изображений типа мультфильмов, но не достаточно для полноцветных изображений живой природы. Если для кодирования цвета использовать 2 байта, можно закодировать уже 65536 цветов. А если 3 байта – 16,5 млн. различных цветов. Такой режим позволяет хранить, обрабатывать и передавать изображения, не уступающие по качеству наблюдаемым в живой природе.

Из курса физики известно, что любой цвет можно представить в виде комбинации трех основных цветов: красного, зеленого, синего (их называют цветовыми составляющими). Если кодировать цвет точки с помощью 3 байтов, то первый байт выделяется красной составляющей, второй – зеленой, третий – синей. Чем больше значение байта цветовой составляющей, тем ярче этот цвет.

Белый цвет – у точки есть все цветовые составляющие, и они имеют полную яркость. Поэтому белый цвет кодируется так: 255 255 255. (11111111 11111111 11111111)

Черный цвет – отсутствие всех прочих цветов: 0 0 0. (00000000 00000000 00000000)

Серый цвет – промежуточный между черным и белым. В нем есть все цветовые составляющие, но они одинаковы и нейтрализуют друг друга.

Например: 100 100 100 или 150 150 150. (2-й вариант - ярче).

Красный цвет – все составляющие, кроме красной, равны 0. Темно-красный: 128 0 0. Ярко-красный: 255 0 0.

Зеленый цвет – 0 255 0.

Синий цвет – 0 0 255.

Вопросы для самопроверки

1. Что называется информацией.
2. Как можно записать информационные сообшения?
3. Как можно записать не текстовые вкличины ?

Типы и структуры данных. Передача данных Двоичное кодирование звуковой и мультимедиа информации. Сжатие информации. Кодирование видеоинформации..

План.

1. Контрольный опрос
2. Типы и структуры данных
3. Двоичное кодирование звуковой и мультимедиа информации.
4. Сжатие информации.

Что такое структуры данных?

Простыми словами, структура данных – это контейнер, который хранит информацию в определенном виде. Из-за такой «компоновки» она может быть эффективной в одних операциях и неэффективной в других. Цель разработчика – выбрать из существующих структур оптимальный для конкретной задачи вариант.

Зачем нужны структуры данных?

Данные являются самой важной сущностью в информатике, а структуры позволяют хранить их в организованной форме.

Какую бы проблему вы не решали, вам приходится иметь дело с данными — будь то зарплата сотрудника, цены на акции, список покупок или даже простой телефонный справочник.

В зависимости от ситуации данные должны храниться в некотором определенном формате. Структуры данных предлагают несколько вариантов таких размещений.

8 часто используемых структур

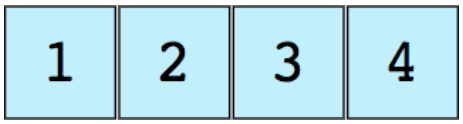
Давайте сначала перечислим наиболее часто используемые структуры данных, а затем рассмотрим их одну за другой:

1. Массив (Array)
2. Стек (Stack)
3. Очередь (Queue)
4. Связный список (Linked List)
5. Дерево (Tree)
6. Граф (Graph)
7. Префиксное дерево (Trie)
8. Хэш-Таблица (Hash Table)

Массивы

Массив – это самая простая и наиболее широко используемая из структур. Стеки и очереди являются производными от массивов.

Вот изображение простого массива размером 4, содержащего элементы (1, 2, 3 и 4).



Каждому из них присваивается неотрицательное числовое значение – индекс, который соответствует позиции этого элемента в массиве. Большинство языков определяют начальный индекс массива как 0.

Существует два типа массивов:

- Одномерные массивы (как на картинке).
- Многомерные массивы (массивы массивов).

Основные операции с массивами

- Insert – вставка.
- Get – получение элемента.
- Delete – удаление.
- Size – получение общего количества элементов в массиве.

Частые вопросы о массивах

- Найти второй минимальный элемент.
- Первые не повторяющиеся целые числа.
- Объединить два отсортированных массива.
- Переставить положительные и отрицательные значения.

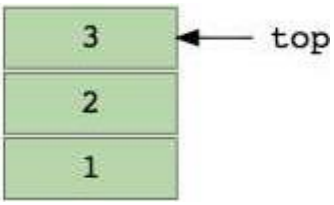
Стеки

Мы все знакомы с опцией Отменить (Undo), которая присутствует практически в каждом приложении. Вы когда-нибудь задумывались, как это работает?

Для этого вы сохраняете предыдущие состояния приложения (определенное их количество) в памяти в таком порядке, что последнее сохраненное появляется первым. Это не может быть сделано только с помощью массивов. Здесь на помощь приходит стек.

Пример стека из реальной жизни – куча книг, лежащих друг на друге. Чтобы получить книгу, которая находится где-то в середине, вам нужно удалить все, что лежит сверху. Так работает метод LIFO (Last In First Out, последним пришел – первым ушел).

Вот изображение стека, содержащего три элемента (1, 2 и 3). Элемент 3 находится сверху и будет удален первым:



Основные операции со стеками

- Push – вставка элемента наверх стека.
- Pop – получение верхнего элемента и его удаление.
- isEmpty – возвращает true, если стек пуст.
- Top – получение верхнего элемента без удаления.

Часто задаваемые вопросы о стеках

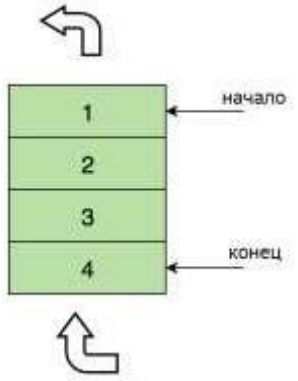
- Вычисление постфиксного выражения с помощью стека.
- Сортировка значений в стеке.
- Проверка сбалансированности скобок в выражении.

Очереди

Как и стек, очередь – это линейная структура данных, которая хранит элементы последовательно. Единственное существенное различие заключается в том, что вместо использования метода LIFO, очередь реализует метод FIFO (First in First Out, первым пришел – первым ушел).

Идеальный пример этих структур в реальной жизни – очереди людей в билетную кассу. Если придет новый человек, он присоединится к линии с конца, а не с начала. А человек, стоящий впереди, первым получит билет и, следовательно, покинет очередь.

Вот изображение очереди, содержащей четыре элемента (1, 2, 3 и 4). Здесь 1 находится вверху и будет удален первым:



Основные операции с очередями

- Enqueue – вставка в конец.
- Dequeue – удаление из начала.
- isEmpty – возвращает true, если очередь пуста.
- Top – получение первого элемента.

Часто задаваемые вопросы об очередях

- Реализация стека с помощью очереди.
- Развернуть первые k элементов.
- Генерация двоичных чисел от 1 до n с помощью очереди.

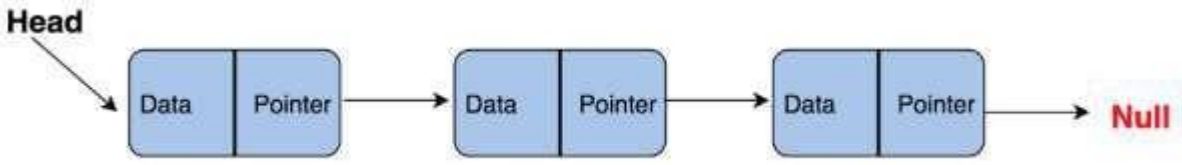
Связный список

Еще одна важная линейная структура данных, которая на первый взгляд похожа на массив, но отличается распределением памяти, внутренней организацией и способом выполнения основных операций вставки и удаления.

Связный список – это сеть узлов, каждый из которых содержит данные и указатель на следующий узел в цепочке. Также есть указатель на первый элемент – head. Если список пуст, то он указывает на null.

Связные списки используются для реализации файловых систем, хэш-таблиц и списков смежности.

Вот визуальное представление внутренней структуры связного списка:



Типы связных списков:

- Однонаправленный
- Двунаправленный

Основные операции со связными списками

- InsertAtEnd – вставка в конец.

- InsertAtHead – вставка в начало.
- Delete – удаление указанного элемента.
- DeleteAtHead – удаление первого элемента.
- Search – получение указанного элемента.
- isEmpty – возвращает true, если связный список пуст.

Часто задаваемые вопросы о связных списках

- Разворот связного списка.
- Обнаружение цикла.
- Получение N-го узла с конца.
- Удаление дубликатов.

Графы

Граф представляет собой набор узлов, соединенных друг с другом в виде сети. Узлы также называются вершинами. Пара (**x**, **y**) называется ребром, которое указывает, что вершина **x**соединена с вершиной **y**. Ребро может содержать вес/стоимость, показывая, сколько затрат требуется, чтобы пройти от **x** до **y**.

Типы графов:

- Неориентированный
- Ориентированный

В языке программирования графы могут быть представлены в двух формах:

- Матрица смежности
- Список смежности

Общие алгоритмы обхода графов:

- В ширину
- В глубину

Часто задаваемые вопросы о графах

- Реализовать поиск по ширине и глубине.
- Проверка, является ли граф деревом.
- Количество ребер в графе.
- Кратчайший путь между двумя вершинами.

Деревья

Дерево – это иерархическая структура данных, состоящая из вершин (узлов) и ребер, соединяющих их. Они похожи на графы, но есть одно важное отличие: в дереве не может быть цикла.

Деревья широко используются в искусственном интеллекте и сложных алгоритмах для обеспечения эффективного механизма хранения данных.

Вот изображение простого дерева, и основные термины:

Типы деревьев:

- N-арное дерево;
- сбалансированное дерево;
- бинарное дерево;
- бинарное дерево поиска;
- дерево AVL;
- красно-чёрное дерево;
- 2-3 дерево.

Из всех типов чаще всего используются бинарное дерево и бинарное дерево поиска.

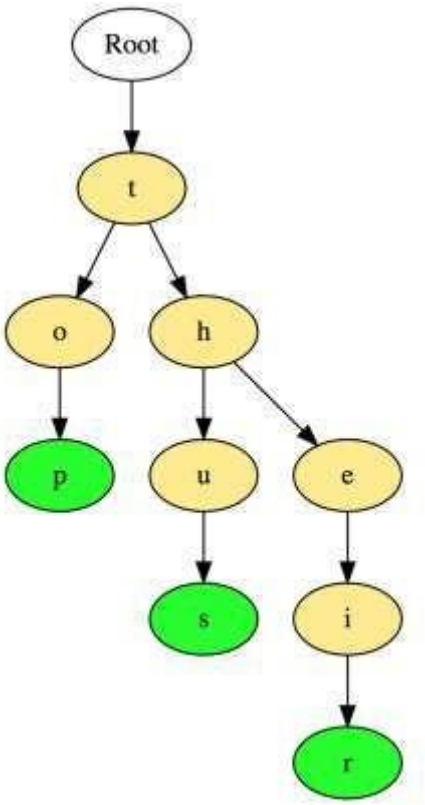
Часто задаваемые вопросы о деревьях

- Высота бинарного дерева.
- Найти k-ое максимальное значение в дереве бинарного поиска.
- Узлы на расстоянии k от корня.
- Предки заданного узла в бинарном дереве.

Префиксное дерево

Префиксные деревья (tries) – древовидные структуры данных, эффективные для решения задач со строками. Они обеспечивают быстрый поиск и используются преимущественно для поиска слов в словаре, автодополнения в поисковых системах и даже для IP-маршрутизации.

Вот иллюстрация того, как три слова top, thus и their хранятся в префиксном дереве:



Слова размещаются сверху вниз. Выделенные зеленым элементы показывают конец каждого слова.

Часто задаваемые вопросы о префиксных деревьях

- Общее количество слов.
- Вывод всех слов.
- Сортировка элементов массива.
- Формирование слов из словаря.
- Создание словаря T9.

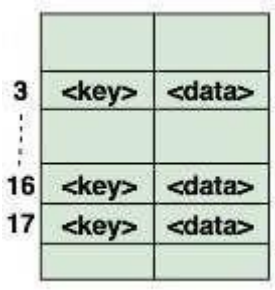
Хеш-Таблица

Хеширование – это процесс, используемый для уникальной идентификации объектов и хранения каждого из них в некотором предварительно вычисленном уникальном индексе – ключе. Итак, объект хранится в виде пары **ключ-значение**, а коллекция таких элементов называется **словарем**. Каждый объект можно найти с помощью его ключа. Существует несколько структур, основанных на хешировании, но наиболее часто используется хеш-таблица, которая обычно реализуется с помощью массивов.

Производительность структуры зависит от трех факторов:

- функция хеширования;
- размер хеш-таблицы;
- метод обработки коллизий.

Вот иллюстрация того, как хэш отображается в массиве. Индекс вычисляется с помощью хеш-функции.



Часто задаваемые вопросы о хеш-таблицах

- Найти симметричные пары.
- Определить, является ли массив подмножеством другого массива.
- Проверить, являются ли массивы непересекающимися.

Теперь вы знаете 8 основных структур данных любого языка программирования. Можете смело отправляться на IT-собеседование.

Контрольные вопросы:

1. Каие типы данных бывают?;
2. Что такое Массив (Array)?
3. Что такое Стек (Stack)?
4. Что такое Очередь (Queue)?
5. Что такое Связный список (Linked List)?
6. Что такое Дерево (Tree)?
7. Что такое Граф (Graph)?

Системы счисления. Непозиционные и позиционные системы счисления. Свойства позиционных систем счисления.

План.

1. Контрольный опрос
2. Системы счисления
3. Непозиционные и позиционные системы счисления
4. Свойства систем счисления

Система счисления — это способ записи (представления) чисел.

Что под этим подразумевается? Например, вы видите перед собой несколько деревьев. Ваша задача — их посчитать. Для этого можно — загибать пальцы, делать зарубки на камне (одно дерево — один палец\зарубка) или сопоставить 10 деревьям какой-нибудь предмет, например, камень, а единичному экземпляру — палочку и выкладывать их на землю по мере подсчета. В первом случае число представляется, как строка из загнутых пальцев или зарубок, во втором — композиция камней и палочек, где слева — камни, а справа — палочки

Системы счисления подразделяются на позиционные и непозиционные, а позиционные, в свою очередь, — на однородные и смешанные.

Непозиционная — самая древняя, в ней каждая цифра числа имеет величину, не зависящую от её позиции (разряда). То есть, если у вас 5 черточек — то число тоже равно

5, поскольку каждой черточке, независимо от её места в строке, соответствует всего 1 один предмет.

Позиционная система — значение каждой цифры зависит от её позиции (разряда) в числе. Например, привычная для нас 10-я система счисления — позиционная. Рассмотрим число 453. Цифра 4 обозначает количество сотен и соответствует числу 400, 5 — кол-во десяток и аналогично значению 50, а 3 — единиц и значению 3. Как видим — чем больше разряд — тем значение выше. Итоговое число можно представить, как сумму 400+50+3=453.

Однородная система — для всех разрядов (позиций) числа набор допустимых символов (цифр) одинаков. В качестве примера возьмем упоминавшуюся ранее 10-ю систему. При записи числа в однородной 10-й системе вы можете использовать в каждом разряде исключительно одну цифру от 0 до 9, таким образом, допускается число 450 (1-й разряд — 0, 2-й — 5, 3-й — 4), а 4F5 — нет, поскольку символ F не входит в набор цифр от 0 до 9.

Смешанная система — в каждом разряде (позиции) числа набор допустимых символов (цифр) может отличаться от наборов других разрядов. Яркий пример — система измерения времени. В разряде секунд и минут возможно 60 различных символов (от «00» до «59»), в разряде часов – 24 разных символа (от «00» до «23»), в разряде суток – 365 и т. д.

Непозиционные системы

Как только люди научились считать — возникла потребность записи чисел. В начале все было просто — зарубка или черточка на какой-нибудь поверхности соответствовала одному предмету, например, одному фрукту. Так появилась первая система счисления — единичная.

Единичная система счисления

Число в этой системе счисления представляет собой строку из черточек (палочек), количество которых равно значению данного числа. Таким образом, урожай из 100 фиников будет равен числу, состоящему из 100 черточек.
Но эта система обладает явными неудобствами — чем больше число — тем длиннее строка из палочек. Помимо этого, можно легко ошибиться при записи числа, добавив случайно лишнюю палочку или, наоборот, не дописав.
Для удобства, люди стали группировать палочки по 3, 5, 10 штук. При этом, каждой группе соответствовал определенный знак или предмет. Изначально для подсчета использовались пальцы рук, поэтому первые знаки появились для групп из 5 и 10 штук (единиц). Все это позволило создать более удобные системы записи чисел.

Римская система

Римская система не сильно отличается от египетской. В ней для обозначения чисел 1, 5, 10, 50, 100, 500 и 1000 используются заглавные латинские буквы I, V, X, L, C, D и M соответственно. Число в римской системе счисления — это набор стоящих подряд цифр.

Методы определения значения числа:

1. Значение числа равно сумме значений его цифр. Например, число 32 в римской системе счисления имеет вид XXXII=(X+X+X)+(I+I)=30+2=32
2. Если слева от большей цифры стоит меньшая, то значение равно разности между большей и меньшей цифрами. При этом, левая цифра может быть меньше правой максимум на один порядок: так, перед L(50) и C(100) из «младших» может стоять только

X(10), перед D(500) и M(1000) — только C(100), перед V(5) — только I(1); число 444 в рассматриваемой системе счисления будет записано в виде CDXLIV = (D-C)+(L-X)+(V-I) = 400+40+4=444.

3. Значение равно сумме значений групп и цифр, не подходящих под 1 и 2 пункты.

Помимо цифирных, существуют и буквенные (алфавитные) системы счисления, вот некоторые из них:
1) Славянская
2) Греческая (ионийская)

Позиционные системы счисления

Как упоминалось выше — первые предпосылки к появлению позиционной системы возникли в древнем Вавилоне. В Индии система приняла форму позиционной десятичной нумерации с применением нуля, а у индусов эту систему чисел заимствовали арабы, от которых её переняли европейцы. По каким-то причинам, в Европе за этой системой закрепилось название “арабская”.

Десятичная система счисления

Это одна из самых распространенных систем счисления. Именно её мы используем, когда называем цену товара и произносим номер автобуса. В каждом разряде (позиции) может использоваться только одна цифра из диапазона от 0 до 9. Основанием системы является число 10.

Для примера возьмем число 503. Если бы это число было записано в непозиционной системе, то его значение равнялось 5+0+3 = 8. Но у нас — позиционная система и значит каждую цифру числа необходимо умножить на основание системы, в данном случае число “10”, возведенное в степень, равную номеру разряда. Получается, значение равно $5*10^2 + 0*10^1 + 3*10^0 = 500+0+3 = 503$. Чтобы избежать путаницы при одновременной работе с несколькими системами счисления основание указывается в качестве нижнего индекса. Таким образом, $503 = 503_{10}$. Помимо десятичной системы, отдельного внимания заслуживают 2-, 8-, 16-ая системы.

Двоичная система счисления

Эта система, в основном, используется в вычислительной технике. Почему не стали использовать привычную нам 10-ю? Первую вычислительную машину создал Блез Паскаль, использовавший в ней десятичную систему, которая оказалась неудобной в современных электронных машинах, поскольку требовалось производство устройств, способных работать в 10 состояниях, что увеличивало их цену и итоговые размеры машины. Этих недостатков лишены элементы, работающие в 2-ой системе. Тем не менее, рассматриваемая система была создана за долго до изобретения вычислительных машин и уходит “корнями” в цивилизацию Инков, где использовались кипу — сложные верёвочные сплетения и узелки.

Двоичная позиционная система счисления имеет основание 2 и использует для записи числа 2 символа (цифры): 0 и 1. В каждом разряде допустима только одна цифра — либо 0, либо 1.

Примером может служить число 101. Оно аналогично числу 5 в десятичной системе счисления. Для того, чтобы перевести из 2-й в 10-ю необходимо умножить каждую цифру двоичного числа на основание “2”, возведенное в степень, равную разряду. Таким образом, число $101_2 = 1*2^2 + 0*2^1 + 1*2^0 = 4+0+1 = 5_{10}$.

Хорошо, для машин 2-я система счисления удобнее, но мы ведь часто видим, используем на компьютере числа в 10-й системе. Как же тогда машина определяет какую цифру вводит пользователь? Как переводит число из одной системы в другую, ведь в её распоряжении всего 2 символа — 0 и 1?
Чтобы компьютер мог работать с двоичными числами (кодами), необходимо чтобы они где-то хранились. Для хранения каждой отдельной цифры применяется триггер, представляющий собой электронную схему. Он может находится в 2-х состояниях, одно из которых соответствует нулю, другое — единице. Для запоминания отдельного числа используется регистр — группа триггеров, число которых соответствует количеству разрядов в двоичном числе. А совокупность регистров — это оперативная память. Число, содержащееся в регистре — машинное слово. Арифметические и логические операции со словами осуществляет арифметико-логическое устройство (АЛУ). Для упрощения доступа к регистрам их нумеруют. Номер называется адресом регистра. Например, если необходимо сложить 2 числа — достаточно указать номера ячеек (регистров), в которых они находятся, а не сами числа. Адреса записываются в 8- и 16-ричной системах (о них будет рассказано ниже), поскольку переход от них к двоичной системе и обратно осуществляется достаточно просто. Для перевода из 2-й в 8-ю число необходимо разбить на группы по 3 разряда справа налево, а для перехода к 16-ой — по 4. Если в крайней левой группе цифр не достает разрядов, то они заполняются слева нулями, которые называются ведущими. В качестве примера возьмем число 101100_2. В восьмеричной — это 101 100 = 54_8, а в шестнадцатеричной — 0010 1100 = $2C_{16}$. Отлично, но почему на экране мы видим десятичные числа и буквы? При нажатии на клавишу в компьютер передаётся определённая последовательность электрических импульсов, причём каждому символу соответствует своя последовательность электрических импульсов (нулей и единиц). Программа драйвер клавиатуры и экрана обращается к кодовой таблице символов (например, Unicode, позволяющая закодировать 65536 символов), определяет какому символу соответствует полученный код и отображает его на экране. Таким образом, тексты и числа хранятся в памяти компьютера в двоичном коде, а программным способом преобразуются в изображения на экране.

Восьмеричная система счисления 8-я система счисления, как и двоичная, часто применяется в цифровой технике. Имеет основание 8 и использует для записи числа цифры от 0 до 7.

Пример восьмеричного числа: 254. Для перевода в 10-ю систему необходимо каждый разряд исходного числа умножить на 8^n, где n — это номер разряда. Получается, что $254_8 = 2*8^2 + 5*8^1 + 4*8^0 = 128+40+4 = 172_{10}$.

Шестнадцатеричная система счисления

Шестнадцатеричная система широко используется в современных компьютерах, например при помощи неё указывается цвет: #FFFFFF — белый цвет. Рассматриваемая система имеет основание 16 и использует для записи числа: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B. C, D, E, F, где буквы равны 10, 11, 12, 13, 14, 15 соответственно.

В качестве примера возьмем число $4F5_{16}$. Для перевода в восьмеричную систему — сначала преобразуем шестнадцатеричное число в двоичное, а затем, разбив на группы по 3 разряда, в восьмеричное. Чтобы преобразовать число в 2-е необходимо каждую цифру представить в виде 4-х разрядного двоичного числа. $4F5_{16} = (100\ 1111\ 101)_2$. Но в 1 и 3 группах не достает разряда, поэтому заполним каждый ведущими нулями: 0100 1111 0101.

Теперь необходимо разделить полученное число на группы по 3 цифры справа налево: 0100 1111 0101 = 010 011 110 101. Переведем каждую двоичную группу в восьмеричную систему, умножив каждый разряд на 2^n, где n — номер разряда: $(0*2^2+1*2^1+0*2^0)$ $(0*2^2+1*2^1+1*2^0)$ $(1*2^2+1*2^1+0*2^0)$ $(1*2^2+0*2^1+1*2^0) = 2365_8$.

Помимо рассмотренных позиционных систем счисления, существуют и другие, например:
1) Троичная
2) Четверичная
3) Двенадцатеричная

Позиционные системы подразделяются на однородные и смешанные.

Однородные позиционные системы счисления

Определение, данное в начале статьи, достаточно полно описывает однородные системы, поэтому уточнение — излишне.

Смешанные системы счисления

К уже приведенному определению можно добавить теорему: “если $P=Q^n$ (P,Q,n – целые положительные числа, при этом P и Q — основания), то запись любого числа в смешанной (P-Q)-ой системе счисления тождественно совпадает с записью этого же числа в системе счисления с основанием Q.”

Опираясь на теорему, можно сформулировать правила перевода из P-й в Q-ю системы и наоборот:

1. Для перевода из Q-й в P-ю, необходимо число в Q-й системе, разбить на группы по n цифр, начиная с правой цифры, и каждую группу заменить одной цифрой в P-й системе.
2. Для перевода из P-й в Q-ю, необходимо каждую цифру числа в P-й системе перевести в Q-ю и заполнить недостающие разряды ведущими нулями, за исключением левого, так, чтобы каждое число в системе с основанием Q состояло из n цифр.

Яркий пример — перевод из двоичной системы счисления в восьмеричную. Возьмем двоичное число 10011110_2, для перевода в восьмеричное — разобьем его справа налево на группы по 3 цифры: 010 011 110, теперь умножим каждый разряд на 2^n, где n — номер разряда, 010 011 110 = $(0*2^2+1*2^1+0*2^0)$ $(0*2^2+1*2^1+1*2^0)$ $(1*2^2+1*2^1+0*2^0) = 236_8$. Получается, что $10011110_2 = 236_8$. Для однозначности изображения двоично-восьмеричного числа его разбивают на тройки: $236_8 = (10\ 011\ 110)_{2\text{-}8}$.

Смешанными системами счисления также являются, например:
1) Факториальная
2) Фибоначчиева

Контрольные вопросы:

1. Зачем нужны системы счисления?
2. Какие системы счисления вы знаете?
3. Как переводить числа из одной системы счисления в другую?

Тема 2.2. Принципы организации ЭВМ

Высказывание - это повествовательное предложение, про которое можно определенно сказать истинно оно или ложно (истина (логическая 1), ложь (логический 0)).

Логические операции - мыслительные действия, результатом которых является изменение содержания или объема понятий, а также образование новых понятий.

Логическое выражение - устное утверждение или запись, в которое, наряду с постоянными величинами, обязательно входят переменные величины (объекты). В зависимости от значений этих переменных величин (объектов) логическое выражение может принимать одно из двух возможных значений: истина (логическая 1) или ложь (логический 0).

Сложное логическое выражение - логическое выражение, состоящее из одного или нескольких простых логических выражений (или сложных логических выражений), соединенных с помощью логических операций.

Логические операции и таблицы истинности

1) Логическое умножение или конъюнкция:

Конъюнкция - это сложное логическое выражение, которое считается истинным в том и только том случае, когда оба простых выражения являются истинными, во всех остальных случаях данное сложенное выражение ложно.

Обозначение: F = A & B.

Таблица истинности для конъюнкции

2) Логическое сложение или дизъюнкция:

Дизъюнкция - это сложное логическое выражение, которое истинно, если хотя бы одно из простых логических выражений истинно и ложно тогда и только тогда, когда оба простых логических выраженbя ложны.

Обозначение: F = A v B.

Таблица истинности для дизъюнкции

3) Логическое отрицание или инверсия:

Инверсия - это сложное логическое выражение, если исходное логическое выражение истинно, то результат отрицания будет ложным, и наоборот, если исходное логическое выражение ложно, то результат отрицания будет истинным. Другими простыми слова, данная операция означает, что к исходному логическому выражению добавляется частица НЕ или слова НЕВЕРНО, ЧТО.

Обозначение: F = ¬A.

Таблица истинности для инверсии

	A

4) Логическое следование или импликация:

Импликация - это сложное логическое выражение, которое истинно во всех случаях, кроме как из истины следует ложь. То есть данная логическая операция связывает два простых логических выражения, из которых первое является условием (A), а второе (B) является следствием.

«A → B» истинно, если из A может следовать B.

Обозначение: F = A → B.

Таблица истинности для импликации

5) Логическая равнозначность или эквивалентность:

Эквивалентность - это сложное логическое выражение, которое является истинным тогда и только тогда, когда оба простых логических выражения имеют одинаковую истинность.

«A ↔ B» истинно тогда и только тогда, когда A и B равны.

Обозначение: F = A ↔ B.

Таблица истинности для эквивалентности

6) Операция XOR (исключающие или)

«A ⊕ B» истинно тогда, когда истинно A или B, но не оба одновременно. Эту операцию также называют "сложение по модулю два".

Обозначение: F = A ⊕ B.

Порядок выполнения логических операций в сложном логическом выражении

1. Инверсия;
2. Конъюнкция;
3. Дизъюнкция;
4. Импликация;
5. Эквивалентность.

Для изменения указанного порядка выполнения логических операций используются скобки. Таблицы истинности можно составить и для произвольной логической функции F(a, b, c…). В общем случае таблицы истинности имеют размер 2^N строк комбинаций для N независимых логических переменных. Поскольку таблица истинности выражения состоит из строк со всеми возможными комбинациями значений переменных, она полностью определяет значение выражения.

Законы алгебры логики

Те, кому лень учить эти законы, должны вспомнить алгебру, где знание нескольких способов преобразования позволяет решать очень сложные уравнения.

Строго говоря, это не законы, а теоремы. Но их доказательство не входит в программу изучения. Впрочем, доказательство обычно основывается на построении полной таблицы истинности.

Замечание. Знаки алгебры логики намеренно заменены на сложение и умножение.

	Для ИЛИ, ∨	Для И, &	Примечание
	$A \vee 0 = A$	$A \& 1 = A$	Ничего не меняется при действии, константы удаляются
	$A \vee 1 = 1$	$A \& 0 = 0$	Удаляются переменные, так как их оценивание не имеет смысла
	$A \vee B = B \vee A$	$AB = BA$	Переместительный (коммутативности)
	$A \vee \neg A = 1$		Один из операторов всегда 1 (закон исключения третьего)

		$A \& \neg A = 0$	Один из операторов всегда 0 (закон непротиворечия)
	$A \vee A = A$	$A \& A = A$	Идемпотентности (NB! Вместо A можно подставить составное выражение!)
	$\neg\neg A = A$		Двойное отрицание
	$(A \vee B) \vee C = A \vee (B \vee C)$	$(A \wedge B) \wedge C = A \wedge (B \wedge C)$	Ассоциативный
	$(A \vee B)\&C=(A\&C)\vee(B\&C)$	$(A\&B) \vee C = (A \vee C)\&(B \vee C)$	Дистрибутивный
0	$(A \vee B)\&(\neg A \vee B) = B$	$(A\&B) \vee (\neg A\&B) = B$	Склеивания
1	$\neg(A \vee B) = \neg A \&\neg B$	$\neg(A\&B) = \neg A \vee \neg B$	Правило де Моргана
2	$A \vee (A\&C) = A$	$A\&(A \vee C) = A$	Поглощение
3	$A \to B = \neg A \vee B$ и $A \to B = \neg B \to \neg A$		Снятие (замена) импликации
4	1) $A \leftrightarrow B = (A\&B) \vee (\neg A\&\neg B)$ 2) $A \leftrightarrow B = (A \vee \neg B)\&(\neg A \vee B)$		Снятие (замена) эквивалентности

Замена операций импликации и эквивалентности

Операций импликации и эквивалентности иногда нет среди логических операций конкретного компьютера или транслятора с языка программирования. Однако для решения многих задач эти операции необходимы. Существуют правила замены данных операций на последовательности операций отрицания, дизъюнкции и конъюнкции.

Так, заменить операцию импликации можно в соответствии со следующим правилом: $\mathbf{A \to B = \neg A \vee B}$

Для замены операции эквивалентности существует два правила:

$$A \Leftrightarrow B = (A \,\&\, B) \vee (\bar{A} \,\&\, \bar{B})$$

$$A \Leftrightarrow B = (A \vee \bar{B}) \,\&\, (\bar{A} \vee B)$$

В справедливости данных формул легко убедиться, построив таблицы истинности для правой и левой частей обоих тождеств.

Контрольные вопросы:

1. Какие логические операции применяются в компьютерах?
2. Как можно записать логическую операцию?
3. Охарактеризуйте основные операции

Понятие алгоритма. Классификация, структура и свойства алгоритмов. Базовые структуры алгоритмов.

План.

1. Понятие алгоритма
2. Классификация, структура и свойства алгоритмов
3. Базовые структуры алгоритмов

Алгоритм – последовательность инструкций, правильное выполнение которых ведет к решению поставленной задачи.

Основными свойствами алгоритма являются:

1. Однозначность(предписания алгоритма должны быть сформулированы таким образом, чтобы не допускалось никакой двусмысленности в их толковании).

2. Дискретность(алгоритм должен быть разделен на ряд инструкций, шагов, каждый из которых представляет собой отдельное законченное действие).

3. Результативность(решение задачи, для которой разработан алгоритм должно быть достигнуто за конечное число шагов).

4. Массовость(правильно разработанный алгоритм может использоваться для решения целого класса сходных между собой задач, которые отличаются друг от друга только значениями исходных параметров).

5. Конечность(каждое действие и алгоритм в целом должны иметь возможность завершения).

Алгоритм может быть сформулирован тремя различными способами:

1) Словесная формулировка алгоритма;

2) Формулировка в виде графической схемы (блок-схема);

3) Запись алгоритма в виде программы, записанной на одном из языков программирования.

Основные алгоритмические структуры:

1) **Линейный алгоритм** – описание действий, которые выполняются однократно в заданном порядке. Исполнитель выполняет действия последовательно, одно за другим в том порядке, в котором они следуют.

2) **Цикличный алгоритм** – описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие.

Перечень повторяющихся действий называют **телом цикла**. Циклические алгоритмы бывают двух типов:

· Циклы со счетчиком, в которых какие-то действия выполняются определенное число раз (используют, когда заранее известно какое число повторений тела цикла необходимо выполнить);

· Циклы с условием, в которых тело цикла выполняется, в зависимости от какого-либо условия. Различают циклы с предусловием и постусловием.

Р3) **Разветвляющий алгоритм –**алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

Условие – это высказывание, которое может быть либо истинно, либо ложно. Существует две формы ветвления – неполная (когда присутствует только одна ветвь, т.е. в зависимости от истинности условия либо выполняется, либо не выполняется действие) и полная (когда присутствуют две ветви, т.е. в зависимости от истинности условия выполняется либо одно, либо другое действие).

Языки программирования, классификация языков программирования. Понятие о системе программирования.

Язык программирования – это фиксированная система обозначений для описания алгоритмов и структур данных.

Языки программирования – искусственные языки. От естественных они отличаются ограниченным числом "слов", значение которых понятно транслятору, и очень строгими правилами записи команд (операторов).

Транслятор — средство для преобразования текстов из одного языка, понятного человеку, в другой язык, понятный компьютеру.

Транслятор программ может производиться двумя различными способами:

Интерпретатор (каждая инструкция, вводимая в ПК, сразу же преобразуется в машинный код и выполняется);

Компилятор (в машинный код переводится только вся программа в целом).

Изначально программирование производилось с помощью машинных кодов. **Машинный код** представляет собой запись команд для компьютера в виде чисел, представленных в двоичной системе счисления.

Язык машинных кодов и язык Ассемблера относятся к языкам **низкого уровня**, т.е. уровня приближенного к машинной логике.

Следующим этапом в развитии программирования явилось создание **языков программирования высокого уровня**, т.е. приближенного к естественным человеческим языкам.

Языки программирования высокого уровня облегчают процесс программирования, т.е. многие инструкции практически представляют собой упрощенные короткие фразы английского языка.

Программы, написанные на языке высокого уровня, являются машинно-независимыми, т.е. перенос программы с компьютера одной конструкции на другой не представляет собой серьезной проблемы.

Система программирования – совокупность программных средств, облегчающих написание, отладку диалоговой программы и автоматизирующих её многоэтапное преобразование в исполняемую программу и загрузку в память для выполнения.

Пример.

```
program Primer; {вычисление суммы двух чисел}
var
x,y,s: integer;
begin
WriteLn('Введите через пробел два числа ');
ReadLn(x,y);
s := x + y;
WriteLn('Сумма чисел равна ',s);
end.
```

Оператор присваивания. Выражение, приоритеты в выражениях.

Оператор присваивания – основной оператор любого языка программирования. Общая форма записи оператора:

имя величины := выражение

Например, V:=A; или V:=A+1;

переменная := выражение;

левая_часть := правая_часть;

Оператор работает **справа налево**, то есть сначала вычисляется то, что записано в правой части, а затем результат записывается «в левую часть».

При помощи оператора присваивания переменной могут присваиваться константы и выражения, значения переменных любого типа.

Как только в программе встречается переменная, для неё в памяти отводится место. Оператор присваивания помещает значение переменной или значение выражения в отведённое место.

Если в процессе выполнения программы встречается переприсваивание (т.е. та же самая переменная принимает другое значение), то старое значение переменной стирается, на свободное место записывается новое значение. Команда присваивания позволяет лучше понять смысл слова переменная (т.е. меняющая своё значение по ходу программы).

Выражение – это формула для вычисления значения. Она образуется из операндов, соединенных знаками операций и круглыми скобками. В качестве операндов могут выступать переменные, константы, указатели функций.

Тип переменной в левой части оператора присваивания обычно должен совпадать с типом значения выражения в правой части. Выражения являются составной частью операторов.

В Pascal выражения вычисляются в соответствии с приоритетами операций. Приоритеты выполнения операций следующие (в порядке убывания):

- одноместный минус;
- операция **NOT**;
- операции типа умножения;
- операции типа сложения;
- операции сравнения (отношения).

Одноместный минус применим к операндам арифметического типа. Операция **NOT**– к операндам логических и целых типов. Если в одном выражении несколько операций одного приоритета, то они выполняются, начиная слева. Приоритеты можно изменить, поставив скобки. В логических выражениях необходимы скобки во избежание конфликта типа по приоритету.

Например, если в выражении **… (X > 5) AND (Y > 10) …** не поставить скобки, то будет синтаксическая ошибка, так как приоритет операции **AND** выше приоритета операций сравнения >.

операции типа умножения – это ***, /, div, mod, and**

операции типа сложения – это **+, -, or, xor**

операции сравнения – это **<>, <, >, <=, >=, in**

Сравнение строк символов выполняется слева направо посимвольно. Более короткие строки дополняются пробелами справа.

Оператор цикла с заранее известным количеством повторений

Инструкция for используется в том случае, если некоторую последовательность действий (инструкций программы) надо выполнить несколько раз, причем число повторений известно.

В общем виде инструкция for записывается следующим образом:

```
for счетчик := нач_знач to кон_знач do
begin
// здесь инструкции, которые надо выполнить несколько раз
end
```

Переменная счетчик, выражение нач_знач и кон_знач должны быть целого типа.

Если в инструкции for вместо слова to записать downto, то поле очередного выполнения инструкций тела цикла значение счетчика будет не увеличиваться, а уменьшаться.

Текстовые.

Перед использованием такого файла в программе в разделе описания переменных должна быть обязательно описана связанная с ним переменная. Общий вид описания следующий:

var *Имя_переменной*: text;

где:

Имя_переменной – имя переменной, которая впоследствии будет связана с файлом;

text — соответствующий тип переменной.

Для того, чтобы с файлом в дальнейшем можно было производить какие-либо действия, необходимо выполнить еще две операции. Во-первых – связать имя файла с именем соответствующей ему переменной. Во-вторых – указать способ работы с файлом – запись в него информации или чтение информации.

Типизированные.

Типизированным называется файл, который состоит из однородных элементов, относящихся к одному и тому же типу. Такими элементами могут

быть переменные различных типов и массивы. Каждый элемент такого файла

имеет свой индекс, подобно элементам массива. Но в отличие от обычного массива, который хранится в оперативной памяти компьютера, и содержимое

которого стирается после выхода из системы программирования, типизированный файл может длительное время сохраняться на жестком диске компьютера или на другом носителе. К каждому элементу такого

файла возможен как прямой (по индексу элемента), так и последовательный доступ, но недостатком его является то, что в отличие от текстового файла,

его нельзя просматривать и редактировать с помощью обычного текстового редактора.

Для того, чтобы в типизированный файл можно было производить

запись или считывать из него информацию, его следует связать с помощью

процедуры assign с файловой переменной подобно тому, как мы это уже

делали с текстовыми файлами. Например, так:

assign (nab,‘D:\inf\nabor’);

Контрольные вопросы:

1. *Что такое алгоритм?*
2. *Какие алгоритмы бывают*?
3. *Как можно записать алгоритм*?

Тема 2.2. Принципы организации ЭВМ

План.

1. Контрольный опрос
2. Основные функциональные элементы ЭВМ;
3. Общее устройство ЭВМ
4. структура вычислительной системы;

Понятие термина "Архитектура ЭВМ"Под термином "архитектура ЭВМ" принято понимать совокупность общих принципов организации аппаратно-программных средств и их основных характеристик, определяющих функциональные возможности ЭВМ. Архитектура ЭВМ охватывает обширный круг проблем, связанных с созданием комплекса аппаратных и программных средств и учитывающих большое количество определяющих факторов. Среди этих факторов самыми главными являются: стоимость,

Сфера применения, функциональные возможности, удобство в эксплуатации.

Основным компонентом архитектуры считаются аппаратные средства. Архитектуру вычислительного средства необходимо отличать от его структуры.

Структура вычислительного средства определяет его текущий состав на определенном уровне детализации и описывает связи внутри средства. Архитектура же определяет основные правила взаимодействия составных элементов вычислительного средства, описание которых выполняется в той мере, в какой необходимо для формирования правил их взаимодействия. Она устанавливает не все связи, а наиболее необходимые, которые должны быть известны для более грамотного использования применяемого средства.
Так, пользователю ЭВМ не важно, на каких элементах выполнены электронные схемы и т. д. Важно несколько другое:
как те или иные структурные особенности ЭВМ связаны с возможностями, предоставляемыми пользователю,
какие альтернативные решения реализованы при создании машины,
по каким критериям принимались решения,
как связаны между собой характеристики устройств, входящих в состав ЭВМ,
какое действие они оказывают на общие характеристики компьютера.
Другими словами, архитектура ЭВМ действительно отражает круг проблем, которые относятся к общему проектированию и построению вычислительных машин и их программного обеспечения. Структура вычислительной системыИз чего состоит любая вычислительная система?

Во-первых, из того, что в англоязычных странах принято называть словом hardware, или техническое обеспечение: процессор, память, монитор, дисковые устройства и т.д., объединенные магистральным соединением, которое называется шиной. Во-вторых, вычислительная система состоит из программного обеспечения (software).
Некоторые сведения об архитектуре компьютера

Основными аппаратными компонентами компьютера являются: основная память, центральный процессор и периферийные устройства.

Для обмена данными между собой эти компоненты соединены группой проводов, называемой магистралью. Рис. 1.1. Некоторые компоненты компьютера
Основная память используется для запоминания программ и данных в двоичном виде и организована в виде упорядоченного массива ячеек, каждая из которых имеет уникальный цифровой адрес. Как правило, размер ячейки составляет один байт. Типовые операции над основной памятью – считывание и запись содержимого ячейки с определенным адресом. Выполнение различных операций с данными осуществляется изолированной частью компьютера, называемой центральным процессором (ЦП).

ЦП также имеет ячейки для запоминания информации, называемые регистрами. Их разделяют на регистры общего назначения и специализированные регистры. В современных компьютерах емкость регистра обычно составляет 4–8 байт. Регистры общего назначения используются для временного хранения данных и результатов операций. Для обработки информации обычно организовывается передача данных из ячеек памяти в регистры общего назначения, выполнение операции центральным процессором и передача результатов операции в основную память.

Специализированные регистры используются для контроля работы процессора. Наиболее важными являются: программный счетчик, регистр команд и регистр, содержащий информацию о состоянии программы.

Программы хранятся в виде последовательности машинных команд, которые должен выполнять центральный процессор. Каждая команда состоит из поля операции и полей операндов, то есть тех данных, над которыми выполняется данная операция. Весь набор машинных команд называется машинным языком.

Выполнение программы осуществляется следующим образом. Машинная команда, на которую указывает программный счетчик, считывается из памяти и копируется в регистр команд. Здесь она декодируется, после чего исполняется. После выполнения

команды программный счетчик указывает на следующую команду. Эти действия, называемые машинным циклом, затем повторяются.

Взаимодействие с периферийными устройствами

Периферийные устройства предназначены для ввода и вывода информации. Каждое устройство обычно имеет в своем составе специализированный компьютер, называемый контроллером или адаптером. Когда контроллер вставляется в разъем на материнской плате, он подключается к шине и получает уникальный номер (адрес). После этого контроллер осуществляет наблюдение за сигналами, идущими по шине, и отвечает на сигналы, адресованные ему.

Любая операция ввода-вывода предполагает диалог между ЦП и контроллером устройства. Когда процессору встречается команда, связанная с вводом-выводом, входящая в состав какой-либо программы, он выполняет ее, посылая сигналы контроллеру устройства. Это так называемый программируемый ввод-вывод.

В свою очередь, любые изменения с внешними устройствами имеют следствием передачу сигнала от устройства к ЦП. С точки зрения ЦП это является асинхронным событием и требует его реакции. Для того чтобы обнаружить такое событие, между машинными циклами процессор опрашивает специальный регистр, содержащий информацию о типе устройства, сгенерировавшего сигнал. Если сигнал имеет место, то ЦП выполняет специфичную для данного устройства программу, задача которой – отреагировать на это событие надлежащим образом (например, занести символ,

введенный с клавиатуры, в специальный буфер).

Такая программа называется программой обработки прерывания, а само событие прерыванием, поскольку оно нарушает плановую работу процессора. После завершения обработки прерывания процессор возвращается к выполнению программы. Эти действия компьютера называются вводом-выводом с использованием прерываний. В современных компьютерах также имеется возможность непосредственного взаимодействия между контроллером и основной памятью, минуя ЦП, – так называемый механизм прямого доступа к памяти.

Контрольные вопросы:

1. Что такое архитектура ЭВМ?
2. Из чего состоит любая вычислительная система?
3. Назовите основные аппаратные компоненты.
4. Как организована основная память?
5. Регистры ЦП и их назначение.
6. Для чего предназначены периферийные устройства?
7. Что такое адаптер или контроллер?

Базовые понятия архитектуры вычислительных систем. Основные принципы построения архитектур вычислительных систем. Принципы Фон Неймана.

План.

1. Контрольный опрос
2. Базовые понятия архитектуры вычислительных систем
3. Основные принципы построения архитектур вычислительных систем
4. . Принципы Фон Неймана.

Архитектура компьютера закрытого типа.
Компьютерами с сосредоточенной обработкой называются такие вычислительные системы, у которых одно или несколько обрабатывающих устройств (процессоров)

расположены компактно и используют для обмена информацией внутренние шины передачи данных. Компьютеры 1-го и 2-го поколения имели архитектуру закрытого типа с ограниченным набором внешнего оборудования. Компьютер, выполненный по этой архитектуре, не имел возможности подключения дополнительных устройств, не предусмотренных разработчиком. Укрупненная схема такой компьютерной архитектуры приведена на рис. 1. Оперативная память хранит команды и данные исполняемых программ. АЛУ обеспечивает не только числовую обработку, но и участвует в процессе ввода-вывода информации, осуществляя ее занесение в оперативную память. Канал ввода / вывода представляет собой специализированное устройство, работающее по командам, подаваемым устройством управления. Канал допускает подключение определенного числа внешних устройств. Устройство управления обеспечивает выполнение команд программы и управляет всеми узлами системы.

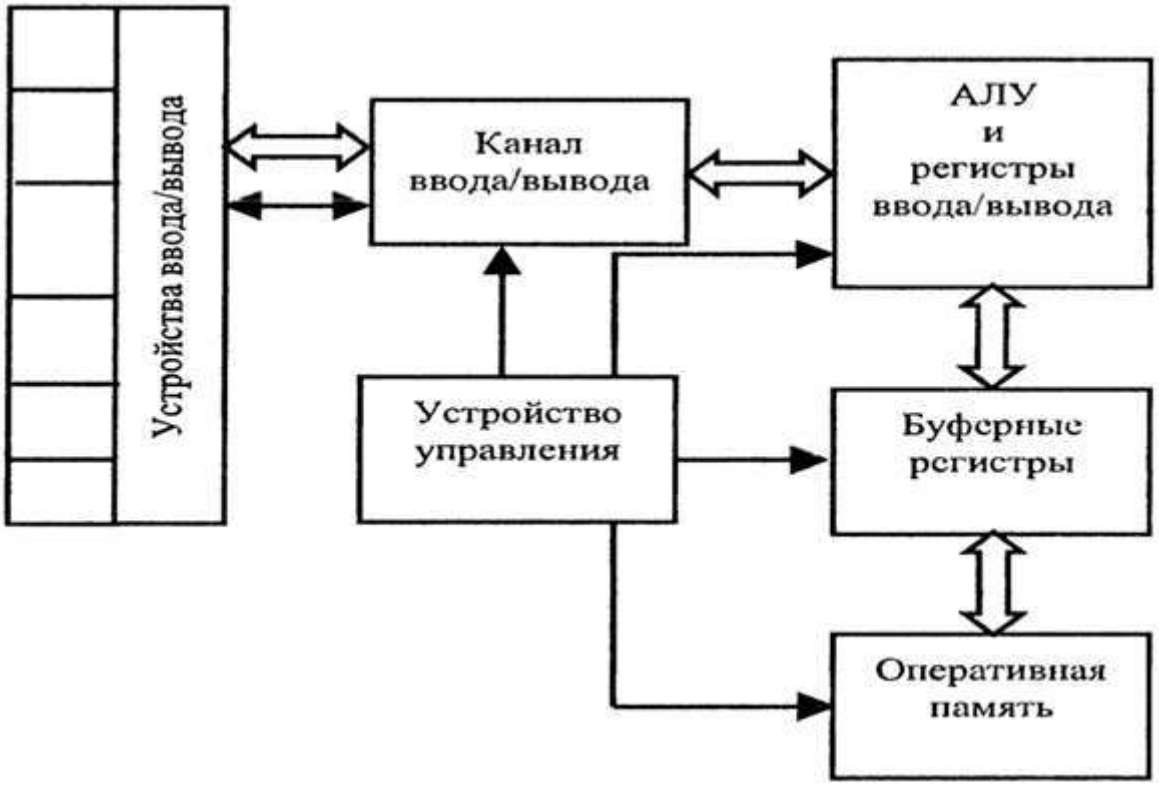


Рис. 1. Архитектура компьютера закрытого типа
Компьютеры такой архитектуры эффективны при решении чисто вычислительных задач. Они плохо приспособлены для реализации компьютерных технологий, требующих подключения дополнительных внешних устройств и высокой скорости обмена с ними информацией.

Архитектуры компьютеров открытого типа.

Открытая архитектура — архитектура компьютера, периферийного устройства или же программного обеспечения, на которую опубликованы спецификации, что позволяет другим производителям разрабатывать дополнительные устройства к системам с такой архитектурой.

В начале 70-х гг. фирмой DEC (Digital Equipment Corporation) был предложен компьютер совершенно иной архитектуры. Эта архитектура позволяла свободно подключать любые периферийные устройства, что сразу же заинтересовало разработчиков систем управления различными техническими системами, так как обеспечивало свободное подключение к компьютеру любого числа датчиков и исполнительных механизмов. Главным нововведением являлось подключение всех устройств, независимо от их назначения, к

общей шине передачи информации. Подключение устройств к шине осуществлялось в соответствии со стандартом шины. Стандарт шины являлся свободно распространяемым документом, что позволяло фирмам – производителям периферийного оборудования разрабатывать контроллер для подключения своих устройств к шинам различных стандартов. Архитектура компьютера открытого типа, основанная на использовании общей шины, приведена на рис. 2.

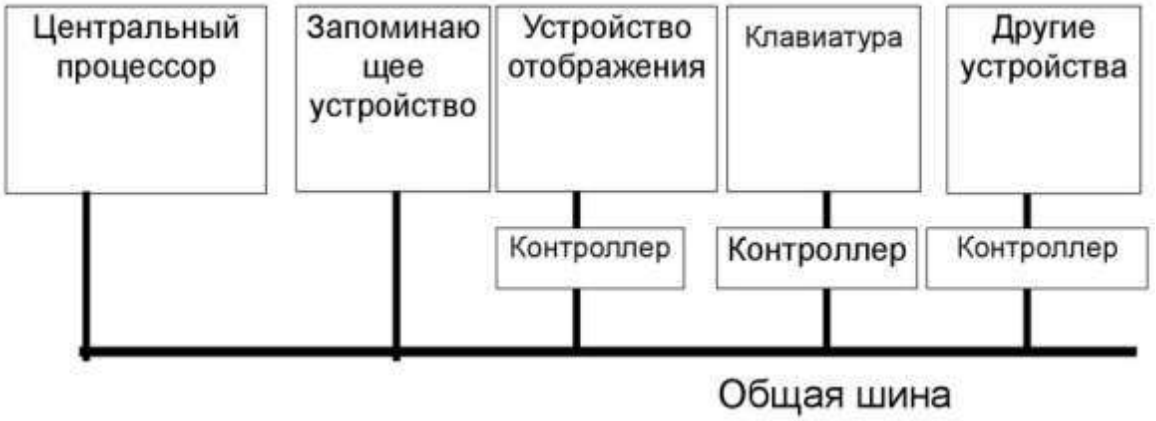


Рис. 2. Архитектура компьютера открытого типа

Общее управление всей системой осуществляет центральный процессор. Он управляет общей шиной, выделяя время другим устройствам для обмена информацией. Запоминающее устройство хранит исполняемые программы и данные и согласовано уровнями своих сигналов с уровнями сигналов самой шины. Внешние устройства, уровни сигналов которых отличаются от уровней сигналов шины, подключаются к ней через специальное устройство – контроллер. Контроллер согласовывает сигналы устройства с сигналами шины и осуществляет управление устройством по командам, поступающим от центрального процессора. Контроллер подключается к шине специальными устройствами – портами ввода-вывода. Каждый порт имеет свой номер, и обращение к нему процессора происходит, также как и к ячейке памяти, по этому номеру. Процессор имеет специальные линии управления, сигнал на которых определяет, обращается ли процессор к ячейке памяти или к порту ввода-вывода контроллера внешнего устройства. Несмотря на преимущества, предоставляемые архитектурой с общей шиной, она имеет и серьезный недостаток, который проявлялся все больше при повышении производительности внешних устройств и возрастании потоков обмена информацией между ними. К общей шине подключены устройства с разными объемами и скоростью обмена, в связи с чем «медленные» устройства задерживали работу «быстрых». Дальнейшее повышение производительности компьютера было найдено во введении дополнительной локальной шины, к которой подключались «быстрые» устройства. Архитектура компьютера с общей и локальной шинами приведена на рис. 3.

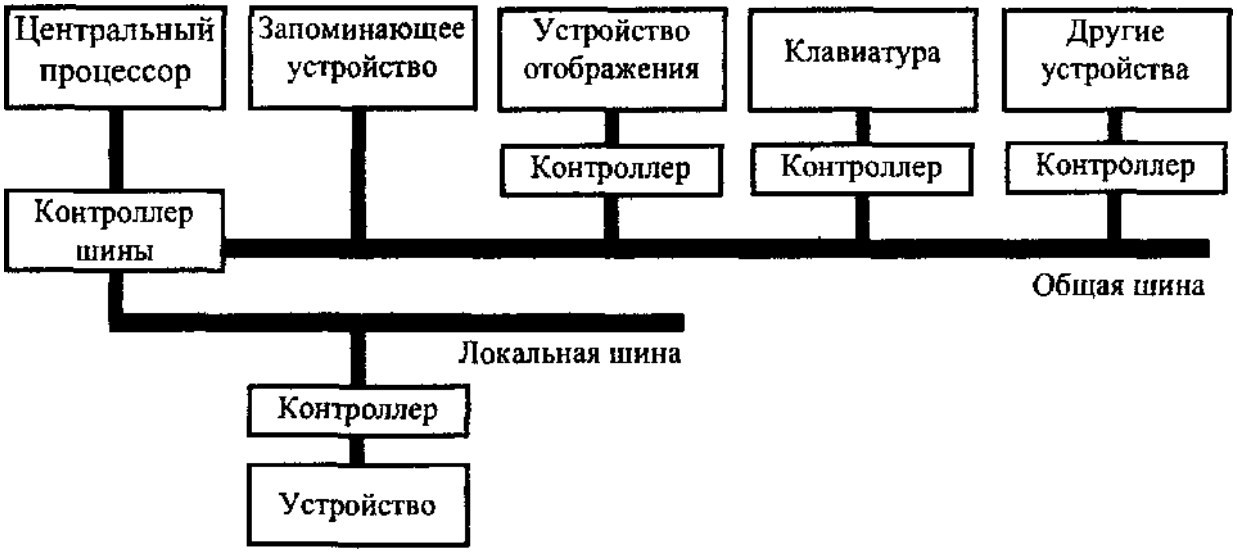


Рис. 3. Архитектура компьютера с общей и локальной шиной

Контроллер шины анализирует адреса портов, передаваемые процессором, и передает их контроллеру, подключенному к общей или локальной шине. Конструктивно контроллер каждого устройства размещается на общей плате с центральным процессором и запоминающим устройством или, если устройство не является стандартно входящим в состав компьютера, на специальной плате, вставляемой в специальные разъемы на общей плате – слоты расширения. Дальнейшее развитие микроэлектроники позволило размещать несколько функциональных узлов компьютера и контроллеры стандартных устройств в одной микросхеме СБИС. Это сократило количество микросхем на общей плате и дало возможность ввести две дополнительные локальные шины для подключения запоминающего устройства и устройства отображения, которые имеют наибольший объем обмена с центральным процессором и между собой. Хотя архитектура компьютера осталась прежней, структура современного персонального компьютера имеет вид, представленный на рис. 2.12,

Рис. 4. Структура персонального компьютера

Центральный контроллер играет роль коммутатора, распределяющего потоки информации между процессором, памятью, устройством отображения и остальными узлами компьютера. Кроме этого в состав микросхемы центрального контроллера

включены устройства, которые поддерживают работу компьютера. К ним относятся системный таймер; устройство прямого доступа к памяти, которое обеспечивает обмен данными между внешними устройствами и памятью и периоды, когда это не требуется процессору; устройство обработки прерываний, которое обеспечивает быструю реакцию процессора на запросы внешних устройств, имеющих данные для передачи. Функциональный контроллер – это СБИС, которая содержит контроллеры для подключения стандартных внешних устройств, таких как клавиатура, мышь, принтер, модем и т.д. Часто в состав этого контроллера входит такое устройство, как аудиокарта, позволяющая получить на внешних динамиках высококачественный звук при прослушивании музыкальных и речевых файлов.
Для подключения специфических устройств часть обшей шины, соединяющая центральный и функциональный контроллеры, имеет слоты расширения для установки плат контроллеров.

Архитектуры многопроцессорных вычислительных систем.

Контрольные вопросы:

1. Что архитектура ЭВМ?;
2. Основные архитектуры ЭВМ?;
3. Что такое открытая архитектура ?;
4. Ч Что такое закрытая архитектура ?

Тема 2.3. Классификация и типовая структура микропроцессоров

Классификация ЭВМ по областям применения

По идеологии открытых систем, все вычислительные платформы должны удовлетворять любые запросы пользователей. Но реализация общего ядра для всех приложений, как и всякая универсализация, ведет к большим накладным расходам. Пользователей интересует не только интерфейс с системой, но время ответа и стоимость услуги. Вычислительные платформы, как комплекс программно-аппаратного оборудования, операционного и сетевого окружения можно классифицировать по спектру информационных услуг, предоставляемых пользователям.
Грубая классификация пользователей /Л.Н. Королев/ открытых систем и аппаратного оборудования такова:
Самым массовым пользователем открытых систем можно считать владельцев ПК, используемых для бытовых нужд, число таких пользователей приближается к сотне миллионов. Современные платформы - ПК таких пользователей предоставляют им доступ к сетям по телефонным каналам. Оборудование - 90% платформы IBM PC (Intei x86) и программное обеспечение, созданное для этой архитектуры.
Пользователи, работающие в сфере бизнеса: банковская сфера,маркетинг, складской учет и т.д. Им требуется доступ к более производительной вычислительной технике, к глобальным сетям, требуются услуги по созданию и изменению СУБД и другие средства для разработки своих приложений. Использует ПК, но переходит на рабочие станции, хост-машины, до многопроцессорных супер-ЭВМ.
Пользователи этого инженерного класса занимаются разработкой приложений для

промышленного производства. Для них характерен доступ к пакетам прикладных программ. Ориентируется на рабочие станции фирм DEC,HP и другие.
Пользователи, проводящие научные расчеты. Для этого класса пользователей необходим доступ высокопроизводительным вычислителям. Мощные платформы: Amdal(IBM), Cray(CDC),SPP(HP).
Пользователи - студенты, требующие услуги по освоению новых информационных технологий. Высшая школа и университетская наука, главным образом, ориентируется на рабочии станции SUN.
Основные характеристики, области применения ЭВМ различных классов
Понятие архитектуры ЭВМ
Сложность современных вычислительных машин закономерно привела к понятию архитектуры вычислительной машины, охватывающей комплекс общих вопросов ее построения, существенных в первую очередь для пользователя, интересующегося главным образом возможностями машины, а не деталями ее технического исполнения.
Круг вопросов, подлежащих рассмотрению при изучении архитектуры ЭВМ, можно условно разделить на вопросы общей структуры, организации вычислительного процесса и общения пользователя с машиной, вопросы логической организации представления, хранения и преобразования информации и вопросы логической организации совместной работы различных устройств, а также аппаратных и программных средств машины.
Основные характеристики ЭВМ
Важнейшими эксплуатационными характеристиками ЭВМ являются ее производительность P и общий коэффициент эффективности машины:
Э = P / (СЭВМ + СЭКС),
представляющий собой отношение ее производительности к сумме стоимости самой машины СЭВМ и затрат на ее эксплуатацию за определенный период времени (например, период окупаемости капитальных затрат) СЭКС.
Так как часто трудно оценить затраты на эксплуатацию данной ЭВМ, а создатели новых машин стремятся приравнять эти затраты к нулю, то оценивают эффективность машины по упрощенной формуле
Э’ = P / СЭВМ .К наиболее распространенным характеристикам ЭВМ относятся:
число разрядов в машинном слове (влияет на точность вычислений и диапазон представляемых в машине чисел);
скорость выполнения основных видов команд;
емкость оперативной памяти;
максимальная скорость передачи информации между ядром ЭВМ (процессор или память) и внешним периферийным оборудованием;
эксплуатационная надежность машины.
При создании новых ЭВМ обеспечивается значительное возрастание отношений производительность/стоимость и надежность/стоимость.
СуперЭВМВ настоящее время к сверх производительным машинам (системам) относят машины с производительностью в сотни и более GFLOP/s. Подобные машины используются для решения особенно сложных научно-технических задач, задач обработки больших объемов данных в реальном масштабе времени, поиска оптимальных в задачах экономического планирования и автоматического проектирования сложных объектов.
Самым ярким примером служит деятельность Cray Research. Эта фирма долго лидировала на рынке суперЭВМ. Но с разрушением «железного занавеса» спрос на ее компьютеры упал, что привело к распаду корпорации. В прошлом году в автокатастрофе погиб и ее основатель – Симур Крей.
Долгое время лидером в области суперкомпьютеров оставалась Cray Research,. По данным на начало 1997 года она занимала 43% всего рынка. Cray Research, приобретенная корпорацией Silicon Graphics в начале 1996 г, продает широкий спектр систем, начиная со старых моделей семейства J90 до машин новой серии Origin, в которых используется

архитектура коммутации, построенная на базе процессора MIPS R10000.
Hewlett-Packard, владеет 7% этого сегмента рынка. Другими американскими производителями мощных компьютеров являются IBM, которая строит свои суперкомпьютеры SP на многокристальной версии PowerPC (14% рынка), а также Digital Equipment, предлагающая кластеры SMP-систем на базе процессора Alpha (13% рынка). И наконец, японские фирмы Fujitsu и NEC занимают твердые позиции на рынке суперкомпьютеров, имея доли в 8 и 4% соответственно.
Сегодня самые быстрые суперЭВМ принадлежат Intel. В настоящее время Intel выполняет заказ министерства энергетики США.
В архитектуре суперЭВМ обнаруживается ряд принципиальных отличий от классической фоннеймонавской модели ЭВМ. Различные архитектуры суперЭВМ будут рассмотрены в теме «архитектурные особенности организации ЭВМ различных классов»
Малые и микроЭВМ.
Имеется большое число, условно говоря, «малых» применений вычислительных машин, таких, как автоматизация производственного контроля изделий, обработка данных при экспериментах, прием и обработка данных с линии связи, управление технологическими процессами, управление станками и разнообразными цифровыми терминалами, малые расчетные инженерные задачи.
В настоящее время малые и микроЭВМ встраивают в различные «умные» приборы (электросчетчики, микроволновки, стиральные машины, модемы, датчики и т.д.).
МинисуперЭВМ и суперминиЭВМ.
В классификации отсутствуют четкие границы между рассмотренными типами ЭВМ. В последнее время стали выделять два промежуточных типа.
К суперминиЭВМ относят высокопроизводительные ЭВМ содержащих один или несколько слабосвязанных процессоров, объединенных с общей магистралью (общей шиной). Для суперминиЭВМ характерно, что скорость выполнения его арифметических операций над числами с плавающей точкой существенно ниже скорости работы, определяемой по смеси команд, соответствующей информационно-логическим запросам. К этому типу можно отнести IBM-овский шахматный компьютер Deep Blue.
МинисуперЭВМ – это упрощенные (в частности за счет более короткого слова) многопроцессорные ЭВМ, чаще всего со средствами векторной и конвейерной обработки, с высокой скоростью выполнения операций над числами с плавающей точкой. К этому типу можно отнести ЭВМ с SMP(Symmetric multiprocessor) архитектурой.

Контрольные вопросы:

1. Какие основные типы ЭВМ классифицированы?
2. Классификация ЭВМ по типу;
3. Классификация ЭВМ по применению;
4. Классификация ЭВМ по поколению;
5. Классификация ЭВМ по архитектуре;

Тема 2.4. Технология повышения производительности
Тема 2.5. Компоненты системного блока

Арифметико-логическое устройство и устройство управления: назначение, принцип работы. Процессор: устройство и принцип работы.

Системы команд и классы процессоров: CISC, RISC, MISC, VLIM

План.

1. Контрольный опрос
2. Арифметико-логическое устройство.
3. Процессор: устройство и принцип работы
4. Системы команд и классы процессоров: CISC, RISC, MISC, VLIM

Теоретическая часть

Архитектура ЭВМ включает в себя как структуру, отражающую состав ПК, так и программно – математическое обеспечение. Структура ЭВМ - совокупность элементов и связей между ними. Основным принципом построения всех современных ЭВМ является программное управление.Основы учения об архитектуре вычислительных машин были заложены Джон фон Нейманом. Совокупность этих принципов породила классическую (фон-неймановскую) архитектуру ЭВМ. Фон Нейман не только выдвинул основополагающие принципы логического устройства ЭВМ, но и предложил ее структуру, представленную на рисунке.

Положения фон Неймана:

- Компьютер состоит из нескольких основных устройств (арифметико-логическое устройство, управляющее устройство, память, внешняя память, устройства ввода и вывода)
- Арифметико-логическое устройство – выполняет логические и арифметические действия, необходимые для переработки информации, хранящейся в памяти
- Управляющее устройство – обеспечивает управление и контроль всех устройств компьютера (управляющие сигналы указаны пунктирными стрелками)
- Данные, которые хранятся в запоминающем устройстве, представлены в двоичной форме
- Программа, которая задает работу компьютера, и данные хранятся в одном и том же запоминающем устройстве
- Для ввода и вывода информации используются устройства ввода и выводаОдин из важнейших принципов – принцип хранимой программы – требует, чтобы программа закладывалась в память машины так же, как в нее закладывается исходная информация.

Арифметико-логическое устройство и устройство управления в современных компьютерах образуют процессор ЭВМ. Процессор, который состоит из одной или нескольких больших интегральных схем называется микропроцессором или микропроцессорным комплектом.Процессор – функциональная часть ЭВМ, выполняющая основные операции по обработке данных и управлению работой других блоков. Процессор является преобразователем информации, поступающей из памяти и внешних устройств.Запоминающие устройства обеспечивают хранение исходных и промежуточных данных, результатов вычислений, а также программ. Они включают: оперативные (ОЗУ), сверхоперативные СОЗУ), постоянные (ПЗУ) и внешние (ВЗУ) запоминающие устройства.Оперативные ЗУ хранят информацию, с которой компьютер работает непосредственно в данное время (резидентная часть операционной системы, прикладная программа, обрабатываемые данные). В СОЗУ хранится наиболее часто используемые процессором данные. Только та информация, которая хранится в СОЗУ и ОЗУ, непосредственно доступна процессору.Внешние запоминающие устройства (накопители на магнитных дисках, например, жесткий диск или винчестер) с емкостью намного больше, чем ОЗУ, но с существенно более медленным доступом, используются для длительного хранения больших объемов информации. Например, операционная система (ОС) хранится на жестком диске, но при запуске компьютера резидентная часть ОС загружается в ОЗУ и находится там до завершения сеанса работы ПК. ПЗУ (постоянные запоминающие устройства) и ППЗУ (перепрограммируемые постоянные запоминающие устройства) предназначены для постоянного хранения информации, которая записывается туда при ее изготовлении, например, ППЗУ для BIOS.В качестве устройства ввода информации служит, например, клавиатура. В качестве устройства вывода – дисплей, принтер и т.д.В построенной по схеме фон Неймана ЭВМ происходит последовательное считывание команд из памяти и их выполнение. Номер (адрес) очередной ячейки памяти, из которой будет извлечена следующая команда программы,

указывается специальным устройством – счетчиком команд в устройстве управления. Классификация процессоров. Cisc, risc, vliw, суперскалярные процессоры, misc.

CISC (Complete Instruction Set Computer) – полный набор команд микропроцессора.

- нефиксированное значение длины команды;
- арифметические действия кодируются в одной команде;
- небольшое число регистров, каждый из которых выполняет строго определённую функцию.

Недостатки CISC архитектуры:

- высокая стоимость аппаратной части;
- сложности с распараллеливанием вычислений.

Состав и назначение их регистров существенно неоднородны, широкий набор команд усложняет декодирование инструкций, на что расходуются аппаратные ресурсы. Возрастает число тактов, необходимое для выполнения инструкций. К процессорам с полным набором инструкций относится семейство x86.

Лидером в разработке CISC-процессоров считается компания Intel со своей серией x86 и Pentium. Эта архитектура является практическим стандартом для рынка микрокомпьютеров.

Для CISC-процессоров характерно:

- сравнительно небольшое число регистров общего назначения;
- большое количество машинных команд, некоторые из которых нагружены семантически аналогично операторам высокоуровневых языков программирования и выполняются за много тактов;
- большое количество методов адресации;
- большое количество форматов команд различной разрядности;
- преобладание двухадресного формата команд;
- наличие команд обработки типа регистр-память.

RISC - Reduced Instruction Set Computer – архитектура компьютера с сокращенным набором команд. RISC-архитектура предполагает реализацию в ЭВМ сокращенного набора простейших, но часто употребляемых команд, что позволяет упростить аппаратурные средства процессора и благодаря этому получить возможность повысить его быстродействие.

Современные процессоры типа RISC характеризуются следующими особенностями:

1. упрощенный набор команд, имеющих одинаковую длину.
2. Отсутствуют макрокоманды, усложняющие структуру процессора и уменьшающую скорость его работы.
3. Взаимодействие с оперативной памятью ограничивается операциями пересылки данных.
4. Уменьшено число способов адресации
5. Используется конвейер команд.
6. Применяется высокоскоростная память.

Новый подход к архитектуре команд процессора значительно сократил площадь, требуемой для него на кристалле интегральной микросхемы. Это позволило резко увеличить число регистров (более 100 по лекциям, а вообще в типовых RISC-процессорах реализуются 32 или большее число регистров по сравнению с 8 - 16 регистрами в CISC-архитектурах). В результате процессор стал на 20-30% реже обращаться к оперативной памяти. Упростилась топология процессора, сократились сроки ее разработки, она стала дешевле.

Особенностью RISC архитектуры является механизм **перекрывающихся окон**, предназначенный для уменьшения числа обращений к оперативной памяти и межрегистровых передач, что способствует увеличению производительности ЭВМ.

Наиболее широко используемые в настольных компьютерах процессоры архитектуры x86 ранее являлись CISC-процессорами, однако новые процессоры, начиная с Intel 486DX, являются CISC-процессорами с RISC-ядром. Они непосредственно перед исполнением преобразуют CISC-инструкции x86-процессоров в более простой набор внутренних инструкций RISC.

Суперскалярные процессоры – дальнейшее развитие конвейеризации. Их отличительной особенностью является возможность выполнения нескольких команд за один процессорный цикл.

Архитектура их вычислительного ядра использует несколько декодеров команд, которые могут нагружать работой множество исполнительных блоков. Если в процессе работы команды, обрабатываемые конвейером, не противоречат друг другу, и одна не зависит от результата другой, то такое устройство может осуществить параллельное выполнение команд. В суперскалярных системах формирование расписания управления команд возлагается на микропроцессор, что требует много ресурсов.

При описании архитектуры суперскалярного процессора часто используется модель **окна исполнения**. При исполнении программы микропроцессор как бы продвигает по статической структуре программы окно исполнения. Команды в окне исполнения могут исполняться параллельно, если между ними **нет зависимости**. Для устранения зависимостей, вызванных командами перехода, используется **метод предсказания**.

VLIW – появилась в России. Она не попадает под принципы фон Неймана (нарушает принцип программного управления, т.е. последовательного выполнения команд).

Архитектура ЭВМ с **длинным командным словом** (**VLIW** – Very Long Instruction Word) позволяет сократить объем оборудования, требуемого для реализации параллельной выдачи несколько команд, базируется на множестве независимых командных устройств. Вместо того чтобы выдавать на эти устройства последовательные команды, операции упаковываются в одну очень длинную команду. Ответственность за вывод параллельно выдаваемых для выполнения команд полностью ложится на компилятор.

В процессорах VLIW задача решается распределения работы между вычислительными модулями во время компиляции. Аппаратные средства, необходимые для реализации параллельной обработки, отсутствуют.

Для машин с VLIW-архитектурой был разработан новый метод планирования выдачи команд – **трассировочное планирование**. Во время планирования генерируется длинное командное слово. Процесс упаковки команд последовательной программы в длинные командные слова продолжается до тех пор, пока не будет оптимизирована вся программа.

Примеры процессоров на этой архитектуре – Intel Itanium (архитектура IA-64, Merced), микропроцессоры серии «Эльбрус» («Эльбрус 2000», «Эльбрус S»).

MISC, minimal instruction set computer — минимальный набор команд компьютера.

Увеличение разрядности процессоров привело к идее укладки нескольких команд в одно большое слово (связку, bound). Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных. Кроме этого, MISC использует стековую модель вычислений.

Процессоры, образующие «компьютеры с минимальным набором команд» (MISC), как и процессоры RISC, характеризуются небольшим числом чаще всего встречающихся команд. Вместе с этим, принцип «очень длинных командных слов» (VLIW) обеспечивает выполнение группы непротиворечивых 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 и VLIW-концепции. Компоненты процессора просты и работают на высоких частотах.

Контрольные вопросы:

1. Для чего используется АЛУ?
2. Назовите основные блоки процессора?
3. Что такое CISC?
4. Что такое RISK?
5. Что такое MISC?

Процессор: устройство и принцип работы.

План лекции:

1 История
2 Архитектура фон Неймана
2.1 Конвейерная архитектура
2.2 Суперскалярная архитектура
2.3 CISC-процессоры
2.4 RISC-процессоры
2.5 MISC-процессоры
2.6 VLIW-процессоры
2.7 Многоядерные процессоры

Системы команд и классы процессоров: CISC, RISC, MISC, VLIM.

Изначально термин центральное процессорное устройство описывал специализированный класс логических машин, предназначенных для выполнения сложных компьютерных программ. Вследствие довольно точного соответствия этого назначения функциям существовавших в то время компьютерных процессоров он естественным образом был перенесён на сами компьютеры. Начало применения термина и его аббревиатуры по отношению к компьютерным системам было положено в 1960-е годы. Устройство, архитектура и реализация процессоров с тех пор неоднократно менялись, однако их основные исполняемые функции остались теми же, что и прежде.

Главными характеристиками ЦПУ являются: тактовая частота, производительность, энергопотребление, нормы литографического процесса, используемого при производстве (для микропроцессоров), и архитектура.

Ранние ЦП создавались в виде уникальных составных частей для уникальных и даже единственных в своём роде компьютерных систем. Позднее от дорогостоящего способа разработки процессоров, предназначенных для выполнения одной единственной или нескольких узкоспециализированных программ, производители компьютеров перешли к серийному изготовлению типовых классов многоцелевых процессорных устройств. Тенденция к стандартизации компьютерных комплектующих зародилась в эпоху бурного развития полупроводниковых элементов, мейнфреймов и мини-компьютеров, а с появлением интегральных схем она стала ещё более популярной. Создание микросхем позволило ещё больше увеличить сложность ЦП с одновременным уменьшением их физических размеров. Стандартизация и миниатюризация процессоров привели к глубокому проникновению основанных на них цифровых устройств в повседневную жизнь человека. Современные процессоры можно найти не только в таких высокотехнологичных устройствах, как компьютеры, но и в автомобилях, калькуляторах,

мобильных телефонах и даже в детских игрушках. Чаще всего они представлены микроконтроллерами, где, помимо вычислительного устройства, на кристалле расположены дополнительные компоненты (память программ и данных, интерфейсы, порты ввода-вывода, таймеры и др.). Современные вычислительные возможности микроконтроллера сравнимы с процессорами персональных ЭВМ десятилетней давности, а чаще даже значительно превосходят их показатели.

Архитектура фон Неймана

Основная статья: Архитектура фон Неймана

Большинство современных процессоров для персональных компьютеров в общем основано на той или иной версии циклического процесса последовательной обработки данных, изобретённого Джоном фон Нейманом.

Дж. фон Нейман придумал схему постройки компьютера в 1946 году.

Отличительной особенностью архитектуры фон Неймана является то, что инструкции и данные хранятся в одной и той же памяти.

В различных архитектурах и для различных команд могут потребоваться дополнительные этапы. Например, для арифметических команд могут потребоваться дополнительные обращения к памяти, во время которых производится считывание операндов и запись результатов.

Этапы цикла выполнения:

Процессор выставляет число, хранящееся в регистре счётчика команд, на шину адреса и отдаёт памяти команду чтения.

Выставленное число является для памяти адресом; память, получив адрес и команду чтения, выставляет содержимое, хранящееся по этому адресу, на шину данных и сообщает о готовности.

Процессор получает число с шины данных, интерпретирует его как команду (машинную инструкцию) из своей системы команд и исполняет её.

Если последняя команда не является командой перехода, процессор увеличивает на единицу (в предположении, что длина каждой команды равна единице) число, хранящееся в счётчике команд; в результате там образуется адрес следующей команды.

Данный цикл выполняется неизменно, и именно он называется процессом (откуда и произошло название устройства).

Во время процесса процессор считывает последовательность команд, содержащихся в памяти, и исполняет их. Такая последовательность команд называется программой и представляет алгоритм работы процессора. Очерёдность считывания команд изменяется в случае, если процессор считывает команду перехода, — тогда адрес следующей команды может оказаться другим. Другим примером изменения процесса может служить случай получения команды останова или переключение в режим обработки прерывания.

Команды центрального процессора являются самым нижним уровнем управления компьютером, поэтому выполнение каждой команды неизбежно и безусловно. Не производится никакой проверки на допустимость выполняемых действий, в частности, не проверяется возможная потеря ценных данных. Чтобы компьютер выполнял только допустимые действия, команды должны быть соответствующим образом организованы в виде необходимой программы.

Скорость перехода от одного этапа цикла к другому определяется тактовым генератором. Тактовый генератор вырабатывает импульсы, служащие ритмом для центрального процессора. Частота тактовых импульсов называется тактовой частотой.

Конвейерная архитектура

Основная статья: Вычислительный конвейер

Конвейерная архитектура (англ. pipelining) была введена в центральный процессор с целью повышения быстродействия. Обычно для выполнения каждой команды требуется осуществить некоторое количество однотипных операций, например: выборка команды из ОЗУ, дешифровка команды, адресация операнда в ОЗУ, выборка операнда из ОЗУ, выполнение команды, запись результата в ОЗУ. Каждую из этих операций сопоставляют одной ступени конвейера. Например, конвейер микропроцессора с архитектурой MIPS-I содержит четыре стадии:

получение и декодирование инструкции,
адресация и выборка операнда из ОЗУ,
выполнение арифметических операций,
сохранение результата операции.

После освобождения {\displaystyle k}k-й ступени конвейера она сразу приступает к работе над следующей командой. Если предположить, что каждая ступень конвейера тратит единицу времени на свою работу, то выполнение команды на конвейере длиной в {\displaystyle n}n ступеней займёт {\displaystyle n}n единиц времени, однако в самом оптимистичном случае результат выполнения каждой следующей команды будет получаться через каждую единицу времени.

Действительно, при отсутствии конвейера выполнение команды займёт {\displaystyle n}n единиц времени (так как для выполнения команды по-прежнему необходимо выполнять выборку, дешифровку и т. д.), и для исполнения {\displaystyle m}m команд понадобится {\displaystyle n\cdot m}n\cdot m единиц времени; при использовании конвейера (в самом оптимистичном случае) для выполнения {\displaystyle m}m команд понадобится всего лишь {\displaystyle n+m}n+m единиц времени.

Факторы, снижающие эффективность конвейера:

Простой конвейера, когда некоторые ступени не используются (например, адресация и выборка операнда из ОЗУ не нужны, если команда работает с регистрами).

Ожидание: если следующая команда использует результат предыдущей, то последняя не может начать выполняться до выполнения первой (это преодолевается при использовании внеочередного выполнения команд — out-of-order execution).

Очистка конвейера при попадании в него команды перехода (эту проблему удаётся сгладить, используя предсказание переходов).

Некоторые современные процессоры имеют более 30 ступеней в конвейере, что повышает производительность процессора, но, однако, приводит к увеличению длительности простоя (например, в случае ошибки в предсказании условного перехода). Не существует единого мнения по поводу оптимальной длины конвейера: различные программы могут иметь существенно различные требования.

Суперскалярная архитектура

Основная статья: Суперскалярность

Способность выполнения нескольких машинных инструкций за один такт процессора путём увеличения числа исполнительных устройств. Появление этой технологии привело к существенному увеличению производительности, в то же время существует определённый предел роста числа исполнительных устройств, при превышении которого производительность практически перестает расти, а

исполнительные устройства простаивают. Частичным решением этой проблемы является, например, технология Hyper-threading.

CISC-процессоры

Основная статья: CISC

Complex instruction set computer — вычисления со сложным набором команд. Процессорная архитектура, основанная на усложнённом наборе команд. Типичными представителями CISC являются микропроцессоры семейства x86 (хотя уже много лет эти процессоры являются CISC только по внешней системе команд: в начале процесса исполнения сложные команды разбиваются на более простые микрооперации (МОП), исполняемые RISC-ядром).

RISC-процессоры

Основная статья: RISC

Reduced instruction set computer — вычисления с упрощённым набором команд (в литературе слово reduced нередко ошибочно переводят как «сокращённый»). Архитектура процессоров, построенная на основе упрощённого набора команд, характеризуется наличием команд фиксированной длины, большого количества регистров, операций типа регистр-регистр, а также отсутствием косвенной адресации. Концепция RISC разработана Джоном Коком из IBM Research, название придумано Дэвидом Паттерсоном (David Patterson).

Упрощение набора команд призвано сократить конвейер, что позволяет избежать задержек на операциях условных и безусловных переходов. Однородный набор регистров упрощает работу компилятора при оптимизации исполняемого программного кода. Кроме того, RISC-процессоры отличаются меньшим энергопотреблением и тепловыделением.

Среди первых реализаций этой архитектуры были процессоры MIPS, PowerPC, SPARC, Alpha, PA-RISC. В мобильных устройствах широко используются ARM-процессоры.

MISC-процессоры

Основная статья: MISC

Minimum instruction set computer — вычисления с минимальным набором команд. Дальнейшее развитие идей команды Чака Мура, который полагает, что принцип простоты, изначальный для RISC-процессоров, слишком быстро отошёл на задний план. В пылу борьбы за максимальное быстродействие, RISC догнал и обогнал многие CISC-процессоры по сложности. Архитектура MISC строится на стековой вычислительной модели с ограниченным числом команд (примерно 20—30 команд).

VLIW-процессоры

Основная статья: VLIW

Very long instruction word — сверхдлинное командное слово. Архитектура процессоров с явно выраженным параллелизмом вычислений, заложенным в систему команд процессора. Являются основой для архитектуры EPIC. Ключевым отличием от суперскалярных CISC-процессоров является то, что для них загрузкой исполнительных устройств занимается часть процессора (планировщик), на что отводится достаточно малое время, в то время как загрузкой вычислительных устройств для VLIW-процессора занимается компилятор, на что отводится существенно больше времени (качество загрузки и, соответственно, производительность теоретически должны быть выше).

Например, Intel Itanium, Transmeta Crusoe, Efficeon и Эльбрус.

Многоядерные процессоры

Основная статья: Многоядерный процессор

Содержат несколько процессорных ядер в одном корпусе (на одном или нескольких кристаллах).

Процессоры, предназначенные для работы одной копии операционной системы на нескольких ядрах, представляют собой высокоинтегрированную реализацию мультипроцессорности.

Контрольные вопросы:

1. Какие предпосылки превели к созданию микропроцессоров?
2. Что такое «Архитектура фон Неймана»?
3. Поясните принцип работы конвейерной архитектуры;
4. Поясните принцип работы суперскалярной архитектуры
5. В чём различие между CISC, RISC, VLIW-процессорами?

Тема 2.6. Запоминающие устройства ЭВМ

Основные принципы построения оперативной памяти. Иерархическая организация памяти. Стратегии управления памятью. Принципы работы кэш-памяти.

Иерархия памяти ЭВМ.Память ЭВМ должна иметь большую информационную емкость V, малое время обращения t (высокое быстродействие), высокую надежность и низкую стоимость. Но с увеличением емкости снижается быстродействие и растет стоимость. Деление памяти на ОЗУ и ВЗУ не снимает это противоречие полностью, так как различие в быстродействии процессора, ОЗУ и ВЗУ очень велико. Поэтому обмен информацией производится через дополнительные буферные устройства, то есть память ЭВМ имеет иерархическую многоуровневую структуру. Чем больше быстродействие ЗУ, тем выше стоимость хранения 1 байта, тем меньшую емкость имеет ЗУ.

Иерархия памяти ЭВМ:

· регистры микропроцессорной памяти, а также кэш-память первого и второго уровня t=10-9-10-6 t=10-9-10-6 с, V=102-104 бит);

· внутренняя память ПЗУ, ОЗУ t=10-6-10-3 t=10-6-10-3 с, V=10-4-107 бит);

· внешняя память (t=10-3-1 с, V=107-109 бит);

· массовая или архивная память (t=1-10 с, V=109-1010 V=109-1010 бит).

Эта система запоминающих устройств работает как единое ЗУ с большой емкостью (за счет внешних ЗУ) и высоким быстродействием (за счет внутренних ЗУ).

Микропроцессорная память -- высокоскоростная память небольшой емкости, входящая в МП и используемая АЛУ для хранения операндов и промежуточных результатов вычислений. *КЭШ-память* -- это буферная, не доступная для пользователя память, автоматически используемая компьютером для ускорения операций с информацией, хранящейся в медленно действующих запоминающих устройствах. Для ускорения операций с основной памятью организуется регистровая КЭШ-память внутри микропроцессора (КЭШ-память первого уровня) или вне микропроцессора на материнской плате (КЭШ-память второго уровня); для ускорения операций с дисковой памятью организуется КЭШ-память на ячейках электронной памяти.

Внутренняя память состоит из ПЗУ (ROM -- Read Only Memory) и ОЗУ (RAM -- Random Access Memory -- память с произвольным доступом). ПЗУ состоит из установленных на материнской плате микросхем и используется для хранения неизменяемой информации: загрузочных программ операционной системы (ОС), программ тестирования устройств компьютера и некоторых драйверов базовой системы ввода-вывода (BIOS -- Base Input-Output System) и др. Из ПЗУ можно только считывать информацию, емкость ПЗУ -- сотни Кбайт. Это энергонезависимая память, -- при отключении ЭВМ информация сохраняется.

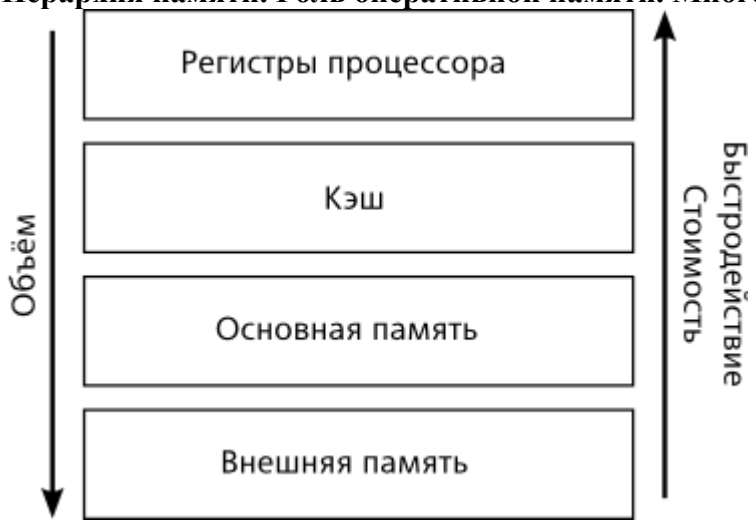
Внешняя память относится к внешним устройствам ЭВМ и используется для долговременного хранения любой информации, которая может потребоваться. В ВЗУ хранится программное обеспечение ЭВМ. Внешняя память: НЖМД и ЖМД, НГМД и ГМД (магнитный диск), стример (НМЛ -- накопитель на магнитной ленте), оптические накопители для CD-ROM и DVD-дисков.

Информационная структура внешней памяти -- файловая. Наименьшей именуемой единицей является файл, -- наименованная совокупность однородных данных. Информация в файле состоит из битов и байтов, но они не имеют адресов, так как носитель (магнитный диск) не дискретный.

Регистр и регистровый вид памяти

Для ускорения выполнения машинных команд в процессоре предусмотрен еще один вид памяти — регистровый. Регистр — это устройство для кратковременного хранения информации в процессе ее обработки. Еще раз обращаем внимание на то, что регистры входят в состав процессора, а не образуют отдельное устройство. Регистр может хранить один или несколько символов, число, код машинной команды, какой-нибудь адрес оперативной памяти. Регистры представляют собой самый быстродействующий вид памяти, но процессор имеет всего один-два десятка регистров

Иерархия памяти. Роль оперативной памяти. Многозадачность.



Кэш память = сверхоперативная память

Известно, что память ЭВМ предназначена для хранения программ и данных, причем эффективность работы ЭВМ во многом определяется характеристиками ее памяти. Во все времена к памяти предъявлялись три основных требования: большой объем, высокое *быстродействие* и низкая (умеренная) *стоимость.*

Все перечисленные выше требования к памяти являются взаимно-противоречивыми, поэтому пока невозможно реализовать один тип ЗУ, отвечающий всем названным требованиям. В современных ЭВМ организуют комплекс разнотипных ЗУ,

взаимодействующих между собой и обеспечивающих приемлемые характеристики памяти ЭВМ для каждого конкретного применения.

I. Регистровая память – местная память процессора.

Отличительные черты:

1)Самая быстрая

2)Самое маленькое время доступа

3)Большая стоимость

4)Маленькая емкость

“регистры”(registers) – модули, построенные на триггерах.

Триггер - это устройство последовательного типа с двумя устойчивыми состояниями равновесия, предназначенное для записи и хранения информации. Под действием входных сигналов триггер может переключаться из одного устойчивого состояния в другое. При этом напряжение на его выходе скачкообразно изменяется. (вдруг спросит)

В регистрах хранятся операнды(числа(двоичные), которые процессор обрабатывает, при выполнении текущей команды)

или результаты их команд, которые выполняет процессор в текущем такте.

II. Сверхоперативная память

СОЗУ обладает максимальным быстродействием (равным процессорному), небольшим объемом (10^5 — 10^7 байтов) и располагается, как правило, на кристалле процессорной БИС. Для обращения к СОЗУ не требуются магистральные (машинные) циклы. В СОЗУ размещаются наиболее часто используемые на данном участке программы данные, а иногда — и фрагменты программы.

В вычислительных системах используют многоуровневый кэш

1. Кэш процессора 1го уровня (L1) — время доступа порядка нескольких тактов, размером в десятки килобайт

2. Кэш процессора 2го уровня (L2) — большее время доступа (от 2 до 10 раз медленнее L1), около полумегабайта или более

3. Кэш процессора 3го уровня (L3) — время доступа около сотни тактов, размером в несколько мегабайт (в массовых процессорах используется с недавнего времени)

В настольных системах обычно используется двухуровневый кэш,

в серверных - трехуровневый.

Кэш служит высокоскоростным буфером между ЦП и относительно медленной основной памятью.

III. Оперативная память

Служит для размещения туда программ целиком и сегментных данных, которые она использует. Связь между процессором и ОЗУ осуществляется по системному или специализированному интерфейсу и требует для своего осуществления машинных циклов

ОЗУ системы — время доступа от сотен до, возможно, тысячи тактов, но огромные размеры в несколько гигабайт, вплоть до десятков. Время доступа к ОЗУ может варьироваться для разных его частей в случае комплексов класса NUMA (с неоднородным доступом в память)

Организационные методы (сплошная и сегментированная) распределения памяти позволяют организовать вычислительную систему, в которой рабочее адресное пространство программы превышает размер фактически имеющейся в системе

оперативной памяти, при этом недостаток оперативной памяти заполняется за счет внешней более медленной или более дешевой памяти (винчестер, флэш-память и т.п.) Такую концепцию называют ***виртуальной памятью***.

Кратко:

Оперативная память

n В оперативной (внутренней) памяти компьютера хранятся данные и программы.

n Оперативная память представляет собой последовательность пронумерованных, начиная с нуля, ячеек.

n В каждой ячейке оперативной памяти может храниться двоичный код.

IV. Внешняя память

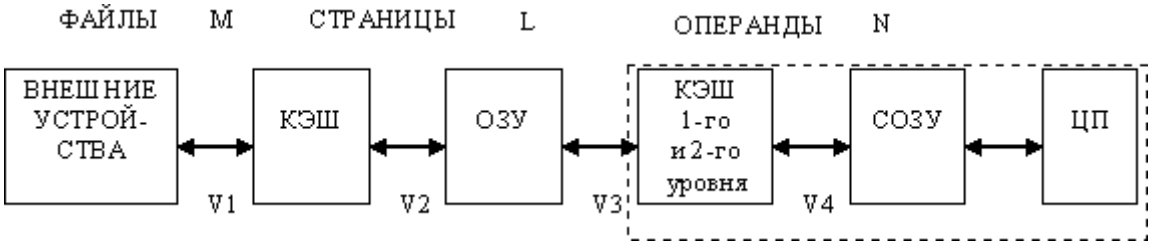
Информация, находящаяся в ВЗУ, не может быть непосредственно использована процессором. Для использования программ и данных, расположенных в ВЗУ, их необходимо предварительно переписать в ОЗУ. Процесс обмена информацией между ВЗУ и ОЗУ осуществляется средствами специального канала или (реже) — непосредственно под управлением процессора. Объем ВЗУ практически неограничен, а быстродействие на 3 — 6 порядков ниже процессорного

Кроме реализации системы виртуальной памяти внешние ЗУ используются для долговременного хранения программ и данных в виде файлов. Большинство операционных систем (ОС) поддерживают многозадачность. Они последовательно переключают задачи одну на другую.В каждый момент времени процессор выполняет только одну задачу. В многопоточных процессорах одновременно могут выполняться несколько задач. ОС планирует какая из задач будет выполняться следующей, выбирает эту задачу и переключает контексты задач. Методы переключения зависят от стратегии, выбранной ОС.

Энергонезависимая память=внешняя память.

Вопрос 17 Иерархия памяти. Роль оперативной памяти. Чем может быть обусловлено применение виртуальной памяти. -

1)



2)

Роль оперативной памяти - ОЗУ предназначено для хранения переменной информации; оно допускает изменение своего содержимого в ходе выполнения вычислительного процесса. Таким образом, процессор берёт из ОЗУ код команды и, после обработки каких-либо данных, результат обратно помещается в ОЗУ.

Причем возможно размещение в ОЗУ новых данных на месте прежних, которые при этом перестают существовать. В ячейках происходит стирание старой информации и запись туда новой. Из этого видно, что ОЗУ является очень гибкой структурой и обладает возможностью перезаписывать информацию в свои ячейки

неограниченное количество раз по ходу выполнения программы. Поэтому ОЗУ играет значительную роль в ходе формирования виртуальных адресов.

3)

Виртуальная память представляет собой совокупность всех ячеек памяти оперативной и внешней, имеющих сквозную нумерацию от нуля до предельного значения адреса.

виртуальная память позволяет модифицировать ресурсы памяти, ***сделать объём оперативной памяти намного больше,*** *для того чтобы пользователь, поместив туда как можно больше программ, реально сэкономил время и повысил эффективность своего труда.*

Контрольные вопросы:

1. Какие виды памяти применяются в ЭВМ?
2. Что такое ОЗУ?
3. Что такое ПЗУ?
4. Что такое КЕШ память?
5. Что такое ВЗУ?

Тема 3.1 Перефирийные устройства вычислительной техники
Тема 3.2. Нестандартные перефирийные устройства
Виды интерфейсов. Внутренние и внешние интерфейсы.

Интерфейс – совокупность средств сопряжения и связи, обеспечивающая эффективное взаимодействие систем или частей.

В интерфейсе обычно предусмотрено сопряжение на двух уровнях:

– механическом (провода, элементы связи, типы соединений, разъемы, номера контактов ит.д.)

– логическом (сигналы, их длительность, полярности, частоты и амплитуда, протоколы взаимодействия).

Все интерфейсы ЭВМ можно разделить на внутренние и внешние:

– внутренние – система связи и сопряжения узлов и блоков ПК между собой;

– внешние – обеспечивают связь ПК с внешними (периферийными) устройствами и другими компьютерами.

Существуют два варианта организации внутреннего интерфейса:

– многосвязный интерфейс: каждый блок ПК соединен с прочими блоками своими локальными проводами; многосвязный интерфейс иногда применяется в качестве периферийного интерфейса (для связи с внешними устройствами ПК);

– односвязный интерфейс: все блоки ПК связаны друг с другом через общую, или системную шину.

В подавляющем большинстве современных ПК в качестве системного интерфейса используется системная шина (совокупность линий связи, по которым информация передается одновременно). Под системной шиной обычно понимается шина между процессором и подсистемой памяти. Шины характеризуются разрядностью (количество линий связи в шине, т.е. число битов, которое может быть передано по шине

одновременно) и частотой (частота, с которой передаются последовательные биты информации по линиям связи).

Если интерфейс является общепринятым, например, утвержденным на уровне международных соглашений, то он называется **стандартным**.

Каждое устройство, используемое в компьютерной системе, должно каким-то образом подключаться к этой системе. Точка подключения называется *интерфейсом*. Устройства хранения не являются исключением — у них тоже есть интерфейсы. Знать об интерфейсах важно по двум основным причинам:

- существует множество разных (зачастую несовместимых) интерфейсов
- разные интерфейсы могут отличаться ценой и производительностью.

К сожалению, не существует одного универсального интерфейса устройств или какого-то одного интерфейса устройств хранения. Поэтому системные администраторы должны знать, какие интерфейсы поддерживаются компьютерами в их организации. В противном случае, планируя обновление компьютеров, есть реальный риск приобрести неподходящее оборудование.

Разные интерфейсы имеют разную производительность, поэтому одни интерфейсы могут больше подходить для определённых окружений, чем другие. Например, интерфейсы, способные поддерживать высокоскоростные устройства, лучше подходят для серверов, тогда как для обычных рабочих станций будет достаточно более медленных интерфейсов. Такая разница в производительности также приводит к разнице в цене, ведь вы всегда получаете то, за что платите. Высокопроизводительные компьютеры стоят недешёво.

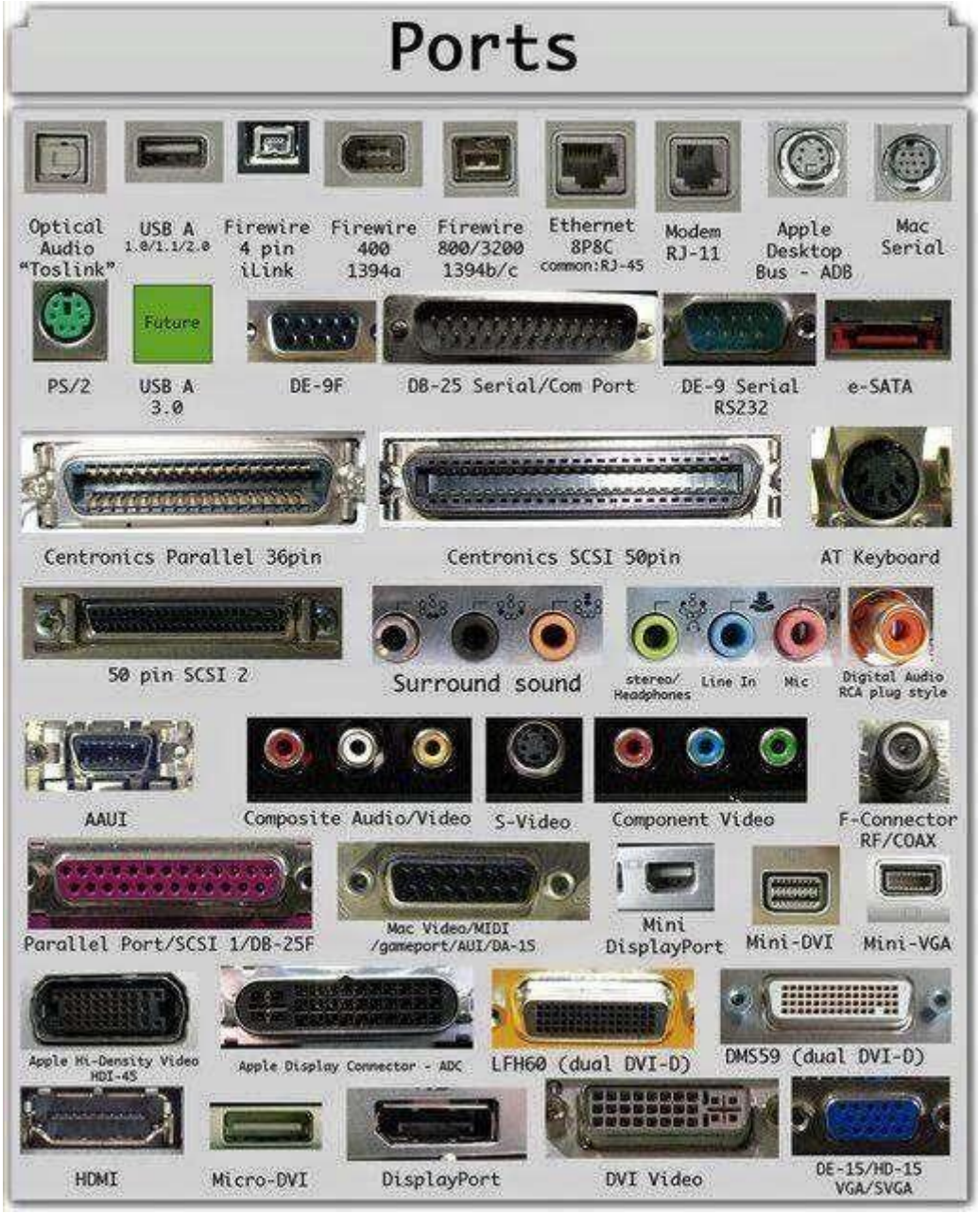
Основные интерфейсы материнской платы

Внутренние интерфейсы предназначены для подключения компонентов, расположенных внутри системного блока. Все контроллеры и шины внутренних интерфейсов размещаются на системной плате.

К важнейшим внутренним интерфейсам относятся:

– системная шина с разъемом процессора;
– шина памяти с разъемами модулей памяти;
– шина и слот видеокарты;
– шины и слоты плат расширения;
– шины и порты накопителей (жесткий диск, дисковод, DVD);
– шина и разъемы электропитания;
– линии и порты интерфейса управления питанием;
– порты и панели индикации;

Внешние порты — представляют собой интерфейс или точку взаимодействия между компьютером и другим периферийным устройством. Основное предназначение таких портов — обеспечение места подключения кабеля устройства для передачи и получения данных от центрального процессора. В этой статье мы рассмотрим какими бывают внешние порты компьютера, а также рассмотрим основные порты и их предназначение.



Внешние разъемы компьютера еще называют коммуникационными портами, так как они отвечают за связь между компьютером и периферийными устройствами. Как правило, основа порта размещается на материнской плате.

Все внешние интерфейсы компьютера делятся на два вида, в зависимости от их вида и протокола, используемого для связи с центральным процессором. Это последовательные и параллельные порты.

Последовательный (serial) порт — это интерфейс, через который устройства могут быть подключены с использованием последовательного протокола. Этот протокол позволяет передавать один бит данных за один раз по одной линии. Наиболее распространенный тип последовательного порта — D-sub, который позволяет передавать сигналы RS-232.

Параллельный порт работает немного по-другому, обмен данными между периферийным устройством осуществляется параллельно с помощью нескольких линий связи. Большинство портов для современных устройств — параллельны. Дальше мы рассмотрим более подробно каждый тип внешних интерфейсов компьютера, а также их предназначение.

Контрольные вопросы:

1. Назовите функции интерфейсов.
2. Каик характеристики имеют современное интерфейсы?
3. Какие внешние интерфейсы не используются в настоящее время?

Основные компоненты программного обеспечения компьютерных систем.

POST (англ. Power-On Self-Test) — процедура проверки работоспособности аппаратного обеспечения компьютера, выполняемая при его включении. После прохождения этой процедуры на экран выводиться сообщение о результатах тестирования и системный динамик издает звуковой сигнал. В случае успешного прохождения POST системный динамик издаёт один короткий звуковой сигнал, в случае сбоя — различные последовательности звуковых сигналов. Пример окна с результатами положительного тестирования процедурой POST Рис.1 Окно процедуры POST Полный регламент работы POST: Проверка регистров процессора; Проверка контрольной суммы ПЗУ; Проверка системного таймера и порта звуковой сигнализации (для IBM PC — ИМС i8255 или аналог); Тест контроллера прямого доступа к памяти; Тест регенератора оперативной памяти; Тест нижней области ОЗУ для проецирования резидентных программ в BIOS; Загрузка резидентных программ; Тест стандартного графического адаптера (VGA); Тест оперативной памяти; Тест основных устройств ввода (НЕ манипуляторов); Тест CMOS; Тест основных портов LPT/COM; Тест накопителей на гибких магнитных дисках (НГМД); Тест накопителей на жёстких магнитных дисках (НЖМД); Самодиагностика функциональных подсистем BIOS; Передача управления загрузчику. Выбор между прохождением полного или сокращенного набора тестов при включении компьютера можно задать в программе настройки базовой системы ввода-вывода, Setup BIOS.

В современных компьютерах процесс загрузки большей частью автоматизирован, однако это не означает, что он не заслуживает того, чтобы с ним познакомиться.

Основные определения и этапы загрузки операционной системы

Включение компьютера, POST, BootMonitor

Начальный этап загрузки операционной системы после включения компьютера начинается в BIOS (Basic Input/Output System — базовая система ввода-вывода). В настройках BIOS мы указываем загрузочное устройство, или ряд загрузочных устройств в порядке их приоритета. Возможны различные варианты загрузки и их комбинации: с жесткого диска, CD/DVD – диска, USB-flash и другие.

Сразу после прохождения POST (Power-On Self-Test — самотестирование после включения) BIOS компьютера начнет поочередно перебирать указанные загрузочные устройства до тех пор, пока на одном из них не найдет подходящую специальную запись, в которой содержится информация о дальнейших действиях.

Загрузчик 1-го уровня. Master Boot Record

Master Boot Record — главная загрузочная запись, расположена в первых физических секторах загрузочных устройств хранения. Она содержит таблицу разделов (Partition Table) и исполняемый код.

Главной задачей программы, записанной в MBR, является поиск активного системного раздела диска и передача управления его загрузочному сектору. Таким образом, эту стадию можно назвать подготовительной, в силу того, что непосредственно загрузки самой ОС еще не происходит.

Системным принято называть раздел диска (устройства хранения) на котором расположены файлы операционной системы, отвечающие за процесс загрузки ОС (сама операционная система может размещаться в другом разделе). В принципе, системных разделов может быть несколько, поэтому один из них отмечается как активный. Именно его ищет программа, загруженная с MBR.

Загрузчик 2-го уровня. Partition Boot Sector

Следующим этапом загрузки компьютера является передача управления исполняемому коду, записанному в PBS (Partition Boot Sector — загрузочный сектор активного раздела). PBS расположен в первом секторе (секторах) соответствующего раздела диска. В коде PBS прописано имя файла загрузчика операционной системы, которому и передается управление на этом этапе.

Начальный этап загрузки операционной системы. Менеджер загрузки ОС

Первоначально в Linux загрузчиком являлся LILO (Linux Loader). В силу имевшихся в нем недостатков, главным из которых была неспособность понимать используемые в Linux файловые системы, позднее начал использоваться загрузчик GRUB (GRand Unified Bootloader) в котором недостатки LILO были исправлены.

Если речь идет о версиях Windows до Vista, например, Windows XP, то будет загружен Ntldr. Он, в свою очередь, считывает информацию из текстового файла Boot.ini, в котором записана информация об установленных операционных системах.

Загрузка ядра операционной системы

Завершающим этапом загрузки операционной системы является загрузка ядра ОС и передача ему управления.

Несколько лет назад в моей практике был такой забавный случай.

Меня попросили помочь одному человеку купить и привести в порядок компьютер для дома. Товарищ этот на тот момент времени только-только вышел на пенсию, а до этого работал в должности начальника и компьютер видел, по большей части, издалека, на столе у своей секретарши. После выхода на пенсию у него появилось время и желание компьютер освоить. Ну что, по-моему, замечательно. Компьютер мы ему подобрали недорогой, но добротный и с хорошим, по тем временам, монитором. Я поставил и настроил кое-какие программы, показал как ими пользоваться. Господин обзавелся книжкой из серии «*Что-то там для чайников*» и мы расстались абсолютно довольные друг другом.

Примерно через неделю мой подопечный позвонил мне по телефону и чуть не плача сообщил, что все пропало и компьютер больше не работает. Благо, запираться он не

стал и честно рассказал, как было дело. А дело было так. Товарищ полез исследовать содержимое своего компьютера и нашел в корне диска C:\ несколько подозрительных и «ненужных», на его взгляд, файлов. Он решил освободить место и файлики эти попросту удалил. Думаю, что все уже догадались, что это были boot.ini, ntldr, ntdetect и др. Поправить ситуацию сложности не представляло, разве что пришлось ехать к нему еще раз.

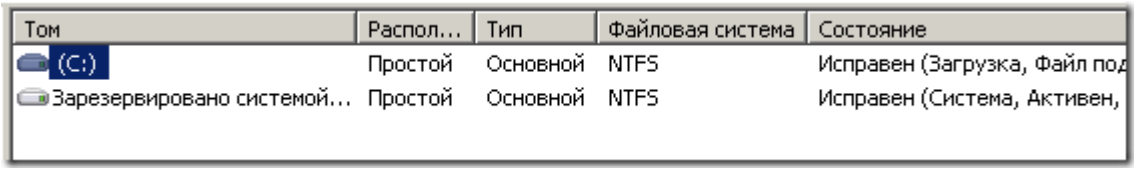
К слову сказать, для того, чтобы получить проблемы с загрузкой Windows XP, можно было и ничего не стирать, а слегка «подправить», например, boot.ini.

Видимо Microsoft как то узнала об этом случае  и в следующей версии своей ОС решила припрятать файлы загрузчика получше.

Особенности загрузки операционных систем Windows Vista / 7 / 8

В Windows Vista или Windows 7 / 8 диспетчер загрузки называется Bootmgr. Он читает список установленных операционных систем не из простого текстового файла, коим является boot.ini, а из системного хранилища BCD (Boot Configuration Data), доступ к которому осуществляется посредством специальной утилиты bcdedit.exe .

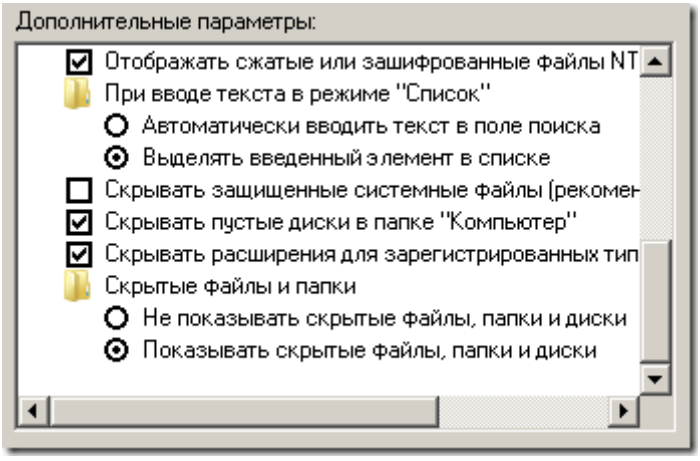
Но это еще не все. Откроем “Управление компьютером” –> “Управление дисками” >



Процесс стандартной инсталляции операционной системы Windows Vista / 7 / 8 создает в начале диска дополнительный раздел «Зарезервировано системой». Ему не присваивается буква диска, благодаря чему, если не сделать дополнительных шагов, и сам раздел и его содержимое будут скрыты от пользователя. Этот основной раздел имеет статус «Системный» и «Активный» и, следовательно, удалить его тоже нельзя.

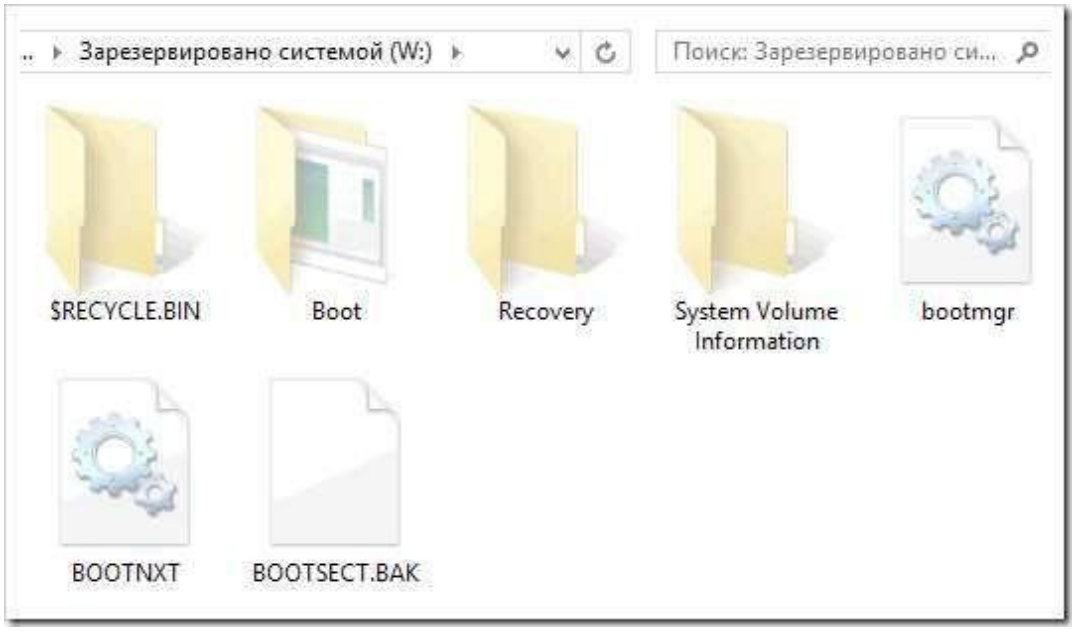
В Windows 7 размер такого раздела составляет 100 МБ, из которых занято около 30 МБ, в Windows 8 – уже 350 МБ, причем занято из них 105 МБ. Такая разница объясняется тем, что в Windows 8 на этот раздел перенесена папка “Recovery”.

Посмотрим содержимое раздела «Зарезервировано системой». Для этого назначим ему букву диска, например, «W» в “Управление дисками”. Дополнительно откроем «Панель управления» –> «Параметры папок» и на вкладке «Вид» снимем галку с «Скрывать защищенные системные файлы» и поставим отметку на «Показывать скрытые файлы, папки и диски».

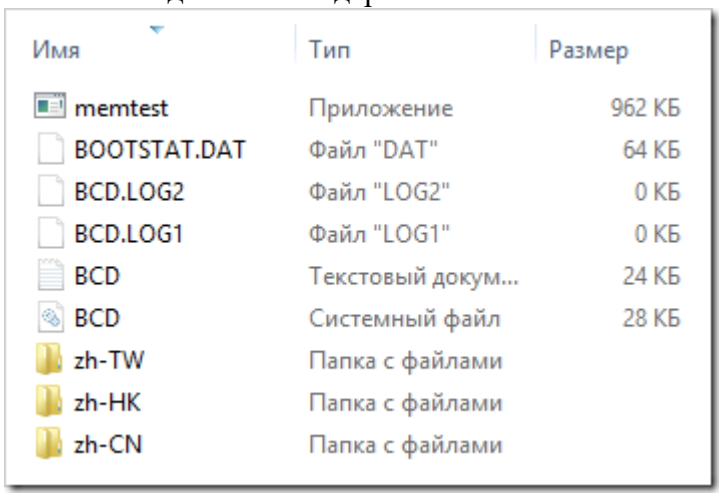


Если в системе установлен файловый менеджер FAR Manager, то для просмотра можно воспользоваться им и не заниматься настройкой видимости скрытых файлов.

Так выглядит содержимое раздела «Зарезервировано системой» в Windows 8 >



Как и ожидалось, в корне раздела мы видим загрузчик Bootmgr.
А так выглядит часть содержимого папки «Boot» >



В папке мы находим базу хранилища данных конфигурации загрузки BCD и сопутствующие ей папки с языковыми файлами и файлами шрифтов.

Для полноты картины осталось сказать о том, какой раздел называется загрузочным. Ответ уже показан на картинке с томами Windows 7. Очевидно, что это раздел, на котором находятся все основные файлы операционной системы.

Ну что же, запомнить названия разделов диска очень легко по принципу «всё наоборот» — на системном нет системы (операционной), но расположен загрузчик третьего уровня, на загрузочном как раз отсутствует загрузчик, но находится сама система . Естественно, эта «запоминалка» работает только тогда, когда есть несколько разделов. Если раздел один, то он может быть сразу системным, активным и загрузочным.

Главной задачей загрузчика 3-го уровня, в роли которого выступает, в зависимости от типа ОС, Bootmgr, Ntldr или GRUB, является чтение с загрузочного диска и загрузка ядра операционной системы. Кроме того, в случае множественной загрузки, когда на компьютере установлено несколько операционных систем, загрузчик 3-го уровня позволяет выбирать нужную при каждом запуске компьютера.

Классической ошибкой, которой Microsoft посвятила отдельную статью, является установка Windows XP после Windows Vista / 7 /8. Установщик Windows XP помечает свой раздел как активный, после чего, во время загрузки, MBR передает управление PBS этого раздела а он, в свою очередь, — Ntldr. Загрузчик Windows XP ничего не знает о более поздних версиях операционных систем Windows и их загрузка становится

невозможной. Лечится достаточно легко, но неискушенного пользователя такая ситуация может поставить в тупик.

Виды ОС:

* **Многопользовательская система**, система с коллективным доступом, система коллективного доступа (multiuser system, multiaccess system) - вычислительная система или ее часть (например операционная система), позволяющая нескольким пользователям одновременно иметь доступ к одной ЭВМ со своего терминала (локального или удаленного). Многопользовательский характер работы достигается благодаря режиму разделения времени, который заключается в очень быстром переключении ЭВМ между разными терминалами и программами и соответственно быстрой отработке команд каждого пользователя. При этом последний не замечает задержек времени, связанных с обслуживанием других пользователей. Примерами разработок указанного вида могут служить помимо Windows операционные системы: NetWare, созданная и развиваемая фирмой Novell (США) для локальных информационных вычислительных систем; Unix фирмы AT&T's Bell Laboratories (США); REAL/32 и др.
* **Однопользовательская система** (one user system) - операционная система, не обладающая свойствами многопользовательской. Примерами однопользовательских ОС являются MS DOS фирмы Microsoft (США) и ОС/2, созданная совместно Microsoft и IBM.
* Сетевая операционная система, СОС (NOS, Network Operating System) - операционная система, предназначенная для обеспечения работы вычислительной сети. Примерами сетевых операционных систем являются Windows NT, Windows 2000, Novel Netware, Unix, Linux и др.

Типы ОС:

* графические (с наличием графического пользовательского интерфейса - GUI) - текстовые (только командная строка);
* бесплатные - платные;
* открытые (с возможностью редактировать исходный код) - закрытые (без возможности редактировать исходный код);
* клиентские - серверные;
* высокая стабильность (устойчивость к сбоям аппаратной части)- низкая стабильность;
* простая в администрировании (для рядового пользователя) - сложная, для системных администраторов;
* 16-разрядная - 32-разрядная - 64-разрядная (в далеком прошлом были еще и 8- разрядные);
* с высоким уровнем безопасности данных - с низким уровнем безопасности;

Понятие операционной системы

Существуют две группы определений ОС: «совокупность программ, управляющих оборудованием» и «совокупность программ, управляющих другими программами». Обе они имеют свой точный технический смысл, который, однако, становится ясен только при более детальном рассмотрении вопроса о том, зачем вообще нужны операционные системы.

Есть приложения вычислительной техники, для которых ОС излишни. Например, встроенные микрокомпьютеры содержатся сегодня во многих бытовых приборах, автомобилях (иногда по десятку в каждом), сотовых телефонах и т. п. Зачастую такой компьютер постоянно исполняет лишь одну программу, запускающуюся по включении. И простые игровые приставки — также представляющие собой специализированные микрокомпьютеры — могут обходиться без ОС, запуская при включении программу, записанную на вставленном в устройство «картридже» или компакт-диске. (Многие встроенные компьютеры и даже некоторые игровые приставки на самом деле работают

под управлением своих ОС).

Операционные системы, в свою очередь, нужны, если:
* вычислительная система используется для различных задач, причём программы, исполняющие эти задачи, нуждаются в сохранении данных и обмене ими. Из этого следует необходимость универсального механизма сохранения данных; в подавляющем большинстве случаев ОС отвечает на неё реализацией файловой системы. Современные ОС, кроме того, предоставляют возможность непосредственно «связать» вывод одной программы с вводом другой, минуя относительно медленные дисковые операции;
* различные программы нуждаются в выполнении одних и тех же рутинных действий. Напр., простой ввод символа с клавиатуры и отображение его на экране может потребовать исполнения сотен машинных команд, а дисковая операция — тысяч. Чтобы не программировать их каждый раз заново, ОС предоставляют системные библиотеки часто используемых подпрограмм (функций);
* между программами и пользователями системы необходимо распределять полномочия, чтобы пользователи могли защищать свои данные от чужого взора, а возможная ошибка в программе не вызывала тотальных неприятностей;
* необходима возможность имитации «одновременного» исполнения нескольких программ на одном компьютере (даже содержащем лишь один процессор), осуществляемой с помощью приёма, известного как «разделение времени». При этом специальный компонент, называемый планировщиком, «нарезает» процессорное время на короткие отрезки и предоставляет их поочередно различным исполняющимся программам (процессам);
* наконец, оператор должен иметь возможность, так или иначе, управлять процессами выполнения отдельных программ. Для этого служат операционные среды, одна из которых — оболочка и набор стандартных утилит — является частью ОС (прочие, такие, как графическая операционная среда, образуют независимые от ОС прикладные платформы). Таким образом, современные универсальные ОС можно охарактеризовать прежде всего как
* использующие файловые системы (с универсальным механизмом доступа к данным),
* многопользовательские (с разделением полномочий),
* многозадачные (с разделением времени).

Многозадачность и распределение полномочий требуют определённой иерархии привилегий компонентов самой ОС. В составе ОС различают три группы компонентов:
* ядро, содержащее планировщик; драйверы устройств, непосредственно управляющие оборудованием; сетевую подсистему, файловую систему;
* системные библиотеки и
* оболочку с утилитами.

Большинство программ, как системных (входящих в ОС), так и прикладных, исполняются в непривилегированном («пользовательском») режиме работы процессора и получают доступ к оборудованию (и, при необходимости, к другим ядерным ресурсам, а также ресурсам иных программ) только посредством системных вызовов. Ядро исполняется в привилегированном режиме: именно в этом смысле говорят, что ОС (точнее, её ядро) управляет оборудованием.

Текущая редакция стандарта на ОС содержит определения около тысячи системных вызовов и других библиотечных подпрограмм (часть из которых должна реализоваться только в определённых классах систем; напр., в системах «реального времени») и около 200 команд оболочки и утилит ОС. Стандарт определяет лишь функции вызовов и команд, и не содержит указаний относительно способов их реализации.

Стандарт, кроме этого, определяет способ адресации файлов в системе, локализацию (установки, касающиеся национально-специфических моментов, таких, как язык сообщений или формат даты и времени), совместимый набор символов, синтаксис регулярных выражений, структуру каталогов в файловой системе, формат командной строки и некоторые другие аспекты поведения ОС.

В определении состава ОС значение имеет критерий операциональной целостности (замкнутости): система должна позволять полноценно использовать (включая модификацию) свои компоненты. Поэтому в полный состав ОС включается и набор инструментальных средств (от текстовых редакторов до компиляторов, отладчиков и компоновщиков). Операциональной замкнутостью обладают системы, удовлетворяющие «разработческому» профилю в терминах стандарта.