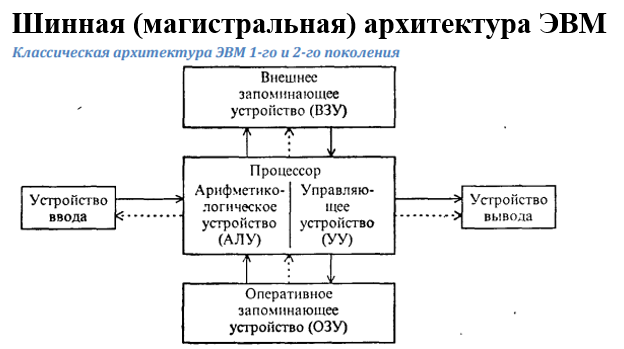
**Вопросы к зачету (2022, весна)**

**по дисциплине “Архитектура вычислительных систем»**

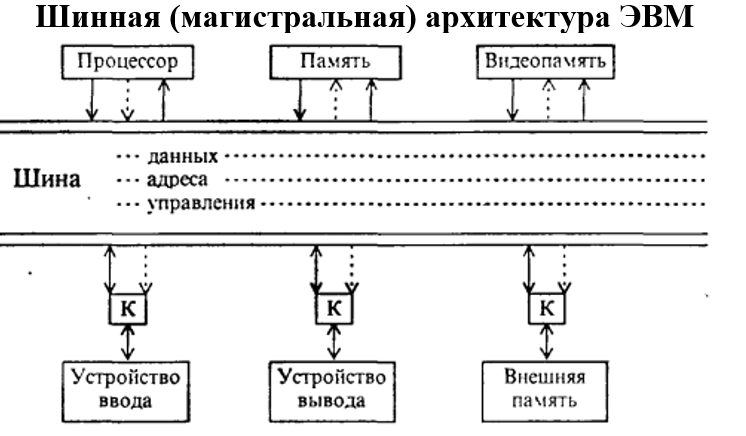
# Архитектура вычислительной машины

**Архитектура вычислительной машины** (англ. сomputer architecture) – концептуальная структура вычислительной машины, определяющая проведение обработки информации и включающая методы преобразования информации в данные и принципы взаимодействия технических средств и программного обеспечения.



**Шинная (магистральная) архитектура** ЭВМ. Наличие интеллектуальных контроллеров внешних устройств стало важной отличительной чертой машин третьего и четвертого поколений.

Контроллер можно рассматривать как специализированный процессор, управляющий работой внешнего устройства. Такой процессор имеет собственную систему команд.

****Результаты выполнения каждой операции заносятся во внутренние регистры памяти контроллера и могут быть в дальнейшем прочитаны центральным процессором.

Центральный процессор при необходимости произвести обмен выдает задание на его осуществление контроллеру. Дальнейший обмен информацией может протекать под руководством контроллера без участия центрального процессора. Последний получает возможность «заниматься своим делом», т.е. выполнять программу дальше (если по данной задаче до завершения обмена ничего сделать нельзя, то можно в это время решать другую).

**Шина** **состоит** из трех частей:

* Шина **данных**, по которой передается информация;
* Шина **адреса**, определяющая, куда передаются данные;
* Шина **управления**, регулирующая процесс обмена информацией.

# Представление чисел в компьютере

**Ячейка** – это часть памяти компьютера, вмещающая в себя информацию, доступную для обработки **отдельной командой** процессора.

Содержимое ячейки памяти называется **машинным словом**. Ячейка памяти разделяется на **разряды**, в каждом из которых хранится разряд числа.

Для представления чисел в памяти компьютера используются два формата: формат с **фиксированной точкой** и формат **с плавающей точкой**. В формате с фиксированной точкой представляются только целые числа, в формате с плавающей точкой – вещественные числа (целые и дробные). Целые числа занимают 1, 2 или 4 байта(char, short int,int), а вещественные 4,8(float,double).

В ЭВМ в целях упрощения выполнения арифметических операций применяют **специальные коды** для представления **целых чисел**

* прямой код числа
* обратный код числа
* дополнительный код числа

**Прямой код** – это представление числа в двоичной системе счисления, при этом первый разряд отводится под знак числа.

**Обратный код** для положительного числа в двоичной системе счисления совпадает с прямым кодом. Для отрицательного числа все цифры числа заменяются на противоположные.

**Дополнительный код** используют в основном для представления в компьютере отрицательных чисел.

* Переводим в двоичную
* Получаем обратный код (меняем 0 и 1 на противоположное)
* Последнее число меняем на 1

# Сложение, вычитание и умножение двоичных чисел

|  |  |  |
| --- | --- | --- |
| Сложение | Вычитание | Умножение |
| 0+0=0 | 0-0=0 | 0 ∙0=0 |
| 0+1=1 | 1-0=1 | 0 ∙1=0 |
| 1+0=1 | 1-1=0 | 1 ∙0=0 |
| 1+1=10 | 10-1=1 | 1 ∙1=1 |

# Формат вещественных чисел. Нормализованные числа. Стандарт IEEE 754 чисел с плавающей точкой. Основы #HEX.

IEEE 754 чисел с плавающей точкой. Основы #HEX.

Обычно используемая запись числа в виде, содержащем целую и дробную части, считается основной формой записи вещественных чисел.

**Числа в форме с плавающей точкой**

Форма представления числа с плавающей точкой, предполагает разбиение числа на две группы цифр. Первая группа цифр называется мантиссой, вторая - порядком.

**Нормализованные числа**

**Нормализованная запись** отличного от нуля действительного числа – это запись вида

a = ± m \* Pq Где q – целое число (положительное, отрицательное или ноль) m – правильная Р-ичная дробь, у которой первая цифра после запятой не равна нулю

Для удобства вещественные числа приводят к виду так называемого нормализованного представления числа. Заключается такое представление в том, что число записывается в виде произведения на основание системы счисления, возведенное в ту или иную степень.

Нормализованная форма числа является наиболее удобной для представления дробных чисел в компьютере.

**Стандарт IEEE 754**  определяет три формата представления вещественных чисел:

1) одинарный (с одинарной точностью) – 32-разрядное нормализованное число со знаком, 8-разрядным смещенным порядком и 24-разрядной мантиссой

2) двойной (с двойной точностью) – 64-разрядное нормализованное число со знаком, 11-разрядным смещенным порядком и 53-разрядной мантиссой

3) расширенный (с повышенной точностью) – 80-разрядное число со знаком, 15-разрядным смещенным порядком и 64-разрядной мантиссой. Позволяет хранить ненормализованные числа.

Расширенный формат предназначен для уменьшения ошибки округления.

**Основы #HEX.**

Шестнадцатиричная система счисления - это система у которой в основании лежит число 16. (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

256 - 100h, 128 - 80h, 64 - 40h, 32 -20h, 16 - 10h, 255 – FFh, 127 - 7Fh, 63 - 3Fh, 31 - 1Fh, 15 - 0Fh

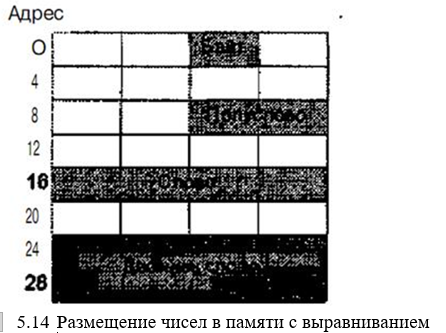
20 10h+4h=14h

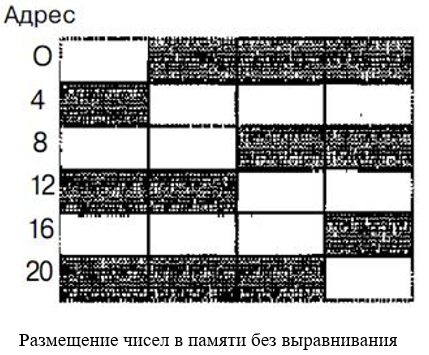
28 20h-4h=1Ch

Всю информацию выгодно представлять байтами. 1 байт это два значения в 16 ричной системе. Соответственно мы можем так представить всю информацию.

# Размещение числовых данных в памяти. Принцип обезличивания кода.

**Размещение числовых данных в памяти.**

В современных ВМ разрядность одной ячейки памяти, как правило, равна одному байту (8 бит). В то же время реальная длина кодов чисел составляет 2, 4, 8 или 16 байт. При хранении таких чисел в памяти последовательные байты числа размещают в нескольких ячейках с последовательными адресами, при этом для доступа к числу указывается только наименьший из адресов. При разработке архитектуры системы команд необходимо определить порядок размещения байтов в памяти, то есть какому из байтов (старшему или младшему) будет соответствовать этот наименьший адрес. На рис. 5.13 показаны оба варианта размещения 32-разрядного числа в четырех последовательных ячейках памяти, начиная с адреса х.

Помимо порядка размещения байтов, существенным бывает и выбор адреса, с которого может начинаться запись числа. Связано это с физической реализацией полупроводниковых запоминающих устройств, где обычно предусматривается возможность считывания (записи) четырех байтов подряд. Причем данная операция выполняется быстрее, если адрес первого байта **А** отвечает условию **A mod 5** = 0 (S = 2, 4, 8, 16). Числа, размещенные в памяти в соответствии с этим правилом называются ***выравненными***

**Принцип обезличивания кода**

Возьмем какой-либо двоичный код, например 1000 11002. Если обратиться к при- веденному ранее фрагменту кодовой таблицы , то можно утверждать, что это код буквы М. В то же время можно сказать, что этим кодом зада- ется цвет одного из пикселов монохромного изображения. Если предположить, что для кодирования использован беззнаковый формат с фиксированной точкой, то можно сказать, что это код числа 14010. Предположение об использовании знакового представления в этом же формате приведет нас к выводу о том, что в поле находится код числа -12010. Что же представляет собой этот код на самом деле? Интерпретация (то есть истолкование смысла) машинного кода может быть самой разной. Один и тот же код разными программами может рассматриваться и как число, и как текст, и как изображение, и как звук. Другими словами, как именно трактуется, понимается тот или иной машинный код, определяется обрабатывающей этот код программой.

# Представление текстовых данных в ЭВМ

Любой текст состоит из последовательности символов. Символами могут быть буквы, цифры, знаки препинания, знаки математических действий, круглые и квадратные скобки и т.д. Особо обратим внимание на символ "пробел", который используется для разделения слов и предложений между собой. Хотя на бумаге или экране дисплея "пробел" — это пустое, свободное место, этот символ ничем не "хуже" любого другого символа. На клавиатуре компьютера или пишущей машинки символу "пробел" соответствует специальная клавиша.

Текстовая информация, как и любая другая, хранится в памяти компьютера в двоичном виде. Для этого каждому символу ставится в соответствие некоторое неотрицательное число, называемое кодом символа, и это число записывается в память ЭВМ в двоичном виде. Конкретное соответствие между символами и их кодами называется системой кодировки.

В персональных компьютерах обычно используется система кодировки **ASCII**. Он ставит в соответствие каждому символу **семиразрядный** двоичный код. Легко определить, что в коде ASCII можно представить **128 символов.**

В системе ASCII закреплены две таблицы кодирования **базовая и расширенная**. **Базовая** таблица закрепляет значения кодов **от 0 до 127**, а **расширенная** относится к символам с номерами от **128 до 255.**

# Кодирование графических данных. Кодирование звуковой информации. Кодирование видеоинформации

**Кодирование графической информации**: векторная графика, растровая графика

|  |  |
| --- | --- |
| Программы, работающие с **векторной** графикой, хранят информацию об объектах, составляющих изображение в виде графических примитивов: прямых линий, дуг окружностей, прямоугольников, закрасок и т.д.  **Достоинства** векторной графики:   * + Преобразования без искажений.   + Маленький графический файл.   + Рисовать быстро и просто.   + Независимое редактирование частей рисунка.   + Высокая точность прорисовки.   + Редактор быстро выполняет операции.   **Недостатки** векторной графики:   * + Векторные изображения выглядят искусственно.   + Ограниченность в живописных средствах. | Программы растровой графики работают с точками экрана (пикселями). Это называется пространственной дискретизацией. |

**Разрешающая способность** находится по формуле: P=n\*m,гор\*вертик

Зная количество цветов, мы сможем найти количество информации, которое используется для **кодирования цвета точки**, что мы будем называть **глубиной цвета. , N- кол-во цветов,i- глубина**

Таким образом, чтобы найти **вес изображения** достаточно перемножить разрешающую способность изображения на глубину цвета: **L=P\*i.**

**КОДИРОВАНИЕ ЗВУКОВОЙ ИНФОРМАЦИИ(Гц)**

При этом на каждое измерение выделяется одинаковое количество бит. Такая характеристика называется — **глубина кодирования.**

Таким образом, чтобы подсчитать **вес звуковой волны** достаточно перемножить частоту дискретизации, глубины кодирования и времени звучания такого звука. При этом, рассматривая современное звучание, количество звуковых волн может быть различное, например, для стереозвука — это 2, а для квадрозвука 4

L=a\*u\*i\*t, где t-время, i-глубина кодированния,u-частота дискретизации,a-кол-во звуковых волн.

**КОДИРОВАНИЕ ВИДЕОИНФОРМАЦИИ**

Изображение в видео состоит из отдельных кадров, которые меняются с определенной частотой. Кадр кодируется как обычное растровое изображение, то есть разбивается на множество пикселей. Закодировав отдельные кадры и собрав их вместе, можно описать все видео.

Видеоданные характеризуются частотой кадров и экранным разрешением. Скорость воспроизведения видеосигнала составляет 30 или 25 кадров в секунду

# Исправление ошибок.

Существует множество различных факторов, которые могут повредить или исказить данные, например, влияние сильных электромагнитных полей от ближайших источников, влияние температурного расширения материалов, из которых состоят устройства, сбои в работе аппаратуры, механические удары и т. д.

Чтобы бороться с ошибками, используются специальные коды, умеющие обнаруживать и исправлять ошибки. В этом случае к каждому слову в памяти особым образом добавляются дополнительные биты. Когда слово считывается из памяти, эти дополнительные биты проверяются, что и позволяет обнаруживать ошибки.

Во многих компьютерах одним из уровней контроля за появлением ошибок является включение в байт дополнительного, контрольного девятого разряда, который формируется аппаратурой автоматически. А те восемь битов, из которых, как мы до сих пор считали, состоит байт, в дальнейшем будем называть информационными. Значение контрольного разряда определяется так, чтобы общее количество разрядов байта (всех информационных и контрольного), которые содержат 1, было нечетным (или четным).

# Код Хэмминга. Коррекция ошибок.

Начало смотреть в *ВОПРОС 8*

Кроме введения контрольного разряда существуют и более развитые способы кодирования, которые не только позволяют установить факт появления ошибок большой кратности, но и обеспечивают их исправление. К таким способам относятся коды, принцип построения которых предложил Р. Хемминг.

А теперь посмотрим, как может использоваться алгоритм Хэмминга при создании кодов исправления ошибок для слов любого размера. В коде Хэмминга, к слову, состоящему из m бит, добавляются r бит четности, при этом образуется слово длиной m + r бит. Биты нумеруются с единицы (а не с нуля), причем первым считается крайний левый. Все биты, номера которых — степени двойки, являются битами четности; остальные используются для данных. Например, к 16-разрядному слову нужно добавить 5 бит четности. Биты с номерами 1,2, 4, 8 и 16 — биты четности, все остальные — биты данных. Всего слово содержит 21 бит (16 бит данных и 5 бит четности). В рассматриваемом примере мы будем использовать проверку на четность (выбор произвольный).

Каждый бит четности позволяет проверять определенные битовые позиции. Общее число битов со значением 1 в проверяемых позициях должно быть четным. Ниже указаны позиции проверки для каждого бита четности:

• бит 1 проверяет биты 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21;

• бит 2 проверяет биты 2, 3, 6, 7, 10, 11, 14, 15, 18, 19;

• бит 4 проверяет биты 4, 5, 6, 7, 12, 13, 14, 15, 20, 21;

• бит 8 проверяет биты 8, 9, 10, 11, 12, 13, 14, 15;

• бит 16 проверяет биты 16, 17, 18, 19, 20, 21.

Коррекция ошибок:

Чтобы понять, как происходит исправление ошибок, рассмотрим, что произойдет, если бит 5 изменит значение (например, из-за резкого скачка напряжения). В результате вместо кодового слова 001011100000101101110 получится 001001100000101101110. Будут проверены 5 бит четности. Вот результаты:

+ неправильный бит четности 1 (биты 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 содержат пять единиц);

+ правильный бит четности 2 (биты 2, 3, 6, 7, 10, 11, 14, 15, 18, 19 содержат шесть единиц);

+ неправильный бит четности 4 (биты 4, 5, 6, 7, 12, 13, 14, 15, 20, 21 содержат пять единиц);

+ правильный бит четности 8 (биты 8, 9, 10, И, 12, 13, 14, 15 содержат две единицы);

+ правильный бит четности 16 (биты 16, 17, 18, 19, 20, 21 содержат четыре единицы).

Общее число единиц в битах 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 и 21 должно быть четным, поскольку в данном случае используется проверка на четность. Неправильным должен быть один из битов, проверяемых битом четности 1 (а именно 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 и 21). Бит четности 4 тоже неправильный. Это значит, что изменил значение один из следующих битов: 4, 5, 6, 7, 12, 13, 14, 15, 20, 21. Ошибка должна быть в бите, который содержится в обоих списках. В данном случае общими являются биты 5, 7, 13, 15 и 21. Поскольку бит четности 2 правильный, биты 7 и 15 исключаются. Правильность бита четности 8 исключает наличие ошибки в бите 13. Наконец, бит 21 также исключается, поскольку бит четности 16 правильный. В итоге остается бит 5, в котором и кроется ошибка.

Поскольку этот бит имеет значение 1, он должен принять значение 0. Именно таким образом исправляются ошибки.

# Многоуровневая компьютерная организация

**Уровни 0-3 разрабатываются сис программистами(машинные языки);4-5 уровни пря прикладных программистов**

**Уровень 0: Цифровой логический уровень**. Объекты на уровне 0 называются вентилями. Вентили состоят из транзисторов и смоделированы как цифровые средства, на входе у которых цифровые сигналы (или набор 0 и 1), а на выходе – результат простых функций («И» или «ИЛИ»). Несколько вентилей формируют 1 бит памяти. Биты памяти формируют регистры.

**Уровень 1: Микроархитектурный уровень**. Представляет собой локальную память (совокупность регистров) и схему, называемую АЛУ (арифметико-логическое устройство). АЛУ выполняет простые арифметические операции. Регистры вместе с АЛУ формируют тракт данных, по которому поступают данные.

**Уровень 2: Уровень архитектуры команд**. Этот уровень включает набор машинных команд, которые выполняются микропрограммой-интерпретатором или аппаратным обеспечением.

**Уровень 3: Уровень операционной системы**. Этот уровень включает набор команд уровня 2. Оставшаяся часть команд интерпретируется операционной системой.

**Уровень 4: Уровень языка ассемблера**. Представляет собой символическую форму одного из языков более низкого уровня. На этом уровне можно писать программы в приемлемой для человека форме.

**Уровень 5: Язык высокого уровня**. Обычно состоит из языков, разработанных для прикладных программистов. Наиболее известные среди них — BASIC, С, C++, Java, LISP и Prolog.

# Логические элементы (история, теория, применение).

Логические элементы — устройства, предназначенные для обработки информации в цифровой фор-ме (последовательности сигналов высокого — «1» и низкого — «0» уровней в двоичной логике, последовательности «0», «1» и «2» в троичной логике, последовательности «0», «1», «2», «3», «4», «5», «6», «7», «8» и «9» — в десятичной). Физически логические элементы могут быть выполнены механическими, электромеханическими (на электромагнитных реле), электронными (в частности, на диодах или транзисторах), пневматическими, гидравлическими, оптическими и другими.

С развитием электротехники от механических логических элементов перешли к электромеханическим логическим элементам (на электромагнитных реле), а затем к электронным логическим элементам: вначале — на электронных лампах, позже — на транзисторах. После доказательства в 1946 году теоремы Джона фон Неймана об экономичности показательных позиционных систем счисления стало известно о преимуществах двоичной и троичной систем счисления по сравнению с десятичной системой счисления. От десятичных логических элементов перешли к двоичным логическим элементам. Двоичность и троичность позволяет значительно сократить количество операций и элементов, выполняющих эту обработку, по сравнению с десятичными логическими элементами.

Логические элементы выполняют логическую функцию (операцию) над входными сигналами (операндами, данными).

Логические операции:



# Триггеры.

Триггер — простейшее последовательностное устройство, которое может находиться в одном из двух возможных состояний и переходить из одного состояния в другое под воздействием входных сигналов. Триггер является базовым элементом последовательностных логических устройств. Входы триггера разделяют на информационные и управляющие. Информационные входы используются для управления состоянием триггера. Управляющие входы обычно используются для предварительной установки триггера в некоторое состояние и для синхронизации.

Асинхронный триггер — изменяет свое состояние непосредственно в момент появления соответствующего информационного сигнала.

Синхронные триггеры в свою очередь подразделяют на триггеры со статическим и динамическим управлением по входу синхронизации C. Статические триггеры воспринимают информационные сигналы при подаче на вход C логической единицы или логического нуля.

Динамические триггеры воспринимают информационные сигналы при изменении сигнала на входе C от 0 к 1 или от 1 к 0.

Статические триггеры в свою очередь подразделяют на одноступенчатые и двухступенчатые. В одноступенчатом триггере имеется одна ступень запоминания информации, а в двухступенчатом — две такие ступени.

# Логические операции.

К основным логическим операциям относят операцию НЕ (отрицание, инверсия – NOT), операцию И (логическое умножение, конъюнкция – AND), операцию ИЛИ (логическое сложение, дизъюнкция – OR).

**Логическая функция** – это закон соответствия между логическими переменными. Логическая переменная – это такая переменная, которая может принимать одно из двух возможных значений: 0 («ложь») и 1 («истина»). Логическая функция может также принимать два значения. Из этого следует, что логические переменные и функции определены на множестве двух значений – {0,1}.

В вычислительной технике также часто используется операция «исключающее ИЛИ» (XOR), которая фактически сравнивает на совпадение два двоичных разряда. На практике, по технологическим причинам в качестве основного логического элемента используется элемент И-НЕ.

# Структура и функции центрального процессора.

Главными структурными компонентами процессора являются арифметико-логическое устройство (АЛУ) и устройство (или узел) управления (УУ).

На АЛУ возлагается функция арифметической и логической обработки данных (преобразования информации).

УУ является функционально наиболее сложным устройством процессора – оно управляет потоком данных и команд, поступающих в процессор и выходящих из него, а также порядком выполнения операций в АЛУ.

Набор регистров (другие названия – микропроцессорная память, внутренняя память процессора и т. д.) представляет собой память самого верхнего уровня в иерархии запоминающих устройств; предназначен для временного хранения адресов, данных, результатов операций, управления ходом выполнения операций в процессе функционирования процессора.

# Как работает компьютер. Шины адреса, управления и данных. Дешифрация(.mp4).

Основа – **процессор**, работа которого это выполнение программы или программного кода. При включении компьютера, процессор считывает информацию из внешней памяти. Текст сложен из последовательности букв – последовательность байтов, значения которых не превышают значения более 255, чтобы процессор знал какому конкретному байту или какой ячейке памяти он обращается. Для этого все ячейки пронумерованы, начиная с нуля. **Ячейка – это адрес**. Память делится на **постоянную** (заранее записывается программа для выполнения) и **оперативную** (хранит временные данные).

Постоянную память (ОЗУ) процессор может только **читать**, а оперативную (ОЗУ) и **читать, и записывать.**

В компьютере ПЗУ – микросхема BIOS и ОЗУ – планки DIMM.

Для обращения процессора к адресу существует **шина адреса** – это некоторое количество проводников, через которые в двоичном коде и передается адрес нужной ячейки.

**Основа работы** любого компьютера **двоичная система счисления**.

Для различия чтения или записи существует **шина управления**, которая подает сигналы. При чем активным сигналом является ноль, т.е. если чтение, то у чтения устанавливается ноль, если запись, то у записи, одновременное выполнение невозможно.

Для ответной реакции запроса существует **шина** **данных**, через которую выставляется содержимое ячейки по адресу, представленное также в двоичной системе.

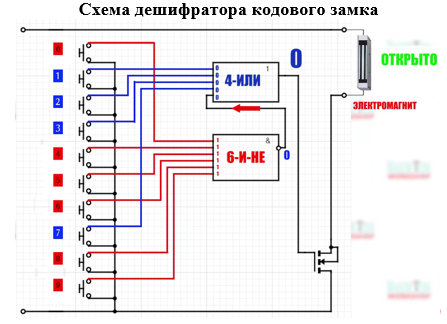
Если мы захотим записать данные в какую-то ячейку, то параллельно подключается ОЗУ и ПЗУ, но как микросхема должна понять к кому обращаемся. Для этого существует **дешифрация** – логический элемент, выдаваемый 0 при совпадении определенной комбинации 0 и 1 на входе.

Чтобы процессор понимал к какому устройству памяти он обращается, адресное пространство делится на диапазоны – ОЗУ и ПЗУ. Само же адресное пространство определяется количеством разрядов шины адреса. (ОЗУ от 65525 по 32768 и ПЗУ от 32767 по 0)

Поэтому ОЗУ и ПЗУ нужно подключить так, чтобы они активировались при обращении к тому диапазону адресов, который будет назначен для каждой из них.

У схем существует также параметры: CS-выбор микросхемы, CE-активация микросхемы, OE-разрешение выхода, RD- чтение и WE-запись соответственно).

# Дешифрация – кодовый замок (.mp4).

Называется комбинационное устройство, преобразующее n-разрядный двоичный код в логический сигнал, появляющийся на том выходе, десятичный номер которого соответствует двоичному коду. Число входов и выходов в так называемом полном дешифраторе связано соотношением m= 2n, где n- число входов, а m— число выходов.

Если в работе дешифратора используется неполное число выходов, то такой дешифратор называется неполным. Так, например, дешифратор, имеющий 4 входа и 16 выходов, будет полным, а если бы выходов было только 10, то он являлся бы неполным.

***Классификация дешифраторов:***

1. По числу разрядов.
2. В зависимости от преобразованных кодов:
   * двоично-десятичный код в семисегментный;
   * двоичный код в десятичное число.
3. По принципу действия:
   * линейный (матричный, или одноступенчатый);
   * пирамидальный (многоступенчатый).

# Принцип работы и устройство АОН (автоматический определитель номера (.mp4)).

**Устройство АОН**

(АОН-автоопределение номера). Для осуществления этой функции номер вызывающего абонента запоминается на время разговора и по запросу сообщается по служебному каналу связи на АТС(автом. телеф. станция), с которой связан вызываемый абонент. Первоначально это устройство использовалось для начисления оплаты за междугородные переговоры, для взаиморасчётов с операторами междугородной связи. Учитывая заинтересованность многих потребителей в информации о вызывающем абоненте, некоторые телефонные аппараты начали комплектовать блоками, формирующими сигналы запроса на АТС и расшифровывающими на своём дисплее ответные сигналы, содержащие номер телефона, с которого поступил вызов.

**Принцип работы:**

Абонент АТС-1, подняв трубку, звонит абоненту АТС-2. При этом у абонента АТС-2 по телефонной сети поступает индукторный вызов (вызов от АТС). Телефон АОН абонента-2 подключается к телефонной линии, понижая напряжение до уровня 22-24 В - блокирует разговорный тракт. Затем, через 250-275 мс (время на окончание переходных процессов при коммутации телефонной линии) АОН абонента-2 выдает на АТС-1 сигнал "Запрос АОН" частотой 495-505 Гц с уровнем 4.3 дб и длительностью 100 мс. На АТС-1 декодируется этот сигнал и выдается "Ответ" -номер телефона звонящего абонента АТС-1 многочастотным методом "безинтервальный пакет" в принятом коде "2 из 6". АОН у абонента-2 осуществляет прием пакета частотной информации в порядке поступления с последующей дешифрацией комбинации.

# Логическое устройство памяти (ОЗУ, ПЗУ). Объем памяти.

RAM и ROM оба являются внутренней памятью компьютера. Где **RAM** – это **временная память**, **ROM** – **постоянная** память компьютера. Существует много различий между ОЗУ и ПЗУ, но основное отличие состоит в том, что ОЗУ – это память для **чтения и запис**и, а **ПЗУ – только для** чтения.

RAM - **оперативная память**; это означает, что центральный процессор может **напрямую** обращаться к любому адресу в оперативной памяти. **Временно** хранит данные. Данные, которые должны быть **обработаны в данный** момент, должны находиться в оперативной памяти.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Байт 0 | Байт 1 | Байт 2 | Байт 3 | Байт 4 | Байт 5 | Байт 6 | Байт 7 |
| ПОЛУСЛОВО | | ПОЛУСЛОВО | | ПОЛУСЛОВО | | ПОЛУСЛОВО | |
| СЛОВО | | | | СЛОВО | | | |
| ДВОЙНОЕ СЛОВО | | | | | | | |

ПЗУ **доступно только для чтения**. ЦП **не** **может напрямую** обращаться к памяти ПЗУ, данные сначала должны быть переданы в ОЗУ, а затем ЦП может получить доступ к этим данным из ОЗУ. В ПЗУ хранится инструкция, необходимая компьютеру во время **начальной загрузки** (процесс загрузки компьютера). Содержимое в ПЗУ не может быть изменено.

**Память компьютера** построена из двоичных запоминающих элементов — **битов**, объединенных в группы по 8 битов, которые называются **байтами**. (Единицы измерения памяти совпадают с единицами измерения информации). Все байты пронумерованы. Номер байта называется его **адресом**.

Байты могут объединяться в ячейки, которые называются также **словами**. Для каждого компьютера характерна определенная длина слова — два, четыре или восемь байтов. Как правило, в одном машинном слове может быть представлено либо одно целое число, либо одна команда. Однако, допускаются переменные форматы представления информации.

# Как работает ЭВМ. Собираем простейший компьютер на базе Z80 эмулятора ПЗУ (.mp4).

Современные ЭВМ построены в соответствии с принципами, сформулированными фон Нейманом в 1945 г.:

1. Принцип программного управления: ЭВМ работает по программе, которая находится в оперативной памяти и выполняется автоматически; программы дискретны и представляют собой последовательность команд, каждая из которых осуществляет отдельный акт преобразования информации; все разновидности команд образуют систему команд машины.

2. Принцип условного перехода: При выполнении программы возможен переход к той или иной команде в зависимости от промежуточных результатов вычислений; это допускает создание циклов.

3. Принцип хранимой информации: Команды как и операнды представляются в машинном коде и хранятся в оперативной памяти. При работе команды обрабатываются устройством управления процессора, а операнды -- арифметико-логическим устройством.

4. Принцип использования двоичной системы счисления: Информация кодируется в двоичной форме и разделяется на элементы, называемыми словами. В двоичной системе используются две цифры 0 и 1, что соответствует двум состояниям двустабильной системы (кнопка нажата-отпущена, транзистор открыт-закрыт, ...)

5. Принцип иерархичности ЗУ: Компромиссом между необходимыми большой емкостью памяти, быстрым доступом к данным, дешевизной и надежностью является иерархия запоминающих устройств: 1) быстродействующее ОЗУ, имеющее небольшую емкость для операндов и команд, участвующих в вычислениях; 2) инерционное ВЗУ, имеющее большую емкость для информации, не участвующей в данный момент в работе ЭВМ.

# АРДУИНО и Микроконтроллеры. Для Начинающих(.mp4).

Arduino – это не только плата микрокомпьютера (чаще всего под названием «Arduino» подразумевается именно это), но и полная платформа, основанная на простом в использовании аппаратном и программном обеспечении. Важно отметить, что это платформа **типа open source**, что означает доступ к бесплатной, подробной документации, а также схемам и источникам программ. Как правило, устройство, выполненное на основе платформы Arduino, **состоит из базовой платы** с микроконтроллером и **прилагаемого к ней модуля расширения**, называемого shield.

Благодаря простоте использования, доступности, а также разнообразным потребностям пользователей, среди проектов, реализованных с помощью Arduino, можно найти решения многочисленных проблем и созданные практически любые устройства. Их можно использовать напрямую или в качестве проекта отнесения. Программное обеспечение для создания приложений (Arduino IDE) очень простое в использовании для новичков, но в то же время достаточно гибкое и дающее множество возможностей продвинутым пользователям. Его можно запустить с помощью операционных систем Mac OS, Windows и Linux. Это делает его доступным для пользователей с различными аппаратными предпочтениями и разными финансовыми возможностями.

Используют Arduino для создания недорогих приборов для измерения не только электрических величин, но и предназначенных для использования в химических и физических экспериментах.

**Микроконтроллер** - это такая специальная микросхема, которую можно программировать.

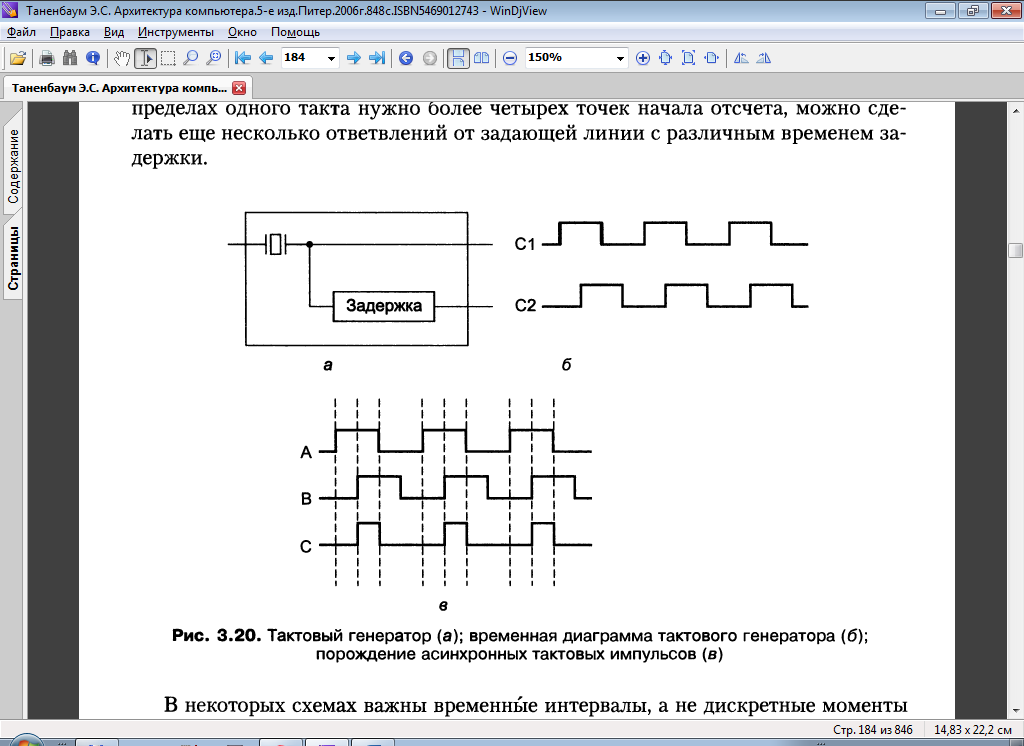
Однако для создания полноценной вычислительной системы требуются и другие элементы: память, элементы ввода-вывода и т.п. В итоге микропроцессорная система состояла из нескольких микросхем, и управлял всей этой системой **микропроцессор**.

Однако техника не стояла на месте. Учёным и инженерам удалось сделать кристаллы микросхем ещё меньше. И кому-то пришла идея объединить все элементы микропроцессорной системы в одну микросхему. Так появился микроконтроллер.

# Память. Тактовые генераторы. Регистры.

*Можно так же посмотреть вопрос 27 или 22, там поменьше*

**Тактовые генераторы**

Тактовый генератор — это схема, которая вызывает серию импульсов. Все импульсы одинаковы по длительности. Интервалы между последовательными импульсами также одинаковы. Временной интервал между началом одного импульса и началом следующего называется **временем такта**. Частота тактового генератора обычно контролируется кварцевым генератором, позволяющим добиться высокой точности.

В компьютере за время одного такта может произойти множество событий. Если они должны осуществляться в определенном порядке, то такт следует разделить на подтакты. Чтобы достичь лучшего разрешения, чем у основного тактового генератора, нужно сделать ответвление от задающей линии тактового генератора и вставить схему с определенным временем задержки.

Связав различные события с разными перепадами (фронтами и спадами), можно достичь требуемой последовательности выполнения действий. Если в пределах одного такта нужно более четырех точек начала отсчета, можно сделать еще несколько ответвлений от задающей линии с различным временем задержки.

Тактовый генератор (а); временная диаграмма тактового генератора (б);порождение асинхронных тактовых импульсов (в)

**Память**

Компью́терная па́мять (*устройство хранения информации*, [*запоминающее устройство*](https://dic.academic.ru/dic.nsf/ruwiki/639011)) — часть [вычислительной машины](https://dic.academic.ru/dic.nsf/ruwiki/1662), физическое устройство или среда для хранения [данных](https://dic.academic.ru/dic.nsf/ruwiki/114702), используемых в вычислениях, в течение определённого времени. Память, как и [центральный процессор](https://dic.academic.ru/dic.nsf/ruwiki/1187537), является неизменной частью компьютера с 1940-х. Память в вычислительных устройствах имеет [иерархическую структуру](https://dic.academic.ru/dic.nsf/ruwiki/1504290) и обычно предполагает использование нескольких запоминающих устройств, имеющих различные характеристики.

В персональных компьютерах «памятью» часто называют один из её видов — [динамическая память с произвольным доступом](https://dic.academic.ru/dic.nsf/ruwiki/105783) (DRAM), которая в настоящее время используется в качестве [ОЗУ](https://dic.academic.ru/dic.nsf/ruwiki/1069659) персонального компьютера.

Задачей компьютерной памяти является хранение в своих [ячейках](https://dic.academic.ru/dic.nsf/ruwiki/231618) состояния внешнего воздействия, запись [информации](https://dic.academic.ru/dic.nsf/ruwiki/4939). Эти ячейки могут фиксировать самые разнообразные физические воздействия (см. [ниже](https://dic.academic.ru/dic.nsf/ruwiki/2182#.D0.A4.D0.B8.D0.B7.D0.B8.D1.87.D0.B5.D1.81.D0.BA.D0.B8.D0.B5_.D0.BF.D1.80.D0.B8.D0.BD.D1.86.D0.B8.D0.BF.D1.8B)). Они функционально аналогичны обычному [электромеханическому переключателю](https://dic.academic.ru/dic.nsf/ruwiki/490142) и информация в них записывается в виде двух чётко различимых состояний — 0 и 1 («выключено»/«включено»). Специальные механизмы обеспечивают доступ (*считывание*, произвольное или последовательное) к состоянию этих ячеек.

Процесс доступа к памяти разбит на разделённые во времени процессы — операцию записи ([сленг.](https://dic.academic.ru/dic.nsf/ruwiki/327) *прошивка*, в случае записи [ПЗУ](https://dic.academic.ru/dic.nsf/ruwiki/12020)) и операцию чтения, во многих случаях эти операции происходят под управлением отдельного специализированного устройства — [контроллера памяти](https://dic.academic.ru/dic.nsf/ruwiki/1527851).

**Регистр**

**Регистр** — это последовательностное логическое устройство, используемое для хранения n-разрядных двоичных чисел и выполнения преобразований над ними. Регистр представляет собой упорядоченную последовательность триггеров, число которых соответствует числу разрядов в слове. С каждым регистром обычно связано комбинационное цифровое устройство, с помощью которого обеспечивается выполнение некоторых операций над словами.

**Типичными являются следующие операции:**

-прием слова в регистр;

-передача слова из регистра;

-поразрядные логические операции;

-сдвиг слова влево или вправо на заданное число разрядов;

-преобразование последовательного кода слова в параллельный и обратно;

**Классификация регистров**

-накопительные (регистры памяти, хранения);

-сдвигающие.

**В свою очередь сдвигающие регистры делятся:**

по способу ввода-вывода информации на:

• параллельные,

• последовательные,

• комбинированные;

по направлению передачи информации на:

• однонаправленные,

• реверсивные.

# Организация памяти.

Запоминающие устройства ЭВМ делятся на основную память, сверхоперативную память (СОЗУ) и внешние запоминающие устройства.

1. Основная память включает в себя два типа устройств: оперативное запоминающее устройство (**ОЗУ, или RAM** - Random Access Memory} и постоянное запоминающее устройство (**ПЗУ, или ROM** -Read Only Memory}

ОЗУ предназначено для **хранения переменной** временной информации. Оно допускает изменение своего содержимого в ходе выполнения процессором вычислительных операций с данными и может работать в режимах записи, чтения, хранения.

ПЗУ содержит **информацию, неизменяемую** в ходе выполнения процессором вычислительных операций, например, драйверы, организующие ввод-вывод информации со стандартных периферийных устройств, стандартные программы и константы. Эта информация заносится в ПЗУ перед установкой микросхемы в ЭВМ. Основные операции, которые может выполнять ПЗУ, - чтение и хранение.

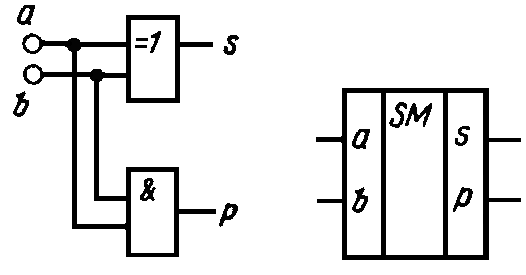
**СОЗУ – регистры, кэш-память.**

Внешние запоминающие устройства – устройства хранения данных (дополнительная память), накопитель.

# Полусумматоры, сумматоры. Назначение, классификация, принцип работы, типовые схемы.(.mp4)

**Полусумматор** — логическая схема, имеющая два входа и два выхода (двухразрядный сумматор, бинарный сумматор). Полусумматор используется для построения двоичных сумматоров. Полусумматор позволяет вычислять сумму A+B, где A и B — это разряды двоичного числа, при этом результатом будут два бита S и C, где S — это бит суммы по модулю 2, а P (C) — бит переноса.

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | S | P |
| 0  0  1  1 | 0  1  0  1 | 0  1  1  0 | 0  0  0  1 |

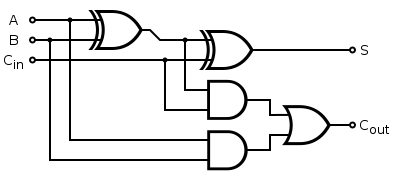
Полусумматор обозначается буквами HS\*.

**Сумматор** – это логическая электронная схема, выполняющая сложение двоичных чисел. Сумматор является главной частью процессора.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | C | P | S |
| 0  0  0  0  1  1  1  1 | 0  0  1  1  0  0  1  1 | 0  1  0  1  0  1  0  1 | 0  0  0  1  0  1  1  1 | 0  1  1  0  1  0  0  1 |

Одноразрядный сумматор должен иметь три входа: А, В – слагаемые и Р0 – перенос из предыдущего разряда и

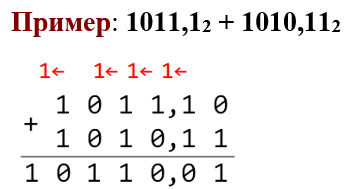
выходы: S – сумма и Р – перенос.



**Последовательный сумматор** состоит из одноразрядного сумматора, на входы которого из сдвигающих регистров, хранящих слагаемые A и В, подаются по тактам разряд за разрядом коды этих чисел, начиная с младшего разряда. Если время элемента задержки равно одному такту, то сигнал переноса от предыдущего разряда, сформированный в предыдущем такте, поступит на нижний вход сумматора только в следующем такте, когда на входы anb будут поданы значения следующего разряда слагаемых.

Значительно меньшее время выполнения операции имеет **параллельный сумматор**. В этом устройстве операция сложения производится одновременно во всех разрядах чисел A и В, поступающих в параллельном коде. Для этого схему составляют из п одноразрядных сумматоров, соединяя выход переноса г-го разряда со входом переноса соседнего (г + 1)-го разряда. Такой сумматор называют сумматором с последовательным переносом. После того как сформируется результат на выходах всех одноразрядных сумматоров, он запоминается в параллельном регистре.

# Арифметико-логическое устройство. Сложение двоичных чисел.(.mp4)

****Арифметико-логическое устройство (АЛУ) предназначено для выполнения арифметических и логических преобразований над числами и словами.

|  |  |  |
| --- | --- | --- |
| Сложение | Вычитание | Умножение |
| 0+0=0 | 0-0=0 | 0 ∙0=0 |
| 0+1=1 | 1-0=1 | 0 ∙1=0 |
| 1+0=1 | 1-1=0 | 1 ∙0=0 |
| 1+1=10 | 10-1=1 | 1 ∙1=1 |

# Видеоурок 74-75. Арифметико-логические устройства (АЛУ). (.mp4)

Что такое АЛУ?

Арифметико-логическое устройство – это один из компонентов процессора, который необходим для осуществле-ния преобразований логического и арифметического типа, начиная элементарными и заканчивая сложными выраже-ниями. Разрядность используемых операндов принято считать длиной слова, или размером.

Главная задача АЛУ заключается в переработке данных, хранящихся в оперативной памяти компьютера. Кроме то-го, арифметико-логическое устройство способно производить сигналы управления, которые направляют ЭВМ на выбор правильного пути для выполнения необходимого вычислительного процесса в зависимости от итоговых ти-пов данных. Все операции задействуют электронные схемы, каждая из которых структурно делится на тысячи эле-ментов. Такие платы обычно быстродейственные и отличаются высокой плотностью.

В зависимости от сигналов, которые поступают на вход, АЛУ выполняют разные типы операций с двумя числами. Любое арифметико-логическое устройство компьютера предусматривает реализацию четырех базовых действий, сдвигов, а также логических преобразований. Набор операций АЛУ – это его главная характеристика.

**Узлы хранения АЛУ**

К этой категории относятся:

* триггеры, хранящие вспомогательные биты и разные признаки результатов;
* регистры, отвечающие за целостность операндов, промежуточных и конечных итогов.

Иногда регистры арифметико-логического устройства могут объединяться в специализированный блок памяти, а триггеры - формировать единый регистр состояния.

**Узлы передачи АЛУ**

К этой категории относятся:

* шины, соединяющие между собой блоки устройства;
* мультиплексоры и вентили, отвечающие за выбор правильного направления выполнения операций.

**Узлы преобразования АЛУ**

Сюда относятся:

* сумматоры, выполняющие микрооперации;
* схемы выполнения логических действий;
* сдвигатели;
* корректоры для десятичной арифметики;
* преобразователи кода, использующиеся для получения обратных или дополнительных данных;
* счетчики для подсчета количества выполненных циклов и для реализации вспомогательных преобразований.

**Узлы управления АЛУ**

К этой категории объектов относятся:

* контрольный блок;
* дешифратор сигналов;
* схемы преобразования логических признаков, необходимые для формирования ветвей для выполнения микропрограмм.

**Операции устройства**

Структура АЛУ предполагает выполнение действий через логические функции, которые делятся на такие группы:

* десятичная арифметика;
* двоичная арифметика для цифр с четко обозначенной точкой;
* шестнадцатеричная арифметика для выражений с плавающим разделителем;
* модификация адресов команд;
* операции логического типа;
* преобразование алфавитно-цифровых полей;
* специальная арифметика.

# Архитектура компьютера на базе процессора 8086. Внешние устройства компьютера.

Процессор Intel 8086 – первый 16-битный процессор.

Состоит из модуля выполнения и модуля интерфейса шины.

Модуль выполнения содержит: блок АЛУ для выполнения логических и арифметических операций, регистры, флаги.

Модуль интерфейса шины отвечает за обработку данных и отправку инструкций, считывание адресов и информации и запись данных с портов ввода-вывода. Включает очередь инструкций (ключевая особенность процессора), способную хранить до 6 байт инструкций в буфере, отсылая новые инструкции по конвейеру после того, как от модуля выполнения поступит соответствующий запрос.

Взаимодействие модулей происходит при помощи шины данных.

Внешние устройства компьютера (периферийные устройства) — внешние компоненты аппаратного обеспечения компьютера. К ним относятся: устройства ввода (клавиатура, мышь, тачпад, трекбол, сканер, джойстик, графический планшет, плоттер), устройства вывода (монитор, принтер, модем, колонки, наушники), хранения и передачи информации (внешний жесткий диск), связанные функционально с центральным процессором в соответствии со структурой компьютерной системы.

# Как работает процессор (.mp4)

**Процессор** — это сердце/мозг любого компьютера. Его основное назначение — арифметические и логические операции

**Два основных компонента процессора**

**Устройство управления** (УУ) помогает процессору контролировать и выполнять инструкции. УУ сообщает компонентам, что именно нужно делать. В соответствии с инструкциями он координирует работу с другими частями компьютера, включая второй основной компонент — арифметико-логическое устройство (АЛУ). Все инструкции вначале поступают именно на устройство управления.

**Арифметико-логическое устройство** выполняет все арифметические и логические операции, например сложение, вычитание, логическое ИЛИ и т. п. АЛУ состоит из логических элементов, которые и выполняют эти операции.

**Хранение информации — регистры и память**

Процессор выполняет поступающие на него команды. Команды в большинстве случаев работают с данными, которые могут быть промежуточными, входными или выходными. Все эти данные вместе с инструкциями сохраняются в регистрах и памяти.

**Регистр** — минимальная ячейка памяти данных. Регистры состоят из триггеров. Триггеры, в свою очередь, состоят из логических элементов и могут хранить в себе 1 бит информации.

**Память (ОЗУ)** — это большая группа этих самых регистров, соединённых вместе. Память у такого хранилища непостоянная и данные оттуда пропадают при отключении питания. ОЗУ принимает адрес ячейки памяти, в которую нужно поместить данные, сами данные и флаг записи/чтения, который приводит в действие триггеры.

**Тактирование процессора**

Быстродействие компьютера определяется тактовой частотой его процессора. Тактовая частота — количество тактов (соответственно и исполняемых команд) за секунду.

Частота нынешних процессоров измеряется в ГГц (Гигагерцы). 1 ГГц = 10⁹ Гц — миллиард операций в секунду.

Чтобы уменьшить время выполнения программы, нужно либо оптимизировать (уменьшить) её, либо увеличить тактовую частоту. У части процессоров есть возможность увеличить частоту (разогнать процессор), однако такие действия физически влияют на процессор и нередко вызывают перегрев и выход из строя.

# Программная модель оперативной памяти.

В процессе разработки и написания программ программист абстрагируется, отвлекается от конкретных физических и технических особенностей, от деталей реализации битов и байтов оперативной памяти. Он представляет себе память с точки зрения ее эффективного использования при написании программ, принимая во внимание только необходимые для этого свойства памяти. Другими словами, программист использует **программную модель оперативной памяти**.

Оперативную память компьютера естественно рассматривать как упорядоченную в порядке возрастания адресов совокупность байтов. Однако любое обращение к памяти со стороны процессора по ряду технических соображений, которые обеспечивают ускорение передачи данных, осуществляется не по произвольным адресам байтов, а только по адресам слов памяти. Это означает, что процессор может за одно обращение к оперативной памяти прочитать содержимое слова с адресом.

Но процессор не может обратиться в память ни к байту с адресом, например, 0000516, ни к нестандартному полю с этим же адресом, ни к любому другому байту или полю с нечетным адресом. Если все-таки требуется выполнить обращение по нечетному адресу, то процессор осуществит его за несколько шагов. Например, при чтении сначала будут выполнены обращения по четным адресам соседних слов, а затем из выбранных слов будут отобраны требуемые байты.

для размещения данных в оперативной памяти, предпочитают выбирать поля с четными адресами. Такой прием называется **выравниванием данных на границах** **слов**. В связи с этим программисты, работающие на уровне машинных команд, часто рассматривают оперативную память как упорядоченную последовательность слов памяти с четными адресами (рисунок 8.6), а не как последовательность ее байтов.

# Программная модель процессора і8086.

Под программной моделью, понимается набор внутренних регистров и флагов процессора(*флаги описаны в вопросе 32*), которые, тем или иным образом доступны программисту.

По назначению и способу использования регистры можно разбить на следующие группы:

* регистры общего назначения (AX, BX, CX, DX, SI, DI, BP, SP);
* сегментные регистры (CS, DS, SS, ES);
* указатель команд (IP);

Любой регистр можно рассматривать как один 16 разрядный регистр или как два 8 разрядных регистра. Например, **bx** – шестнадцатиразрядный регистр, **bh** и **bl** –восьмиразрядные регистры, причем **bh**  старший, а **bl** – младший.

*Общего назначения*

Основной функцией **ax, cx, dx** и **bx**  регистров является временное хранение промежуточных результатов вычислений.

*Сегментные регистры*

Эти регистры используются для сегментирования адресов, которое является разновидностью модификации адресов и которое используется для сокращения размера команд.

*Указатель команд*

В регистре IP всегда находится адрес команды, которая должна быть выполнена следующей.

# Архитектура x86. («Уверены, что знаете свой домашний компьютер (.mp4)»)

**С точки зрения архитектуры** микропроцессор Intel 8086 состоял из двух аппаратных модулей – модуля выполнения и модуля интерфейса шины. Модуль выполнения указывал модулю интерфейса шины, откуда получать данные инструкций, а после этого приступал к их подготовке и выполнению. Его суть сводилась к управлению данными с помощью декодера инструкций и блока ALU, при этом сам модуль не имел прямого соединения с шинами данных, и работал исключительно через модуль интерфейса шины.

Чуть позже, в 1979 году, был разработан **Intel 8088**, который работал на тех же частотах, что и **Intel 8086, но** **использовал 8-разрядную** шину данных (внутренняя шина процессора осталась 16-разрядной) для обеспечения большей совместимости с имевшейся в то время в ходу периферией. Благодаря более низкой цене, широко использовался в ранних системах IBM PC вместо 8086.

**В 1982 году были выпущены 80186 и 80188**

80386 (i386)- **Первый 32-разрядный** процессор, работал на частотах 16-40 МГц. Появился **в 1985 году**. Знаменовал собой революцию в мире процессоров x86. Основные принципы, заложенные в этом чипе, без кардинальных изменений дожили и до наших дней (за всё это время изменения касались, в основном, повышения производительности, расширения набора команд, увеличения разрядности). Первые 386 процессоры содержали серьёзную ошибку, приводящую к невозможности функционирования в защищенном режиме. Исправленная версия называлась 386DX.

Также выпускались более дешевые процессоры i386SX с урезанной до 16 бит внешней шиной данных и 24-битной шиной адреса.

i386 — первый процессор, который мог использовать кэш-память (расположенную на внешнем чипе).

80486 (i486)

Pentium (i586)

Pentium (1993 год. Intel отказалась от номерных названий типа 8086, 80286 и др., потому что не могла запатентовать числа) — **первый суперскалярный и суперконвейерный** процессор Intel.

**Суперскалярность** — означает, что процессор позволяет выполнять более одной операции за один такт. **Суперконвейерность** означает, что процессор имеет несколько вычислительных конвейеров. У Pentium их два, что позволяет ему при одинаковых частотах в идеале быть вдвое производительней 486, выполняя сразу 2 инструкции за такт.

# Архитектура ЭВМ. Лекция 1. Как работает компьютер.mp4

Основа – **процессор**, работа которого это выполнение программы или программного кода. При включении компьютера, процессор считывает информацию из внешней памяти. Текст сложен из последовательности букв – последовательность байтов, значения которых не превышают значения более 255, чтобы процессор знал какому конкретному байту или какой ячейке памяти он обращается. Для этого все ячейки пронумерованы, начиная с нуля. **Ячейка – это адрес**. Память делится на **постоянную** (заранее записывается программа для выполнения) и **оперативную** (хранит временные данные).

Постоянную память (ОЗУ) процессор может только **читать**, а оперативную (ОЗУ) и **читать, и записывать.**

В компьютере ПЗУ – микросхема BIOS и ОЗУ – планки DIMM.

Для обращения процессора к адресу существует **шина адреса** – это некоторое количество проводников, через которые в двоичном коде и передается адрес нужной ячейки.

**Основа работы** любого компьютера **двоичная система счисления**.

Для различия чтения или записи существует **шина управления**, которая подает сигналы. При чем активным сигналом является ноль, т.е. если чтение, то у чтения устанавливается ноль, если запись, то у записи, одновременное выполнение невозможно.

Для ответной реакции запроса существует **шина** **данных**, через которую выставляется содержимое ячейки по адресу, представленное также в двоичной системе.

Если мы захотим записать данные в какую-то ячейку, то параллельно подключается ОЗУ и ПЗУ, но как микросхема должна понять к кому обращаемся. Для этого существует **дешифрация** – логический элемент, выдаваемый 0 при совпадении определенной комбинации 0 и 1 на входе.

Чтобы процессор понимал к какому устройству памяти он обращается, адресное пространство делится на диапазоны – ОЗУ и ПЗУ. Само же адресное пространство определяется количеством разрядов шины адреса. (ОЗУ от 65525 по 32768 и ПЗУ от 32767 по 0)

Поэтому ОЗУ и ПЗУ нужно подключить так, чтобы они активировались при обращении к тому диапазону адресов, который будет назначен для каждой из них.

У схем существует также параметры: CS-выбор микросхемы, CE-активация микросхемы, OE-разрешение выхода, RD- чтение и WE-запись соответственно).

# Архитектура ЭВМ. Лекция 3. Процессоры x86.mp4

Разрядность регистров – 32 бита: -ЕАХ — аккумулятор -ЕВХ — адрес данных -ЕСХ — счётчик циклов -ЕDХ — для хранения данных - ЕSI — адрес источника - EDI — адрес приёмника - ЕВР — указатель на данные в стеке - ЕSР — указатель на вершину стека - ЕIР — счётчик команд -EFLAGS — регистр флагов

Архитектура х86 ограничена 4 Гбайтами памяти, с которыми может работать компьютер и его приложения. Имеет плоскую модель памяти. Число инструкций общего назначения ⁓ 257.

Самые ходовые флаги состояния:

ZF — zero flag (флаг нуля) показывает равенство результата нулю

СF — carry flag (флаг переноса) показывает наличие переполнения в беззнаковой целочисленной арифметике

SF — sign flags (флаг знака) показывает знак результата

ОF — overflow flags (флаг переполнения) показывает наличие переполнения в знаковой целочисленной арифметике

РF — parity flags (флаг чётности) показывает чётность результата

АF — auxiliary flags (вспомогательный флаг переноса) показывает наличие переполнения, а двоично-десятичной арифметике

DF — direction flag (флаг направления)

Другие архитектуры процессоров: ARM, Atmel, Microship PIC, PowerPC, SPARC, MIPS, Эльбрус.

# Машинные команды процессора і8086.

команды

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| * Mov | * Int 21h(для работы с ос) | * ret(возвращение из процедуры) | * ADD * INC(++) | * SUB(-) * DEC(--) * NEG(\*-1) * CMP()compare | * IMUL * IDIV(разделить) * DIV(.без знака) | * AND(лог и) * OR(лог или) * NOT(лог не) |

Флаги

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| JZ/JNZ | JA/JAE(>**/**>=) | JO/ JNOпереполн | JB(<)  JBE(<=) | JSфлаг знака устан | JNS | JP/JNP флаг четност | JE/JNE |

# Прерывания.

Прерывания:

* аппаратные – это сигнал любого устройства системы для процессора, который по этому сигналу должен обслужить данное устройство
* программные – создается программами BIOS или DOS для вызова сервисных подпрограмм

команда прерывания: ***INT*** *number*

вызывает подпрограммы операционной системы. Эти прерывания имеют номера от 0 до FFh.Перед вызовом команда в регистр АХ помещают номер функции, который определяет необходимую подпрограмму. Другие регистры тоже могут использоваться в прерывании.

* параметр number сообщает процессору местонахождение вектора в таблице векторов прерываний;
* процессор передает управление по указанному в векторе прерываний адресу процедуры обработки прерывания (ПОП);

# Сергей Шумский. Интеллект машинный и человеческий.mp4

В видео рассказывается, как функционирует мозг и из чего он состоит.

По принципам устройсва человеческого мозга был построено принцип глубокого обучения(нейросети с большим кол-вом слоев).

Неокортекс обучается методом без учителя.

**Неокортекс** – это часть мозга, отвечающая за выполнение функций мозга более высокого порядка.

Базальные ганглии обучаются с подкреплениями. Тут расположены тормозные нейроны, тормозят таламус.

Мозжечок обучается с учителем. Учителем выступает кора. Кора состоит из рабочих модулей(колонки из нейронов).

Кора состоит из карт, где нейроны имеют характерные признаки и близкие признаки расположены близко друг к другу. В первичной зрительной коре есть колонки, где каждая специализируется на своем сигнале. Колонки распознают наклоны градиента света в какой-то точке сетчатки.

Зная это, мы может предположить, что вся кора состоит из такого же рода карт, где кодируется таким же образом информация, только работают с другой информацией.

Допамин – сигнал к обучение(гормон радости). В случае правильного поведения выделяет и выливается в щель между корой и стреатумом. Испытываем радость.

Таламус(коммуникационный центр). Является связующим звеном. Объединяет полученную информацию из разных участков коры мозга. Обеспечивает сознание, внимание и управление этим вниманием.

Самое большое различие между мозгом человека и искусственной нейросетью – это масштабы нейронных сетей мозга. Важно не просто количество нейронов в мозге (которое исчисляется миллиардами), но и поразительное количество связей между ними.

# Архитектура фон Неймана. Семейства компьютеров.

Принципы, лежащие в основе архитектуры ЭВМ, были сформулированы в 1945 году Джоном фон Нейманом, который развил идеи Чарльза Беббиджа, представлявшего работу компьютера как работу совокупности устройств: обработки, управления, памяти, ввода-вывода.

Принципы фон Неймана.

1. Принцип однородности памяти. Над командами можно выполнять такие же действия, как и над данными.

2. Принцип адресуемости памяти. Основная память структурно состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти, так чтобы к хранящимся в них значениям можно было бы впоследствии обращаться или менять их в процессе выполнения программы с использованием присвоенных имен.

3. Принцип последовательного программного управления. Предполагает, что программа со-стоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.

4. Принцип жесткости архитектуры. Неизменяемость в процессе работы топологии, архитектуры, списка команд.

Семейство компьютеров - электронных технических приспособлений для переработки информации - довольно велико и разнообразно. Есть маленькие счетные устройства - микрокалькуляторы, которые помещаются в наручных часах, шариковых ручках: крохотные кнопки-числа, которые нужно нажимать иголкой или остриём карандаша, и несколько операций - четыре действия арифметики, вычисление процентов, возведение в степень, извлечение корня.

Компьютеры побольше - размером с карточку - календарь и такие же плоские. На них и кнопок никаких нет, и вообще нет никаких движущихся деталей. Есть индивидуальные компьютеры с памятью на компакт-диске.

# Процессоры. Устройство центрального процессора.

Процессор называют мозгом компьютера. С его помощью обрабатывается программный код, работает операционная система и устанавливаемые приложения. Чем выше скорость процессора, тем быстрее работает компьютер.

**Процессор состоит из следующих компонентов:**

1. **ядро** (ядра) — сюда входят регистр (внутренняя память), кэш (быстрая память), арифметико-логическое устройство;
2. **шины** — с их помощью происходит передача данных, а также управление операциями и внешними составляющими компьютера.

**ЦПУ выполняет две важные функции:**

управление всеми операциями ПК, начиная сложением двух чисел и заканчивая обработкой видео или запуском игр;

обработка данных с помощью арифметических и логических операций.

**В состав центрального процессора входят:**

* устройство управления (УУ);
* арифметико-логическое устройство (АЛУ);
* запоминающее устройство (ЗУ) на основе регистров процессорной памяти и кэш-памяти процессора;
* генератор тактовой частоты (ГТЧ).

Устройство управления организует процесс выполнения программ и координирует взаимодействие всех [устройств ЭВМ](https://studopedia.ru/13_121580_ustroystvo-i-printsip-deystviya-evm.html) во время её работы.

Арифметико-логическое устройство выполняет арифметические и логические операции над данными: сложение, вычитание, умножение, деление, сравнение и др.

[Запоминающее устройство](https://studopedia.ru/11_202763_vneshnie-zapominayushchie-ustroystva-vidi-printsipi-raboti-osnovnie-harakteristiki.html) — это внутренняя память процессора. Регистры служит промежуточной быстрой памятью, используя которые, процессор выполняет расчёты и сохраняет промежуточные результаты. Для ускорения работы с оперативной памятью используется [кэш-память](https://studopedia.ru/2_60167_kesh---pamyat.html), в которую с опережением подкачиваются команды и данные из оперативной памяти, необходимые процессору для последующих операций.

Генератор тактовой частоты генерирует электрические импульсы, синхронизирующие работу всех узлов компьютера. В ритме ГТЧ работает центральный процессор.

# Принципы разработки современных компьютеров.

**Все команды должны выполняться непосредственно аппаратным обеспечением**. То есть, обычные команды не должны интерпретируются микрокомандами. Устранение уровня интерпретации повышает скорость выполне-ния большинства команд.

**Компьютер должен запускать на выполнение как можно больше команд в единицу времени**. Этот принцип предполагает, что параллелизм должен играть главную роль в повышении производительности, поскольку запустить на выполнение большое количество команд за короткий промежуток времени можно только в том случае, если есть возможность одновременного выполнения нескольких команд.

**Команды должны легко декодироваться**. Предел количества запускаемых в единицу времени команд зависит от темпа декодирования отдельных команд. Декодирование команд позволяет определить, какие ресурсы им необходимы, и какие действия нужно выполнить. Полезно все, что способствует упрощению этого процесса. Например, можно использовать единообразные команды с фиксированной длиной и с небольшим количеством полей. **Чем меньше разных форматов команд, тем лучше**.

**К оперативной памяти должны обращаться только команды загрузки и** сохранения. Один из самых простых способов разбить операцию на отдельные шаги — сделать так, чтобы операнды большей части команд брались из регистров и возвращались туда же.

К оперативной памяти должны обращаться только команды загрузки и сохранения **(LOAD и STORE).**

**Регистров должно быть много**. Поскольку доступ к оперативной памяти происходит довольно медленно, в компьютере должно быть много регистров общего назначения (по крайней мере, 32). Если слово однажды вызвано из оперативной памяти, при наличии большого числа регистров оно может содержаться в регистре до тех пор, пока не потребуется. Возвращение слова из регистра в оперативную память и новая загрузка этого же слова в регистр нежелательны. Лучший способ избежать излишних перемещений — наличие достаточного количества регистров.

# Кэш-память. Вспомогательная память.

Для ускорения доступа к оперативной памяти используется специальная сверхбыстродействующая **кэш-память**, которая расположена «**между**» процессором и оперативной памятью. Она хранит копии наиболее часто используемых участков оперативной памяти.

При обращении процессора к памяти прежде всего проводится поиск нужных сведений в кэш-памяти, поэтому и можно считать, что **функционально** **она** **расположена** перед *оперативной памятью*, хотя **конструктивно** **ее размещение** на материнской плате *может быть другим*.

**Объем** кэш-памяти невелик (обычно от 128 до 512 Кбайт), и его **недостаток** особенно ощущается при работе с внешними устройствами (например, при записи информации на компакт-диски).

**Оперативная память**, описанная Дж. Нейманом как оперативное **запоминающее устройство**, предназначена для хранения кода выполняемой программы, данных для их обработки процессором и получаемых при этом результатов.

Оперативную память часто называют **RAM** (Random Access Memory) — память произвольного доступа, поскольку при выполнении программ процессор может обращаться к любой доступной ячейке. Данные в оперативной памяти **сохраняются только на текущий сеанс работы**, т. е. до момента выключения компьютера. **При** **выключении** компьютера содержимое оперативной памяти **стирается**.

# Микроархитектурный уровень.

Над цифровым логическим уровнем находится микроархитектурный уровень. Его задача — интерпретация команд уровня 2 (уровня архитектуры команд). Строение микроархитектурного уровня зависит от того, каков уровень архитектуры команд, а также от стоимости и предназначения компьютера. В настоящее время уровень архитектуры команд часто содержит простые команды, которые выполняются за один цикл. В других системах на этом уровне имеются более сложные команды; выполнение одной такой команды занимает несколько циклов.

# Тракт данных.

Тракт данных — это часть центрального процессора, состоящая из АЛУ и его входов и выходов. Тракт данных состоит из регистров (обычно от 1 до 32), АЛУ (арифметико-логического устройства) и нескольких соединяющих шин. Содержимое регистров поступает во входные регистры АЛУ. В них находятся входные данные АЛУ, пока АЛУ производит вычисления.

АЛУ выполняет сложение, вычитание другие простые операции над входными данными и помещает результат в выходной регистр. Этот выходной регистр может помещаться обратно в один из регистров. Он может быть сохранен в памяти, если это необходимо. Отметим, что входные и выходные регистры есть не у всех компьютеров.

Большинство команд можно разделить на две группы: команды типа регистр-память и типа регистр-регистр. Команды первого типа вызывают слова из памяти, помещают их в регистры, где они используются в качестве входных данных АЛУ. Словом может быть, например, целое число. Другие команды этого типа помещают регистры обратно в память.

# Понятие микрокоманды.

**Микрокоманда** – это набор одной, двух или более микроопераций, которые могут быть выполнены одновременно.

**Микрооперация** – это преобразование информации в каком-либо функциональном узле.

Каждая микрокоманда представляет собой **управляющее слово**, которое должно где-то храниться. Ее извлечение потребует какого-то времени, а ее выполнение можно начинать только после ее извлечения.

# Общий обзор уровня архитектуры команд. Свойства уровня команд.

Свойства уровня команд

Уровень команд – это то, каким представляется компьютер программисту машинного языка. Программа уровня архитектуры команд –это то, что выдает компилятор. Чтобы произвести программу уровня команд, составитель компилятора должен знать, какая модель памяти используется в компьютере, какие регистры, типы данных и команды имеются в наличии и т.д.

В связи с этим определением такие вопросы как программируется ли микроархитектура или нет, конвейеризирован компьютер или нет, является ли компьютер суперскалярным не относится к уровню архитектуры команд.

Модели памяти

Большинство машин имеет единое адресное пространство. В некоторых машинах содержатся отдельные адресные пространства для команд и данных.

Один из аспектов модели памяти – семантика памяти. В некоторых машинах микрокоманды переупорядочиваются. Возникает опасность, что память не будет действовать так, как ожидалось (сопроцессор).

Регистры

Во всех компьютерах имеется несколько регистров, которые видны на уровне архитектуры команд. Они нужны для хранения промежуточных результатов, для контроля выполнения программы и для других целей.

Регистры уровня команд можно разделить на две категории: специальные регистры и регистры общего назначения.

# Примеры уровня команд. Типы данных. Форматы команд.

Архитектура набора команд — часть архитектуры компьютера, определяющая программируемую часть ядра микропроцессора. На этом уровне определяются реализованные в микропроцессоре конкретного типа:

* архитектура памяти,

|  |  |
| --- | --- |
| **ОСНОВНЫЕ ТИПЫ ДАННЫХ** | |
| ТИП | БАЙТ |
| ЦЕЛЫЕ ТИПЫ | |
| char | 1 байт |
| int | 4 байта |
| float | 4 байта |
| double | 8 байт |
| bool | 1 байт |
| void | 0 байт |

* взаимодействие с внешними устройствами ввода/вывода,
* режимы адресации,
* регистры,
* машинные команды,
* различные типы внутренних данных (например, с плавающей запятой, целочисленные типы и т. д.),
* обработчики прерываний и исключительных состояний.

**Форматом команды** называется заранее оговоренная структура полей ее кода с разметкой номеров разрядов (бит), определяющих границы отдельных полей команды, или с указанием числа разрядов (бит) в определенных полях, позволяющая ЭВМ распознавать составные части кода.

Важной и сложной проблемой при проектировании ЭВМ является выбор структуры и форматов команды, т.е. ее длины, назначения и размерности отдельных ее полей.

Вместе с тем, для **упрощения аппаратуры и повышения** **быстродействия** ЭВМ длина формата команды должна быть по возможности короче, укладываться в машинное слово или полуслово. Решение проблемы выбора формата команды значительно усложняется в микропроцессорах, работающих с коротким словом.

# Критерии разработки для форматов команд. Примеры форматов команд.

Критерии:

1. Если пропускная способность кэш-памяти команд составляет t бит/с, а средняя длина команды - г бит, то кэш-память способна передавать самое большее t/r команд в секунду. Отметим, что это - верхний предел скорости, с которой процессор может выполнять команды, хотя в настоящее время предпринимаются попытки преодолеть данный барьер. **Ясно, что скорость, с которой могут выполняться команды (то есть быстродействие процессора), может ограничиваться длиной команд. Чем короче команды, тем быстрее работает процессор.** Поскольку современные процессоры способны выполнять несколько команд за один цикл, то вызов нескольких команд за цикл обязателен. Этот аспект применения кэш-памяти команд **делает размер команд важным критерием, который нужно учитывать при разработке.**
2. **Еще один критерий - достаточный объем пространства в формате команды для представления всех требуемых операндов**. Машина, поддерживающая 2п операций и длину команды менее п бит, невозможна. В этом случае в коде операции было бы недостаточно места для того, чтобы указать, какая нужна команда. К тому же история снова и снова доказывает, что обязательно нужно оставлять большое количество свободных кодов операций для будущих дополнений набора команд.
3. **Третий критерий связан с числом битов в адресном поле**. Рассмотрим проект машины с 8-разрядными символами и основной памятью, которая должна содержать 232 символов. Разработчики вольны были приписать последовательные адреса блокам по 8, 16, 24 или 32 бита. Представим, что бы случилось, если бы команда разработчиков разбилась на две воюющие группы, одна из которых утверждает, что основной единицей памяти должен быть 8-разрядный байт, а другая требует, чтобы основной единицей памяти было 32-разрядное слово. Первая группа предложила бы память из 232 байт с номерами 0, 1, 2, 3,4 294 967 295. Вторая группа предложила бы память из 230 слов с номерами 0, 1, 2, 3, 1 073 741 823. Первая группа скажет, что для того, чтобы сравнить два символа при организации по 32-разрядным словам, программе приходится не только вызывать из памяти слова, содержащие эти символы, но и выделять соответствующий символ из каждого слова для сравнения. А это потребует дополнительных команд и, следовательно, дополнительного пространства. 8-разрядная организация, напротив, обеспечивает адресацию каждого символа, что значительно упрощает процедуру сравнения. Сторонники 32-разрядной организации скажут, что их проект требует всего лишь 230 отдельных адресов, что дает длину адреса всего 30 бит, тогда как при 8-разрядной организации требуется целых 32 бита для обращения к той же самой памяти. Если адрес короткий, то и команда будет более короткой. Она займет меньше пространства в памяти, и к тому же для ее вызова потребуется меньше времени. В качестве альтернативы они могут сохранить 32-разрядный адрес для обращения к памяти в 16 Гбайт вместо каких-то там 4 Гбайт. Этот пример демонстрирует, что для получения оптимальной дискретности памяти требуются более длинные адреса и, следовательно, более длинные команды. Одна крайность - это организация памяти, при которой адресуется каждый бит (например, Burroughs В1700). Другая крайность - это память, состоящая из очень длинных слов (например, серия CDC Cyber содержала 60-разрядные слова).

**Примеры форматов команд:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Команды передачи данных**  MOV A, A – передать данные из А в А  MOV A, В – передать данные из В в А | **Арифметические команды**  ADD A – сложить А с А  ADD В – сложить В с А | **Логические команды**  ANA B – логическая операция “and” (B and A)  XRA B – логическая операция “xor” (B xor A) ORA B – логическая операция “or” (B and A)  CMP B – сравнить А с В | **Команды передачи управления**  JMP ADDR – перейти по адресу ADDR  RET – возврат из программы  RST – завершить программу пользователя  HLT – остановить микропроцессор |

# Адресация. Способы адресации.

**Режим адресации** — это метод определения операнда инструкции. Основная функция микропроцессора - выполнение группы инструкций, хранящихся в памяти, для выполнения определенной задачи.

Для операции требуются две вещи - **код операции и операнд**. Например, если мы хотим вычесть два числа x и y, тогда эти два числа будут **операндами**, а минус (-) или знак вычитания - **оператором**.

**Способы адресации**

- **непосредственная адресац**ия, когда адресная часть команды используется непосредственно для размещения операнда, как правило, какой-нибудь константы.

- **прямая регистровая** адресация, когда операнд находится в регистре, номер которого указывается в адресной части команды.

- **неявная адресация**, когда местоположение операнда предопределено кодом операции. Обычно операнд находится в специальном регистре процессора, например, в регистре-аккумуляторе или в вершине стека.

- **прямая адресация,** когда в адресной части команды непосредственно указывают адрес ячейки ОП.

- **страничная адресация**, когда физический адрес ячейки ОП формируется из двух полей: номера страницы (старшие разряды адреса) и номера ячейки в странице (младшие разряды адреса).

- **относительная или базовая адресация**, когда физический адрес получается сложением базового адреса сегмента, сдвинутого влево на несколько разрядов со смещением

- **косвенная адресация**, когда указывается не прямой, а косвенный адрес операнда, т.е. адрес адреса операнда, называемый указателем.

- **косвенная регистровая** адресация, когда указатель адреса операнда хранится не в ОП, а в регистре процессора, номер который указывается в адресной части команды.

- **индексная адресация**, когда физический адрес ячейки ОП формируется сложением базового адреса, указанного в адресной части команды или в базовом регистре и смещения, находящегося в специальном регистре процессора, который называется индексным.

- **индексно - относительная адресация,** когда совместно используются индексная и относительная адресации.

# Сравнение способов адресации.

|  |  |  |  |
| --- | --- | --- | --- |
| **Основа для сравнения** | | **Прямая адресация** | **Косвенная адресация** |
| Базовый | Содержит факт. операнд в адресном поле кода инструкции. | | Поле адреса инструкции содержит адрес операнда. |
| Количество необх. ссылок на память | Два | | Три |
| Количество адресного пространства | Маленький | | Большой |
| Дополнительный расчет | Не требуется | | Это единственный способ выполнить операцию. |
| Скорость | больше | | меньше |

Основное различие между режимом прямой и косвенной адресации заключается в том, что в прямом режиме расположение памяти указывается напрямую. Напротив, в режиме косвенной адресации указывается адрес области основной памяти.