

## **Лекция 1. Поколения компьютеров.**

### **Поколение компьютеров:**

#### **Поколение 0** - Механические компьютеры (1642-1945):

1. Суммирующая машина Паскаля: создана в 1642, реализует сложение и вычитание. Создатель - Блез Паскаль;
2. Арифмометр Лейбница: В 1670х Лейбниц создаёт машину с реализацией сложения и деления;
3. Разностная Машина Бэббиджа:
  - a. 1822 – разностная машина – сложение и вычитание;
  - b. 1834 – аналитическая машина (не закончена): Ввод и вывод данных, 1000 слов по десятичным 50 разрядам, вычислитель.

#### Электривакуумная лампа (Почти как диод):

- 1883 – изобрёл Эдисон;
- 1896 – Ли Де Форест добавил управляющую сетку.

#### **Поколение 1** - Электронные лампы (1945 – 1955):

1. Colossus:
  - a. Разработан в 1943 британцами;
  - b. 1500 ламп, 2500 в Mark II;
  - c. Расшифровывал немецкие сообщения;
  - d. 10 экземпляров, все уничтожены;
  - e. Проект рассекречен в 2000.
2. ENIAC (1945 год):
  - a. 18000 ламп, 1500 реле;
  - b. 167 квадратных метров, 30 тонн, 174 кВт;
  - c. 6000 переключателей и кабелей;
  - d. 5000 операций сложения в секунду.
3. EDSAC (1949 год):
  - a. Первый компьютер с программами в постоянной памяти, двоичной арифметикой, архитектурой Фон-Неймана (Данные и код в одном месте);
  - b. Память – 4096 слов по 40 бит;
  - c. Первая видеоигра – крестики-нулики.
4. IBM (Первые коммерческие компьютеры):
  - a. IBM 701 – 1953 год;
  - b. IBM 704 – 1956 год;
  - c. IBM 709 – 1958 год;

Первый баг: Мотылёк попал в реле.

Память:

1. Память на электронно-лучевых трубках (Энергозависимая);
2. Память на магнитных сердечниках (Энергонезависимая);

Транзисторы:

- Изобретены в 1947 году компанией Bell Labs;
- Тот же принцип работы, что и у лампы;
- Меньше и надёжнее лампы.

**Поколение 2** – транзисторы (1955 – 1965):

1. IBM:
  - a. 1959 – IBM 7090 (709 на транзисторах) – 400 000 команд в секунду, 3 000 000 \$;
  - b. 1960 – IBM 7094 и IBM 1401 – 2500 \$ в месяц.
2. PDP-1 (1960 год):
  - a. Один из первых компьютеров с монитором;
  - b. На нем написали видеоигру Spacewar.
3. 1964 – CDC 6600 от компании Сеймура Крея:
  - a. Многопроцессорный компьютер.
4. DEC PDP-8 (1965 год):
  - a. Первый компьютер с шинной организацией;
  - b. 16 000\$;
  - c. Продано 50 000 штук.

Шинная структура:

- Каждое устройство подключено к другому через шину;

**Поколение 3** – интегральные схемы (1964 – 1980). В 1958 году Джек Килби изобрёл интегральную схему – объединение нескольких элементов в один «чип».

1. EPROM – Erasable Programmable Read Only Memory (Стираемая программируемая память только на чтение);
2. IBM SYSTEM/360 (1964 год):
  - a. Единые архитектура и система команд;
  - b. Разное быстродействие;
  - c. Многозадачность;
  - d. Эмуляция (Поддержка старых программ).

**Поколение 4** – Сверхбольшие интегральные схемы (1980 – по наше время):

1. Apple I (1976 год);
2. Apple II (1977 год);
3. IBM PC (1981 год) – выпускалась с книгой электронных схем за 49\$, что привело к пиратству;
4. Osborne-1 (1981 год) – первый портативный компьютер;
5. Apple Macintosh (1984 год);
6. Intel 80386 (x86) (1985 год):
  - a. Первый из x86 архитектуры;
  - b. Первый 32-разрядный.
7. В 1992 компания DEC представила DEC Alpha – 64-разрядный процессор для персонального компьютера;
8. В 2001 компания IBM представила двухъядерную архитектуру POWER4 – два процессора на одной подложке.

**Поколение 5** – малая мощность, невидимость (1990 – по наше время):

1. 1989 – GridPad – Первый планшетный компьютер;
2. 1993 – Apple MessagePad (Newton) – прадедушка iPad;
3. 1996 – Palm Pilon – первый карманный PDA (Personal Digital Assistant).
4. 2000 – Ericsson R380 – первый смартфон;
5. 2002 – Nokia 7650 – первая Nokia с ОС Symbian;
6. 2007 – Apple iPhone – мультитач, нет клавиатуры;
7. 2008 – HTC Dream – первый андроид;
8. 2014 – Google Glass – носимая электроника.

**Будущее** – квантовые компьютеры (производителей только на некоторых специфичных классах задач):

- 1980 – первые идеи;
- 1998 – компьютер с 2 кубитами;
- 2001 – компьютер с 7 кубитами от IBM;
- 2012 – 84 кубита от D-Wave;
- 2015 – больше 1000 кубит от D-Wave;

## Лекция 2. Типы компьютеров.

### Характеристики компьютеров:

1. Сложность вычислений;
2. Размер;
3. Портативность/автономность;
4. Характеристики аппаратного обеспечения:
  - a. Объём, тип и скорость памяти;
  - b. Архитектура, разрядность и скорость процессора;
  - c. И так далее.
5. ОС/ПО/программы;
6. Интерфейсы Input/Output: USB, PCI, HDMI и так далее;
7. Протоколы взаимодействия: NFC, Bluetooth и так далее;
8. Совместимая периферия;
9. Предназначение: видеоигры, вычисления и так далее;
10. Надёжность:
  - a. Ударостойкость, влагозащищённость;
  - b. Избыточность систем;
  - c. Надёжность хранения данных.
11. Модульность;
12. Форм-фактор:
  - a. Ноутбук;
  - b. Стойка;
  - c. Системный блок;
  - d. Моноблок.
13. Цена.

### Виды компьютеров:

**«Одноразовые компьютеры»** - Устройство с простейшей логикой, выполняет одну задачу. Если сделан с ошибкой или испортился – проще купить новый чем чинить старый:

- Дешёвые – стоят несколько рублей. Покупаются оптом, поэтому разница в стоимости на пару копеек – значительна;
- Маленькие – размером с симку;
- Малое энергопотребление – хватает энергии принятого сигнала;
- Минимум логики / ПО – выполняют одну функцию. Создаются с защитой программой, изменить её нельзя;
- Узкая направленность применения – Создаются под конкретную задачу и применяются только под нее.

Примеры:

- Симка;
- Radio Frequency IDentification (RFID) – Радио-метка, срабатывает от сигнала и посылает ответный. Вклеивают в товары в магазинах. С помощью таких штук можно сделать покупку в магазине без очередей. На каждый товар клеится метка, у пользователя – своя. Пользователь проходить через сканер, все метки считываются, и сумма вычитается со счет пользователя.

**Микроконтроллер** (MCU – microcontroller unit) – чип для управления электроникой.

- Процессор с памятью, интерфейсами ввода-вывода и управления периферией. Прошивка хранится во внутренней памяти МК;
- Микроконтроллеры бывают специализированные – для одной задачи и универсальные – для работы с разной периферией и решением разных задач;
- Дешёвые – от 10 рублей. Разница в пару копеек все еще критична при покупке оптом.
- ПО «прошивается». Можно модифицировать прошивку и залить новую. Это позволяет исправлять ошибки хотя бы на программном уровне;
- Работают в реальном времени. Принимают/обрабатывают данные, генерируют ответные и передают их на другое устройство;
- Малая память – редко более мегабайта. Realtime-устройствам память нужна только для обрабатываемых данных и хранения прошивки;
- Малый вес/размер/энергопотребление;
- Различная разрядность (4/8/16/32);

Применение:

- Бытовые приборы (будильники, стиральные машины, сушильные аппараты, микроволновые печи, охранные сигнализации);
- Мобильные устройства (Беспроводные и сотовые телефоны, факсимильные аппараты, пейджеры);
- Периферийные устройства (принтеры, сканеры, модемы, приводы CDROM);
- Медиа-устройства (Видеомагнитофоны, DVD-плееры, музыкальные центры, MP3-плееры, телеприставки, телевизоры, цифровые

фотокамеры, видеокамеры, объективы, фотокопировальные устройства);

- Медицинское оборудование (рентгеноскопические аппараты, томографы, кардиомониторы, цифровые термометры);
- Торговое оборудование (торговые автоматы, кассовые аппараты);
- Военные комплексы (крылатые ракеты, межконтинентальные баллистические ракеты, торпеды);
- Игрушки (Говорящие куклы, приставки для видеоигр, радиоуправляемые машинки и лодки).

### **Встраиваемые компьютеры:**

- Используются в системах с опасностью физического воздействия. Они должны пережить падение с полки, тряску, вибрации, влажность и перепады давления. Используются на железных дорогах, в авиации, в горнодобывающей промышленности;
- Из всего этого вытекают жесткие требования по весу, габаритам, охлаждению, вибрациям, обслуживанию и т. д. Компьютер должен прожить (хотя бы) свой срок эксплуатации;
- Нет движущихся частей, так как они ломаются от вибрации;
- Бывают на MCU, бывают на CPU;
- Может быть как комплексом из микроконтроллеров, так и сверхзащищённым ПК.

### **Мобильные и мультимедийные:**

- Ограничены по периферии, набору ПО, расширяемости;
- Шире выбор ОС для устройства – некоторые андроиды можно прошить под разные ОС;
- Железо подобрано и адаптировано для обработки графики, долгой работы от одного заряда батареи (у хендхелдов);
- Программы адаптируются под железо. Конфигураций таких устройств несколько, так что программисты могут писать аккуратный низкоуровневый код для каждого из вариантов (В отличие от ПК, конфигураций которых так много, что адаптировать софт под каждый невозможно). Благодаря этому некоторые медиа-устройства работают шустрее, чем ПК с аналогичной конфигурацией;
- Примеры – смартфоны, планшеты, игровые- и теле-приставки.

### **Персональные компьютеры:**

- Компьютеры общего назначения;
- Большой выбор конфигураций;
- Широкий выбор ОС и ПО;
- Производительность зависит от бюджета пользователя.

**Серверы** – системы для расчета и обработки данных по внешним запросам пользователей:

- Мощные компьютеры для бизнеса;
- Дорогие;
- Может быть обычным ПК, а может быть стойкой. Стойки состоят из юнитов. В юнитах те же элементы, что и в обычных компьютерах – память, процессоры, мониторы;
- Используются для обработки больших объемов данных;
- Работают 24/7;
- Могут поддерживать безопасность данных.

**Мейнфрейм** – класс надежных компьютеров для «критически важных систем»

- Специализированное ПО. Например, системы реального времени;
- Надёжность – работают до 15 лет. Мейнфреймы покупаются один раз и надолго;
- Устойчивость к ошибкам (за счёт дублирования компонентов). Неисправные компоненты быстро заменяются, системы восстанавливаются и работают без остановок;
- Память с коррекцией ошибок. Данные дублируются на несколько жестких дисков. Это позволяет восстанавливать данные, если они были повреждены;
- Избыточность.

**Кластеры** – объединение компьютеров для масштабных вычислений:

- Состоят из нескольких компьютеров и воспринимаются как цельная единица;
- Пользователь подает запрос, управляющий компьютер передает его на обработку подходящей машине и потом возвращает результат;
- Используются для массивных вычислений – научных, коммерческих;
- Может быть организован в пределах одного помещения – тогда компьютеры соединяются высокоскоростными шинами;

- Может быть организован по глобальной сети. Такой вариант используется, когда вычисления не срочные.
- Пример – Folding@home – Медицинская программа по расчету свертывания белков. Организована по сети. Приложение для подключения к кластеру установлены на все PS3.

**Интернет вещей (IOT)** – сеть связанных через интернет объектов, способных собирать данные и обмениваться данными, поступающими со встроенных сервисов. Применение: умная инфраструктура, система безопасности, здравоохранение, транспорт, промышленность.

#### Типы компьютеров:

1. Сервисные:
  - a. Одноразовые компьютеры;
  - b. Микроконтроллеры;
  - c. Встраиваемые компьютеры.
2. Личные:
  - a. Мобильные компьютер;
  - b. Мультимедийные компьютеры;
  - c. Персональные компьютеры.
3. Рабочие:
  - a. Серверы;
  - b. Мейнфреймы;
  - c. Кластеры.
4. Интернет вещей.

#### Архитектуры компьютеров:

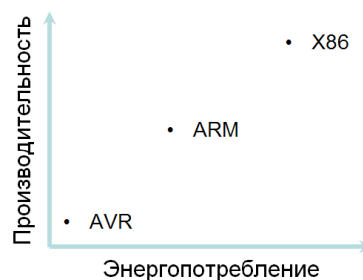
1. **x86**:
  - a. На основе архитектуры CISC – Complex Instruction Set Computing. Это значит, что в системе много команд для процессора. Много команд = ёмкий код, который проще писать, но работает он медленней. Некоторые из команд выполняют несколько операций более низкого уровня (обращение к памяти, простейшая арифметика).
  - b. CPU – Central Processing Unit;
  - c. Память только для обрабатываемых данных;
  - d. С другими компонентами компьютера соединяется через материнскую плату;
  - e. Высокая производительность;
  - f. Высокое энергопотребление;



- g. Основные производители – Intel и AMD;
  - h. ~100% потребительских ПК;
  - i. ~50% серверов и встраиваемых ПК.
2. **ARM** – Advanced RISC Machine:
- a. ARM разработан ARM Limited;
  - b. Чтобы использовать эту архитектуру, нужно купить лицензию;
  - c. На основе RISC – Reduced Instruction Set Computer. Только самые необходимые команды. Работает быстрее, чем CISC. Для ПК это не критично и не заметно, для серверов и мобильных устройств – важно.
  - d. MCU/CPU;
  - e. Высокая энергоэффективность;
  - f. 90% мобильных устройств и периферии;
  - g. Значимые модели: ARMv7, Cortex – A9;
  - h. На основе ARM: NVIDIA Tegra, Qualcomm Snapdragon, Apple A6.
3. **AVR (Alf and Vegard’s RISC или Advanced Virtual RISC)**:
- a. MCU/CPU;
  - b. Используется в низкопроизводительных встроенных системах – пультах, сигнализациях, игрушках, Arduino;
  - c. Имеет «на борту» всё необходимое для систем такого класса – таймеры, часы реального времени, I/O, интерфейс I2C (для связи с другими микроконтроллерами).

Основные семейства архитектуры процессоров компьютеров:

- **X86** – персональные компьютеры, серверы. Архитектура Intel, ядро и основные расширения открыты. Разрабатывают процессоры на основе x86 все желающие.
- **ARM** – встраиваемые и мобильные устройства. Лицензия принадлежит ARM Limited. Разрабатывать на основе этой архитектуре можно только после покупки лицензии.
- **AVR** – MCU, встраиваемые и одноразовые компьютеры. Выпускаются только AVR.



### Лекция 3. Многоуровневая организация ЭВМ.

**Компьютер** – устройство, выполняющее заданные команды. Человек использует компьютер для решения различных задач.

**Программа** – последовательность команд на машинном языке. Программа решает конкретную задачу.

**Машинный язык** – набор команд, выполняемый электронными схемами компьютера.

Базовые команды машинного языка:

- Получить данные из памяти;
- Изменить данные в памяти;
- Записать данные в память.

Машинный язык определяется набором команд. Все команды машинного языка должны поддерживаться машиной, для которой он создан. Чем больше команд – тем проще человеку писать и читать на этом языке. Первые компьютеры программировались напрямую на машинном языке, с помощью проводов и переключателей. Один неправильно подключенный провод, и машина работала не так, как должна.

Проблема машинных языков:

- Если машинный язык содержит **мало** команд – сложно писать большие программы;
- Если машинный язык содержит **много** команд – разработка машины для него сложная и дорогая.

Одно из решений проблем машинных языков – создание многоуровневой организации. Каждый новый уровень повышает уровень абстракции (от физического до пользовательского).

**Многоуровневая организация ЭВМ:**

1. Пользователи;
2. Интерфейс пользователя – Стандартные обслуживающие программы (оболочка, компиляторы и т.д.)
3. Интерфейс библиотечных функций – Стандартная библиотека (open, close, read, write и так далее);
4. Интерфейс системных вызовов – Операционная система (управление процессами, памятью, файловая система и так далее);
5. Аппаратное обеспечение (центральный процессор, память, диски и так далее).

2, 3 – Режим пользователя. 4 – Режим ядра.

Пусть Я(N) – язык N-ного уровня по многоуровневой организации.

**Я(0)** – язык команд электроники компьютера, который сложный для человека;

**Я(1)** – язык, предназначенный для человека.

Я(1) нужно как-то перевести на Я(0). Есть два способа это сделать – **компиляция и интерпретация**.

**Компиляция (трансляция)** – замена каждой команды языка Я(1) эквивалентным набором команд языка Я(0), для последующего выполнения компьютером, выполняется один раз.

**Интерпретация** – программа на языке Я(0) обрабатывает последовательность команд на языке Я(1) и выполняет эквивалентный ей набор команд. Выполняется при каждом запуске программы.

Компиляция и интерпретация рациональны и эффективны, если языки Я(0) и Я(1) близки. Это значит, что Я(1) понятнее Я(0), но не сильно.

**Уровни языков ЭВМ:**

1. Уровень физических устройств – Транзисторы;
2. Я(0) Цифровой логический уровень – Вентили (наборы транзисторов, которые в последствии организуют логические блоки: НЕ, ИЛИ, И, НЕ-И, НЕ-ИЛИ);
3. Я(1) Уровень микроархитектуры – Организация блоков;
4. Я(2) Архитектура набора команд – 0 и 1;
5. Я(3) Уровень ОС – Системные вызовы и часть команд предыдущего уровня. Системные вызовы – обращение к предопределённой процедуре ОС;
6. Я(4) Уровень ассемблера – Язык ассемблера. Ассемблер – сборщик, который переводит язык ассемблера в машинный код. Для каждой архитектуры свой ассемблер. Оператор языка ассемблера соответствует команде машинного языка. Работать с ним нужно, если критически нужна производительность или нужен полный доступ к аппаратному обеспечению;
7. Я(5) Языки высокого уровня – C++, Java, Python и так далее. Высокий уровень абстракции – работают с переменными, файлами, потоками. Платформонезависимы – должны работать на любой платформе, где есть компилятор или интерпретатор. Компилируются или интерпретируются на 4 или 3 уровень.

Вывод: Компьютер проектируется как иерархический набор уровней, где верхние управляют нижними. Набор команд, типов данных и прочих характеристик называется архитектурой уровня. Любой уровень можно реализовать как виртуально, так и физически.

### **Виртуальная машина**

Пусть существует гипотетический компьютер, работающий с языком Я(N).

Назовём его виртуальной машиной М(N).

Фактически, виртуальная машина – это интерпретация: программы на языке Я(N) интерпретируются программой-интерпретатором, работающей на машине более низкого уровня, либо транслируются на язык машины более низкого уровня (при этом программы на Я(0) выполняются непосредственно электронными схемами)

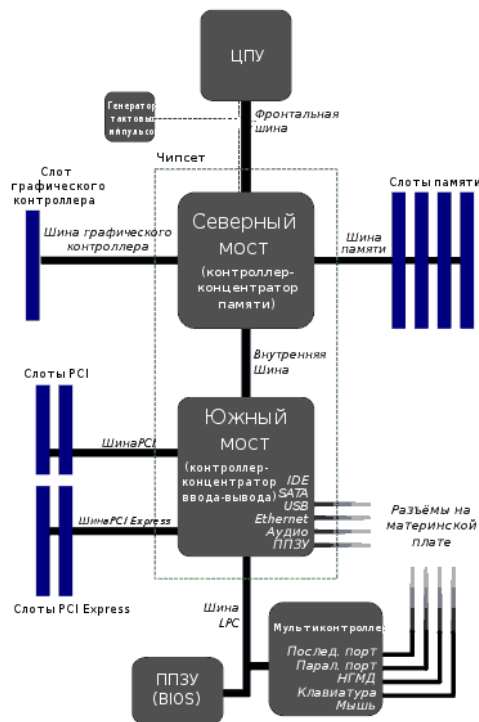
## Лекция 4. Чипсет и периферия.

**Периферийные устройства** – аппаратура ввода/вывода информации; не обязательны для работы системы.

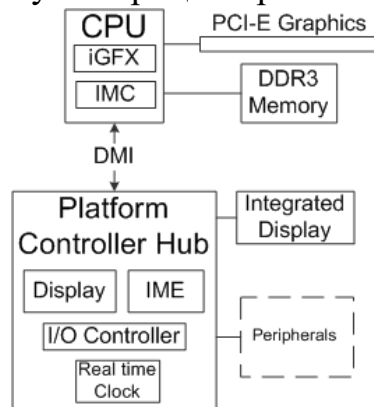
- Устройства ввода/вывода
- Устройства хранения информации

**Чипсет** – часть мат. платы, с помощью которой связываются ЦП и адаптеры других устройств.

- Северный мост – обеспечивает взаимодействие высокоскоростных компонентов (ЦП, ОП и GPU). Содержит контролер памяти для быстрого доступа к ОП. Соединен с PCI Express (видеокарты, редко сопроцессор или сетевые карты);
- Южный мост – соединяет периферию и низкоскоростные устройства с северным мостом. Контролирует ввод/вывод (ЖД, USB, сетевые карты и пр.). Подключаются шины такие как PCI, USB, SATA, IDE.



Иногда вместо юж. моста ставят Platform Controller Hub (PCH). Контроллеры видеокарты и памяти перенесены на ЦП, все остальные контроллеры вынесены в PCH. В таком случае процессор сам является северным мостом.



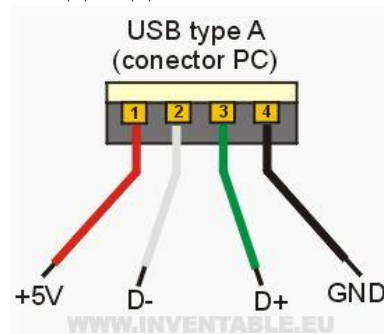
**BIOS** (Basic I/O System) – низкоуровневое системное ПО, запускаемое при загрузке ПК.

- Инициализация компонентов ПК
- Запуск ОС
- Обновление микрокода процессора (прошивка)

**UEFI** (Unified Extensible Firmware Interface) – логическое развитие BIOS'а: гибко настраиваемый, поддерживает больше памяти, новое оборудование и сеть на этапе загрузки системы (удаленный запуск и контроль системы).

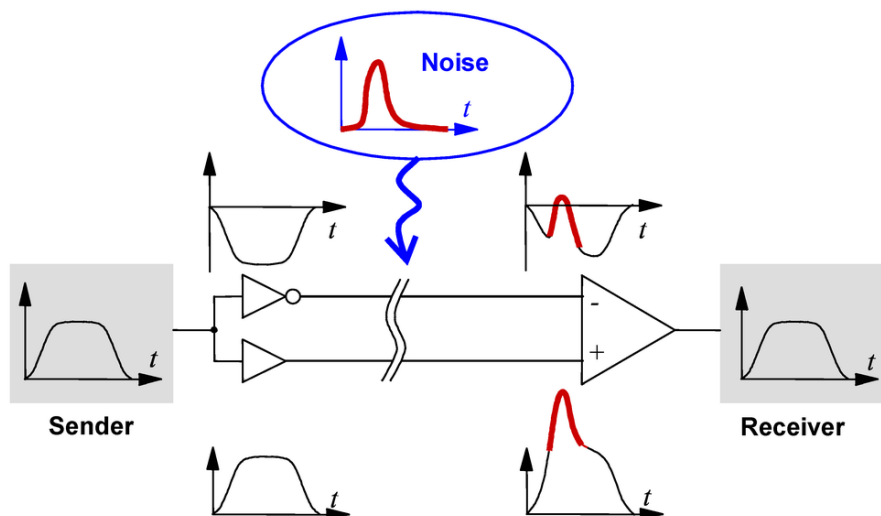
**PCI Express** (Peripheral Component Interconnect) – интерфейс, который позволяет устанавливать высокопроизводительные модули для вычислений/памяти (видеокарты). Поддерживает горячую замену и контроль целостности данных.

**USB** (Universal Serial Bus) – «универсальная последовательная шина». Два контакта по бокам используются для питания - подача напряжения и заземление, два по середине – для данных.



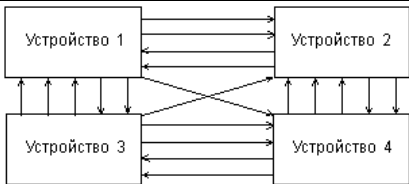
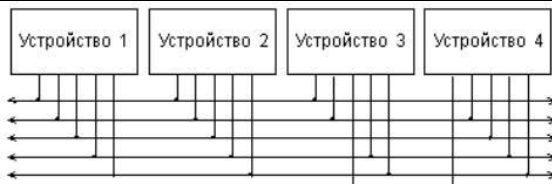
Цель дублирования сигнала - уменьшить шумы при передаче:

По D+ передается исходный сигнал, по D- сигнал, умноженный на (-1). По пути они получают шумы, которые попадают в противофазу. На приемнике D- опять перемножается на (-1), потом D+ и D- складываются и сумма делится пополам. Т.о. шумы «схлопываются» и в итоге получаем чистый сигнал.

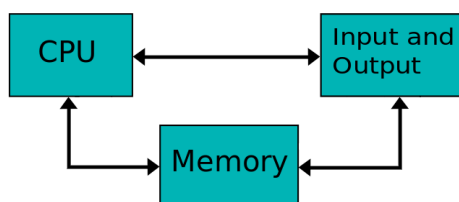


## Лекция 5. Шина. Характеристики шины.

### Связь компонентов компьютера

Классическая структура связи (устройства связаны между собой напрямую)	Общая шина (магистраль)
- есть набор коннекторов для связи с другими устройствами	- делят одни и те же провода для передачи данных
- передача данных по одному каналу не влияет на передачу по другим каналам	- в каждый момент времени данные могут передаваться только между двумя устройствами
- асинхронность	- последовательная передача данных
- быстро	- медленно
	

### Классическая структура связей:

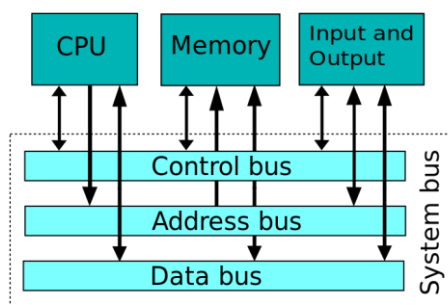


Все сигналы и коды между устройствами передаются по отдельным линиям связи, все они при этом независимы.

Плюсы	Минусы
Асинхронность	Множество связей
Параллельная передача данных	Сложность связей
Высокая скорость	Различные протоколы обмена информацией

**Шина** – набор параллельно связанных проводов для передачи адресов, данных и управляющих сигналов.

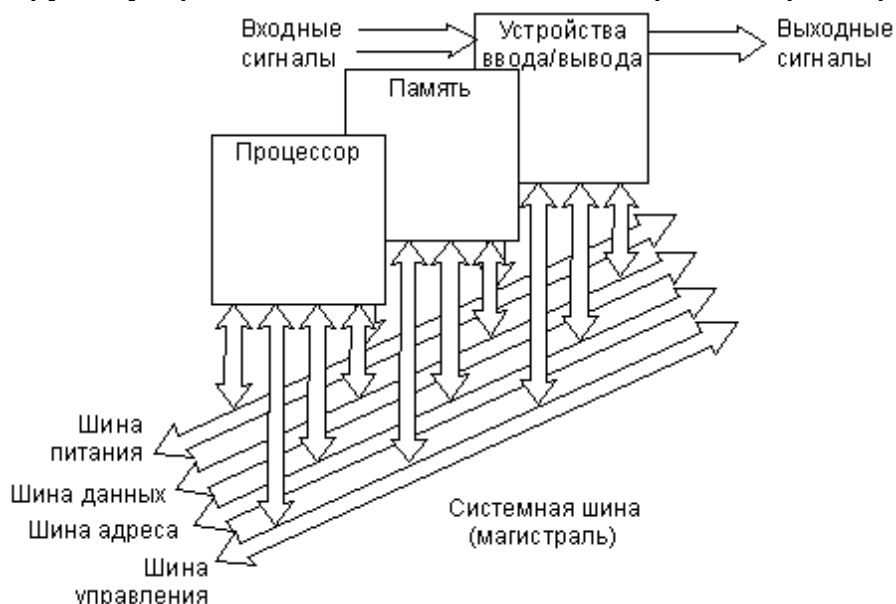
### Шинная структура связей:



Все устройства сидят на одной «системной шине», которая объединяет шину управления, шину адреса и шину данных.

Плюсы	Минусы
Единая спецификация - «протокол шины»	Последовательная передача (только одно устройство)
Единый протокол обмена данными	Неисправность одного устройства может вывести из строя всю систему
Унификация устройств	Сложность отладки (данные от всех устройств)
Удешевление производства	Необходимость арбитража («кто в данный момент использует шину»)

**Системная шина (магистраль)** – шина, по которому процессор соединен с другими устройствами компьютера. К шине напрямую подключен только процессор, другие устройства подключены к ней через контроллеры.



- Шина питания – 5V
- По шине управления (контроля) передается команда, которая считывается всеми устройствами и выполняется только теми, для которых она предназначена.
- Если для выполнения команды нужны какие-либо адреса («в какой адрес памяти записать»), то они передаются по шине адреса.
- Шина данных предназначена для передачи данных между устройствами.

**Задающее устройство** – устройство, которое подает сигнал на шину контроля.

Задающее устройство	Подчиненное устройство	Пример
ЦП	Память	Вызов команд и данных
ЦП	Устройство ввода-вывода	Инициализация передачи данных



ЦП	Сопроцессор (GPU)	Передача команды от процессора к сопроцессору
Устройство ввода-вывода	Память	Прямой доступ к памяти
Сопроцессор	ЦП	Вызов сопроцессором операндов из ЦП

Передача данных может происходить:

- Между процессором и другим устройством (память/периферия):
- Между периферийным устройством и памятью (DMA – Direct Memory Access)

Процесс передачи данных:

- На шине адреса устанавливается адрес устройства, с которым будет обмениваться данными процессор
- На шине управления устанавливается характер обмена (чтение/запись/инициализация/сброс и т.д.)
- Передача данных по шине данных

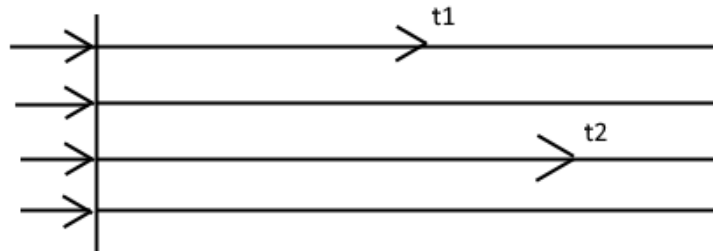
Область применения:

- Внутренние шины (северный/южный мост, PCI Express)
- Внешние шины (USB, HDMI, DisplayPort, COM)



(далее просто идет перечисление характеристик, схемы выше будет достаточно)

- Ширина шины – количество линий передачи (важна для шины адреса, шины данных)
- Мультиплексирование – объединение шины адреса и данных. Передача адреса и данных разделена по времени.
  - Бóльший объём адресуемой памяти
  - Сокращение количества проводников
  - Снижение стоимости
  - Снижение скорости
- Направленность
  - Однонаправленная: задающим устройством всегда является ЦП
  - Двухнаправленная: задающим устройством может быть не только процессор
- Параллельные шины:
  - Заданная ширина N
  - За один такт передаётся N бит
  - + большая теоретическая скорость
  - - цена, сложность, расфазировка между линиями (clock skew – ситуация, когда данные начинают одновременно передаваться по нескольким проводам, но в конечную точку приходят в разное время, скорость будет ограничена самой медленной линией)

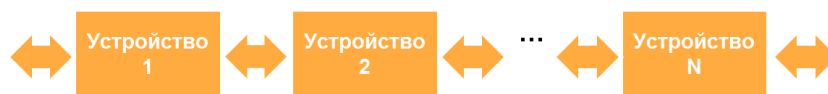


- Последовательные шины:
  - Ширина всегда 1 бит
  - За один такт всегда передаётся 1 бит
  - + дешевле, проще, нет расфазировки между линиями
- Топология шины:

Параллельная топология (multidrop):



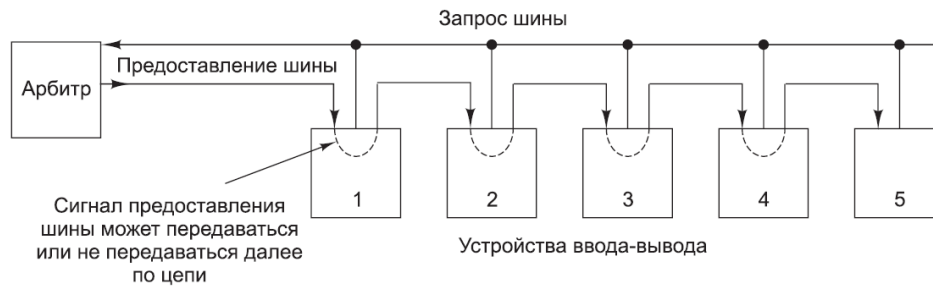
Цепная топология (daisy chain):



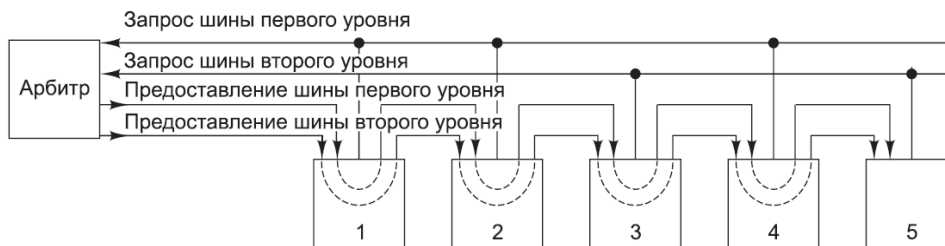
- Время цикла – время, за которое должно выполняться чтение/запись на шине. Задаётся в тактах.



Если в данный момент шина не используется, то он кидает байт на канал предоставления шины, проходящий по цепочке от первого подключённого устройства к последнему.

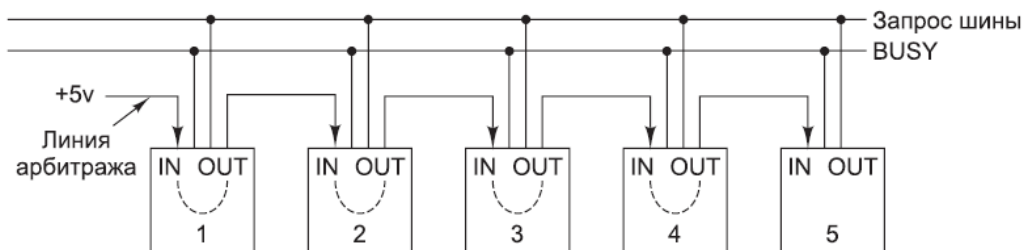


- Арбитраж с приоритетом:



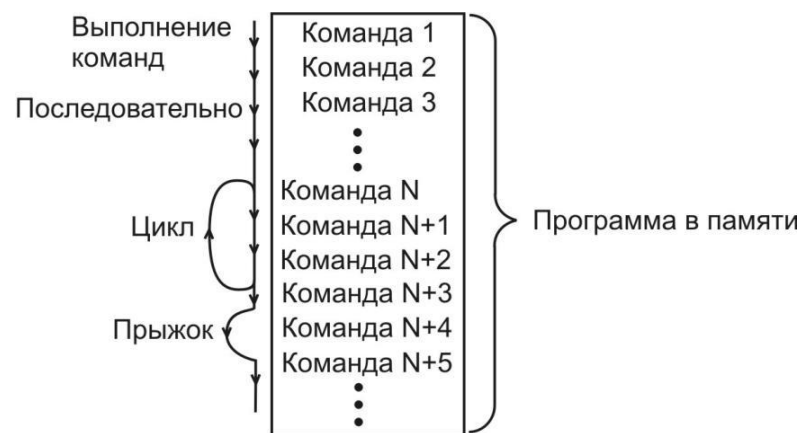
- Децентрализованный арбитраж – арбитраж без отдельного устройства-арбитра.

Принцип работы: Если устройство работает с шиной, то оно выставляет байт в канал BUSY. Если какому-то устройству нужна шина, то подается байт на линию запроса шины и дальше каждое устройство по линии арбитража в порядке своего приоритета проверяет, нужна ли в данный момент шина или нет.

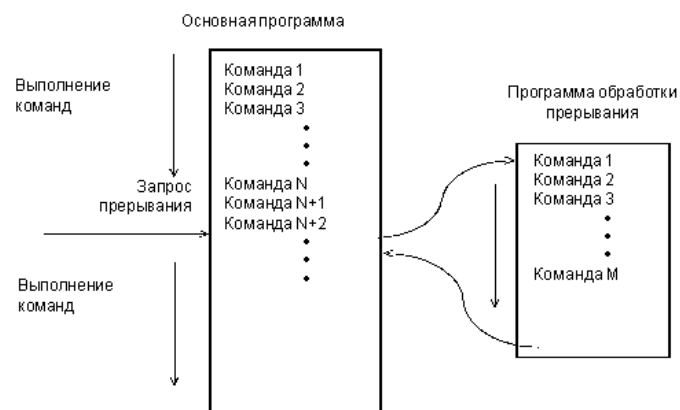


### Режимы обмена данными по системной шине:

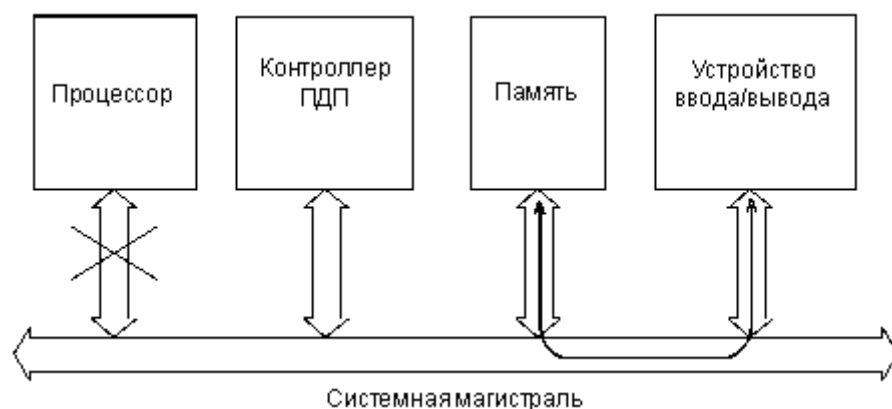
- Программный обмен информацией (процессор инициирует обмен с другими устройствами)



- Обмен с использованием прерываний (устройство прерывает исполнение кода на процессоре и просит срочно что-либо обработать)



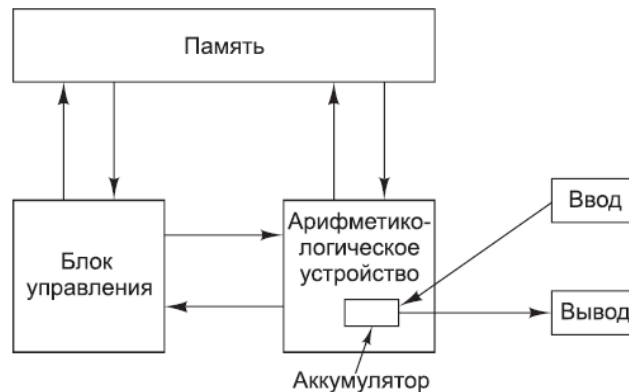
- Прямой доступ к памяти (устройство общается с памятью без участия процессора)



## Лекция 6. Архитектуры процессоров.

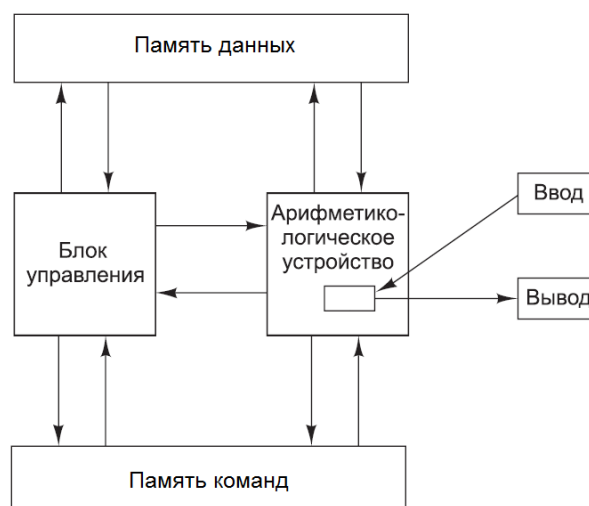
### Классические архитектуры

- Архитектура фон Неймана



- **Однородность памяти: команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы.**
- Адресность: структурно основная память состоит из пронумерованных ячеек, причем процессору в произвольный момент доступна любая ячейка, но только одна.
- Программное управление: все вычисления, предусмотренные алгоритмом решения задачи, должны быть представлены в виде программы, состоящей из последовательности управляющих слов — команд.
- Двоичное кодирование: вся информация, как данные, так и команды, кодируются цифрами 1 и 0.

- Гарвардская архитектура



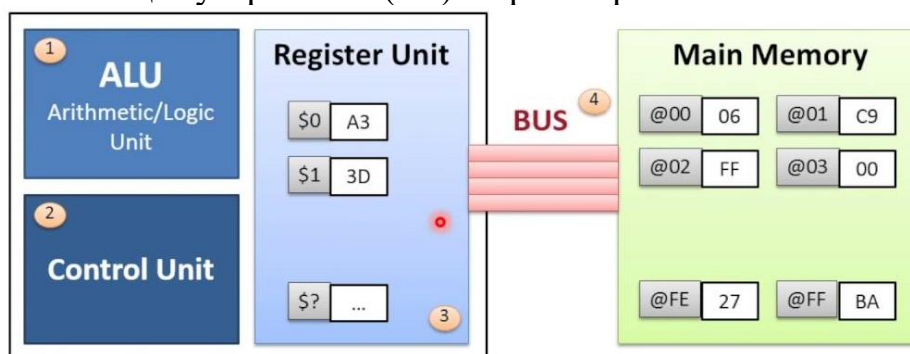
- **Неоднородность памяти: команды и данные хранятся отдельно, с асинхронным доступом к ним.**
- Адресность.
- Программное управление.
- Двоичное кодирование.

### Основное различие:

- В архитектуре Фон-Неймана команды и данные хранятся в одной и той же памяти. Команды и данные считаются одним и тем же и хранятся в одном и том же блоке памяти. Процессор в один момент может выбирать или команды, или данные, но не одновременно.
- В Гарвардской архитектуре память данных и память команд разделена. Поэтому процессор может одновременно(условно) работать и с одним, и с другим. Так выходит быстрее.

### Компоненты процессора:

- Устройства ввода-вывода (УВВ) — передают данные с/на шину.
- Устройство управления (УУ) — выбирает команды для выполнения с УВВ, передает арифметику и работу с данными на остальные компоненты.
- Арифметико-логическое устройство (АЛУ) — блок для выполнения простых арифметических и логических преобразований над данными. Входные данные — операнды.
- Запоминающее устройство (ЗУ) — регистры и кэш-память.



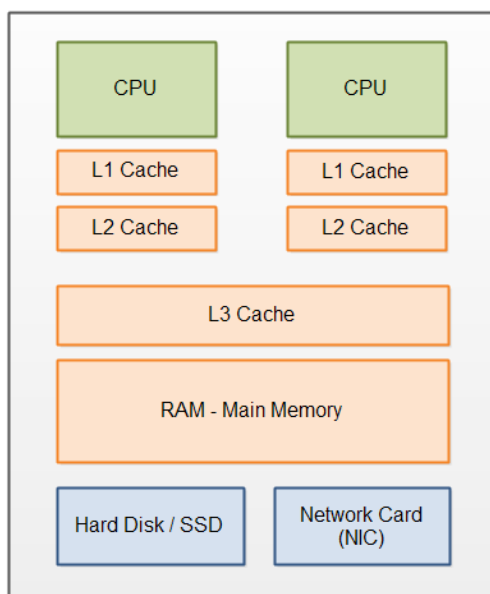
\*\*\*

- FPU (floating point unit) — блок выполнения операций с плавающей запятой.
- AGU (address generation unit) — блок вычисления адресов в памяти.

**Регистры** — сверхбыстрая память внутри процессора. Очень быстрые, но дорогие, и их мало. Используются для хранения промежуточных данных.

**Кэш-память** — очень быстрая память небольшого размера, используется для уменьшения среднего времени доступа к ОП. Чуть медленнее, чем регистры, их больше, но всё равно недостаточно, поэтому используют несколько уровней кэш-памяти.

Уровни кэша: L1, L2 (ядро процессора), L3 (общий на все ядра)



### Архитектуры CISC и RISC:

- CISC (Complex Instruction Set Computer)
  - Много команд на все случаи жизни.
  - Команды для сложных операций на несколько тактов.
  - Из минусов: большое количество сложных команд отрицательно влияет на производительность.
- RISC (Reduced Instruction Set Computer)
  - Минимальный набор простых команд.
  - Высокая скорость выполнения.
  - Из минусов: большое количество запускаемых команд (1 команда CISC  $\approx$  4-5 команд RISC).

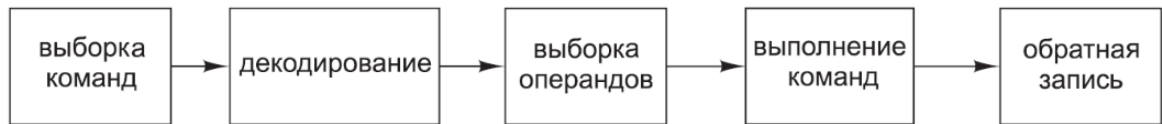
#### Принципы RISC:

- Все команды выполняются непосредственно аппаратным обеспечением.
- Запуск как можно большего количества команд в секунду.
- Команды должны легко декодироваться.
- К памяти должны обращаться только команды загрузки и сохранения.
- Много регистров.



- Гибридная архитектура - «поздние» x86 процессоры (Intel Pentium 4, AMD Athlon) – CISC-совместимы, но являются процессорами с RISC-ядром. CISC-инструкции преобразуются в набор внутренних RISC-команд.

Алгоритм работы процессора Pipeline (примитивный):

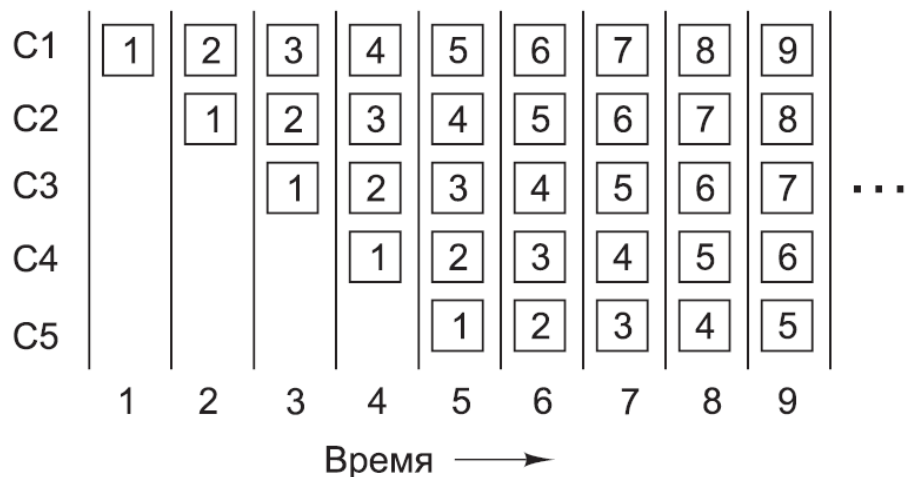


В процессорах может быть реализован параллелизм:

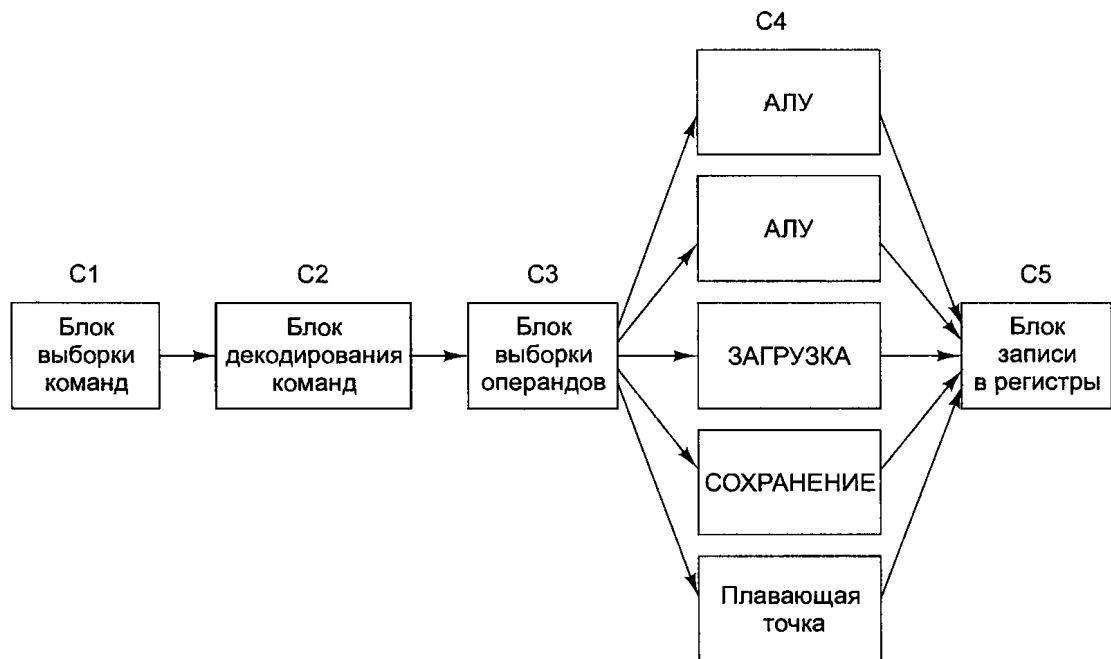


### Параллелизм команд

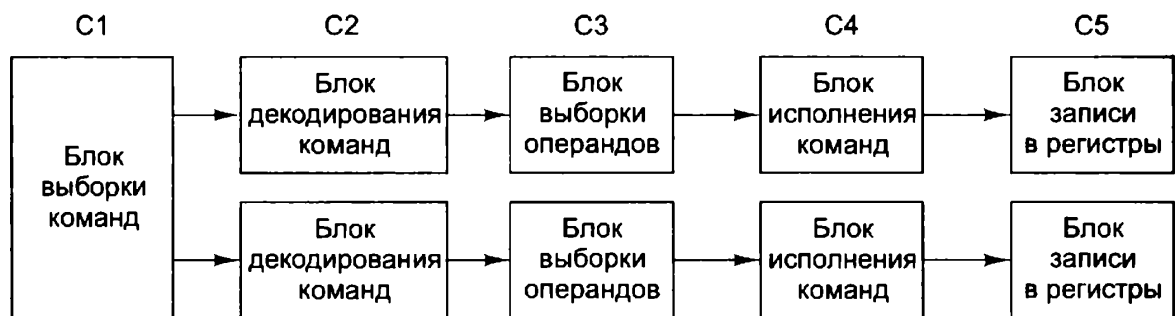
Конвейерная архитектура – в каждый момент времени процессор работает над различными стадиями выполнения нескольких команд, причем управляющее устройство состоит из нескольких блоков, которые занимаются одной стадией задачи.



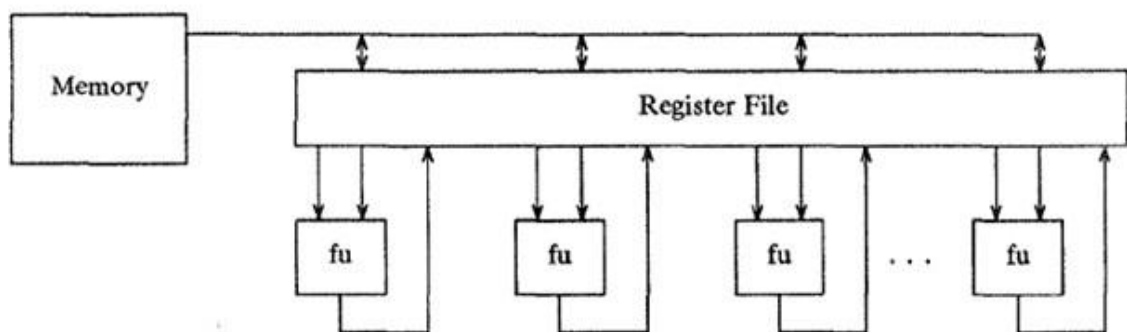
Суперскалярная архитектура – в процессоре несколько блоков выполнения команд, несколько инструкций выполняются параллельно за один такт.



Можно также реализовать несколько конвейеров для большего параллелизма:



Архитектура VLIW (very long instruction word) – архитектура с несколькими АЛУ. Одна команда в таком случае содержит несколько операций, которые должны выполняться параллельно (к примеру, add «128 чисел»).



## Таксономия Флинна

	Одиночный поток команд (Single Instruction)	Множество потоков команд (Multiple Instruction)
Одиночный поток данных (Single Data)	SISD	MISD
Множество потоков данных (Multiple Data)	SIMD	MIMD

- SISD (ОКОД) – традиционная ф-Н машина без параллелизма
- SIMD (ОКМД) – одни команды на всех процессорах, разные данные (векторные компьютеры, способные оперировать векторами данных)
- MISD (МКОД) – одни данные, разные команды для каждого процессора (спорный класс, примеров реальных систем нет)
- MIMD (МКМД) – команды и данные для всех процессоров разные (многоядерные и многопоточные процессоры, мультипроцессорные машины, труп параллелизм)

## Параллелизм процессоров

Мультипроцессоры – система из нескольких параллельных процессоров, имеющих общую память. Такие процессоры называются «сильно связанными».

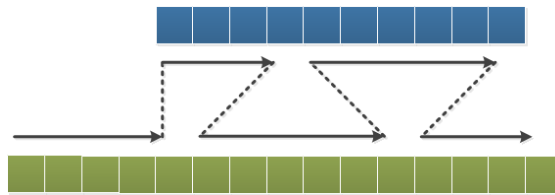
Мультикомпьютеры (кластеры) – системы из большого числа взаимосвязанных компьютеров, у каждого из которых есть собственная память.

**Многопоточность** - способность процессора выполнять одновременно несколько потоков выполнения.

Поток выполнения — наименьший набор команд, который может выполняться (планироваться) независимо от других.

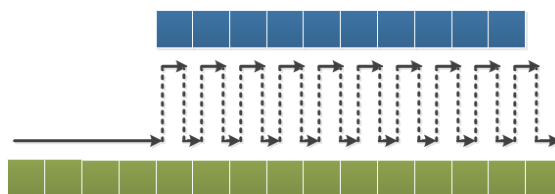
- Временная многопоточность  
В каждый момент времени выполняется только одна задача, но процессор может быстро переключаться между ними.
  - Крупнозернистая (Coarse-grained)  
В конвейере исполняется только один поток некоторый промежуток времени. Задача переключается, если поток встал (например, если требуются данные из памяти или если задано

ограничение на количество тактов). Т.о. конвейер процессора не простаивает и почти всегда занят исполнением того или иного потока.



- Мелкозернистая (Fine-grained)

Процессор переключается между потоками при каждом такте. Т.о. гарантируется исполнение всех потоков, приписанных к процессору. Выполнение конкретного потока замедляется, но общая пропускная способность процессора увеличивается.



- Одновременная многопоточность

Используется в процессорах с суперскалярной архитектурой. На этапе выполнения команды могут выполняться параллельно на разных потоках. Может происходить, только если несколько ядер на процессоре, или несколько процессоров, или несколько машин.

## Лекция 7. Предсказатель переходов.

Условный переход – любое изменение последовательности выполнения кода.

- Условные переходы вперед (if, break, switch)
- Условные переходы назад (do, while, for)

Условные переходы «ломают» конвейер процессора, это неэффективно.

**Предсказатель переходов** нужен для спекулятивного выполнения, т.е. выполнения команд до того, как команда станет нам нужной (может и никогда не стать нужной). Это нужно для того, чтобы процессор заранее что-то считал и не простаивал без дела. Если он угадал, то хорошо, если нет, то проигрываем не так много.

(Предсказатель переходов  $\Leftrightarrow$  Блок выборки команд, выбирает в каком порядке передавать команды на исполнение)

**Важно:** спекулятивно выполняются только команды, результат которых можно отменить.

Если спекулятивно выполненная команда вызвала ошибку, устанавливается **ядовитый бит (poison bit)**. Если далее результат выполнения команды используется, проверяется ядовитый бит и выбрасывается исключение. А если нет, то просто сбрасывается.

Пример:

```
x = 0;
```

```
if (x > 0)
```

```
    z = y / x;
```

(Если 3 строка выполнится спекулятивно, до блока if, деление на ноль вызовет установку ядовитого бита.)

**Методы работы предсказателя:**

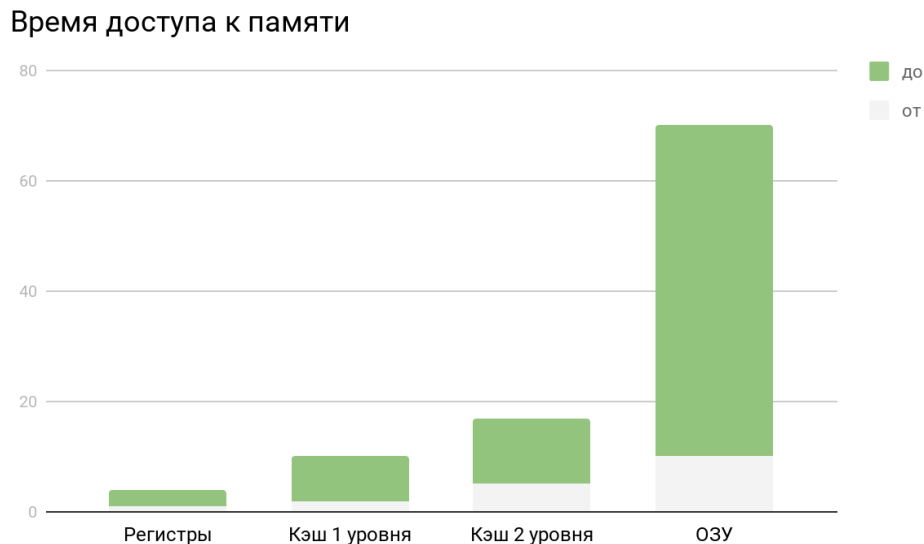
- Статический метод – предполагает, что условный переход всегда выполняется или всегда не выполняется.
- Счётчик – собирает историю о прошлых переходах (выполнился ли тот или иной переход или нет). Учитывает только последний условный переход.
- Счётчик с насыщением и адаптивный двухуровневый предсказатель (или вместе предсказатель переходов с учётом истории) – в отличие от обычного счётчика учитывают полную историю условных переходов, пока хватает памяти.

- Предсказатель цикла – использует счётчик цикла для отсчёта количества переходов. Мы точно знаем, что цикл будет крутиться, пока не перевалит за указанное значение.
- Предсказатель адреса возврата из функции – используется таблица с адресами возврата. Мы точно знаем, куда мы прыгнем после окончания функции, и это можно предсказать.

## Лекция 8. Кэш процессора.

- Регистры процессора очень быстрые, но дорогие.
- Кэш достаточно быстрый, и не такой дорогой.
- ОП медленная в сравнении с кэшем и регистрами.

Поэтому всё, что может понадобиться из ОП, перекладываем в кэш.



### Локализация данных:

- Пространственная локализация – в случае обращения к ячейке памяти, с большой вероятностью можно ожидать обращения к близлежащим ячейкам. Мы храним то, что нам может в скором времени понадобиться, с точки зрения пространства, если мы близко подошли к какой-то памяти.
- Временная локализация – к ячейкам памяти, к которым недавно производилось обращение, с большой вероятностью будет обращение в ближайшем будущем. Т.е. если мы недавно использовали какие-то данные, т.е. большая вероятность, что мы их будем использовать ещё раз.

**Попадание кэша** – это когда мы находим в кэше нужные данные.

**Промах кэша** – когда нужных данных ещё нет и их нужно подгрузить.

### Типы промахов:

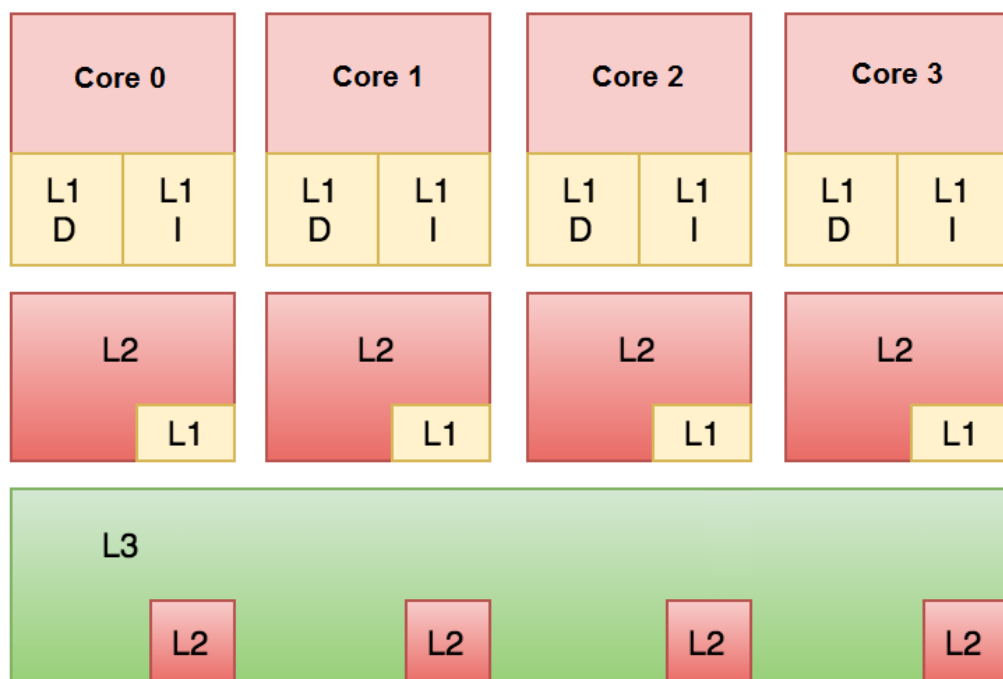
- Промах по чтению кэша инструкций (запрос ещё не закэшированной инструкции) — большая задержка, простой конвейера.

- Промех по чтению кэша данных (запрос ещё не закэшированных данных) — небольшая задержка, не даёт простоя конвейера.
- Промех по записи кэша (хотим положить что-то в кэш, но пока что кэш недоступен) — наименьшая задержка, может быть поставлена в очередь.

#### Алгоритмы вытеснения из кэша:

- LRU (Least Recently Used) – вытеснение строк, к которым давно не было обращения.
- LRR (Least Recently Replaced) – вытеснение наиболее старых строк.
- LFU (Least Frequently Used) – вытеснение наименее часто запрашиваемых строк.
- Random - вытеснение рандомных строк (самое не эффективное).

У процессора есть несколько уровней кэша. Обычно три: L1, L2, L3. Некоторые уровни кэша могут быть общими сразу для нескольких ядер процессора.





## Лекция 9. Память в компьютере.

**Бит** – минимальная единица хранения информации в современных компьютерах. На основе битов организуются системы счисления:

- двоичная
- двоично-десятичная (данные в виде десятичных цифр, записанных в двоичном виде, под каждый разряд 4 бита)
- код Грея (каждый разряд отличается от предыдущего изменением одного бита)

Биты хранятся в ячейках памяти размером 8 до 16 бит в среднем. Раньше размер ячейки и машинного слова мог быть любым. В современных машинах существует стандарт:

- 1 ячейка = 8 бит = 1 байт
- Машинное слово - набор из N байт
- N - зависит от архитектуры (x86: N = 4; x64: N = 8)

Big Endian (подобен привычному порядку записи «слева-направо»):

| 7 ... 0 | 7 ... 0 |  
15    8   7    0

Little Endian (обратный привычному порядку записи чисел; x86):

| 7 ... 0 | 7 ... 0 |  
7    0 15    8

Разные порядки следования байтов часто являются проблемой при передачи данных между машинами. В x86 архитектуре используется Little Endian.



В памяти могут возникать ошибки (особенно в ОП). Необходимы способы обнаружения и исправления ошибок, один из таких способов – **код исправления ошибок**:

**Кодовое слово** – слово, состоящее из  $m$  бит данных и из  $r$  бит контрольных разрядов. Общая длина:  $n = m + r$ .

Контрольные разряды помогают поддерживать целостность данных.

- Бит чётности.  
Обнаружение одиночных ошибок. Один контрольный разряд ( $r = 1$ ).  
Бит чётности равен сумме битов по модулю 2. Позволяет только обнаруживать ошибки, но не исправлять их.
- Код Хэмминга.  
Позволяет обнаруживать и исправлять одиночные ошибки.  
Количество контрольных бит:  $2^r \geq r + m + 1$

Виды памяти:

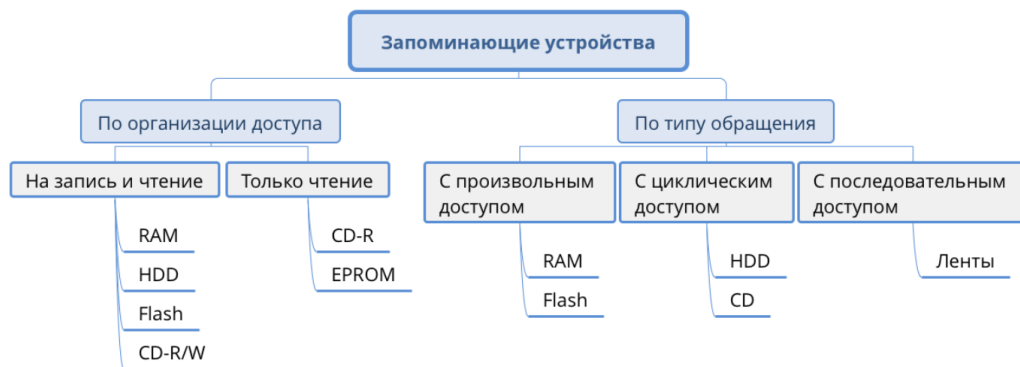
В **оперативной памяти** храним всё, что нам нужно прямо сейчас. Является энергозависимой памятью (как только отключаем питание, то сразу теряем все данные).

- ОЗУ - оперативное запоминающее устройство.
- RAM - Random Access Memory (память с произвольным доступом).
- ECC - Error-Correcting code memory (память с коррекцией ошибок).

**Постоянная память.** Является энергонезависимой памятью.

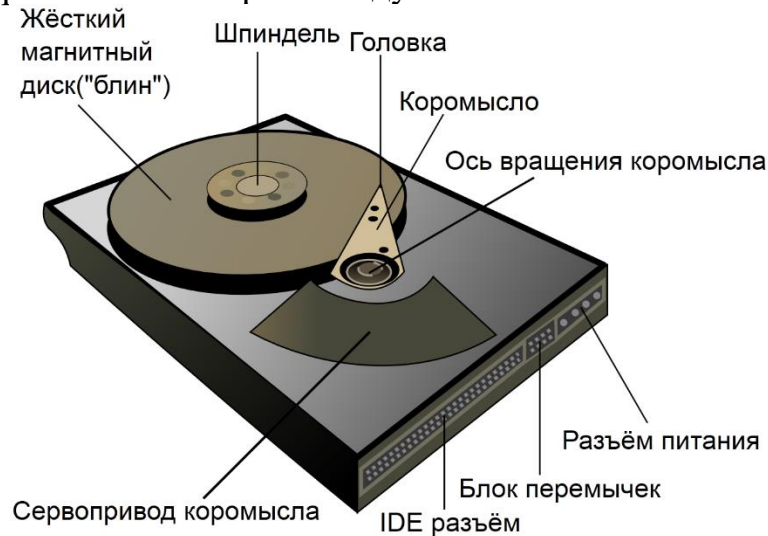
- ПЗУ - постоянное запоминающее устройство.
- ROM - read-only memory, память только для чтения.

## Лекция 10. Запоминающие устройства.



### Жёсткий диск (HDD).

У жёсткого диска есть блины, на которых записывается информация и считывающая головка, которая считывает с них информацию. Блины вращаются, а головка движется по ним и считывает ячейки. У пластин (блинов) обычно две поверхности (верхняя и нижняя). На каждой поверхности ставится головка для чтения и записи. Сами пластины (блины) делятся на дорожки и сектора. Внутри сектора есть своя разметка. Сначала идёт преамбула с метаинформацией, потом данные и в конце код исправления ошибок (ECC). Так же между секторами есть межсекторный интервал для различия секторов между собой.



### Адресация жёстких дисков.

**Сектор** - минимально адресуемая единица.

- CHS (Cylinder, Head, Sector) - сектор адресуется тремя координатами: цилиндр, головка, сектор.
- LBA (Logical Block Addressing) - сектор имеет номер (целое число, с нуля).

Возможен перевод из CHS в LBA и обратно.

**RAID** массив (Redundant Array of Inexpensive/Independent Disks) – избыточный массив недорогих/независимых дисков. RAID даёт нам или производительность, или отказоустойчивость, или всё вместе.

- JBOD (Just a Bunch of Disks) - самый примитивный RAID. Просто набор нескольких жёстких дисков.
- RAID 0 - чередование полос.  
Пишем данные на каждый диск по очереди. Если один диск откажет, то пропадает половина данных. Даёт небольшое увеличение в скорости. Пока на один диск записываются данные, мы можем работать с другим.
- RAID 1 - зеркалирование полос.  
Пишем одни и те же данные на оба диска. Даёт отказоустойчивость. Если один жёсткий диск ломается, то те же данные всё ещё есть на втором диске.
- RAID 2 - RAID с кодом Хэмминга.  
Целые отдельные диски выделяются под коды исправления ошибок, которые могут позволить нам восстановить данные с сломанного диска.
- RAID 1+0 - комбинирование.  
Можем скомбинировать RAID 0 и RAID 1, тогда получим и чередование полос, и дублирование данных. Поэтому система будет иметь повышенную скорость и отказоустойчивость.

**Flash-память** - энергозависимое полупроводниковое запоминающее устройство с произвольным доступом. Производителю жёстких дисков, но менее надёжная. Есть несколько технологий, определяемых вместимостью одной ячейки:

- SLC - one bit per cell. Один бит на одну ячейку даёт возможность хранить 2 значения. Самая надёжная, быстрая, но самая дорогая.
- MLC - two bits per cell. Два бита на ячейку, 4 возможных значения. Менее надёжная, средняя скорости и по стоимости.
- TLC - three bits per cell. Три бита на ячейку, 8 возможных значений. Вообще не надёжная, медленная, но дешёвая.

Внутри Flash-памяти возможно сжатие данных. Так же контроллер резервирует себе часть памяти под запасные ячейки, либо биты чётности для исправления ошибок.

Так как Flash-память быстро изнашивается, существует выравнивание износа. Если мы будем постоянно записывать всё в одни и те же ячейки, то они быстро умрут. Поэтому есть специальные алгоритмы, которые позволяют равномерно записывать данные во Flash-память.

### **Твердотельный накопитель (SSD).**

Немеханическое постоянное запоминающее устройство на основе Flash-памяти или DRAM.

#### **Отличия SSD от HDD:**

<b>Плюсы</b>	<b>Минусы</b>
Полное отсутствие шума	Ограниченное количество циклов перезаписи
Низкое энергопотребление	Высокая цена
Малые габариты и вес	Сложность восстановления информации после сбоя
Широкий диапазон рабочих температур	Боятся радиации
Меньшая чувствительность к внешним влияниям	
Стабильная скорость чтения	
Высокая скорость чтения/записи	

По итогу можно сказать, что SSD быстрее по скорости чтения и записи, а HDD более надёжный.

## Лекция 11. Операционные режимы работы процессора.

Режимы работы процессора:

- Реальный режим
- Защищённый режим
- Режим виртуального i8086
- Режим системного управления

### 1. Реальный режим

- После подачи питания процессор переходит в реальный режим
- 1 мб памяти
- Единственный режим работы до модели i80286
- Нет многозадачности
- Нет защиты памяти (полный доступ ПО к памяти)
- В этом режиме работают загрузчик ОС/BIOS

В реальном режиме существует проблема адресации памяти, когда значения регистра не покрывают все возможные адреса (например, если 16-разрядные регистры и 20-разрядная шина).

Поэтому в данном режиме используется сегментная адресация памяти:

- Память делится на сегменты
- Адресация внутри сегмента — смещение

⇒ получается логический адрес, который занимает два регистра:  
<сегмент:смещение>

[ Физический адрес ] =

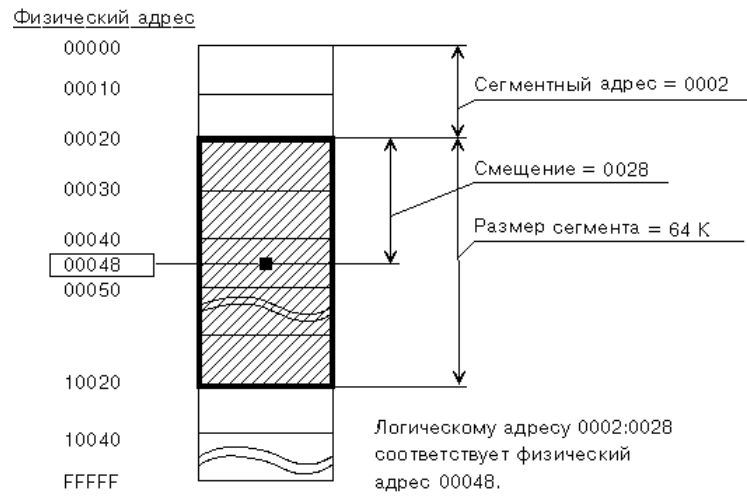
[ номер сегмента ] \* 16 + [ смещение сегмента ]



Одному физ. адресу могут соответствовать несколько логических (0001h:0400h === 4001h:0000h)

Структура адресного пространства:

Вся память делится на 16 сегментов по 64 кб. Эти сегменты называют **страницами памяти**.



Основная область памяти (сегменты 0-9): доступна пользователю, 640 кб.

Upper Memory Area (сегменты 10-15): 384 кб, информация об аппаратной части компьютера.

High Memory Area: 64 кб минус 16 байт, лежащие за первым мегабайтом: FFFFh:0010h – FFFFh:FFFFh (использовалась в MS-DOS для выгрузки ядра, тем самым освобождая дополнительные 46 кб основной области памяти для приложений).

Расширенная память: лежит за пределами НМА.

Для обеспечения сегментации памяти, номер сегмента хранится в сегментном регистре. На 8086 — четыре сегментных регистра (CS, DS, ES, SS), на 80386 — шесть регистров.

## **2. Защищенный режим**

- Основной режим работы.
- Виртуальное адресное пространство.
- Защита памяти.
- Многозадачность.
- Сегментная организация виртуальной памяти.
- Страничное управление памятью.

### Сегментная адресация виртуальной памяти:

- Для программы память — набор сегментов.
- Сегменты разного размера.
- Сегмент имеет дескриптор (адрес сегмента, предел сегмента, тип, права доступа, гранулярность, присутствие и пр.)

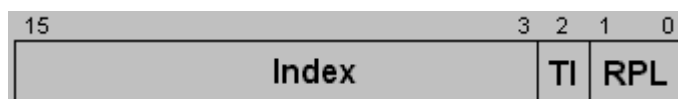
**Гранулярность** – единица измерения размера (байты или страницы)

**Дескрипторные таблицы** – служебные структуры данных, содержащие дескрипторы сегментов.

- GDT (Global Descriptor Table)
  - Сегменты операционной системы
  - Сегменты разделяемых структур данных
- LDT (Local Descriptor Table) – для каждого процесса/задачи
- до 8192 записей в таблице
- до 8192 LDT
- GDT всегда одна
- в каждый момент времени процессору доступна только одна LDT

Виртуальный адрес в данном режиме вида <селектор:смещение>

**Селектор** – номер дескриптора из таблицы дескрипторов.



- Index — номер дескриптора (0–8191)
- TI (Table Indicator) (0 — GTD, 1 — LDT)
- RPL (Requested Privilege Level) — уровень привилегий программы

### Страничное управление памятью

Вся память разбита на отдельные логические страницы.

**Логическая страница** — виртуальная непрерывная область памяти фиксированного размера, может быть отображена на любой физический адрес (абстракция над обычной памятью). Причем виртуальная память может быть логически шире, чем физическая память.

Каждая страница имеет два адреса: линейный и физический.

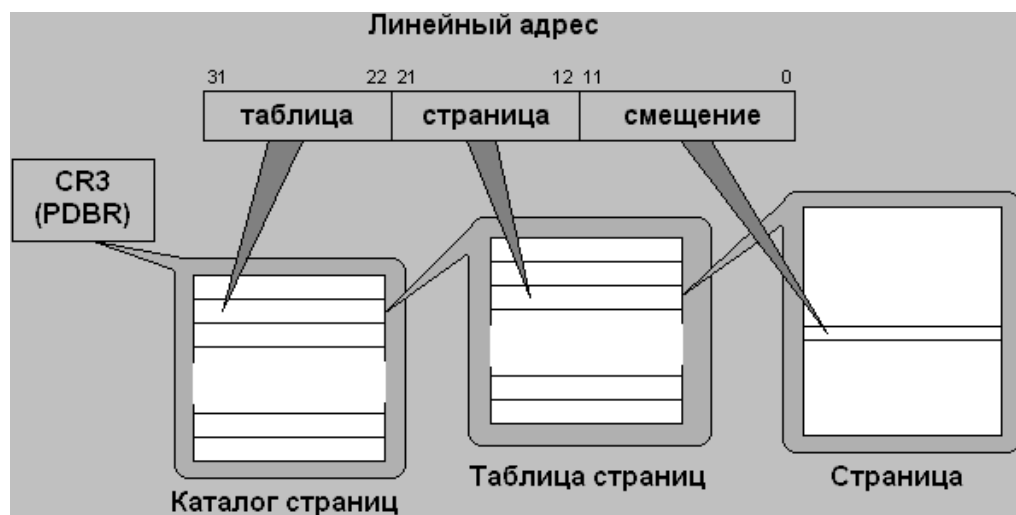




Селектор  $\Rightarrow$  GDT/LDT  $\Rightarrow$  смещение  $\Rightarrow$  **виртуальный** линейный адрес  $\Rightarrow$  физический адрес

Со страничной организацией памяти связаны несколько структур данных:

- Page Directory — каталог страниц
- Page Table — таблица страниц
- Page Frame — сами страницы
- CR3 (PDBR) — регистр, указывающий на начало каталога страниц



**PMMU** (Paged Memory Management Unit) – аппаратный компонент в процессоре, отвечающий за трансляцию адресов виртуальной памяти в физические адреса.

\*\*\*

Защищенный режим в x64:

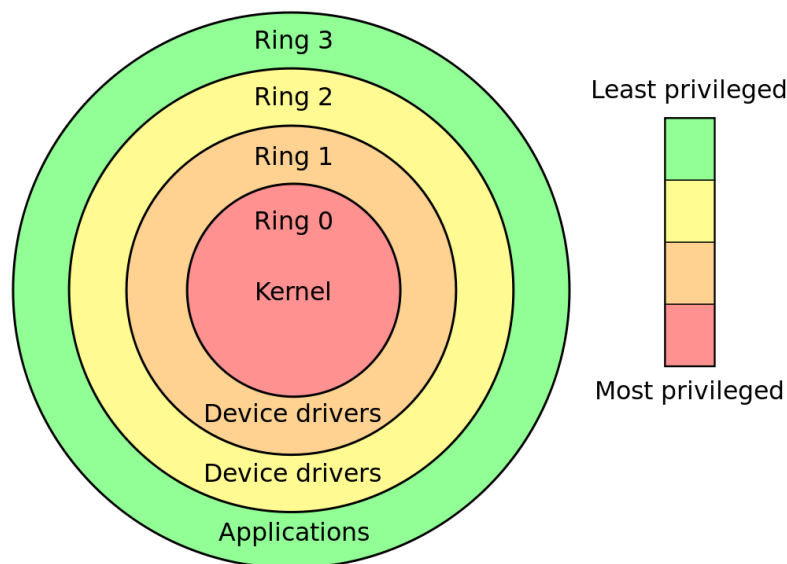
- Сегментация устарела, существует лишь для обратной совместимости
- “Плоская модель памяти” — один сегмент размером в  $2^{64}$  байт
- Новые таблицы:
  - PDPT — Page-Directory Pointer Table — таблица указателей на каталоги страниц.

**Задача** — единица измерения заданий процессора, которую процессор может диспетчеризовывать (т.е. управлять).

- содержимое регистров
- пространство памяти задачи
  - адресное пространство задачи
  - сегмент состояния задачи (информация о состоянии регистров и пр.)

Процессор очень быстро переключается между задачами, создавая иллюзию многозадачности.

Уровни привилегий задач:



(прим.: + ещё Ring «-1» – Hypervisor)

Уровни привилегий обеспечивают защиту памяти:

- Контроль наличия сегмента/страницы
- Контроль предела сегмента/страницы

- Контроль типа сегмента
  - Код
  - Данные
- Контроль типа страницы
  - Read-only
  - Read-Write
- Контроль привилегий на доступ/выполнение

**Исключения процессора** –прерывания, генерируемые процессором на нарушения защиты.

- Ошибка (fault) — можно исправить (например, segfault)
- Ловушка (trap) — брейкпоинты
- Авария (abort) — продолжение работы невозможно

### **3. Режим виртуального 8086**

- Выполнение программ для реального режима, но с защитой
- Можно включить только из защищённого режима
- Нужен для обратной совместимости

### **4. Режим системного управления**

- Самый привилегированный
- Останавливает исполнение всего кода
- Применение:
  - Отладка процессора
  - Выполнение эксплойтов