Лекция 1:

[Направления развития ЭВМ. История механических и электромеханических приборов для вычислений]

Два направления развития вычислительной техники

Эволюционная модификация ЭВМ Фон Неймана	Параллельные вычислительные системы
• последовательная обработка данных	• параллельная обработка информации
• увеличение производительности за счет улучшения технических характеристик	• повышение производительности за счет увеличения количества вычислителей и улучшения технических характеристик отдельного вычислителя

Два направления развития вычислительной техники

Эволюционная модификация ЭВМ Фон Неймона

- последовательная обработка данных
- увеличение производительности за счет улучшения технических характеристик

Параллельные вычислительные системы

- параллельная обработка информации
- повышение производительности за счет увеличения количества вычислителей и улучшения технических характеристик отдельного вычислителя

История вычислительной техники

- VI–V вв. до н.э. — Абак
- 1670 — Лейбниц — арифметическая машина
- 1808 — Жаккард — станки с перфокартами
- 1812 — Бэббедж — механический мозг
- 1878 — Чебышев — арифмометр
- 1938 — Цузе — машины Z1,Z2,Z3

История механических вычислительных приборов выше.

Абак - счёты. Содержит некоторые элементы, которые можно перемещать. (Лучше возьми из гугла, ей богу)

С **Арифметической машины Лейбница** начинаются машины для вычисления.

Жаккард по заданию Наполеона создал ткацкие **станки с перфокартами**. Она вставляется в станок. По заданной перфокарте воспроизводилось то, что записано на ней. Строительство этих станков стало массовым. Начались забастовки против технического прогресса, т.к. много кто остался без работы. Перфокарты были первым цифровым носителем информации. Перфокарта - картонка или что-то такое с отверстиями. В зависимости наличия дырки или её отсутствия станок считывал 1 или 0.

Используя разработки перфокарт, **Бэббедж** теоретически разработал машину, которую назвал **механическим мозгом**, что была очень близка к современной ЭВМ по архитектуре, но была механической. Там были характерные для ЭВМ узлы, устройства ввода-вывода, память и так далее. Использовал перфокарты Жаккарда. Но не смог собрать, но это хрень заложила архитектуру.

Первый реальный промышленный прибор был **арифмометр Чебышева**. Он выполнял основные действия с многозначными числами. Получило большую популярность. Использовалось для бухгалтерии, навигации, артилерии и многих других вещей.

Цузе начал производство первых **вычислительных машин**, работающих на электричестве. В качестве базового элемента они юзали электро-механическое реле. Эти машины назывались **Z1, Z2, Z3**. Машины Цузе в некоторой степени соответствовали машине Бэббеджа, то есть была универсальная вычислительная машина, в отличие от арифмометра. Была программируемая.

Все эти приборы либо механические либо электромеханические.

[Про разницу между электронным и электрическим прибором]
Электронный прибор (элемент) имеет нелинейную вольтамперную характеристику, а электрический прибор (элемент) имеет линейную вольтамперную характеристику.
Пример с нелинейной: транзистор, диод.
Пример с линейной: реле (через него ток идёт), резистор, электрическая лампочка.
Электрическая лампа и электронная лампа - это разные устройства.

Эдисон обнаружил интересный эффект, работая с лампочками: теле-электронная эмиссия (?). Приводит к нелинейной вольтамперной характеристике (при помощи использования определенных материалов для нити накаливания). Этот эффект он отметил в записях, хоть ему он и не понравился.

Флеминг же разработал первую электронную лампу, первую лампу с нелинейной вольтамперной характеристикой.

Ли де Форест создал триод (аналог транзистора, только на электронных лампах).

Благодаря ему **Бонч-Бруевич** создали триггер. Триггер - ячейка памяти. Триггер - прибор имеющий два устойчивых состояния, и позволяющий с помощью внешнего воздействие изменить свое внутреннее состояние. Например. если ничего не подаем на вход, то на выходе одно состояние, например 1. Если подаем единицу на вход, то состояние меняется, например, если была единица, то станет 0. На триггерах построили всякую шнягу типа счётчиков. Триггер - самый массовый прибор в эвм(собранный уже не на лампах, а транзисторах).

Отдельно Пиккард предложил **полупроводниковый диод** (отдельно от электронных ламп).

Предложили **полевой транзистор**. Может, это был не **Лилиенфельд** (ходят споры), но обычно ему присуждают.

Браттейн предложил **биполярный транзистор**

[Поколения ЭВМ ?]

Поколения ЭВМ

1	1945-1955	
	электронная лампа	
	перфокарта	
2	1955-1965	
	транзистор	
	магнитная лента	
3	1965-1975	
	ИС малой интеграции	
	дискета	
4	1975-н.в.	
	БИС и СБИС	
	жесткий диск	

Пояснение к таблице (для такого аутиста, как я): 1 - года выпуска; 2 - базовый элемент; 3 - носитель информации

ЭВМ - это вычислительная машина имеющая в своём составе электронные приборы, то есть приборы с нелинейной вольтамперной характеристикой, а это либо полупроводниковые приборы, либо электронная лампа.
Какой был первый прибор в ЭВМ? Лампы. Ламповый ЭВМ. Их проще было изготавливать.

[С перфокартами ещё прикол есть. Они до сих пор существуют и ими ещё пользуются на станках.]

1945 - 1955 гг - **ЭВМ 1-го поколения**. Одна ЭВМ - 80 m^2. Производительность несколько тысяч операций в секунду (существенно меньше, чем у нынешних телефонов). Было множество (порядка 20к) ламп, многие выходили из строя быстро (каждые 20 минут), приходилось часто менять. Первое время всё это делалось для военных целей. Для расчёта использовались ЭВМ. Ракетки там в космос запускали, оружие ядерное. Были ужасно дорогие.

1955 - 1965 гг - **ЭВМ 2-ого поколения**. Базовый элемент - транзистор (меньше электронной лампы, надёжность выше). Уже была массовая эта моча, да. 1 m^2 всего размером был. Носителем была магнитная лента. Магнитная лента на катушках вставлялась в специальное устройство (считыватель), откуда считывалась информайия и ЭВМ могла производить вычисления
Использовались на предприятиях уже, использовалось также всё ещё в военных целях. Производительность десятки и сотни тысяч операций в секунду.
Стоимость порядка 100к долларов

1965 - 1975 гг - **ЭВМ 3-его поколения**. Люди научились объединять транзистор в микросхему. В то время это были схемы малой интеграции. Стало ещё компактнее. Носитель информации - дискета. Производительность - миллионы операций в секунду. ЭВМ выпускались массово. Тут ещё языки программирования высокого уровня подвезли впервые. Раньше писалось двоичным кодом.

1975 - по сегодняшний день - **эвм 4-ого поколения**. Базой стали большие интегральные схемы и сверхбольшие интегральные схемы, носителем стал Жёсткий Диск (ну или уже ssd).

Для смены поколения нужно сменить базовый элемент и носитель информации.

[Квантовые компьютеры у Майданова сосут. На самом деле, их мало слишком, не массовое производство.]

ЭВМ Фон Неймана очень крутое. В лекции есть структура.

В где-то 1945 году Фон Нейман предложил структуру и принципы базовые, по которым строится большая часть нынешних ЭВМ.

Последовательное выполнение команд

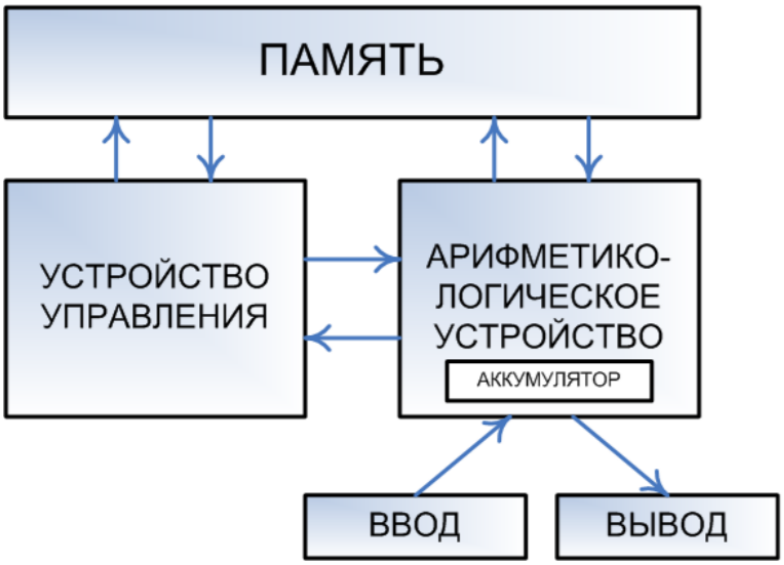
Блочная организация ЭВМ (каждый блок выполняет свою задачу)

Принцип однородности памяти (до этого, в большей части, было две разных памяти для хранения программы и данных, после программа модифицируется средствами самой эвм, также как и память)

Принцип адресности (каждая ячейка имеет адрес, все равноправны)

Принцип программного управления (осуществляется при помощи программы, а не при помощи электрической схемы)

Схема ЭВМ Фон Неймана



Отдельно выделенные устройства. **Арифметическо-логическое устройство (АЛУ)** - арифметические и логические операции, а также обеспечивает интерфейс для ввода-вывода и адресует ячейки в памяти. **Память** хранит программы и данные. **УУ** координирует работу всех устройств в составе ЭВМ. **Устройство ввода** используется для ввода данных. **Устройство вывода** - для вывода данных.

Чем современные ЭВМ отличаются от структуры Фон Неймана?
Арифметическо-логических устройств (АЛУ) может быть несколько. Последовательная обработка информации у Фон Неймана уступает современным параллельным способом обработки информации. У современных компьютеров также есть прямой доступ памяти, оно минует АЛУ (например для устройств ввода), поэтому работа происходит быстрее.

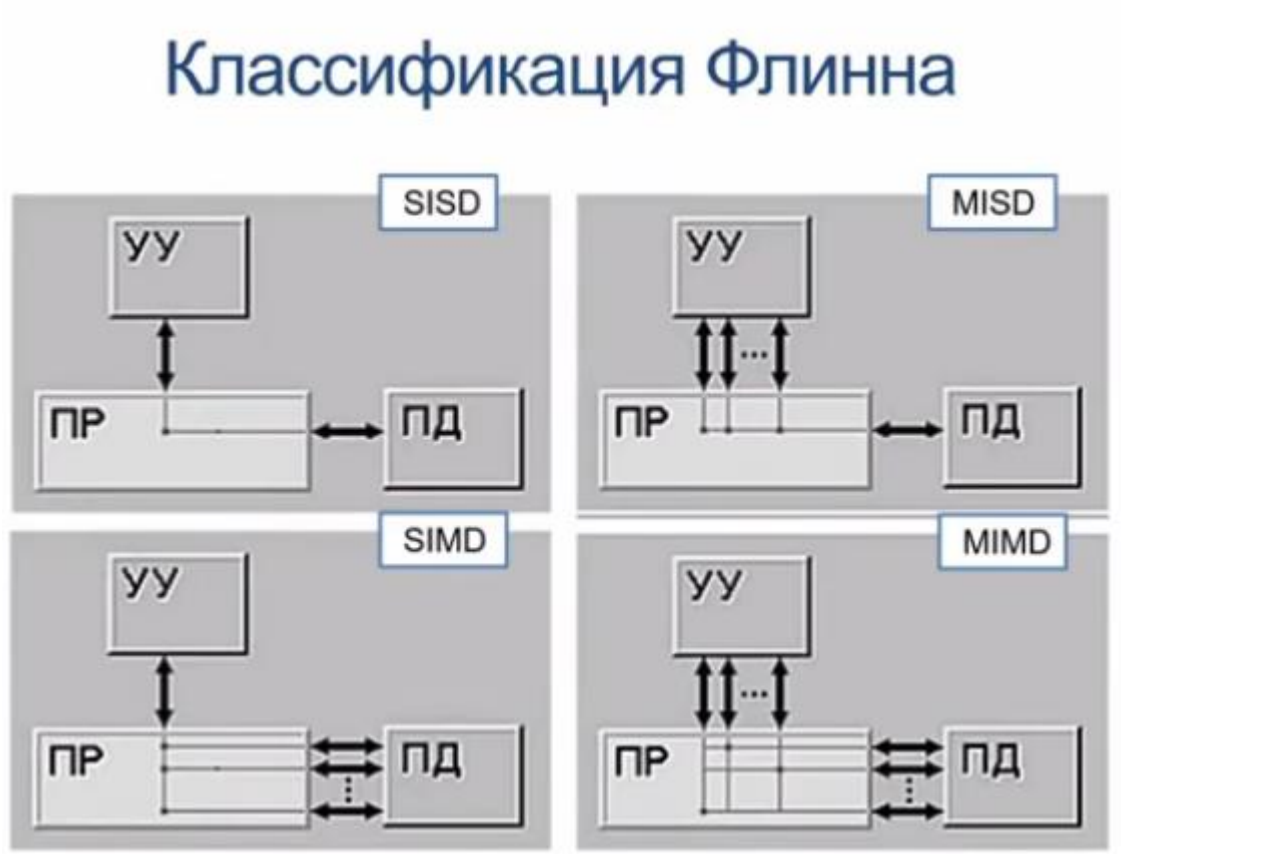
[Типы архитектур вычислительных систем ?]

Архитектура ПВТ Классификация Флинна

Классификация Флинна

- SISD — ОКОД
  (одиночный поток команд, одиночный поток данных)
- MISD — МКОД
  (множественный поток команд, одиночный поток данных)
- SIMD — ОКМД
  (одиночный поток команд, множественный поток данных)
- MIMD — МКМД
  (множественный поток команд, множественный поток данных)

Картинки (классификация Флинна)



УУ - управляющее устройство (команды)
ПР - процессор
ПД - поток данных

SISD - под него попадает структура Фон Неймана. Один поток команд, один поток данных. Подаются на процессор, обрабатываются по потоку. (коммент из беседы по лекциям - единственный класс с последовательной обработкой инфы simd не подходит под последовательную, потому что над несколькими данными совершаются одинаковые операции.)

MISD. Множество команд, одиночный поток данных. Можно представить в виде конвейера. Подаётся один поток данных, над потоком выполняются различные операции. Поток данных последователен, но одновременно выполняются несколько команд над этими данными. Параллелен: на одном этапе делают первую операцию, одновременно на втором делают вторую операцию и так далее. В один момент производится столько операций, сколько ступеней конвейера. Существенно увеличивает производительность за счёт параллельности.

SIMD. Один поток команд, множество данных. Над всеми этими данными идёт одна операция. Иначе называется векторная обработка информации.

MIMD. Множество команд, множество данных. Самый сложный, но самый перспективный. Можно из него получить все предыдущие три.

Основная проблема конвейера, если встанет один, то встанут и последующие. Чем меньше данных, тем меньше эффективность. Также если данные зависят от предыдущего этапа конвейера, то текущий этап будет ожидать. Это называется рисками при работе с конвейером. Конвейерная обработка информации хорошо тогда, когда нет ветвлений.

Лекция 2:

[Шинная организация ЭВМ]

Шина – совокупность линий, идущих параллельно и имеющих одинаковое функциональное назначение. Грубо говоря шина - это набор проводников, объединённых передачей сигнала одного типа.

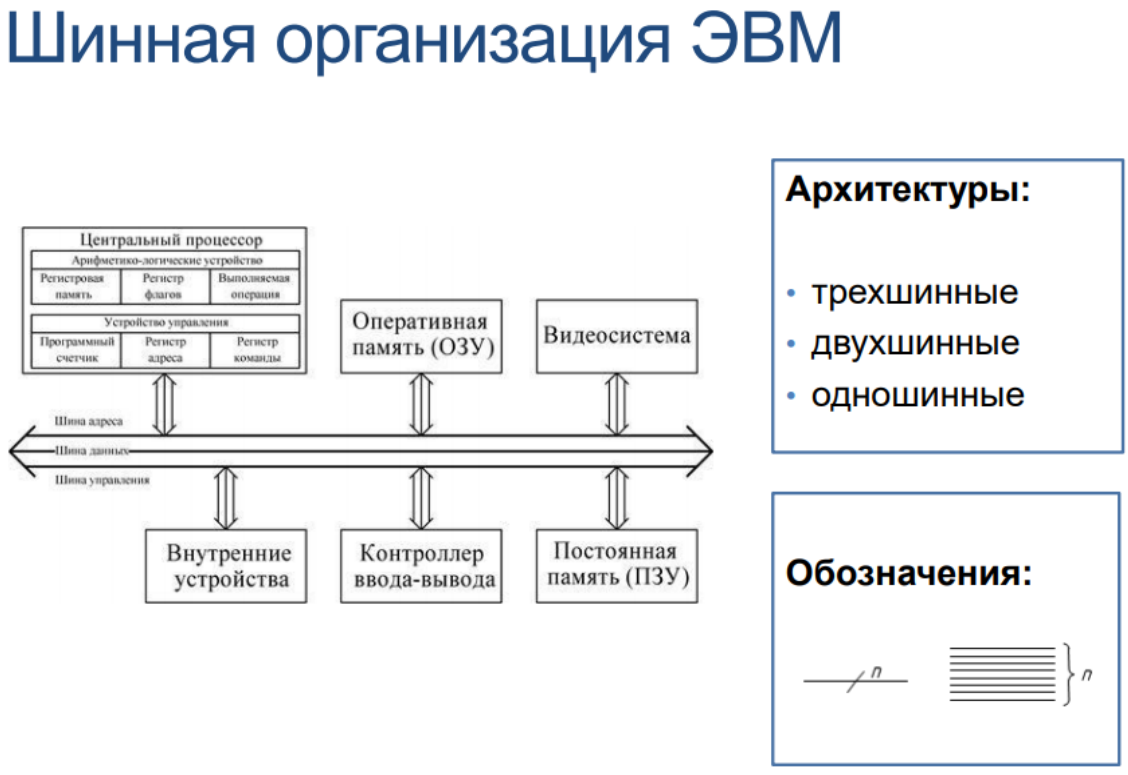
В ЭВМ присутствует 3 типа шин:

Шина данных. Предназначена для пересылки кодов обрабатываемых данных, а также машинных кодов команд между устройствами ЭВМ.

Шина адреса. Несет адрес ячейки памяти или порта ввода-вывода, который взаимодействует с микропроцессором.

Шина управления. Несет сигналы управления, обеспечивающие правильное взаимодействие блоков ЭВМ друг с другом и с внешней средой. По ней осуществляется синхронизация различных устройств

На рисунке представлена общая шина, которая состоит соответственно из шины адреса, шины данных и шины управления. В данной глобальной общей шине подключается центральный процессор, оперативная память, видеосистема, другие внутренние устройства и контроли работы вывода, а также ПЗУ.



Шинная организация ЭВМ

Как видно все устройства взаимодействуют через перечисленные три шины, но на самом деле в реальных системах бывают как три отдельные шины (В правом квадратике, где Архитектуры). Когда речь идёт про **трёхшинную** архитектуру, соответственно присутствуют физически линии все три: шины адреса, шина данных и шина управления.

В **двухшинной** шину адреса и шину данных объединяют в одну шину, соответственно по физическим по одним и тем же проводникам передаются в разное время разная информация. В первый период времени микропроцессор выставляет адрес устройства ввода/вывода памяти на эту совмещённую шину адреса и данных, а в дальнейшем по этой шине передаются данные.

Одношинные. В более примитивных случаях все три шины объединяются. По одним и тем же проводникам передаются адреса, данные и осуществляется управление всеми устройствами.

В настоящее время распространены двухшинные архитектуры. Почему объединяют шину адреса и данных?
1) Существенно удешевляет конструкцию
2) Дополнительная шина она потребует при современных разрядностях практически удвоить количество ножек процессору и увеличится расстояние, соответственно упадут скорости. Так как в процессе обмена процессора с оперативной памятью или процессора с нижними устройствами адрес требуется только в самом начале обмена, то предпочитают совмещать шину адреса и шину данных (Сокращает расстояние передачи данных, что соответственно сокращает скорость).

[**Мультиплексор** - устройство, имеющее несколько сигнальных входов, один или более управляющих входов и один выход. (В лк он такого не говорил)]
Мультиплексор объединяет шины в одну (объединяет соответствующие линии в одну), а демультиплексор разделяет.

Вот в процессоре эти шины объединяются, а дольше уже после процессора у нас имеется выделенная шина адреса и шина данных. Соответственно физически по проводникам процессора одна и та же шина передаёт в разный момент времени адрес и данные, а после демультиплексора уже имеются две раздельные шины. Соответственно в начальный момент, когда процессор выставляет адрес, то с помощью специального регистра запоминается внутри мультиплексора и присутствует на протяжении всего обмена данными процессора с внешними устройствами, либо с оперативной памятью.

Шинная организация данных

По способам передачи данных шины делятся на:

Параллельные - данные переносятся потактовое словами: каждый бит - отдельным проводником.

([Его пояснения:] когда за один такт обмена передаётся несколько разрядов данных. Соответственно передающее устройство на шину выставляет определённую порцию данных сразу, и приёмная за одно обращение считывает всю эту порцию данных).

Последовательные - биты данных переносятся поочередно по каналу, например, паре проводников.

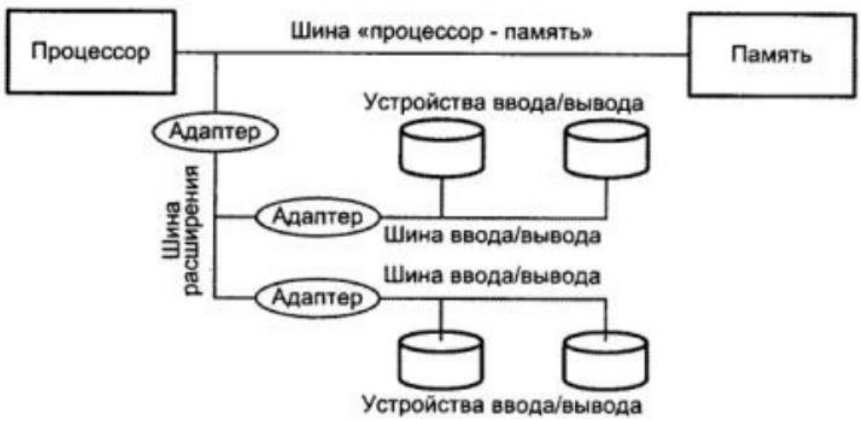
([Его пояснения:] биты передаются поочерёдно. Как правило имеется два проводника и по этой паре передаются разделённые по времени данные. То есть в один момент времени по шине передаётся один бит информации).

Есть свои достоинства и свои недостатки у этих способов и утверждать тоже невозможно кто из них быстрее. Всё зависит от стандартов и прочих моментов.

[Иерархия шин. Арбитраж шин]

[Иерархия шин.]

Иерархия шин



Самая быстрая шина - шина связывающая процессор с оперативной памятью. Скорость обмена максимально возможная для ЭВМ. В первых экземплярах ЭВМ к этой же шине подключались устройства ввода/вывода, то есть все внешние устройства подключались к этой же шине. Из-за этого была проблема: когда осуществлялся обмен между процессором и устройством

ввода/вывода скорость шины определялась возможностями ввода/вывода, которые как правило существенно медленнее чем память (из-за этого проседала вся производительность ЭВМ).

После этого была разработана схема, позволяющая строить архитектуру, в которой присутствует одна скоростная шина, связывающая процессор и память, к которой подключаются через адаптеры дополнительные шины (расширение). Скорость передачи данных на таких шинах соответственно ниже, чем по основной шине. Основная функция адаптеров при этом - это адаптация скоростей. То есть, если медленному устройству требуется передать данные в процессор, то оно на заданной скорости в своей шины ввода/вывода передаёт эти данные в адаптер, адаптер буферизует эти данные, и дальше уже на большей скорости адаптер, к которому подключено устройство ввода/вывода передаёт по шине расширения эти данные в головной адаптер. Головной адаптер также буферизует эти данные и уже на той скорости, на которой осуществляется взаимодействие процессора и памяти происходит считывание этих данных процессором. Соответственно, в то время, когда данные буферизуются в адаптере, шина процессор-память у нас свободна.

Вот таких веток, которые приведены на рисунке, где у нас один адаптер ответвления от шины процессор-памяти у нас на самом деле таких ответвлений несколько. Центральная шина, пока осуществляется буферизация, она используется другими устройствами ввод/вывода, либо памятью. Соответственно производительность ЭВМ не определяется пропускной способностью, устройство ввода/вывода. В обратную же сторону, если процессору требуется передать данные на медленное устройство ввода/вывода, то процессор на высокой скорости, характерной для шины процессора памяти передаёт данные в адаптер, подключенный к центральной шине на высокой скорости. Эти данные буферизуются в адаптере и всё, дальше шина процессор-памяти свободна. Дальше процессор может взаимодействовать с памятью.

В свою очередь по шине расширения центральный адаптер передаёт эти данные в адаптер той линии шины вывода, которой соответствует необходимое устройство, тем самым освобождается шина расширения, по которой также возможна передача иных данных между адаптерами, а потом уже адаптер, к которому подключено внешнее устройство, отдаёт внешнему устройству данные с той скоростью, с которой внешнее устройство способно принять эти данные. Таким образом вся система ввода/вывода работает максимально эффективно.

Помимо этого адаптер осуществляет первичный арбитраж шин.

[Арбитраж шин]

Под арбитражем шин понимается предоставление шины только одному устройству (типа автомагистрали). Взаимодействие возможно только между двумя устройствами. Соответственно одно устройство передаёт, а другое принимает. Чтобы два устройства не подключилось к этой шине у нас осуществляется арбитраж шины. Арбитраж осуществляется по приоритетам, если он чётко не выделен, то идёт переход на смену схемы приоритетов.

Схемы смены приоритетов:

Простая циклическая смена приоритетов. Заключается она в том, что выстраивается цепочка из её иерархий и через определённый промежуток времени происходит сдвиг в этой цепочке. Первое устройство становится последним, второе первым и так далее. Так получается равный доступ к шине.

Циклическая смена приоритетов с учётом последнего запроса. Через определённый промежуток времени меняем приоритет. Учитываем те устройства, которые в последнее время осуществляли обмен. Дольше всего не было обмена? -> максимальный приоритет. Позволяет повысить скорость передачи для всех устройств в сумме.

Приоритет по случайному закону. Через определённый промежуток времени запускаем генератор случайных чисел, который проставляет приоритеты всем устройствам. До следующего запуска генератора случайных чисел обмен осуществляется согласно данным приоритетам.

Схема равных приоритетов. Когда все устройства имеют равный приоритет (в долгосрочной перспективе). То есть те устройства, которые чаще всего захватывают эту шину, их приоритет понижается, а те устройства, у которых реже увеличивается. Таким образом доступность шины выравнивается.

Алгоритм наиболее давнего использования. За определённый промежуток анализируется информация о доступе устройств к центральной шине и то устройство, которое наиболее давно не пользовалось шиной, оно получает наибольший приоритет. То устройство, которое чаще всего использовало шину, оно получает наименьший приоритет.

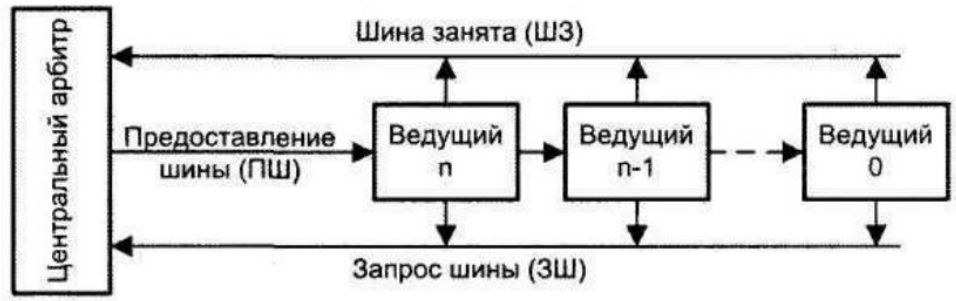
Алгоритм очереди. Накапливаются все запросы и просто по времени кто первый пришёл, тот первым вышел (первым обслужен). Остальные находятся в очереди. Каждое следующее устройство добавляется в конец очереди и будет обслужено после обслуживания после всех остальных устройств.

Алгоритм фиксированного кванта времени. Устройству, которому требуется обмен выделяем какое-то фиксированное время на обмен и после этого устройство освобождает шину и передаёт эту шину следующему устройству. Следующее устройство в свой квант времени также осуществляет взаимодействие и потом отдаётся это всё другому устройству.

[Какой принцип разделения устройств на схеме выше? **Хуй знает, он не сказал. [по скорости, скорее всего]]**
Почему клавиатуру и видеоадаптер на одну шину класть нельзя? У них разные скорости. Из-за этого будет прерывание сигнала. Шина адаптируется на самое медленное устройство.

Вот эти все принципы приоритетов они характерны для случая, когда приоритеты одинаковы, если они разные, то обслуживаются устройства с большим приоритетом
Адаптер шины уравнивает скорости.
Ого ебать, прикол с mp3. Раньше надо было в биос заходить, чтобы музыка нормально играла.
В старых компьютерах можно было повысить приоритет устройства, чтобы оно лучше работало.

Схема работы арбитража
Арбитраж шин

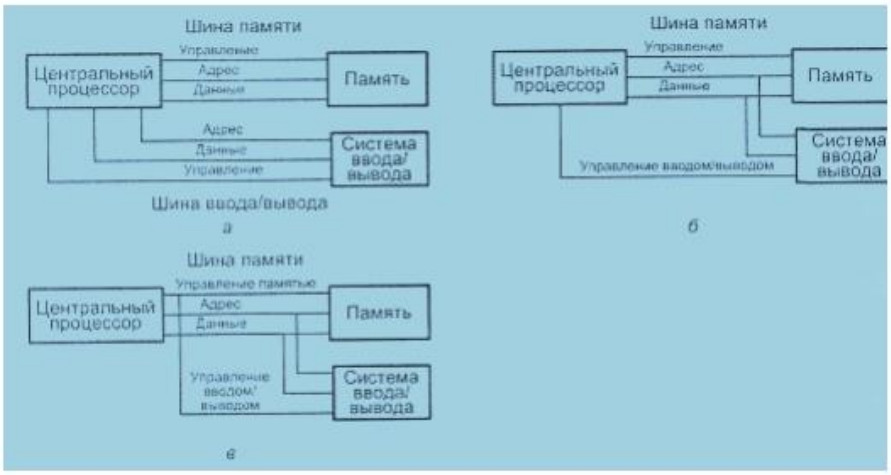


Тут есть центральный арбитр, который взаимодействует с множеством других. Эти арбитры между собой по определенному закону согласуют доступ к шинам. Грубо говоря есть главный орбитер на шине и имеются потребители этой шины. Иногда возникают ситуации когда 2 устройства получают доступ к шинам и избежания таких ситуаций существуют специальные алгоритмы.

Система ввода/вывода, основные функции

Основные функции:

- обеспечение интерфейса с ЦП и памятью («большой» интерфейс);
- обеспечение интерфейса с одним или несколькими периферийными устройствами («малый» интерфейс)



В составе ЭВМ присутствуют два интерфейса по большому счёту:
Большой интерфейс, согласно скорости присутствующей на данном интерфейсе. По этому интерфейсу осуществляется обмен данными процессора и памятью

Малый интерфейс. На этом интерфейсе процессор обменивается данными с периферийными устройствами. При этом возможны следующие варианты организации:

а) на рисунке а) показана ситуация, когда на каждый интерфейс имеется три шины для взаимодействия соответственно: шина управления, шина адреса и шина данных. Процессор имеет по сути два экземпляра шин управления, адреса и данных. Одни для малого интерфейса, другие для большого интерфейса. Помимо того, что шина адреса и памяти разделены, у нас ещё и добавляется шина адреса и шина данных, которые предназначены для системы ввода/вывода и для памяти отдельно. В реальности это очень много пРоВоДоВ. (Для телефонных станций и станков такая схема необходима ?).

б) Шины управления разделены, имеется шина управления памятью и шина управления ввода/вывода, а шина адреса и шина данных используется для

системы ввода/вывода одни и те же с памятью. Т.е. у нас имеется арбитр шин, который ответвляется шину расширения от центральной шины и обеспечивает взаимодействие с системами ввода вывода

в) Все шины совмещены, что для малого интерфейса, что для большого.

В современных ЭВМ используется нечто среднее между вариантами б) и в). Имеется специальные сигналы выделены на отдельные проводники, которые предназначены специально для системы ввода-вывода, либо специально для памяти. И имеется общая шина управления координации памяти и ввода-вывода

У всех свои достоинства и недостатки

а) Позволяет избежать какой либо дотации скоростей, по сути процессор проектируется с учётом адаптации скоростей и эта схема гораздо более сбалансирована, чем все остальные на систему ввода вывода и память. Но она требует очень большое количество проводников при высокой разрядности. Стоимость такого варианта возрастает. Сложность процессора также возрастает.

б) и в) В этом случае процессор взаимодействует с шиной одной и внешними по отношению к процессору средствами определяется, что у нас подключается к шине памяти или системе ввода/вывода. В первом варианте, вся эта нагрузка ложится на процессор.

Адресное пространство системы ввода/вывода

Ячейкам памяти сопоставляется физический адрес, чтобы процессору обратиться к какой то ячейке памяти процессор выставляет на шину адреса адрес этой ячейка и эта ячейка подключается к большому интерфейсу и процессор способен считать данные из этой ячейки памяти.

Аналогично осуществляется взаимодействие и с портами внешних устройств, каждый порт внешних устройств имеет свой адрес и процессор выставляет этот порт на шину адреса и осуществляется обмен данных
[У ячеек есть адрес. Процессор может считывать с неё. Каждый порт внешних устройств имеет свой адрес. Процессор выставялет шины (совмещённые или раздельные), а затем обмен данных.]

Само адресное пространство может быть организовано 3-мя разными способами.

Совмещение с памятью. В одном адресном пространстве находятся и ячейки памяти и внешние устройства. Процессору не важно, обращается ли он к ячейке памяти, либо к внешнему устройству. Просто определённый сегмент адресного пространства выделен под внешнее устройство, а остальной сегмент под память.

Преимущества: все операции доступны для портов внешних устройств, все эти множества способов адресации, они в основном ориентированы на память. (В данном случае вполне можно использовать для внешних устройств).

Недостатки: если адресная шина имеет заданную разрядность, то случай совмещённого адресного пространства портов ввода/вывода и памяти мы вынуждены это адресное пространство поделить между этими двумя группами и таким образом суммарное количество доступной памяти снизится, которое можно адресовать. Это представляет определённую проблему. Помимо этого требуется сложная обвязка, которая сама по нужным адресам осуществляет коммутацию шины на память и внешнее устройство.

Разделение. При разделении адресного пространства адресные пространства памяти и портов внешних устройств независимы. 4 внешних устройства и 4 ячейки памяти к примеру, *а в случае совмещенного например 3 ячейки памяти и 1 внешнее устройство*. Суммарное количество адресуемых устройств и ячеек памяти увеличивается. Возрастает сложность (требуется больше схем), количество способов адресации существенно снижается. Чтобы решить эти проблемы придумали **комбинированный** способ.

Если адресная шина состоит из 2 проводников, сколько всего устройств на ней можно адресовать? Ответ 4, -> 00 01 10 11

Достоинства: суммарный объем адресуемых ячеек памяти увеличивается.

Недостатки: возрастает сложность, требуются дополнительные схемы. Также те команды, которые доступны для работы с памятью не доступны для работы с внешними устройствами (у них свои команды). И количество способов адресации существенно снижается.

Комбинирование. Часть адресного пространства памяти отводится под внешнее устройство и, помимо этого, существует отдельное адресное пространство для других внешних устройств. К примеру есть две адресные линии, мы можем адресовать 5 внешних устройств и 3 ячейки памяти. Мы одно внешнее устройство внесли в адресное пространство памяти и сделали доступным для него все команды и сложные способы адресации, которые доступны для памяти. Упростили существенно работу. Но мы отняли всего один адрес, при этом остальные устройства, которые не требуют сложных способов адресации и сложных команд мы выделили в своё адресное пространство и они не отнимаю адресное пространство ячеек памяти. Соответственно количество доступной памяти нашей ЭВМ будет больше.

На сегодняшний день более популярен комбинированный способ.

Лекция 3:

[Характеристики памяти. Классификация запоминающих устройств]

Память

Основные характеристики:

- емкость (Бит, Байт)
- разрядность
- быстродействие (tсч = tп + tч + tр, tзп = tп + tз)
- массо-габаритные показатели
- энергозависимость
- удельная стоимость (руб./байт)

Классификация механизмов доступа:

- последовательный
- прямой
- произвольный
- ассоциативный

Компьютерная память - часть вычислительной машины, физическое устройство или среда для хранения данных.

Память

- **ёмкость** (емкость) - показывает количество информации, которое можно хранить в памяти (но при этом эта величина достаточно условная, потому что количество информации определяется природой информации)
- **разрядность** - показывает количество информации, которую можно считать или записать в память за одно обращение
- **быстродействие** (tсч = tп + tч + tр, tзп = tп + tз) - характеристика определяющая быстродействие, а именно: время считывания и время записи.

Время считывания (tсч) = время поиска + время чтения + время регенерации (это время необходимо для восстановления памяти в то состояние, в котором она была до считывания [нужно, если память основана на конденсаторах ил чём-то ещё?])
Время записи (tзп) = время поиска ячейки памяти + время записи в ячейку памяти

- **массо-габаритные показатели** - определяют вес, вид, размеры, объёмы, которые занимает память
- **энергозависимость** - способность памяти хранить информацию при отключении электропитания (например оперативная память энергозависима, в отличие от флэшек всяких)
- **удельная стоимос**ть - определяет количество денежных единиц на единицу хранимой информации

Классификация механизмов доступа:

- **последовательный** (чтобы считать один байт информации необходимо предварительно считать все единицы информации, которые размещаются на носителе до той единицы, которую нам надо считать) [пример: магнитная лента]
- **прямой** (очень похож на последовательный. отличается лишь тем, что перемещаемся в блок информации и уже блок считываем последовательно информацию из блока) (вся информация делится на блоки определённого размера, мы имеем произвольный доступ к блоку. В пределах блока мы имеем последовательную механизм) [пример: чаще жёсткие диски]
- **произвольный** (когда к любой ячейке памяти мы можем обратиться имея её адрес и считать, при этом нам не требуется считывать или как-то обходить другие ячейки памяти) [пример: современная оперативная память]
- **ассоциативный** (когда мы можем по определённым признакам считать ячейки информации) [например, запросить все ячейки памяти, где второй бит нулевой. Один из самых перспективных]

[Классификация запоминающих устройств]

Запоминающие устройства

Оперативные:

- DRAM (Dynamic Random Access Memory — динамическая память)
- SRAM (Static Random Access Memory — статическая память)

Постоянные:

- EPROM (Erasable Programmable ROM — стираемые программируемые ПЗУ)
- EEPROM (Electrically Erasable Programmable ROM — электрически стираемые программируемые ПЗУ)
- флэш-память

Запоминающие устройства
Оперативные:

- **DRAM** (Dynamic Random Access Memory - динамическая память) с произвольным доступом *[странно, ещё он сказал, что доступ случайный, но этого не объяснил]*. (к каждой ячейке памяти мы можем обратиться произвольно, признак динамической памяти означает, что при считывании мы стираем содержимое. Скажем, память организована на конденсаторах. Конденсатор заряжен, чтобы считать, мы разряжаем конденсатор и нужно восстановить данную информацию)
- **SRAM** (Static Random Access Memory - статическая память) - не требует восстановления при считывании. [Как я понимаю, то же самое, что DRAM, только не нужно восстанавливать, т.к. построена память иначе]

Постоянные (есть энергонезависимость):

- **EPROM** (Erasable Programmable ROM - стираемые программируемые ПЗУ) - нужна отчистка для перезаписи (при помощи какого-нибудь ультрафиолета)
- **EEPROM** (Electrically Erasable Programmable ROM - электрически стираемые программируемые ПЗУ) - нужна очистка для перезаписи, но не требует внешних воздействий физических

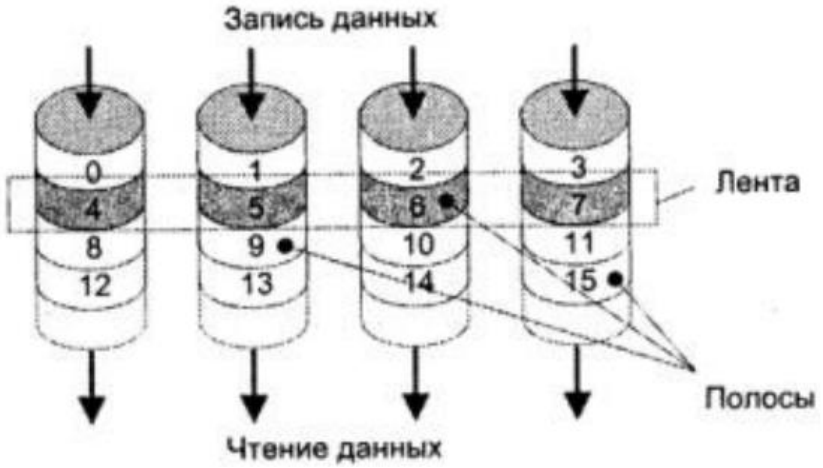
- **флеш память** (каждую ячейку можно перезаписывать независимо и при этом такая память способна длительное время сохранять информацию)

Raid (массивы дисков) [Raid-массивы, рейд-массивы, рэйд-массивы]

Raid представляет собой набор физических данных ЗУ, управляемых операционной системой и рассматриваемых как один логических диск
(Есть ещё RAID 7, но он запатентован компанией какой-то, также встречаются RAID 0 1, RAID 1 0, но это комбинации нижеперечисленных)
Помимо увеличения объёма диска, т.е. когда мы добавляем несколько дисков, вторую задачу решает – скорость доступа.

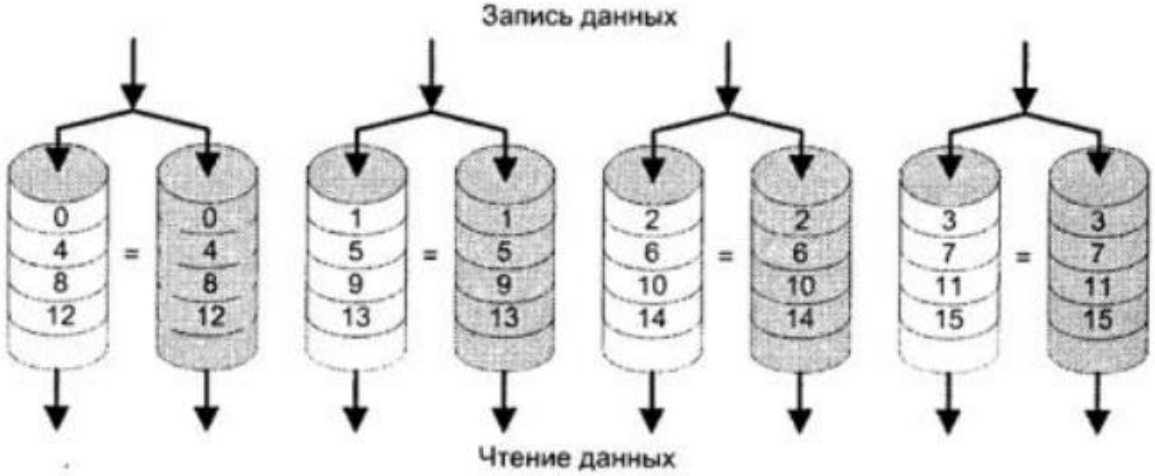
Общепринятые типы:

RAID 0
Не содержит избыточности и нацелен только на повышение производительности несмотря на ущерб надёжности. В основе лежит расслоение данных. Полосы распределены по всем дискам массива, запоминающих устройств. Данные записываются по циклической схеме. Соответственно нулевая полоса записывается на первый жёсткий диск, первая на второй, вторая на третий и т.д. Считывание происходит со всех дисков одновременно. Если вы поставите два диска, то вы **удвоите скорость чтения**, т.к. данные считываются с двух дисков и кусочно. **Проста в реализации [и используется весь объём памяти?].** При выходе из строя любого диска мы теряем всю информацию. **Надёжность дисков снижается. [поэтому неотказоустойчив?].**



RAID 1
(Самая популярная в домашних, но и самая дорогая, юзается на предприятиях не часто) Отказоустойчивость хранения данных. Избыточность достигается путём

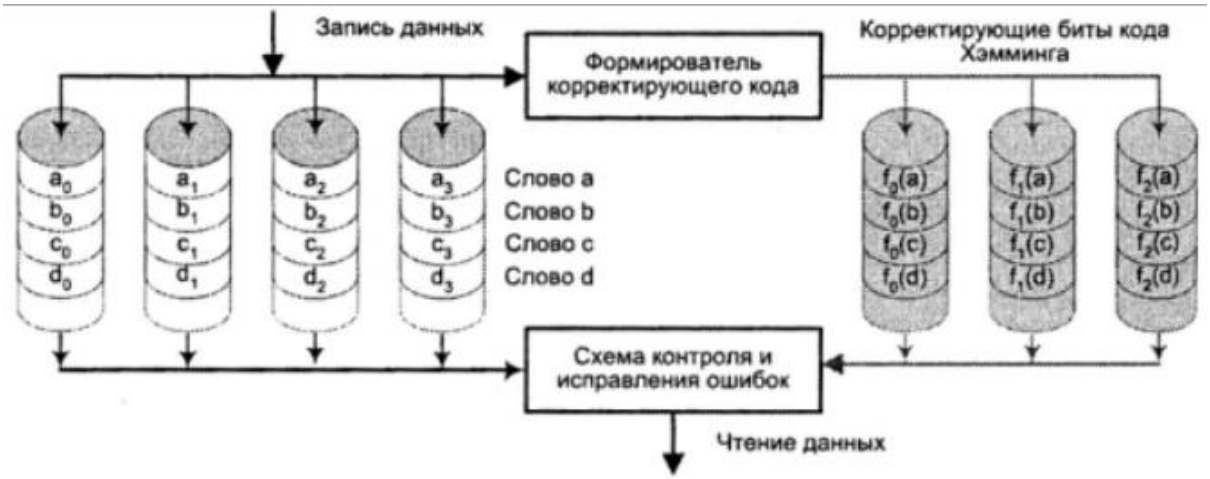
дублирования дисков. Если имеется два жёстких диска, то они работают параллельно. Информация записывается и на первый, и на второй жёсткий диск. Если дисков больше, а их должно быть чётное количество, то первые данные записываются на первую пару, следующие на вторую и так далее. **Скорость чтения увеличивается, запись остаётся прежней как и с одним жёстким диском. Не требуется вычисления контрольной суммы. По сути дела, происходит дублирование информации, контроллер проверяет состояние жёстких дисков, и если один из дисков отказал, то контроллер даёт эту информацию и начинает работать с одним жёстким диском, ожидая, когда вы замените жёсткий диск.** При замене жёсткого диска, информация клонируется, восстанавливая RAID 1. **[Из минусов: эффективная емкость хранилища составляет только половину от общей емкости диска, поскольку все данные записываются дважды]**



RAID 2 (отказоустойчивый дисковый массив с использованием кода Хемминга или чего-то ещё)

На картинке 4 жд (жестких диска). На каждый из них пишется разная информация. При этом имеется некая схема “Формирователь корректирующего кода”, который при записи обрабатывает всю записываемую информацию на жд и на отдельно выделенные специальные жд записывают определённые биты проверки. Жд при такой технологии синхронизированы так, что головки каждого запоминающего устройства в каждый момент времени находится в одинаковых позициях. Данные распределяются на полосы и распределяются по дискам таким образом, что полное машинное слово представляется поясом. То есть число дисков равно длине машинного слова в битах. То есть каждое слово записывается сразу одновременно на все *работчики сгимна* (это не то, но лучше

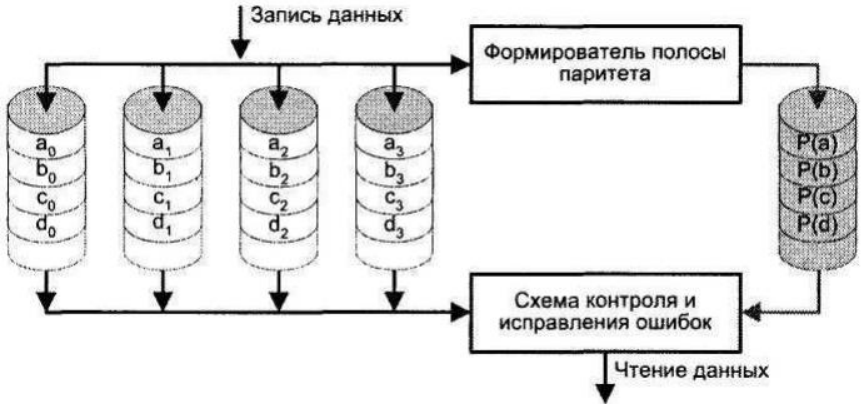
будет сказать, что на все диски данных или рабочие диски системы, хз). Благодаря вычислению кода Хэмминга или иного кода позволяющего восстанавливать данные у нас имеется возможность при выходе из строя жёсткого диска восстановить его содержимое. Если у нас жд выпал, то мы имея содержимое резервных дисков восстанавливаем благодаря обратному преобразованию содержимое выпавшего диска. Если же выпал тот, на котором были записаны корректирующие биты, то эти коррект. биты могут быть заново вычислены с тех дисков, которые у нас есть, и записаны обратно на этот жд. **Raid 2 позволяет достичь высокой скорости ввода вывода при работе с большими последовательными записями, но становится неэффективным при обслуживании записи небольшой длины. Основное преимущество raid 2 состоит в высокой степени защиты информации. В частности код Хемминга в современных жд встроен в каждый отдельный жд для данных этого диска. Из-за требуемой большой избыточности Raid 2 в настоящее время практически не используется)**



RAID 3

Очень похожа на технологию Raid 2. На картинке имеется 4 жд, которые разбиты на полосы: a, b, c, d и тд. Соответственно **a** делится на 4 части: a0, a1, a2, a3. На основе этих данных при записи формируется полоса паритета, которая хранится на пятом жд и она соответствует той полосе в которой сейчас записывается. Мы один слой положили в те диски и соответственно корректирующая информацию, полосу паритета, мы записали на специально выделенный для этого резервный диск, который называется диск паритета. Все диски нам необходимо синхронизировать в такой ситуации. Как правило для формирования полосы паритета используется всё тот же код Хэмминга. (Код Хэмминга не рассматривается в ходе этого курса, но мы его рассмотрим в ходе курса “Теория информации”. Довольно популярный код для помехоустойчивого

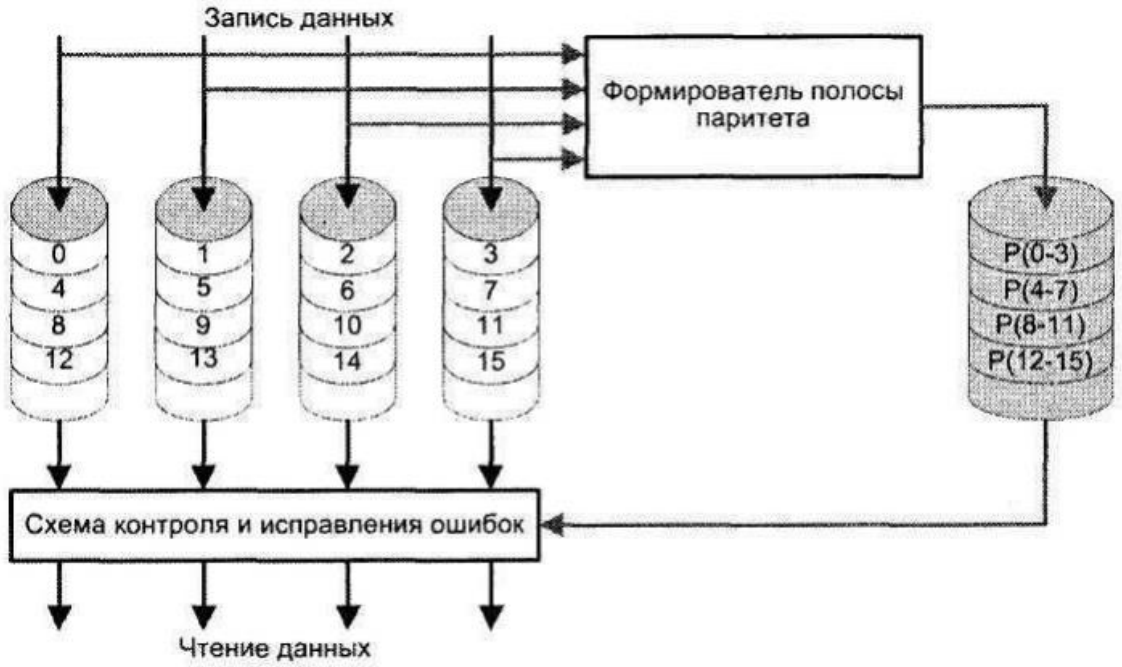
кодирования). Если один из жд выходит из строя (любой из этих пяти), то благодаря информации о паритете (наличия полосы паритета), содержимое этого жд мы можем восстановить. Например: вышел четвёртый диск из строя. Мы имея содержимое трёх исправных дисков и диска полосы паритета восстанавливаем содержимое за счёт обратного преобразования и перезаписываем этот диск на новый. Код Хэмминга позволяет исправлять одиночные ошибки, то есть есть присутствует система контроля исправления ошибок. **Raid 3 позволяет достичь очень высоких скоростей передачи данных**. Каждый запрос приводит к обращению ко всем жд. **При этом Raid 3 плохо работает с одиночными запросами. То есть, если мы читаем данные большими блоками, то эта технология очень эффективна. Если мы работаем с дисковой системой путём одиночных запросов, то производительность существенно падает**. **Главная особенность этой системы, что для хранения избыточной информации требуется всего один жд, причём независимо от их количества в массиве, то есть мы можем множество дисков воткнуть и при этом у нас всего один дополнительный диск потребуется для хранения полосы паритета.** Поэтому Raid 3 достаточно популярен в промышленных системах. **т.к. вероятность выхода из строя нескольких дисков крайне мала**.



RAID 4

По своей идее и технике формирования избыточной информации идентичен Raid 3, только размер полос в Raid 4 значительно больше. Обычно это несколько физических блоков на диске. Главное отличие состоит в том, что в Raid 4 используется техника независимого доступа, когда каждое запоминающее устройство в массиве состояние функционирует независимо так что отдельные запросы на ввод вывод могут удовлетворяться параллельно. Если требуется какая-то порция данных с одного жёсткого диска и одновременно порция данных с другого жд в системе Raid 4 мы можем получить эту порцию данных независимо обращаясь к этим жд. В остальном всё очень похоже. Также имеется один жд дополнительный, только блоки, которые обрабатываются кодом

Хэмминга у нас существенно больше и данные записываются существенно большими порциями, чем в технологии Raid 3. Для технологии Raid 4 характерны издержки обусловленные независимостью дисков. Если в Raid 3 запись производилась одновременно для всех полос одного пояса, то в Raid 4 осуществляется запись полос в разные пояса. Соответственно Raid 4 более хорошо работает с данными малого размера. И мы должны понимать, что каждая операция записи на любой из этих жд она потребует обновления информации на диске полосы паритета. Соответственно, когда мы меняем данные на диске, то у нас считается старая полоса паритетов, новые данные, и на основе этого формируется новая полоса паритета, которая позволит восстановить жд при его отказе. Raid 4 наиболее подходит для приложений требующих поддержания высокого темпа поступления запросов ввода вывода и уступает Raid 3 там, где приоритетен большой объём пересылки данных. [из минусов, наверное, можно выделить, что данные не считаются записанными, пока они не будут записаны в диске паритета, но я хз, гугл вообще про рейд 4 не особо хочет трепать]

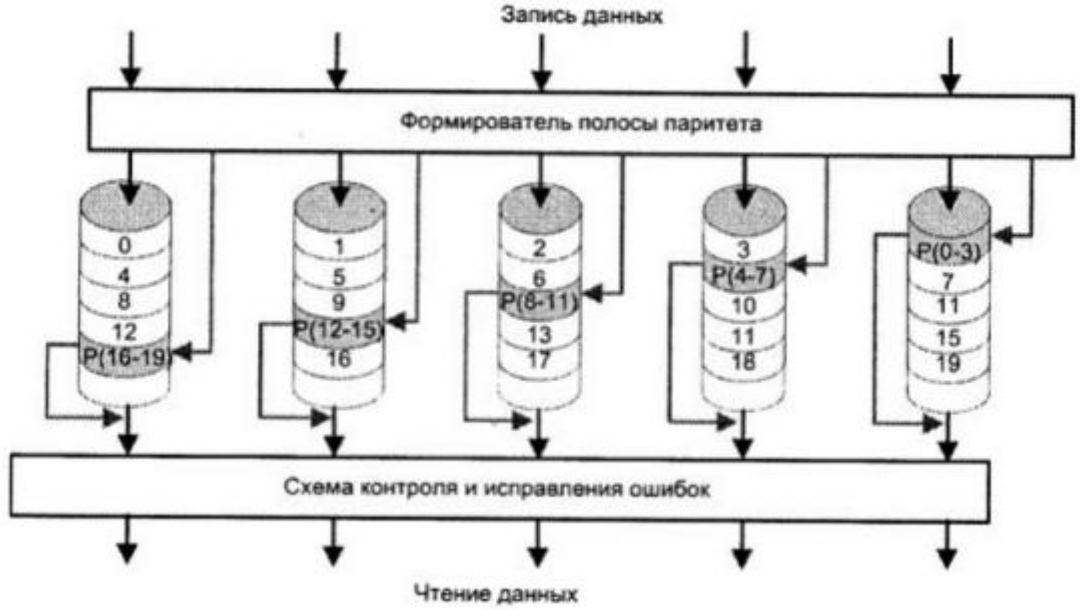


RAID 5

Имеет структура, напоминающую Raid 4. Различие заключается лишь в том, что Raid 5 не содержит отдельного диска для хранения полос паритета. Эти полосы паритета разносятся по всем дискам. Типичное распределение полос париетета осуществляется по циклической схеме. То есть мы видим, что в данном случае имеется 5 жд, нулевая порция данных на первом диске, первая на втором, вторая на третьем, третье на 4, четвёртое на третьем и на пятом жд у нас такое же место занимает полоса паритета. Далее соответственно для следующих порций данных

используется следующий жд для хранения полосы паритета и т.д. То есть по сути дела у нас нету выделенного диска для хранения полосы паритета. Полосу паритета мы храним на каждом жд.

[+: Время чтения быстрое, при отказе диска, можно восстановить всю информацию. -: при замене диска восстановление может продлиться очень много времени. Если какой-то диск за время восстановления выйдет из строя, то данные могут быть потеряны не всегда. Очень сложная.]

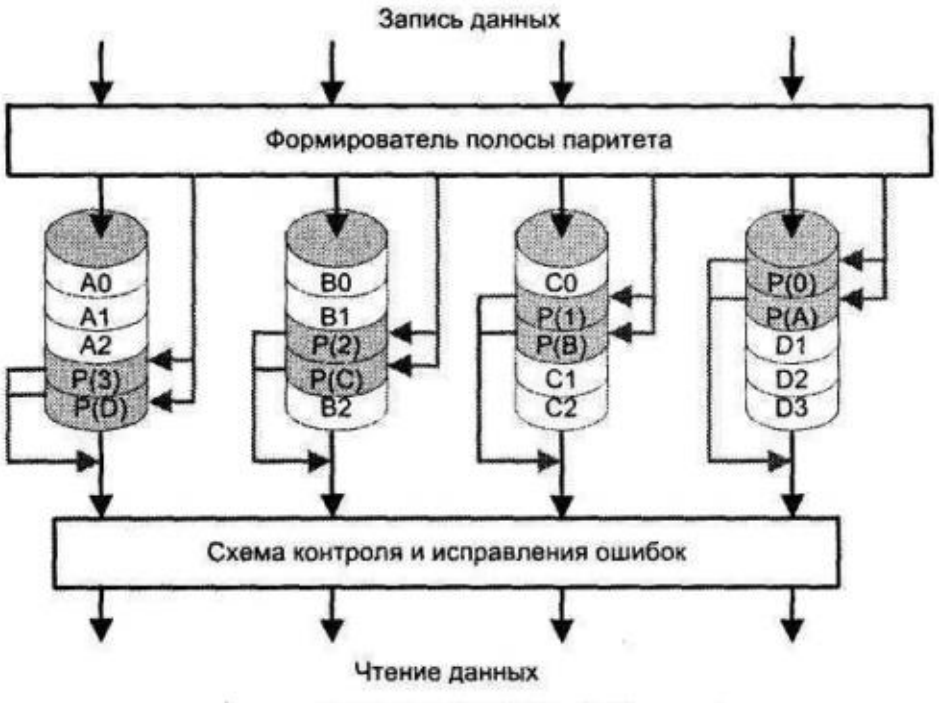


RAID 6

Очень похож на Raid 5. Данные также разбиваются на полосы размером в блок и распределяются по всем данным диска массива. Аналогично полосы паритета распредены по разным дисками и аналогично они располагаются циклически. Доступ к полосам независимый и асинхиронный. Различие состоит в том, что на каждом диске хранится не одна, а две полосы паритета. В дополнение формируется и записывается вторая полоса паритета, контролирующая все полосы какого-то одного диска, то есть мы имеем: одна полоса паритета (можно назвать её вертикальным срезом), другая полоса паритета это горизонтальный слой (срез). То есть мы видим, что у нас на 4-ом диске хранится горизонтальный срез, то есть данные a0, b0, c0, от них вычисляется код Хемминга и записывается порция, которая называется p0. Вторая порция входов на этом же диске хранит вертикальный срез со всего диска с порциями данных **a** и так далее. Соответственно второй жд хранит порцию **b**, однако помимо горизонтальных срезов он хранит вертикальных срез порции данных **C**.

Особенность данной схемы в том, что у нас при выходе из строя двух жд система способна восстановить информацию. То есть у нас допускается выход из строя двух жд. Мы используя эти полосы паритета сможем

восстановить содержимое этих дисков. Недостатком является, что дополнительное дисковое пространство по сути дела ам эм а.. имеем один дополнительный диск относительно тех трёх, то есть на один диск нам больше требуется. В данном случае нам потребуется несколько больше информации, по сути дела как второй диск дополнительный, но информация распределена по всем жд системы. Такая система значительно дороже Raid 5, поэтому используется реже.
+: Быстрое чтение. Возможность восстановить два диска (безопаснее РЕЙДа 5). -: Сложный - восстановление может продлиться долго. Низкая скорость записи (так как надо больше паритетов высчитать)



Существует ещё один Riad 7, который запатентован определенной фирмой.

По сути дела Raid 0 1 - это массив Raid 1 состоящий из двух вложенных массивов Raid 0, то есть используем две технологии Raid 0 и Raid 1. Raid 1 для обеспечения отказоустойчивости, а Raid 0 для обеспечения производительности. Естественно количество дисков массива Raid 0 должно быть одинаковым. Это будет Raid 0 1 у нас технология. Также существует технология Raid 1 0, которая внешне представляется как Raid 0, однако каждый из этих дисков входящих в систему Raid 0 внутри дублируется и позволяет обеспечивать отказоустойчивость.

Лекция 4:

[Способы ввода-вывода информации ?]

Для самых примитивных устройств используется **непосредственный обмен**. Если процессору необходимо обратиться к состоянию перемычки, то нет необходимости опрашивать готова ли к обмену перемычка аналогично и со светодиодами.

Второй обмен более сложный **по опросу готовностью**. Инициатором обмена здесь выступает также микропроцессор. Управление также осуществляет процессор. Перед началом обмена процессор опрашивает состояние устройства, если устройство не готово, то процессор выполняет холостые такты, т.е. эвм ожидает готовности устройства и по факту парализовано. Если же устройство готово к обмену, то начинается обмен. Недостаток - простой при неготовности (по факту, эвм парализуется). Такой метод обмена используется с достаточно быстрыми устройствами с высокой степенью готовности, например аналого-цифровые преобразователи или цифро-аналоговые. Т.е. устройства которые 99% времени готовы к обмену. В этом способе вероятность потерять данные значительно меньше чем с непосредственным. Достоинство: меньший риск потери информации при обмене.

Третий способ это **обмен по запросу на прерывание**. Инициатор - внешнее устройство. Управление - микропроцессор. При необходимости обмена внешнее устройство обращается к процессору через контроллер прерываний или непосредственно к процессору и сообщает, что оно готово к обмену. Процессор, получив этот сигнал, завершает выполнение текущей операции и переходит к выполнению специальной программе обработчику прерываний (эта программа предусматривает взаимодействие с этим устройством). При этом процессор сохраняет свое текущее состояние в стеке перед началом обмена. По окончанию обмена процессор возвращает прерванное состояние из стека, которое до этого выполнял. Достоинства: обмен происходит только тогда, когда устройство готово (процессор ничего не выжидает); подходит для обмена информацией почти со всеми устройствами в составе ЭВМ. Недостаток - сложная организация (необходимость поддержки), сложная приоритезация при обращении нескольких устройств одновременно.

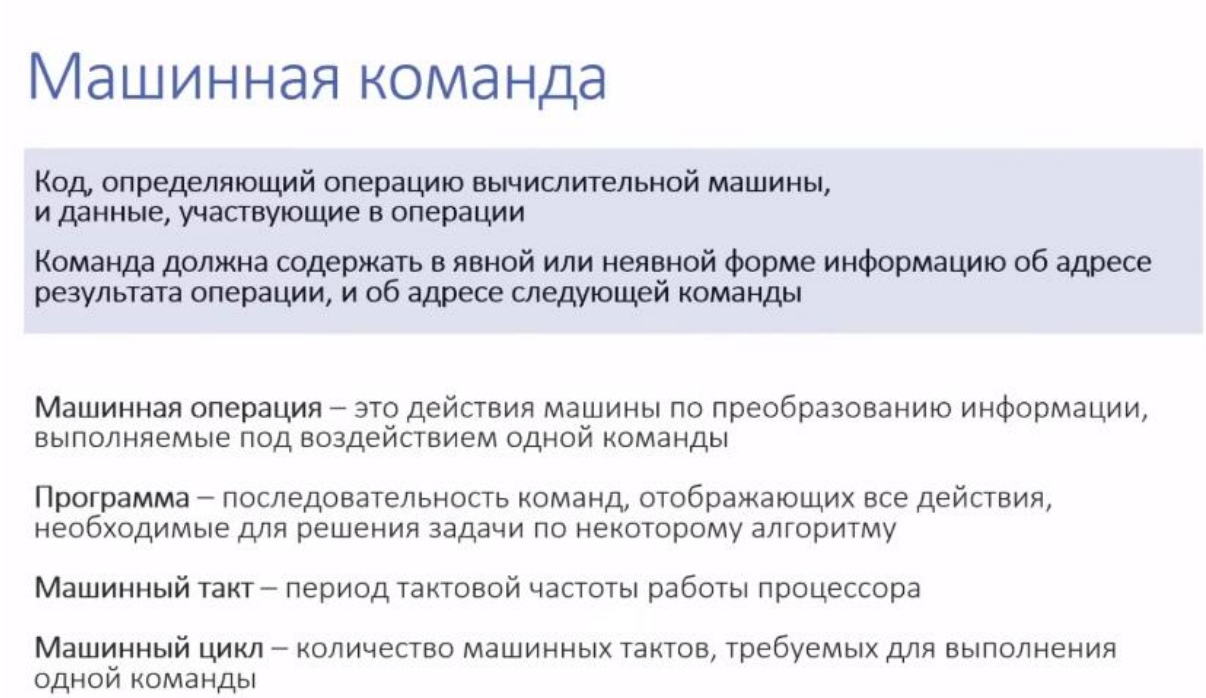
Самый быстрый метод обмена информации - **прямой доступ к памяти**. Осуществляется без участия процессора. Инициатор - внешнее устройство. Управление - контроллер прямого доступа к памяти. При необходимости обмена устройство подает запрос на специальный вход процессора и процессор завершает выполнение текущей операции, после чего передает все управление

контроллеру. Контроллер в свою очередь формирует адреса в памяти и т.д. Данные проходят не через процессор, а напрямую в память. Если происходит чтение из устройства, то информация сразу попадает в память, исключая из этой цепи процессор. По окончанию обмена контроллер прямого доступа к памяти возвращает управление процессору и продолжает выполнение программы. В отличии от прерываний нет необходимости сохранять текущее состояние.
Достоинства: самый быстрый из всех методов.
Недостатки - сложность организации (самый сложный) требуется очень сложный контроллер прямого доступа к памяти, который способен выполнять адресацию в памяти, координировать устройства и т.д. На время обмена процессор отключен от внешних шин и останавливаются операции на время обмена. Такой метод используется для быстрых устройств, которым требуется передать много информации. Например жесткий диск, видеоадаптер.

[Машинная команда, структура]

Машинная команда

Машинная команда - инструкция для эвм, которая определяет какую операцию необходимо выполнять над эвм и над какими данными. Каждая команда содержит информацию о операции, которую нужно выполнить, и адреса операнд, над которыми нужно произвести операцию, помимо этого адрес приемника результата и адрес следующей команды в общем виде.
Машинная операция, Программа, Машинный такт, Машинный цикл



Основные группы команд
Основные виды команд
Арифметические операции над числами с фиксированной или плавающей точкой
Команды двоично-десятичной арифметики
Логические (поразрядные) операции
Пересылка операндов (считывает из памяти и записывает в регистр и наоборот)
Операции ввода-вывода (процессор читает или записывает с внешнего устройства)
Передача управления (условные и безусловные переходы (if, while, goto))
Управление работой центрального процессора (смена частоты, остановить, произвести отладку).

Структура машинной команды

Форматы машинных команд
В зависимости от структуры бывают разные форматы
Фиксированный - все команды занимают одинаковое количество бит, и каждое поле занимает строго заданное количество бит. Более перспективный.
Плавающий - команды могут иметь разную длину и структура может отличаться.

Четырехадресная, трехадресная, двухадресная, одноадресная, безадресная
Коп - код операции

Можно избавиться от **адреса след. команды**, если после текущей подразумевается следующая команда. Также если нужно передать управления (перейти не на следующая команду, а в другой участок кода), то выделяется специальная команда.

Если **адрес результата** записывается в регистр (аккумулятор) процессора, то можно избавиться от адреса результата. Также можно вместо **второго адреса операнда** использовать аккумулятор.

Таким образом сокращается длина команды, это положительно сказывается на объеме. Также время на чтение команды из памяти уменьшается. Увеличивается разрядность под остальные поля.

Короче, чем меньше операндов, тем больше производительность.

Также есть команды, которые не используют операнды вообще, например команда остановки

У нас в курсаче была одноадресная. В современных пк используется одноадресная и безадресная. Двухадрессные и больше используются в специализированных ЭВМ.

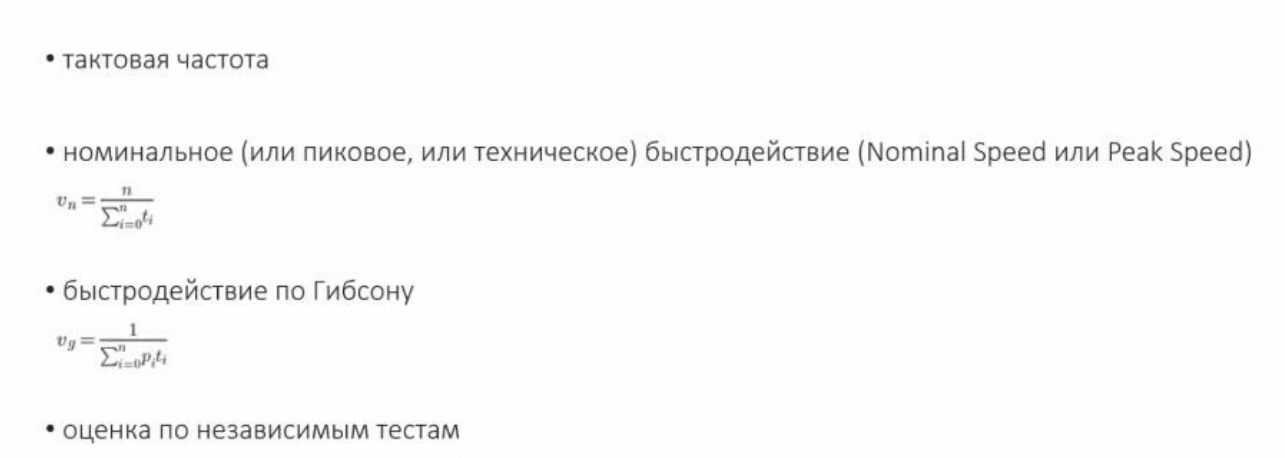
Производительность ЭВМ

Под производительностью ЭВМ понимается потенциальная возможность по обработке информации (а не реальная, учитывающая аномальности в работе ЭВМ, например, простои из-за отказов, из-за профилактического обслуживания и т.п.) (реальная ниже из-за нюансов)

В процессе обработки информации в ЭВМ реализуются те или иные операции из ее набора (или системы) операций.

Состав набора операций характеризует архитектуру ЭВМ и, следовательно определяет ее производительность. (Помимо технологического процесса по которому произведены детали эвм и т.д.)

[Показатели производительности]
Показатели производительности



Тактовое частатота. Чем тактовая частота выше, тем выше производительности, но это если архитектура одинакова. Если она отличается, то одинаковые команды могут занимать разное количество тактов, и тогда сравнение некорректно.

Номинальное быстродействие - величина, обратное среднему времени выполнения одной команды. Лучше характеризует производительность, чем тактовая частота. Но есть недостаток. Вклад каждой команды в оценку производительности будет одинаковый, а на самом деле некоторые команды выполняются чаще, а другие реже, поэтому вклад должен быть разный.

Быстродействие по Гибсану. Также команда умножается на вес. Чем чаще выполняется команда, тем выше вес. Грубо говоря вес - вероятность выполнение команды. Данный показатель очень точный.

Оценка по независимым тестам. Берется спец программа, которая оценивает производительность каких то значений. Часто тесты спонсируются производителями железа и считать их независимыми нельзя.

Единицы измерения производительности
Для оценки тактовой частоты: Герцы (Гц), Мегагерцы (МГц), Гигагерцы (ГГц)
Для оценки номинального быстродействия и быстродействия по гибсону, для чисел с фиксированной запятой: MIPS, MOPS
По результатам тестов используются FLOPS флопсы
1 флопс - 1 одна операция с плавающей запятой в секунду
Мегафлопсы - миллионы
Гигафлопсы - миллиарды
Терафлопсы - триллионы

Петафлопсы - квадраллионы операций в секунду
Для средних пк используют мега- и гигафлопсы, тера- и петафлопсы используют для высокопроизводительных вычислительных систем. Как правило это параллельные кластерные вычислительные системы специализированного назначения.

Лекция 5:

[Процессор. Типы архитектур]

Микропроцессор

В общем случае процессор — это мозг компьютера. Он читает инструкции из памяти, которые указывают, что делать компьютеру.

Инструкции — это просто числа, которые интерпретируются специальным образом. (Инструкция для процессора представлена в числовом виде и Интерпретируется им специальным образом. Из него процессор понимает, что делать, с чем делать, куда помещать и какая команда следующая.)

Микропроцессоры (CPU) выполняют простые операции:

load r1, 150 - загрузка в регистр r1 из ячейки памяти 150

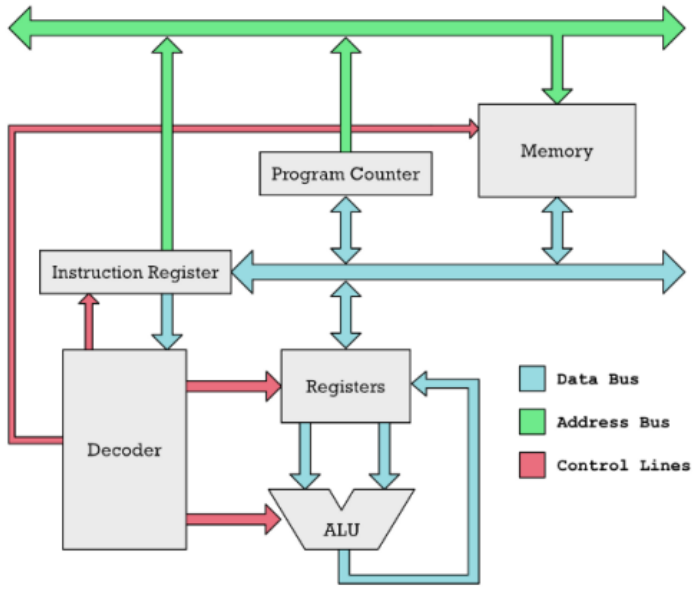
load r2, 200 - загрузка в r2 из ячейки 200

add r1, r2 - сложение содержимого регистров r1 и r2, результат помещается в r1

store r1, 310 - выгрузка r1 в ячейку 310

Диаграмма операций в процессоре:

Program counter, Memory, Instruction Register, Decoder, Registers, ALU



У нас имеется шина данных, шина адреса и шина управления. Они подключены к памяти. Все операции выполняется в АЛУ (арифметико-логическое устройство), который оперирует содержимым регистра. Команда после считывания попадает в инстракшн регистр, после декодируется в декодере и попадает в алу, в котором выполняется. Также показан программный счетчик, который используется для вычисления адреса следующей команды. В случае фиксированной длины команды, когда известна длина команды, счётик просто прибавляет к адресу текущей команды фиксированную величину, по которой можно вычислить адрес следующий команды. Адрес следующей команды выставляется на шину адреса. Оттуда адрес попадает в память, из памяти

считывается команда по шине данных. Поступает в регистр команд instruction register. Опять же декодируется, подается на АЛУ и на регистр, процессор выполняет операцию. Ну и процесс продолжает далее.

Архитектура набора команд (ISA)
Важной отличительной особенностью процессоров является архитектура набора команд.
У каждого процессора существует фиксированное количество команд, которые он понимает. Каждый тип архитектур имеет свой набор команд. В мире представлено множество различных микропроцессоров, и они не используют одинаковый набор команд. Иными словами, они интерпретируют числа в инструкции по-разному.

Микропроцессоры, например, Intel и AMD, используют архитектуру набора команд x86. А микропроцессоры, например, A12, A13, A14 от Apple, понимают набор команд ARM. Теперь в список ARM-процессоров можно включить M1.

Набор команд x86 и ARM не является взаимозаменяемым. (x86 является запатентованной архитектурой, а ARM является открытой. Компания ARM не продает как таковые процессоры, а разрабатывает концепцию и предоставляет экземпляры процессоров для демонстрации.) Архитектура набора команд сильно влияет на архитектуру процессора. Использование определенной архитектуры набора команд может усложнить или упростить задачу по созданию высокопроизводительного или энергоэффективного процессора. Набор команд определяет свойства этого процессора.

Изначально производители процессоров стремились увеличить количество и сложность добавляемых команд (даже разложение в ряд фурье добавили). Затем было замечено (примерно в 1985), что компьютер 90% времени выполнял простые команды, поэтому производители процессоров начали стремиться минимизировать время работы простых команд и избавления от сложных команд. Вся нагрузка начала ложиться на компиляторы, которым необходимо было преобразовать сложные команды в простые, поддерживаемые процессором.

Архитектура CISC
CISC — COMPLEX INSTRUCTION SET COMPUTER (РАСШИРЕННЫЙ НАБОР КОМАНД)
Отличительные особенности:
• большой набор команд (поддерживались достаточно сложные высокоуровневые команды.)

• использует переменные (плавающие) диапазоны форматов от 16-64 бит на инструкцию (позволяет экономить память при больших сложных командах)
• использует много способов адресации от 12 до 24 (проще реализовывать компиляцию программы)
• архитектура использует 8-24 регистров общего назначения (мало)
• использует механизм памяти к памяти для выполнения операций (большая часть операций производилась над ячейками памяти и помещает результат в эту же ячейку)
• использует унифицированный кеш для данных и инструкций
• имеет микрокодирование и использует управляющую память (ПЗУ)

Архитектура RISC

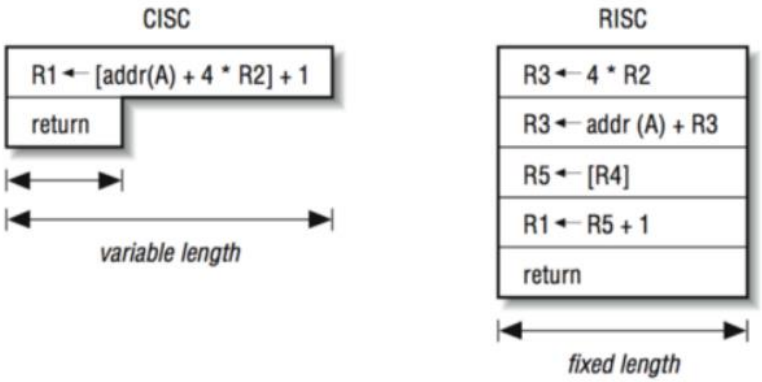
RISC - REDUCED INSTRUCTION SET COMPUTER (СОКРАЩЕННЫЙ НАБОР КОМАНД)

Отличительные особенности:

• малый набор команд
• фиксированный формат машинной команды (32 бита) (позволяет очень быстро переходить к следующей команде позволяет, легче декодировать команду)
• использует один такт на команду и режим ограниченной адресации (т.е. способов адресации 3-5)
• число регистров общего назначения колеблется от 32 до 192 (большое)
• основные операции в процессоре выполняются над содержимым регистров - данные загружаются в регистры, затем внутри регистров проводится некая операция (повышается эффективности и скорости вычислений)
• имеет разделенный дизайн данных и кеш инструкций
• большая часть управления процессором является аппаратной

Хоть RISC процессоры и производительнее (быстрее) CISC при одинаковой тактовой частоте и прочих параметрах, но большинство современных компьютеров используют x86 архитектуру (т.е. это CISC процессоры). Потому что большая часть софта писалось под циск и неэффективно его переписывать под риск.
Такая проблема было уже в 80-х годах. RISC пользователей очень мало, поэтому софт продолжался писаться под CISC. Также на производстве уже имелся налаженный рынок сбыта CISC и себестоимость была меньше, чем у RISC. Поэтому RISC используется в новых устройствах. В планшетах там в телефонах и т.п.

Отличие форматов команд CISC и RISC



Гибридные архитектуры

Идея — обеспечить программную совместимость с CISC процессорами, но добиться эффективности RISC процессоров.
Микропроцессор имеет интерфейс CISC, а команды выполняет как RISC.
Поддерживается конвейеризация.

Процессор имеет циск. Внутри ставится декодер с циск на риск. И внутри вычисления происходят на риск. Конвертация подобного рода занимает время (снижает производительность), но позволяет создать совместимость циск и риск архитектур.

Лекция 6:

[Способы адресации]

Способ адресации

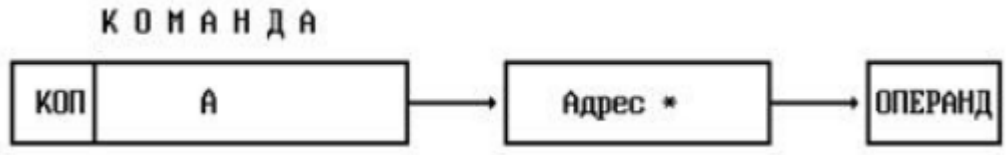
В адресном поле команды содержится командный (исполнительный) адрес (т.е. адресное поле 1, адресное поле 2 и т.д. т.е. все адреса, которые входят в команду.)

Команда выбирается по физическому адресу (в том числе ячейка, связанная с адресом самой команды, ячейка связанная с операндами и ячейка записи результата, они также выбираются процессором по физическому адресу, в большинстве случаев физический адрес не совпадает с командным.)

Способ адресации—алгоритм получения физического адреса по командному

То есть, то что содержится в команде преобразуется, и получается физический адрес, по коему и изымается операнд, либо определяется приемник результата (именна та ячейка памяти, куда происходит запись результата), либо адрес следующей команды.

Способ определяется кодом операции. То есть процессор считывает код операции и в нём также определяется порядок работы. (Кодом операции задается способ адресации.)



На рисунке видно, что у нас есть команды, в ней есть некий адрес. Этот адрес является командным, он определенным образом преобразуется. Формируется физический адрес и вынимается операнд из памяти уже по физическому адресу.

Классификация способов адресации

Явные - в адресном поле есть командный адрес (вот если поле присутствует и там есть какое то значение)

Неявные - нет адреса в адресном поле (дескать, второй операнд во внутреннем регистре хранится или сам внутренний регистр процессора подразумевается под приемником результата) (т.е. подразумевается какое то значение)

Способы адресации. Явные:

Непосредственная адресация: операнд (байт или слово) непосредственно в самом коде команды. (Например если складываем число с константой (скажем 5). Мы указываем код операции, затем адрес операнда и константу)

[(код операции - адрес операнда - число или слово (скажем, 5)]

Прямая адресация: физический адрес совпадает с командным (единственный в своём виде, редко используется) !!! (используется в курсаче, физический адрес совпадает с командным, чтобы упростить задание)
Косвенная адресация: командный адрес содержит адрес ячейки памяти или регистра, где находится операнд. (пример: я не знаю, где этот, но я знаю где тот, кто знает) - удобно, не меняя код, можно менять содержимое ячейки памяти и адресовать разные ячейки памяти, обращаясь к ним одним и тем же кодом
Автоинкрементная: после выполнения операции с данными адрес операнда увеличивается на 1, если формат БАЙТ, или на два, если операнд формата СЛОВО или иную константу. Используется для вычисления следующей команды (иногда). Автодекрементная - то же самое, только не +, а - (тот же тип)
Базовая адресация: физический адрес является суммой базового адреса и содержимого адресного поля (командного адреса). Базовый адрес обычно хранится в внутреннем регистре процессора, а командный адрес в команде. (Удобно при работе с одномерными массивами, когда адрес начала массива задаётся базовым адресом, а конкретный элемент при помощи командного адреса. При этом массив должен располагаться в памяти одним куском)
Базово-индексная адресация: физический адрес является суммой базового, индексного и командного адресов. Базовый и индексный обычно находятся во внутреннем регистре процессора, а командный в команде. Сложение трёх величин этих = физическому адресу. (удобно при работе с двумерными массивами, когда базовый начало расположения массива в памяти, индексный - адрес строки; и командный адрес - конкретный элемент в строке. Меняя содержимое индексного регистра шагаем по строкам, задавая базовый адрес - начало расположения массива в памяти, а командный - обратиться к конкретному элементу массива)
Укороченная адресация: адресный код содержит часть физ.адреса, другая лишь подразумевается (сокращает память под адрес и программы в целом). Вычисление физ.адреса по командному заключается в добавлении с одной из сторон, чаще всего слева, к тому адресу, который в команде, некоего числа.
Регистровая адресация: адресный код содержит номер регистра. Дескать, надо сделать над регистром операцию. Задается либо номер либо другой признак регистра.
Стековая адресация: указывается вершина стека. Процессор изымает из стека один операнд, далее второй, сделает операцию и помещает результат обратно в вершину стека. (часто используется, популярна) [можно для разнообразия ещё рассказать кратко, что такое стек]

Неявные:

1. Подразумевается операнд (типа, a++; - подразумевается a = a+1)

2. Подразумевается адрес операнда или приемника результата (в курсаче машинная команда имеет два поля: код операции и адрес первого операнда. Если взять сложение, например, то первый операнд сложат с тем, что в аккумуляторе, а записываться это всё будет в аккумулятор). Следующая команда - сдвигается на единицу от текущей (текущая + 1).

Лекция 7:

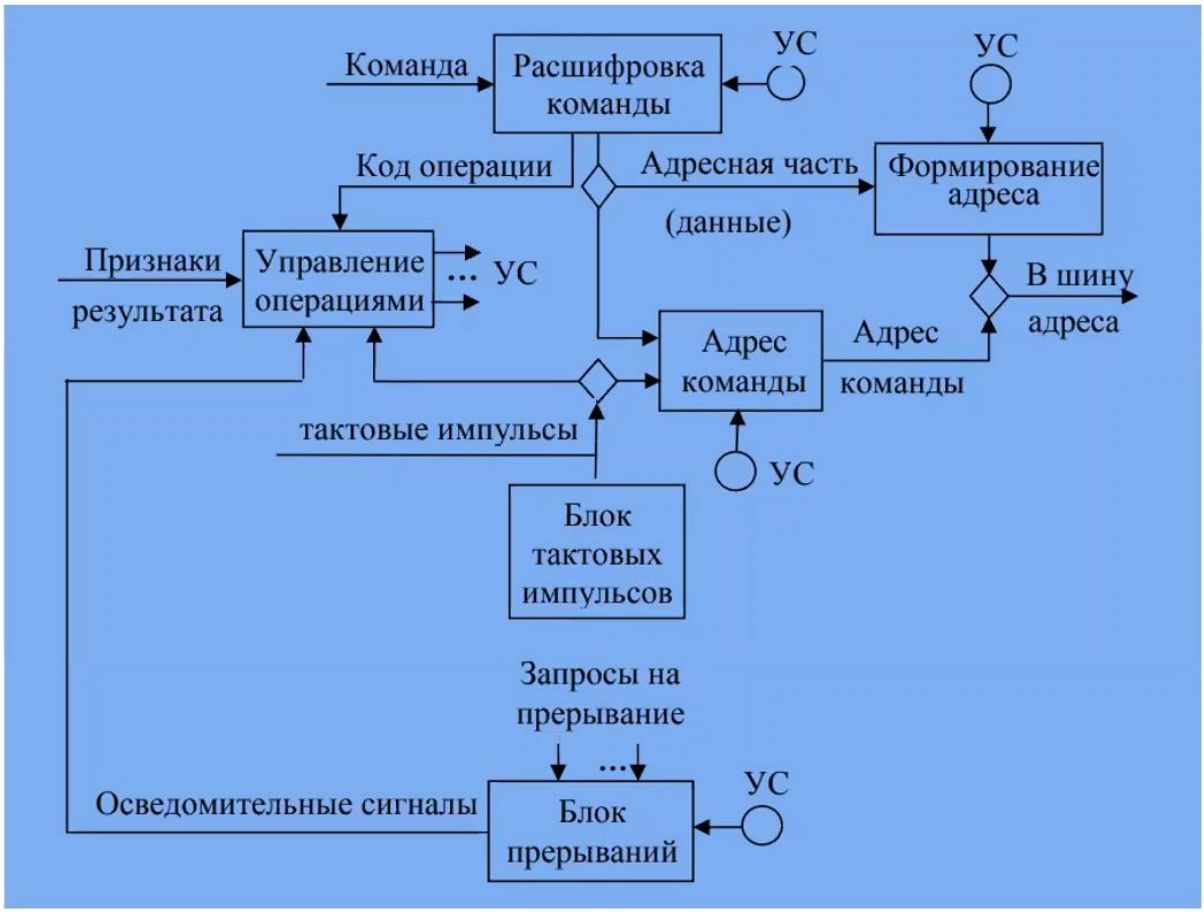
[Устройство управления. Жесткое и микропрограммное управление]

Задача устройства управления (УУ)

В общем случае УУ формирует управляющие сигналы для выполнения следующих функций:

•выборка из основной памяти (ОЗУ или ПЗУ) кода очередной команды;
•расшифровка кода операции и признаков выбранной команды;
•формирование исполнительного адреса операнда;
•выборка операндов и выполнение машинной операции;
•анализ запросов на прерывание исполняемой программы;
•формирование адреса следующей команды

Общая структура и функционирование устройства управления

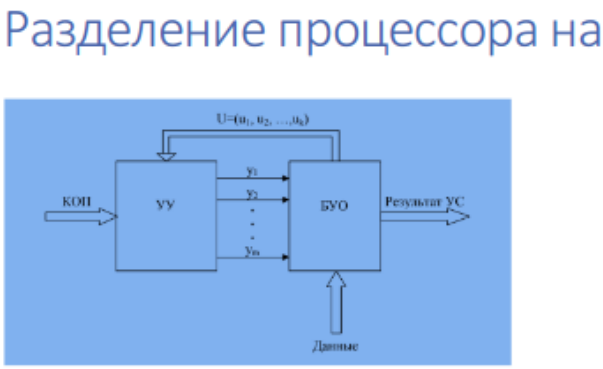


Команда поступает в блок, где расшифровывается команда. Туда же подаются управляющие сигналы. После расшифровки, из команды выделяются код операции и адресная часть (ведь команда состоит из кода и адресной части, но адреса может и не быть). Код операции подается на блок управления операциями, который используя признаки результата предыдущий команды формируент последовательность управляющих сигналов, помимо этого на этот блок подаются тактовые импульсы от блока тактовых импульсов. Также подаются осведомительные сигналы,

обеспечивающие обратную связь с другими устройствами, в частности, из блока прерываний (куда посылаются запросы на прерывания). Адресная часть команды и адрес следующей команды подаются в блок формирования адреса, и далее выставляется на шину адреса.

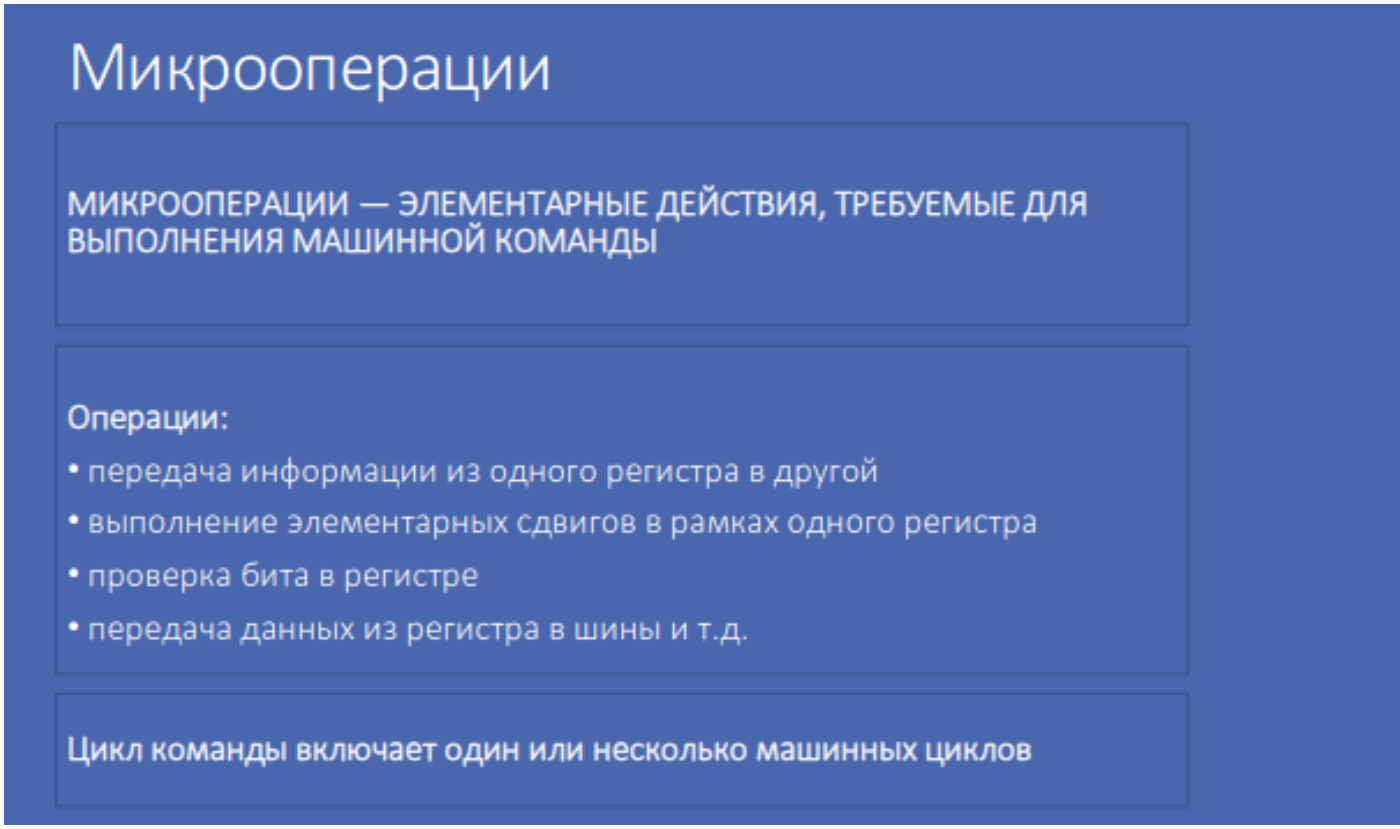
Разделение процессора на составляющие

Блок управления операциями, устройство управление, сигналы обратной связи, множество двоичных управляющих сигналов



Если процессор разделить на две составляющие части: блок управления операциями и устройство управления. Устройство управления получает код операции. Множество U, состоящее из u1 u2 u3… - сигналы обратной связи, которые используют УУ. УУ на выходе формирует множество управляющих сигналов, подающихся на блок управления операциями. Согласно этим управляющим сигналам и осуществляется данная операция

Микрооперации



Обработка состоит из микроопераций - элементарных действий, требуемых для выполнения машинной команды. В частности

- передача информации из одного регистра в другой
- выполнение элементарных сдвигов в рамках одного регистра
- проверка бита в регистре
- передача данных из регистра в шины и т.д.

Цикл команды - период времени за который выполняется одна машинная команды

Принципы построения УУ:

1. **Управляющие устройства с жёсткой логикой** - закон функционирования определяется способом соединения логических элементов. (когда при производстве ЭВМ заранее определяются все управляющие сигналы, и строят таблицу истинности для этих сигналов, синтезируют логическую схему. В конце выходит устройство УУ. Законы определяются заданной таблицы истинности. Имеет жёсткую логику, представляет собой некий набор логических элементов, заранее связанных между собой) - **обычно быстрое**. **Если обнаружился дефект или надо что-то заменить, то сделать это невозможно, ибо физическая**. Логическая схема, анализирующая набор признаков и с помощью блоков И и ИЛИ формирующая управляющие сигналы. То есть с общей шины снимается ряд признаков без инверсии и ряд признаков с инверсией, объединяются по схеме И и далее объединяются по схеме ИЛИ (результат всех и). Так можно сформировать почти любой управляющий сигнал. Можно добавить обратную связи (когда мы сигнал с выхода управляющего устройства либо других

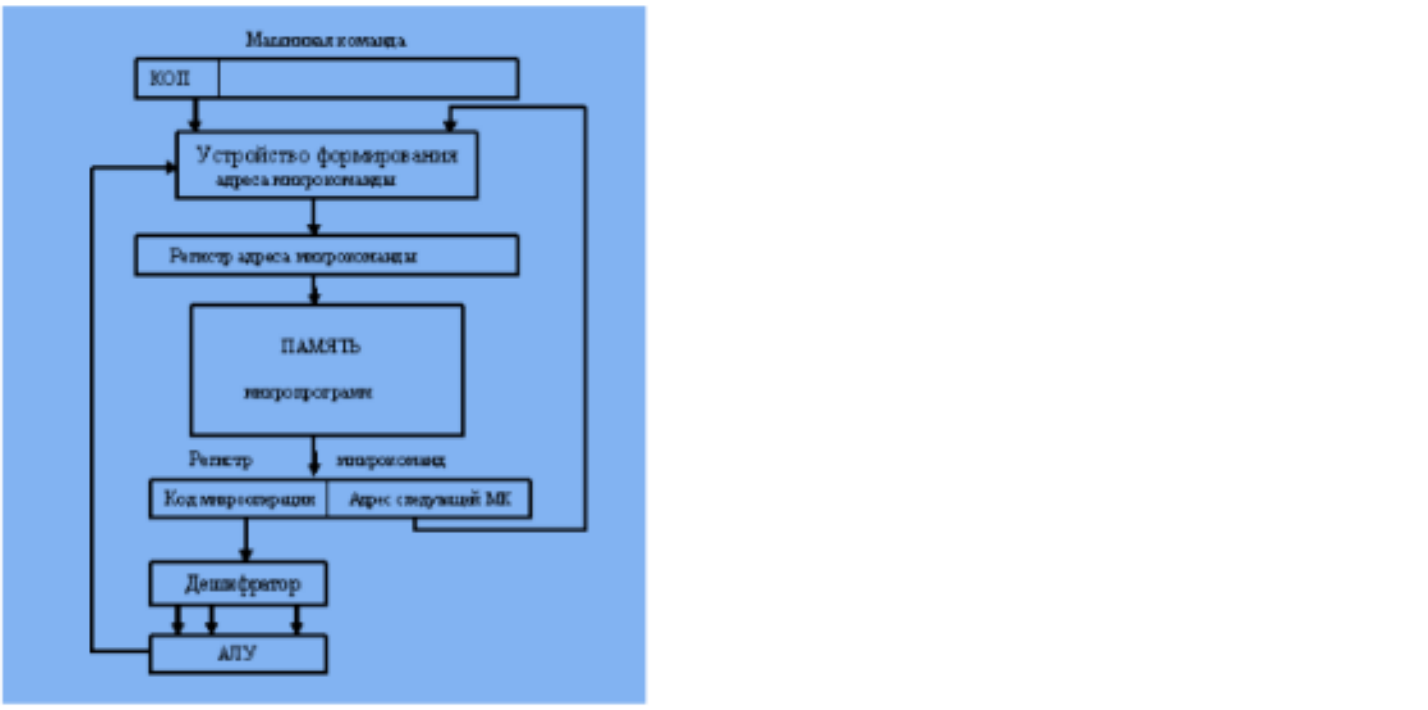
устройств также подаем на эту шину и используем в качестве признаков для формирования новых управляющих сигналов.) *[(таблица формирует устройство)]*

Устройства управления с жесткой логикой

2. **Микропрограммные устройства управления** (закон функционирования задается программой, хранимой в ячейках памяти, называемых управляющей памятью) - законы создаются программой. Лишено проблем УУсЖЛ (которое предыдущее) (можно делать перепрошивку, т.е. изменить содержимое управляющей памяти). **Легко модифицировать. + можно выпустить одно управляющее устройство, а потом внести новые программы = будет вести иначе** (разные управляющие сигналы выдавать). Массовое производство может быть дешевле, не смотря на **сложность.**
   Машинная команда раскладывается на две составляющих: код операций и . На основе кода операции формируется управляющие сигналы. Код операции из машинной команды поступает на устройство формирования адреса микрокоманды. Через специальный регистр (регистр кода операций) устройство формирования адреса использует адрес следующей команды из предыдущей команды. Помимо этого, использует сигналы обратной связи от АЛУ и других устройств, которые на него подаются совместно с кодом операции. То есть, когда только подаём новый код операции, предыдущей команды не было, тогда эти данные не юзаются, только код операции. В результате работы устройства формирования адреса микрокоманды, формируется адрес, который

запоминается в регистре адреса микрокоманды. По этому адресу из памяти микропрограмм достаётся микрокоманда. Адрес следующей команды не убирается, он передаётся на устройство формирования адреса. Микрокоманда состоит из двух частей: код микрооперации и адрес следующей микрокоманды. Всё это сохраняется в регистр микрокоманд, из неё выделяется код микрооперации, который подаётся на дешифратор. Задача дешифратора сформировать управляющие сигналы на арифметико-логическое устройство и все иные устройства в составе ЭВМ. По сути, могли бы хранить все управляющие сигналы в микрокоманде, подавая их непосредственно на все устройства, только поразрядно, но в этом случае у нас микропрограмма занимала бы много памяти, потому информация об управляющих сигналах хпанится в неком закодированным виде и для неё требуется дешифратор. Он упакованную информацию о всех необходимых управляющих сигналов превращает в управляющие сигналы. Далее подаёт на все устройства, от которых обратная связь поступает обратно, на устройство формирования адреса микрокоманды. Так осуществляется работа микропрограммного устройства управления. (т.е. Из кода машинной команды формируется последовательность адресов микрокоманд, по этим адресам из памяти извлекаются микрокоманды, дальше код микроопераций дешифруется каждый раз и подаются управляющие сигналы на все устройства ЭВМ)

Микропрограммное устройство управления



Лекция 8:

[Периферийные устройства ЭВМ]

Назначение периферийных (внешних) устройств

Периферийные устройства входят в состав внешнего оборудования ЭВМ, обеспечивают ввод/вывод данных, промежуточное и длительное хранение данных, передачу информации, но не определяют архитектуру и принципы функционирования ЭВМ.

Осуществляют связь ЭВМ с различными «поставщиками» и «потребителями» информации.

Многие устройства обеспечивают взаимодействие с ЭВМ на языке слов и десятичных чисел, а периферийные устройства производят кодирование (декодирование) информации, пересылаемой в/из ЭВМ, так как в ЭВМ информация обрабатывается в двоичном виде.

ОСНОВНЫЕ ФУНКЦИИ:

• хранение информации в том или ином физическом представлении на разных носителях данных

• преобразование информации соответственно функциям, выполняемым устройством

Упрощенная схема взаимодействия устройств в ЭВМ

В нижней части разделенной пунктиром показаны центральные устройства, в верхней периферийные. Центральные устройства - арифметико-логическое устройство (АЛУ), устройство управление (УУ), и оперативное запоминающее

устройство (ОЗУ). АЛУ и УУ вместе представляют собой центральный процессор.
К переферийным устройства относятся устройства ввода (УВв), устройства вывода (УВыв) и внешние запоминающие устройства (ВЗУ).
Сплошной линией изображены информационные потоки, в частности входная информация в ЭВМ подается через устройство ввода. От устройства ввода информационный поток поступает в оперативное запоминающее устройство (ОЗУ), с которым работаем арифметико-логическое устройство (АЛУ).
Выходная информация из ЭВМ поступает через устройство вывода, некоторые данные хранятся на внешних запоминающих устройствах.
Пунктирной линией изображены потоки управления от УУ к остальным устройствам.

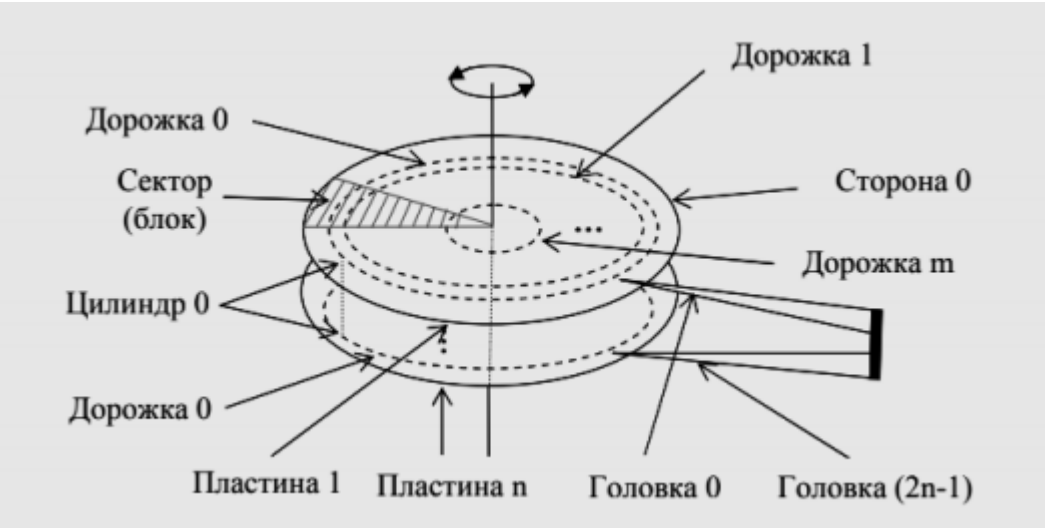
ОПЕРАЦИИ ОБМЕНА:
1. ЗАПИСЬ (ВЫВОД) — перенесение информации из оперативной памяти на внешнее устройство
2. ЧТЕНИЕ (ВВОД) — перенесение информации из внешнего устройства в оперативную память

Классификация внешних устройств по назначению
• устройства для связи «человек – ЭВМ» (устройства ввода, устройства вывода, интерактивные устройства)
• устройства массовой памяти (внешние запоминающие устройства большой емкости) - для хранения больших объём информации
• устройства для связи с объектами управления (датчики, реле, устройства преобразования непрерывных сигналов с датчиков в цифровые сигналы и обратного преобразования, и т.д.)
• средства передачи данных на большие расстояния (средства телекоммуникации)

ПО ВЫПОЛНЯЕМЫМ ФУНКЦИЯМ:
• средства ввода-вывода информации
• средства хранения информации
• средства телеобработки (коммутации и приема-передачи информации)

Жесткие магнитные диски
Жесткий диск состоит из одной или нескольких стеклянных или металлических пластин, каждая из которых покрыта с одной или двух сторон магнитным материалом, помещенных в герметичный корпус.

Схема на рисунке. Жесткий диск состоит из пластин, магнитных головок, которые перемещаются по поверхности жд. На каждом из дисков есть дорожки с информацией. Совокупность дорожек на пластинах - цилиндры. Каждая дорожка сегментирована на секторы. Сектор - минимально адресуемая единица в пределах жд. Жд вращается, по нему перемещаются головки, производится считывание или записи. Чтение или запись проводится путём намагничивания или считывания намагниченности конкретных сегментов жёсткого диска.



1956 год ВЫПУЩЕН ПЕРВЫЙ ЖЕСТКИЙ ДИСК
• IBM — модель 305 RAMAC
• 5 мегабайт
• 970 килограмм
• 8.8 байта в секунду
• 10 000$
• 1000 изделий продано
• в 1961 снят с производства

Учитывая, что дорожки разного радиуса имеют разную длину, в зависимости от того насколько близка дорожка к внутренней границе. Поэтому возможно размещение на разных дорожках различного количества секторов

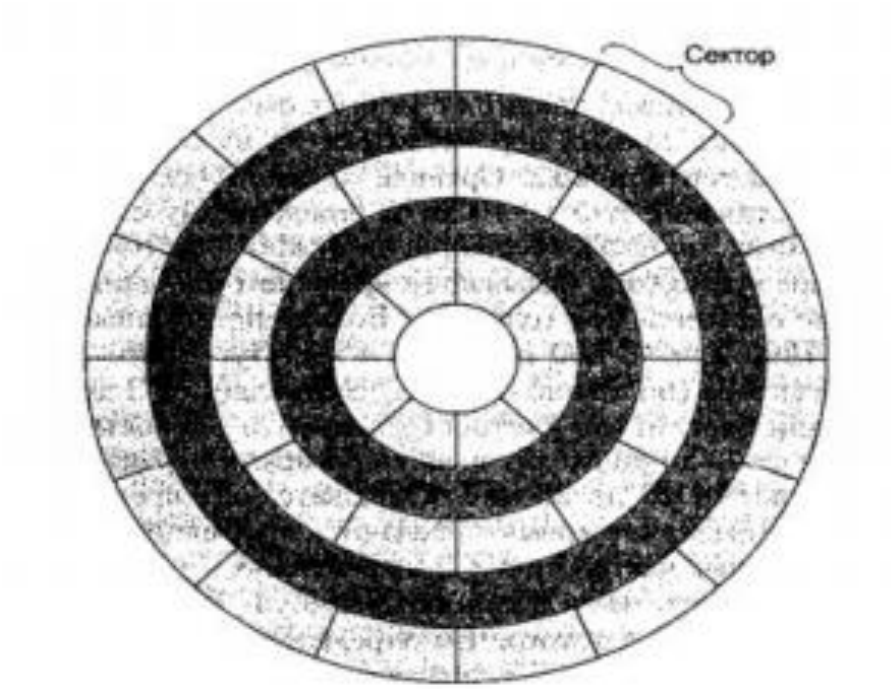
Сектор — наименьшая адресуемая единица обмена данными

Для того чтобы контроллер мог найти на диске нужный сектор, необходимо задать ему все составляющие адреса сектора: номер поверхности, номер цилиндра (совокупность дорожек) и номер сектора в пределах этого цилиндра.

За одно обращение считывается целый сектор ЖД.

ЖД относятся к носителям информации с произвольным доступом, так как сообщим данный адреса, мы считываем конкретный сектор.

В современных дисковых устройствах дорожки (цилиндры) делятся на зоны



Полезный объём диска обычно меньше на 15% заявленного. Не все секторы накопителей используются в качестве рабочих. Часть секторов являются запасными

В процессе работы ЖД часть секторов выходит из строя. Контроллер переносит инфу с тех секторов, которые плохо читаются на запасные сектора и размещает данные о перемаркировки специальной инженерной зоне ЖД. Т.к. эта информация очень важна, то таких мест где размещаются данная информация несколько (от 2 до 6 инженерных зон (в зависимости от производителя)).

Размер отформатированного жесткого диска меньше неразмеченного (на 15%) вот почему. Скорость вращение диска является не равномерной. Да и сам принцип (мы намагничиваем определенный участок и считываем намагниченность) требует определенного интервала между полезными участками информации на поверхности жесткого диска, помимо этого на диске есть прединдексны интервал, который компенсирует неравномерность скорости вращения диска

При первоначальной разметке дисков на заводе изготовителе производится проверка поверхности диска, и информация об обнаруженных дефектных участках записывается в таблицу дефектов, которая размещается в инженерной зоне.

В процессе функционирования винчестера эта таблица используется для переназначения (переадресации) обращения к дефектным участкам (секторам) на обращение к хорошим секторам, которые как раз и размещаются на запасных дорожках

Ввиду важности служебной информации инженерная зона различных моделей дисков может содержать от 2 до 6 копий

Часто производители указывают размер неформатированного (неразмеченного на сектора) диска, как будто дорожки содержат только данные. В действительности, каждый сектор несет не только данные, но и служебную информацию:
1. Блок данных со служебной информацией
2. Интервал между записями (секторами), необходимый для того, чтобы застраховать следующий сектор от записи на предыдущий, из-за неравномерной скорости вращения диска. (сам принцип записи на ЖД - намагничиваем и считываем - требует интервала между полезными участками инфы)
3. Прединдексный интервал — для компенсации неравномерности скорости вращения диска. Емкость форматированного диска обычно на 15% меньше емкости неформатированного

Структура блока данных со служебной информацией:
1. Заголовок (prefix), включающий идентификатор (ID) (информацию о номере цилиндра, головки и сектора - полный адрес) и первую контрольную сумму (CRC)
2. Интервал включения записи
3. 512 байт данных
4. Заключение (suffix), включающее вторую контрольную сумму (CRC)
Контрольная сумма - в простейшем случае, подразумевается, скажем, есть у нас байт инфы, 8 бит, мы можем подсчитать количество единиц в нём, если нечётное - дописать 1. Таким образом всегда чётное. Если вышло нечётное, то - недействительная. На самом деле куда сложнее.

Общие характеристики
Интерфейс — набор линий связи, сигналов, посылаемых по этим линиям, технических средств, поддерживающих эти линии, и правил обмена (часто это SATA, бывают microSATA, USB).
Физический размер (форм-фактор) — размер накопителей для ПК и серверов 3.5" либо 2.5" (3.5 - это сама пластина, а сам он на деле 3.9)
Надежность — среднее время наработки на отказ (исправляется рейд-массивами)
Сопротивляемость ударам — измеряется в единицах допустимой перегрузки во включённом и выключенном состоянии.
Скорость вращения шпинделя — количество оборотов шпинделя в минуту (4200, 5400, 7200, 10000 и 15000 об./мин. [10000 и 15000 юзаются, в основном, в серверном оборудовании. У таких ещё малая сопротивляемость ударам]). От

этого параметра в значительной степени зависят время доступа и скорость передачи данных.

Увеличению скорости вращения шпинделя в винчестерах ноутбуков препятствует гироскопических эффект, влияние которого крайне мало в неподвижных компьютерах. Обычно не превышает 5400 оборотов в минуту.

Производительность жестких дисков
Время поиска – требуемое для перемещения с произвольной дорожки до заданной, но без готовности чтения; среднее время поиска между дорожками составляет от 5 до 10 мс, между смежными — менее 1 мс.
Время установки головки – необходимое для стабилизации вибраций головки в конце этапа поиска.
Время задержки из-за вращения диска (время ожидания сектора) – требуемое головке чтения для поворота от произвольного до требуемого сектора на той же дорожке (типовое время задержки из-за вращения диска составляет около 4 мс)

2019 год Seagate продемонстрировала жесткий диск Exos с технологией MACH.2 Multi Actuator, который обеспечивает скорость передачи данных до 480 МБ/с
Поэтому показателю накопитель вплотную приблизился к SSD и обгоняет жесткие диски со скоростью вращения шпинделя 15 000 об/мин примерно на 60%

Время произвольного доступа – за которое выполняется операция позиционирования головки чтения/записи на произвольный участок магнитного диска, то есть время, необходимое для перемещения головки чтения с текущей дорожки до начала считывания данных из заданного сектора
Время передачи информации — зависит от плотности записи и скорости вращения (чем выше плотность и скорость, тем лучше)
Объем буфера промежуточной памяти, (сюда осуществляется предварительное считывание с жд и запись на жд) предназначенной для сглаживания различий скорости чтения/записи и передачи по интерфейсу
Позволяет повысить скорость работы жд, так как жд механический, скорость не всегда линейна, а буферизация позволяет сгладить различия скорости чтения/записи и передачи по интерфейсу.
Закон Мура справедлив и для жестких дисков. Каждый год максимальная ёмкость накопителя увеличивается в два раза - данное правило действует уже много лет

Оптические диски
Немагнитные внешние запоминающие устройства
Запоминание и поиск информации реализуется оптическими средствами. Применяются полупроводниковые лазеры и оптические системы, которые генерируют очень маленькую световую точку, фиксируемую на тонком слое среды диска для выборки бита информации; интенсивность отраженного луча соответствует значениям 0 или 1. Запись идёт по спирали. Лазер перемещается и считывает совокупность 0 и 1, после передает их в эвм.
Характерна более высокая плотность записи, чем у магнитных устройств. Т.к. нет конфликта между соседними секторами.
Не требуют плотного контакта между носителем и считывающей головкой

Первый компакт диск был представлен в 1980 году компаниями Philips и Sony.

Информация на оптическом диске записывается на одну спиралевидную дорожку, содержащую чередующиеся участки с различной отражающей способностью. Чувствительный слой находится под прозрачным защитным покрытием.
При работе CD и DVD устройств используется красный лазерный луч. Оптические устройства, основанные на работе с синим лазером, имеющим меньшую длину волны, чем красный, имеют значительно большую емкость (HP DVD и Blu-Ray)

Первым компакт-диском поступившим в музыкальные магазины стал альбом Билли Джоэла 1978 года 52nd Street. Продажи CD с этой записью начались в Японии 1 октября 1982 года.

Flash-память

Энергонезависимый тип памяти, позволяющий записывать и хранить данные в микросхемах, помещенных в миниатюрный корпус. В основном, не требуется корпус. Диски такие представляют плату с набором микросхем, которая подсоединяется в спец. разъём.
Карты flash-памяти не имеют в своем составе движущихся частей (преимущество перед классических жд), что обеспечивает высокую сохранность данных [при их использовании в мобильных устройствах]

Твердотельные накопители SSD:

NAND SSD. Накопители, построенные на использовании **энергонезависимой** памяти. Характеризуются относительно небольшими размерами и низким энергопотреблением. Высокая скорость чтения и записи, и так далее.
RAM SSD. Накопители, построенные на использовании **энергозависимой** памяти. Содержат в себе аккумулятор или батарейку, которые позволяют длительно хранить информацию. Характеризуются сверхбыстрыми чтением, записью и поиском информации. Основным их недостатком является чрезвычайно высокая стоимость и необходимость наличия аккумулятора. Гибрид между оперативной памятью и аккумулятором.

Преимущества SSD solid state drive
1. Отсутствие движущихся частей, отсюда:
• полное отсутствие шума (уровень шума — 0 дБ)
• высокая механическая стойкость
• стабильность времени считывания файлов вне зависимости от их расположения или фрагментации - адресуемые микросхемы, не нужно перемещаться между дорожками, можно произвольно считывать.
• высокая скорость чтения/записи
• скорость выполнения операций (IOPS – количество операций ввода-вывода в секунду) выше в десятки тысяч раз, чем у жесткого диска
• низкое энергопотребление - не надо питать энергией механические части, а только на микросхемы.
• широкий диапазон рабочих температур;
• большой модернизационный потенциал - возрастает объём, скорость и их применяемость возрастают.

2. Отсутствие магнитных пластин, отсюда:
• намного меньшая чувствительность к внешним электромагнитным полям
• малые габариты и вес (нет необходимости делать увесистый корпус для экранирования)

Недостатки SSD
1. Главный недостаток NAND SSD — ограниченное количество циклов перезаписи. Для борьбы с неравномерным износом применяются схемы балансирования нагрузки. Контроллер хранит информацию о том, сколько раз какие блоки перезаписывались и при необходимости «меняет их местами» (более частые на менее частые). Данный недостаток отсутствует у RAM SSD
2. Проблема совместимости SSD накопителей с устаревшими и даже многими актуальными версиями ОС, которые не учитывают специфику SSD накопителей и дополнительно изнашивают их (например, использование операционными

системами механизма свопинга (подкачки) на SSD с большой вероятностью, уменьшает срок эксплуатации накопителя
3. Цена гигабайта SSD-накопителей существенно выше цены гигабайта накопителя на жёстких магнитных дисках - HDD. К тому же, стоимость SSD прямо пропорциональна их ёмкости, в то время как стоимость традиционных жёстких дисков зависит не только от количества пластин и медленнее растёт при увеличении объёма накопителя. Основная стоимость - микросхемы.
4. Применение в SSD-накопителях команды TRIM делает невозможным восстановление удалённой информации системными утилитами. TRIM — команда, позволяющая операционной системе уведомить твердотельный накопитель о том, какие блоки данных уже не содержатся в файловой системе и могут быть использованы накопителем для физического удаления.
5. Невозможность восстановить информацию при перепаде напряжения. Так как контроллер и носитель информации в SSD находятся на одной плате, то при превышении или перепаде напряжения чаще всего сгорает весь SSD носитель с безвозвратной гибелью информации. Напротив, в жёстких дисках чаще сгорает только плата контроллера, что делает возможным восстановление информации с приемлемой трудоёмкостью. Вообще, если произошёл аппаратный отказ SSD из-за выхода из строя чипа контроллера или флеш-памяти, это делает процесс восстановления информации практически неосуществимым

Ввод-вывод информации

Среди устройств можно выделить:
• средства обмена с документами
• средства непосредственного взаимодействия человека с ЭВМ
Параметры классификации устройств обмена:
• тип информации (текстовый, графический)
• функциональное назначение (ввод, вывод)
• степень автоматизации процесса ввода
• тип носителя информации (печатный документ, электронный документ

[Периферийные устройства ввода ЭВМ]

Устройства ввода информации

[По той картинке которую до этого была, там был поток входной информации, который поступал на устройства ввода]

• **Клавиатуры (проводные и беспроводные)**
• **Мыши (проводные и беспроводные)**
• **Сканеры** - которые позволяют переносить бумажную инфу в электронный вид
• **Графические планшеты** - ввод графической информации в эвм
• **Пенмаусы (аналог шариковой авторучки**, только шарик вращаясь передает инфу на датчик, который распознает пройденный путь и направление вращения)
• **Световые перья** (небольшое световое пятно позволяет отследить перемещение устройства ввода)
• **Цифровые видеокамеры и фотоаппараты, веб-камеры**
• **Джойстики**
• **Сенсорные панели (тачпады)** (характерны для наутбуков, представляют из себя поверхность, воздействие на которые передают информацию в эвм.)
• **Средства речевого ввода** (набирают популярность, окей гугл, алиса, сири)

[Периферийные устройства вывода ЭВМ]

Устройства вывода информации

• **Мониторы (ЭЛТ, ЖК, LED, OLED, qLED)**
(ЭЛТ - встречается мб только в музеях, электронная лучевая трубка. ЖК - жидкие кристаллы.
LED - не следует путать с LED подсветкой. каждый пиксель представляет из себя трехцветный светодиод. Он сам себя подсвечивает.
OLED - органические светоизлучающие диоды.
qLED - мониторы на основе квантовых точек совмещенных с жидкими кристалами)

• **Сенсорные экраны (резистривные** - меняется сопративление, **емкостны**е - меняется емкость(в современных тел юзается)) (позволяет как вводить так и выводить инфу)
• **Синтезаторы звука (спикер** (простейший), **бытовые** (звуковые карты), профессиональные (звуковые карты)) (звуковые карты позволяют обеспечить качественную работу со звуком.)
• **Проекторы (LCD, DLP, LCoS, CRT)**
(LCD - жк матрица из разноцветных кристалов, оч популярны.
DLP - в основе лежит система из микрозеркал, которые управляются микропроцессором, каждое микрозеркало - один пиксель. Достоинства - высокая контрастность картинки, глубокий уровень детализации, надежны. Недостатки - высокая стоимость.
LCoS - комбинация LCD и DLP, свотовой луч отражается от ЖК матрицы, позволяет избежать сетку характерную для LCD.
CRT - в основе лежит катодно анодная трубка. Большие габориты.
• **Принтеры (ударные, струйные, электро и магнито-графические, электростатические, термические)**
ударные - часто называют матричные. Когда с помощью иголок выставляется определенный символ и он пробивает через красящую ленту на бумагу, оставляя там соответствующий символ. В быту почти не используются. Самая низкая стоимость печати
Струйные - через определенную матрицу наносятся чернила на бумагу, высокое качество.
[Электро и магнито-графические - ничего не сказал]
лазерные - смесь электростатических и термических, активно используется в быту черно белые принтеры. Когда на барабан с помощью лазера проецируется изображение, которое нужно напечатать, дальше обеспечивается намагниченность или электростатичность изображения, к которому прилипает краска. Потом она запекается после термической обработки. Лазерный принтер в большинстве случаев печатает пластиком.
• **3D принтеры (FDM, печать пластиком), SLA (фотополимерная печать), SLS (печать полиамидом), SLM (печать металлом)**
FDM - печать пластиком, который выглядит как катушка, которая поступает в принтер. Далее расплавляется и механически подается на поверхность. Достаточно дешевые. Расходный материал тоже оч дешевый. Поэтому распространены.
SLA - печать осуществляется специальной жидкостью, на которую воздействуют световые лучи. Под ними жидкость затвердевает. Высокая разрешение. Значительно дороже. Позволяет печатать небольшие модели.
SLS - принтер печатает порошком, который отвердивает.\

SLM - печать металлом, на проф производствах юзается.
• Плоттеры (векторного, растрового типов, планшетные, барабанные, перьевые, струйные, электростатические, термические, карандашные)
Самая дешевая печать на плоттере - карандашные, когда грифель закрепляется, раскручивается барабан и позволяет этим “карандашом” воспроизводить изображение.
Плоттеры используются на производстве, для воспроизведения чертежей. Больших плакатов и т.д. на бумажный носитель большого размера. Они бывают барабанные, которые позволяют распечатать оч длинное изображение. И бывают на определенный лист. Грубо говоря принтеры оч большого размера.

Жидкокристаллический монитор (LCD)

Представляет собой следующую структуру. С двух сторон ограничено стеклом. Ближе к пользователю стекло оснащено поляризационным фильтром и цветовым фильтрам. Также требуется источник освещения, который проходит через поляризационную пленку.
ЖК экран состоит из крошечных сегментов со специальным веществом, способным менять отражательную способность под воздействием очень слабого электрического поля, создаваемого электродами, подходящими к каждому сегменту

Устройства обмена данными (средства телекоммуникации)
• модем (модулятор-демодулятор) (внешние, внутренние, встроенные)

Представляют собой модулятор - демодулятор. Позволяют обеспечить передачу данных в совершенно разной среде. Бывают проводные и беспроводные. Беспроводные у нас в телефонах 3g, 4g и т.д. Есть модемы работающие по технологии в аймакс. Классические работали по телефонной линии и позволяли передавать данные по ней. Задача модема обеспечить механизм передачи информации в определенной среде. Над входными данными демодуляция, над выходными модуляция.

• сетевой адаптер

Позволяет подключить эвм к сети. Есть разные технологии. Есть сетевые адаптеры, которые адаптируют передаваемую инфу к медным проводникам, и там есть разные технологии(100 МБ, 1 ГБ, 10 ГБ все это использует разные технологии). Еще есть оптическая среда передача. Сетевой адаптер осуществляет адаптацию к оптической линии связи.

• Wi-Fi-адаптер (2.4 ГГц, 5 ГГц)

Достоинством 2.4 ГГц - больше расстояние.

Достоинством 5 ГГц - больше скорость.

Недостаток 5 ГГц - меньшее расстояние скорее уже является достоинством, потому что вайфай точек много, и соответственно они перекрывают друг друга.

• повторитель (регенерация цифрового сигнала, затухающего в среде)

Будь то оптическая, воздушная, где с помощью оптических волн передается информация, либо медные провода. Через определенное время сигнал искажается и необходимо восстанавливать его.

• концентратор (многопортовый повторитель)

Передает сигнал во всех направлениях, почти не используется из-за низкой скорости (из-за большого количества подключений снижается пропускная способность)

• сетевой мост (обеспечивает программную адресную передачу на уровне фреймов)

Поступила инфа от одного компа и он определяет кому она адресована и именно ему передает.

• коммутатор (switch, переключатель) (обеспечивает аппаратную адресную передачу данных)

Сетевой мост и коммутатор - близкие устройства, ток коммутатор - аппортно это делает. Широко распространены. Есть под разные технологии даже медные.

Лекция 9

[Микропроцессор i8086 (К1810)]

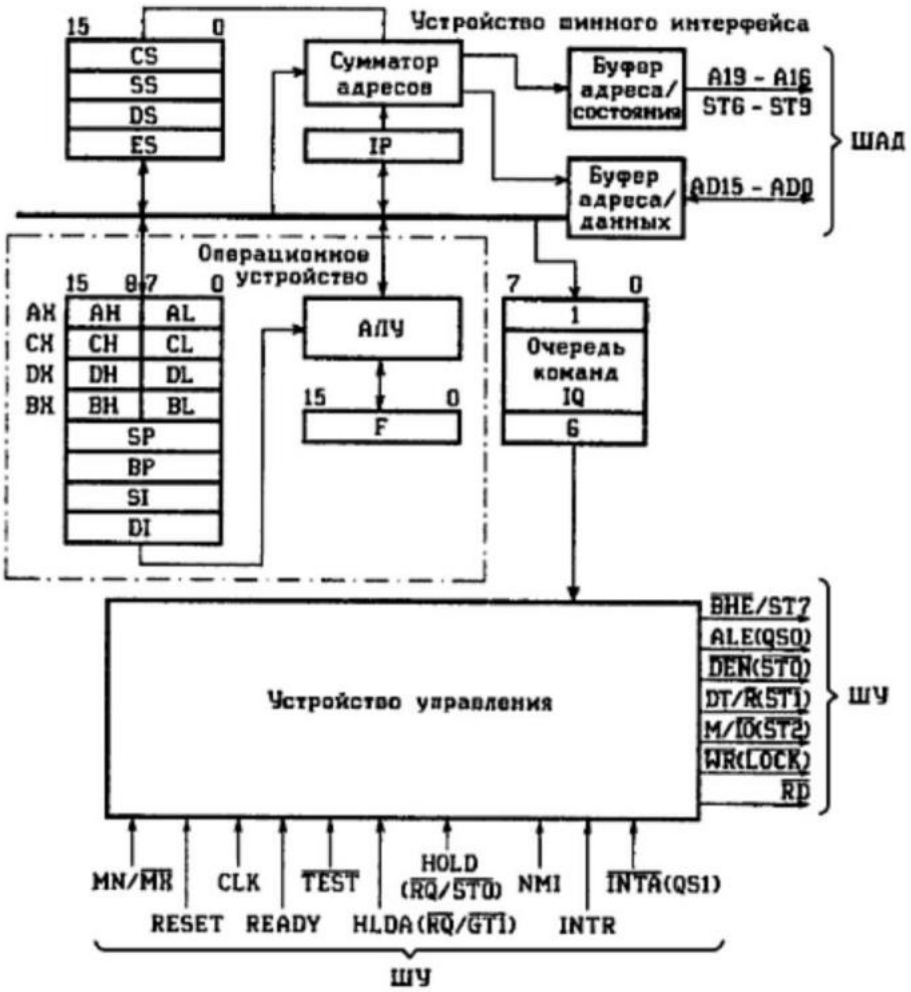
[Внутренняя структура процессора i8086]

Особенности i8086

• CISC архитектура
• 20 адресных линий
• может адресовать 2 Мб оперативной памяти
• *поддерживает прямой доступ к памяти*
• *обмен по запросу на прерывание*
• внутренние регистры 16-ти разрядные
• *поддерживает конвейер команд* (который скорее очередь команд)
• тактируется от внешнего генератора

Разработала команда интел. Началась разработка в 1976 году и был положен в основу IBM PC.

Внутренняя структура



Условно можно разделить на 3 блока:

Операционное устройства, куда входят АЛУ, регистры общего назначения, регистр флагов, который содержит признаки выполнения предыдущей операции, либо управляющие инструкции для процессора, как выполнять следующие операции.
Конвейер команд здесь представлен очередью команд. Очередь может иметь до 6 команд.

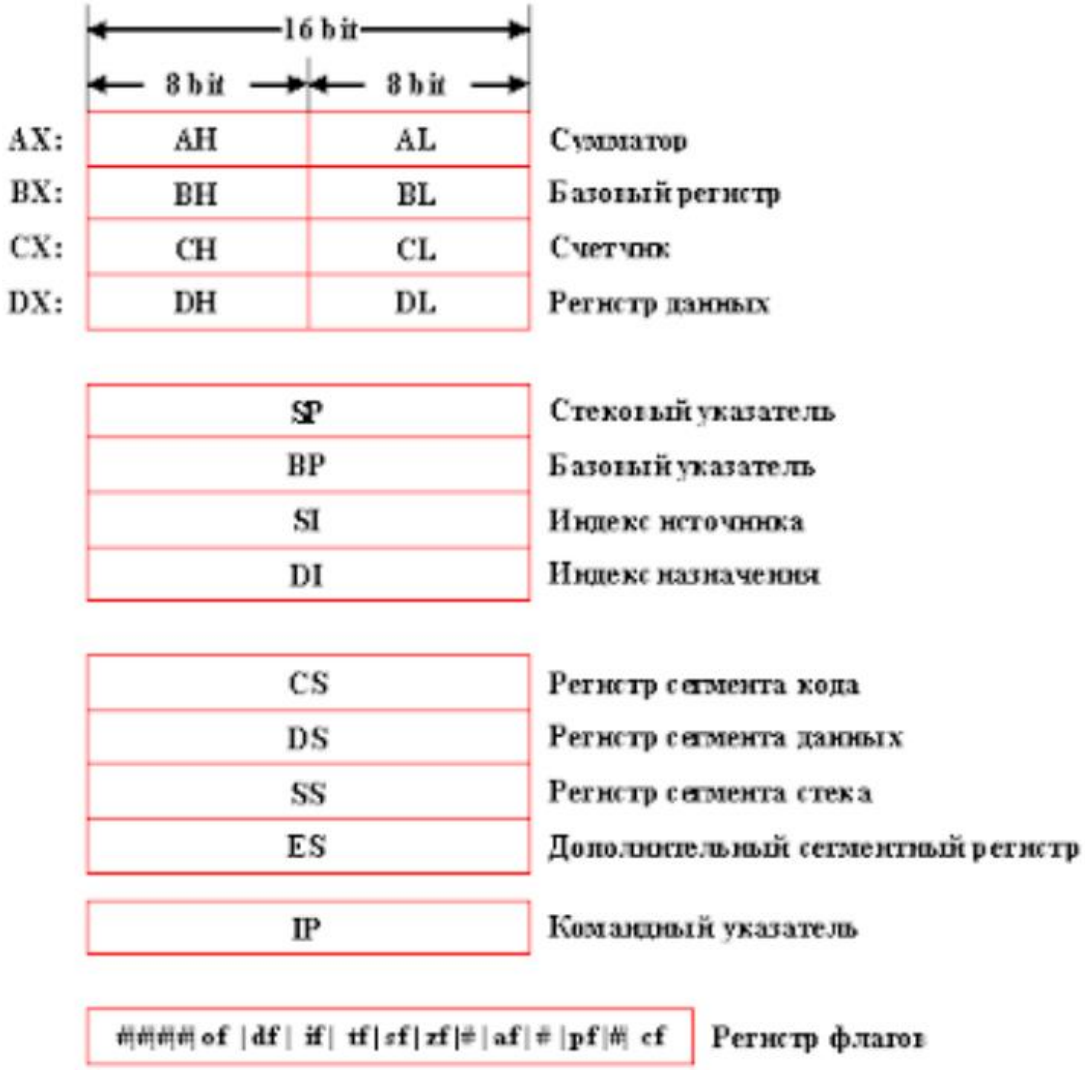
Сверху над операционным устройством находится устройство шинного интерфейса. Обеспечивает сопряжение процессора с внешней средой. В частности шинный интерфейс занимается формированием адресов (сумматор адресов, которому подается информация с регистра IP - счетчик команд в которой хранится номер текущей команды). Слева шестнадцатиразрядные сегментные регистры, которые определяют расположение в памяти соответствующего сегмента. CS - код, SS - стек, DS - данные, ES - дополнительный сегмент. Также имеются буферные регистры на выходе. (Совмещенная шина адреса и данных).

Устройство управление находится снизу. Содержит множество входов и выходов, которые совмещаются в шину управление.

Получается шина адреса и данных совмещены. А шина управления отдельно.

Это был первый 16-битный процессор с 16-битным ALU, 16-битными регистрами, внутренней шиной данных и 16-битной внешней шиной данных, что привело к более быстрой обработке.

Программная модель процессора



Под программной моделью подразумевается его внутренние регистры. У первых четырех регистров есть особенность. Они могут использоваться как один шестнадцатиразрядный, либо как два восьмиразрядных.

AX - сумматор, он же аккумулятор у нас в курсаче. Он выступает в качестве приемника результата и часто в качестве второго операнда.
BX - базовый регистр, используется для базовой адресации
CX - счетчик, используется в циклах
DX - регистр данных
SP - стековый указатель, хранит адрес вершины стека, соответственно с SS (регистром сегмента) позволяет формировать адреса в стеке
BP - базовый указатель, в базово-индексной адресации хранит второй из трех слагаемых. (индекс)
SI - индекс источника при различных способах адресации
DI - индекс приемника назначения

Основная функция сегментных регистров CS, DS, SS, ES формирование различных адресов
CS - хранит начальный адрес сегмента кода в памяти
DS - хранит начальный адрес сегмента данных в памяти
SS - хранит начальный адрес сегмента стека
ES - хранит начальный адрес дополнительного сегмента данных

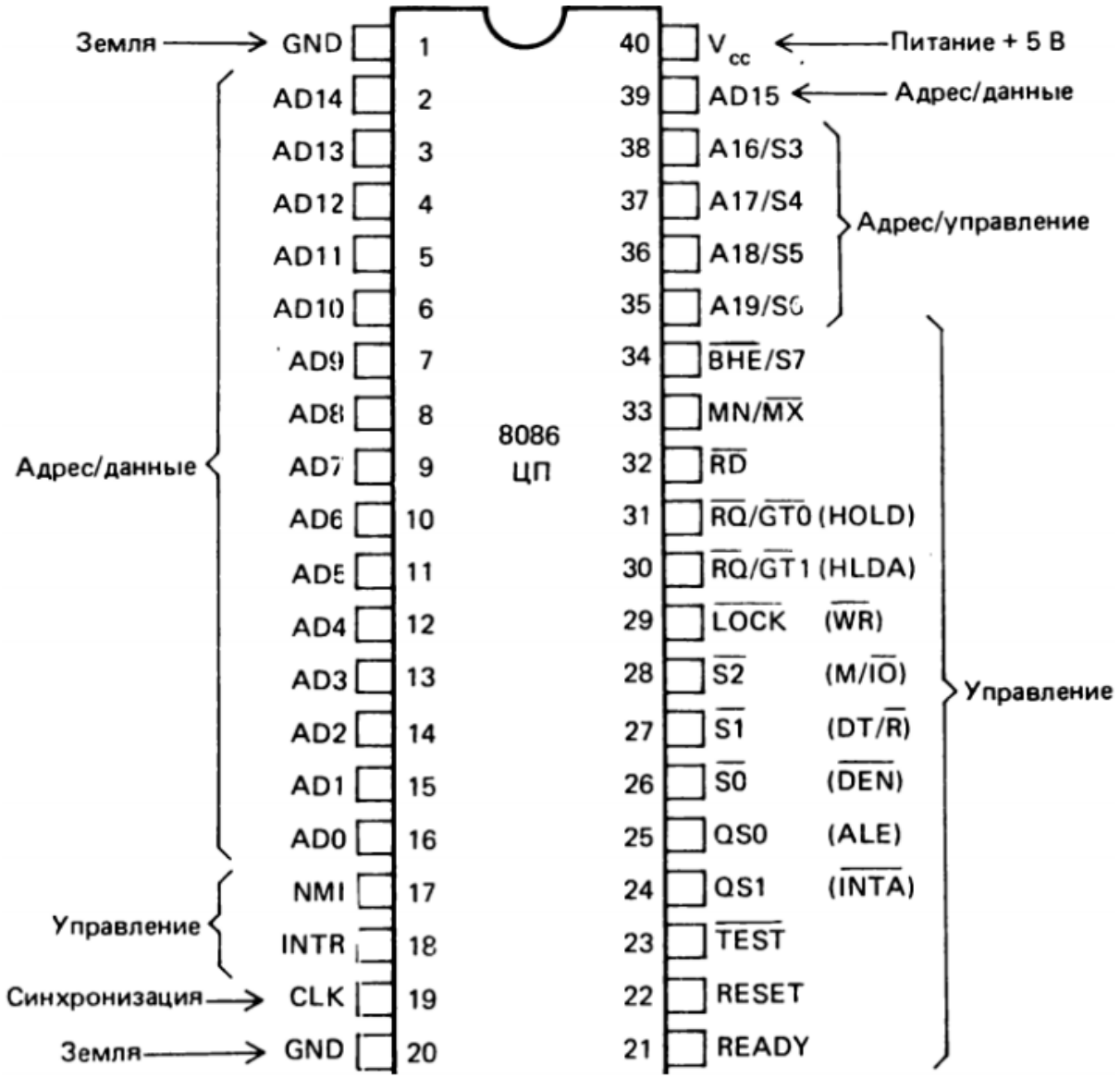
IP - командный указатель. Указывает на текущую выполняемую команду.

Формирование адреса происходит следующим образом: содержимое сегментного регистра сдвигается на 4 разряда и к этому прибавляется смещение. При формировании различных адресов юзаются содержимое разных сегментных регистров и разные регистры используются для смещения.
Например для получение адреса следующий команды необходимо содержимое регистра сегмента кода сдвинуть на 4 разряда влево (тоже самое, что умножить на 16) и прибавить значение командного указателя.
Для обращение к адресу вершины стека в берется SS сдвигается на 4 разряда и прибавляется SP
Для обращения к данным DS или ES, сдвигается на 4 разряда, в качестве смещение используется DI или SI.
[Там в 10 лекции есть более подробная табличка]

Регистр флагов
Каждый бит регистра флагов отвечает за определённый признак предыдущей операции или управляющая инструкция, определяющая поведение процессора.
Состоит из 16 бит, из них используется только 9, остальные зарезервированные.

Признаки предыдущей операции
CF — содержит перенос(заем) из старшего бита после арифметических операций и сдвигах
PF — показывает четность младших восьми битовых данных (1 – четное, 0 – нечетное) (по сути, дублирует младший бит и инвертирует его)
AF — признак меж тетрадного переноса (из 3 бита) Если осуществляется перенос или заем в третьем бите тетрады (от слова тетрада - 4 бита), то выставляется в 1. Используется в двоично-десятичных операциях.

ZF — флаг нулевого результата. (если вышло 0, то флаг 1 и далее можно строить работу по нему. В курсаче для условных переходов, опираясь на этот флаг.)
SF — флаг знака (совпадает со старшим битом результата, 0 – плюс, 1 - минус).
Флаги, которые влияют на режим работы процессора
TF — флаг пошагового режима (используется при отладке)
IF — флаг разрешения аппаратных прерываний. Разрешает(1) или запрещает(0) маскируемые прерывания (немаскируемые прерывания нельзя программно запретить. Юзаются в
критических ситуаций, процессор всегда их обработает. А маскируемые только при наличии разрешения на обработку. Как раз этот флаг разрешает или нет)
DF — флаг направления при строковых операциях. Обозначает левое или правое направление
OF — указывает на переполнение старшего бита (есть в курсаче команда условного перехода, но у некоторых вариантов).
Входы и выходы процессора (микросхема)



Поддерживает 2 режима минимальный и максимальный, результат (всм текущий режим) в 33 MN/MX (1 минимальный, 0 максимальный)
Например в максимальной используются S3, а в минимальном A16.
Мы рассматриваем в минимальном режиме.
Макс режим юзается в спец устройствах. Например если юзаем несколько процессоров таких в объеденинении.

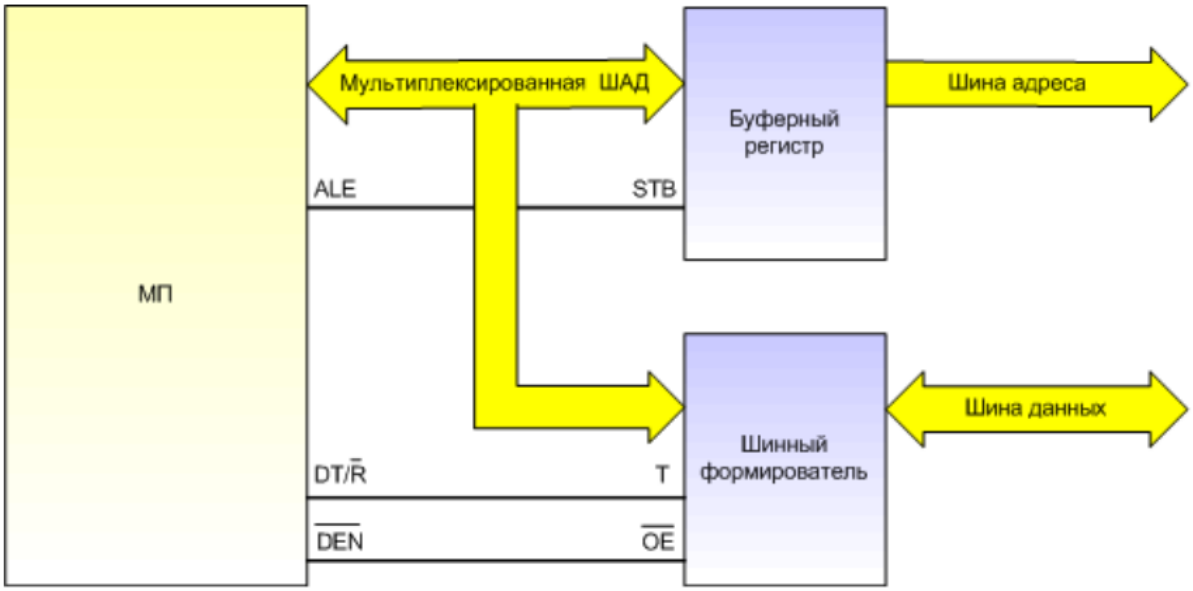
Также уточню, что если сверху черта, то это означает, что 0 означает логическую единицу. (то бишь сигнал на выходе инвертирован)
Входы процессора [Назначение входов процессора i8086]
CLK — вход тактовых импульсов, задающих частоту процессора
NMI — немаскируемые запросы на прерывания (нельзя остановить, процессор всегда будет обрабатывать)
INTR — вход маскируемых запросов на прерывания (проверяет флаг IF)

RESET — аппаратный сброс процессора (обнуляет внутренний регистр и возвращается на начальный адрес)
READY — готовность внешнего устройства или памяти к обмену данными (процессор сообщает устройству, что надо выполнить обмен данными и переходит в режим ожидания этого флага. Из этого минимизируется шанс потери данных. (Обмен по запросу))
TEST — входной сигнал проверки. Используется в сочетании с сопроцессорами. (процессор ожидает окончания работы сопроцессора. Если есть математический сопроцессор, то часть операций проводит процессор, часть сопроцессор. Когда поступает сопроцессору инструкция на тест, процессор на входе TECT ожидает окончания операции и потом сам подключается и выполняет)
MN/MX — задает режим работы процессора (минимальный или максимальный)
HOLD — вход запроса на прямой доступ к памяти (сюда внешние устройства подают запросы, если разрешено, то подтверждает)
+5 — питание (+5В)
GND — заземление (отрицательный контакт питания)

Выходы процессора [Назначение выходов процессора i8086]

AD0-AD15 — совмещенная шина адреса и данных
A16-A19 — старшие разряды шины адреса
BHE — для разрешения подключения старшего банка памяти (16-ти разрядный или 8-ми) (этот выход оперирует процессор 8 или 16 разрядными данными. Процессор определяет, нужно ли подключать старший банк)
RD — чтение из памяти или внешнего устройства (0 - процессор читает)
WR — запись в порт внешнего устройства или в память (0 - признак операции записи)
DT/R — 1 - передача данных, 0 - прием
INTA — для подтверждения обработки запроса на прерывание (вход, маскируемые разрешены -> подтверждения прерывания. 0 - разрешно)
HLDA — выход подтверждения запроса на прямой доступ к памяти
DEN — по совмещенной шине передаются данные (1 - передаётся адрес, 0 - передаются данные)
ALE — при выставлении адреса на шину формируется стробирующий импульс (по мере готовности, появляется соответствующий сигнал)
M/IO — разделение адресных пространств памяти и внешних устройств (1 - обращается к памяти, 0 - внеш.устройств)

[Схема демультипликации шин адреса и данных]
Схема демультиплексирования шины адреса и данных



По шине адреса и данных они идут не вперемешку, а разделены по времени. В один момент времени передаются данные, в другой адрес (вначале адрес нужен, затем данные). От процессора исходит совмещенная шина адреса и данных и передаются в буферный регистр и шинный формирователь. По сути, буферный регистр - ячейка памяти, способная что-то хранить.
При появлении единичного сигнала на входе ALE регистр запоминает на входе и выставляет на шину адреса. По обстрабирующему импульсу, понимает, что нужно запомнить и передавать это всё на выход. Если ALE - 0, то буферный регистр игнорирует данные, а на выходе запомненный адрес.
Шинный формирователь, который должен обеспечить согласование, нагрузочную способность, может отключаться и управляется DEN, Если 1, то шинный формирователь отключается, если 0, то обеспечивает передачу данных в обоих направлениях.

[Пример: процессору нужно считать ячейку памяти. Процессор выставляет на совмещённой шине адрес ячейки, стробирует его сигналов на выходе ALE (в DEN - 1, значит, он отключён там. ALE - 0, ничего запоминать не надо). Передаётся в оба устройства, но шинный формирователь игнорит полученное (пока у него нет нолика на OE), а буферный регистр ожидает абстрагирующий импульс. Он запоминает адрес ячейки памяти, и он присутствует там. Мы физически обратились к нужной ячейки памяти. Передаётся 0 в DEN, процессор отдаёт данные шинный формирователь (адрес уже подключен). Адрес запоминается буферным регистром. Процессор по шине совмещённой получает эти данные из ячейки памяти.]

Организация адресного пространства ввода/вывода и памяти

1. Раздельная организацие. Если на выходе M/IO 1, то обращаемся к памяти и работаем в адресном пространстве памяти, если 0, то работаем в адресном пространстве ввода и вывода. +: все адреса для ячеек памяти и поротов ввода-вывод доступны. Существуют одинаковые адреса (типа, память 00 и порт 00 - проблем нет). -: для памяти много способов адресации этим процессором и множество разных команд, с портами всё ограниченно.
2. Совместная организация. На выходе M/IO всегда присутствует 1, проц всегда считает, что обращается к памяти, на самом деле определенный диапазон адресов выделен под внешние устройства, соответственно часть адресного пространства памяти используется для использования для устройств ввода/вывода. +: все команды доступны и все способы адресации. -: пропадает часть адресного пространства памяти, часть ячеек будет поставить, невозможно адресовать (то же самое с портами ввода-вывода)
3. Комбинированная организация. Когда к тем портам ввода вывода, к которым необходимо применять сложные команды характерные для памяти, используется совмещение адресных пространств, теряя немного адресного пространства памяти, но зато все команды можно использовать. А для которых не требуется большого количества команд или сложной способов адресации используется разделение (по M/IO).

   В современных вычислительных системах распространен третий вариант. [В конце 3 лекции тоже самое]

Лекция 10

[Организация памяти x86]

Модель памяти

[Организация памяти i8086]

Согласно принципам Джона фон Неймана, электронная вычислительная машина (ЭВМ) выполняет вычисления в соответствии с программой, которая располагается в памяти ЭВМ.

Любая программа включает в себя команды (операторы) и данные (операнды). Программа выполняется с целью получения результирующих данных на основе преобразования исходных, с возможным формированием промежуточных данных. В соответствии с концепцией хранимой в памяти программы, и команды и данные располагаются в единой памяти и представлены в двоичных кодах.

Память представляет собой набор ячеек, каждая из которых имеет свой уникальный номер – адрес. Команды и данные хранятся в ячейках, и их местоположение в памяти определяется адресами соответствующих ячеек. Поскольку команды и данные на уровне кодов неотличимы друг от друга, то для различия команд и данных используется их размещение в различных областях памяти – сегментах.
В курсовой работе нет сегментации памяти. У нас юзается специальный бит, который отличает команду и данные.

Сегментация памяти

Сегмент - это прямоугольная область памяти, характеризующаяся начальным адресом и длиной. Начальный адрес (адрес начала сегмента) – это номер (адрес) ячейки памяти, с которой начинается сегмент.

Длина сегмента – это количество входящих в него ячеек памяти. Сегменты могут иметь различную длину. Все ячейки, расположенные внутри сегмента, перенумеровываются, начиная с нуля (имеют свою адресацию).

Адресация ячеек внутри сегмента ведется относительно начала сегмента; адрес ячейки в сегменте называется смещением или эффективным адресом - EA (относительно начального адреса сегмента).

Переполнения при формировании адреса

В процессоре 8086 сегментация организована так, что перенос из старшего бита, который может возникнуть при суммировании, игнорируется. Это приводит к

так называемой кольцевой организации памяти, при которой за ячейкой с максимальным адресом FFFFF следует ячейка с нулевым адресом.

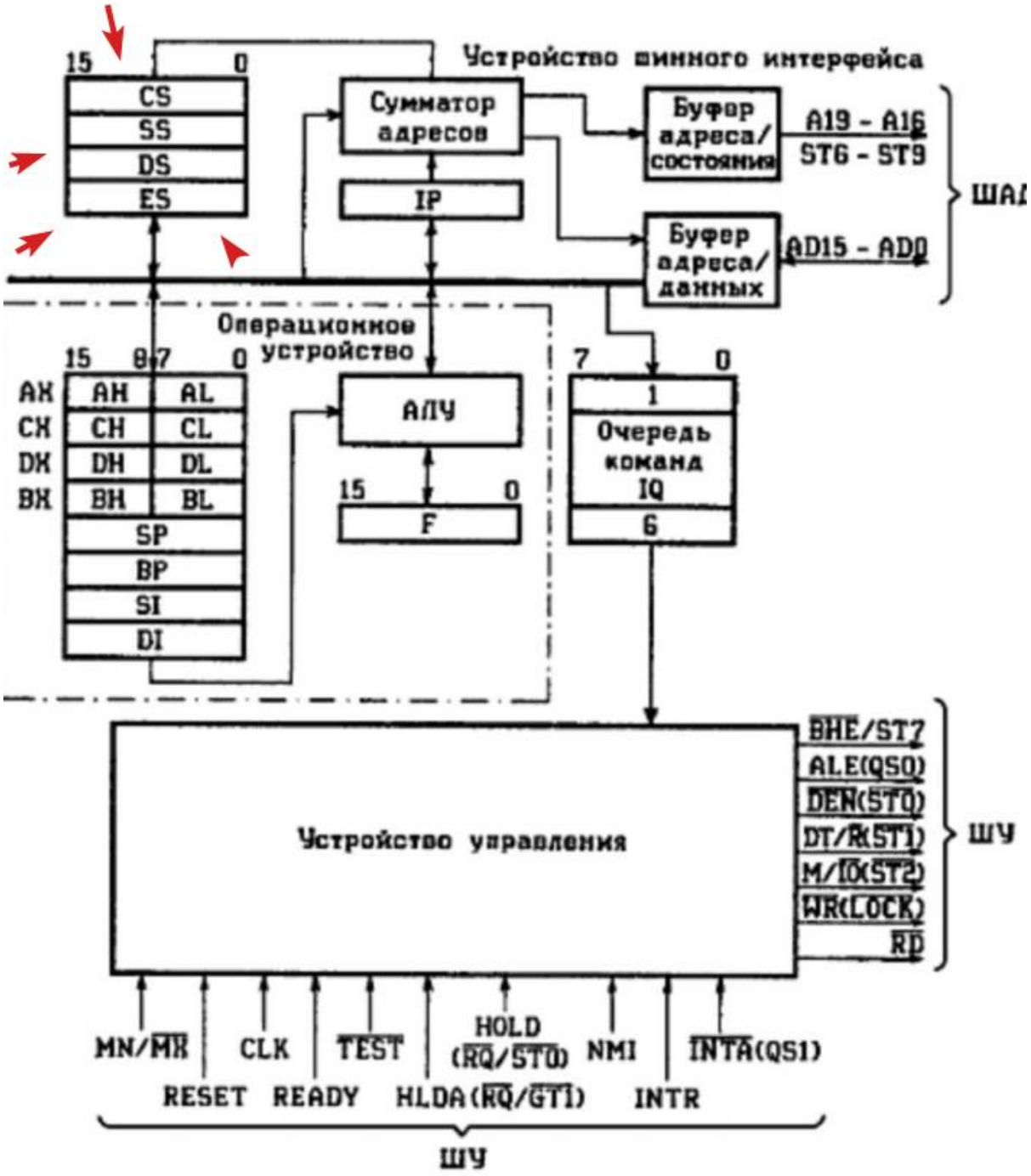
Аналогичную кольцевую организацию имеет и каждый сегмент.
Выйти за пределы памяти невозможно.

[Сегментация памяти i8086]

Сегментные регистры

Сегментный регистр 8086 - рассматривали ранее где-то в 9 лк.
16-битная x86-архитектура, благодаря наличию четырёх сегментных регистров, позволяет одновременно иметь доступ к четырём сегментам памяти. Назначение сегментных регистров:
DS (data segment) — сегмент данных;
CS (code segment) — сегмент кода;
SS (stack segment) — сегмент стека;
ES (extra segment) — дополнительный сегмент.
Каждый из этих регистров 16-ти разрядный. И каждый из них содержит начальный адрес соответствующего сегмента. В процессе работы программы содержимое сегментого регистров можно менять и соответственно переходить к другому сегменту, это не означает, что весь код должен быть в одном сегменте кода.
Логический адрес на такой платформе принято записывать в виде сегмент:смещение, где сегмент и смещение задаются в шестнадцатеричной системе счисления.
В реальном режиме для вычисления физического адреса байта памяти происходит сдвиг влево на 4 разряда значения соответствующего сегментного регистра, а затем добавляется смещение.
Адресных линий 20 в 8086. А регистры 16-иразрядные. Чтобы получить 20-иразрядное значение, мы сдвигаем на 4 разряда влево, а затем добавляем смещение.
Например, логический адрес 7522:F139 дает 20-битный физический адрес:
75220 + F139 = 84359

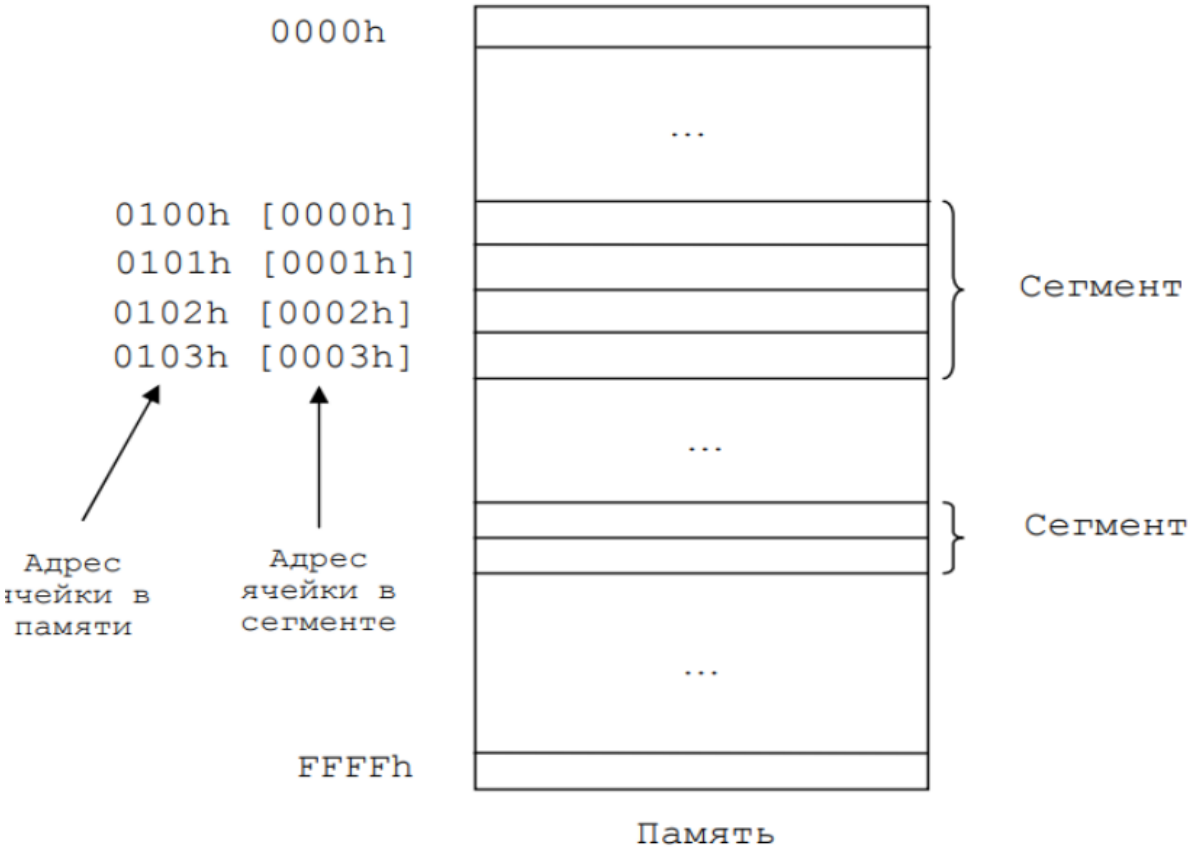
Внутренняя структура



Сегменты в левой верхней части, содержимое этой верхней части подаются в сумматор адресов, куда также подаются данные с внутренней шины и содержимое регистра IP, который является счетчиком команд. Сумматор адресов - это то устройство, которое осуществляет сложение со сдвигом.

Именно это устройство производит сложение со сдвигом и формирует из 16-ти разрядных регистров 20-ти разрядные адреса. Передает его в буфер.

Пример организации памяти



Память представляет из себя некий массив, который делится на сегменты. Сегментов может быть множество. Внутри сегментов существует своя адресация, которая определяется смещением и идет с нулевой точки (адреса сегмента).

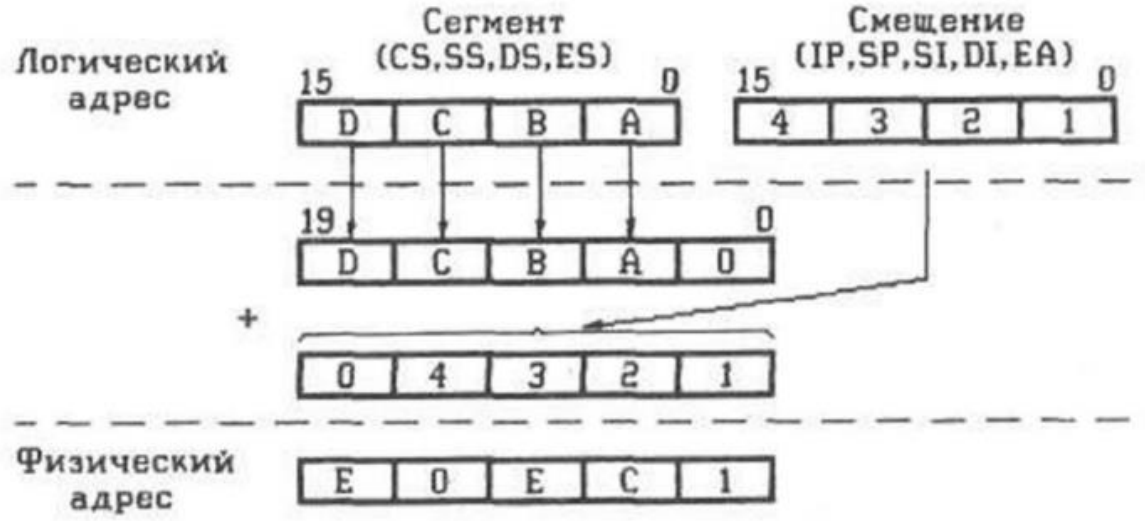
Для команд, для данных и для стека используется различные комбинации сегментных регистров и различные операции смещения.

Источник логического адреса

Тип обращения к памяти	*Сегмент (по умолчанию)*	*Вариант*	*Смещение*
Выборка команды	CS	Нет	IP
Стековая операция	SS	Нет	SP
Переменная	DS	CS, SS, ES	EA
Цепочка-источник	DS	CS, SS, ES	SI
Цепочка-приемник	ES	Нет	DI
BP как базовый регистр	ES	CS, SS, DS	EA

EA – эффективный адрес, вычисляемый в соответствии с заданным способом адресации
При декодирования команды определяется смещение. Меняя комбинации регистров, мы можем передать управление в любую точку команды. Все стековые операции выполняются с регистром стека и указателя стека регистра SP.
С данными можно использовать любой сегментный регистр.

Пример формирования адреса



На схеме сегментный регистр 16-разрядный.
На первом шаге сдвигаем содержимое сегментного регистра влево на 4 разряда, справа появляется 4 нуля.
После прибавляем смещение и получаем физический адрес. Это значение и будет передано по шине адреса.

Особенности сегментации памяти в микропроцессоре i8086

- Сегменты памяти определяются только сегментными регистрами. (какое значение запишем в сегментный регистр, там и будет начинаться сегмент памяти)
- Начальный адрес сегмента кратен 16. (Что бы мы ни написали в сегментный регистр, всё равно начальный регистр будет кратен 16, потому что это содержимое регистра сдвигается на 4 и справа всегда появляется 4 нуля)
- Никаких средств проверки правильности использования сегментов нет. (мы сами определяем сегментацию, но только цикловая структуры памяти и цикловая структура сегментов не даёт выходить за пределы памяти)
- Размещение сегментов в памяти достаточно произвольно. (сами определяем или компилятор)
- Сегменты могут частично или полностью перекрываться, или не иметь общих частей. (одинаковое содержимое записать в сегментный регистр стека и кода,

тогда сегмент стека совпадёт с сегментов кода - ничего хорошего, но это возможно, размер сегмента стека 64 кб)
- Программа может обращаться к любому сегменту как для считывания, так и для записи данных и команд. (можно менять на ходу содержимое программы)
- Для защиты памяти от несанкционированного доступа других программ требуются специальные "внешние" схемы или программы (ну или самому написать в коде).
- Система не делает различий между сегментами данных, кода и стека. (на выходе формируется 20-иразрядный физический адрес, не важно куда обращается)
- Нет никаких препятствий для обращения к физически не существующей памяти.
- При обращении к несуществующей памяти результат непредсказуем (все зависит от разработчика аппаратного обеспечения компьютера.)

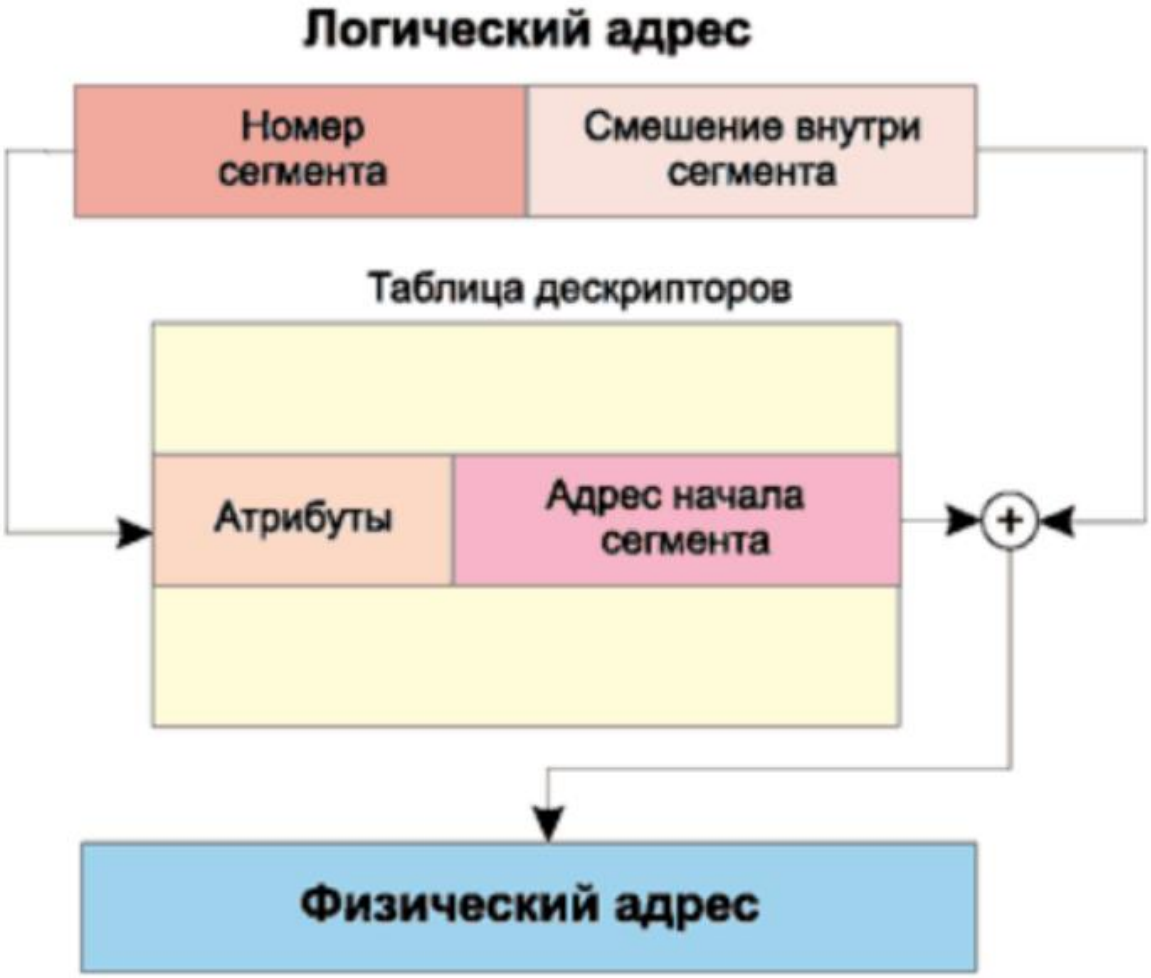
Защищенный режим

Появился в процессоре 80286, сохранив понятия сегмента и смещения, и радикально модернизировал механизм сегментации, предложив концепцию логического (виртуального) адреса.

В защищенном режиме работы процессора содержимое сегментных регистров используется не как слагаемое при вычислении адреса, а как индекс или селектор, выбирающий дескриптор сегмента в специальной структуре, называемой дескрипторной таблицей и описывающей свойства каждого сегмента: базовый адрес, размер и атрибуты, связанные с управлением привилегиями и защитой памяти.

По сути, сегментные регистры сохранились, но только при инициализации компьютера, заполняется в памяти таблица, по которой вычисляются физические адреса. Содержание сегментного регистра не является слагаемым при формировании.

Преобразованию логического адреса в физический



Номер сегмента используется не напрямую, при формировании физического адреса, а используется для поиска соответствующего дескриптора, который содержит определённые атрибуты, влияющие на защиту, расположение в памяти. Оттуда уже извлекается оттуда соответствующая запись, которая суммируется, но для этого куча проверок проводится.
Используя дескрипторов, можно адресовать больше физической памяти, так как в таблице дескрипторов мы определяем расположение сегментов, у нас уже иная система адресация, которая определяется логически в соответствующей таблице памяти.
Грубо говоря у нас есть таблица, где номеру сегмента соответствует определенный физический адрес.
При запуске там опрашивается что где, и соответственно ответсвтенность с программиста снимается, потому что нельзя обратиться к несуществующей памяти

Таблица дескрипторов

Любой процессор x86 (80286) начинает свою работу в реальном режиме. Затем его переключает специальная программа в защищенный режим.

Программа, переключающая процессор в защищенный режим, должна подготовить в памяти управляющие структуры, используемые процессором в этом режиме. Одна из таких структур — таблица дескрипторов сегментов, описывающая свойства адресуемых областей памяти.

При старте платформы и выполнении процедуры POST, обязанности по управлению контекстом процессора возлагаются на BIOS или UEFI, что подразумевает использование таблиц страниц и сегментов, построенных firmware. То есть в современном компьютере, все эти таблицы, с которыми работают прикладные программы готовит BIOS или UEFI. Готовятся таблицы необходимые, опрашивается всё железо, вносится в таблицу, чтобы мы не имели прямого доступа.

При выполнении Legacy-boot (режима совместимости), firmware (программное обеспечение БИОСа или УЕФИ) передает управление загрузчику в 16-битном режиме Real Mode (реального времени). Переключение процессора в Protected Mode и управление его контекстом входит в обязанности ОС. Либо не выполняется, в случае 16-битной системы. Вместе с тем, до момента передачи управления от BIOS к загрузчику ОС, в частности во время инициализации и тестировании памяти, процессор может временно переводиться в Protected Mode процедурами POST.
По сути, после подготовки таблицы, в реальном режиме передаётся управление ОС, а ОС при помощи инструкций переводит процессор в защищенный режим, а затем работает с таблицами. Может не перевести и работать в 16-битном режиме. (БИОС сам может включить защищённый режим и выполнить некоторые операции, но всё равно передаётся управление ОС в реальном времени. Далее ОС сама.

При UEFI-boot, в момент передачи управления от firmware к загрузчику операционной системы, процессор уже работает в Protected Mode, тем не менее, ОС повторно инициализирует CPU (процессор), предварительно создав собственные таблицы дескрипторов и сегментов, прекращая полномочия контекста UEFI, вызовом функции ExitBootServices().

Виртуальная память

В 32-битном процессоре 80386, сегментация была дополнена механизмом трансляции страниц или Paging, лежащим в основе организации виртуальной памяти.

Виртуальная память - схема адресации памяти компьютера, при которой память представляется программному обеспечению непрерывной и однородной, в то время как в реальности для фактического хранения данных используются отдельные (разрывные) области различных видов памяти, включая кратковременную (оперативную) и долговременную (жёсткие диски, твердотельные накопители).
Виртуальная память позволяет использовать ЖД в качестве памяти, оперативной, к примеру. Мы обращаемся к линейной структуре (виртуальной), которая может быть представлена в виде системы различных устройств.
Уровень абстракции возрастает.

Лекция 11:

[Организация прерываний x86]

[Организация прерываний на примере i8086]

Существуют различные методы поиска источника запроса на прерывание. Как мы помним у процессора 8086 имеется два входа запросов на прерывание: NMI и INTR. Вот когда приходит запрос на прерывание, то процессору каким-то образом требуется узнать какое конкретное устройство отправило данный запрос. Существует два распространённых способа:

1) (Примитивно) Это устройство перед тем как подать запрос на прерывание пишет определённый бит в определённый регистр и процессор, когда ему требуется определить источник запроса на прерывание, он обращается к этому регистру и считывает содержимое регистра и в результате узнает какое устройство отправило данный запрос

2) (Наиболее распространённый) векторное прерывание. Когда используется контроллер прерываний, который передаёт номер прерывания процессору и процессор таким образом уже знает, какое устройство конкретно сформировало данный запрос. Такой способ - векторное прерывание. Вектором (в системе прерываний) называется адрес обработчика прерывания, который хранится в специальной таблице векторов прерываний. Когда контроллер передаёт номер прерывания процессору, процессор по этому номеру извлекает адрес обработчика этого прерывания.

Адрес в процессоре 8086 20-разрядный и формируется с помощью содержимого двух регистров, а именно с помощью сегментов кода CS и с помощью регистра счётчика команд IP. Так как эти два регистра 16-разрядные, то есть на хранение содержимого каждого регистров требуется нам два байта, то у нас каждый вектор занимает 4 байта. И вот эти все вектора, то есть все адреса обработчиков прерываний хранятся в начальном сегменте памяти и занимает 1024 байта, всё это называется таблицей векторов прерываний.

Таблица векторов прерываний

В реальном режиме работы в системе прерываний используется понятие вектора прерывания.

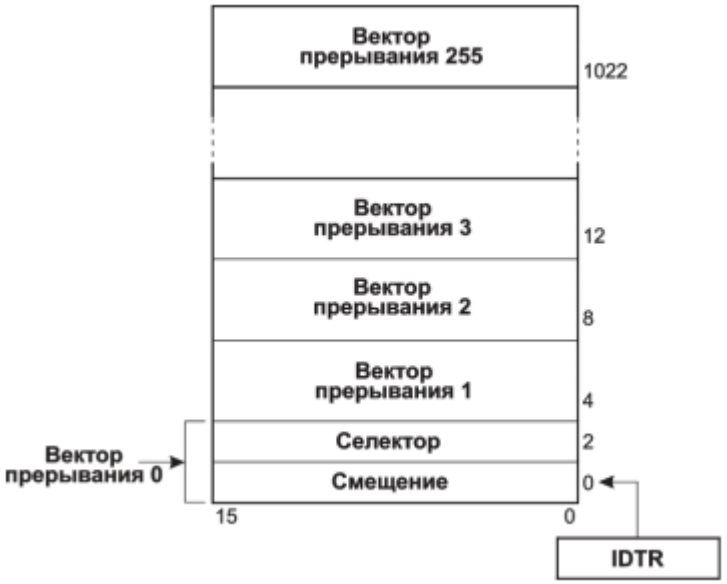
Каждый вектор прерывания состоит из четырех байтов, или двух слов: первые два содержат новое значение для регистра IP, а следующие два — новое значение для регистра CS. Таблица векторов прерывания занимает 1024 байт. Таким образом, в ней может быть задано 256 векторов прерываний. В процессоре i8086 эта таблица располагается на адресах 00000H-003FFH.

Вся эта таблица располагается в начальном участке памяти.

Таблица векторов прерываний

Нулевой вектор прерывания состоит из селектора и смещения.

В качестве селектора выступает содержимое сегментного регистра CS
В качестве смещения содержимое регистра счётчика IP
Каждое содержимое регистра требует 2 байта, соответственно каждый вектор занимает 4 байта памяти.
Если знаем номер вектора и знаем, что вектора последовательно расположены и каждый занимает 4 байта. Значит, можно номер умножить на 4, чтобы найти физический адрес прерывания.



Виды прерываний
Прерывания бывают двух видов: внутренние и внешние.
Внутренние прерывания возникают в результате работы процессора в ситуациях, которые нуждаются в специальном обслуживании, или при выполнении специальных команд (INT, INTO). (иначе говоря, ситуация внутри процессора)
Это следующие прерывания:
- прерывание при делении на ноль (номер прерывания 0);
- прерывание по флагу TF (номер прерывания 1); - трассировка (для отладки)
- прерывания, возникающие при выполнении команд INT называются программными.
В качестве операнда команды INT указывается номер прерывания, которое нужно выполнить.
Внешние прерывания возникают по сигналу какого-нибудь внешнего устройства. Существует два специальных внешних сигнала среди входных сигналов процессора, при помощи которых можно прервать выполнение текущей программы и тем самым переключить работу центрального процессора. (иначе говоря, ситуации внешними по отношению к процессору устройству)

NMI (No Mask Interrupt — немаскируемое прерывание) - прерывание будет обработано в любом случае

INTR (Interrupt Request — запрос на прерывание) - обработка в случае, если флаг даёт это сделать. Иначе игнорирует.

(то что внутри контура это всё процессор и соответственно внутренние прерывания)

Рис. 1.7. Источники прерываний

Этапы обработки прерываний

1. Контроллер прерываний получает заявку от определенного периферийного устройства и, соблюдая схему приоритетов, генерирует сигнал INTR (запрос на прерывание), который является входным для микропроцессора.
2. Микропроцессор проверяет флаг IF в регистре флагов. Если он установлен в 1, то переходим к шагу 3. В противном случае работа процессора не прерывается.
3. Микропроцессор генерирует сигнал INTA (подтверждение прерывания). В ответ на этот сигнал контроллер прерываний посылает по шине данных номер прерывания.
4. В стек помещается регистр флагов. (регистр флагов может меняться, потому что нужно будет вернуться в программу без изменений) - 2 байта
5. Флаг включения-выключения прерываний IF и флаг трассировки TF, находящиеся в регистре флагов, обнуляются для блокировки других маскируемых прерываний и исключения пошагового режима исполнения команд. (вначале отключается, остальное разработчик обработки прерываний решает)
6. Значения регистров CS и IP сохраняются в стеке. (чтобы вернуться к прерванной точке) - 2 байта и 2 байта

7. Вычисляется адрес вектора прерывания и из вектора, соответствующего номеру прерывания, загружаются новые значения IP и CS.
8. Выполнение подпрограммы обработчика прерывания. (менять содержимое регистров флагов можем, к примеру)
9. Извлечение из стека IP и CS (восстановление точки в программе)
10. Извлечение из стека регистра флагов.
11. Процессор продолжает работу с того места, где он был прерван

Использование прерываний

- для обмена информацией между процессором и внешним устройством (ВУ); (позволяет обмениваться информацией с устройствами разных скоростей без замедления процессора)
- в аварийных ситуациях, например, при понижении напряжения питания; (для сохранения данных, к примеру)
- при исключительных условиях, таких, как переполнение, (деление на ноль);
- для индикации аппаратных сбоев, приводящих к ошибкам при обработке данных; (ошибка контроля чётности, переспросить искажённый пакет)
- при программных сбоях;
- для координации работы в многопроцессорных системах; (пересылка данных, к примеру)
- для профилактики, ремонта, тестирования и отладки системы (TF) (просмотр содержимого регистра и всё такое)