

# ЭВМ

VG6

13 ноября 2025 г.

## 1 До этого мне было лень

### Вопросики (23 октября 2025)

Какой принцип Неймана позволяет привесить скорость света?

Вычислительный процесс следует организовывать параллельным образом.

Почему ЭВМ в двоичной системе счисления, а не в десятичной?

5 принцип Неймана: разработка устройств для других операций ненцелесообразно (кроме сумматора). В теории он прав, а на практике нет. К примеру аппаратные умножители.

**Главный принцип Неймана:** и программа и данные лежат в одном и том же месте в одном и том же виде. Не нужно для заменять машину, можно заменить только программу.

**Схема** - графическое обозначение элементов и связи между ними.

## 2 Структура классической ЭВМ.

Состоит из 3х блоков.



### 2.1 Описание блоков.

1. Арифметико-Логическое устройство

- Выполняет преобразование информации представленной в двоичном виде. Например: числа при выполнении арифметической операции.
- **Операнд** - мы будем называть участника арифметической или логической операции.
- АЛУ преобразует операнды и получает результат и признаки результата (дополнительные сведения о том, какой результат получился Правильный/Неправильный).
- Какой операцией заняться определяют управляющие сигналы.
- **Операция** - арифметическое или логическое действие выполняемое в АЛУ над операндами. То, что может делать АЛУ называется операцией.

## 2. Запоминающее устройство

- Предназначение: хранить двоичные числа.  
Для этого нужны 3 операции:
  - (а) Запись
  - (б) Чтение
  - (с) Выборка (адресация не совсем одно и тоже)
- Запоминающее устройство классической ЭВМ является адресным устройством. Каждое число хранится в ячейке имеющей свой уникальный адрес, который тоже является числом. Впростейшем случае - порядковый номер ячейки памяти.
- Запоминающее устройство представляет собой одномерный массив, где в качестве индекса выступает адрес (номер ячейки).
- 4 действия (тоже + хранение) определяется управляющими сигналами, среди которых сигналы: операции записи, чтения и многое другое.

## 3. Устройство Управления

- Занимается управлением вычислительным процессом. То есть его выходом является порождение управляющих сигналов (сигналов управления), которые потребляются другими ящиками и самим собой.
- Помимо внешних блоков он управляет и самим собой.
- Цели и задачи управления: реализация выполнения команды. Для этого формируемые им сигналы несут 2 сущности: что делать, когда это надо сделать.
- Эту команду откуда-то надо взять. А они лежат в запоминающем устройстве, там же, где и данные.
- На выходе сигналы, на входе ... из памяти
- нужно формировать адрес ячейки откуда взять команду, получить её и выполнить.

Связи между этими блоками изображены на рисунке. По признакам результата УУ влияет на дальнейшее прохождение вычислительного процесса (изменять последовательность выполнения команд).

Например, произошло переполнение, значит нужно сформировать признак переполнения и передать УУ.

УУ получает команды из памяти, однако и операнды и команды лежат в разных ячейках запоминающего устройства (адресам) поэтому УУ должен формировать адреса как команд, так и operand.

### 2.2 Цикл выполнения команды.

Структура организована циклически. И этот бег по кругу является бесконечным повторением.

### 2.3 Память. Запоминающее устройство.

Строится по иерархическому принципу. **Регистр** - запоминающее устройство ёмкостью в 1 число. Есть 2 типа:

1. Специализированные регистры, функции которых предопределены конструкцией ЭВМ и являются неизменными и регистры
2. Общего назначения, функционал которых может предопределяться. Доступны для программистов.

### 2.3.1 Виды памяти

**Сверхоперативные ЗУ** - обычно безадресного типа. К таким устройствам относятся буферные запоминающие устройства, стек.

**Буфер** - запоминающее устройство между ЗУ с разной скоростью работы для сглаживания по времени. Обычно организуется очередь (FIFO).

**Стек** - первым вылетает последний заряженный патрон. (LIFO). История перехода к подпрограммам. Используется в системе прерываний.

**Постоянное запоминающее устройство (ROM)** - адресное запоминающее устройство без функции записи. **EPROM** - оставляет возможность сменить содержимое путём перезаписи, что требует специальное устройство (программатор).

**КЭШ L1, L2, L3, L4** - безадресное ассоциативное запоминающее устройство. Небольшая ёмкость, но кратно ускоряет скорость работы вычислительного устройства. Благодаря ассоциативному доступу (тэгами) аккумулируют в себе те команды или данные, которые используются наиболее интенсивно. Тем самым создавая копию ячеек Оперативной памяти кратно уменьшают доступ к этим данным.

**Основная оперативна память ООП, ОЗУ, RAM, память с произвольной выборкой, память ЭВМ** - (синенько на схеме) память, в которой хранятся те самые команды и данные по Нейману.

**Специализированные блоки памяти** - обмен между вычислительным ядром внешним миром (многопортовая память, ассоциативные ЗУ (используется в КЭШе и т.д. при поиске не по адресу, а по признаку), видеопамять)

**Внешние запоминающие устройства** - то, что подключается к ... через интерфейс. (... , облачные хранилища, Data-центры, ...)

## 2.4 ОЗУ. Оперативное запоминающее устройство.

Совокупность ячеек, пре данный вид памяти для работы в качестве основной оперативной памяти ЭВМ должно обладать свойством произвольной выборки.

**Памятью с произвольной выборкой** мы будем называть адресное ЗУ, время выборки которого не зависит от адреса ячейки и последовательности обращений к ячейкам этого устройства.

Технические характеристики ЗУ:

- 1 бит - 1 двоичный разряд
- 1 байт - 8 бит. Попытка представить символы алфавита при помощи таблички кодирования (ASCII).
- К(Кило) -  $2^{10} = 1024$
- М(Мега) -  $2^{20} = \dots$
- Г(Гига) -  $2^{30} = \dots$

**Память** - информационная ёмкость 1 адресуемой ячейки того, что имеет адрес.

**Организация ЗУ** - произведение числа ячеек на их разрядность, например: 4Гх8

Характеристики запоминающего устройства - временные.

**Быстродействие (производительность) ЗУ** - оценивают временами считывания и записи, длительностью циклов считывания/записи

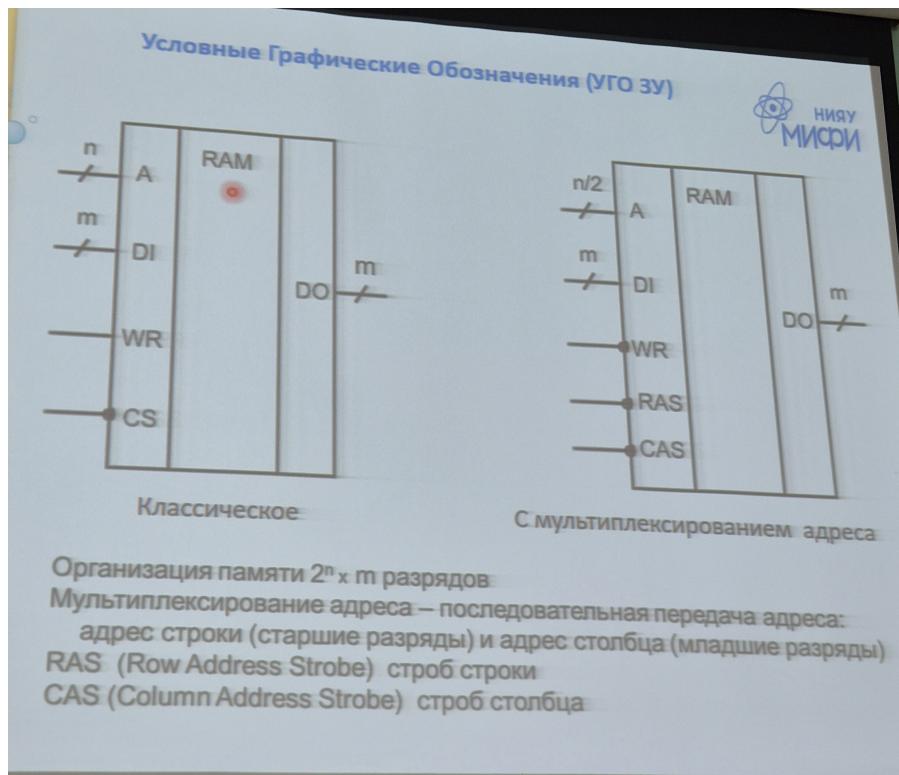
**Время считывания** — интервал времени между моментом сигнала на считывание (адрес или разрешение считывания) и моментом выдачи данных на выходы памяти.

**Время записи** - интервал времени после задания сигнала записи, достаточный для установления ячейки в состояние, задаваемое входными данными.

**Цикл считывания/записи** - минимальный интервал между последовательными обращениями.

## 2.5 Примеры условных графических обозначений (УГО ЗУ)

Запоминающее устройство:



**n** - разрядность шины адреса (количество проводов).

**A** - адрес.

**DI** - Data Input. **m** - размер слова этого запоминающего устройства.

**WR** - Write Read. Какую операцию выполнять

**CS** - Chip Select. Вход разрешения работы этого модуля. Если на схеме выколотая точка, то это означает, что разрешающий сигнал "0".

Организация памяти:  $2^n * m$

### 3 Семинар 2.

#### 3.1 Вопросики (30.10.2025)

Из скольки частей состоит классическая ЭВМ? - из 3х

Какое волшебное слово объясняет, как эти 3 кубика работают? - циклически

Предназначение АЛУ? - выполнять АЛ операции. Что порождает АЛУ? - признак результата и результат. что делать - статический сигнал

когда делать - импульс

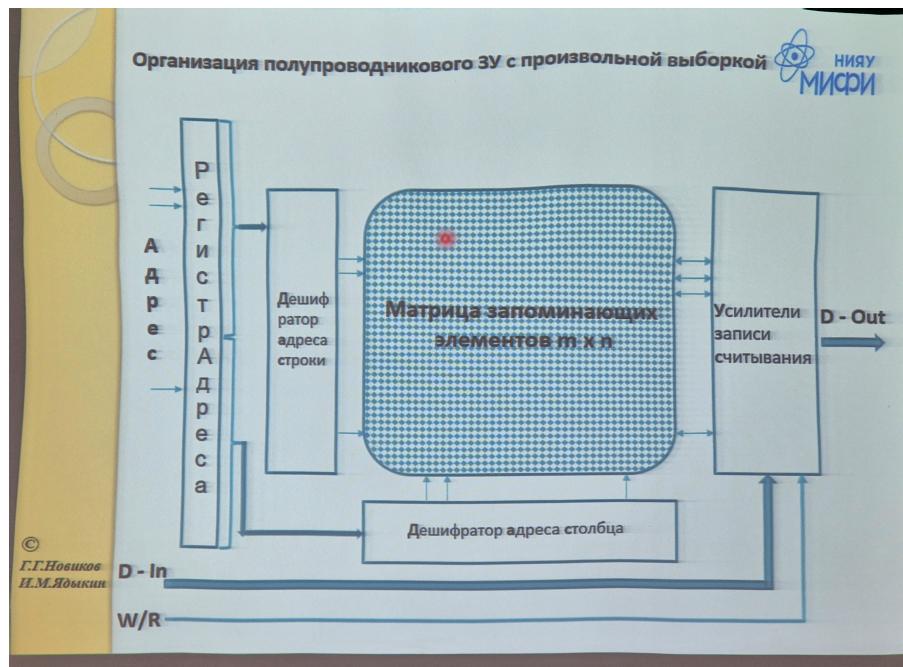
Адресуема только вся ячейка, так что любая опреация происходит сразу со всей ячейкой.

#### 3.2 Продолжаем УГО ЗУ

DI и DO абсолютно одинаовые

Если сигнал на CS не совпадает с разрешающим, то устройство находится в состоянии хранения.

### 3.3 Организация полупроводникового ЗУ с произвольной выборкой



Плоская двумерная матрица из запоминающих элементов. Бистабильные элементы.

Организованно так, чтобы можно было обратиться к любому элементу.

**Дешифратор** - электронное устройство, преобразующее позиционный двоичный код в код унитарный.

**Унитарный** - двоичный код, содержащий 1 единственную единицу. Значение определяется номером разряда, в котором находится единица.

Таким образом мы получаем указатель этой единицей на элемент строки или столба.

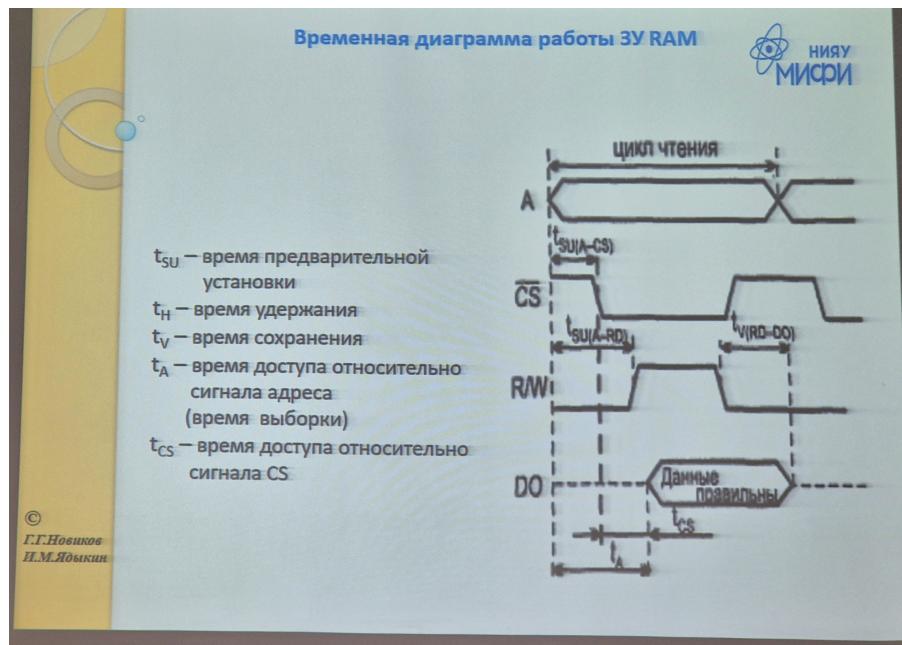
Регистр хранит 1 число - адрес, которое разделяется на 2 числа пополам, которые считаются номером адреса строки и номером адреса столбца.

#### 3.3.1 Усилители записи считывания

Правильнее сказать: формирователи.

В зависимости от того, из чего сделаны элементы хранения могут быть разные потребности в вольтах и амперах. И суть этих устройств в согласовании внутренних сигналов и внешних стандартных логических сигналов.

### 3.4 Временная диаграмма работы ЗУ RAM.



Временная диаграмма - изображение на осях времени электрических сигналов несущих, значения логических переменных.

#### 3.4.1 Запоминающее устройство на феритовом сердечнике



Это сплав железа и чем-то ешё так, чтобы получилось хорошо (речь о магнитных свойствах) и испекли пирог в виде кольца. Это кольцо обладает способностью быть намагниченным в двух направлениях. Вокруг тока образуется по правилу буравчика направление намагничивания, а после того

Функции:

1. запись ... (всё те же 4)

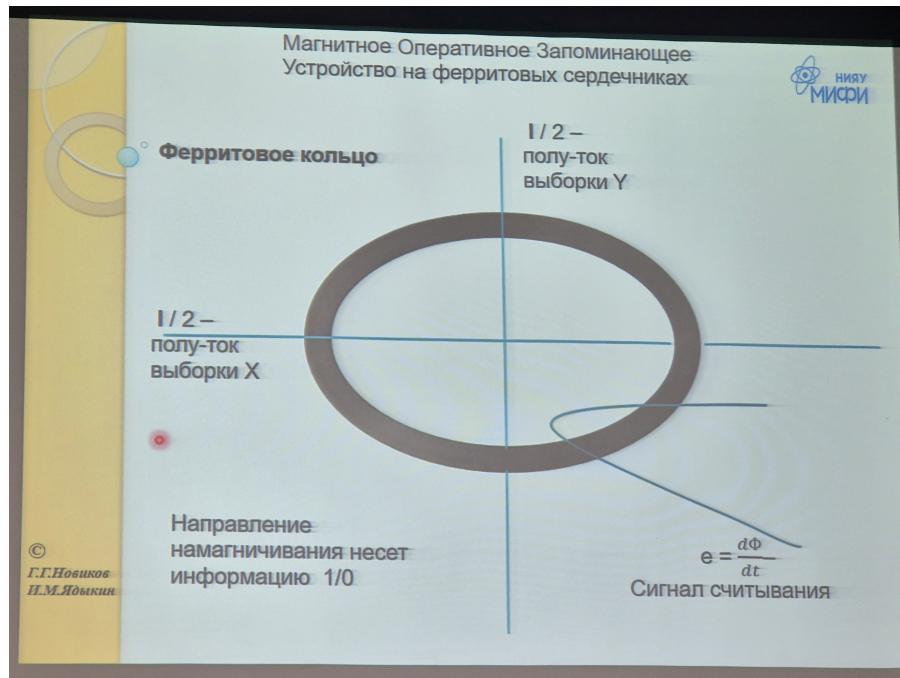
Запись:

Мы пустим половину тока по 1 проволке, а вторую половину по второй. (только полный ток перемагнитит). Там где токи пересекаются ячейка перемагничивается, а другие кольца даже не почешутся.

Чтение:

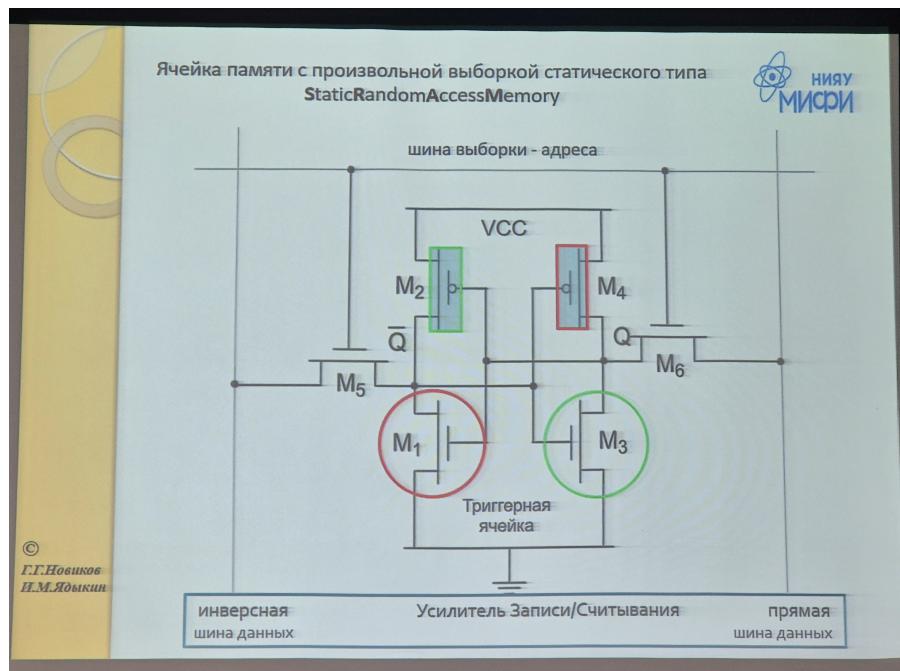
Пустим 3-ю проволочку (справа кривулька), не пуская по ней ток.

1. Без изменения намагничивания вольтметр ничего не покажет.
2. При изменении намагничивания возникнет импульс, вольтметр зафиксирует это.



Свойства: всё равно на радиацию. Разрушить можно только кувалдой.

### 3.5 Ячейка памяти с произвольной выборкой статического типа



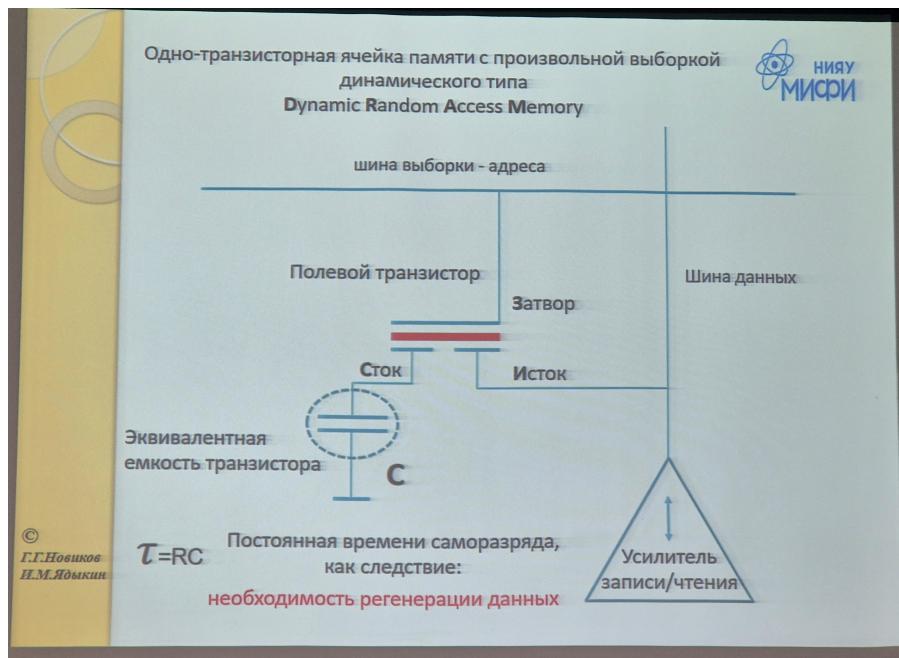
Пара ... выход одного соединён со входом другого. Исключает возможность прямого тока от шины питания к земле. Реализован на полевых транзисторах. Хранит только до тех пор, пока есть питание. Функции выборки: M<sub>5</sub>, M<sub>6</sub>. Они подключены к плечам ячейки и будучи открыты шиной выборки соединяют выбранную ячейку с усилителем, подавая туда 2 сигнала противоположной полярности.

Зелёное и красное на слайдах меняется, поэтому бистабильно :)))

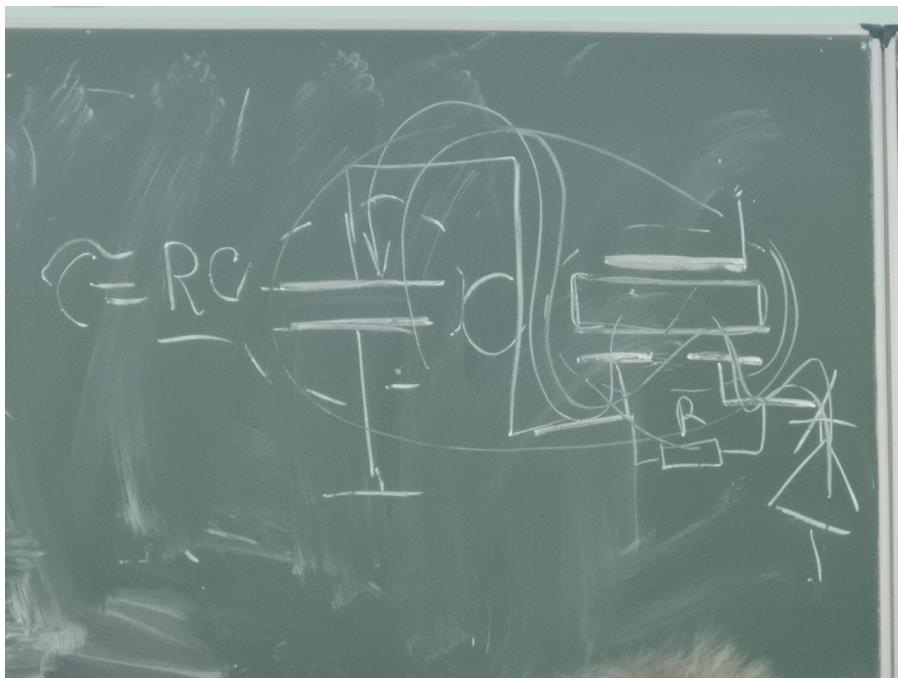
Здесь плохо, что ячейка хранит очень мало. 6 транзисторов на 1 бит.

Альтернативный вариант динамическое запоминающее устройство на полевом транзисторе.

### 3.6 Одно-транзисторная ячейка памяти с произвольной выборкой динамического типа.



Можно использовать конденсатор в качестве элемента хранения.



Эквивалентная схемма с открытым транзистором и подключённый к усилителю, а затем закроем транзистор. Если опять открыть, то заряд потечёт в усилитель.

Но так как это полуизолятор мы можем повесить дополнительное сопротивление и со временем заряд стечёт и информация будет потеряна. Чтобы этого не произошло нужно считать её, записать и делть это постоянно.

То есть хранение информации в такой ячейке происходит динамически и требует постоянного чтения и записи (регенерации информации). Почти 90% всем ЗУ устроены так.

Разреженная регенерация - регенерация не всего массива, а только тех, кто нужен. Любое обращение к ячейке вызывает регенерацию.

## 4 Лекция 10 от 06.11.2025.

Вопросики:

Чем отличается память от склероза? В том, что на экзамене от всё забудет :(

Какие 4 операции есть в ЗУ?

В каком виде ЗУ в процессе хранения необходимо слово выборка? - Регенирация памяти. Динамическая память.

Что такое регенирация? - чтение и обратная запись того, что прочли.

А каком виде ЗУ так же требуется записывать после простения? - В ЗУ на основе фееритового сердечника из-за разрушающего свойства.

В какой метрике среда хранения полупроводникового ЗУ с произвольной выборкой? - двумерный массив.

Что такое хорошо, а что такое плохо, когда речь о ячейке статического типа? Весь тот гемор с обращением динамически к памяти не нужен.

Чем больше ёмкость тем дольше он будет хранить в цикле регенерации.

### 4.1 АЛУ



Выполняет арифметические и логические операции. Для арифметических операций необходимо (по Нейману) иметь сумматор, но могут быть и дополнительные ускорители процессов вычисления.

Минимальное число сумматоров, которые необходимы для работы ЭВМ - 1.

Для логических же операций необходимы:

- Логические элементы (минимально нужно базис)
- Дополнительное оборудование, которое могло бы выбрать какую из операций необходимо выполнить.

Операции АЛ выполняются над **операндами** - число участвующее в А или Л опреации.

Схемма сумматора не обладают свойством сохранять свой состояния в отличие от ЗУ.

Комбинационная схема - схема значение которой есть только тогда, когда на вход поданы элементы.

Необходимо добавить устройство для хранения операндом размером 1 (регистр). На схеме это регистры операнда А и В.

Выходы регистров подключены к входам блока операций, а входы к тому месту где живут операнды (в памяти) соединяются магистралью с ЗУ.

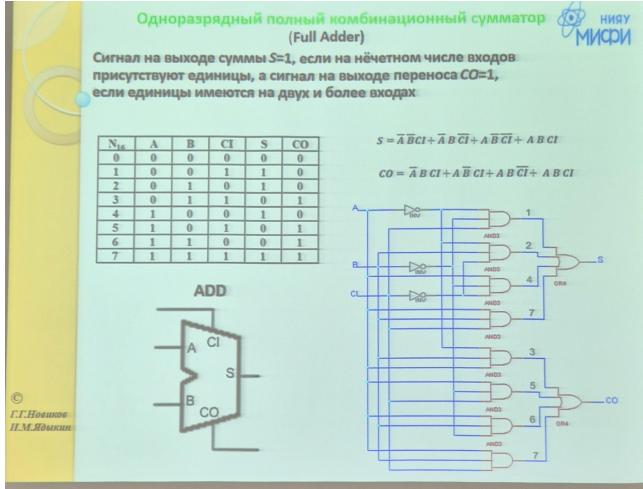
Результаты хранятся в памяти (записывается из регистра результата).

Дополнительная обратная связь между регистром результатата и ... позволяет использовать результат в следующем вычислении. Появляется регистр аккумулятор (накапливающий).

Нужно указать АЛУ какую операцию выполнить. Это указание несёт в себе **КОП** - код операции - число определяющее номер операции в списке возможных операций.

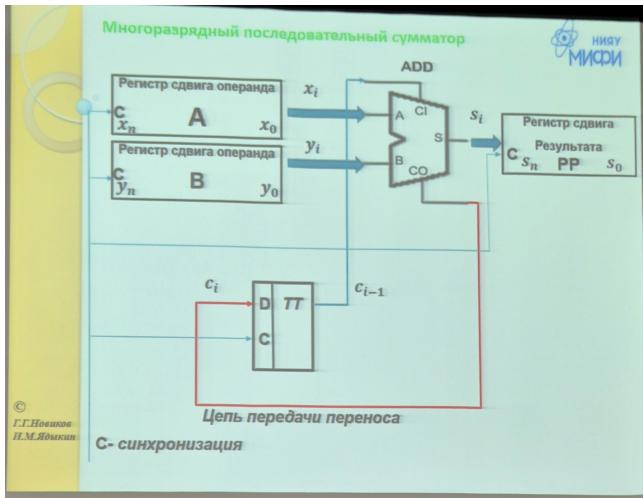
Признаки результатов не нарисованы и мстное устройство управления ( в это не лезем ).

## 4.2 Одноразрядный полный комбинационный сумматор



Нужно сделать из него сумматор многоразрядный последовательный сумматор.

## 4.3 Многоразрядный последовательный сумматор



Вокруг обычного сумматора располагаются 3 регистра.

PP - регистр результаты.

Однако идея данного устройства в том, что это регистры обладающие функцией сдвига кода.

Картинка с нарушениями нужна только для демонстрации.

Сдвиги происходят одновременно во всех 3 регистрах из-за управляющего сигнала.

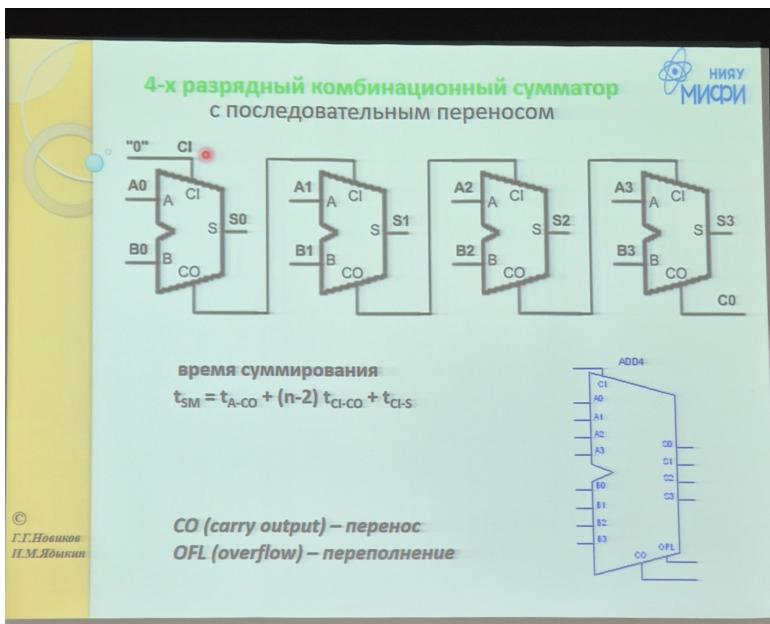
Триггер типа D передаёт в следующий такт то, что было на его входе в текущем (D - delay).

Все имеют вход с буквой С - вход для сингаринизации. Подаётся сигнал синхронизации на все элементы которые должны выполнить действия одновременно.

Хорошо: объём оборудования весьма скромный ( 1 комбинационный сумматор и несколько регистров с функцией сдвига + триггер).

Плохо: Время выполнения суммирования прямо зависит от количества операндов (8 тактов для байта).

#### 4.4 4-х разрядный комбинационный сумматор.



Это уже кусочек схемы. Входы слева, выходы справа. Логическое распространение сигналов слева направо, сверху вниз. Для каждой пары разрядов свой собственный отдельный одноразрядный комбинационный сумматор. Так что вычисление суммы происходит одновременно за 1 такт.

Плохо: с переносом проблема.

## 5 Устройство управления

Что делает? - Пораждает сигналы управляющие.

Из чего пораждает? - Исходя из программы.

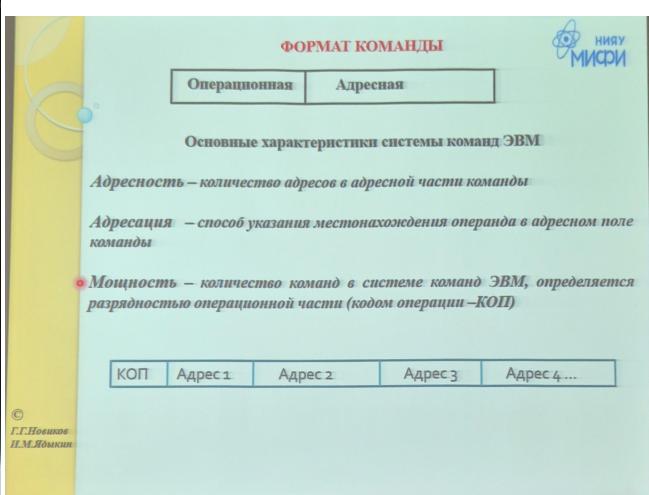
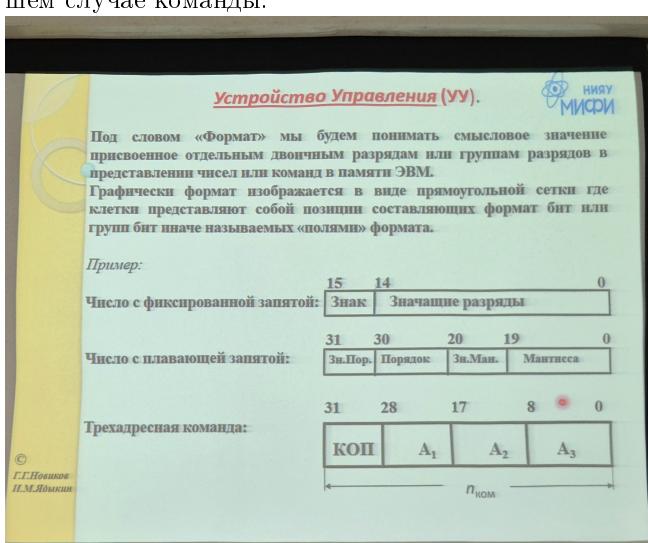
Программа - алгоритм решения задачи, записанный с помощью команд ЭВМ.

Алгоритм - последовательность действий необходимые для решения задачи.

Устройство Управления формирует управляющие сигналы исходя из того, какую команду оно выполняет. Суть - выполнение команды программы.

Команда программы - число.

Что такое формат? - смысловое значение присвоенное отдельным разрядам в двоичной записи. В нашем случае команды.



1. Операционная часть

2. Адресная часть

Всякая команда состоящая из этих частей имеет некоторый набор характеристик. Операционная часть команлы содержит в себе:

- КОП (Код Операции - двоичное число идентифицирующее команду в системе команд ЭВМ)

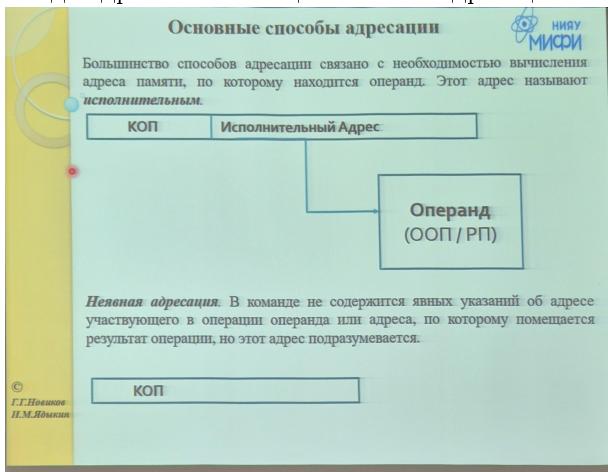
Какая связь между кодом операции и количество разрядов кода опреации:  $2^n$  - Мощность . Адресная часть:

1. Адреса operandов, участвующих в этой команде.

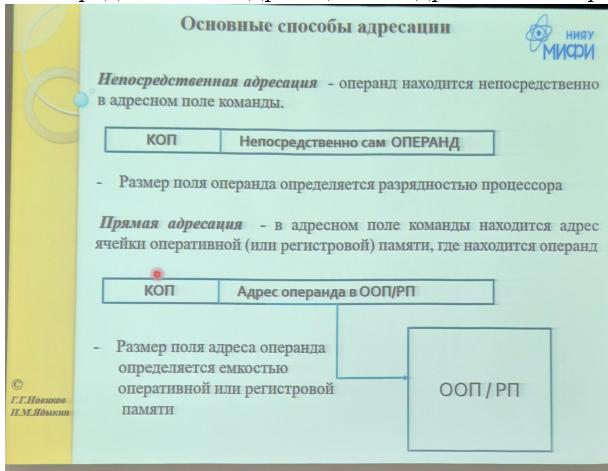
**Адресация** - способ указания местоположения этого операнда в адресном поле команды. Способов этих можно придумать бесконечно.

#### Основные способы адресации:

1. Когда адреса нет вообще. Неявная адресация. Нет указания куда помещать результат.

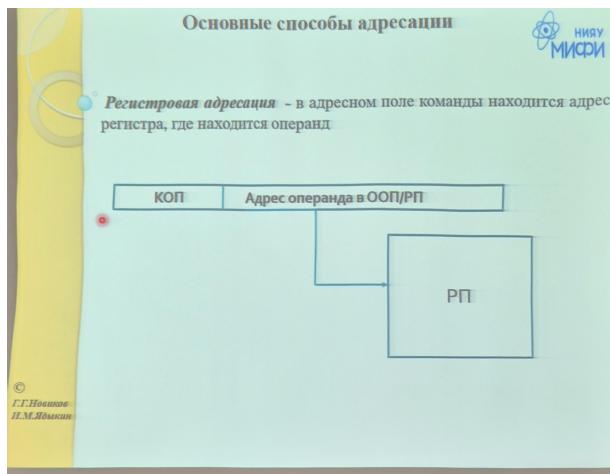


2. Непосредственная адресация. В адресном поле располагается сам операнд.



Чем команда меньше, тем быстрее она будет извлекаться.

3. Прямая адресация.



#### 4. Регистровая адресация

#### 5. Косвенная адресация



Адрес адреса операнда. Может быть построена на нескольких уровнях вложенности.

## 6 Лекция 11. От 13.11.2025

### 6.1 Вопросики

Что такое формат команд? - 2 части: операционная и адресная. В операционной - код операции (двоичное число, которое однозначно идентифицирует команду в системе команд)

С какой характеристикой ЭВМ связана величина поля ввода операции? - Мощности системы команд эвм, что равно количеству команд в штуках.

Какая характеристика позволяет оценить количество полей в адресной части команды? - количество адресов (адресность)

Как называется то, что определяет способ записи местонахождения операнда в этом адресном поле - адресация или способ адресации.

Где следует искать операнд если использована непосредственная адресация - непосредственно в этом адресном поле.

А если адресация прямая - искать в оперативной памяти по адресной ячейки, который написан в адресном поле команды.

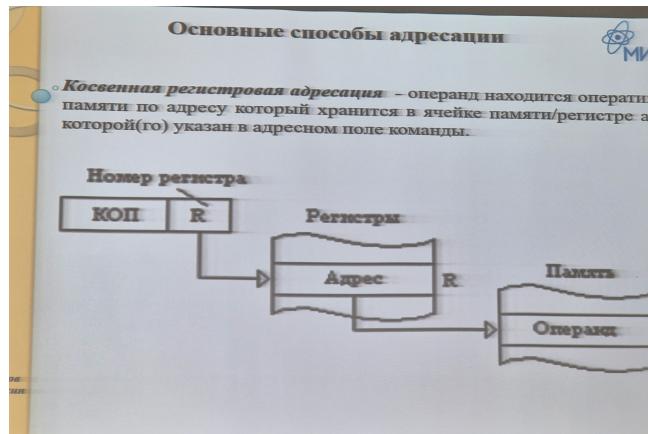
А если косвенное - искать адрес операнда в ячейке адрес которой указан в адресном поле команды. Адрес адреса операнда.

Поругайте мне косвенную адресацию - многократное обращение в памяти ( время докапывания до операнда (минимум 2 раза)).

О какой характеристики ЭВМ можно судить о величине адресного поля при использовании косвенной адресации - по объёму оперативной памяти.

Как сохранить косвенность и справиться с проблемами косвенной адресации? - использование варианта косвенной адресации, в котором хранится информация не в ячейке ОП, а в регистре, получается косвенная регистровая адресация.

## 6.2 Регистровая адресация



Скорость наивысшая.

1 обращение в регистровую память, 2 обращение по адресу, извлечённому из регистра обращаемся в ОП. Получаем все плюшки косвенной адресации.

## 6.3 Относительная адресация

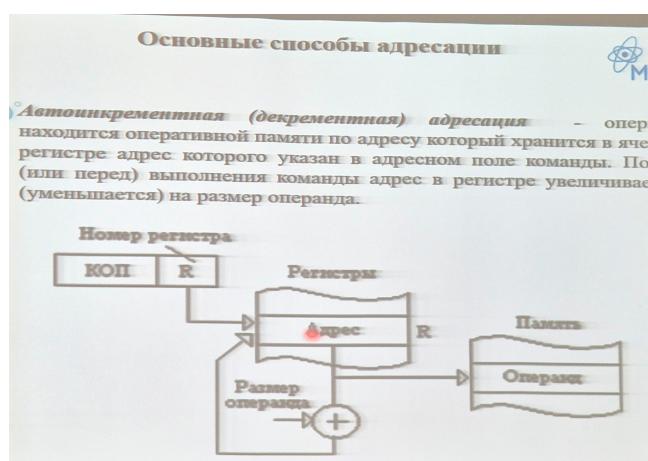


Местоположение операндов в памяти задаётся с помощью смещения относительно чего-то. Обычно относительно некоторого базового адреса. Т.о. получается, что при формировании адреса операнда используется суммирование, а именно сумма адреса базового и смещение.

Крестик на фото и есть суммирование.

СК - счётчик команд - хранит следующую команду. Местонахождение операнда определяется относительно местонахождением команды. В момент, когда надо загрузить команду в память задаётся начальное смещение в виде точки, куда будет загружаться программа - классика при мультипрограммном режиме.

## 6.4 Автоинкрементная адресация

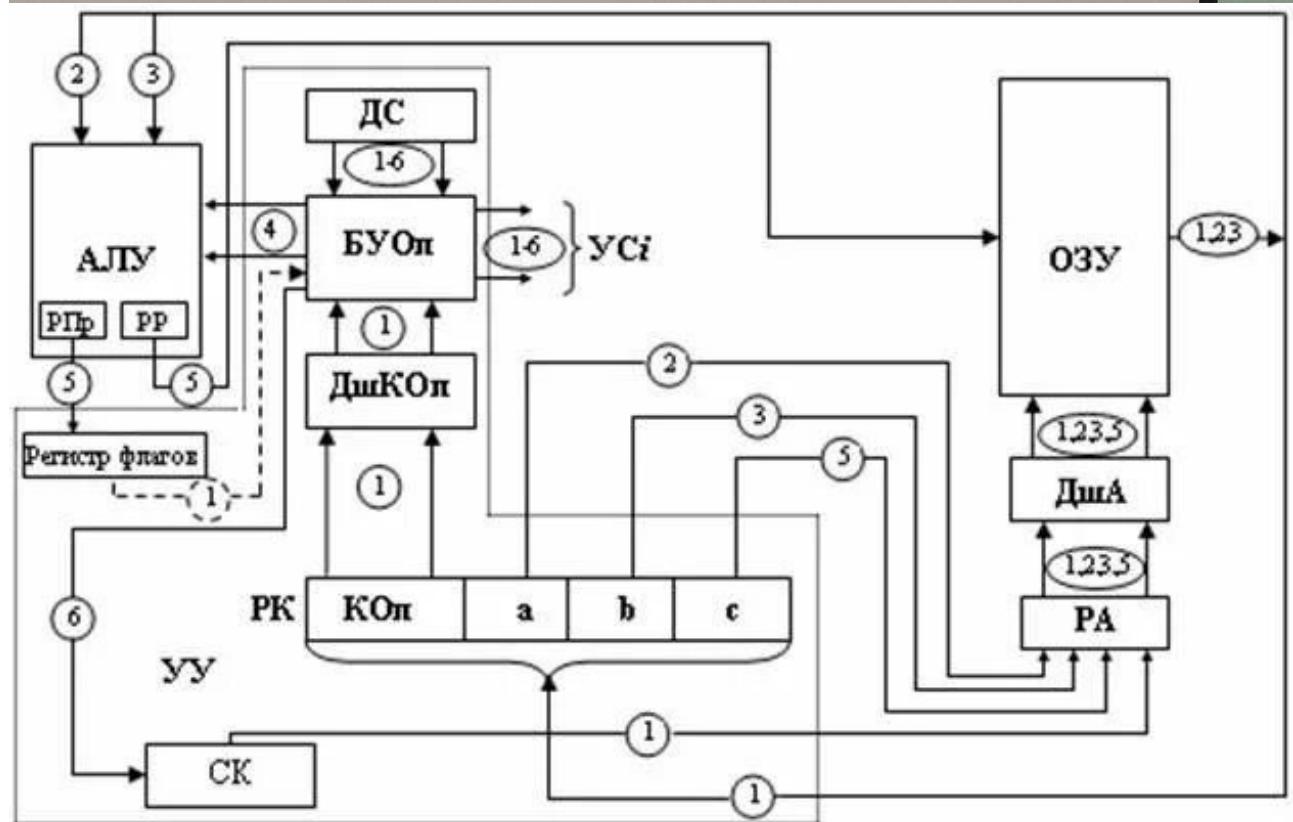
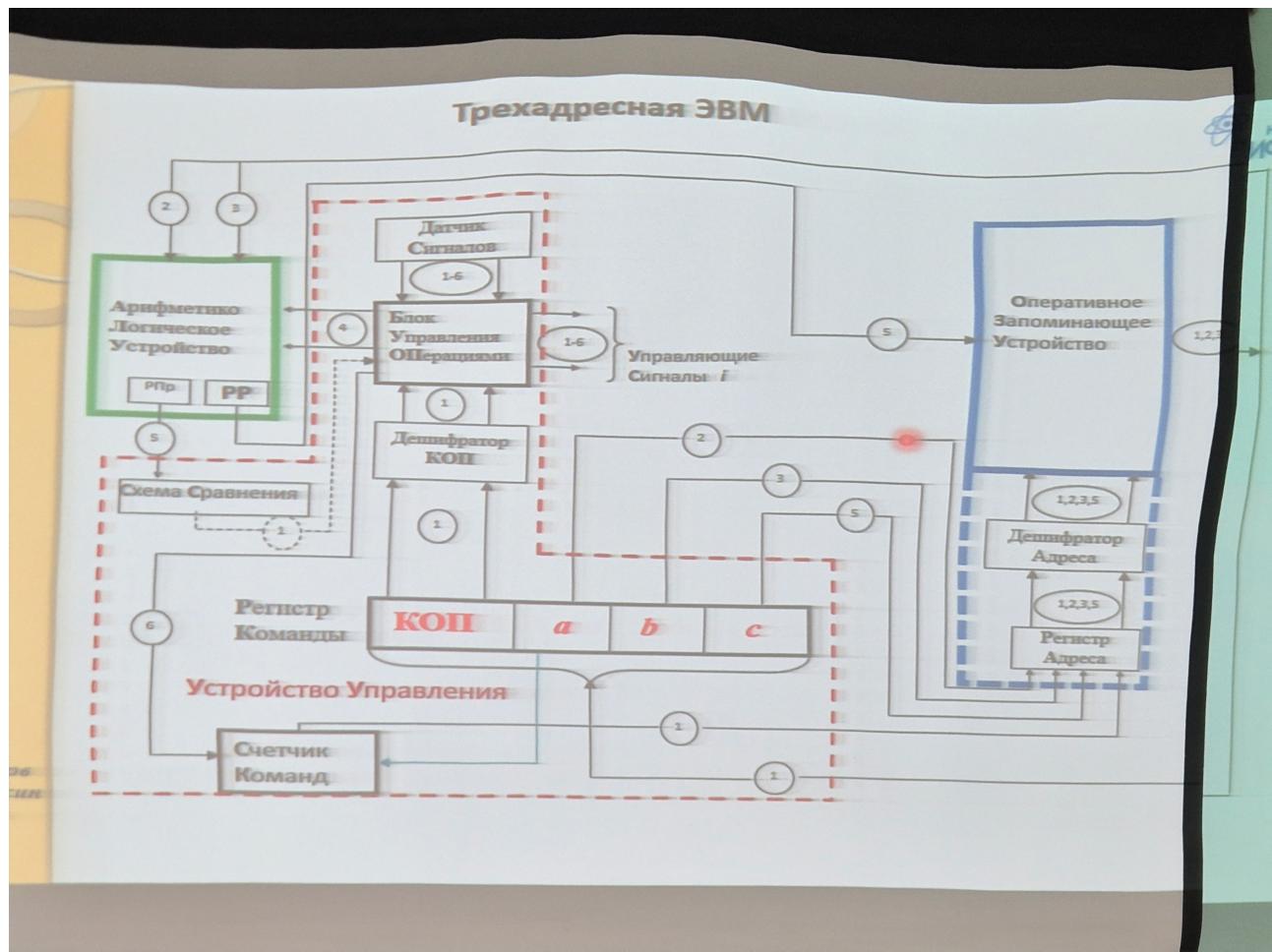


Не что иное как косвенная/косвенная регистровая адресация, у которой адрес операнда может меняться путём увеличения инкремента на константу при каждом использовании этой команды.  
Т.о. при запуске такой команды в цикле получим последовательное обращение подряд идущих элементов (нужно например для массивов).  
Существует в варианте инкремента и декримента - одно увеличивает, другое уменьшает.

## 6.5 Заключение по видам адресаций

Адресаций может быть бесконечно много. Нужно знать неявную, прямую, косвенную, регистрающую, косвенную регистрающую, автоинкремент декримент, относительную адресации.

## 6.6 Трёхадресная ЭВМ



Состоит из:

1. Арифметико-Логическое Устройство. В нём логические элементы. На вход принимает операнды и

пораждается результат (РР) и признак результата (РПр).

2. Оперативное Запоминающее Устройство. Функции: хранение, чтение, запись, выборка.

3. Устройство Управления.

Оно разделяется на 2 основные части:

(а) РК - Регистр команды. Обращаем внимание на то, что речь про 1 и только 1 команду.

Число, которое хранится в этом регистре представляет собой команду. Формат этого числа - формат команды.

Структура регистра команды определяется аппаратной реализацией формата команды. Число записанное туда как раз и занимает поля формата команды и содержит всё что там полагается. Могут обитать 3 адреса: a, b, c, а также сам Код Операции.

Команд много, а регистр может хранить 1 единственную программу - выполняемую в данный момент времени. Именно по этой причине **Функция регистра команды**: хранение команды в процессе её выполнения (той самой 1 единственной).

Логичные вопросы, которые возникают:

- Как команда попадает в регистр команды?
- Как команды в памяти превращаются в программу?

**Программа** - алгоритм для решения задачи, записанный командами ЭВМ.

**Способ адресации команд программы** (не путать со способами адресации операндов) по сути способ связывания команд в программу:

i. **Принудительная адресация программ команд программы.**

Суть заключается в том, что команда в своей адресной части имеет адресное поле с адресом следующей команды (обычно где-то в конце). Получается, что текущая команда выполняемая в данный момент несёт в себе адрес следующей программы. Получается структура, которая называется **связанный список** (блок-чайн без шифрования ;)).

Плюсы и минусы такого способа:

- Никаких усилий для вычисления адреса следующей программы не нужно. Команды могут располагаться как угодно в любых ячейках памяти.
- Занимает место. Приходится тащить поле размером с адрес основной оперативной памяти, а команд много, так что количество необходимой памяти сильно увеличивается.

ii. **Естественная адресация команд программы.**

Берётся память (одномерный массив ячеек или же вектор) и в ней последовательно в том порядке, в котором они должны выполняться вносятся команды. При таком способе к конструкции добавляется устройство, хранящее адрес команды и обладающее возможностью инкремента для добавления некоторой  $\Delta$ , что является размером данной команды, - это называется **счётчик команд**.

Плюсы и минусы такого способа:

- Избавляемся от поля для команды, значит не приходится тратить лишнюю память.
- Нужно класть команды в ячейки строго в порядке выполнения и необходимо потратить дополнительное оборудование для формирования адреса следующей команды + небольшое время добавления дельты к адресу.

Преимущество в размере команды превалирует над первым способом, поэтому обычно используется естественная адресация.

В трёхадресной ЭВМ используется именно естественная адресация (видно счётчик команд ;))

**Счётчик команд** - Instruction Pointer - IP - указатель на команду.

**ОЧЕНЬ ВАЖНО!** Никакой связи IP с интернет протоколами на зачёте нет.

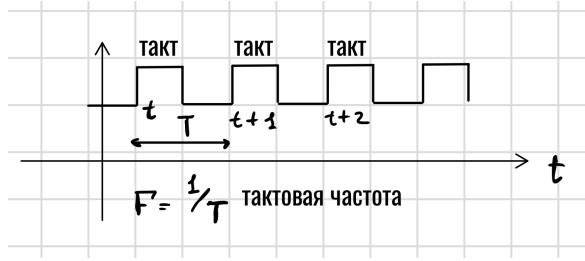
Счётчик команд передает адрес следующей команды на Регистр Адреса ЗУ, указывая на ячейку памяти, где она хранится и считанное по этому адресу число загружается в Регистр Команды.

(б) Управление сигналами:

Помним, что Устройство Управления решает 2 задачи: что делать и когда это делать.

Что делать понятно из команды, расположенной в регистре команд, код операции которой передаётся в дешифратор кода операции. Смысл этого действия в том, чтобы сообщить той части, которая формирует сигналы (Блоку Управления Операциями), какая это команда.

Вопросом когда занимается Датчик Сигнала (служба времени было-бы красивее @Новиков).



Что такое время с точки зрения физики?

Время - это физическая величина, которую не определить, но оно обладает свойствами: время течёт равномерно и непрерывно.

Заменяем физическое время на **время дискретное**.

Для этого мы используем генератор квантов (сигналов) и представляем время при помощи импульсов.

**Импульс** - изменение во времени состояния электрического сигнала несущего значение логической переменной из исходного состояние в противоположное.

Пусть 1 импульс будет равен 1 **кванту времени** или же по другому 1 **такту**, тогда другой импульс будет другим тактом и т.д.

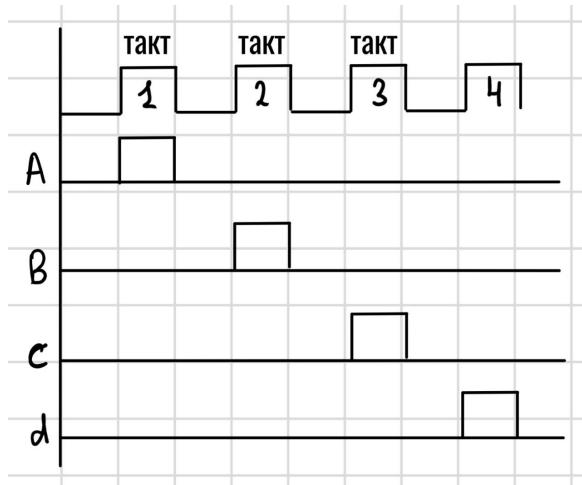
А между импульсами ничего нет. Тогда в нашем новом дискретном времени все события происходят в этих тактах. Поэтому эти такты мы можем пронумеровать ( $t$  - сейчас,  $t+1$  - следующий момент времени,  $t + 2, \dots$ ). Можно и в обратную сторону ( $t - 1, t - 2, \dots$ ).

Порадив такое квантовое время и поняв, что события происходят только в тактовый импульс мы становимся хозяевами того, что происходит в каждый конкретный импульс. Но физическое время никуда не делось и в нём происходят импульсы, а раз они формируются в физическом времени, то у них есть тот же набор характеристик, что и у импульсного сигнала:

- Длительность
- Амплитуда
- Фронт
- Период ( $T$ ) - время между одинаковыми частями 2 соседних импульсов.
- Частота (тактовая частота) - величина обратная периоду:  $F = \frac{1}{T}$ . Её порождает генератор тактовых импульсов.

Изменяя частоту мы изменяем насколько часто будут происходить события в нашем устройстве. Чем быстрее тактовая частота тем быстрее работает устройство.

Мы должны иметь возможность внутри циклы выполнения команды указывать на моменты времени, в которые должны выполняться те самые действия, которые будут повторяться. Т.о. мы должны не только разбить время на кванты, но и иметь возможность выбирать (наверное тут было: какой квант выполнять). В этом помогает **распределитель импульсов цикла**.



В этом примере мы получили, что у нас есть 4 сигнала, которые связаны с друг с другом так, что импульсы на тактах 1, 2, 3, 4 будут соответствовать ...

Подавая с линии А выполнится 1 такт, а на линии С - 3 такт.

Схематически это выглядит так:



Вот тут у меня плохо получилось:

Каждый тактовыё импульс изменяет состояние счётчика на +1. Дешифратор преобразует позиционный код в унитарный.

Пришёл импульс, счётчик переключился, одразовался один ноль (единица двоичная) единичка на дешифраторе переместилась на 1 выход. След на дешифратор, счётчик на 2. 1 не выходе дешифратора переместилась на выход номер 2. Т.о. прошёл всё да в пизду.

Дешифратор кода определяет какие именно импульсы нужны для реализации команды. Объединяет всё блок управления операциями (что и когда). Из него во все стороны и формируются i-е управляющие сигналы, подключающиеся к устройствам.