Билеты

- [x] 1. Две формы представления информации.
  - [x] Способы представления дискретной информации. Системы счисления, используемые в ВТ: двоичная, 8-я, 10-я, 16-я, двоично-десятичная.
- [x] 2. Представление чисел с фиксированной точкой. Прямой, обратный и дополнительный код. Формирование битовых признаков переноса, переполнения, отрицательного результата, нуля.
- [x] 3. Представление символьных и строковых данных
  - [x] Принципы построения кодовых таблиц ASCII, КОИ-8, ISO8859-5, Windows-1251 UTF-8, UTF-16.
- [x] 4. Базовые элементы вычислительной техники
  - [x] ячейки
    - [x] DRAM
    - [x] SRAM
  - [x] регистры
  - [x] шины
  - [x] тактовые генераторы
  - [x] вентили
  - [x] логические схемы
  - [x] триггеры
  - [x] счетчики
  - [x] сумматоры
- [ ] 5. Структура и принцип функционирования ЭВМ.
  - [ ] Порядок функционирования простого процессора на примере калькулятора.
- [x] 6. Операционная система Unix
  - [x] ядро ОС
  - [x] файловая система.
- [x] 7. Операционная система Unix
  - [x] интерпретаторы,
  - [x] стандартные потоки ввода вывода
  - [x] фильтры.
- [x] 8. Операционная система Unix
  - [x] основные команды,
  - [x] права файлов
  - [x] способы задания прав
- [ ] 9. Состав и структура БЭВМ
  - [ ] Адресные пространства БЭВМ
  - [ ] Система команд БЭВМ
  - [ ] форматы команд
  - [ ] машинные циклы
- [ ] 10. Организация вычислений в БЭВМ
  - [ ] сдвиги
  - [ ] арифметические и логические операции
  - [ ] цикл выборки команды
- [ ] 11.
  - [ ] Организация массивов данных

    - [ ] Режимы адресации
    - [ ] Цикл выборки адреса БЭВМ
    - [ ] Цикл выборки операнда БЭВМ
- [ ] 12. Управление вычислительным процессом в БЭВМ
    - [ ] Команды ветвлений
    - [ ] цикл исполнения команды LOOP
- [ ] 13. Подпрограммы в БЭВМ
    - [ ] Цикл исполнения команд перехода и возврата из подпрограммы
    - [ ] Стек, передача параметров. Позиционно-независимый код
    - [ ] Загрузчик и библиотеки
- [ ] 14. Организация ввода-вывода в вычислительных системах
    - [ ] Инициация обмена, передача информации и завершение обмена
    - [ ] Драйверы
- [ ] 15. Организация ввода-вывода в БЭВМ
    - [ ] Устройства ввода-вывода, команды
- [ ] 16. Организация асинхронного обмена в БЭВМ
    - [ ] Пример программы
    - [ ] Временные издержки асинхронного обмена
- [ ] 17. Организация прерываний в БЭВМ
    - [ ] Вектора прерываний, контроллер прерывания
- [ ] 18. Организация обмена по прерыванию программы в БЭВМ
    - [ ] Пример программы
    - [ ] Цикл прерывания
- [ ] 19. Понятие многоуровневой ЭВМ
    - [ ] Понятие и пример программы на разных уровнях
- [ ] 20. Микропрограммный уровень БЭВМ
    - [ ] Структура МПУ
    - [ ] Форматы микрокоманд
- [ ] 21. Структура и принципы работы АЛУ
    - [ ] Структура и принцип работы коммутатора
    - [ ] Регистр состояния БЭВМ
- [ ] 22. Микропрограммное управление вентильными схемами
    - [ ] Схема управления
    - [ ] Интерпретатор БЭВМ
- [ ] 23. Архитектура ЭВМ
    - [ ] Гарвардская и фон-Неймановская архитектура
    - [ ] Организация обмена архитектуры ЭВМ с использованием шин
- [ ] 24. Архитектура многопроцессорных ЭВМ
    - [ ] Системный коммутатор
    - [ ] Архитектура UMA
    - [ ] Архитектура NUMA
- [ ] 25. Структура современных процессоров
    - [ ] Окружение процессора
    - [ ] CISC
    - [ ] RISC
    - [ ] VLIW

- [ ] 26. Адресуемая память, организация и временные диаграммы
  - [ ] Конструктивные особенности современной памяти
- [ ] 27. Память, ориентированная на записи (блочная память)
  - [ ] Организация дисковой памяти и памяти на магнитных лентах
- [ ] 28. Характеристики запоминающих устройств
  - [ ] Пирамида памяти
- [ ] 29. Ассоциативная память
  - [ ] Кэш-память
  - [ ] Влияние промахов кэш-памяти на производительность
- [ ] 30. Предназначение и организация виртуальной памяти
  - [ ] Сегментно-страничная организация
  - [ ] Устройство управления памятью (MMU)
  - [ ] Буфер трансляции (TLB)
- [ ] 31. Сетевые технологии
  - [ ] Понятие сети ЭВМ
  - [ ] Классификация компьютерных сетей
  - [ ] Сообщение и пакет
  - [ ] Модель взаимодействия открытых систем
- [ ] 32. Модель TCP/IP:
  - [ ] передающая среда
  - [ ] канальный и сетевой уровень
  - [ ] Адресация, передача и маршрутизация пакетов
- [ ] 33. Модель TCP/IP:
  - [ ] выделение адресов (DHCP)
  - [ ] сервисы имен
  - [ ] транспортный и прикладной уровни
- [ ] 34. Интерфейсы ввода-вывода
  - [ ] Контроллеры внешних устройств
  - [ ] Уровни стандартизации, сопряжения с системной шиной, циклы обмена
  - [ ] Регистры контроллера
- [ ] 35. Параллельная передача данных
  - [ ] Контроллеры параллельной передачи и приема
- [ ] 36. Синхронные последовательные интерфейсы
  - [ ] Контроллеры последовательной передачи и приема
- [ ] 37. Асинхронный обмен
  - [ ] Принципы деления частоты
  - [ ] Формат кадра
- [ ] 38. Контроллер передачи асинхронного последовательные интерфейса.
- [ ] 39. Контроллер приема асинхронного последовательные интерфейса.
- [ ] 40. Организация прямого доступа к памяти
  - [ ] Контроллер DMA(ПДП)

1. Две формы представления информации. Способы представления дискретной информации. Системы счисления, используемые в ВТ: 2ая, 8ая, 10ая, 16ая, двоично-десятичная.

Существует две формы представления информации - аналоговая(или непрерывная) и дискретная(или цифровая).

Величины, представленные в аналоговой форме могут принимать любые значения в каком-то диапазоне, и количество значений, которые можгут принимать эти величины - бесконечно велико. Бесконечно даже если взять самый маленький промежуток: возьмём 0-0.001. Получается что значения из этого промежутка могут быть 0.0002, 0.00002342874 и так далее. Это объясняет название "непрерывная" - такие величины не имеют разрывов, или промежутков между значениями.

Величины, представленные в дискретной форме, принимают не все возможные значения, а лишь конкретные и вполне-определённые. В отличие от непрерывной величины количество значений дискретной величины всегда будет конечным.

Возьмём простой пример: измеряем напряжение в цепи. Предположим, что напряжение в этой конкретной цепи будет ограничено в диапазоне от 0 до 5 вольт.

Для дискретного вида информации: напряжение будет представляться в виде конечной и определённой величины - к примеру, 1,4 вольта, если значения будут представляться с точностью до одной десятой вольта.

Для аналогового вида информации: полученным напряжением может быть, к примеру, 1,4 вольта. Или 1,44 вольта, или 1,442 вольта, или 1,442138739853.... так можно уточнять до бесконечности, в чём главное как преимущество, так и недостаток аналоговой информации.

Аналоговые эвм точны, но одновременно крайне громоздки, сложны в использовании и узконаправлены, зато крайне эффективны в своей задаче, в то время как цифровые - более универсальны, проще в реализации, хотя и проигрывают в быстродействии аналоговым эвм в конкретных задачах.

> В качестве примера применения аналоговой информации можно привести виниловую пластинку, у которой звуковая дорожка непрерывно изменяет форму.

> А для дискретной - компакт-диск, у которого звуковая дорожка содержит участки с различной отражающей способностью.

Способы представления дискретной информации:

Каждое значение из набора исходных данных задачи и результатов её решения может быть представлено в ЭВМ в виде нескольких электрических сигналов, один из которых соответствует число единиц в значении, другой - числу десятков, третий - числу сотен и так далее. Однако, это не не наилучшее решение с технических позиций в виду сложной обработки нескольких сигналов. Значительно проще обрабатывать два состояния. Тем более это обуславливается целесообразностью представления информации в таком виде:

- Тумблер on/off. Сигнал есть/нету. Конденсатор заряжен/разряжен e.t.c

Так создатели первых ЭВМ решили перейти от привычной десятичной системы счисления к двоичной. Обе эти системы являются позиционными, но ЭВМ куда проще работать с двоичной, тогда как человеку для записи и чтения чисел удобна десятичная.

Для сокращения трудоёмкости ручной обработки кодов чисел и команд применяется восьмеричная и шестнадцатеричная системы счисления.

- Восьмеричное представление
  - от 0 до 7. Восьмеричная система счисления была более распространена в ранних компьютерах, которые работали с группами бит, размер которых был кратен 3. (12, 24, 36). Это делало 8ую СС удобной для представления и манипулирования данными.
- Шестнадцатеричное представление
  - от 0 до F. Удобна тем, что минимальной адресуемой единицей памяти является байт (8 бит), значения которого удобно записывать двумя 16-разрядными цифрами.
  - используется в rgb

Также стоит упомянуть о двоично-десятичной системе счисления. Она широко используется в цифровых устройствах, где основная часть операций связана не с обработкой и хранением вводимой информации, а с самим её вводом и выводом в десятичном представлении(к примеру, вывести десятичное число на дисплей калькулятора)

- Двоично-десятичное представление
  - (1001) (0010) (0101) ... система счисления, где десятичные цифры от 0 до 9 представляют 4-разрядные двоичные комбинации от 0000 до 0001.
  - очевидно, требует больше памяти, в сравнении с обычной двоичной системой счисления
  - усложняется счёт в двоично-десятичной системе
  - для дробных чисел при переводе в десятичный формат не теряется точность

2. Представление чисел с фиксированной точкой. Прямой, обратный и дополнительный код. Формирование битовых признаков переноса, переполнения, отрицательного результата, нуля.

Наборы двоичных цифр позволяют закодировать любую информацию, вплоть до графической и звуковой. Один из способов представить вещественное число в памяти ЭВМ - число с фиксированной запятой. Так обозначают числа, где знак, целая и дробная части имеют фиксированное количество бит, что приводит к тому, что все числа с фиксированной точкой должны размещаться в разрядной сетке. Этот формат удобен при работе с числами, где важна конкретная точность до определённого знака после запятой. С помощью фиксированной точки представляются как целые, так и вещественные числа.

Прямой код представляет собой самый прямой способ кодирования числа в двоичной системе, где младшие биты числа представляют абсолютное значение числа, а старший бит используется для обозначения знака числа (0 - для положительных чисел, 1 - для отрицательных).

Например: 6 в 10сс (>0) = 0110 в 2сс -6 в 10сс (<0) = 1110 в 2сс

Обратный код для положительных чисел совпадает с прямым, для отрицательных чисел образуется из прямого кода положительного числа путем инвертирования битов

Например: 6 в 10сс (>0) = 0110 в 2сс -6 в 10сс (<0) = 1001 в 2сс

Дополнительный код получается прибавлением единицы к младшему разряду обратного кода. Он часто используется для представления отрицательных чисел, так как упрощает операции вычитания.

Например: 6 в 10сс (>0) = 0110 в 2сс -6 в 10сс (<0) = 1010 в 2сс

Формирование битовых признаков: *Признаки: N, Z, V, C*

1. N - `negative`. Признак отрицательного результата. Выставляется, если в знаковом(последнем) бите числа - единица.
2. Z - `zero`. Признак нулевого результата. Выставляется, если ВСЕ биты числа - нулевые. (По сути это (не(OR)) всех битов числа)
3. V - `overflow`. Признак переполнения для знаковых чисел. Выставляется при несовпадении переносов(сложение) или заёмов(вычитание) ИЗ и В старший разряд числа.
4. C - `carry`. Признак переполнения для беззнаковых чисел. Выставляется при заёме В старший разряд(вычитание) или при переносе ИЗ старшего разряда(сложение).

3. Представление символьных и строковых данных. Принципы построения кодовых таблиц ASCII, КОИ-8, ISO8859-5, Windows-1251, UTF-8, UTF-16.

`Символ в ЭВМ` - это графическое изображение, используемое для создания слов, текстов, цифр и других видов информации. К символам относятся буквы, цифры, знаки препинания, символы валют и т.д.

`Строка в ЭВМ` - это последовательность символов.

В современных вычислительных системах есть два разных способа размещения строк:

1. Конец строки - специальный символ NUL. Такие строки называются NULL terminated String.
2. Первое слово строки - хранит длину строки.

Разные ОС по-разному кодируют конец строки:

- Unix-системы используют CR(carriage return).
- Windows использует LF(line feed) + CR

Поэтому символы LF, CR, BS(back symbol), NUL - также есть в кодовой таблице.

`Кодовая таблица` - это таблица соответствия каждому символу его уникального порядкового номера или кода.

Таким образом, `представление текстовой информации` в ЭВМ основывается на кодировании символов при помощи `кодовой таблицы`. То есть все символы хранятся в памяти в виде числовых кодов, определённых кодировкой символов.

Чтобы отобразить эти числовые кода на экране, используются шрифты(fonts) для преобразования числового кода в графическое изображение. `Шрифты` - это наборы графических образов символов в заданном алфавите. Обычно устанавливаются вместе с операционной системой, драйвером принтера или микропрограммой BIOS. Отдельно хранятся изображения строчных и заглавных букв, цифр, специальных символов и т. д.

Шрифты бывают `растровыми` и `векторными`. В растровых хранится изображение символа. В векторных хранится последовательность линий и кривых, которые нужно начертить для отображения символа.

Принципы построения кодовых таблиц:

ASCII

> Разработана в США в 1963 г.
>
> ASCII использует 7 бит для представления символа, что позволяет кодировать 128 различных символов, 8-й (старший) разряд использовался для контроля четности битов передаваемого по каналам связи символа.
>
> > Первые 32 кода (0-31) предназначены для управляющих символов (например, перенос строки, возврат каретки).
> >
> > Коды с 32 по 126 используются для стандартных символов клавиатуры, включая буквы, цифры и знаки препинания.

Extended ASCII

> Разработана в США в 1970-х годах.
>
> Расширен набор символов до 256 - был введён 8 бит

КОИ-8

> Разработана в СССР в 1974 г.
>
> Разработана для представления кириллицы после появления extended ASCII. Латинские символы совпадают с extended ASCII, а кириллица располагается в верхней части таблицы (коды 128-255). Также, помимо кириллицы, была добавлена псевдо-графика.
>
> Следует отметить интересную особенность КОИ-8: если ПО не поддерживало 8-битные кодировки и обрезало старший бит при контроле чётности, то русские буквы перемещались в младшую часть таблицы, и их можно было прочитать по сходным по начертанию английским символам, то есть, транслитом.
>
> КОИ-8 до сих пор используется в качестве стандарта для обмена электронной почтой.

ISO-8859-5

> Разработана Международной организацией по стандартизации в 1988 г.
>
> Нижняя часть таблицы кодировки полностью соответствует кодировке ASCII. Предоставляет символы кириллицы, включая русские, болгарские и сербские символы. Неудобная кодировка в виду отсутствия частоиспользуемых символов, таких как тире, кавычки, градус(—, «», °) и др. Русские буквы по алфавиту, 1 символ – 1 байт.

Windows-1251

> Разработана в 1990-х годах компанией Microsoft.
>
> Стандартная 8-битная кодировка для русских версий Microsoft Windows. Выгодно отличается от других кириллических кодировок наличием практически всех возможных символов, использующихся в русской типографии. Русские буквы по алфавиту, 1 символ – 1 байт.

Unicode

> Разработана в 1991 г. для унификации кодировок и представления всех символов всех письменных систем мира. Изначально было выделено 16 бит для символов, но в конечном итоге юникод был расширен до 21 бита. (>1 млн символов) Таблица символов ставит каждому символу в соответствие значение в виде U+[число в 16сс]. Например: U+0053 соответствует заглавной латинской букве S
>
> > `Unicode = UCS + UTF` (Universal character set + Unicode transformation format)
> >
> > `UCS` - набор символов. Присваивает каждому символу уникальный код в виде U+[число в 16сс]. Например: U+0053 соответствует заглавной латинской букве S
> >
> > `UTF` - способ кодирования символов из UCS (utf-8, utf-16, utf-32, ...) UTF определяет как символы из UCS переводятся в байты для хранения и передачи

UTF-8

> Разработана в 1992 году для эффективного представления символов Unicode. Каждый `символ` может кодироваться от `1 до 4 байтами`.
>
> В UTF-8 первые 128 символов полностью совпадают с ASCII. Если символ в UTF-8 превышает 7 разрядов, то он представляется в виде двух байтов. При этом первые 5 битов предваряются преамбулой 110 и упаковываются в байт, следующий байт идет с преамбулой 10, и остальные 6 разрядов копируются во второй байт. Символы, превышающие 11 разрядов, кодируются в соответствии с определенными правилами. Например, каждому русскому символу соответствует 2 байта. Однако проблема UTF-8 состоит в том, что один символ может быть закодирован 1-4 байтами, что делает сложным определение длины строки в байтах по количеству символов.
>
> Пример: 00010010 (H) 10100110 (е) 00110110 (l) 00110110 (l) 11110110 (о).

UTF-16

> Разработана в 1996 году для эффективного представления символов Unicode и решения проблемы несоответствия длины строки и количеству байтов в utf-8. Каждый `символ` кодируется `2 или 4 байтами`.
>
> Буквы русского и английского алфавита передаются без изменений при помощи 16 бит, при этом старшие незначащие биты принимают нулевое значение. Остальные символы, номера которых в двоичном представлении формируются количеством бит больше 16, кодируются 32 битами с использованием специального алгоритма.
>
> Однако, несмотря на то что UTF-16 упрощает определение длины строки, для представления строки в UTF-16 используется больше памяти, чем в UTF-8.
>
> Пример: 00010010 00000000 (H) 10100110 00000000 (е) 00110110 00000000 (l) 00110110 00000000 (l) 111110110 00000000 (о).

4. Базовые элементы вычислительной техники: ячейки, регистры, шины, вентили, тактовые генераторы, логические схемы, триггеры, счетчики, сумматоры.

> Для хранения информации и данных в ЭВМ используется оперативная память, информция в которой сохраняется, пока ЭВМ включена. Оперативная память состоит из `ячеек памяти`.

Ячейка памяти

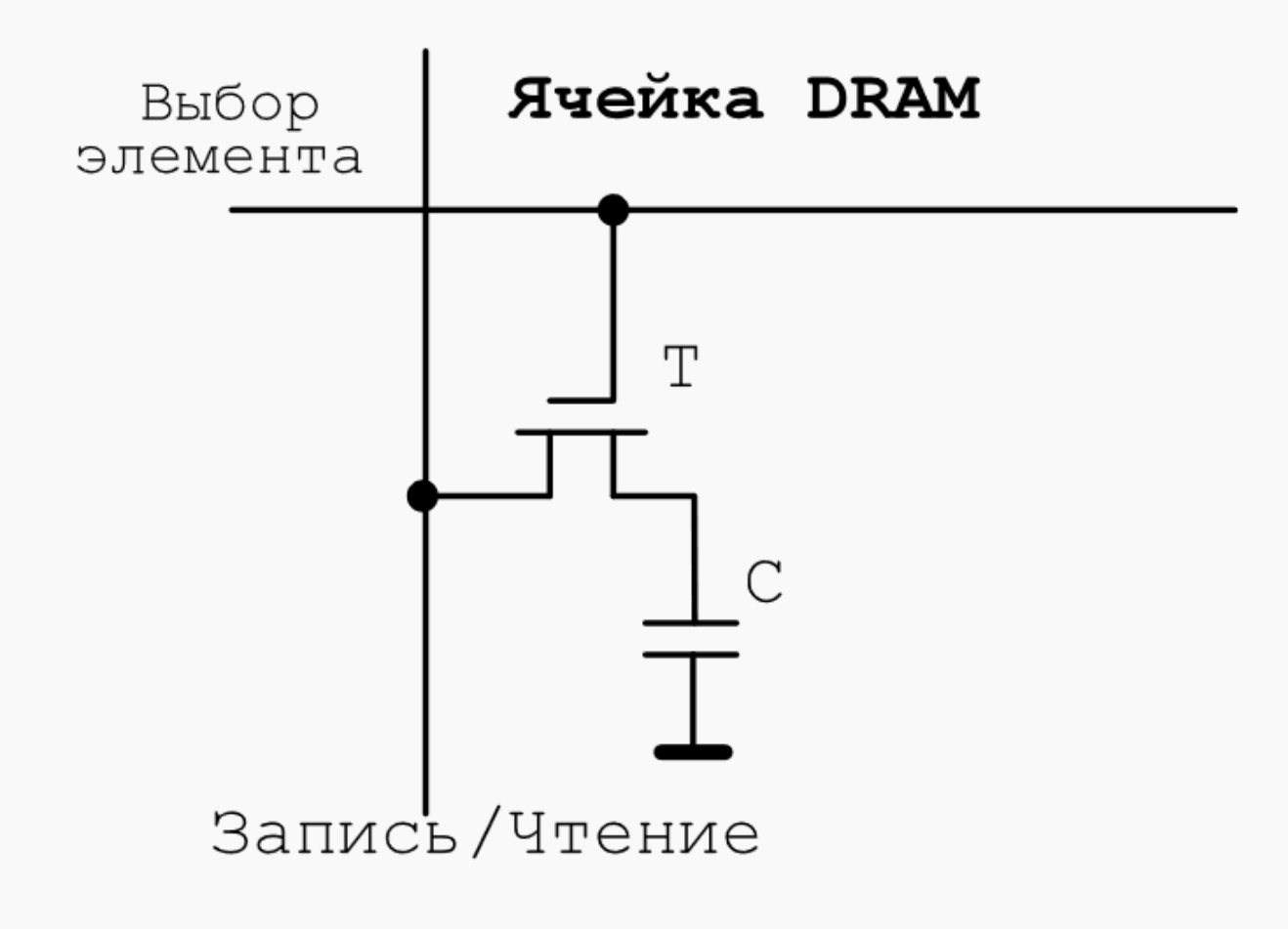
Ячейка — это минимальный адресуемый элемент запоминающего устройства ЭВМ.

- ячейки имеют адрес (порядковый номер, число), по которому к ним обращается процессор
- находятся в одном из двух физических состояний, в первом создаётся высокий уровень выходного напряжения, а во втором - низкий, что воспринимается за двоичные 0 и 1.

В современных ЭВМ существует два вида оперативной памяти - DRAM и SRAM. Вот как они устроены:

1. *Ячейка DRAM* - Dynamic random access memory. Информация хранится на конденсаторе, где разряженный конденсатор эквивалентен 0, а заряженный - 1.

Ячейка состоит из транзистора(T) и конденсатора(C)



> 1) запись: Чтобы записать "1" или "0", на линию выбора элемента подаётся высокое напряжение. => транзистор T открывается Затем, на линию "Запись/Чтение" подаётся высокое напряжение в случае если записываем 1 и низкое напряжение, если записываем 0. => конденсатор заряжается, либо разряжается в зависимости от поданного напряжения На линию выбора элемента прекращается подача высокого напряжения => транзистор закрывается

> 2) чтение: Чтобы прочитать значение ячейки на линию выбора элемента подаётся высокое напряжение. => транзистор T открывается Затем, заряд на конденсаторе либо поднимет напряжение на линии "Запись/Чтение", либо оставит без изменений => прослушиваем напряжение на линии "Запись/Чтение" и если оно низкое - значит в ячейке 0, а если высокое -

> значит в ячейке 1 => На линию выбора элемента прекращается подача высокого напряжения => транзистор закрывается

!Важно - как вы уже могли догадаться, после чтения ячейки конденсатор разряжается, потому что его заряд перетекает на линию "Запись/Чтение", поэтому для DRAM нужно постоянное питание. Если не производить чтение памяти, то конденсатор в любом случае постепенно разряжается, соответственно, чтобы избежать потери информации, в современных ЭВМ предусмотрено устройство регенерации памяти, которое раз в определённое время сканирует всю оперативную память, считывает каждую ячейку и возобновляет заряд на её конденсаторе.

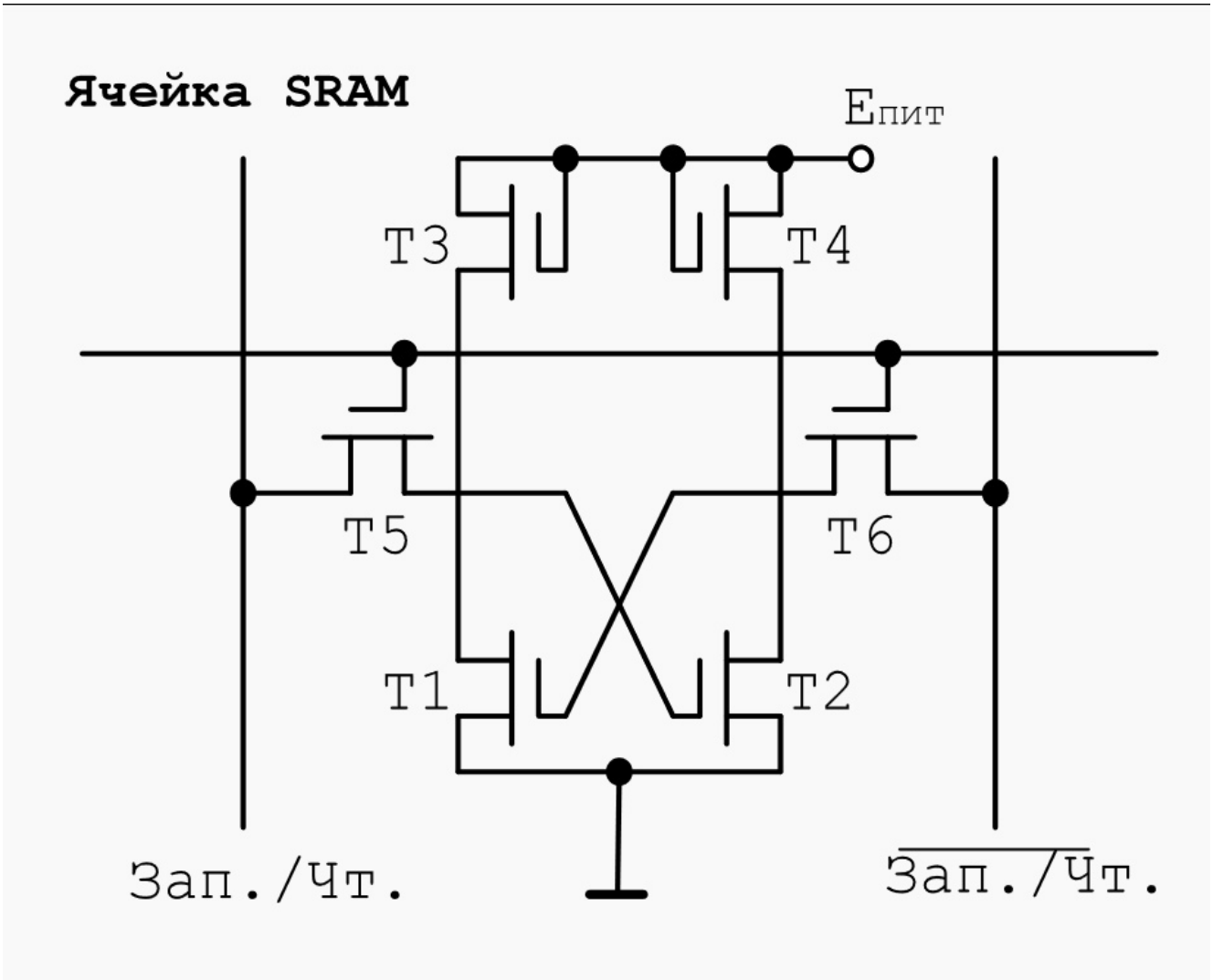
Резюме:

> - DRAM дешёвая, т.к. нужен всего 1 транзистор на 1 бит информации.
> - DRAM легче производить в виду уже упомянутых выше конструктивных особенностей
> - DRAM нужно постоянное питание и регенерация, потому что конденсатор разряжается со временем и может произойти потеря информации.

2. *Ячейка SRAM* - Static random access memory. Информация хранится в виде состояния бистабильного триггера.

Информация хранится в попарном состоянии транзисторов.

Ячейка состоит из 6 транзисторов(T1-T6).



Шина данных

Шина - это объединение линий передачи(или проводов). В ЭВМ шина служит для передачи информации от одного регистра к другому.

> Чтобы избежать помех -> скручивают прямой и обратный провода, образуя витую пару, где помехи взаимно уничтожаются.

> Шина = информационные линии + управляющие линии + земля

> Шины бывают однонаправленными и двунаправленными, что обычно указывается стрелками на них.

Триггеры

Триггер - это класс электронных устройств, обладающих способностью длительно находиться в одном или более устойчивых состояний и изменять их под воздействием внешних сигналов.

Триггеры бывают:

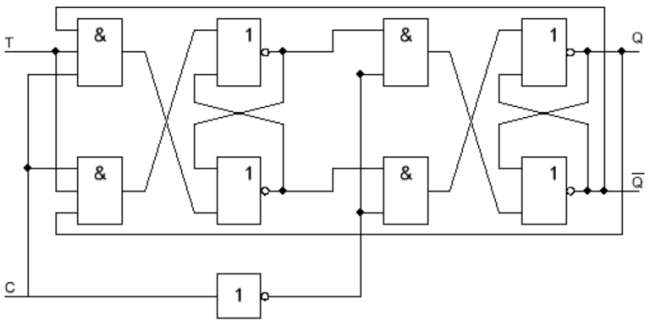
1. Моностабильные - используются для генерации однократного импульса / задержки сигнала: моностабильный мультивибратор
2. Бистабильные - используются для хранения битов информации в цифровых схемах: D, T, RS, JK
3. Астабильные - используются для тактовых генераторов: триггер Шмидта, астабильный мультивибратор

Также триггеры делятся по типу на синхронные(с импульсным входом C - clock) и асинхронные (без C)

Виды бистабильных триггеров (наиболее часто используемые в ЭВМ):

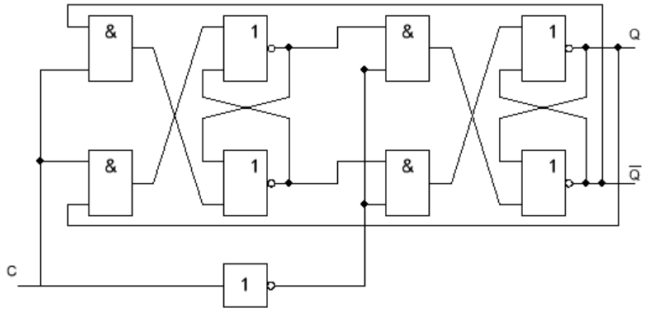
- T-триггер - счётчик по модулю два, меняющий состояние на противоположное при каждом поступающем импульсе на вход C (Clock).

> *Особенности и назначение:* Используется в цифровых счетчиках и делителях частоты для переключения состояний. Имеет один вход - информационный(T) в асинхронном виде или два входа - информационный(T) и импульсный(C) в синхронном виде. Имеет всегда 2 выхода.

Описание работы	**Схема T-триггера на логических элементах**
Синхронный T-триггер: При T=1, триггер переключается с каждым импульсом C; При T=0, состояние не меняется.	

Описание работы | **Схема Т-триггера на логических элементах**

Асинхронный Т-триггер: С каждым импульсом `C` триггер меняет своё состояние на `противоположное`



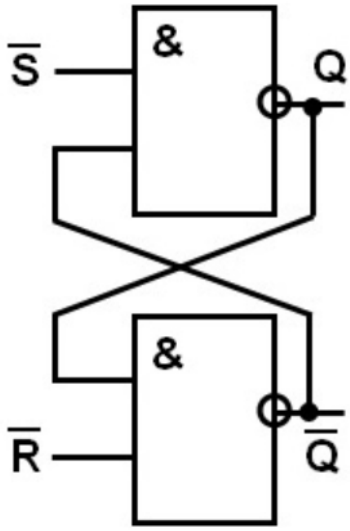
- `RS-триггер` - название триггера говорит само за себя. `RS` - `reset`/`set`

> *Особенности и назначение:* Используется в цифровых схемах для хранения одного бита информации. Имеет два входа(R и S) и два выхода.

Описание работы | **Схема асинхронного RS-триггера на логических элементах "2И-НЕ"**

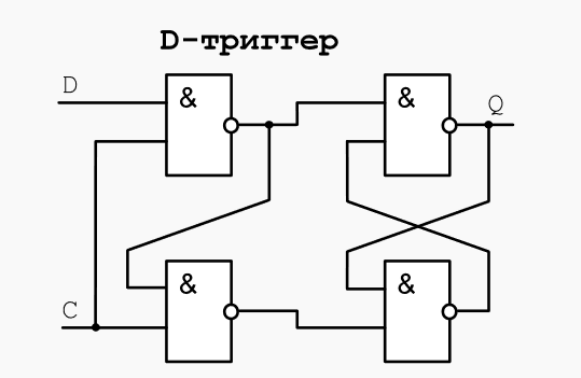
`S=1, R=0` задает состояние `1`; `S=0, R=1` - состояние `0`; `S=0, R=0` - `сохраняет` состояние; `S=1, R=1` - `недопустимо`.

S^t	R^t	Q^{t+1}
0	1	0
1	0	1
0	0	Q^t
1	1	–



- `D-триггер` - триггер, принимающий и запоминающий значение, поданное на вход D (Data), на каждом восходящем фронте тактового сигнала.

> *Особенности и назначение:* Используется для записи, хранения и передачи данных. Имеет два входа - данные(D) и синхронизация(C) и два выхода.

Описание работы	Схема синхронного D-триггера
Если D=1, то на сл. восходящем фронте тактового сигнала триггер установится в 1. Аналогично с D=0	

- JK-триггер - триггер, подобный ранее рассмотренному RS-триггеру, за исключением того, что вариант R=1, S=1 для него допустим. В этом случае JK-триггер изменит своё состояние на противоложное.

Регистры

Регистр - это устройство, состоящее из набора бистабильных триггеров, которые служат для записи, хранения и считывания двоичных данных.

> Каждый бит данных представлен одним триггером, так что регистр размером в n бит будет состоять из n триггеров.

Тактовый генератор

Тактовый генератор – это устройство, генерирующее электрические импульсы заданной частоты. Используется для синхронизации процессов передачи информации между устройствами ЭВМ.

> Все пересылки данных, арифметические или логические операции могут происходить *только* в строго заданное время, определяемое размером такта генератора. Одни операции могут выполняться по фронту сигнала, другие по значению, некоторые по спаду сигнала. К концу такта *все* операции должны быть завершены

> Как устроен тактовый генератор?
>
> Это явный пример астабильного триггера - астабильный мультивибратор на основе триггера Шмидта При использовании триггера Шмитта, конденсатор заряжается и разряжается через резистор, и процесс снова повторяется, когда напряжение на конденсаторе достигает определенного порога. Это приводит к смене состояния на выходе триггера и генерации квадратной волны.

Вентили

Для передачи информации в строго заданное ТГ время в ЭВМ предусмотрены вентили.

> Это логические устройства, которые функционируют по принципу крана или тумблера и содержат два входа и один выход. На вход подаются информационный и управляющий сигналы. На выходе вентиля будет единица только в том случае, если И информационный И управляющий сигналы будут равны единице (кран открыт и течёт вода).

Логические схемы

`Функциональная логическая схема` - это совокупность логических элементов и связей между ними.

Логические элементы: `Буфер`, `Инвертор`, `И`, `ИЛИ`, `И-НЕ`, `ИЛИ-НЕ`, `xor`, `xnor`

Счётчики

`Счётчик` - это устройство, которое определяет число поступивших импульсов.

> Основной параметр счётчика — `модуль счёта` То есть, максимальное число единичных сигналов, которое может быть сосчитано счётчиком.

Сумматоры

`Сумматор` - это Сумматор — устройство, преобразующее информационные сигналы (аналоговые или цифровые) в сигнал, эквивалентный сумме этих сигналов.

5. Структура и принцип функционирования ЭВМ. Порядок функционирования простого процессора на примере калькулятора.

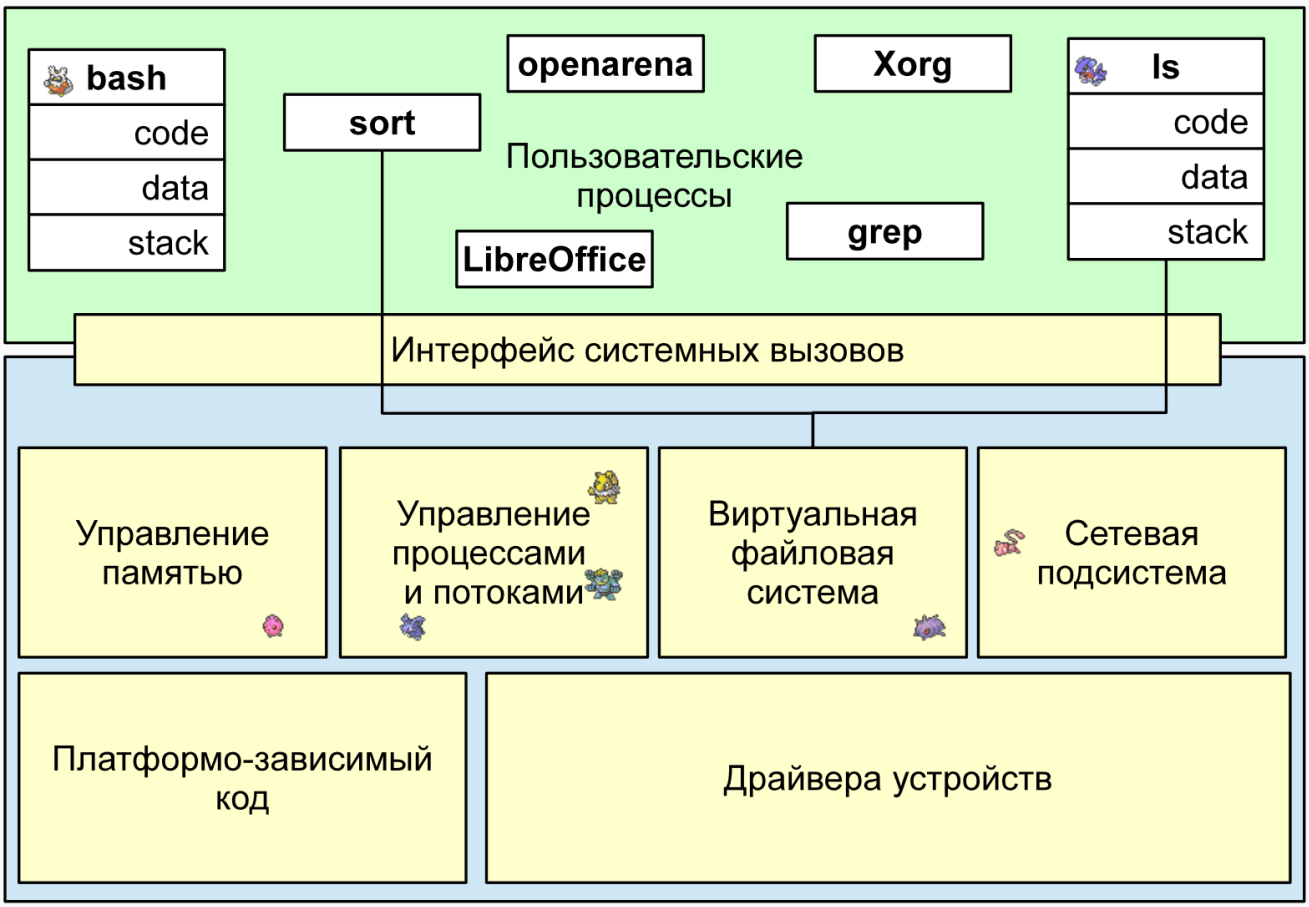
> ЭВМ = cpu + память + устройства ввода-вывода cpu = АЛУ +

6. Операционная система Unix, ядро ОС, файловая система.

`UNIX` — семейство переносимых, многозадачных и многопользовательских операционных систем.

> Отличие UNIX-подобных систем от других ОС в том, что это изначально многопользовательские многозадачные системы. То есть в один и тот же момент времени сразу множество людей может выполнять множество вычислительных задач (процессов).
>
> Также заслуга Unix в её мультиплатформенности. Ядро системы написано таким образом, что его легко можно приспособить практически под любой микропроцессор.

Ядро *NIX



`Ядро ОС` - это набор подсистем и программ, которые управляют оборудованием и программами пользователя. Служебные операции ОС выполняются в ядре. Взаимодействие программ пользователя с ядром осуществляется с помощью *интерфейса системных вызовов*.

`Интерфейс системных вызовов` - это стандартизированное средство взаимодействия программы пользователя и ядра: доступ к сервисам ядра можно получить только через этот интерфейс.

> Основные подсистемы, нужные в любой ОС:
>
> 1. Управление памятью
> 2. Управление процессами и потоками
> 3. Виртуальная файловая подсистема
> 4. Сетевая подсистема
> 5. Драйвера устройств
> 6. Платформо-зависимый код

Файловая система

> Понятие `файла` является одним из наиболее важных для ОС UNIX.
>
> > Все файлы, с которыми огут манипулировать пользователи, располагаются в файловой системе, представляющей собой дерево, где `промежуточные вершины` = `каталоги`, а `листья` - `файлам` и `пустым каталогам`.
>
> Наверху иерархии файловой системы директория `root` (корень иерархии), имеющая путь `/`.

Абсолютный путь - это путь от корня ФС Относительный путь - это путь от текущего рабочего каталога
! В UNIX всё является файлом, кроме потоков, процессов и ядра

> Каждый каталог содержит два обязательных файла. Файл «.» -это ссылка на текущий каталог и файл « .. » - это ссылка на директорию-родителя.

В файловой системе есть специальные виды файлов, которые могут помочь организовать необходимую для пользователя или программ иерархию файлов - ссылки на другие файлы.

> Ссылки бывают двух типов: жесткие и символические.
>
> - Символические содержат в себе только путь к файлу
> - Жёсткие имеют точно такой же inode, что и у исходного файла
>
> > В ОС UNIX каждому файлу внутри файловой системы для каждого накопителя присваивается свой уникальный номер - inode.
> >
> > Директория - это просто файл, где указаны имена файлов и их inode. Если два имени файла имеют один и тот же inode (внутри одной точки монтирования) - значит это один и тот же файл с таким же содержимым. Или можно сказать, что один файл является жесткой ссылкой на другой.
> >
> > Файл существует в файловой системе до тех пор, пока не удалена последняя жесткая ссылка, в то время как при символическая ссылка после удаления просто будет указывать "в никуда".

7. Операционная система Unix, интерпретаторы, стандартные потоки ввода вывода, фильтры.

Интерпретаторы

Интерпретатор командной строки, или оболочка (shell) - это программа, предоставляющая пользователю интерфейс для общения с командной строкой. Оболочка предоставляет средства для выполнения команд, управления переменными окружения, перенаправления потоков ввода-вывода и анализа кодов возврата. Большинство оболочек предоставляют простой язык программирования для создания пользовательских скриптов. *Примеры оболочек*: bash (bourne again shell), zsh (Z Shell), sh, ksh (Korn Shell), csh (C Shell) и другие.

> Рассмотрим, например, оболочку bash, которая широко используется в Unix-подобных системах. При запуске пользовательской программы, bash настраивает связи между потоками ввода-вывода программы и терминалом, с которого была запущена оболочка. Если указаны специальные символы для перенаправления ввода-вывода, bash заменяет стандартные потоки ввода-вывода на указанные файлы или связывает команды друг с другом. Это часто используется для создания "конвейеров" команд, позволяющих обрабатывать потоки данных между несколькими программами.

Стандартные потоки ввода-вывода

Потоки ввода-вывода - это файлы, предназначенные для взаимодействия и обмена с пользователем информации.

> Существуют `три стандартных потока`: `stdin` (ввод), `stdout` (вывод), и `stderr` (вывод ошибок). В контексте командной оболочки, `stdin` обычно соответствует `вводу` с клавиатуры, `stdout` и `stderr` - выводу на экран, за исключением того, что на `stderr` выводится вся `служебная информация`, которая не должна попадать в стандартный поток вывода. Потоки могут быть перенаправлены в файлы или другие программы. Стандартные потоки привязаны к файловым дескрипторам с номерами `0, 1, 2` для `stdin, stdout, stderr` соответственно.
>
> Примеры стандартных перенаправлений:
>
> > **> file** - перенаправляет `stdout` в **file** (с перезаписью содержимого *file*) **>> file** - добавляет `stdout` в **file 2> file** - перенаправляет `stderr` в **file 2>> file** - добавляет `stderr` в **file < file** - использует **file** в качестве **stdin << EOF** - записывает в `stdin` из терминала до символов **"EOF" 2>&1** - объединяет `stderr` и `stdout`
>
> **/dev/null** - специальный файл, используемый для отбрасывания всех данных, записываемых в него, и возвращающий конец файла при попытке чтения.
>
> **Конвейер (pipe)** - это механизм в Unix, который позволяет вывод одной команды использовать в качестве входных данных другой: **command1 | command2**.

Фильтры

`Фильтры` - это различные пользовательские утилиты, поставляющиеся с Unix, осуществляющие *фильтрацию* данных, поступающим им на вход.

> Фильтры обычно используются в конвейерных конструкциях (**pipelines**) в командной строке для обработки данных "*на лету*". Это позволяет соединить несколько команд в цепочку, где вывод одной команды становится вводом для следующей.
>
> Частоиспользуемые фильтры: `grep`, `sort grep` - осуществляет выборку строк, совпадающих с шаблоном `sort` - сортировка строк по заданному условию `cut` - удаляет определенные части каждой строки ввода.

8. Операционная система Unix. Основные команды, права файлов, способы задания прав.

Основные команды

- `touch файл`: Создает пустой файл, а если он уже есть – обновляет время последней модификации.
- `mkdir каталог`: Создает пустой каталог.
- `rm файл`: Удаляет файл. `-r --recursive` рекурсивно стирает каталоги. Если этого флага нет, файл не может быть каталогом. `-f --force` принудительное удаление.
- `rmdir каталог`: Стирает только пустые каталоги.
- `echo`: Выводит строку текста.
- `cat файл`: Выводит содержимое файла.
- `pwd`: Выводит имя текущего каталога.
- `ls файл`: Выводит список файлов в каталоге.
    - `-l --format=long`: Длинный формат. Выводится с подробной информацией о каждом файле.

    - `-a --all`: Вывод вместе со скрытыми файлами.
    - `-F`: К имени файла добавляется его тип.
    - `-R --recursive`: Рекурсивно выводит подкаталоги.
- `cd каталог`: Переходит в каталог.
- `cp файл1 файл2`: Копирует файл в другой файл.
- `mv файл каталог`: Перемещает файл в каталог.
- `ln файл1 файл2`: Создает новую жесткую ссылку на файл.
    - `-s --symbolic`: Создает символическую ссылку.
- `head/tail файл`: Выводит первые/последние 10 строк файла.
    - `-n`: Первые/последние n строк.
    - `-c`: Первые/последние c байт.
- `wc файл`: Выводит количество строк, слов и байт в файле.
    - `-l --lines`: Только кол-во строк.
    - `-w --words`: Только кол-во слов.
    - `-c --bytes`: Только кол-во байт.
    - `-m --chars`: Кол-во символов.
- `find выражение`: Ищет файлы в иерархии каталогов по заданным параметрам.
- `man команда`: Выводит справку по команде.

Права доступа к файлам

Для каждого файла существуют следующие категории пользователей:

- `u` (user): Владелец файла.
- `g` (group): Члены группы, владеющей файлом.
- `o` (others): Все остальные.

Каждая из этих категорий может иметь любую комбинацию из следующих прав:

- `r` (read): Право на чтение файла/просмотр каталога.
- `w` (write): Право на запись в файл/добавление или удаление каталога.
- `x` (execute): Право на исполнение файла/поиск и переход в каталог.

Способы задания прав

> 1. `Права` (для *владельца*, *группы* и *остальных*) обозначаются в виде **трех восьмеричных цифр**. Команда: `chmod {число} <имя файла>` Пример: `chmod 644 <имя файла>`: установит права **rw r r** для *user, group, others* соответственно.

> 2. **Добавить**, **удалить** или **установить** выбранную комбинацию прав для выбранной комбинации категорий. Команда: `chmod [ugoa]{+-=}[rwx] <имя файла>` (**a = all**) Пример: `chmod u+x,o-x myfile`: добавит право на выполнение (`x`) к правам user (`u`) и уберёт у others (`o`) для файла `myfile`.

> 3. **Скопировать** права одной категории и присвоить их другой. Команда: `chmod кат1 = кат2 <имя файла>` Пример: `chmod g=u myfile`: скопировать права user (`u`) и присвоить их group (`g`) для файла `myfile`.

9. Состав и структура БЭВМ. Адресные пространства БЭВМ, Система команд БЭВМ, форматы команд, машинные циклы.

Состав и структура БЭВМ

БЭВМ включает в себя несколько функциональных блоков и регистров:

- **Память** - состоит из 2048 ячеек. Каждая ячейка занимает 16 разрядов. Для обращения к памяти существует два регистра: 11-разрядный регистр адреса (AR - Address Register), в который нужно поместить адрес, прежде чем обратиться к памяти; 16-разрядный регистр данных (DR - Data Register), который предназначен для чтения или записи данных в/из ячеек памяти. Чтение данных и запись данных происходит по шинам, которые подключаются к ячейке памяти.
- **11-разрядный счетчик команд (IP - Instruction Pointer)**. Хранит в себе адрес следующей исполняемой команды.
- **Арифметико-логическое устройство или АЛУ (ALU - Arithmetic-n-Logic Unit)** может выполнять несколько операций: сложение, логическое умножение, инверсия и прибавление единицы. При операций «сложение» возможен выход за пределы разрядной сетки и формирование битов переполнения и переноса. Выход из АЛУ через коммутатор подключается к шине, по которой информация может быть передана в любой другой регистр БЭВМ.
- **Буферный регистр (BR - Buffer Register)** это 16-разрядный регистр, который используется для организации промежуточного хранения данных во время работы.
- **Регистр команд (CR - Command Register)** - используется для хранения кода команды и декодирования операций, происходящих во время работы.
- **Аккумулятор (AC - ACcumulator)**. БЭВМ относится к ЭВМ, которые называются ЭВМ аккумуляторного типа, где все вычисления с данными производятся через этот регистр.
- **Указатель стека (SP - Stack Pointer)**, как и IP и AR 11-ти разрядный, и всегда указывает на вершину стека - особого участка памяти, который предназначен для хранения адресов возвратов и параметров подпрограмм и прерываний.
- **16-разрядный клавишный регистр (IR - Input Register)** - находится в составе пульта оператора ЭВМ и предназначен для ввода адреса программы, кодов программы и данных, запуска программы на выполнение и управления режимами работы БЭВМ.
- **16-ти разрядный регистр состояния (PS - Program State)** хранит биты управляющие работой БЭВМ (работа, прерывание и пр.) и признаки результата.