

# Краткое содержание сегодняшнего материала

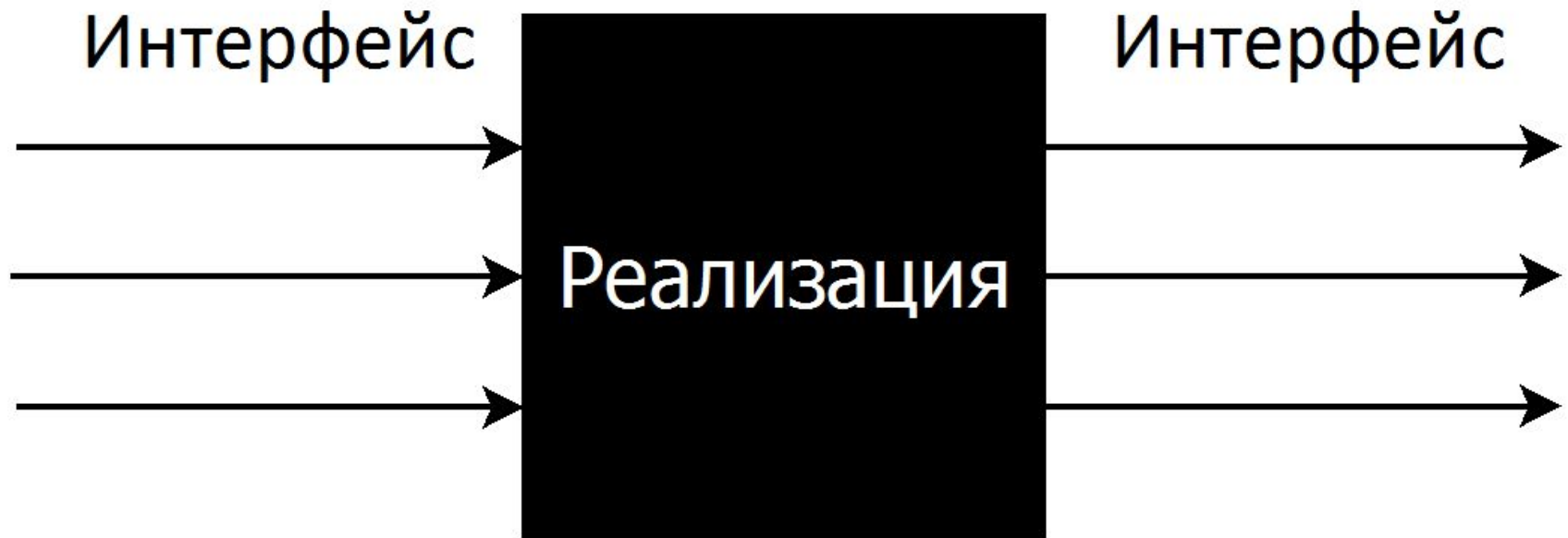
- Принцип черного ящика
- Организация памяти компьютера
- Как компьютер выполняет программу?
- Использование среды Keil

# Принцип черного ящика



Чтобы пользоваться черным ящиком, не обязательно знать его внутреннее устройство

# Инкапсуляция



Чтобы пользоваться классом или библиотекой через интерфейс,  
необязательно понимать, как они реализованы

# Магия

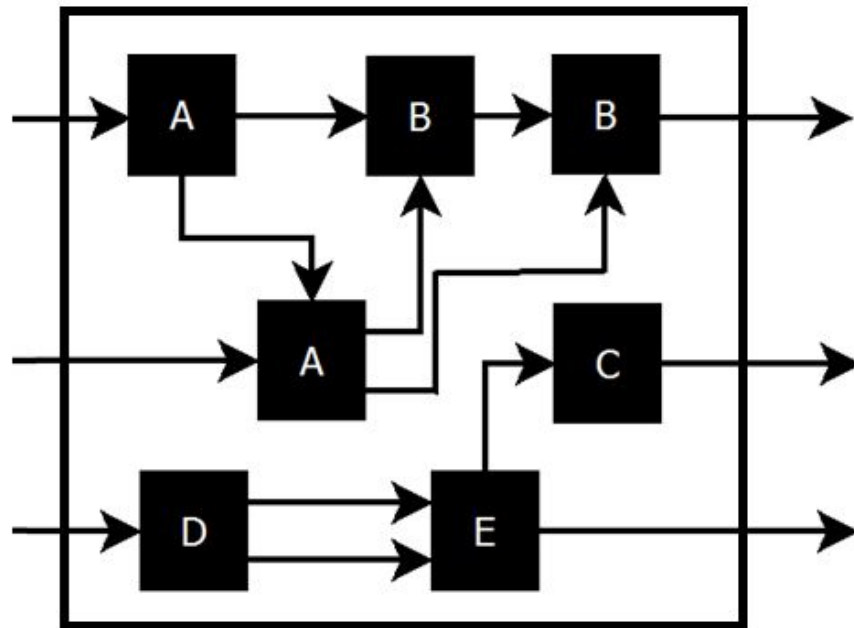


Как работает заклинание или ритуал? Просто работает! Причины не важны!

# А что у черного ящика внутри?



Много черных



# «Закон дырявых абстракций»

**Все нетривиальные абстракции  
дырявы**

**(Дж. Спольски)**

Следствие 1: единственный компетентный способ залатать эти дыры - выучить, как работают абстракции, и какие подробности они скрывают.

Следствие 2: абстракции экономят наше *рабочее* время, но не экономят *учебное* время.

# Абстракции имеют ограниченную область применимости

Сложение скоростей  
в классической  
механике:

$$\vec{v}_a = \vec{v}_r + \vec{v}_e$$

Сложение скоростей  
в релятивистской  
механике:

$$v_{rel} = \frac{v_1 + v_2}{1 + \frac{v_1 v_2}{c^2}}$$

# Как устроена память?

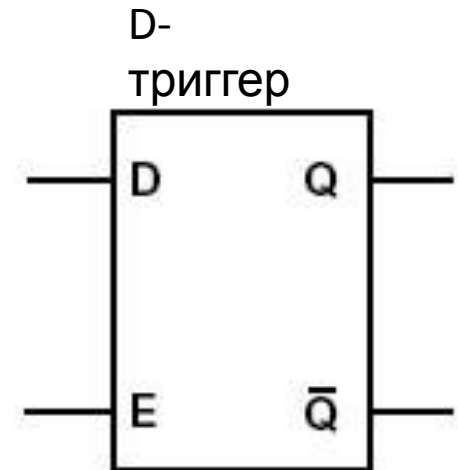
Позволяет хранить 1 бит информации:

D – вход данных;

E – включение: триггер запоминает состояние входа D при логической единице на входе E

Q – прямой выход (совпадает с хранимым состоянием)

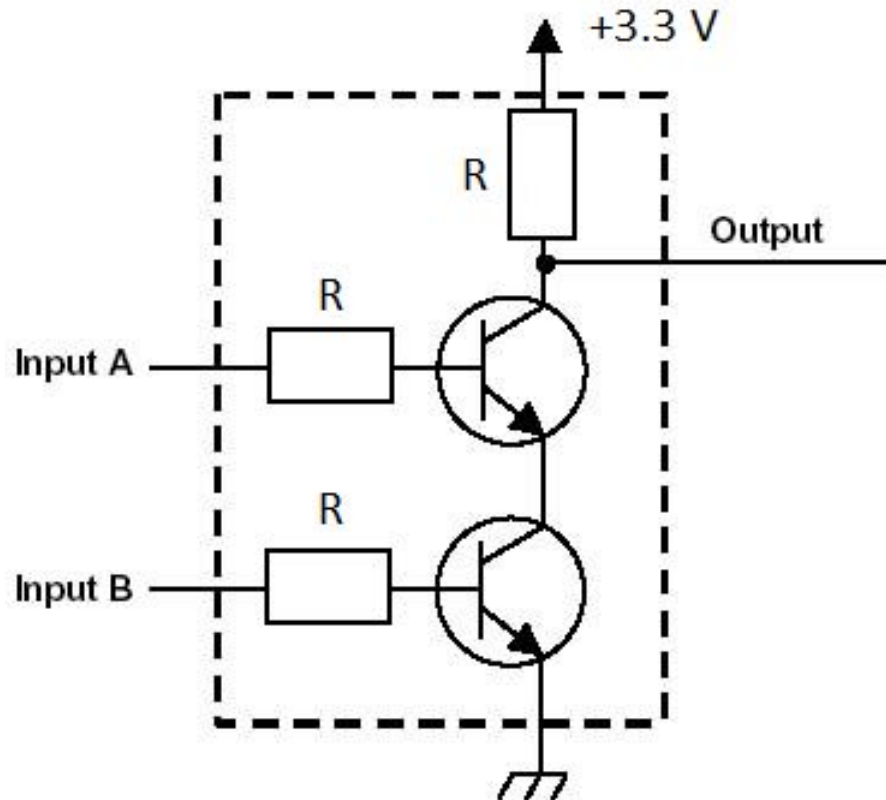
$\bar{Q}$  – инверсный выход (инверсия хранимого состояния)







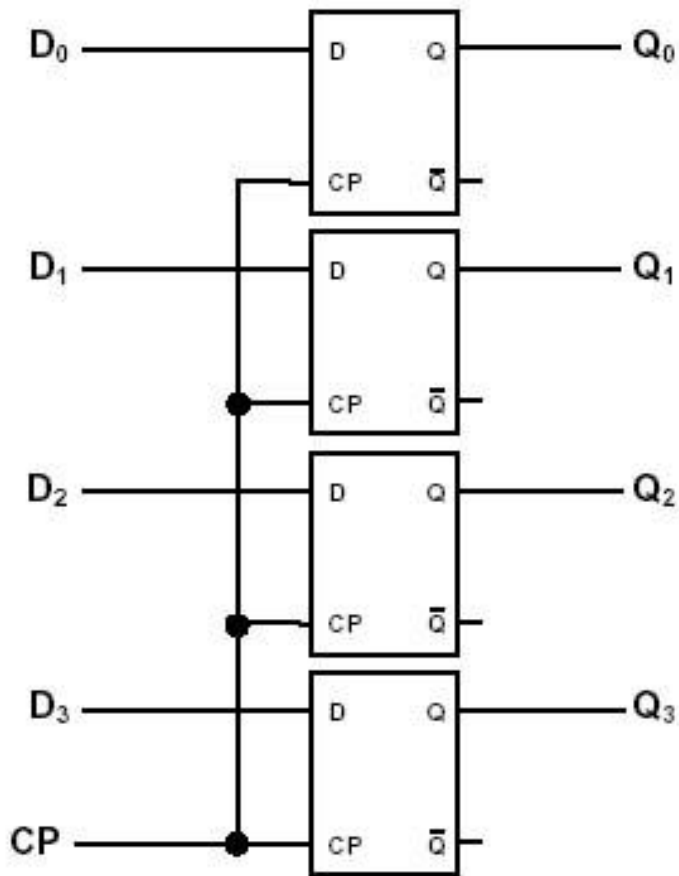
# А как устроен элемент НЕ-И?



# А как устроен транзистор?



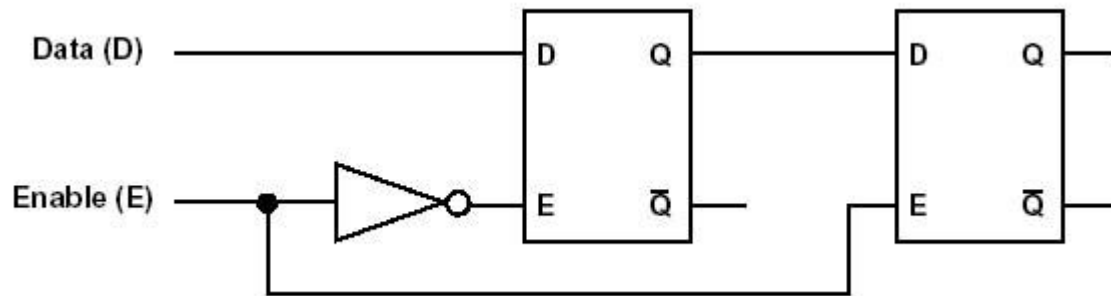
# Как хранить несколько бит?



Емкость	Кол-во проводов
1 бит	2 входа данных 4 выхода данных 1 вход E (всего 9)
8 бит	17
N бит	$2 * N + 1$
4 гигабайта	4 миллиарда?!

# Как хранить много-много бит?

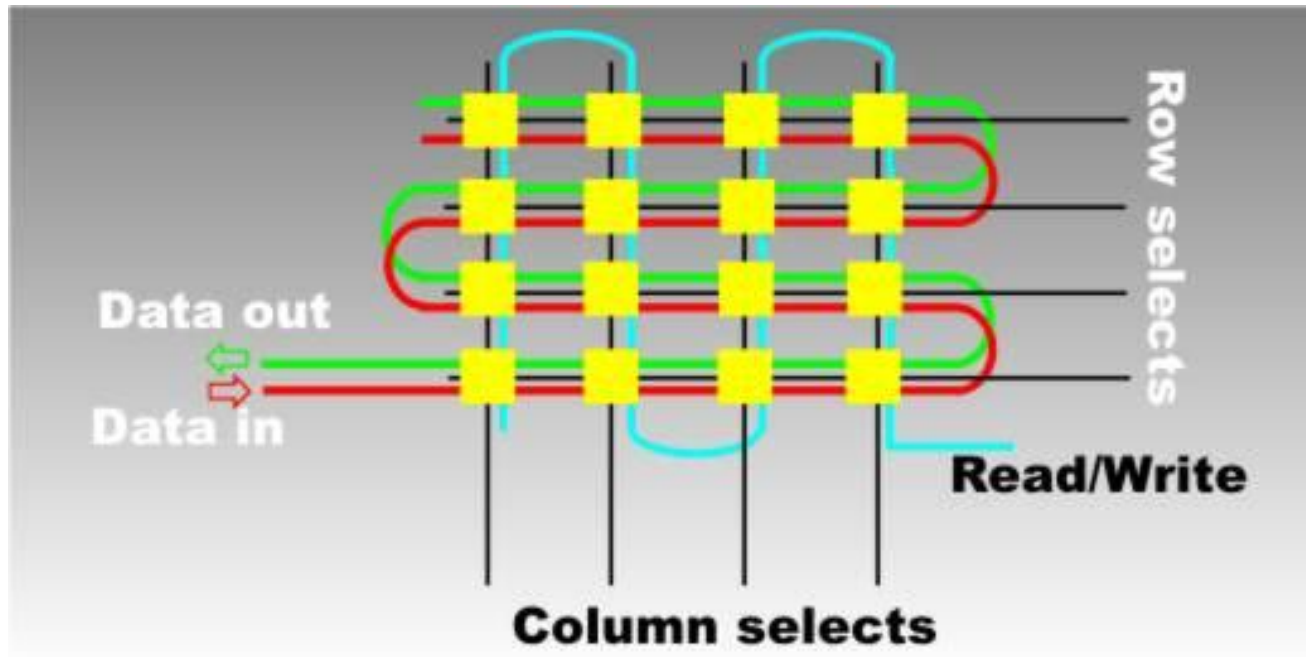
Последовательно: - N бит – 3 провода (но N тактов)



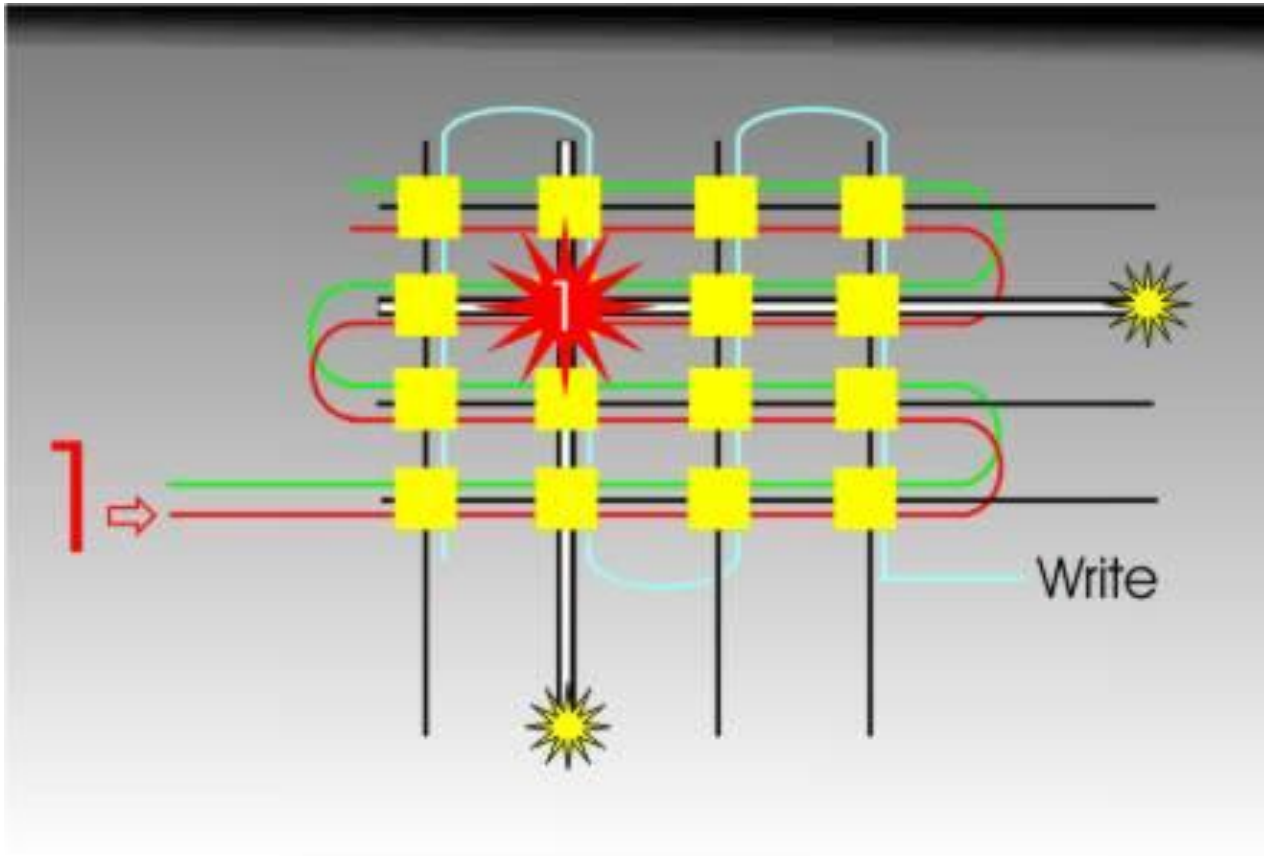
Обычно все же байтами или группами байт. Такой подход используется в стримерах, винчестерах, дискетах, CD/DVD/BD и так далее

# Как хранить много-много бит?

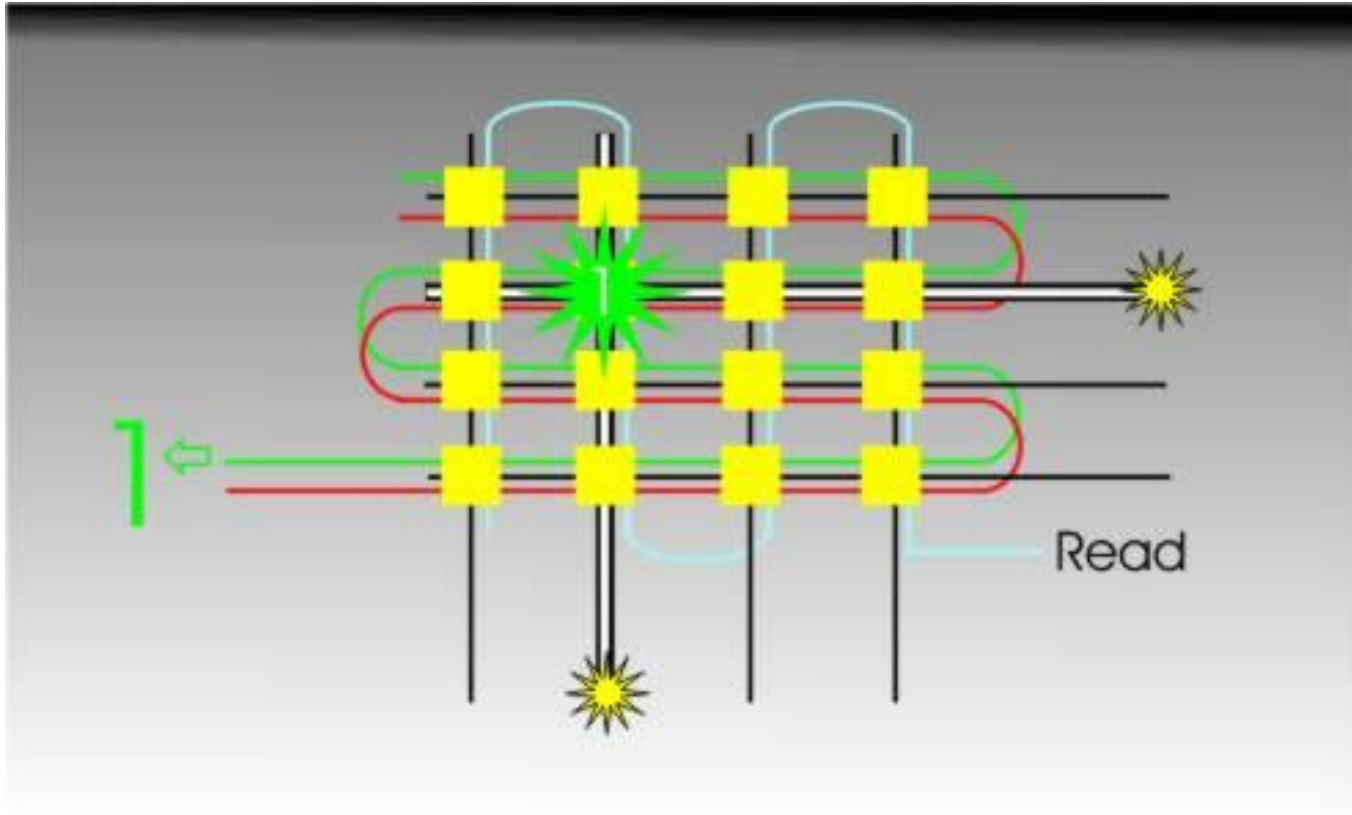
В виде матрицы:  $N$  бит –  $3 + \sqrt{N}$   
проводов



# Запись



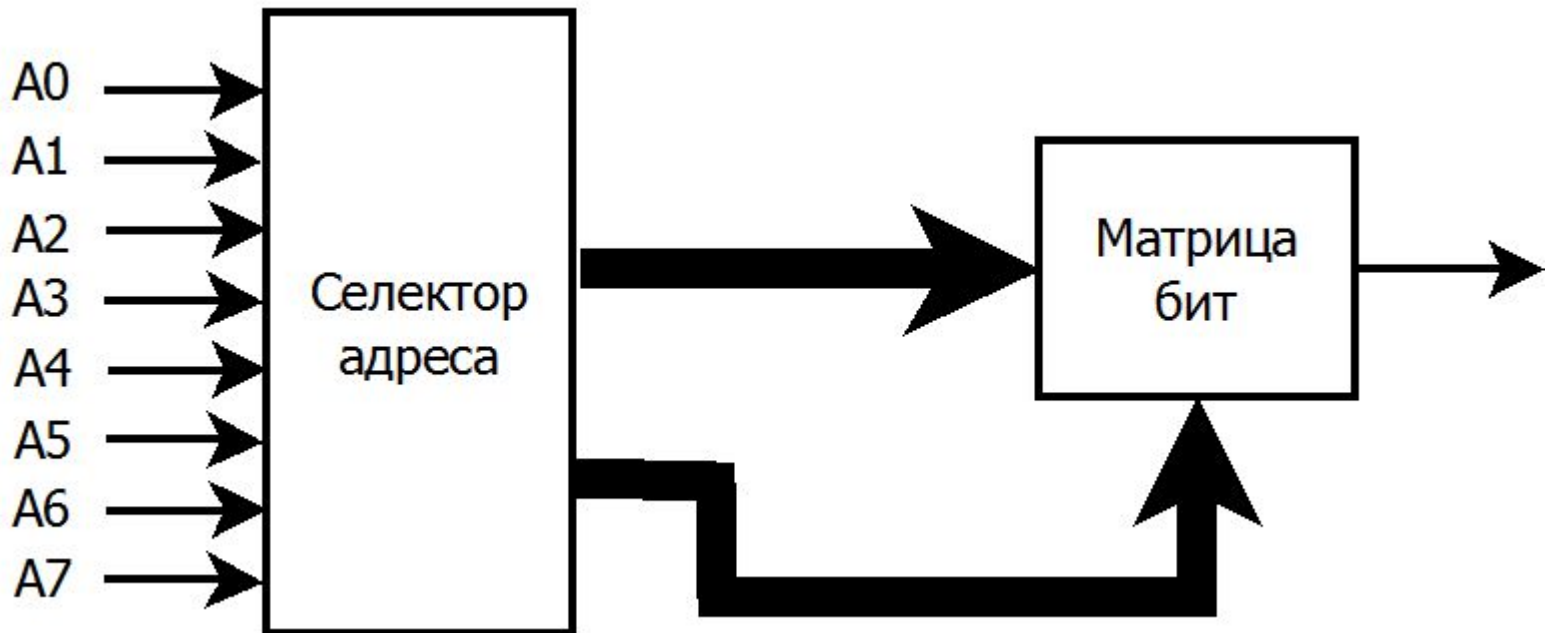
# Чтение





# Как хранить много-много бит?

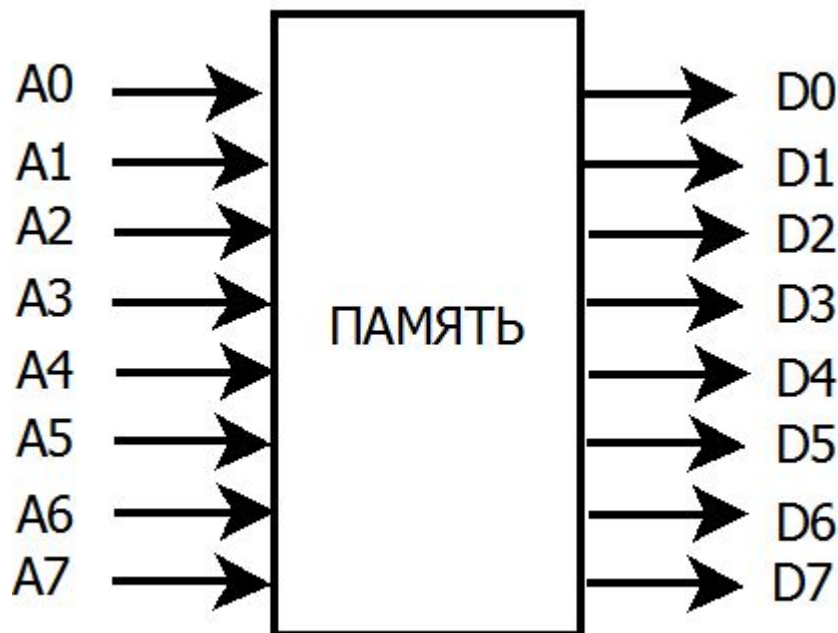
Использовать АДРЕСА: - N бит –  $\sim \log_2 N$



В реальности минимальная адресуемая единица (МАЕ) – не бит, а байт или более.

# Как хранить много байт?

$8 \cdot N$  бит данных ( $N$  байт) –  $\log_2 N$  проводов для шины адреса +  $N$  для шины данных



# Память с адресной структурой

- Начальный адрес – обязательно 0
- «Памятей» может быть несколько, причем адреса могут численно совпадать
- Может хранить разные по смыслу данные

# А как хранятся многобайтные объекты?

Объекты в смысле «штуковины», а не экземпляр класса C++

Как хранится массив char  
a[5]?

Поряд  
!

А ближе к началу адресов  
элемент a[0] или элемент  
a[4]?

a[0]

Допустим, int занимает 4  
байта.

Поряд  
!

Как они хранятся?

А в каком  
порядке?

Возможны  
варианты

# Endianness (порядок байт)

Little-endian

Big-endian

В меньшем адресе младший байт

В меньшем адресе старший байт

`Int x = 0x11223344;`

Значение	Адрес	Значение	Адрес
0x44	0	0x11	0
0x33	1	0x22	1
0x22	2	0x33	2
0x11	3	0x44	3

Еще бывает смешанный, но давайте лучше не будем про него вспоминать

# А как хранится код?

- Код на языке высокого уровня компилируется в ассемблерные инструкции – машинные команды
- Эти команды и выполняет процессор
- Каждая инструкция – это число
- Числа хранятся в памяти!
- Код – это данные!

# Архитектуры ЭВМ

Признак	Гарвардская	фон Неймана	ARMv7
Единое адресное пространство для кода и данных	иногда	всегда	да
Код может выполняться из памяти	иногда	всегда	да
данных могут храниться в памяти кода	иногда	всегда	да
Две шины (для данных и для кода)	всегда	иногда	да

ARM официально считает ARMv7 «модифицированной гарвардской архитектурой»

# Какие бывают инструкции ассемблера?

- Арифметика (сложить два числа, вычесть, умножить, разделить и т.д.).
- Битовые операции (сдвиг, ИЛИ, И, НЕ и т.д.).
- Перемещения между ячейками памяти (из переменной в переменную и т.д.).
- Условные и безусловные переходы (if-else и goto).
- Прочие



# Что такое процессорные регистры?

- «Сверхоперативная память»
- В них сохраняются результаты промежуточных вычислений
- В RISC процессор не имеет прямого доступа к оперативной памяти
  - Есть спец-команды «загрузить значение из памяти в регистр» и
  - «выгрузить значение из регистра в память»

# Приблизительное превращение из C в ассемблер

```
int x = 5 + 7;
```

- 1) загрузить в регистр 0  
число 5
- 2) загрузить в регистр 1  
число 7
- 3) сложить числа в регистрах 0 и  
1;  
результат поместить в регистр  
2
- 4) выгрузить число из регистра  
2  
в память по адресу &x