Введение в Архитектуру ЭВМ

Для вопросов по курсу:
natalya.razmochaeva@moevm.info

Берленко Татьяна Андреевна
Шевская Наталья Владимировна
СПбГЭТУ “ЛЭТИ”, ФКТИ, МОЭВМ

Позиционные системы счисления

- Десятичная система счисления

$$56789_{10} = 5 * 10^4 + 6 * 10^3 + 7 * 10^2 + 8 * 10^1 + 9 * 10^0$$

- Двоичная система счисления

$$10011_2 = 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$

Позиционные системы счисления

- Восьмеричная система счисления

$$567_8 = 5 * 8^2 + 6 * 8^1 + 7 * 8^0$$

- Шестнадцатеричная система счисления

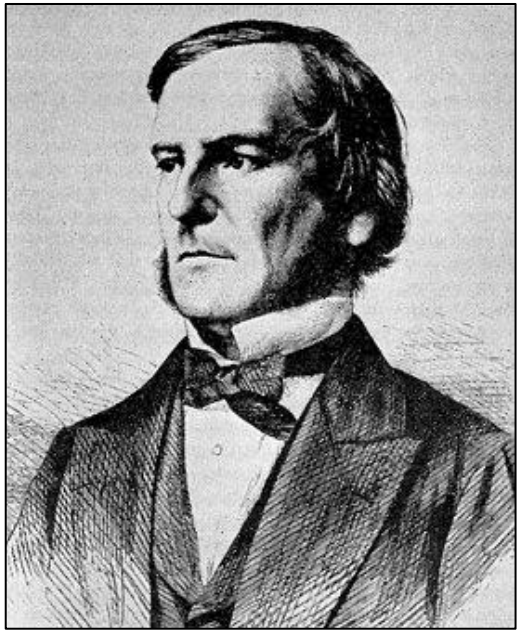
$$56A8C_{16} = 5 * 16^4 + 6 * 16^3 + A * 16^2 + 8 * 16^1 + C * 16^0$$

Булева Алгебра

a	¬a
0	1
1	0

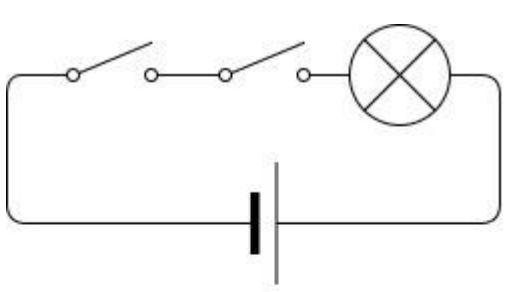
a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

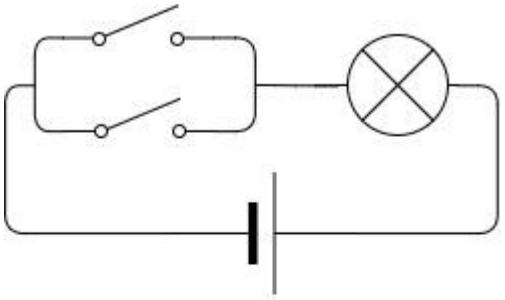
a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

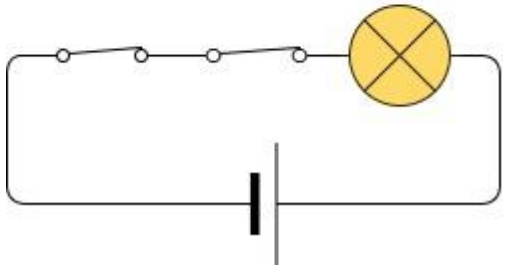


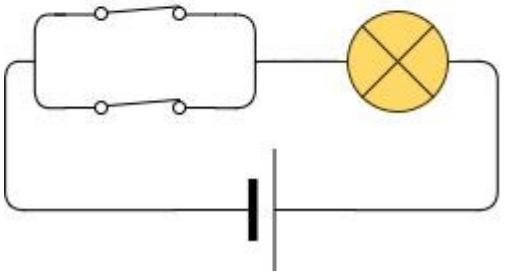
Джордж Буль
(2 ноября 1815 -- 8 декабря 1864)

Применение. Схемы И и ИЛИ



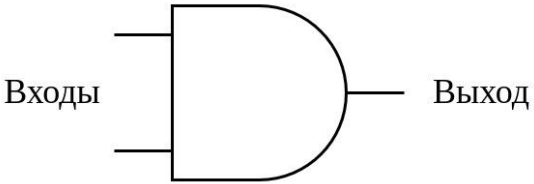




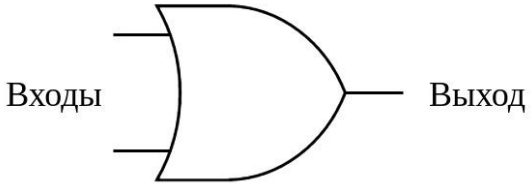


Применение. Вентили и инвертор

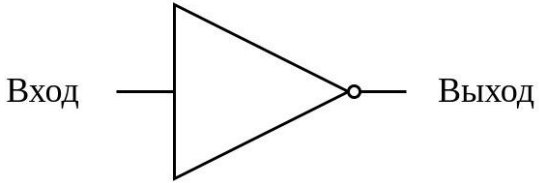
И



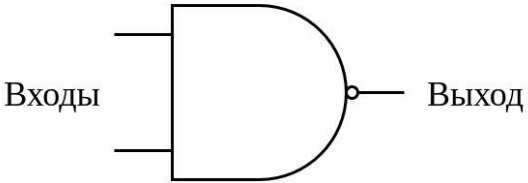
ИЛИ



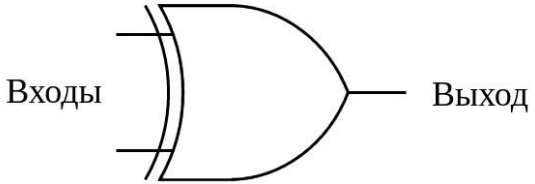
Инвертор



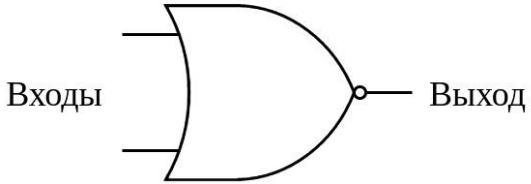
И-НЕ



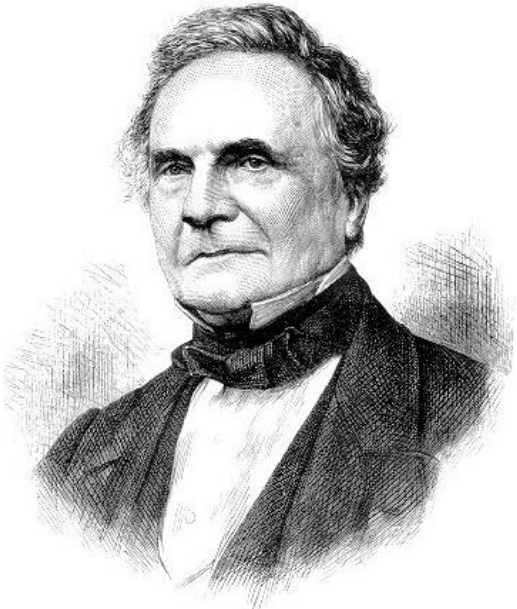
Исключающее ИЛИ



ИЛИ-НЕ



Разностная машина (Difference Engine)



Чарлз Бэббидж (26 декабря 1791 - 18 октября 1871) английский математик, изобретатель

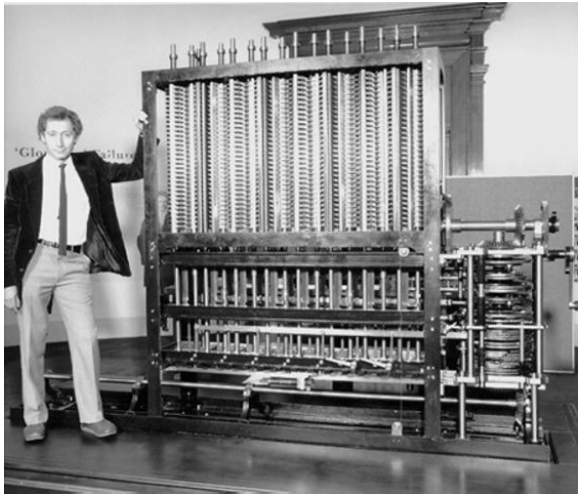
Первое изобретение (незавершенное): Разностная машина (1820 - 1833 гг.)

Конструкция разностной машины основывалась на использовании десятичной системы счисления.

Аналитическая машина

Конструкция:

- хранилище (память)
- «мельница» (арифметическое устройство)
- устройство ввода-вывода



Ада Лавлейс (10 декабря 1815 – 27ноября 1852) создала первую в мире программу для аналитической машины Беббиджа

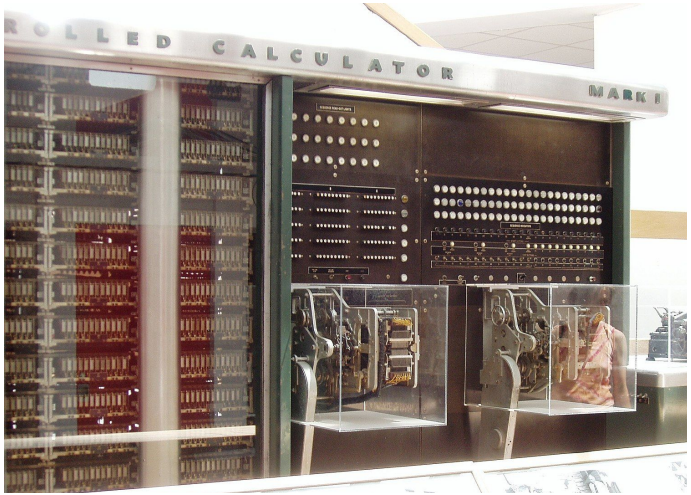
Реле

Марк I

1943 г., первый цифровой компьютер:

«Automated Sequence Controlled Calculator», позже получивший имя «Марк I».

Компьютер оперировал 72 числами, состоящими из 23 десятичных разрядов, делая по 3 операции сложения или вычитания в секунду. Умножение выполнялось в течение 6 секунд, деление — 15,3 секунды.



ENIAC

ЭНИАК (1945) - первый **электронный** цифровой вычислитель.

Построен на 18000 радиолампах.

Самый большой компьютер в истории (30т).

Домашние компьютеры обогнали его по быстродействию в 1977 г.

Вычисления производились в десятичной с.с.

EDVAC. Архитектура фон Неймана

Использовалась двоичная с.с., использовались условные переходы.

Время операции сложения - 864 микросекунды, умножения - 2900 микросекунд (2,9 миллисекунды).

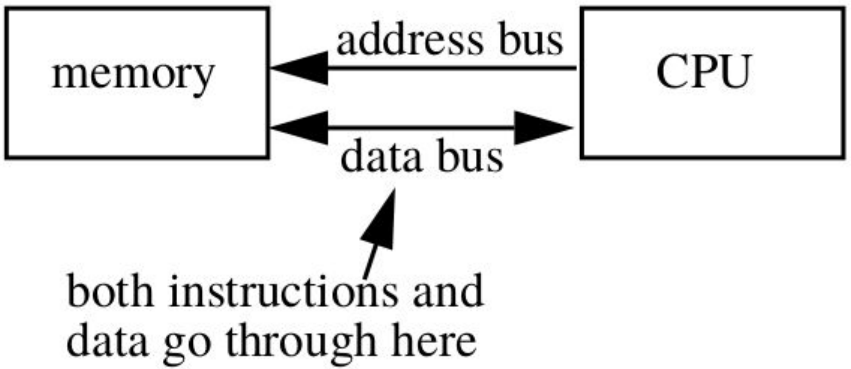
Первый компьютер на базе архитектуры фон Неймана.

Архитектура Фон-Неймана

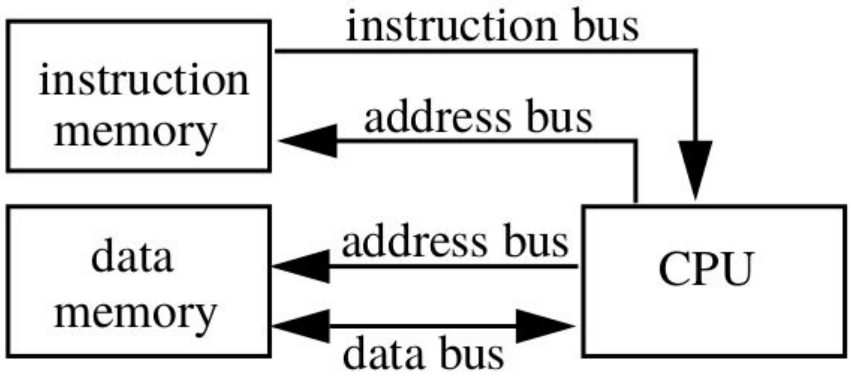
1. Адресность
2. Однородность памяти
3. Программное управление

Типы архитектур

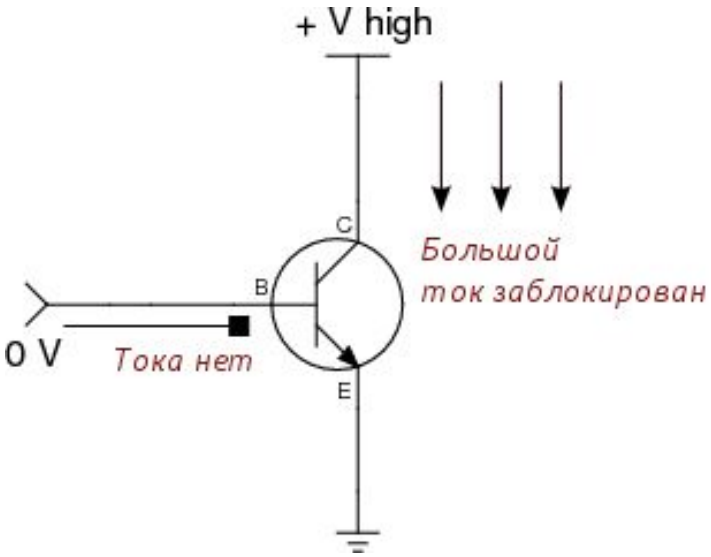
Von Neumann Architecture

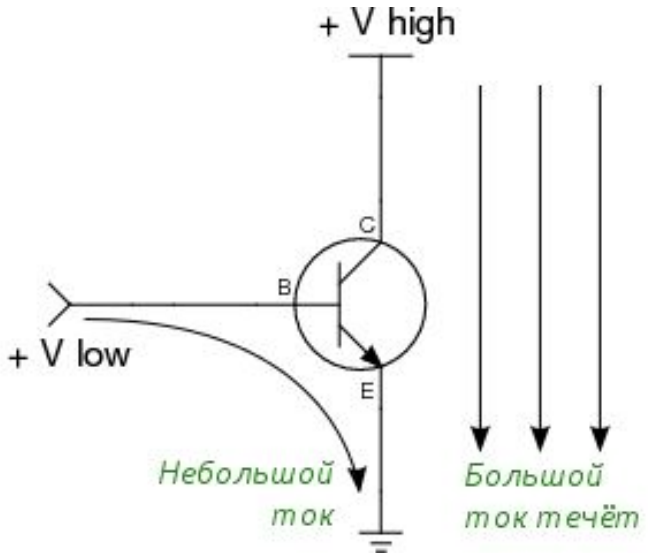


Harvard Architecture



Транзистор

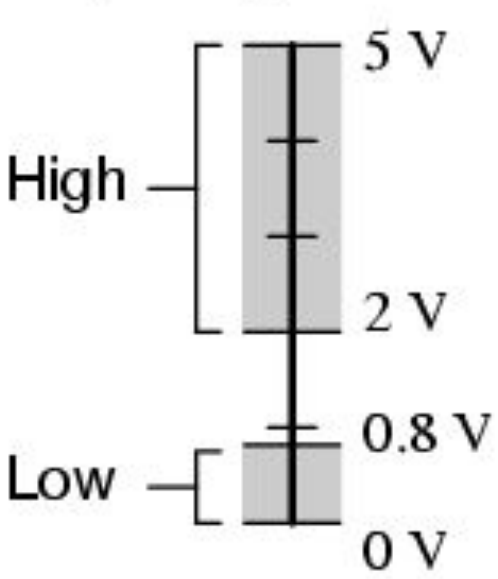

+ V high
C
B
E
0 V
Тока нет
Большой
ток заблокирован


+ V high
C
B
E
+ V low
Небольшой
ток
Большой
ток течёт

Связь цифрового и аналогового мира

Для микросхемы ТТЛ (транзисторно-транзисторной логики):

- 0 соответствует Low
- 1 соответствует High



Сумматор

Сумматор. Начало

- Используем переключатели, лампочки и логические вентили.
- Практически все операции в выч. машине представимы через сумму.
- Для двоичных чисел принцип тот же, что и для десятичных -- сложение в столбик
- Как представить “один в уме” в логической схеме?
- Сперва:

сумма	**0**	**1**
0	0	1
1	1	10

Допишем незначащие нули, что результат был двухбитовым

⇒

** занимал одинаковое кол-во ячеек*

сумма	**0**	**1**
0	00	01
1	01	10

Сумматор: сумма и перенос

- вынесем отдельно бит, который “в уме” (перенос разряда)

сумма	0	1
0	00	01
1	01	10

⇒

перенос	0	1
0	0	0
1	0	1

∪

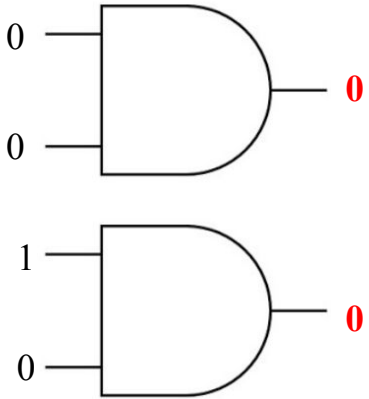
“сумма”	0	1
0	0	1
1	1	0

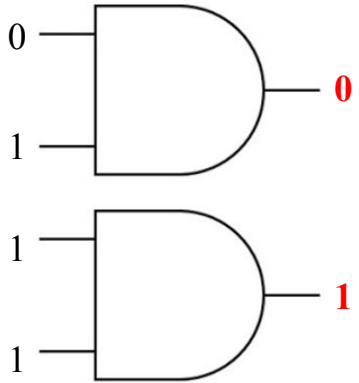
Сумматор ⇒ И | ∪ Искл. ИЛИ XOR

Сумматор. Вентиль “И” для переноса

- нарисуем работу вентиля по таблице истинности ⇒ нарисуем, как работает “1 в уме” (перенос разряда) в логической схеме

перенос	0	1
0	0	0
1	0	1

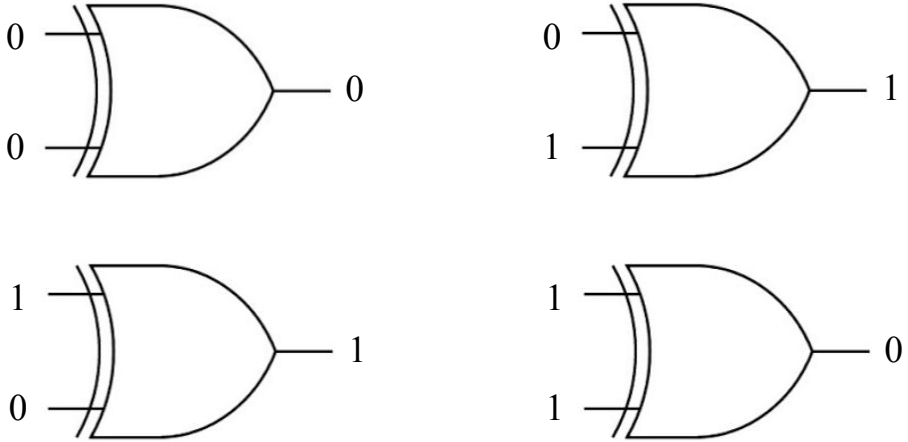




Сумматор. Искл. ИЛИ для суммы

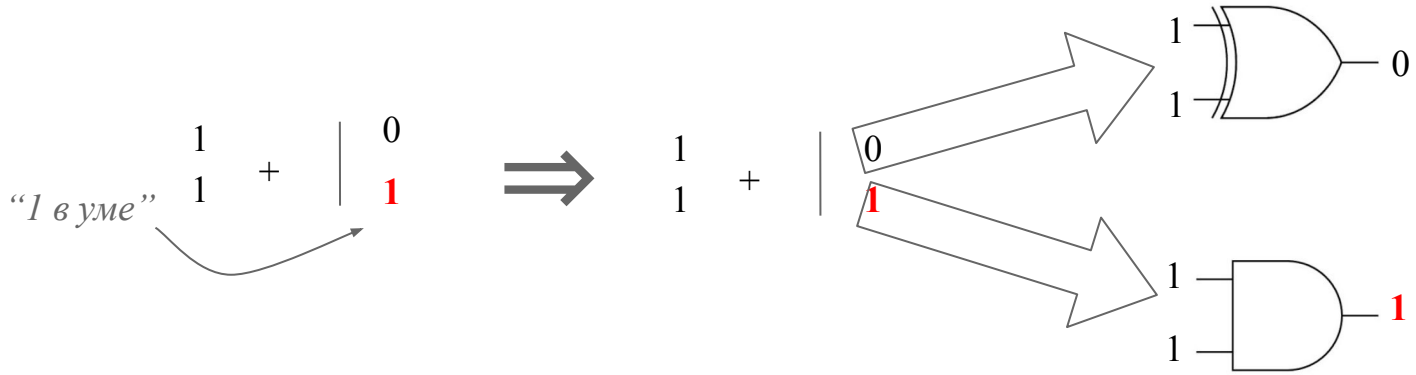
- нарисуем работу вентиля по таблице истинности ⇒ нарисуем, как работает “сумма” в логической схеме

“сумма”	**0**	**1**
0	0	1
1	1	0



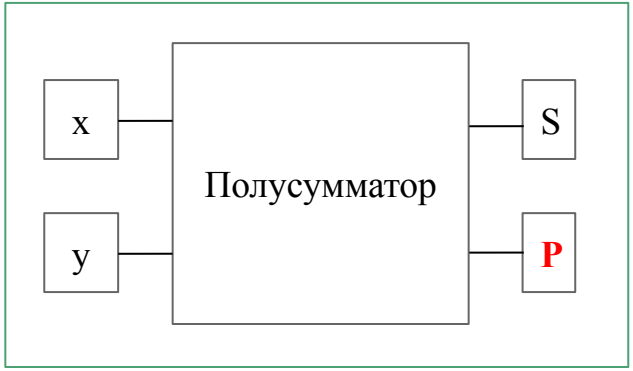
Сумматор. Как соединить вентили в схему?

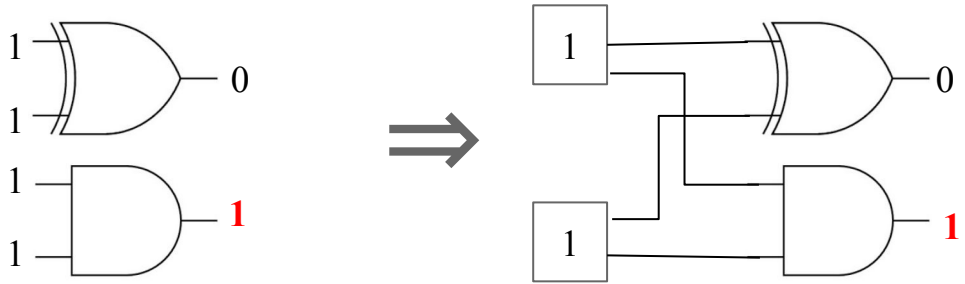
- одноразрядное сложение в двоичной системе
- “положим” сложение в столбик на бок



Полусумматор

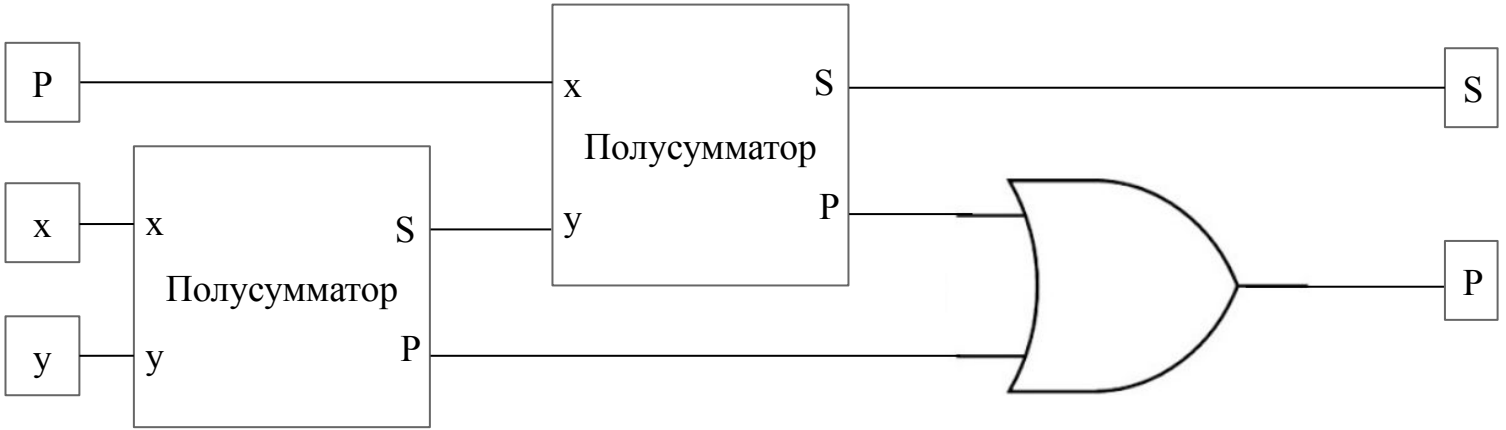
- избавляемся от дублирования входов
- прокладываем провода подлиннее
- введем обозначения:
  - входных сигналов x, y (одноразрядные)
  - выходных сигналов S (сумма), P (перенос)



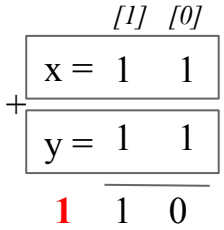


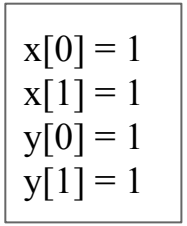
Полный сумматор

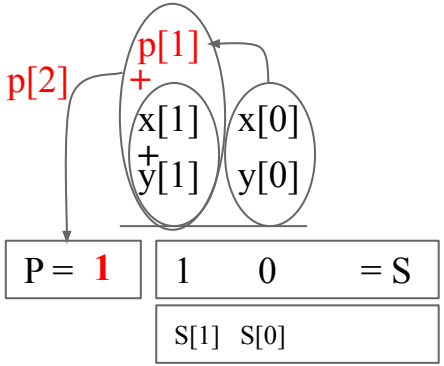
- для трех двоичных цифр (с учетом переноса P, поданного на вход)



Двухразрядное сложение

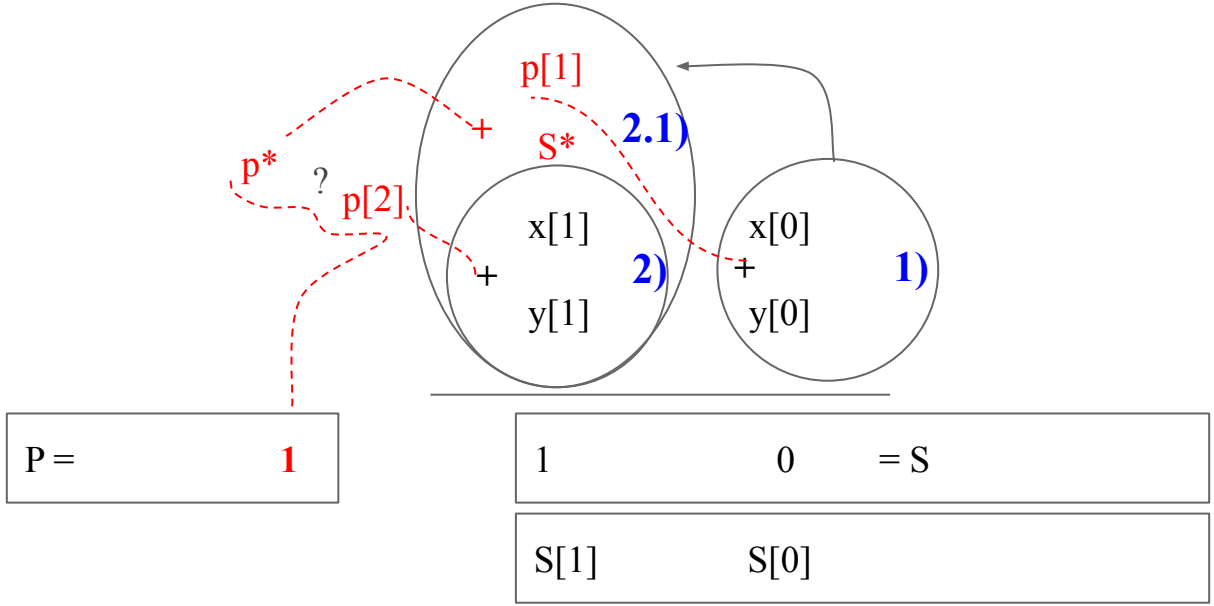






- дважды держали “1 в уме” ⇒ два переноса ⇒ два вентиля И
- два разряда в итоговой “сумме” ⇒ два вентиля Искл. ИЛИ:
  - S[0] ⇒ получился естественным путем, породил первую единицу в уме p[1]
  - S[1] ⇒ получился за счет сложения x[1]+y[1]+p[1], что породило вторую единицу в уме **p[2]**
  - x[1]+y[1] ⇒ известный нам полусумматор

Детализация

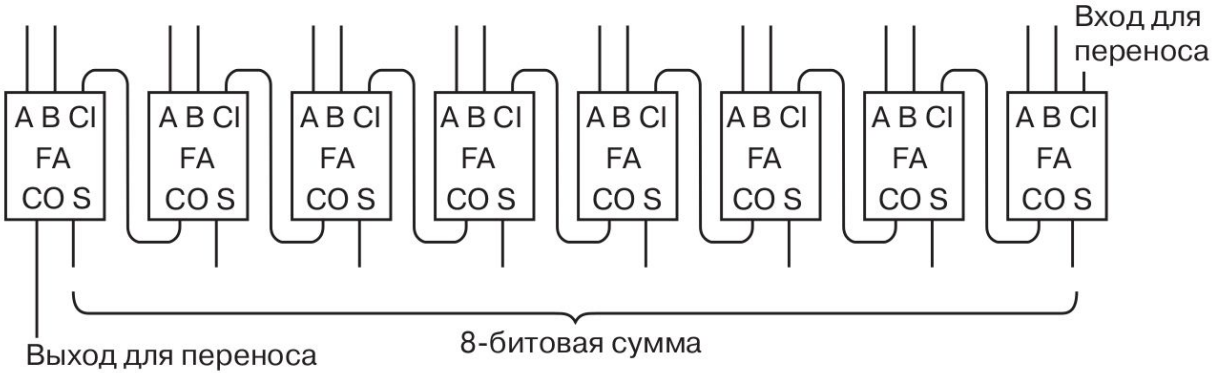


красным -- то, что держим в уме

каждый + в этой схеме генерирует элемент суммы и перенос

1) x[0] + y[0] = 1 0 ⇒ S[0] = 0 и p[1] = 1
2) x[1] + y[1] = 1 0 ⇒ S* = 0 p[2] = 1
2.1) реализуем p[1] : S* + p[1] => S[1] = 1 и p* = 0
остались незадействованные p* и p[2] ⇒ ИЛИ (см. след. слайд)

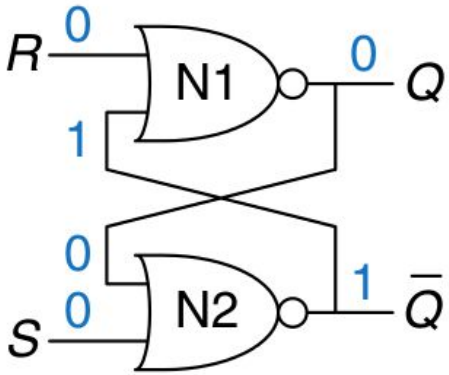
Сумматор. 8-битовая сумма



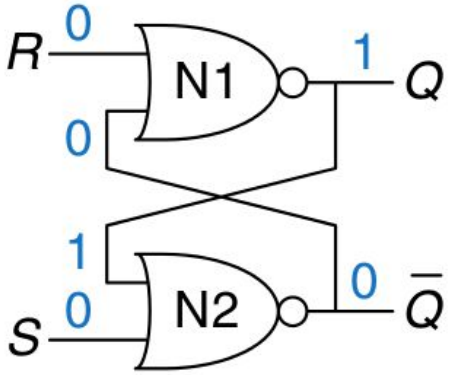
- Первый вход для переноса -- заземление.
- Обозначени:
  - CI / CO - Carry In / Carry Out (вход / выход для переноса)
  - FA -- Full Adder (HF -- Half Adder)
  - S -- Summ

картинка -- стр. 165, Ч. Петцольд “Код”

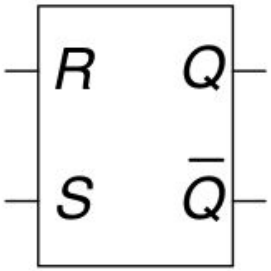
SR(RS)-Защелка



(a)

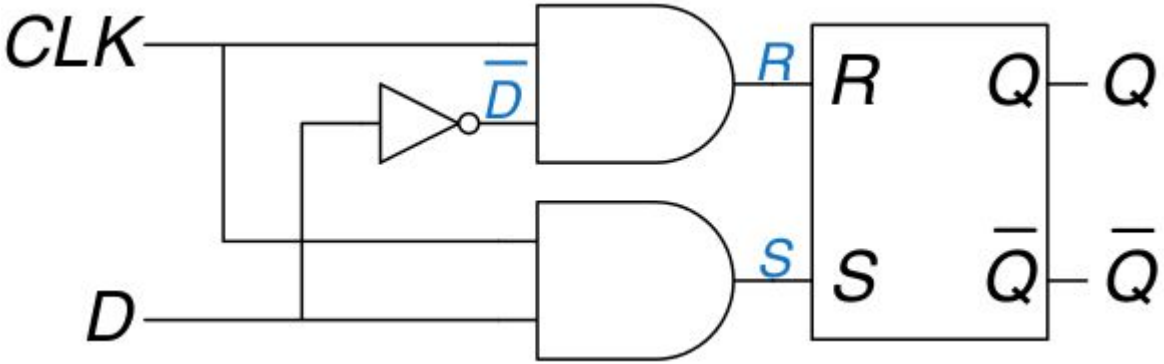


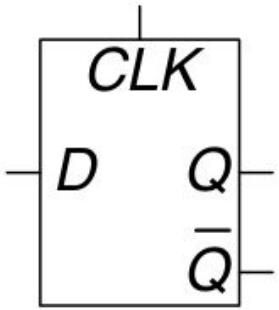
(b)



Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0

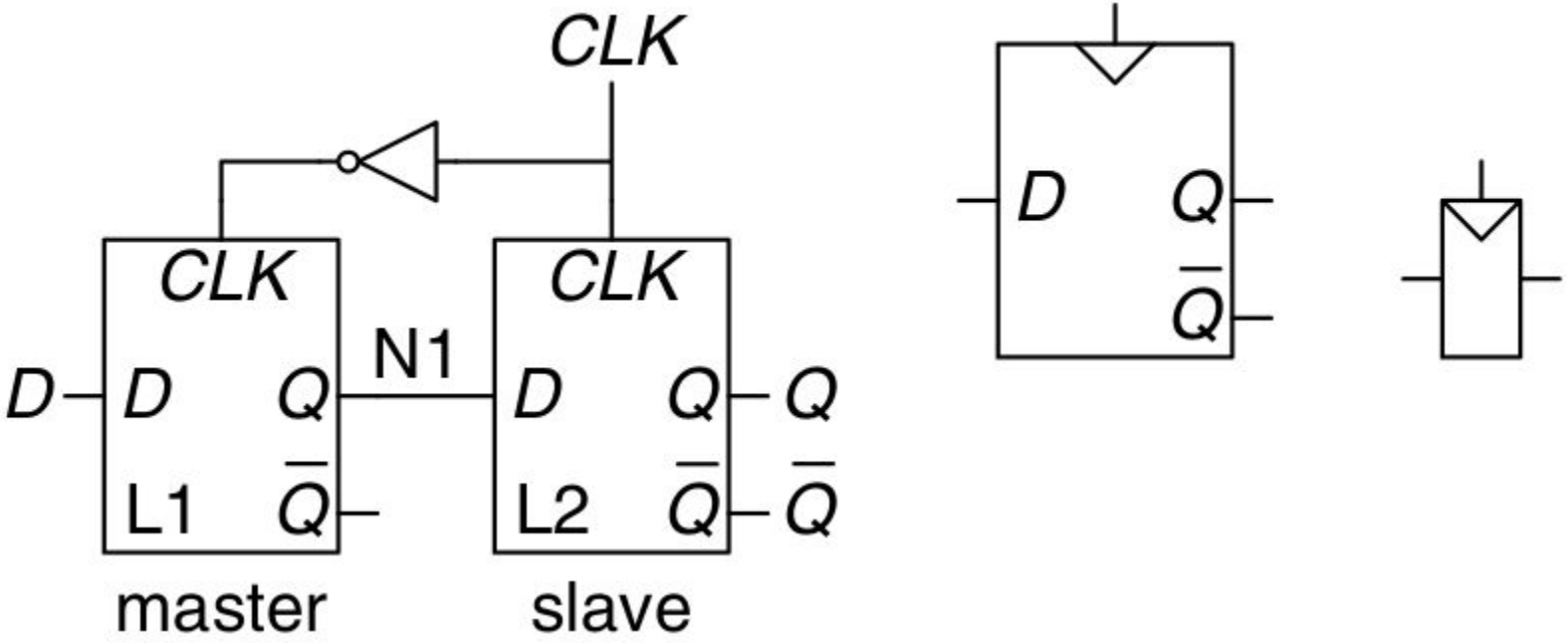
D-Защелка (D-Latch)





CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D-Триггер (D-flip-flop)

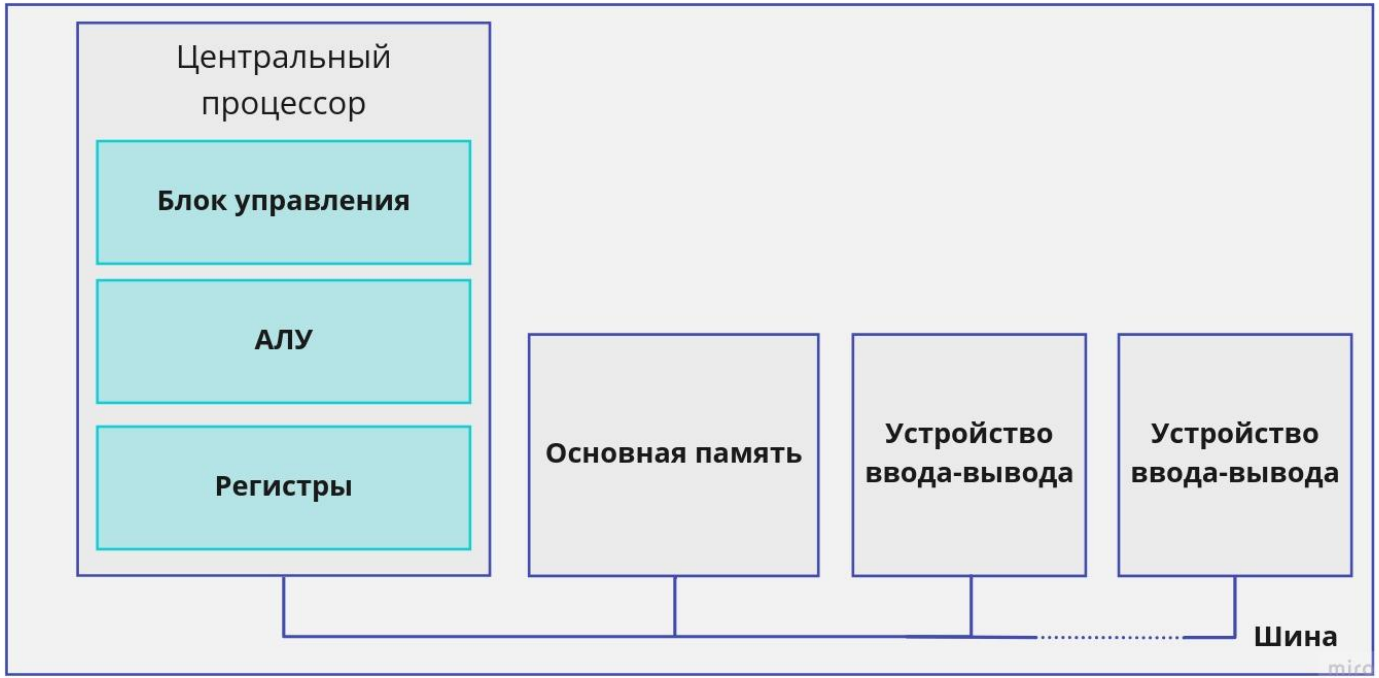


Регистр

Регистр – логическая схема, предназначенная для хранения двоичных чисел заданной разрядности.

Регистр состоит из группы триггеров (например, D-триггеров).

Как устроено простое вычислительное устройство



Формат представления чисел на компьютере

Числа конечной точности - числа, представляемые в фиксированном количестве разрядов.

Арифметические операции с числами конечной точности имеют ограничения и могут вызвать **переполнение**.

Формат представления целых чисел

Беззнаковые:

123:

0	1	1	1	1	0	1	1

Диапазон значений:

0 … 2^n -1, где n - разрядность архитектуры

Формат представления целых чисел

Знаковые:

- Прямой код ($-2^{n-1}+1 \dots 2^{n-1}-1$)

-123:

1	1	1	1	1	0	1	1

- Обратный код ($-2^{n-1}+1 \dots 2^{n-1}-1$)

-123:

1	0	0	0	0	1	0	0

- Дополнительный код ($-2^{n-1} \dots 2^{n-1}-1$)

-123:

1	0	0	0	0	1	0	1

Дополнительный код -- единственный ноль

В 10-СС	В 2-СС	Инвертируем	Выделяем знаковый бит	Дополняем до двух	В 10-СС и доп. кода
0	000	111	1|11	0|00	0
1	001	110	1|10	1|11	-1
2	010	101	1|01	1|10	-2
3	011	100	1|00	1|01	-3
4	100	011	0|11	1|00	-4
5	101	010	0|10	0|11	3
6	110	001	0|01	0|10	2
7	111	000	0|00	0|01	1

Примеры диапазонов

Разрядность	Диапазон беззнаковых чисел	Диапазон знаковых чисел (дополнительный код)
8	От 0 до 255	От –128 до 127
32	От 0 до 4 294 967 295	От –2 147 483 648 до 2 147 483 647
64	От 0 до 18 446 744 073 709 551 615	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

Формат представления чисел с плавающей точкой

Стандарт IEEE 754:

- **одинарная точность** (single precision) - 4 байта.
  Пример: float в C
  примерно от 10^{-38} до 10^{38}

- **двойная точность** (double precision) - 8 байт.
  Примеры: double в C, float в Python
  примерно от 10^{-308} до 10^{308}

Одинарная точность

- 1 бит - знак (0 - положительные числа, 1 - отрицательные)
- 8 бит - порядок
- 23 бита - дробная значащая часть числа - мантисса
- 127 - смещение

$111{,}1101 = 1{,}111101 * 2^2$

1,111101 - мантисса, записывается только дробная часть

2 - *истинный* порядок, 129 - *смещенный* порядок

знак	порядок								мантисса									
0	1	0	0	0	0	0	0	1	1	1	1	1	0	1	...	0	0	0

Одинарная точность

Специальные случаи:

- Если порядок и мантисса равны 0, число равно 0.
- Если порядок равен 255 и мантисса равна 0, число в зависимости от знака -∞ или +∞.
- Если порядок равен 255 и мантисса не равна 0, значение считается недопустимым числом и является NaN (Not a Number).

знак	порядок								мантисса						
0	0	1	0	0	0	0	0	1	1	1	1	...	0	0	0

Двойная точность

- 1 бит - знак (0 - положительные числа, 1 - отрицательные)
- 11 бит - порядок
- 52 бита - дробная значащая часть числа - мантисса
- 1023 - смещение

$111{,}1101 = 1{,}111101 * 2^2$

1,111101 - мантисса

2 - *истинный* порядок, 1025 - *смещенный* порядок

знак	порядок							мантисса						
0	0	1	0	...	0	0	1	1	1	1	...	0	0	0

Сравнение чисел с плавающей точкой

Пример:

```
a = 0.1 + 0.2
if a == 0.3:
    print('Числа равны')
else:
    print('Числа не равны')
```

Что будет выведено на экран?

Формат представления символов на компьютере

➢ ASCII — 7-битовая кодировка, доступно 128 символов.

В Python есть функция ascii(obj), которая возвращает строковое представление объекта с экранированными не-ASCII символами.

```
>>> ascii('привет')
```

➢ Unicode — 16-битовая кодировка, доступно 65 536 символа.

в Python доступны функции:

```
>>> ord('A')
```

```
>>> chr(95)
```

Поразрядные операции

Операнд	Описание
|	Побитовый OR
^	Побитовый XOR
&	Побитовый AND
<<, >>	Смещения
~x	Побитовый NOT

Чем выше приоритет оператора, тем ниже он находится в таблице и тем раньше он выполняется в смешанных выражениях.

Вспомогательные функции Python

```
>>> b = 127

>>> b.bit_length()

....

>>> bin(127)

....

>>> "{:b}".format(127)
```

```
>>> int(0b10010, 2)

....

>>> 0b11101000 + 0b10010

....

>>>  '{:b}'.format(0b11101000 >> 2)

....
```

Источники

- Ч. Петцольд “Код”
- Э. Таненбаум “Архитектура Компьютера”
- https://stepik.org/course/253 Курс на Stepik “Введение в Архитектуру ЭВМ”

Вопросы по курсу можно задавать:

Шевская Наталья Владимировна
natalya.razmochaeva@moevm.info,

Берленко Татьяна Андреевна
tatyana.berlenko@moevm.info