

Компьютер изнутри

1. Основные принципы
2. Персональный компьютер
3. Хранение целых чисел
4. Битовые операции
5. Вещественные числа

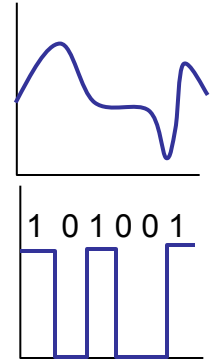
Компьютер изнутри

Тема 1. Основные принципы

Определения

Компьютер (*computer*) – это **программируемое** электронное устройство для обработки данных.

- **аналоговые** компьютеры – складывают и умножают аналоговые (непрерывные) сигналы
- **цифровые** компьютеры – работают с цифровыми (дискретными) данными.



Hardware – аппаратное обеспечение, «железо».

Software – программное обеспечение, «софт».

Программа – это последовательность команд, которые должен выполнить компьютер.

Команда – это описание операции:

- **код операции**
- **операнды** – исходные данные (числа) или их адреса
- **результат** (куда записать).



Процессор – микросхема, которая обрабатывает информацию и управляет всеми устройствами компьютера.

АЛУ – арифметико-логическое устройство

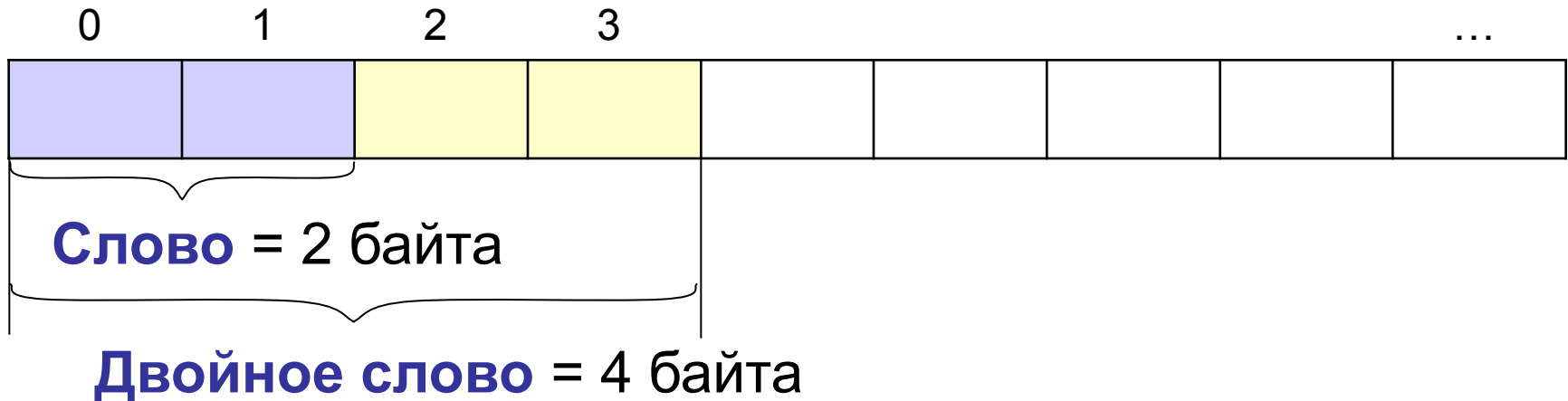
УУ – устройство управления

Регистр – ячейка быстродействующей оперативной памяти, расположенная внутри процессора.

Структура памяти

- Память состоит из **нумерованных ячеек**.
- **Линейная структура** (адрес ячейки – одно число).
- **Байт** – это наименьшая ячейка памяти, имеющая собственный адрес (4, 6, 7, 8, 12 бит).

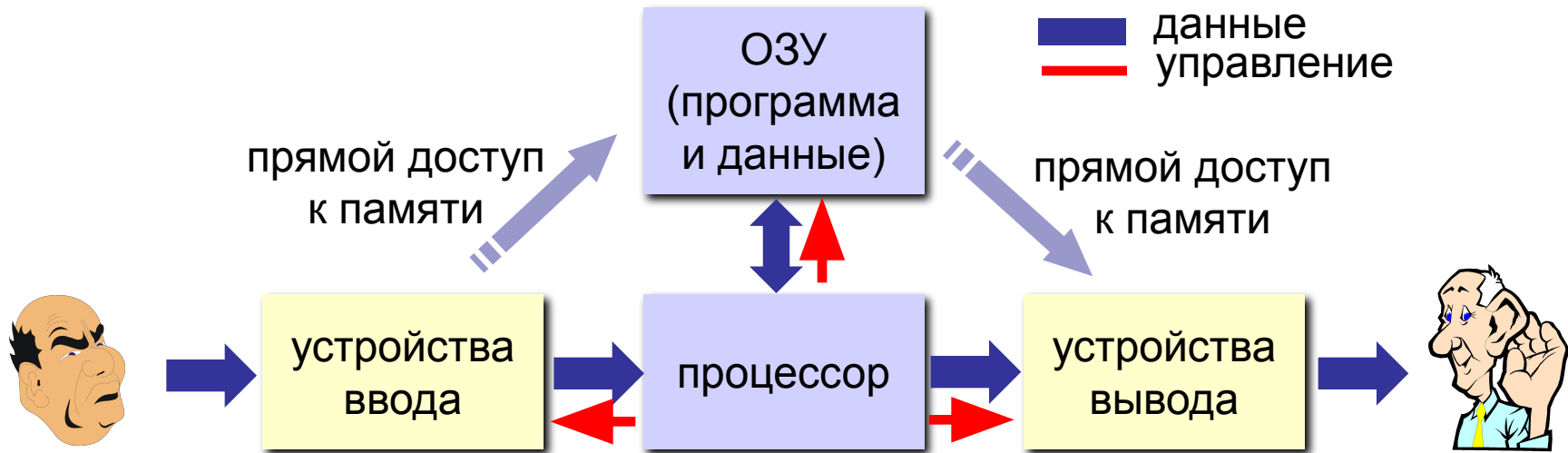
На современных компьютерах **1 байт = 8 бит**.



Архитектура компьютера

Архитектура – принципы действия и взаимосвязи основных устройств компьютера (процессора, ОЗУ, внешних устройств).

Принстонская архитектура (фон Неймана):



Гарвардская архитектура – программы и данные хранятся в разных областях памяти.



скорость (одновременно читаем команду и данные)



нужно больше контактов у процессора

А. Беркс, Х. Голдстайн, Д. Нейман «Предварительный доклад о машине EDVAC» (1945)

1. **Принцип двоичного кодирования:** вся информация кодируется в двоичном виде.
2. **Принцип программного управления:** программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.
3. **Принцип однородности памяти:** программы и данные хранятся в одной и той же памяти.
4. **Принцип адресности:** память состоит из пронумерованных ячеек; процессору в любой момент времени доступна любая ячейка.



Джон фон Нейман

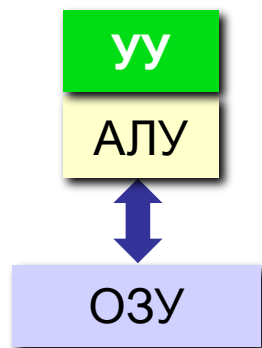
Выполнение программы

Счетчик команд (*IP = Instruction Pointer*) – регистр, в котором хранится адрес следующей команды.

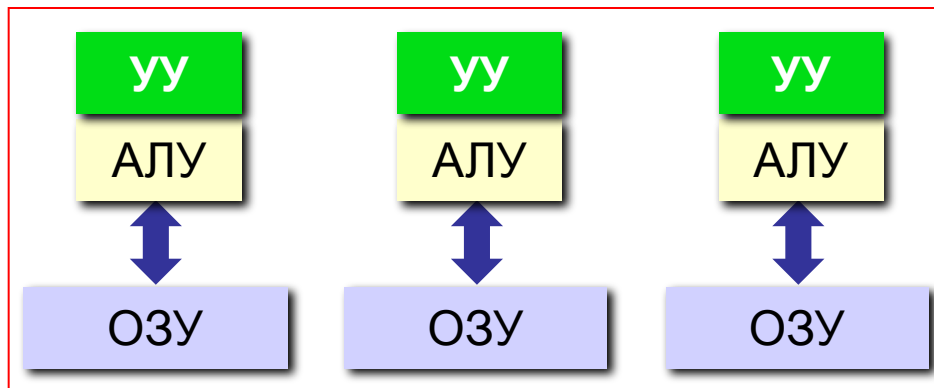


1. **Команда**, расположенная по этому адресу, передается в **УУ**. Если это не команда перехода, регистр **IP** увеличивается на длину команды.
2. УУ расшифровывает **адреса операндов**.
3. Операнды загружаются в **АЛУ**.
4. УУ дает команду АЛУ на **выполнение операции**.
5. **Результат** записывается по нужному адресу.
6. Шаги 1-5 повторяются до получения команды **«стоп»**.

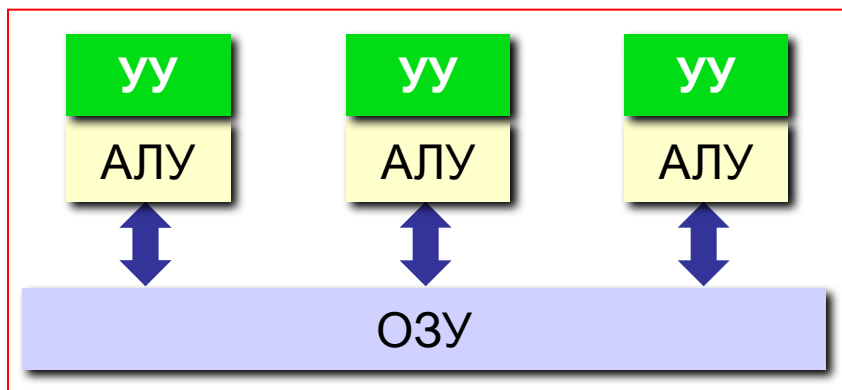
фон Неймана



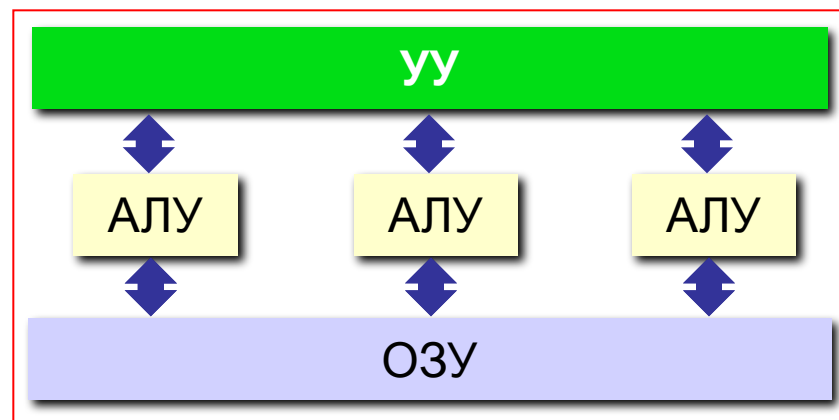
многомашинная (независимые задачи)



многопроцессорная (части одной задачи, по разным программам)



параллельные процессоры (части одной задачи, по одной программе)



Компьютер изнутри

Тема 2. Персональный компьютер

Персональный компьютер (ПК)

ПК – это компьютер, предназначенный для личного использования (доступная цена, размеры, характеристики).



1977 Apple-II



1981 IBM PC
(*personal computer*)



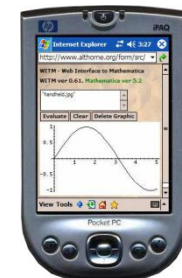
iMac (1999)



**PowerMac G4
Cube (2000)**



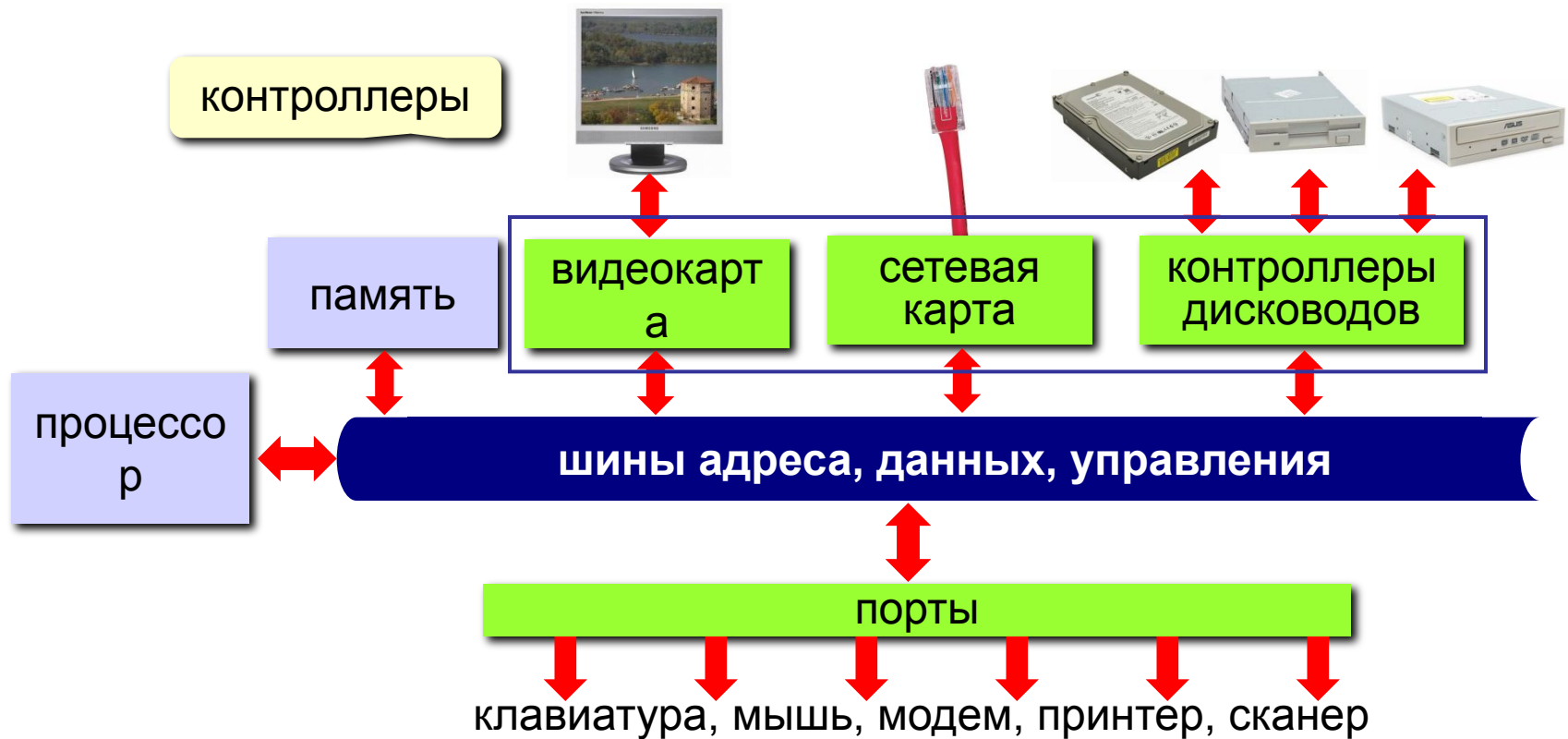
EC-1841



- на **материнской плате** расположены только узлы, которые обрабатывают информацию (процессор и вспомогательные микросхемы, память)
- схемы, управляющие другими устройствами (монитором и т.д.) – это отдельные **платы**, которые вставляются в **слоты расширения**
- **схема стыковки** новых устройств с компьютером общедоступна (стандарт)



- **конкуренция**, удешевление устройств
- производители могут изготавливать **новые** совместимые устройства
- пользователь может собирать ПК «**из кубиков**»



Шина – многожильная линия связи, доступ к которой имеют несколько устройств.

Контроллер – электронная схема, управляющая внешним устройством по сигналам процессора.

Компьютер изнутри

Тема 3. Хранение целых чисел

Целые беззнаковые числа

Беззнаковые данные – не могут быть отрицательными.

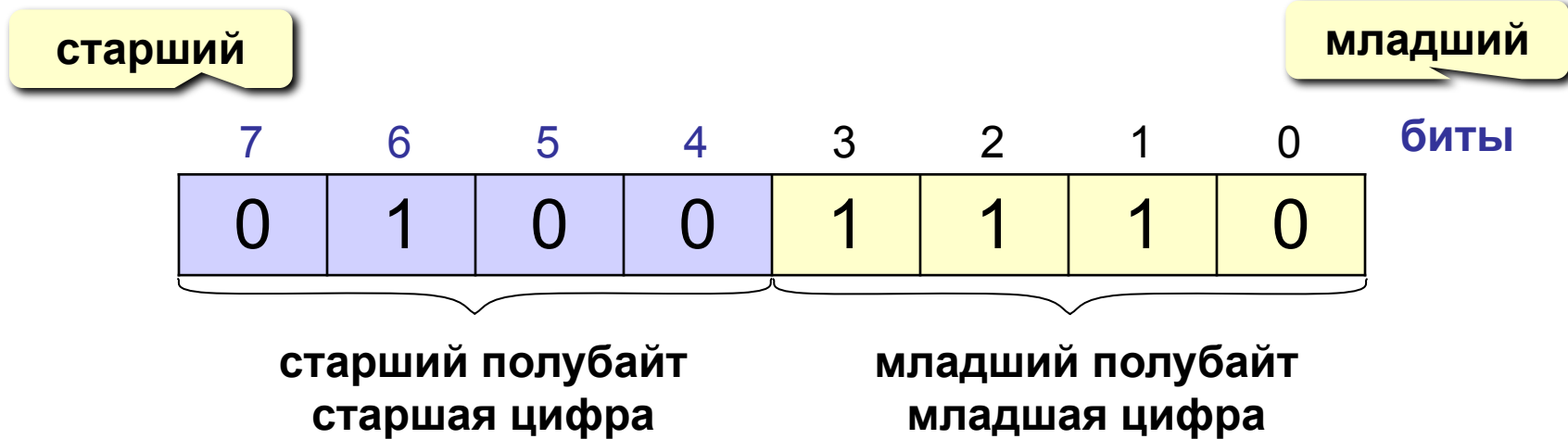
Байт (символ)

память: **1 байт = 8 бит**

диапазон значений **0...255**, $0...FF_{16} = 2^8 - 1$

Си: *unsigned char*

Паскаль: *byte*

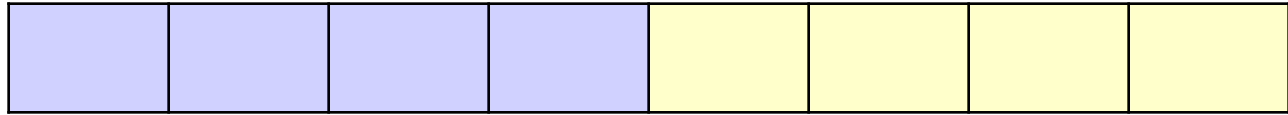


4_{16}

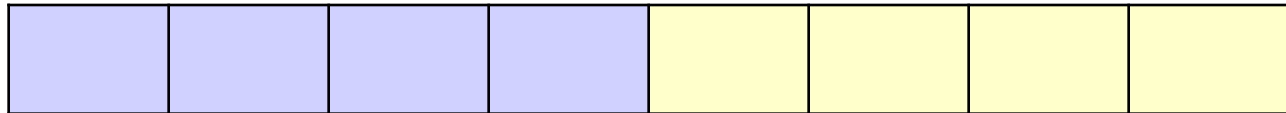
E_{16}

$$1001110_2 = 4E_{16} = 'N'$$

78 =



115 =



Целые беззнаковые числа

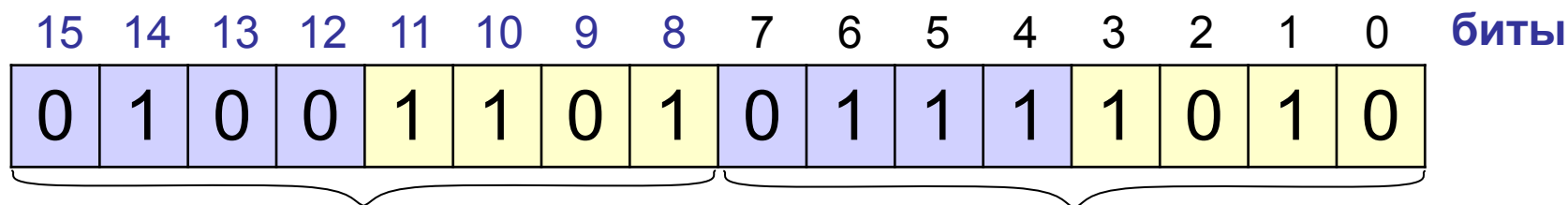
Целое без знака

память: **2 байта = 16 бит**

диапазон значений **$0 \dots 65535$, $0 \dots \text{FFFF}_{16} = 2^{16} - 1$**

Си: *unsigned short int*

Паскаль: *word*



старший байт

младший байт

$4D_{16}$

$7A_{16}$

$100110101111010_2 = 4D7A_{16}$

Длинное целое без знака

память: **4 байта = 32 бита**

диапазон значений **$0 \dots \text{FFFFFFFF}_{16} = 2^{32} - 1$**

Си: *unsigned int*

Паскаль: *dword, longword*

Целые числа со знаком



Сколько места требуется для хранения знака?

Старший (знаковый) бит числа определяет его знак. Если он равен 0, число положительное, если 1, то отрицательное.

«-1» – это такое число, которое при сложении с 1 даст 0.

1 байт:

-1

не помещается в 1 байт!

$$FF_{16} + 1 = 1\ 0\ 0_{16}$$

2 байта:

$$FFFF_{16} + 1 = 1\ 0\ 0\ 0\ 0_{16}$$

4 байта:

$$FFFFFFFF_{16} + 1 = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0_{16}$$

-1

Двоичный дополнительный код

Задача: представить отрицательное число $(-a)$ в двоичном дополнительном коде.

Решение:

1. Перевести число $a-1$ в двоичную систему.
2. Записать результат в разрядную сетку с нужным числом разрядов.
3. Заменить все «0» на «1» и наоборот (*инверсия*).

Пример: $(-a) = -78$, сетка 8 бит

4. $a - 1 = 77 = 1001101_2$

5.

0	1	0	0	1	1	0	1
---	---	---	---	---	---	---	---

6.

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

$= -78$

знаковый бит

Двоичный дополнительный код

20

Проверка: $78 + (-78) = ?$

78 =

0	1	0	0	1	1	1	0
---	---	---	---	---	---	---	---

+

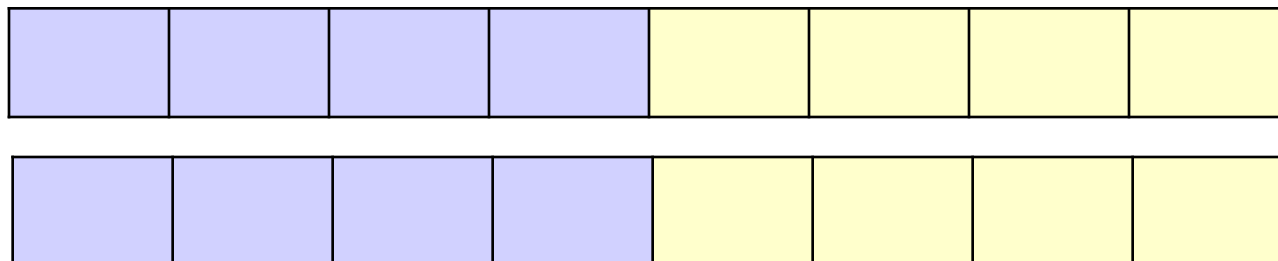
-78 =

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

--	--	--	--	--	--	--	--

$(-a) = -123$, сетка 8 бит

$-123 =$



Целые числа со знаком

Байт (символ) со знаком

память: 1 байт = 8 бит

диапазон значений:

max	0	1	1	1	1	1	1	127
min	1	0	0	0	0	0	0	-128

$-128 = -2^7 \dots 127 = 2^8 - 1$

Си: *char* Паскаль: *shortint*



можно работать с отрицательными числами



уменьшился диапазон положительных чисел

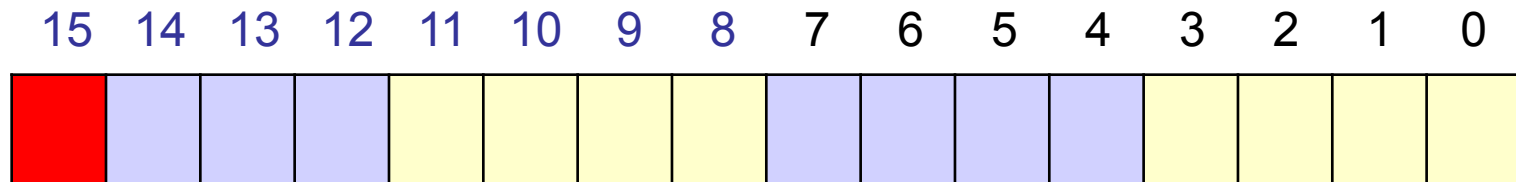
Целые числа со знаком

Слово со знаком

память: 2 байта = 16 бит

диапазон значений

– 32768 ... 32767



Си: *short int* Паскаль: *smallint*

Двойное слово со знаком

память – 4 байта

диапазон значений

– 2^{31} ... $2^{31}-1$

Си: *int* Паскаль: *integer*

Переполнение разрядной сетки: в результате сложения больших положительных чисел получается отрицательное (**перенос в знаковый бит**).

	7	6	5	4	3	2	1	0	
	0	1	0	0	0	0	0	0	64
+	0	1	0	0	0	0	0	0	64
	1	0	0	0	0	0	0	0	- 128

Перенос: при сложении больших (по модулю) отрицательных чисел получается положительное (перенос за границы разрядной сетки).

	7	6	5	4	3	2	1	0	
	1	0	0	0	0	0	0	0	– 128
+	1	0	0	0	0	0	0	0	– 128
1	0	0	0	0	0	0	0	0	0

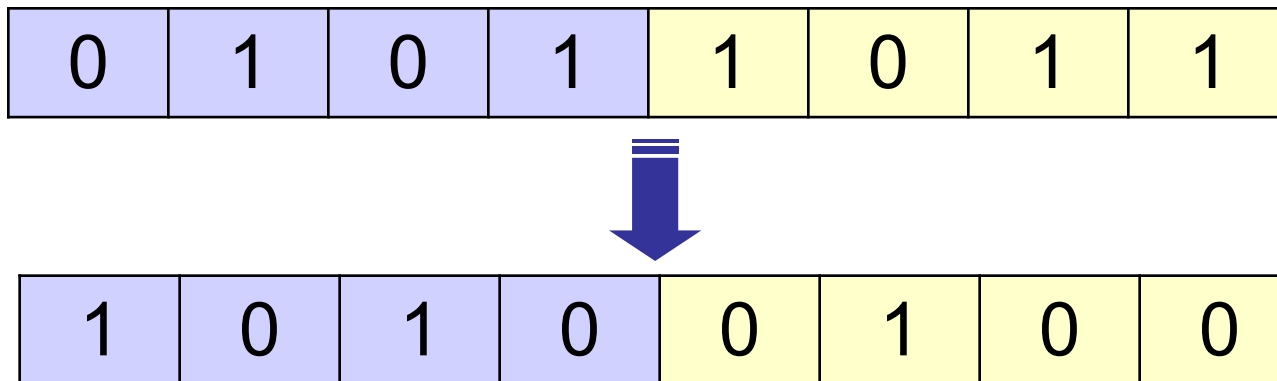
в специальный
бит переноса

Компьютер изнутри

Тема 4. Битовые операции

Инверсия (операция НЕ)

Инверсия – это замена всех «0» на «1» и наоборот.



Си:

```
int n;  
n = ~n;
```

Паскаль:

```
var n: integer;  
n := not n;
```

Операция И

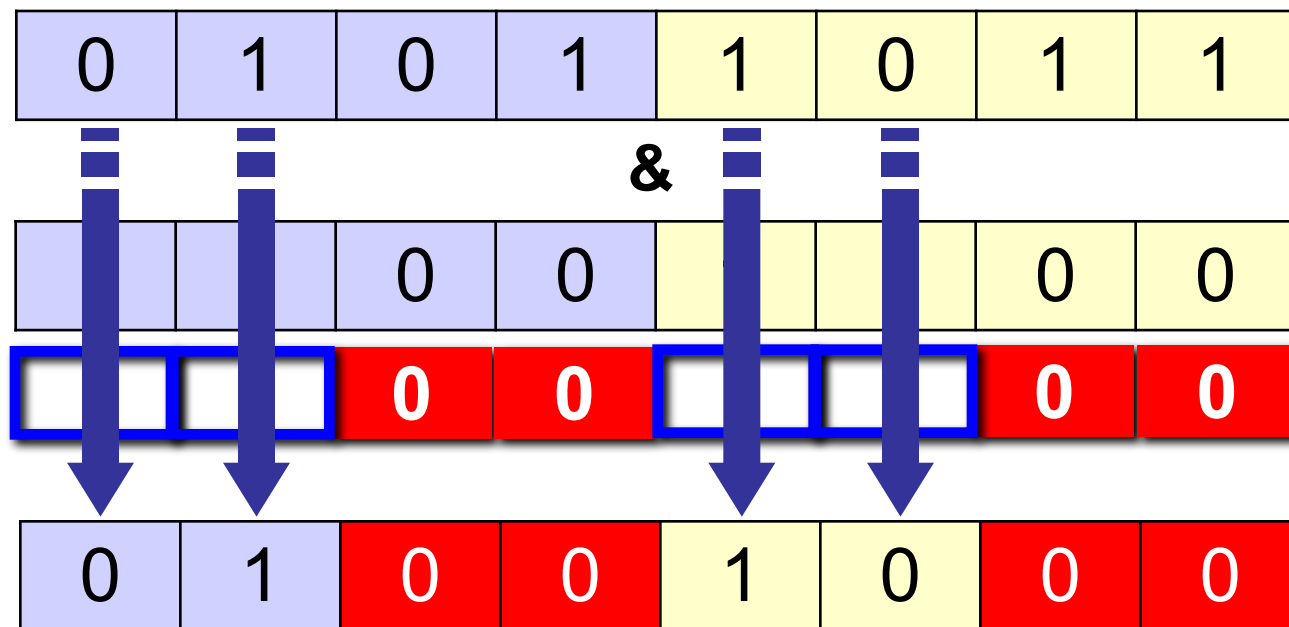
Обозначения:

И, $\wedge$, & (Си), and (Паскаль)

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

$x \& 0 = 0$
 $x \& 1 = x$

маска

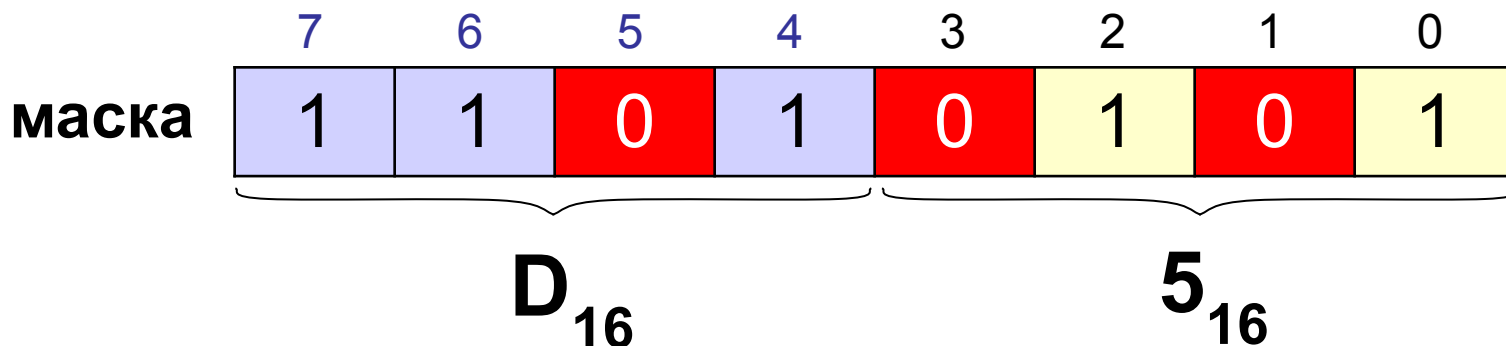


$$5B_{16} \& CC_{16} = 48_{16}$$

Операция И – обнуление битов

Маска: обнуляются все биты, которые в маске равны «0».

Задача: обнулить 1, 3 и 5 биты числа, оставив остальные без изменения.



Си:

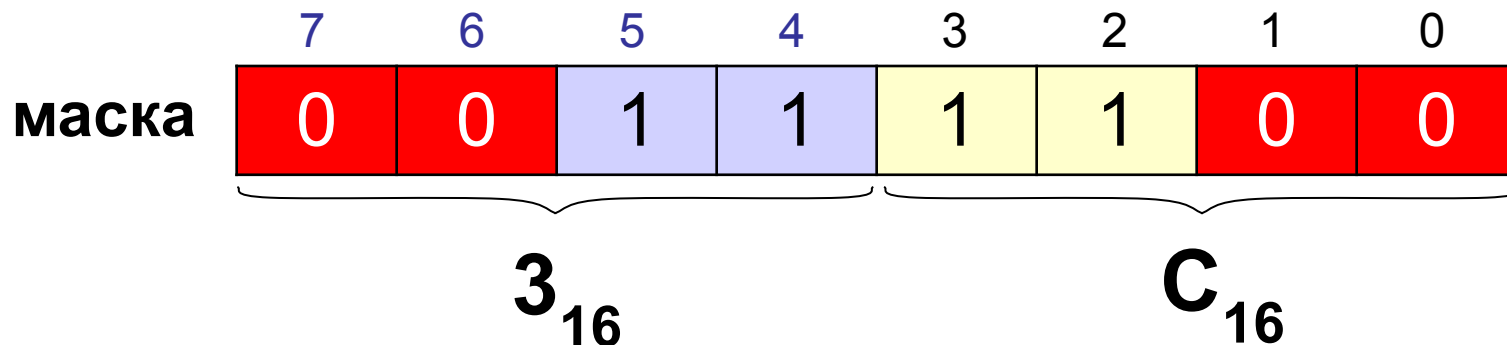
```
int n;  
n = n & 0xD5;
```

Паскаль:

```
var n: integer;  
n := n and $D5;
```

Операция И – проверка битов

Задача: проверить, верно ли, что все биты 2...5 – нулевые.



Си:

```
if ( n & 0x3C == 0 )  
    printf ("Биты 2-5 нулевые.");  
else printf ("В битах 2-5 есть ненулевые.");
```

Паскаль:

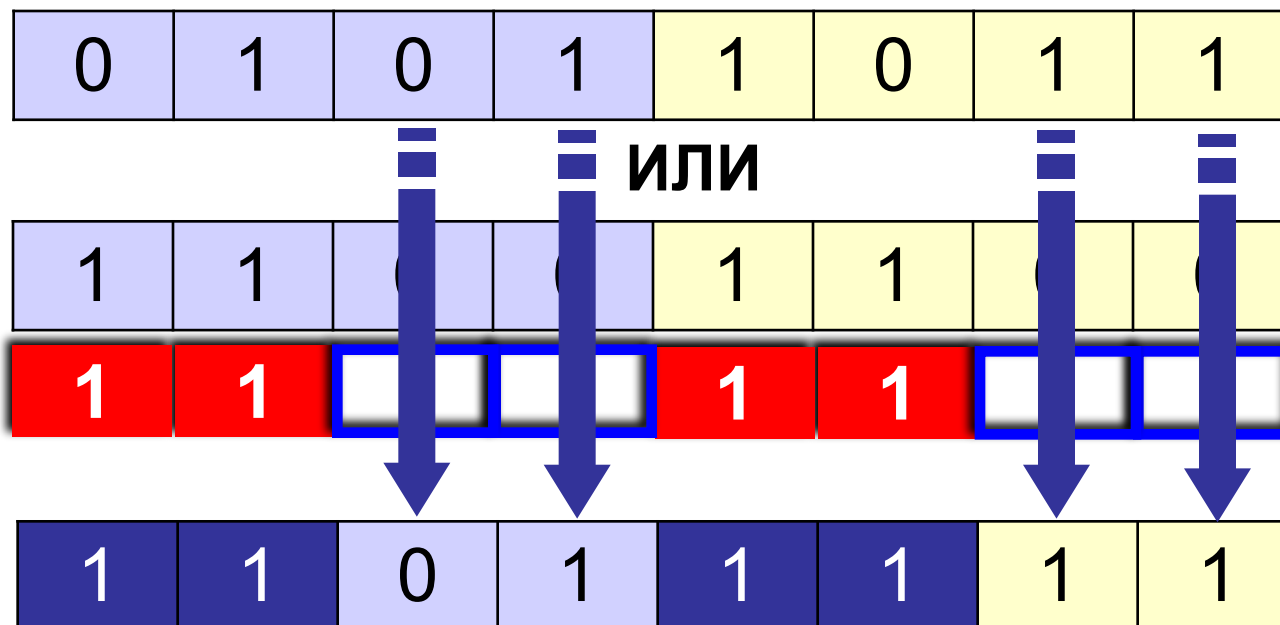
```
if (n and $3C) = 0  
    writeln ('Биты 2-5 нулевые.')  
else writeln ('В битах 2-5 есть ненулевые.');
```

Операция ИЛИ

Обозначения:

ИЛИ, $\vee$, $|$ (Си), **or** (Паскаль)

A	B	A или B
0	0	0
0	1	1
1	0	1
1	1	1

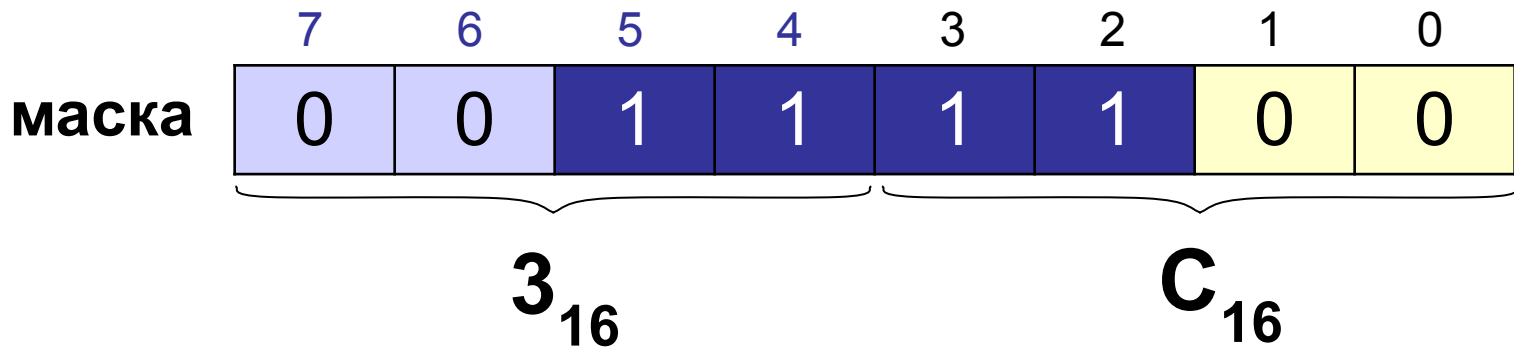


x ИЛИ 0 = x
x ИЛИ 1 = 1

$$5B_{16} | CC_{16} = DF_{16}$$

Операция ИЛИ – установка битов в 1

Задача: установить все биты 2...5 равными 1, не меняя остальные.



Си:

```
n = n | 0x3C;
```

Паскаль:

```
n := n or $3C;
```

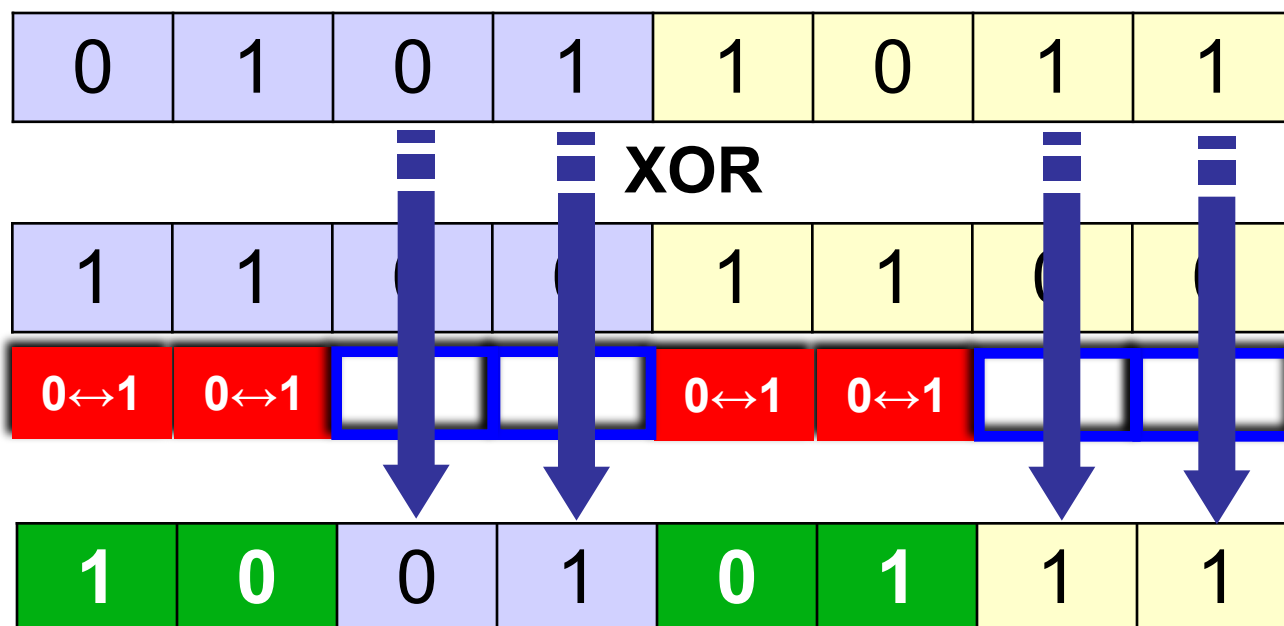

Операция «исключающее ИЛИ»

Обозначения:

$\oplus$, $\wedge$ (Си), **xor** (Паскаль)

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

$x \text{ XOR } 0 = x$
 $x \text{ XOR } 1 = \neg x$

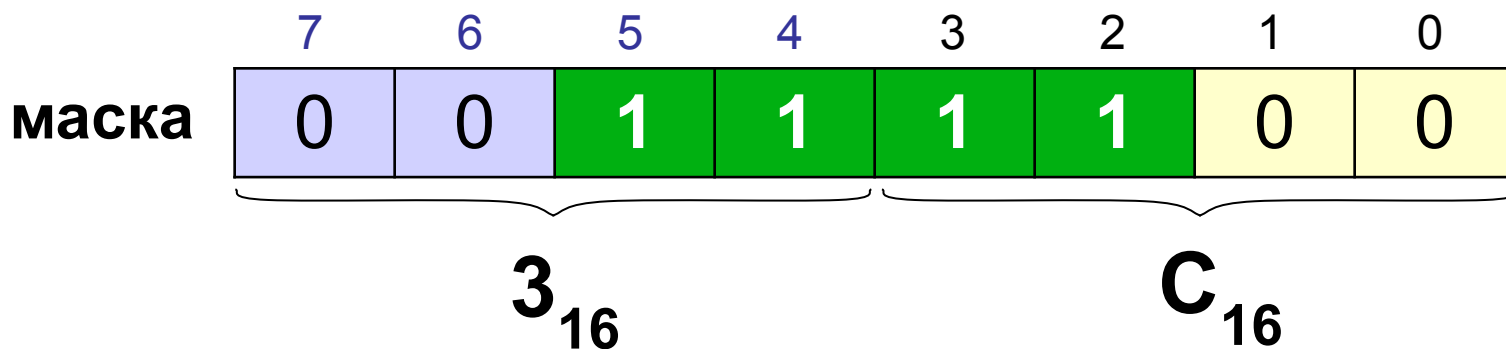


маска

$$5B_{16} \wedge CC_{16} = 97_{16}$$

«Исключающее ИЛИ» – инверсия битов

Задача: выполнить инверсию для битов 2...5, не меняя остальные.



Си:

```
n = n ^ 0x3C;
```

Паскаль:

```
n := n xor $3C;
```

«Исключающее ИЛИ» – шифровка

$$(0 \text{ xor } 0) \text{ xor } 0 = 0$$

$$(0 \text{ xor } 1) \text{ xor } 1 = 0$$

$$(1 \text{ xor } 0) \text{ xor } 0 = 1$$

$$(1 \text{ xor } 1) \text{ xor } 1 = 1$$

код (шифр)

$$(X \text{ xor } Y) \text{ xor } Y = X$$



«Исключающее ИЛИ» – обратимая операция.

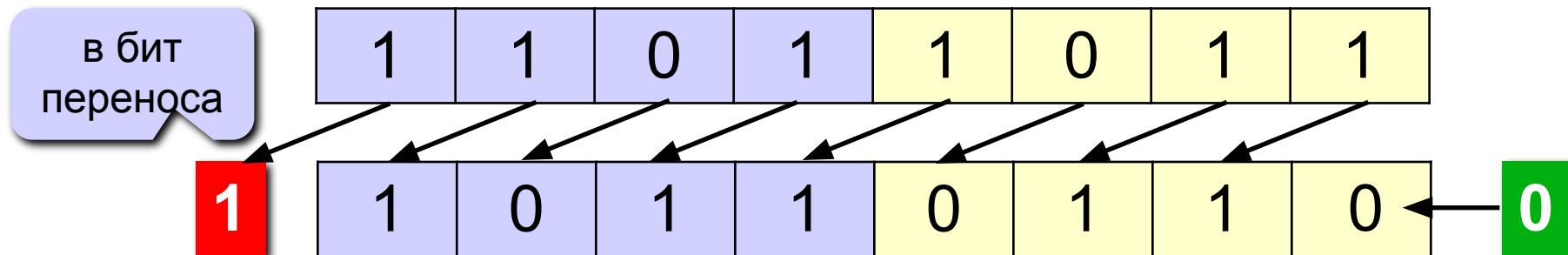
Шифровка:

выполнить для каждого байта текста операцию **XOR** с байтом-шифром.

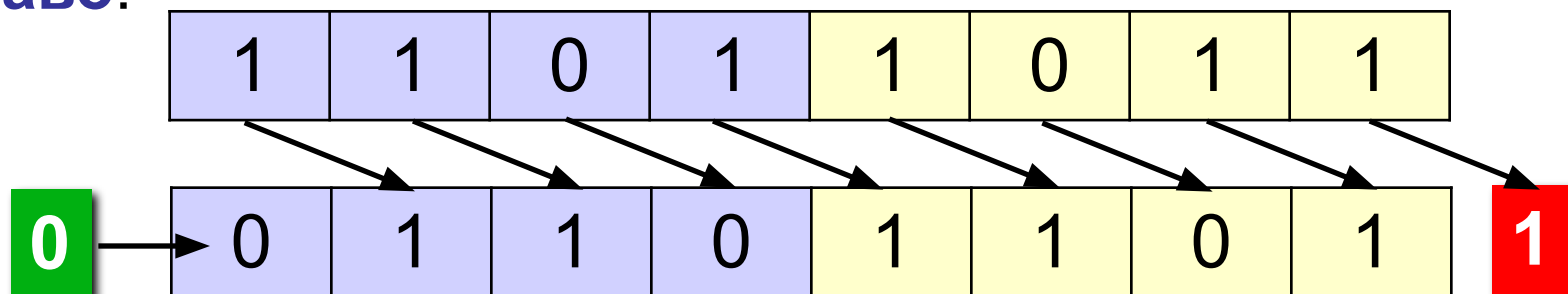
Расшифровка: сделать то же самое с тем же шифром.

Логический сдвиг

Влево:



Вправо:



Си:

```
n = n << 1;
n = n >> 1;
```

Паскаль:

shift left

в бит переноса

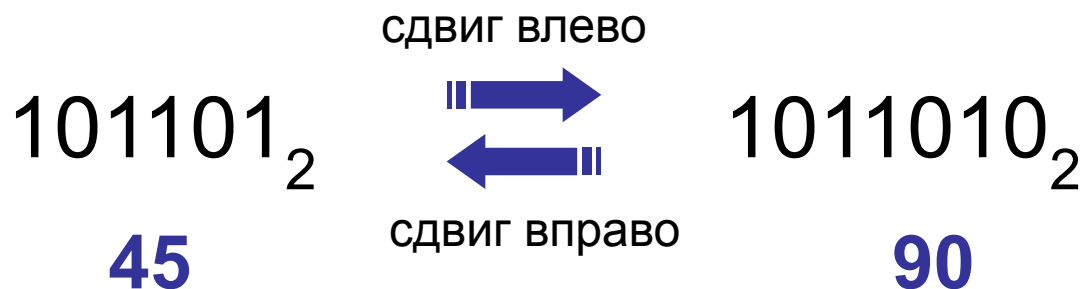
```
n := n shl 1;
n := n shr 1;
```

shift right

Логический сдвиг



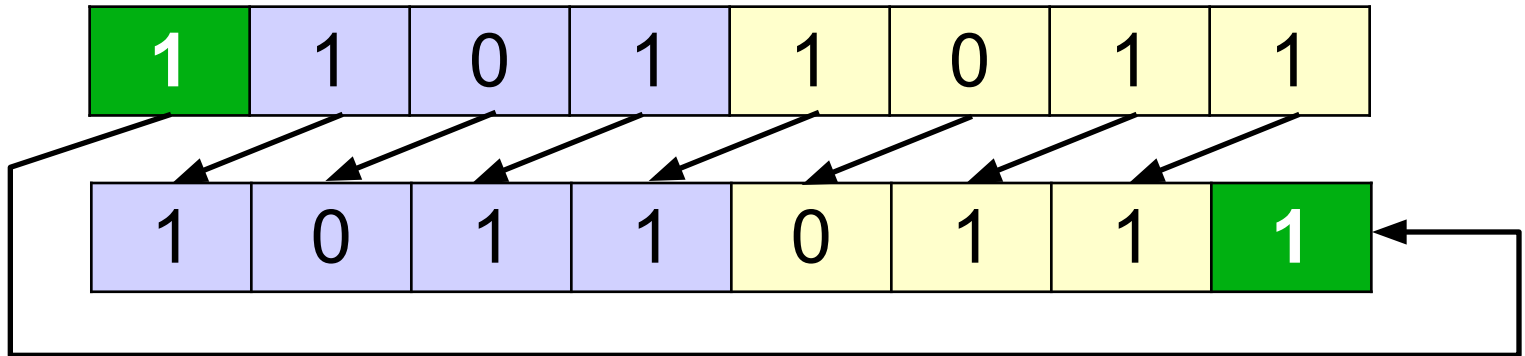
Какой арифметической операции равносильен логический сдвиг влево (вправо)? При каком условии?



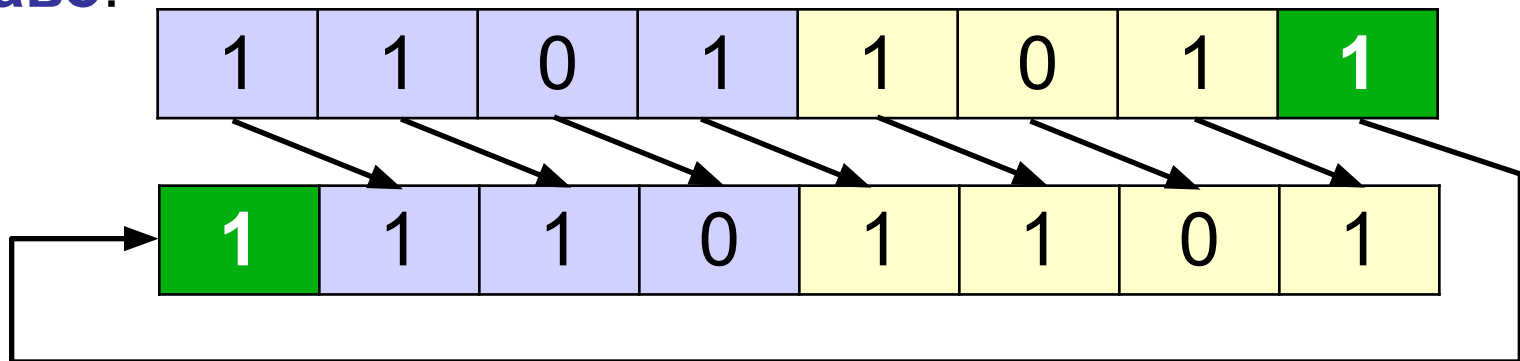
Логический сдвиг влево (вправо) – это быстрый способ **умножения** (деления без остатка) **на 2**.

Циклический сдвиг

Влево:



Вправо:

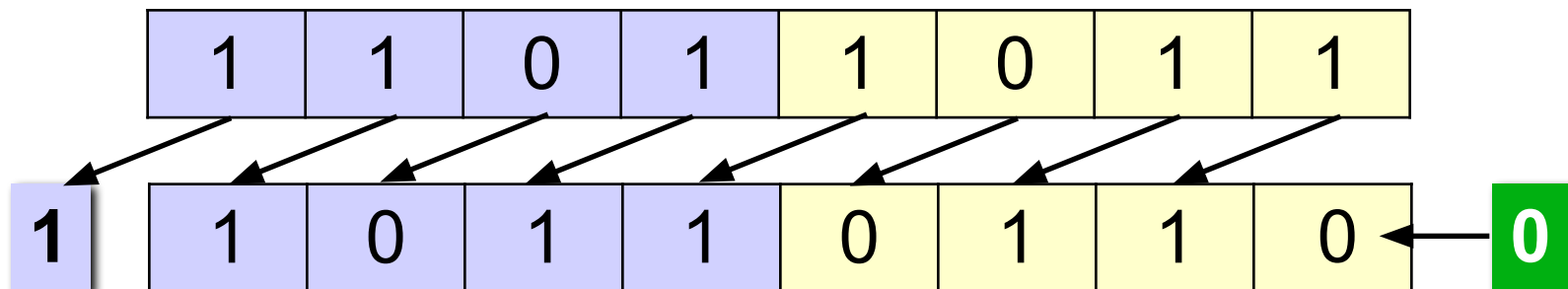


Си, Паскаль: –

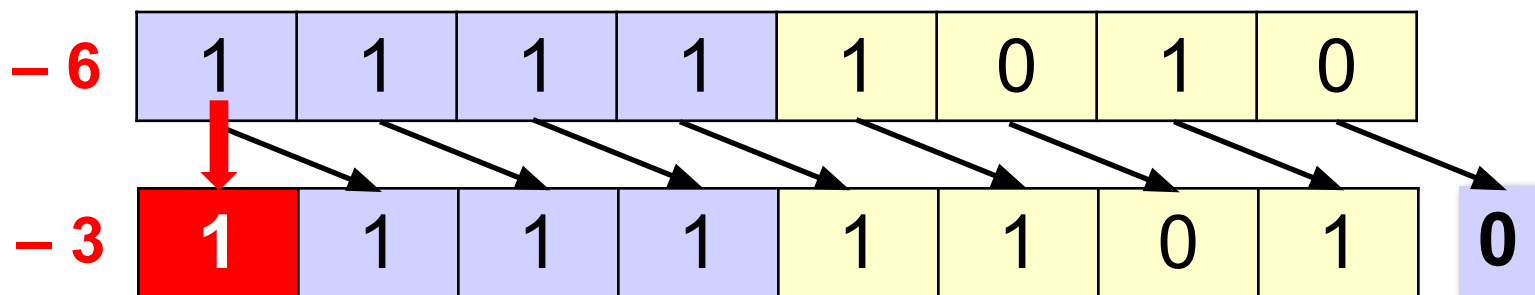
только через Ассемблер

Арифметический сдвиг

Влево (= логическому):



Вправо (знаковый бит не меняется!):



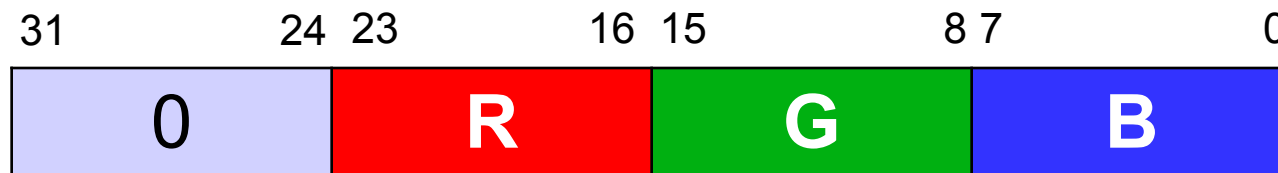
Си:

```
n = -6;
n = n >> 1;
```

Паскаль: –

Пример

Задача: в целой переменной **n** (**32 бита**) закодирована информация о цвете пикселя в **RGB**:



Выделить в переменные R, G, B составляющие цвета.

Вариант 1:

1. Обнулить все биты, кроме **G**.

Маска для выделения **G**: **0000FF00**₁₆



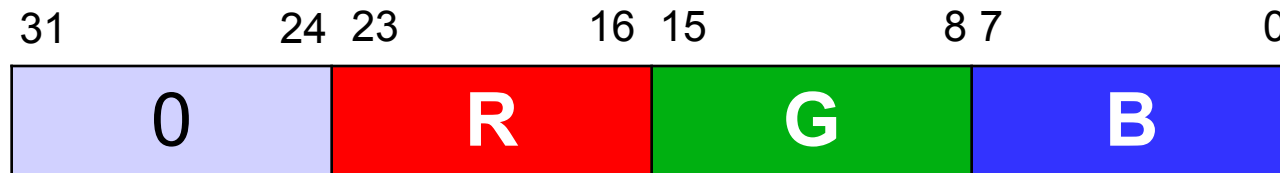
А надо ли
обнулять?

2. Сдвинуть вправо так, чтобы число **G** передвинулось в младший байт.

Си: `G = (n & 0xFF00) >> 8;`

Паскаль: `G := (n and $FF00) shr 8;`

Пример

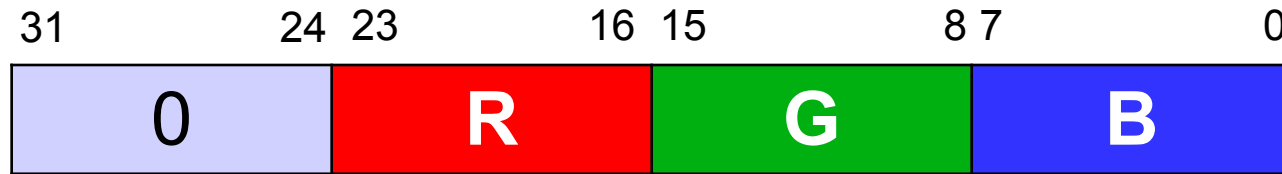


Вариант 2:

1. Сдвинуть вправо так, чтобы число **G** передвинулось в младший байт.
2. Обнулить все биты, кроме **G**.
Маска для выделения **G**: **000000FF**₁₆

Си: `G = (n >> 8) & 0xFF;`

Паскаль: `G := (n shr 8) and $FF;`



Си: R =

B =

Паскаль: R :=

B :=

Компьютер изнутри

Тема 5. Вещественные числа

Перевод дробных чисел

10 → 2

$$0,375 = 0,011_2$$

$$\begin{array}{r} \times \quad 2 \\ \hline \end{array}$$

$$0,750$$

$$0,75$$

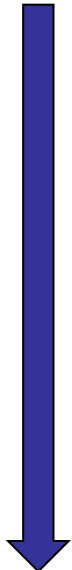
$$\begin{array}{r} \times \quad 2 \\ \hline \end{array}$$

$$1,50$$

$$0,5$$

$$\begin{array}{r} \times \quad 2 \\ \hline \end{array}$$

$$1,0$$



$$0,7 = ?$$

$$0,7 = 0,101100110...$$

$$= 0,1(0110)_2$$

Многие дробные числа нельзя представить в виде **конечных** двоичных дробей.

Для их точного хранения требуется **бесконечное** число разрядов.

Большинство дробных чисел хранится в памяти с ошибкой.

2 → 10

$$2^{-2} = \frac{1}{2^2} = 0,25$$

2 1 0 -1 -2 -3 разряды

$$101,011_2$$

$$= 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

$$= 4 + 1 + 0,25 + 0,125 = 5,375$$

Примеры:

$$0,625 =$$

$$3,875 =$$

Нормализация двоичных чисел

$$X = s \cdot M \cdot 2^e$$

s – **знак** (1 или -1)

M – **мантисса**, $M = 0$ или $1 \leq M <$

e – **порядок** 2

Пример:

знак

мантисса

порядок

$$15,625 = 1111,101_2 = \textcircled{1} \boxed{1,111101_2} \cdot \textcircled{2^3}$$

$$3,375 =$$

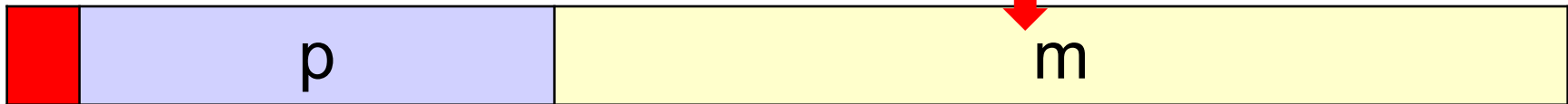
Нормализованные числа в памяти

IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754)

$$15,625 = 1 \cdot 1,111101_2 \cdot 2^3$$

$$s = 1 \quad e = 3$$

$$M = 1,111101_2$$



Порядок со сдвигом:
 $p = e + E$ (сдвиг)

Дробная часть мантииссы:
 $m = M - 1$

Знаковый бит:
 0, если $s = 1$
 1, если $s = -1$



Целая часть M всегда 1,
 поэтому не хранится в памяти!

Нормализованные числа в памяти

Тип данных	Размер, байт	Мантисса, бит	Порядок, бит	Сдвиг порядка, Е	Диапазон модулей	Точность, десятичн. цифр
float single	4	23	8	127	$3,4 \cdot 10^{-38}$... $3,4 \cdot 10^{38}$	7
double double	8	52	11	1023	$1,7 \cdot 10^{-308}$... $1,7 \cdot 10^{308}$	15
long double extended	10	64	15	16383	$3,4 \cdot 10^{-4932}$... $3,4 \cdot 10^{4932}$	19

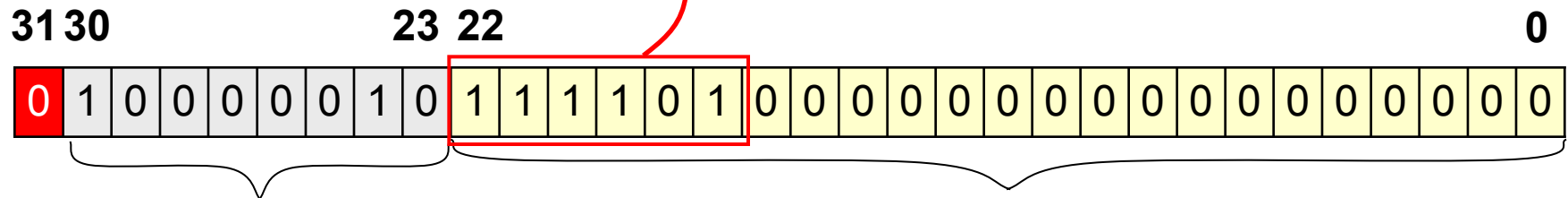
Типы данных для языков: **Си**

Паскаль

Вещественные числа в памяти

$$15,625 = \cancel{1},111101_2 \cdot 2^3$$

4 байта = 32 бита

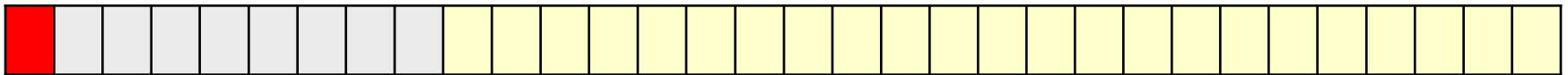


$$p = e + 127 = 130$$

$$= 10000010_2$$

$$m = M - 1 = 0,111101_2$$

$$3,375 =$$



Арифметические операции

сложение

$$5,5 + 3 = 101,1_2 + 11_2 = 8,5 = 1000,1_2$$

1. Порядок выравнивается до большего

$$5,5 = 1,011_2 \cdot 2^2$$

$$3 = 1,1_2 \cdot 2^1 = 0,11_2 \cdot 2^2$$

2. Мантиссы складываются

$$\begin{array}{r} 1,011_2 \\ + 0,110_2 \\ \hline 10,001_2 \end{array}$$

3. Результат нормализуется (с учетом порядка)

$$10,001_2 \cdot 2^2 = 1,0001_2 \cdot 2^3 = 1000,1_2 = 8,5$$

ВЫЧИТАНИЕ

$$10,75 - 5,25 = 1010,11_2 - 101,01_2 = 101,1_2 = 5,5$$

1. Порядок выравнивается до большего

$$10,75 = 1,01011_2 \cdot 2^3$$

$$5,25 = 1,0101_2 \cdot 2^2 = 0,10101_2 \cdot 2^3$$

2. Мантиссы вычитаются

$$\begin{array}{r} 1,01011_2 \\ - 0,10101_2 \\ \hline 0,10110_2 \end{array}$$

3. Результат нормализуется (с учетом порядка)

$$0,1011_2 \cdot 2^3 = 1,011_2 \cdot 2^2 = 101,1_2 = 5,5$$

Арифметические операции

умножение

$$7 \cdot 3 = 111_2 \cdot 11_2 = 21 = 10101_2$$

1. Мантиссы умножаются

$$7 = 1,11_2 \cdot 2^2$$

$$3 = 1,1_2 \cdot 2^1$$

$$\begin{array}{r} 1,11_2 \\ \times 1,1_2 \\ \hline 111_2 \\ 111_2 \\ \hline 10,101_2 \end{array}$$

2. Порядки складываются: $2 + 1 = 3$

3. Результат нормализуется (с учетом порядка)

$$10,101_2 \cdot 2^3 = 1,0101_2 \cdot 2^4 = 10101_2 = 21$$

Арифметические операции

деление

$$17,25 : 3 = 10001,01_2 : 11_2 = 5,75 = 101,11_2$$

1. Мантииссы делятся

$$17,25 = 1,000101_2 \cdot 2^4$$

$$3 = 1,1_2 \cdot 2^1$$

$$1,000101_2 : 1,1_2 = 0,10111_2$$

2. Порядки вычитаются: $4 - 1 = 3$

3. Результат нормализуется (с учетом порядка)

$$0,10111_2 \cdot 2^3 = 1,0111_2 \cdot 2^2 = 101,11_2 = 5,75$$

