Компьютер изнутри

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

© К.Ю. Поляков, 2007

Компьютер изнутри

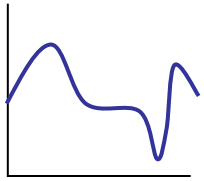
Тема 1. Основные принципы

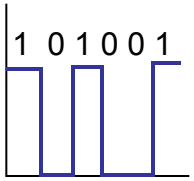
© К.Ю. Поляков, 2007

Определения

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

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





Hardware – аппаратное обеспечение, «железо».

Software – программное обеспечение, «софт».

Определения

Программа – это последовательность команд, которые должен выполнить компьютер.

Команда – это описание операции (1…4 байта):

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

Типы команд:

- **безадресные** (1 байт) **inc AX** – увеличить *регистр* AX на 1

  регистр – ячейка быстродействующей памяти, расположенная в процессоре

- **одноадресные** (2 байта) **add AX, 2** AX ← AX + 2

add ax,	2

- **двухадресные** (3 байта)

add	X	2

X ← X + 2

- **трехадресные** (4 байта)

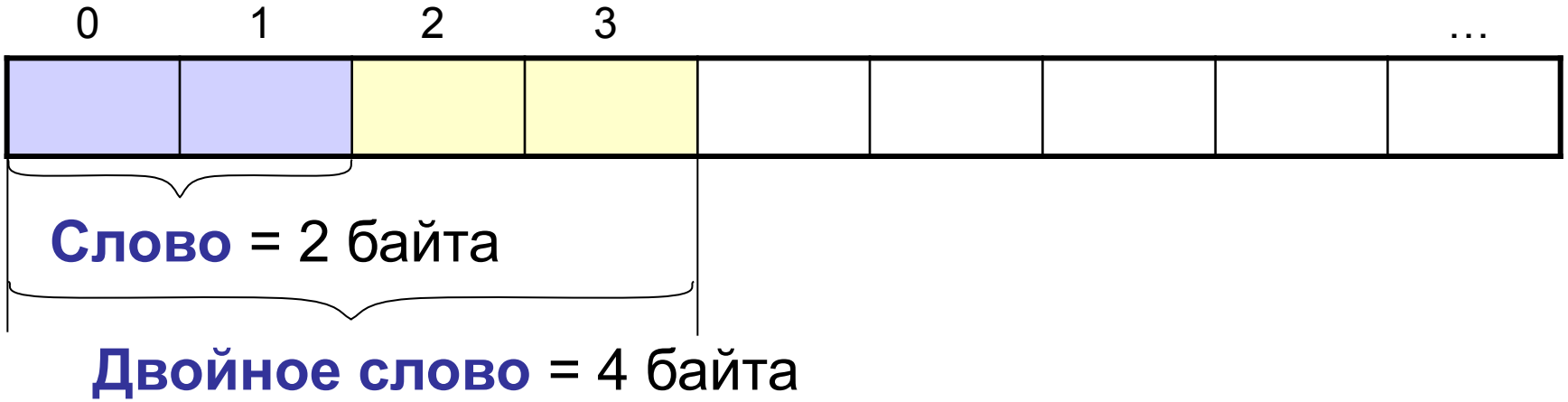
add	X	2	Y

Y ← X + 2

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).

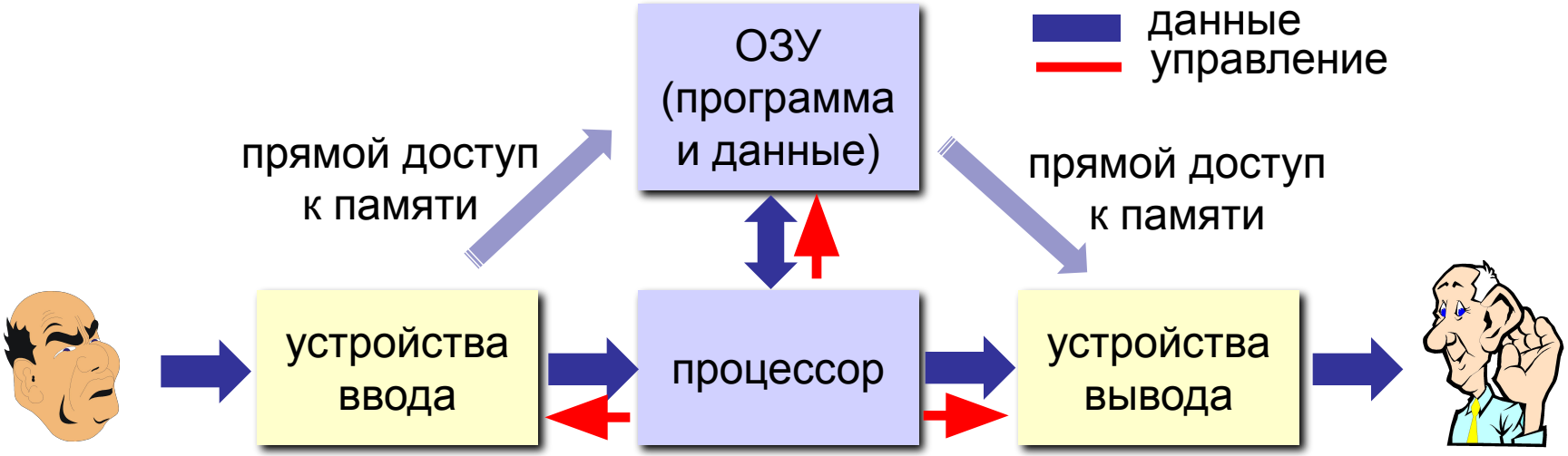
На современных компьютерах **1 байт = 8 бит**.



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

Принципы фон Неймана

«Предварительный доклад о машине EDVAC» (1945)

Джон фон Нейман

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.

Выполнение программы

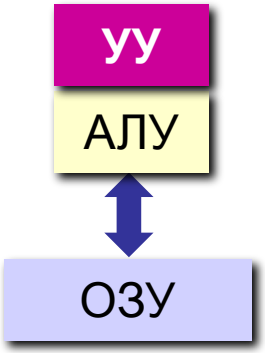
Счетчик команд (*IP* = *Instruction Pointer*) – регистр, в котором хранится адрес следующей команды.

IP $AB3D_{16}$ → по адресу $AB3D_{16}$

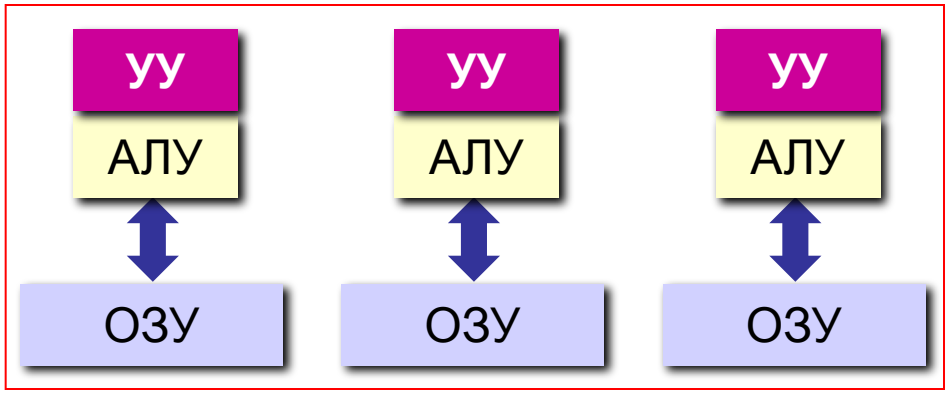
1. **Команда**, расположенная по этому адресу, передается **в УУ**. Если это не команда перехода, регистр **IP** увеличивается на длину команды.
2. УУ расшифровывает **адреса операндов**.
3. Операнды загружаются **в АЛУ**.
4. УУ дает команду АЛУ на **выполнение операции**.
5. **Результат** записывается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

Архитектуры компьютеров

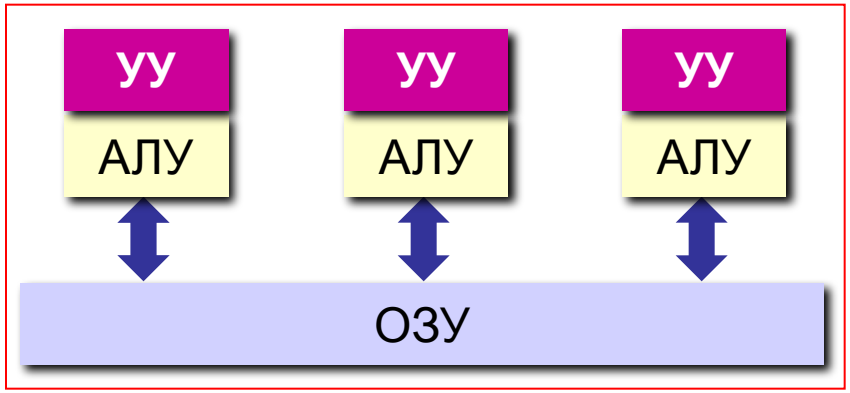
фон Неймана



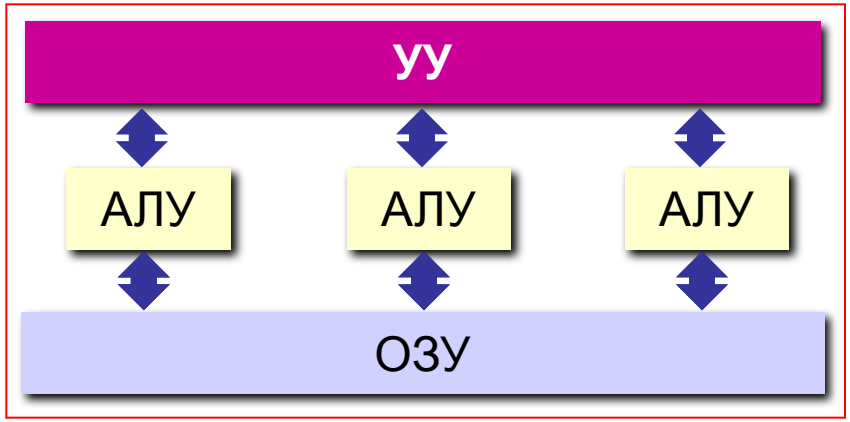
многомашинная
(независимые задачи)



многопроцессорная
(части одной задачи,
по разным программам)



параллельные процессоры
(части одной задачи,
по одной программе)



Компьютер изнутри

Тема 2. Персональный компьютер

© К.Ю. Поляков, 2007

Персональный компьютер (ПК)

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

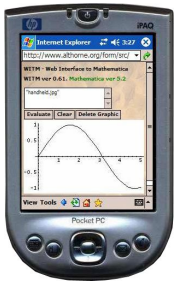
1977 Apple-II

1981 IBM PC
(*personal computer*)

***iMac* (1999)**

***PowerMac G4 Cube* (2000)**

EC-1841

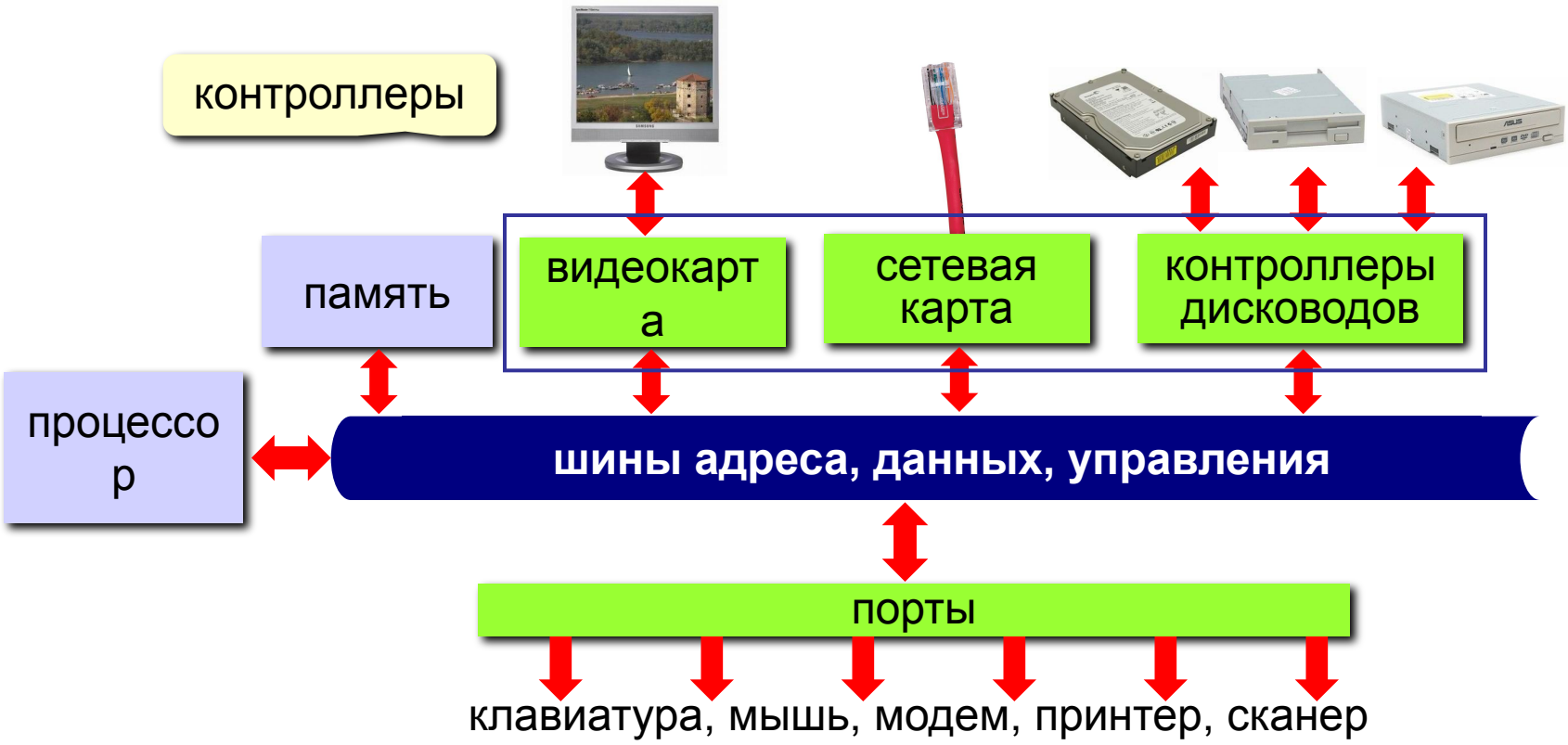


Принцип открытой архитектуры

- 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**»

Взаимосвязь блоков ПК



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

© К.Ю. Поляков, 2007

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***

старший **младший**

7	6	5	4	3	2	1	0	**биты**
0	1	0	0	1	1	1	0	

старший полубайт
старшая цифра

4_{16}

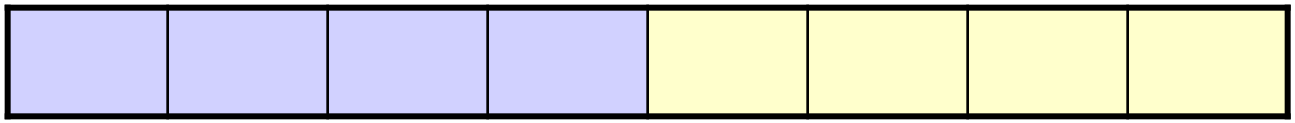
младший полубайт
младшая цифра

E_{16}

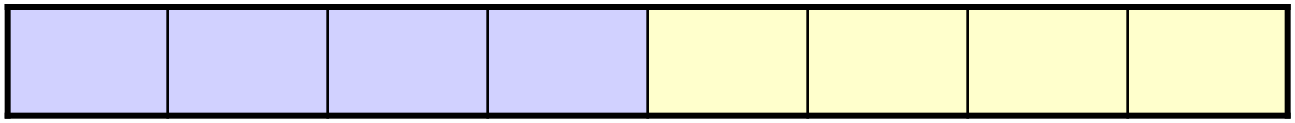
$1001110_2 = 4E_{16}$ = ‘N’

Примеры

78 =



115 =



Целые беззнаковые числа

Целое без знака

память: **2 байта = 16 бит**

диапазон значений **0…65535,** **$0…FFFF_{16} = 2^{16}-1$**

Си: ***unsigned int*** **Паскаль:** ***word***

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	биты
0	1	0	0	1	1	0	1	0	1	1	1	1	0	1	0	

старший байт **младший байт**

$4D_{16}$ $7A_{16}$

$$100110101111010_2 = 4D7A_{16}$$

Длинное целое без знака

память: **4 байта = 32 бита**

диапазон значений **$0…FFFFFFFF_{16} = 2^{32}-1$**

Си: ***unsigned long int*** **Паскаль:** ***dword***

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.

не помещается в 1 байт!

1 байт:

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

2 байта: $FFFF_{16} + 1 = 1\,0\,0\,0\,0_{16}$

4 байта: $FFFFFFFF_{16} + 1 = 1\,0\,0\,0\,0\,0\,0\,0\,0_{16}$

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

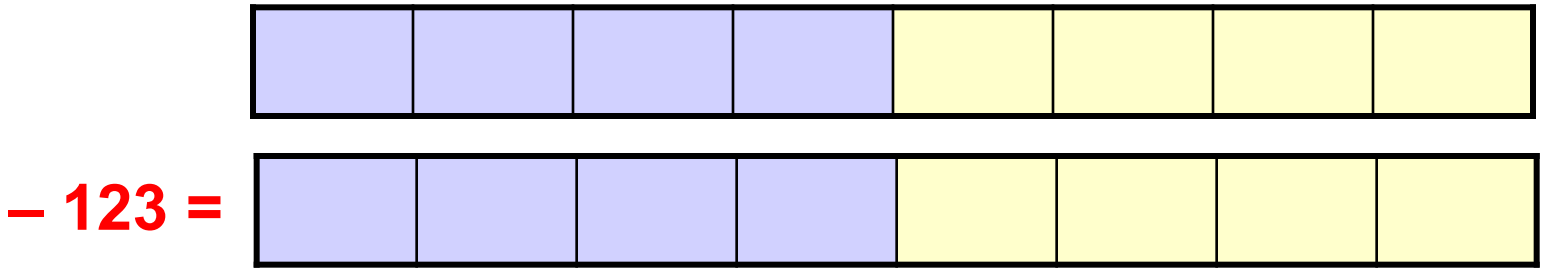
Двоичный дополнительный код

Проверка: **78 + (– 78) = ?**

78 =	0	1	0	0	1	1	1	0
+								
– 78 =	1	0	1	1	0	0	1	0

Пример

(– a) = – 123, сетка 8 бит



Целые числа со знаком

Байт (символ) со знаком

память: **1 байт = 8 бит**

диапазон значений:

max	0	1	1	1	1	1	1	1	**127**
min	1	0	0	0	0	0	0	0	**– 128**

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

Си: ***char*** **Паскаль: –**

можно работать с отрицательными числами

уменьшился диапазон положительных чисел

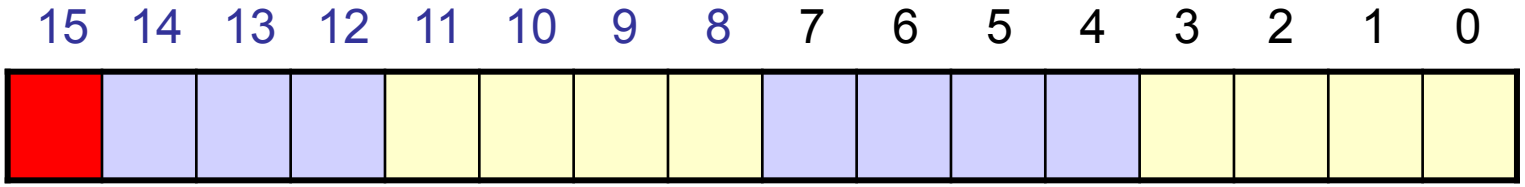
Целые числа со знаком

Слово со знаком

память: **2 байта = 16 бит**

диапазон значений

– 32768 … 32767



Си: ***int*** **Паскаль:** ***integer***

Двойное слово со знаком

память – **4 байта**

диапазон значений

– 2^{31} … 2^{31}-1

Си: ***long int*** **Паскаль:** ***longint***

Ошибки

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**

Ошибки

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

© К.Ю. Поляков, 2007

Инверсия (операция НЕ)

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

0	1	0	1	1	0	1	1

1	0	1	0	0	1	0	0

Си:

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

Паскаль:

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

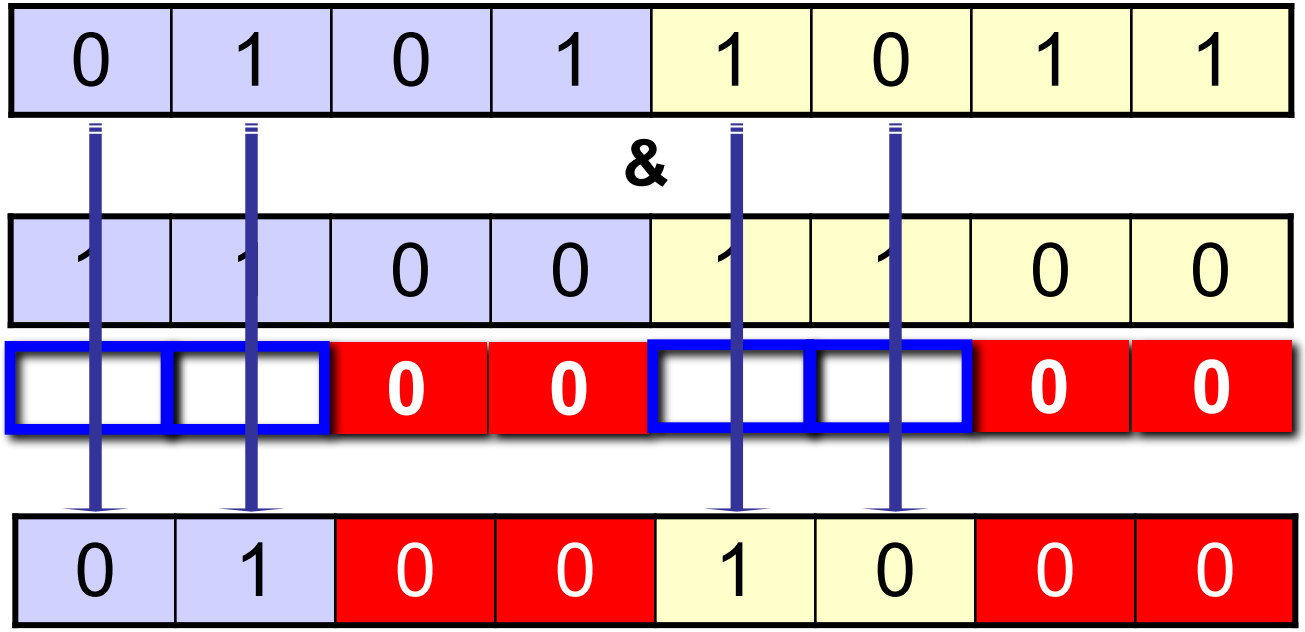
Операция И

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

Обозначения:

И, **∧**, **&** (Си), **and** (Паскаль)

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.

	7	6	5	4	3	2	1	0
маска	1	1	0	1	0	1	0	1

D_{16} 5_{16}

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.

	7	6	5	4	3	2	1	0
маска	0	0	1	1	1	1	0	0

3_{16} C_{16}

Си:

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

Паскаль:

```
if (n and $3C) = 1
     writeln ('Биты 2-5 нулевые.')
else writeln ('В битах 2-5 есть ненулевые.');
```

Операция ИЛИ

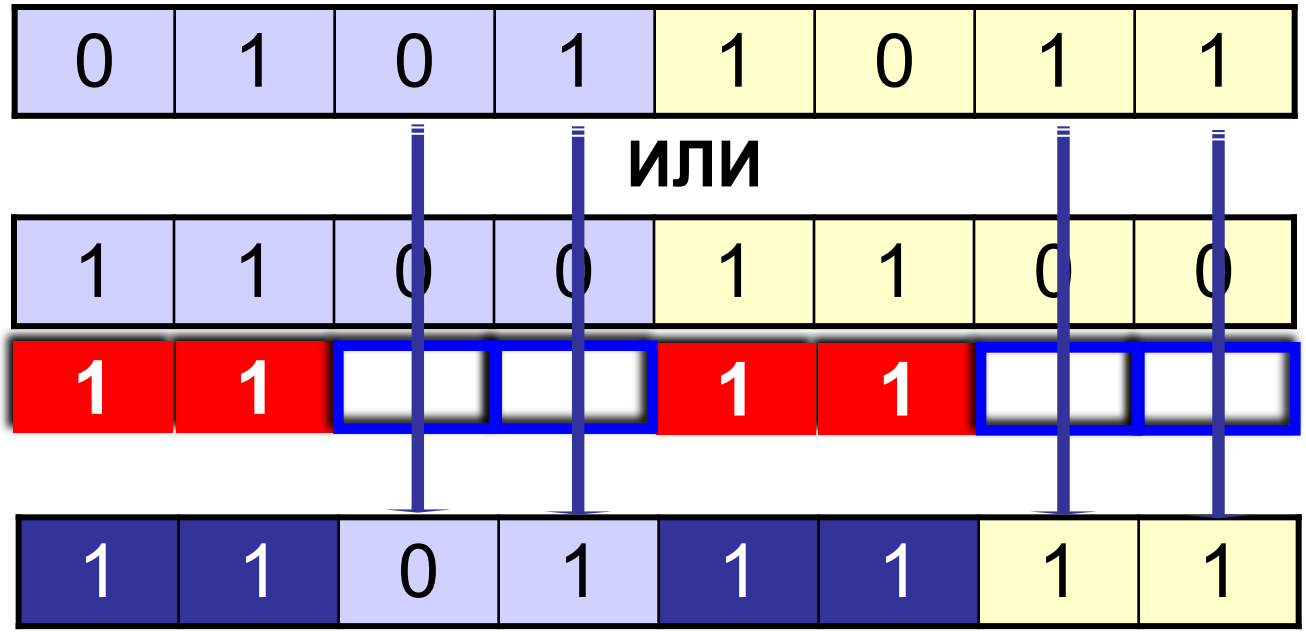
A	B	A или B
0	0	0
0	1	1
1	0	1
1	1	1

Обозначения:

ИЛИ, ∨, | (Си), **or** (Паскаль)

x **ИЛИ** 0 =**x**

x **ИЛИ** 1 =**1**



маска

$$5B_{16} \mid CC_{16} = DF_{16}$$

Операция ИЛИ – установка битов в 1

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

	7	6	5	4	3	2	1	0
маска	0	0	1	1	1	1	0	0

3_{16} C_{16}

Си:

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

Паскаль:

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

Операция «исключающее ИЛИ»

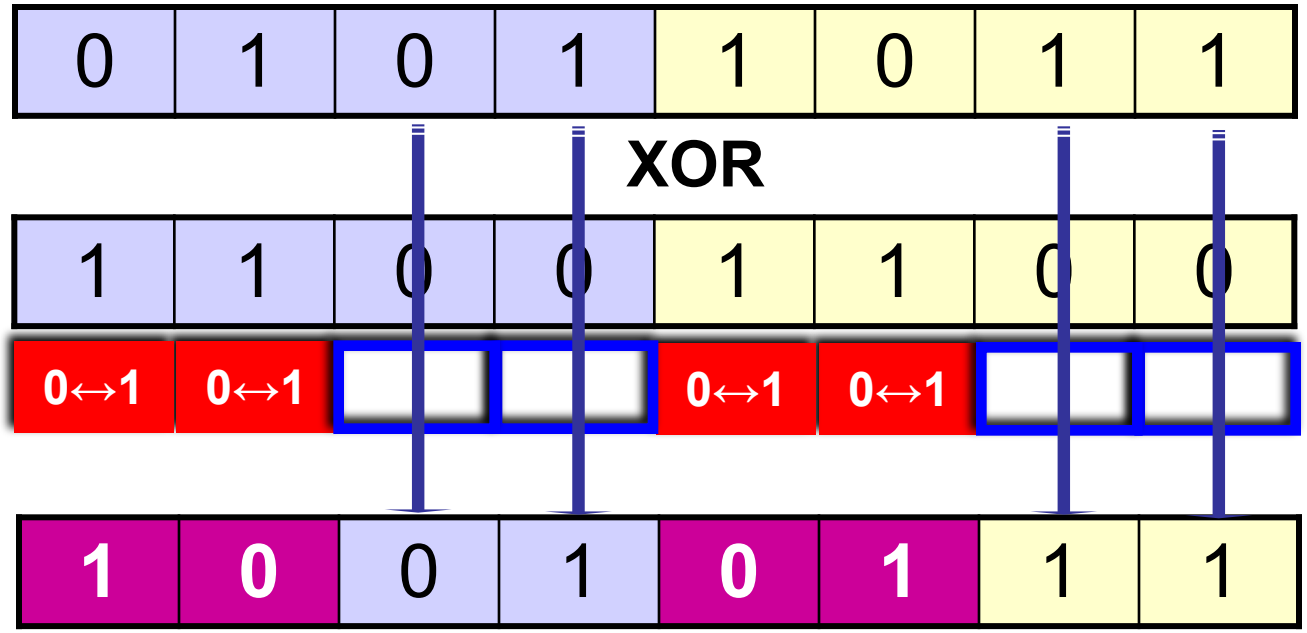
A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

Обозначения:

⊕, **^** (Си), **xor** (Паскаль)

x **XOR** 0 = **x**

x **XOR** 1 = **НЕ x**



маска

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

«Исключающее ИЛИ» – инверсия битов

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

	7	6	5	4	3	2	1	0
маска	0	0	1	1	1	1	0	0

3_{16} C_{16}

Си:

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

Паскаль:

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

«Исключающее ИЛИ» – шифровка


```
(0 xor 0) xor 0 = 0        (0 xor 1) xor 1 = 0
(1 xor 0) xor 0 = 1        (1 xor 1) xor 1 = 1
```

код (шифр)

```
(X xor Y) xor Y = X
```

«Исключающее ИЛИ» – обратимая операция.

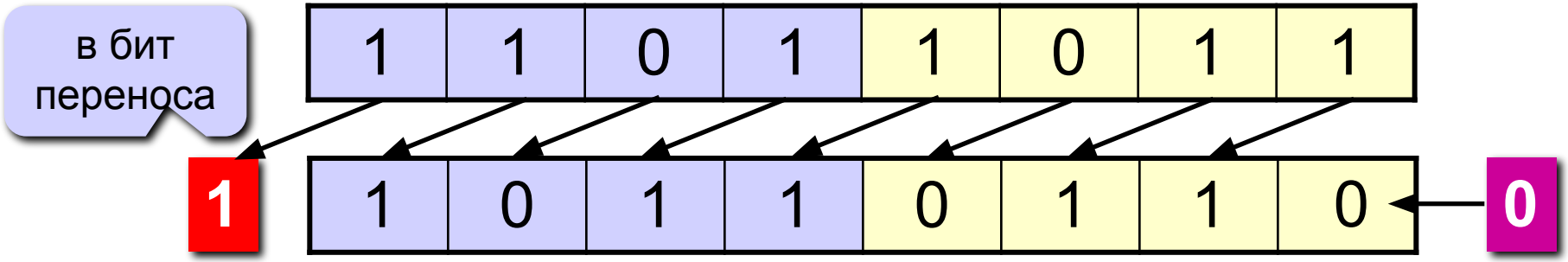
Шифровка:

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

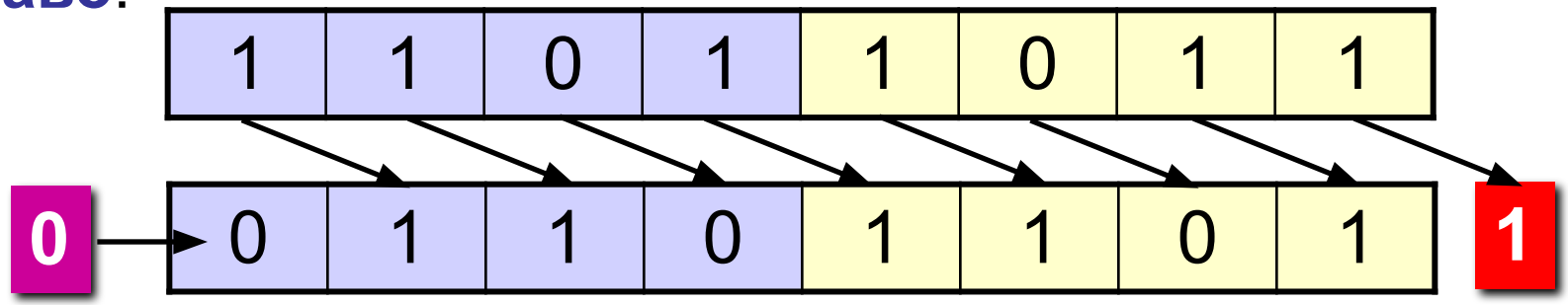
Расшифровка: сделать то же самое с **тем же** шифром.

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 равносилен логический сдвиг влево (вправо)? При каком условии?

сдвиг влево

101101_2 ⇄ 1011010_2

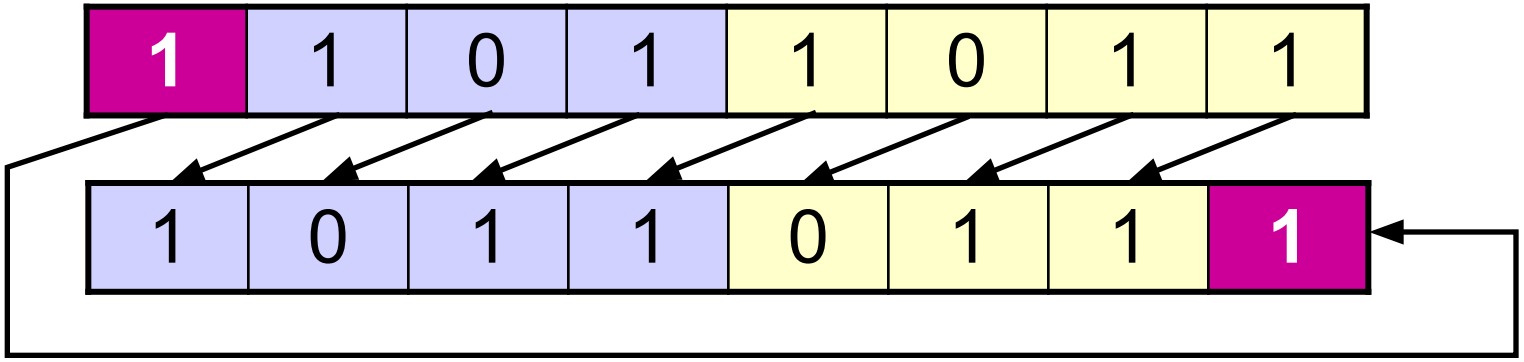
сдвиг вправо

45 **90**

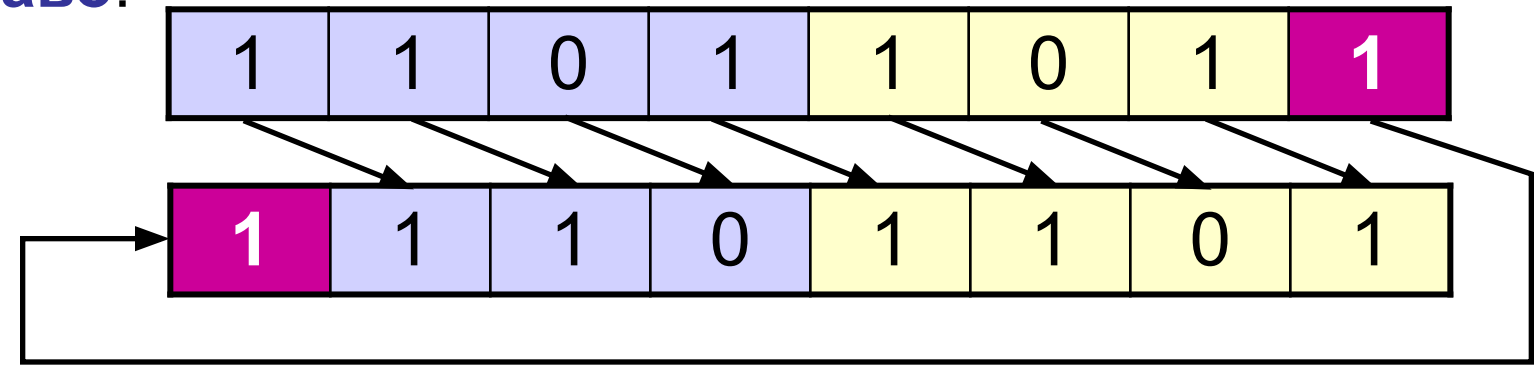
Логический сдвиг влево (вправо) – это быстрый способ **умножения** (деления без остатка) **на 2**.

Циклический сдвиг

Влево:



Вправо:

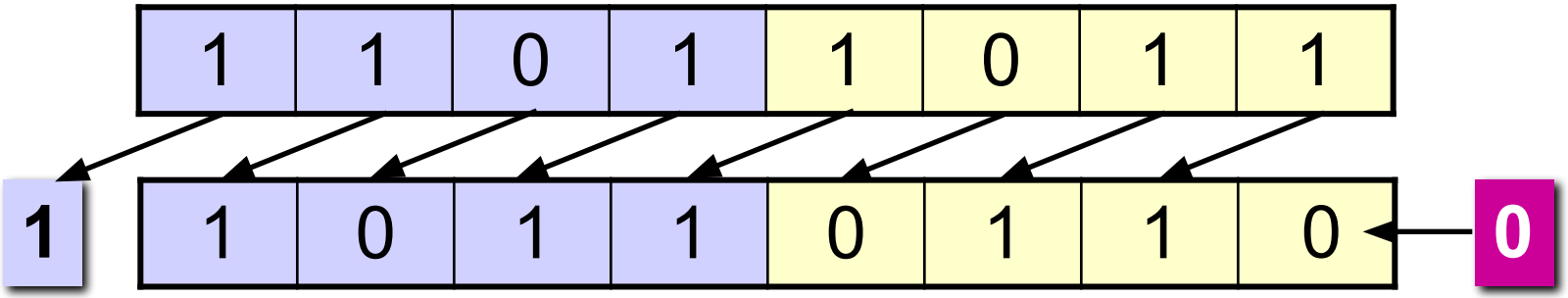


Си, Паскаль: –

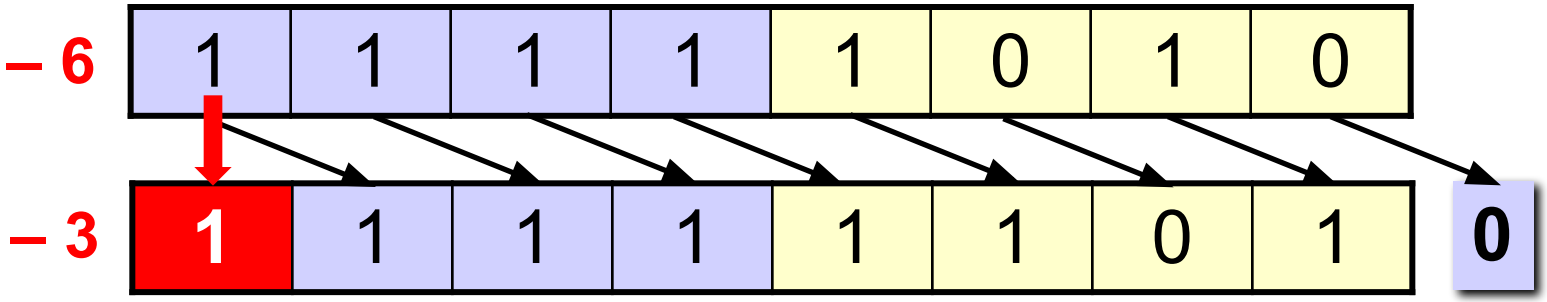
только через Ассемблер

Арифметический сдвиг

Влево (= логическому):



Вправо (знаковый бит не меняется!):



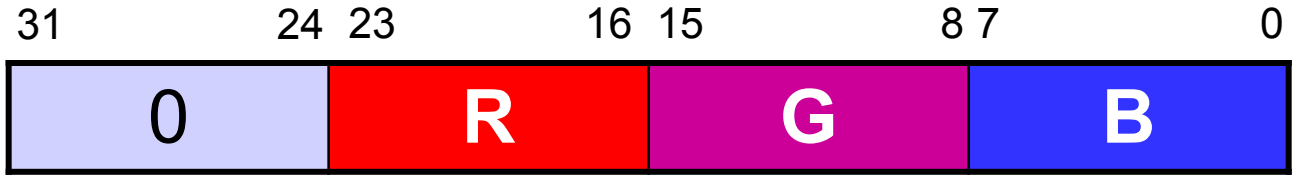
Си:

```
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**: $0000FF00_{16}$
2. Сдвинуть вправо так, чтобы число **G** передвинулось в младший байт.

А надо ли обнулять?

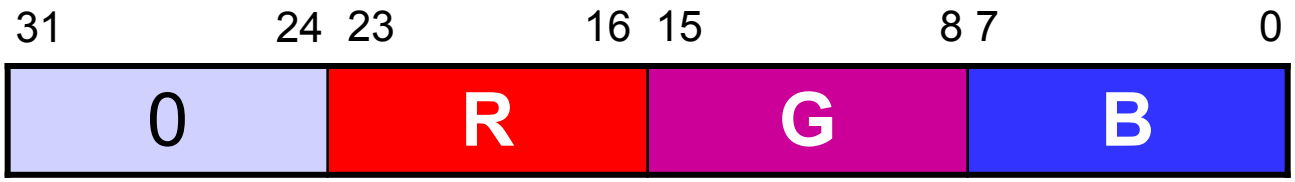
Си:

```
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**: **$000000FF_{16}$**

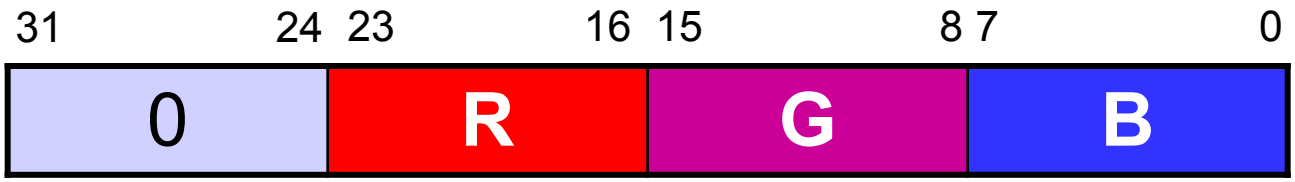
Си:

```
G = (n >> 8) & 0xFF;
```

Паскаль:

```
G := (n shr 8) and $FF;
```

Пример



Си:

```
R =

B =
```

Паскаль:

```
R :=

B :=
```

Компьютер изнутри

Тема 5. Вещественные числа

© К.Ю. Поляков, 2007

Нормализация двоичных чисел

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

s – **знак** (1 или -1)

M – **мантисса**, $M = 0$ **или** $1 \le M < 2$

e – **порядок**

Пример:

знак мантисса порядок

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

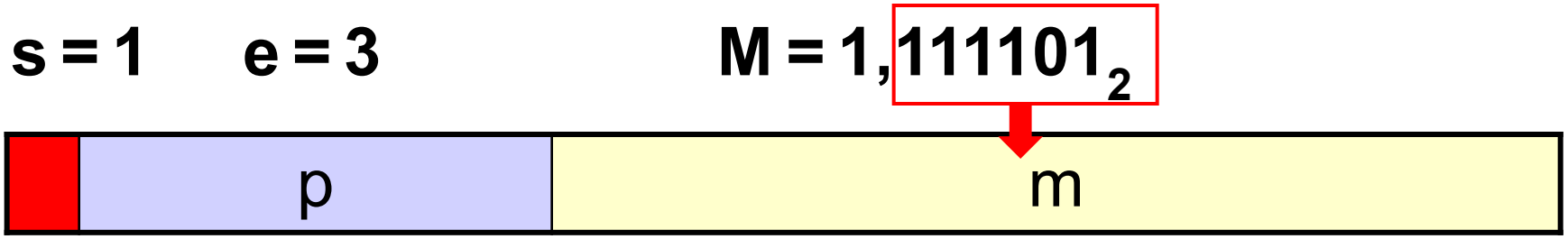
$$3{,}375 =$$

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 **e = 3** **M = 1,111101$_2$**



Порядок со сдвигом:
p = e + E (сдвиг)

Дробная часть мантиссы:
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, **E**	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

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

4 байта = 32 бита

31	30							23	22																							
0	1	0	0	0	0	0	1	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$p = e + 127 = 130$
$= 10000010_2$

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

$3{,}375 =$

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)

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

Арифметические операции

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

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

1. Мантиссы умножаются

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

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

$$\begin{array}{r} 1{,}1\,1_2 \\ \times\quad 1{,}1_2 \\ \hline 1\,1\,1_2 \\ 1\,1\,1_2 \\ \hline 1\,0{,}1\,0\,1_2 \end{array}$$

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. Мантиссы 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)

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

Конец фильма