

Компьютерная арифметика

§ 28. Операции с целыми числами

Сложение и вычитание



Операции с положительными и отрицательными числами выполняются по одинаковым алгоритмам!

$$\begin{array}{r}
 + \quad 5 \\
 - \quad 9 \\
 \hline
 -4
 \end{array}
 \quad
 \begin{array}{r}
 + \quad 0000 \quad 0101 \\
 \quad 1111 \quad 0111 \\
 \hline
 \quad 1111 \quad 1100
 \end{array}$$

←



Вычитание = сложение с дополнительным кодом вычитаемого!

Переполнение

дополнительный
бит

$$\begin{array}{r}
 0 \quad | \quad 01100000 \quad | \quad 96 \\
 + \quad 0 \quad | \quad 00100001 \quad | \quad 33 \\
 \hline
 0 \quad | \quad 10000001 \quad | \quad -127 \\
 S' \quad S \quad | \quad \text{знаковый бит}
 \end{array}$$

$$\begin{array}{r}
 1 \quad | \quad 10100000 \quad | \quad -96 \\
 + \quad 1 \quad | \quad 11011111 \quad | \quad -33 \\
 \hline
 1 \quad | \quad 01111111 \quad | \quad 127 \\
 S' \quad S
 \end{array}$$



Если бит S не совпадает с битом S',
произошло переполнение и результат неверный.

Умножение

$$\begin{array}{r}
 \times \quad 00001001 \quad 9 \\
 \quad 00000101 \quad 5 \\
 \hline
 \quad 00001001 \\
 + \quad 00000000 \\
 \quad 00001001 \\
 \hline
 0000101101 \rightarrow 45
 \end{array}$$

$$\begin{array}{r}
 \times \quad 11110111 \quad -9 \\
 \quad 00000101 \quad 5 \\
 \hline
 \quad 11110111 \\
 + \quad 00000000 \\
 \quad 11110111 \\
 \hline
 10011010011 \rightarrow -45
 \end{array}$$



Умножение выполняется с помощью сложения и сдвига.

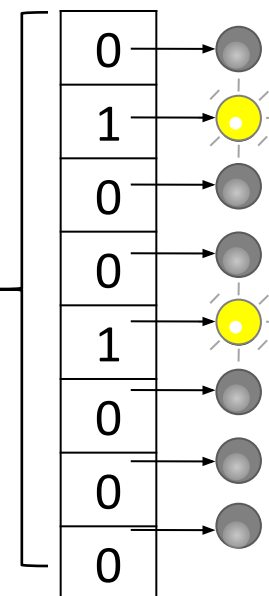
Поразрядные логические операции

Поразрядные операции выполняются с отдельными битами числа и не влияют на остальные.



Сложение – это поразрядная операция?

регистр



Операция «НЕ» (инверсия, not):

R

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

not R

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

Логическая операция «И» (and, &)

данные

маска

D	M	D and M
0	0	0
1	0	0
0	1	0
1	1	1

Маска – константа, которая определяет область применения логической операции к битам многоразрядного числа.

$$AA_{16} \text{ and } 6C_{16} = ?$$

	7	6	5	4	3	2	1	0	
D	1	0	1	0	1	0	1	0	AA_{16}
M	0	1	1	0	1	1	0	0	$6C_{16}$
D and M	0	0	1	0	1	0	0	0	28_{16}



С помощью операции «И» можно сбросить (установить в ноль) биты, для которых маска равна 0!

Логическая операция «ИЛИ» (or, |)

D	M	D or M
0	0	0
1	0	1
0	1	1
1	1	1

$$AA_{16} \text{ or } 6C_{16} = ?$$

	7	6	5	4	3	2	1	0	
D	1	0	1	0	1	0	1	0	AA_{16}
M	0	1	1	0	1	1	0	0	$6C_{16}$
D or M	1	1	1	0	1	1	1	0	EE_{16}



С помощью операции «ИЛИ» можно записать единицу в биты, для которых маска равна 1!

Операция «исключающее ИЛИ» (**xor**, \wedge)

D	M	D xor M
0	0	0
1	0	1
0	1	1
1	1	0

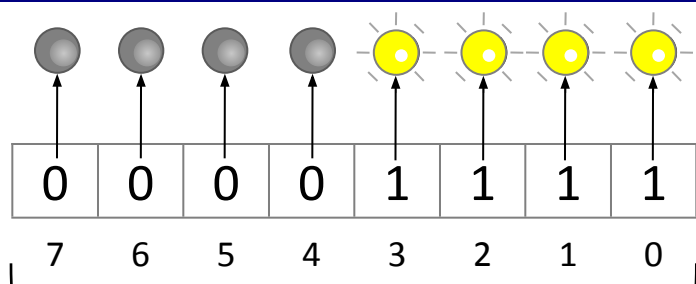
$$AA_{16} \mathbf{xor} 6C_{16} = ?$$

	7	6	5	4	3	2	1	0	
D	1	0	1	0	1	0	1	0	AA_{16}
M	0	1	1	0	1	1	0	0	$6C_{16}$
D xor M	1	1	0	0	0	1	1	0	$C6_{16}$

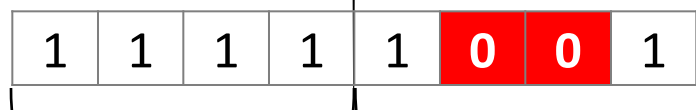
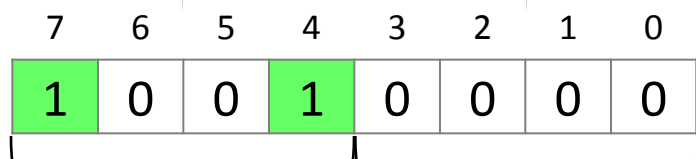
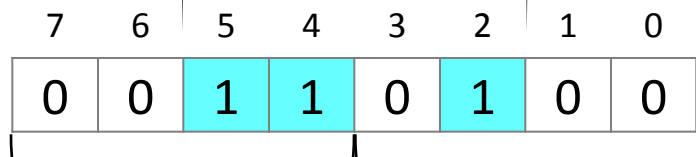


С помощью операции «исключающее ИЛИ» можно инвертировать биты, для которых маска равна 1!

Битовые логические операции (итог)



R

F₁₆9₁₆9₁₆0₁₆3₁₆4₁₆

1) отключить лампочки 2 и 1, не трогая остальные

$$R = R \text{ and } F9_{16}$$

2) включить лампочки 7 и 4

$$R = R \text{ or } 90_{16}$$

3) изменить состояние лампочек 5, 4 и 2

$$R = R \text{ xor } 34_{16}$$

Шифрование с помощью xor

Идея: $(A \text{ xor } B) \text{ xor } B = A$



Операция «исключающее ИЛИ» *обратима*, то есть ее повторное применение восстанавливает исходное значение!

Текст: $2 * 2 = 4$

Коды символов:

$$'2' = 32_{16} = 00110010_2$$

$$'*' = 2A_{16} = 00101010_2$$

$$'=' = 3D_{16} = 00111101_2$$

$$'4' = 34_{16} = 00110100_2$$

Шифрование с помощью **xor**

Маска: $23 = 17_{16} = 00010111_2$

$$'2' \rightarrow 32_{16} \text{ xor } 17_{16} = 25_{16} \rightarrow '\%'$$

$$'*' \rightarrow 2A_{16} \text{ xor } 17_{16} = 3D_{16} \rightarrow '='$$

$$'=' \rightarrow 3D_{16} \text{ xor } 17_{16} = 2A_{16} \rightarrow '*'$$

$$'4' \rightarrow 34_{16} \text{ xor } 17_{16} = 23_{16} \rightarrow '\#'$$

Исходный текст: $2*2=4$

Зашифрованный текст: $\%=\%*\#$

Расшифровка:

$$'%' \rightarrow 25_{16} \text{ xor } 17_{16} = 32_{16} \rightarrow '2'$$

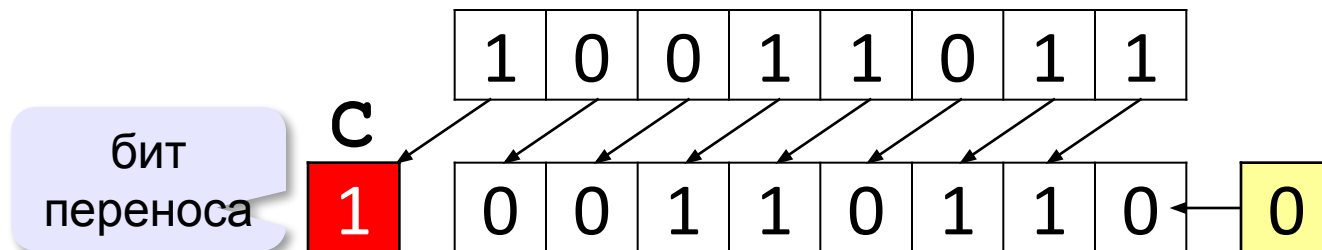
$$'=' \rightarrow 3D_{16} \text{ xor } 17_{16} = 2A_{16} \rightarrow '*'$$

$$'*' \rightarrow 2A_{16} \text{ xor } 17_{16} = 3D_{16} \rightarrow '='$$

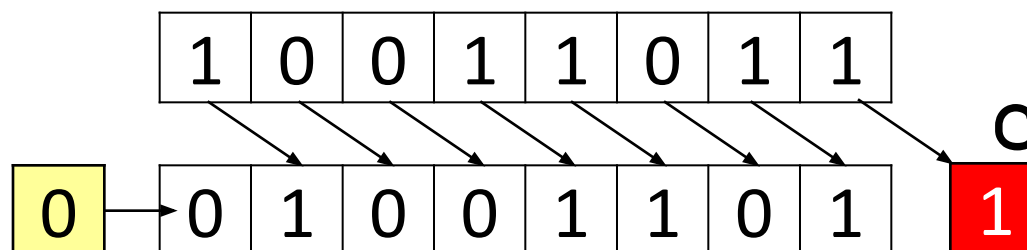
$$'\#' \rightarrow 23_{16} \text{ xor } 17_{16} = 34_{16} \rightarrow '4'$$

Логический сдвиг

Влево:



Вправо:



C, C++, Python :

```
N = N << 1;
N = N >> 1;
```

Паскаль:

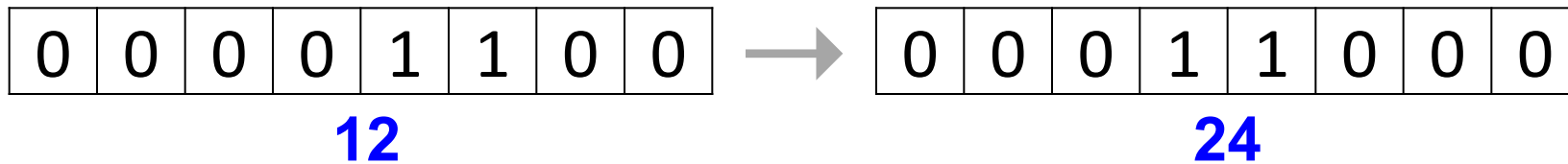
shift left

```
N := N shl 1;
N := N shr 1;
```

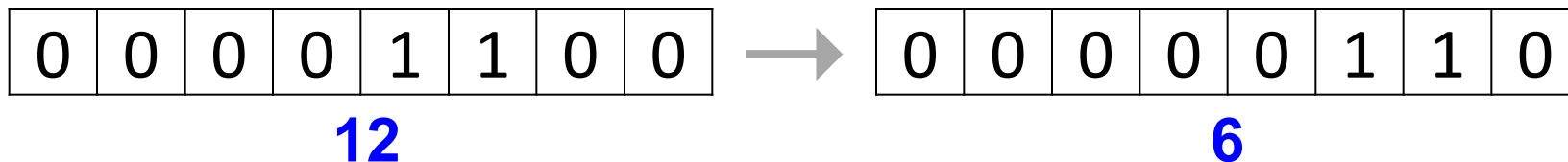
shift right

Логический сдвиг

Влево:



Вправо:



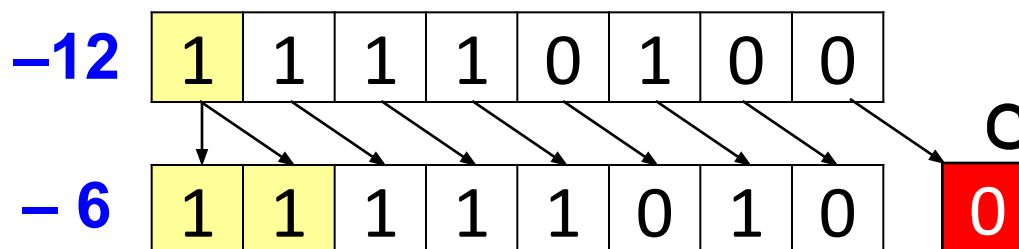
Если число нечётное?

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



Если число отрицательное?

Арифметический сдвиг (вправо)



Если число нечётное?

Примеры:

20 → 10

-20 → -10

15 → 7

-15 → -8

11 → 5

-11 → -6

3 → 1

-3 → -2

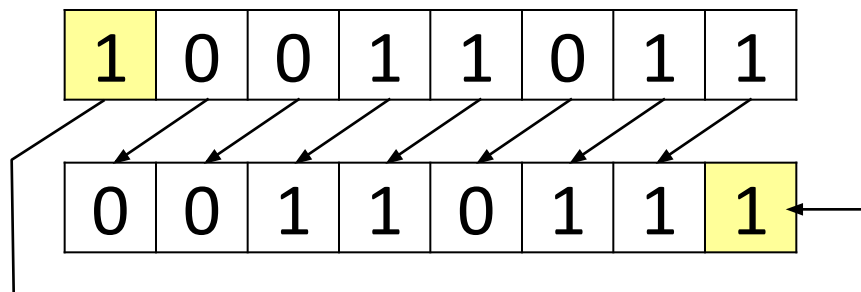
1 → 0

-1 → -1

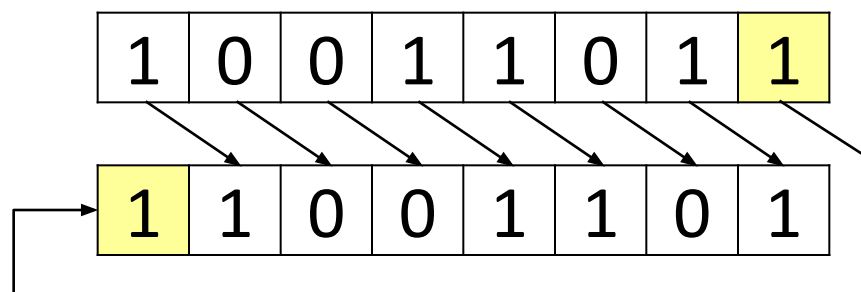
Арифметический сдвиг вправо – деление на 2 нацело с округлением «вниз» (к ближайшему меньшему целому).

Циклический сдвиг

Влево:



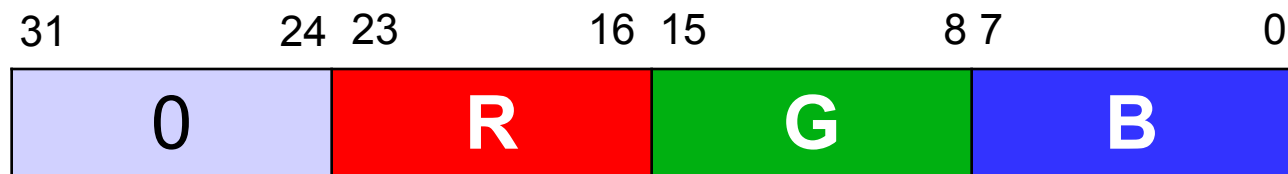
Вправо:



Циклический сдвиг предназначен для последовательного просмотра битов без потери данных.

Пример

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



Записать в переменные R, G, B составляющие цвета.

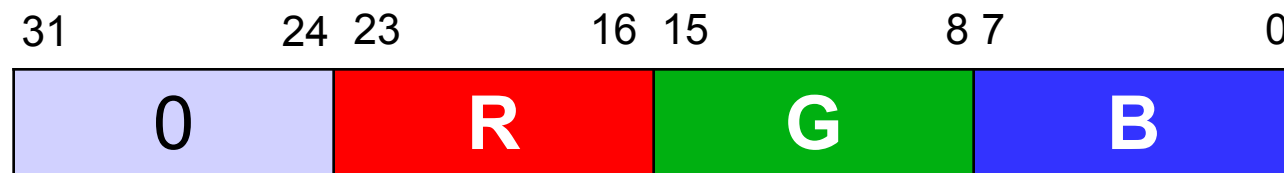
Вариант 1:

1. Обнулить все биты, кроме **G**.
Маска для выделения **G**: **0000FF00**₁₆
2. Сдвинуть вправо так, чтобы число **G** передвинулось в младший байт.

C, C++, Python: `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

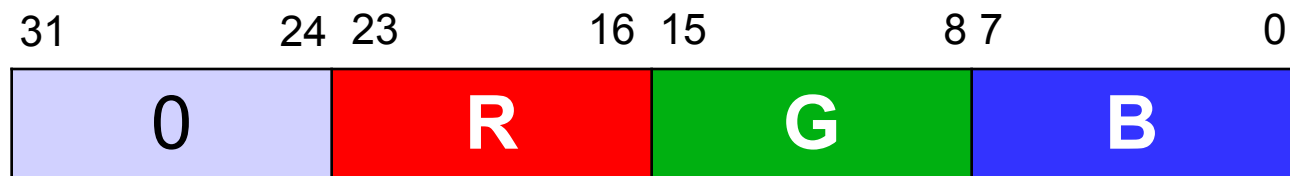
C, C++, Python: `G = (N >> 8) & 0xFF;`

Паскаль: `G := (N shr 8) and $FF;`



Как решить, используя только сдвиги?

Пример



Си: R =

B =

Паскаль: R :=

B :=

Конец фильма

ПОЛЯКОВ Константин Юрьевич

д.т.н., учитель информатики

ГБОУ СОШ № 163, г. Санкт-Петербург

kpolyakov@mail.ru

ЕРЕМИН Евгений Александрович

к.ф.-м.н., доцент кафедры мультимедийной

дидактики и ИТО ПГГПУ, г. Пермь

eremin@pspu.ac.ru

Источники иллюстраций

1. авторские материалы