ТЕМА: Преобразования чисел в позиционных системах счисления

Позиционные системы счисления

Двоичная	Восьмеричная	Десятичная	Шестнадцатеричная
0 1	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9 A (дес. значение 10) B (дес. значение 11) C (дес. значение 12) D (дес. значение 13) E (дес. значение 14) F (дес. значение 15)
$q = 2$			$q = 16$
	$q = 8$		
		$q = 10$	

Позиционные системы счисления

$$A_{(q)} = a_n q^n + a_{n-1} q^{n-1} + \ldots + a_1 q^1 + a_0 q^0 + a_{-1} q^{-1} + a_{-2} q^{-2} + \ldots + a_{-m} q^{-m} = \sum_{i=-m}^{n} a_i q^i$$

,

$A_{(q)}$ - ***число системы q*** $A_{(q)} = a_n a_{n-1} \ldots a_1 a_0 a_{-1} a_{-2} \ldots a_{-m}$ q^i – *вес разряда*

$$A_{10} = 349.17 = 3 \cdot 10^2 + 4 \cdot 10^1 + 9 \cdot 10^0 + 1 \cdot 10^{-1} + 7 \cdot 10^{-2}$$

a_2 a_1 a_0 a_{-1} a_{-2}

q=2 **– двоичная;** ***0; 1.***
q=8 **– восьмеричная;** ***0; 1; 2; 3; 4; 5; 6; 7.***
q= 10 **– десятичная;** ***0; 1; 2; 3; 4; 5; 6; 7; 8; 9;*** **(см. пример выше).**
q= 16 **– шестнадцатеричная** ***0; 1; 2; 3; 4; 5; 6; 7; 8; 9; A; B; C; D; E; F.***

Десятичное число	Двоичное	Восьмеричное	Шестнадцатеричное
0	0	0	0
1	1	1	1
2	10*	2	2
3	11	3	3
4	100*	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000*	10*	8
9	1001	11	9
10*	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000*	20	10*
17	10001	21	11
18	10010	22	12

**** Означает перенос в старший разряд***

$$A_{(2)} = 1101.001_{(2)} \quad A_{(2)} \to A_{(10)} = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} = 13.125$$

$a_3 \quad a_2 \quad a_1 \quad a_0 \quad a_{-1} \quad a_{-2} \quad a_{-3}$

Перевод из одной системы в другую

$(101101)_2 = (?)_8$ $\quad (101 \quad 101)_2$ – *число из триад* $\quad (101101)_2 = (55)_8$

$(101101)_2 = (0010)(1101) = (2D)_{16}$

тетрады

$A_{(8)} \blacktriangleright A_{(2)}$

$(6531)_8 =$ 6 5 3 1

(110) (101) (011) (001) ? $A_{(2)} =$ 110101011001

$(6F3D)_{16}$ = 6 F 3 D = $(0110111100111101)_2$

(0110) (1111) (0011) (1101)

тетрады

$(7614)_8$ → $(?)_{10}$

$(7614)_8 = 7 \cdot 8^3 + 6 \cdot 8^2 + 1 \cdot 8^1 + 4 \cdot 8^1 = 3584 + 384 + 8 + 4 = (3980)_{10}$

(512) (64) (8) (1) – позиционные значения

(7) (6) (1) (4) - восьмеричная цифра в позиции исходного числа

?

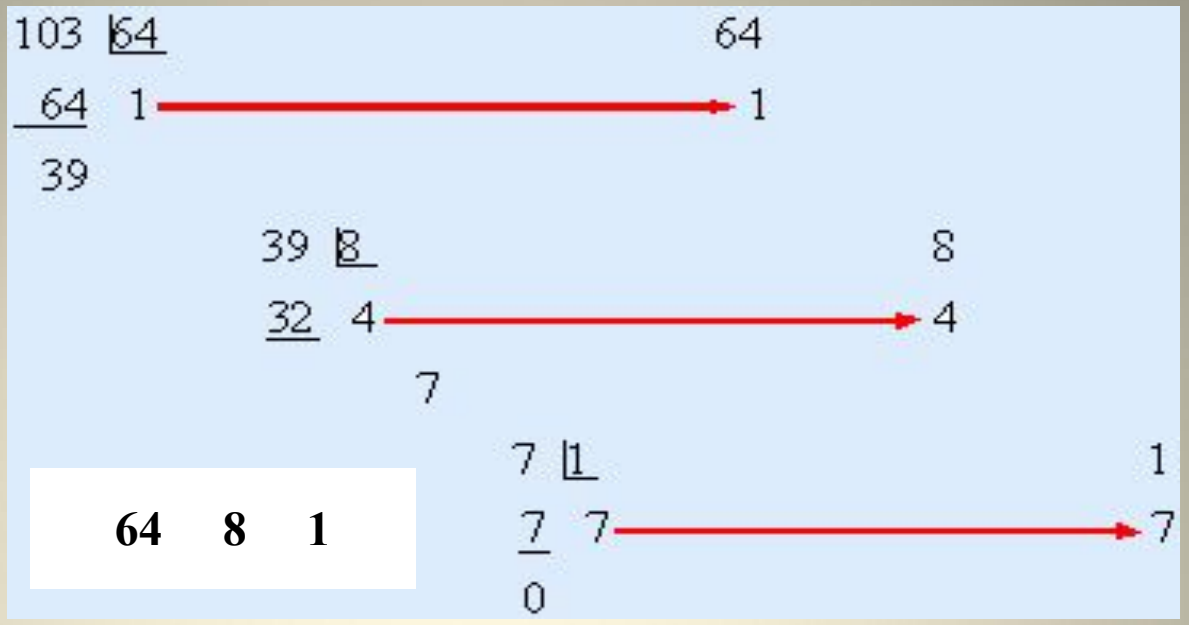
$(ADBB)_{16} = A \cdot 16^3 + D \cdot 16^2 + 3 \cdot 16^1 + B \cdot 16^0 = (44347)_{10}$

(10) · (4096) (13) · (256) (3) · (16) (11) · (1)

40960 3328 48 11 – после сложения

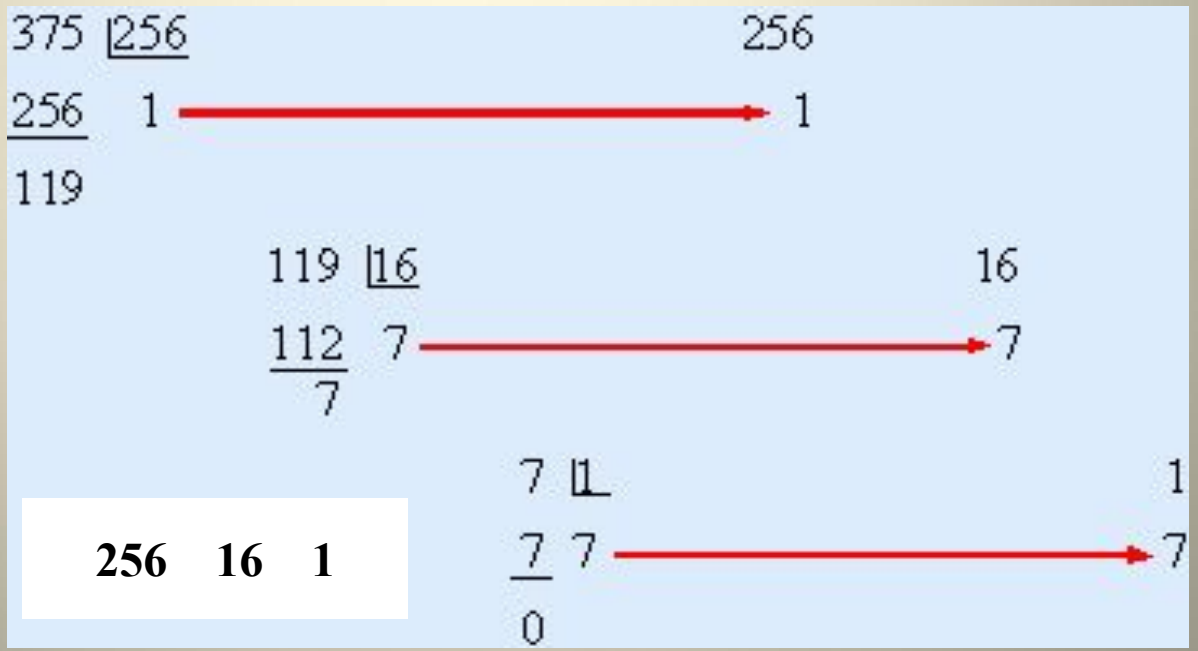
этих чисел получаем результат преобразования $(44347)_{10}$

$(103)_{10}$



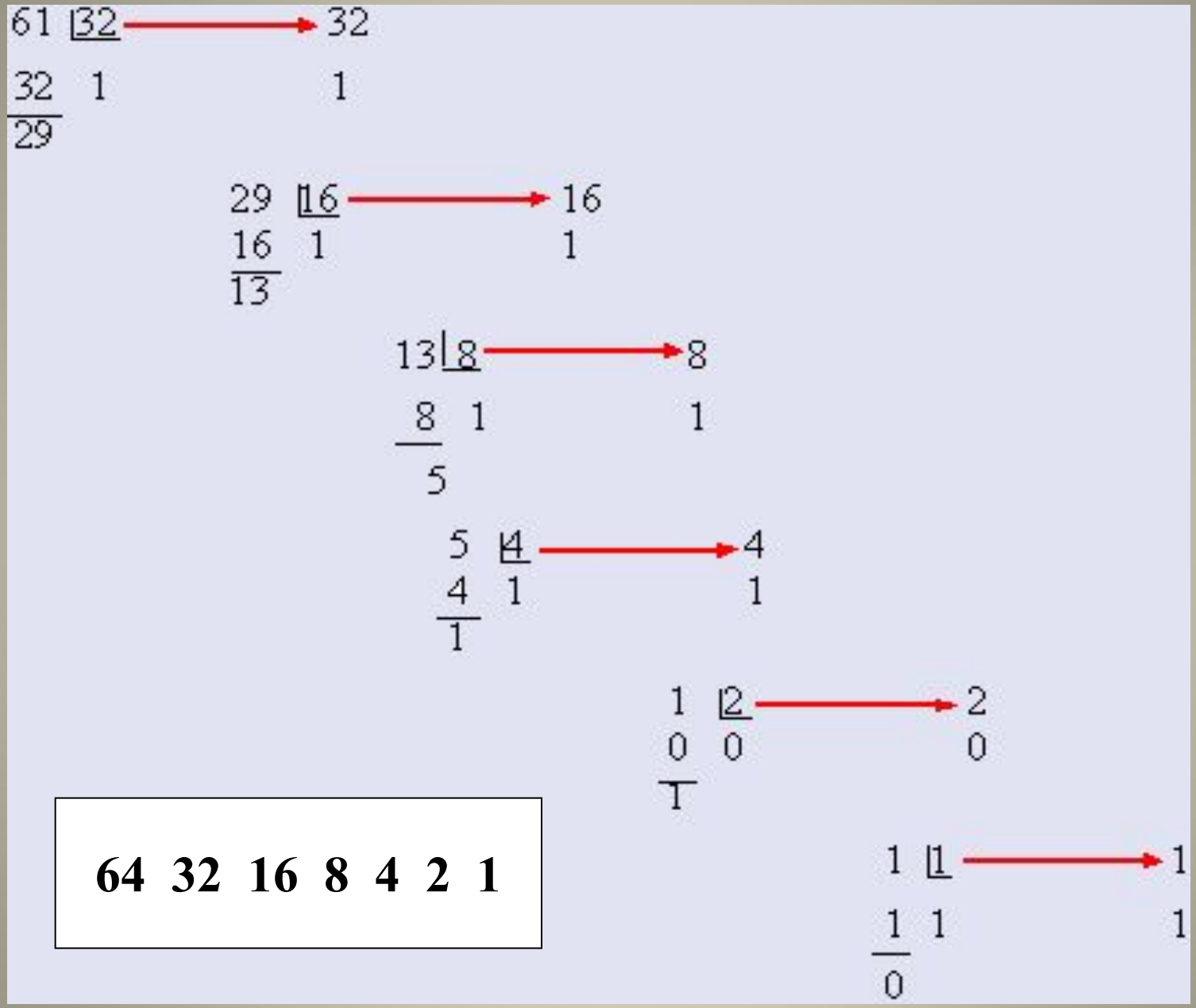
$(147)_{8}$

$(375)_{10}$



$(177)_{16}$

$(61)_{10}$



$(111101)_2$

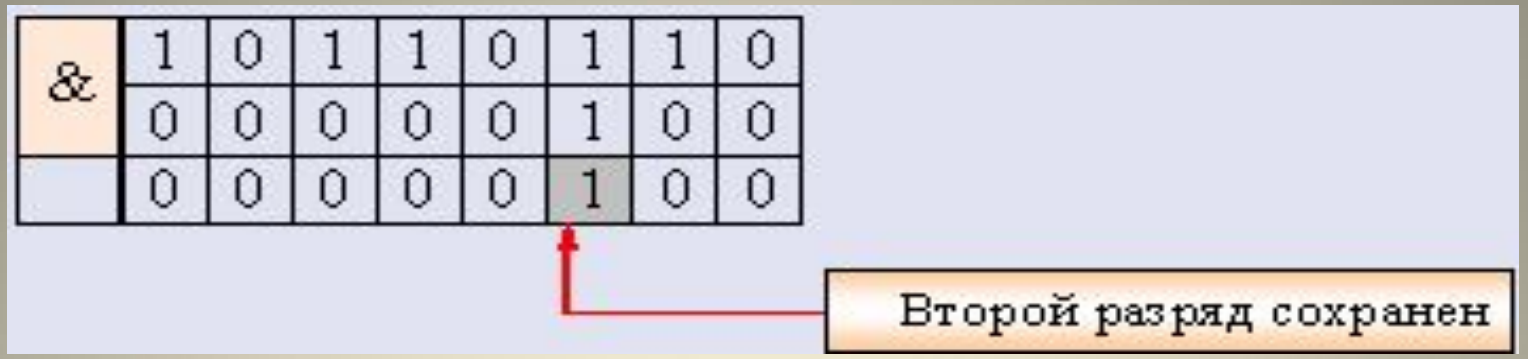
Логические поразрядные операции

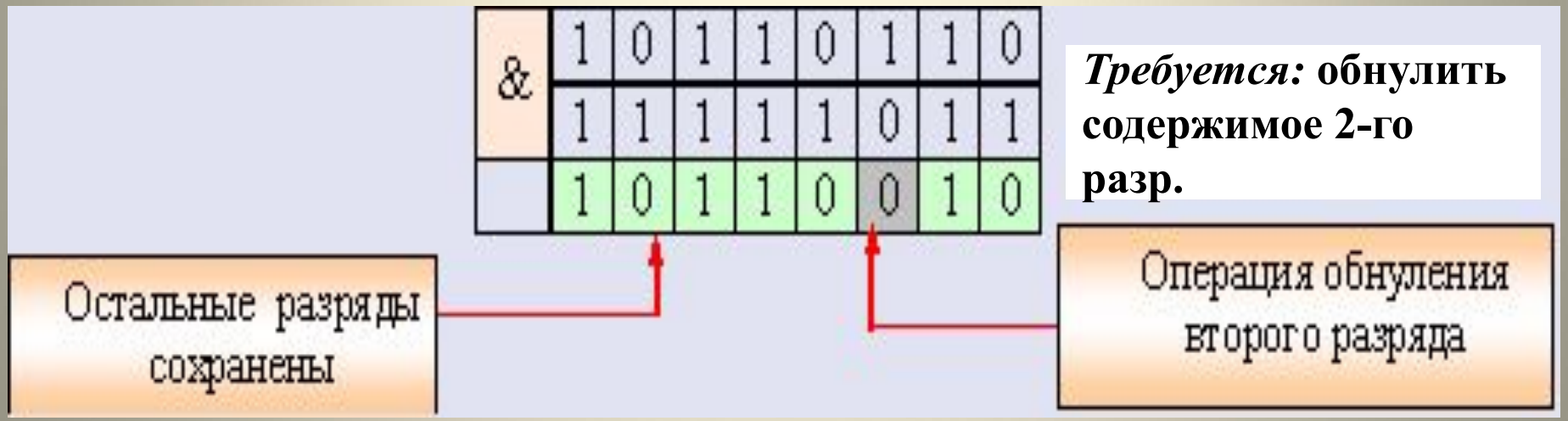
Знаки операций и их тип	Обозначение в алгебраической символике
& – бинарная	И
| – бинарная	ИЛИ
^ – бинарная	Исключающее ИЛИ
<< – бинарная	Сдвиг влево
>> – бинарная	Сдвиг вправо
~ – унарная	Дополнение до единицы

приоритеты: 1 – *( )*, 2 – ~, 3 – <<, 4 – >>, 5 – &, 6 – ^, 7 – |

Примеры:

Дано: **байт 10110101**
Требуется: **заменить на "1"**
только 1-ый разряд


&
1 0 1 1 0 1 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
Второй разряд сохранен


&
1 0 1 1 0 1 1 0
1 1 1 1 1 0 1 1
1 0 1 1 0 0 1 0
Требуется: обнулить содержимое 2-го разр.
Остальные разряды сохранены
Операция обнуления второго разряда

Дано: **8-разрядное число (байт) 00101101.**
Требуется: **записать число в дополнительном коде.**

~	0	0	1	0	1	1	0	1	?	+	1	1	0	1	0	0	1	0	промежуточный результат
											0	0	0	0	0	0	0	1	единица младшего разряда
											1	1	0	1	0	0	1	1	число в дополнительном коде

Дано: **8-разрядное число (байт) 01011011 .**
Требуется: **инвертировать только 4 младших разряда**

.

^	0	1	0	1	1	0	1	1	
	0	0	0	0	1	1	1	1	– число, с помощью которого мы хотим инвертировать
	0	1	0	1	0	1	0	0	– результат

Инвертированы

Сдвиговые операции:

операнд « выражение
выражение » операнд

```
# include <stdio.h>
  int main (void)
  {
    int x=1;              // x □ 00000001
    printf ("\n%d, %d, %d, %d, %d, %d, %d", x<<1,
              x<<2,  x<<3, x<<0,  x<<30,  x<<-32768,
              x<<-32767,  x<<-32766);
    return 0;
  }//?будет напечатано 2, 4, 8, 1, 0, 1, 2, 4
```

“отрицательные” значения выражения или значения, равные или превышающие число битов в операнде, в общем случае недопустимы и **дают неопределенные результаты**.

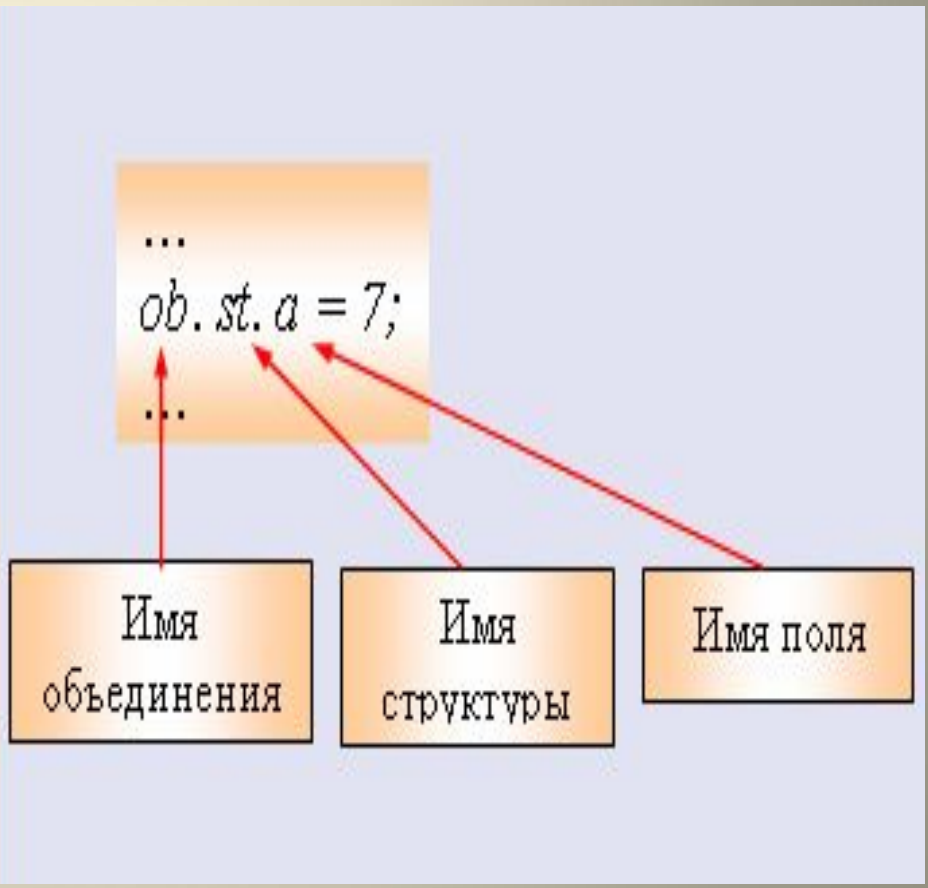
Проверить самостоятельно!

```
// ?      11111101 11101000
# include <stdio.h>
  int main ( )
  {
    unsigned a;// это беззнаковое целое число
    void  ecran_bit (unsigned);   // Прототип функции.
    printf ("Введите беззнаковое целое число:");
    scanf ("%u", &a);
    ecran_bit (a);   // Применяет операцию & к переменным b и c,
                     // где c = Maska (определена в неглавной функ
    return 0;
  }
    void ecran_bit (unsigned b)
    {
      unsigned i, Maska = 1<<15; //и выражение и операнд числа
      printf ("%7u = ", b);
      for (i = 1; i<=16; i++)  {
        putchar (b & Maska ? '1':'0');   // Для текущего кратного слова бита.
        b<<=1;
        if (i%8 == 0) // зачем делить на 8          (?)
        putchar (' ');   // функция возвращает символ.
      }
        putchar ('\n');   // В случае ошибки возвращает EOF.
    }   // End ecran_bit.
```

! исследовать алгоритм и дать (письменно) анализ работы программы при a=65000

Пример битовых полей в структуре, входящей в объединение

```
...
 union
 {
  unsigned char ch;
  struct
  {
    unsigned int a: 5;   // Младший бит.
    unsigned int b: 3;   // Старший бит.
  } st;
 } ob;
```



Пример программы с использованием объединений, структур и битовых полей

```
// Побитовая печать содержимого регистра данных.
# include <stdio.h>
  void main (void)
  {
    unsigned char k;   // Будет равен битовому коду.
    int m, n;   // Они делятся на 16 для формирования битовых полей.
   void binar (unsigned char);
      /* В функцию входит байт и там происходит обратное преобразование –
      побитово расшифровывается за счет обращения к отдельным полям. */
   unsigned char cod (int, int);
 /* В функции производится запись данных в битовые поля, а результат
  возвращается из того же объединения в виде одного байта. */
    printf ("\nm =");
    scanf ("%d", &m);
    printf ("\nn =");
    scanf ("%d", &n);
    k = cod (m, n);
    printf ("cod = %u", k);
    binar (k);
  }
```

В лекции 10 эта программа уже рассматривалась

```
unsigned char cod (int a, int b)    // a,b-для формирования
                                    //битовых полей
 {
   union
   {
     unsigned char z; //будет равен битовому коду un.z
     struct
     {
       unsigned int x: 4;   // Младшие биты
       unsigned int y: 4;   // Старшие биты
     }  hh;
   }  un;
     un. hh. x = a%16;
     un. hh. y = b%16;   // Упаковка в один байт.
     return un. z;
 }   // End cod.
```

```
void binar (unsigned char ch)
        {
          union
          {
             unsigned char ss;
            struct
            {
              unsigned  a0: 1;   unsigned  a1: 1;
              unsigned  a2: 1;   unsigned  a3: 1;
              unsigned  a4: 1;   unsigned  a5: 1;
              unsigned  a6: 1;   unsigned  a7: 1;
            } byte;
          } cod;
            cod. ss = ch;
            printf ("\nНомера битов:     7 6 5 4 3 2 1 0");
            printf ("\nЗначения битов: %d %d %d %d %d %d %d %d",
            cod. byte. a7, cod. byte. a6, cod. byte. a5, cod. byte. a4,
            cod. byte. a3, cod. byte. a2, cod. byte. a1, cod. byte. a0);
}
```
// Печатаем, как предписано заголовком, т.е. со старшего бита.

Пример битовых операций в фрагменте программы управления роботом

```
…
void POZIC (int N, int S) //ниже фрагмент функции
                          //реализующей управление двигателем
{
int Esc, S0, S1;
double tact;      // задается переменная для хранения числа импульсов
DRV0=0x00;   //переменные, отождествленные с внутренними регистрами БУ УРТК
DRV1=0x80;
      if (S<0)
      {
        write (0x00, 0x11);  //функция записи байта в регистр управления
        write (0x0A, DRV1);
        write (0x0A, DRV0=DRV0 | (0x01<<N*2));
      } // N, S – вспомогательные переменные
      if (S>0)
      {
        write (0x00, 0x11);
        write (0x0A, DRV1);
        write (0x0A, DRV0=DRV0 | (0x01<<(N*2+1)));
      }  while (Esc!=27)
        {
        …
```