Numbers and Bits

PC FAF 2020

Integer numbers representation

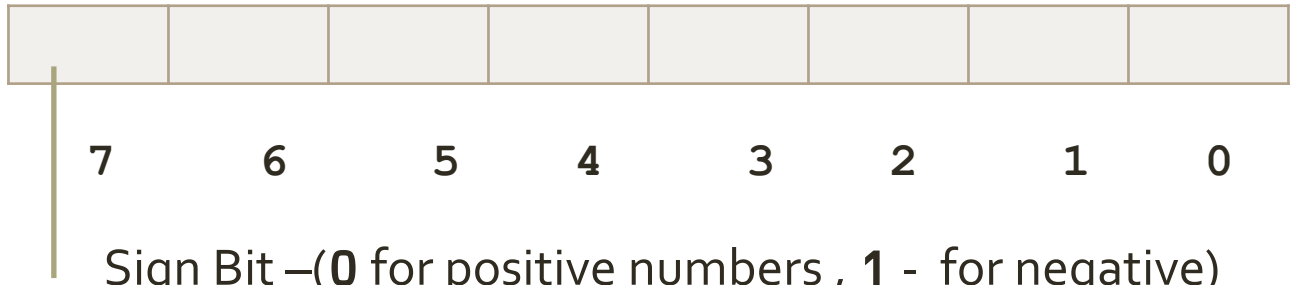
(2)

Integers are represented in computer memory in binary code, using sequences of **0** and **1**.

Usually, the length of these sequences is divisible by 8:

`8 / 16 / 32 bits` …

We will analyze the simplest case of representation, on 8 bits:



Representation types

(Codes)

Direct code: The number is encoded in base 2, is placed in the bits indexed from **0** to **6** (right aligned), free bits are setted to **0**. Sign bit is setted according to number sign.

Examples:

55

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

-36

1	0	1	0	0	1	0	0

Inverse code

Inverse code: is obtained from direct code. For positive numbers is the same with direct code, for negatives each bit is inversed, excepting sign bit.

Examples:

55

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

-36

Cod direct

1	0	1	0	0	1	0	0

Cod invers

1	1	0	1	1	0	1	1

Complementary code

Complementary code: obtained from inverse code for negative numbers. In order to obtain the complementary code just add a binary 1 to inverse code. For pozitive numbers is the same with direct code.

Example:

55

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

-36

Direct code	1	0	1	0	0	1	0	0
Inverse code	1	1	0	1	1	0	1	1
Comp-tary:	1	1	0	1	1	1	0	0

Bit operators

Negație (~)

Initial value	operator	Rezult
27	~	-28
binary		
00011011		11100100

```
00011011
11100100
```

```c
#include <stdio.h>

int a;

int main()
{
    a = 28;
    printf("\n initial value %d", a);
    a = ~a;
    printf("\n value after ~ operation %d", a)

    return 0;
}
```

```
C:\Users\CTI UST\Desktop\Proiect 22\surse\bits_001.exe

 initial value 28
 value after ~ operation -29
--------------------------------
Process exited after 0.02645 seconds with
Press any key to continue . . .
```

C example

Conjunction

&

Value X	operator	Value Y	Rezult X & Y
27	&	-28	0
	binary		
00011011		11100100	00000000

00011011
11100100
00000000

ValueX	operator	Value Y	Rezult X & Y
12	&	20	4
	binary		
00001100		00010100	00000100

00001100
00010100
00000100

C example

bits_004.c | bits_003.c | [*] bits_002.c

```
 7      int k;
 8      if (p == 1 ) printf("%d", n);
 9      else
10      {
11          k = n % 2; tentobin(n / 2, p-1); printf("%d",k);
12      }
13  }
14  int main()
15  {
16      a = 35; c = 74; b = 47;
17      printf("\n initial values %d\t", a); tentobin(a, 8);
18      printf("\n initial values %d\t", b); tentobin(b, 8);
19      printf("\n initial values %d\t", c); tentobin(c, 8);
20      res = a & c;
21      printf("\n value %d & %d = %d\n", a, c, res);
22          int __cdecl printf (const char * __restrict__ _Format, ...)
23      tentobin(a,8); printf(" & "); tentobin(c,8);
24      printf(" = "); tentobin(res,8);
25      res = b & c;
26      printf("\n value %d & %d = %d\n", b, c, res);
27      tentobin(res,8); printf("\n");
28      tentobin(b,8); printf(" & "); tentobin(c,8);
29      printf(" = "); tentobin(res,8);
30      return 0;
31  }
```

C example

```
 initial values 35        00100011
 initial values 47        00101111
 initial values 74        01001010
 value 35 & 74 = 2
00000010
00100011 & 01001010 = 00000010
 value 47 & 74 = 10
00001010
00101111 & 01001010 = 00001010
--------------------------------
Process exited after 0.02375 seconds w
Press any key to continue . . .
```

Disjunction

|

Value X	operator	Value Y	Rezult X & Y
27	|	-28	0
	binary		
00011011		11100100	11111111

00011011
11100100
11111111

Value X	operator	Value Y	Rezult X & Y
12	|	20	4
	binary		
00001100		00010100	00011100

00001100
00010100
00011100

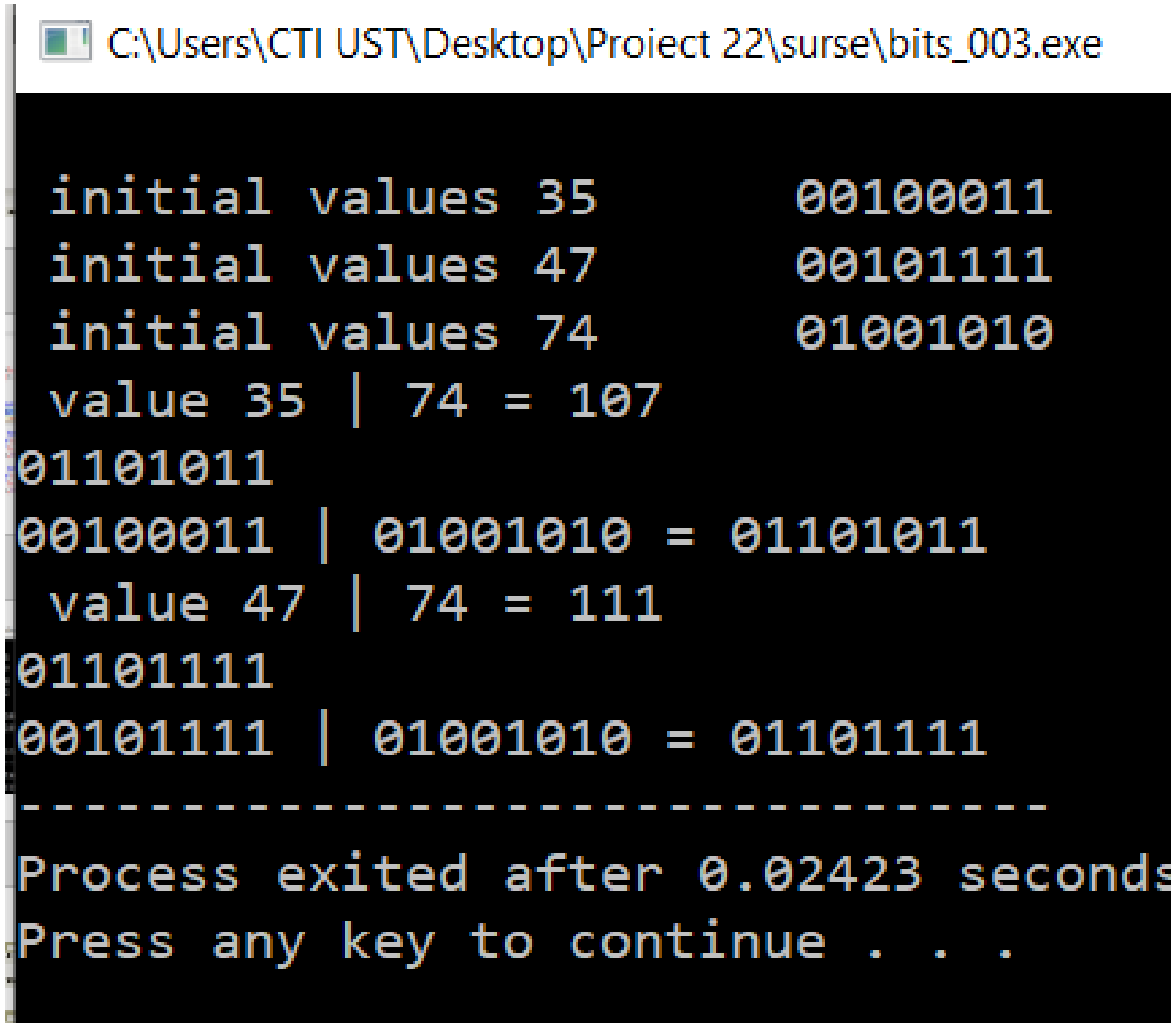
C example

bits_004.c [*] bits_003.c

```c
#include <stdio.h>
int a, b, c, res;
void tentobin(int n, int p)
{   int k;
    if (p == 1 ) printf("%d", n);
    else { k = n % 2; tentobin(n / 2, p-1); printf("%d",k); }
}
int main()
{   a = 35; c = 74; b = 47;
    printf("\n initial values %d\t", a); tentobin(a, 8);
    printf("\n initial values %d\t", b); tentobin(b, 8);
    printf("\n initial values %d\t", c); tentobin(c, 8);
    res = a | c;
    printf("\n value %d | %d = %d\n", a, c, res);
    tentobin(res,8); printf("\n");
    tentobin(a,8); printf(" | "); tentobin(c,8);
    printf(" = "); tentobin(res,8);
    res = b | c;
    printf("\n value %d | %d = %d\n", b, c, res);
    tentobin(res,8); printf("\n");
    tentobin(b,8); printf(" | "); tentobin(c,8);
    printf(" = "); tentobin(res,8)
    return 0;
}
```

1/ 2 void tentobin (int n, int p)

C example


C:\Users\CTI UST\Desktop\Proiect 22\surse\bits_003.exe
initial values 35 00100011
initial values 47 00101111
initial values 74 01001010
value 35 | 74 = 107
01101011
00100011 | 01001010 = 01101011
value 47 | 74 = 111
01101111
00101111 | 01001010 = 01101111
Process exited after 0.02423 seconds
Press any key to continue . . .

XOR

^

Valoare X	operator	Valoare Y	Rezultat X & Y
27	^	-28	
	binar		
00011011		11100100	11111111

00011011
11100100
11111111

Valoare X	operator	Valoare Y	Rezultat X & Y
12	^	20	4
	binar		
00001100		00010100	00011000

00001100
00010100
00011000

C example

```
 initial values 35        00100011
 initial values 47        00101111
 initial values 74        01001010
 value 35 | 74 = 105
01101001
00100011 ^ 01001010 = 01101001
 value 47 ^ 74 = 111
01101111
00101111 ^ 01001010 = 01101111
--------------------------------
Process exited after 0.02459 seconds w
Press any key to continue . . .
```

SHIFT Left

(<<)

```
a = 1;
for ( i = 1, i <= 8, i++)
{
   printf("%d\n", a);
   a = a << 1;
}
```

```
1
2
4
8
16
32
64
128
```

Example C

```
 2       00000010
 4       00000100
 8       00001000
 16      00010000
 32      00100000
 64      01000000
 128     10000000
------------------------
Process exited after 0
Press any key to conti
```

bits_004.c | bits_003.c | bits_005.c

```c
#include <stdio.h>

int i, a;

void tentobin(int n, int p)
{
    int k;
    if (p == 1 ) printf("%d", n);
    else
    {
        k = n % 2; tentobin(n / 2, p-1); printf("%d",k);
    }
}
int main()
{
    a = 1;
    for (i = 1; i < 8; i++)
    {
        a = a << 1;
        printf("\n %d \t", a); tentobin(a, 8);
    }

    return 0;
}
```

```
14  int main()
15  {
16      a = 10;
17      for (i = 1; i < 8; i++)
18      {
19          a = a << 1;
20          printf("\n %d \t", a); tentobin(a, 16);
21      }
22
23      return 0;
24  }
```

```
 20      0000000000010100
 40      0000000000101000
 80      0000000001010000
 160     0000000010100000
 320     0000000101000000
 640     0000001010000000
 1280    0000010100000000
--------------------------------
Process exited after 0.02973
Press any key to continue .
```

SHIFT Right

(>>)

```
a = 127;
for ( i = 1, i <= 8, i++)
{
   printf("%d\n", a);
   a = a >> 1;
}
```

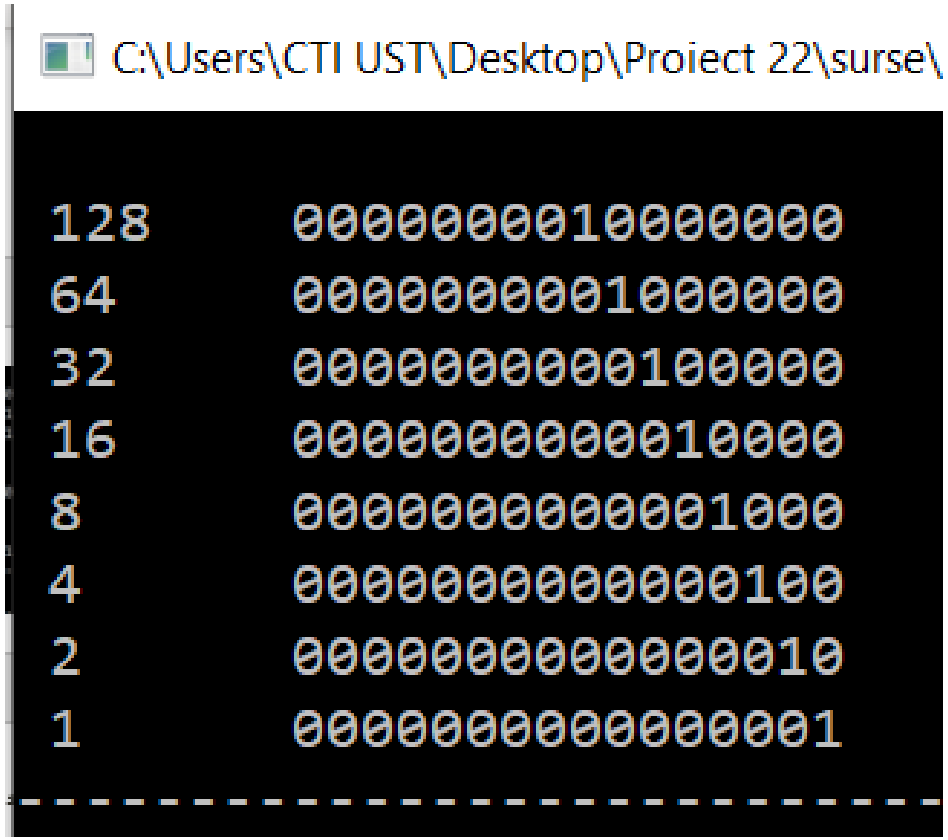
```
127
 63
 31
 15
  7
  3
  1
  0
```

```c
#include <stdio.h>

int i, a;

void tentobin(int n, int p)
{
    int k;
    if (p == 1 ) printf("%d", n);
    else
    {
        k = n % 2; tentobin(n / 2, p-1); printf("%d",k);
    }
}
int main()
{
    a = 128;
    while (a > 0)
    {
        printf("\n %d \t", a); tentobin(a, 16);
        a = a >> 1;
    }

    return 0;
}
```



```c
#include <stdio.h>

int i, a;

void tentobin(int n, int p)
{
    int k;
    if (p == 1 ) printf("%d", n);
    else
    {
        k = n % 2; tentobin(n / 2, p-1); printf("%d",k);
    }
}
int main()
{
    a = 115;
    while (a > 0)
    {
        printf("\n %d \t", a); tentobin(a, 16);
        a = a >> 1;
    }

    return 0;
}
```

```
 115     0000000001110011
 57      0000000000111001
 28      0000000000011100
 14      0000000000001110
 7       0000000000000111
 3       0000000000000011
 1       0000000000000001
--------------------------------
Process exited after 0.02848
Press any key to continue .
```

Set value 1

k^{th} bit

```c
#include <stdio.h>
int i, a, b;
void tentobin(int n, int p)
{
    int k;
    if (p == 1 ) printf("%d", n);
    else
    { k = n % 2; tentobin(n / 2, p-1); printf("%d",k);  }
}
int main()
{   a = 115;
    for (i = 0; i < 8; i++)
    {
        printf("\n %d \t", a); tentobin(a, 16);
        b = a;
        b = b | (1 << i);
        printf("\n %d \t", i); tentobin(b, 16);
    }
    return 0;
}
```

Example

115	0000000001110011
0	000000000111001**1**
115	0000000001110011
1	00000000011100**1**1
115	0000000001110011
2	0000000001110**1**11
115	0000000001110011
3	000000000111**1**011
115	0000000001110011
4	00000000011**1**0011
115	0000000001110011
5	0000000001**1**10011
115	0000000001110011
6	000000000**1**110011
115	0000000001110011
7	00000000**1**1110011

Set value o

k^{th} bit

bits_004.c | bits_003.c | bits_009.c

```c
#include <stdio.h>
int i, a, b;
void tentobin(int n, int p)
{
int main()
{    a = 115;
     for (i = 0; i < 8; i++)
     {
          printf("\n %d \t", a); tentobin(a, 16);
          b = a;
          b = b & (255 - (1 << i));
          printf("\n %d \t", i); tentobin(b, 16);
     }
     return 0;
}
```

Set value
o

k^{th} bit

115	0000000001110011
0	0000000001110010
115	0000000001110011
1	0000000001110001
115	0000000001110011
2	0000000001110011
115	0000000001110011
3	0000000001110011
115	0000000001110011
4	0000000001100011
115	0000000001110011
5	0000000001010011
115	0000000001110011
6	0000000000110011
115	0000000001110011
7	0000000001110011

Check value of

k^{th} bit

```
107    a=23; b=1;
108    for(k=0;k<8;k++)
109    {
110        b = 1 << k;
111        c = a & b;
112        binaryprint(c);
113    }
```

```
00000001
00000010
00000100
00000000
00010000
00000000
00000000
00000000
```

bits_004.c | bits_003.c | bits_009.c

```c
#include <stdio.h>
int i, a, b, c;
void tentobin(int n, int p)
{
int main()
{   a = 121;
    for (i = 0; i < 8; i++)
    {
        printf("\n %d \t", a); tentobin(a, 8);
        b = 1 << i;
        c = a & b;
        printf("\n %d \t", i); tentobin(c, 8);
    }
    return 0;
}
```

```
121     01111001
0       00000001
121     01111001
1       00000000
121     01111001
2       00000000
121     01111001
3       00001000
121     01111001
4       00010000
121     01111001
5       00100000
121     01111001
6       01000000
121     01111001
7       00000000
```

Problemă

Un dispozitiv dispune de o mulțime de porturi p1,p2, p3, … Pk.

Un echipament periferic poate fi conectat prin setul de porturi e1, e2, e3, …, em.

Să se determine setul de porturi prin care echipamentul poate fi conectat la dispozitiv