Operații pe biți

Sau cum accesăm biții

Aplicabilitate

- Operatorii pe biți se pot aplica asupra datelor de tip caracter sau întreg.
- Operatorii pot să fie unari sau binari

Reprezentarea numerelor naturale

- Numerele sunt memorate în baza 2 pe un număr de cifre binare egal cu dimensiunea tipului de dată corespunzător:

Tip	Dimensiune biți	Valoare minimă	Valoare maximă
unsigned char	8	0	255
usigned int	32	0	4294967295
usigned short int	16	0	65535

- Exemplu:
- unsigned char x = 37;   // x are valoarea 00100101
- unsigned short int x = 37;   // x are valoarea 0000000000100101

Reprezentarea numerelor întregi cu semn

- În cazul reprezentării numerelor cu semn primul bit e 0, dacă numărul e pozitiv, altfel e 1.
- Reprezentarea numerelor se face în **cod complementar**. Într-un tip de dată cu **n** biți se pot reprezenta 2^{n-1} valori negative, 0 și $2^{n-1}-1$ valori pozitive.
- Matematic, reprezentarea unui număr negativ în complement față de doi este valoarea 2^n – x, unde x este valoarea absolută a numărului reprezentat.
- Exemplu:

  char x = -37; este reprezentat în memorie 11011011 care se obține:

  2^n(10000000) – 37(00100101) = 11011011

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37								0
Transport							1	

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37							0	0
Transport						1		

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37						0	0	0
Transport					1			

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37					0	0	0	0
Transport				1				

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37				0	0	0	0	0
Transport			1					

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37			0	0	0	0	0	0
Transport		1						

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37		0	0	0	0	0	0	0
Transport	1							

Reprezentarea numerelor întregi cu semn

- De ce se face memorarea numerelor negative în complement de 2:
- char x = -37, y = 37;

Nr	7	6	5	4	3	2	1	0
-37	1	1	0	1	1	0	1	1
37	0	0	1	0	0	1	0	1
-37+37	0	0	0	0	0	0	0	0
Transport								

Reprezentarea numerelor întregi cu semn

- O altă metodă de a calcula complementul față de 2 este:

1. Se pornește de la reprezentarea valorii absolute ex. 37 00100101
2. Se schimbă fiecare 0 în 1 și fiecare 1 în 0 (complement față de 1) 11011010
3. Se adună 1 11011011

- Metoda de reprezentare a numerelor este aceeași, indiferent de dimensiunea tipului de dată.

Operatori

Operator	Descriere
~	Negație - cod complementar față de 1
&	Și - Conjuncție, și logic pe bit
|	Sau – Disjuncție, sau logic pe bit
^	XOR – Sau exclusiv logic pe bit
<<	Deplasare la stânga
>>	Deplasare la dreapta

~ Negația

- Tabel de valori (x are 1 bit)

x	~x
1	0
0	1

- Exemplu

```
unsigned char a, b, c;
a=1;   //00000001 în binar
b=~a;
```

Ce valoare are b în binar ?

254 adică 11111110 în binar

~ Negația

- Exemplu

```
unsigned char a, b, c;
a=37;   //00100101 în binar
b=~a;
```

Ce valoare are b în binar ?

218 adică 11011010 în binar

& Și logic pe bit

- Tabel de valori (x, y au 1 bit)

x	y	x&y
0	0	0
0	1	0
1	0	0
1	1	1

- Exemplu

unsigned char a, b, c;

a=5;

b=4;

c=a&b;

Ce valoare are c ?

Indicație a=00000101 iar b=00000100 în binar

4

| sau logic pe bit

- Tabel de valori (x, y au 1 bit)

x	y	x|y
0	0	0
0	1	1
1	0	1
1	1	1

- Exemplu

```
unsigned char a, b, c;
a=5;
b=4;
c=a|b;
```

Ce valoare are c ?

Indicație a=00000101 iar b=00000100 în binar

5

^ sau exclusiv logic pe bit

- Tabel de valori (x, y au 1 bit)

x	y	x^y
0	0	0
0	1	1
1	0	1
1	1	0

- Exemplu

```
unsigned char a, b, c;
a=5;
b=4;
c=a^b;
```

Ce valoare are c ?

Indicație a=00000101 iar b=00000100 în binar

1

<< deplasare pe biți la stânga

- x<<y ar fi echivalent cu $x=x*2^y$
- Obs. y ar trebui sa fie pozitiv pentru rezultate predictibile
- Exemplu

```
unsigned char a, b, c;
a=5;
b=4;
c=a<<b;
```

Ce valoare are c ?

00000101 devine 01010000

80

>> deplasare pe biți la dreapta

- $x >> y$ ar fi echivalent cu $x = x/2^y$

- Obs. y ar trebui sa fie pozitiv pentru rezultate predictibile

- Exemplu

unsigned char a, b, c;

a=59;

b=4;

c=a>>b;

Ce valoare are c ?

00111011 dispar 00111011 și devine 00000011

3

Tips & Tricks

- Verificare dacă un număr natural **n** e putere a lui 2: n & (n-1) este 0.
- De ce ?
- Dacă n e o putere a lui 2 atunci e format din 1 urmat de cifre de 0, ex. 16: 00010000
- Dacă scădem 1 din acest număr se obține 00001111
- Dacă facem și pe biți obținem valoarea 0 00000000

Tips & Tricks

- Determinarea celei mai mari puteri de 2 care divide pe n: **n & -n**.
- De ce?
- Ex. n = 40   00101000

  -40   11011000 (se obține din 11010111 + 00000001)

  00001000

Tips & Tricks

- Setarea bitului k (numărat de la final) din n pe valoarea 1: n |= 1<<k
- De ce:
- Exemplu: n=37, k=6;

  37 00100101

  1<<6 01000000

  37|1<<6 01100101 =101 în zecimal

Tips & Tricks

- Verificarea dacă bitului k (numărat de la final) din n este pe valoarea 1: n &(1<<k)
- De ce:
- Exemplu: n=37, k=5;

  37 00100101

  1<<5 00100000

  37&(1<<5) 00100000 = 32 în zecimal

Dacă valoarea n &(1<<k) e diferită de 0 atunci bitul k are valoarea 1.

Tips & Tricks

- Setarea bitului k (numărat de la final) din n pe valoarea 0: n &= ~(1<<k)
- De ce:
- Exemplu: n=37, k=5;

  37 00100101

  1<<5 00100000

  ~(1<<5) 11011111

  n&~(1<<5) 00000101

Tips & Tricks

- Se observă că **n ^ n** e 0 indiferent de valoarea lui n
- Din această cauză **a ^ b ^ a** este b
- Utilizare: putem determina dacă într-un șir de valori există o valoare care apare de număr impar de ori.