PROGRAMAREA CALCULATOARELOR
– SEMINAR NR. 2 –

1) Să se interschimbe valorile a două variabile de tip întreg folosind operatorul ^ (XOR/ sau exclusiv pe biți).
2) Implementarea operațiilor de deplasare circulară (rotire) pe biți pentru numere întregi.
3) Testarea/setarea/resetarea/comutarea valorii unui bit din reprezentarea binară internă a unei valori de tip întreg (cu sau fără semn).
4) Să se verifice dacă un număr natural este de forma 2^k (este putere a lui 2) sau nu și, în caz afirmativ, să se afișeze valoarea lui k.

1.

```
#include <stdio.h>

int main()
{
   int a,b;
   scanf("%d %d", &a, &b);
   a=a^b;
   b=a^b;
   a=a^b;
   printf("%d %d", a, b);
   return 0;
}
```

2.

```
#include <stdio.h>

int main()
{
   short int x, mask,k;
   scanf("%d %d",&x,&k);
   mask=(~(1<<k))&x;
   printf("%d\n",mask);
```

```
    x=x>>k;
    x=x|(mask<<k);
    printf("%d\n",x);
}
```

main.c

```c
#include <stdio.h>

int main()
{
    short int x, mask,k;
    scanf("%d %d",&x,&k);
    mask=(~(1<<k))&x;
    printf("%d\n",mask);
    x=x>>k;
    x=x|(mask<<k);
    printf("%d\n",x);
}
```

```
45 3
37
301
```

```
#include <stdio.h>


int main()

{

   int x, mask,k;

   scanf("%d %d",&x,&k);

   mask=((1<<k)-1)&x;

   x=x>>k;

   x=x|(mask<<(k+1));

   printf("%d\n",x);

}//corect


4.

#include <stdio.h>

int main()
{
   int n, ok=0, k=0;
   scanf("%d", &n);
   while(n!=0 && ok<=1)
   {
    if (n&1) ok++;
    k++;
    n >>= 1;
   }
   k--;
   if(ok==1) printf("%d",k);
   else printf("NU");
   return 0;
}
3.
#include <stdio.h>

int main()
{
   int n, ok=0, k;
   scanf("%d %d", &n, &k);
   int mask;
   mask=(1<<k)&n;
```

```
   if(mask)
   {
      printf("%d",1);
   }
   else
      printf("%d",0);
   printf("%d\n",mask);
   return 0;
}
```

4.

```
#include <stdio.h>

int main()
{
   int x,k, nr;
   scanf("%d", &x);
   k=0;
   nr=0;
   while(x)
   {
      if(x&1)
      {
         k++;
      }
      x>>=1;
      nr++;
   }
   if(k==1)
   {
      printf("Este putere a %d lui 2\n", nr-1);
   }
   else printf("Nu este putere a lui 2\n");
   return 0;
}
```

Resurse online:

https://www.infoarena.ro/operatii-pe-biti

http://www.dponline.ro/articol.php?idarticol=81

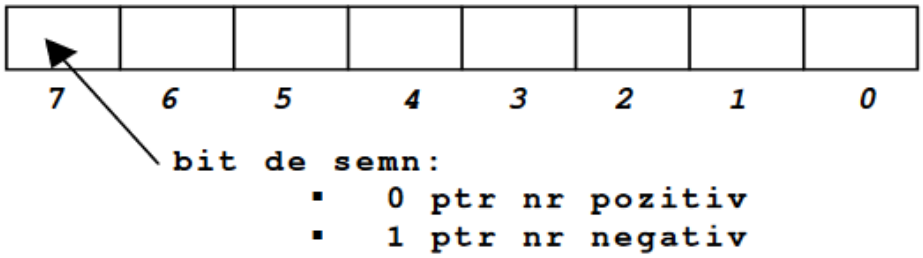
http://campion.edu.ro/arhiva/www/arhiva_2009/papers/paper21.pdf

Operații pe biți

1. Reprezentarea internă a numerelor întregi

- Reprezentarea în memorie a numerelor întregi se face printr-o secvență de cifre de 0 și 1.
- Această secvență poate avea o lungime de 8, 16 sau 32 de biți.
- Forma de memorare a întregilor se numește **cod complementar**.
- În funcție de lungimea reprezentării se stabilește domeniu valorilor care pot fi stocate.

Modul de reprezentare în cod complementar va fi prezentat în cele ce urmează, folosind reprezentarea pe o lungime de 8 biți, valabilă pentru tipul char în C/C++.



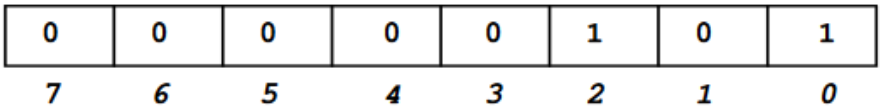
- numerotarea pozițiilor se face de la dreapta la stânga (de la 0 la 7), poziția 7 fiind bit de semn (0 pentru numerele pozitive și 1 pentru numerele negative)
- Rezultă că doar 7 biți (pozițiile 0-6) se folosesc pentru reprezentarea valorii absolute a numărului.
- Numerele întregi pozitive se convertesc în baza 2, apoi se face completarea cu cifre 0 nesemnificative, până la completarea celor 7 biți.

Exemplu:

Forma de reprezentare a numărului întreg 5: $5_{10} = 101_2$

Vor fi necesare 4 cifre de 0 nesemnificative, pentru completarea primelor 7 biți, iar poziția 7 (bitul 8) va fi 0 deoarece numărul este pozitiv.

Deci reprezentarea internă este următoarea:



Reprezentarea numerelor întregi negative. Pași:

1) determinarea reprezentării interne a numărului ce repezintă valoarea absolută a numărului inițial. Acesta are bitul de semn egal cu 0.
2) se calculează complementul față de 1 a reprezentării obținute la pasul anterior (bitul 1 devine 0, iar bitul 0 devine 1)
3) se adună 1 (adunarea se face în baza 2) la valoarea obținută.

Exemplu: Determinarea reprezentării numărului -5:

1) Reprezentarea valorii absolute a numărului -5 este 00000101.
2) Complementul față de 1 este 11111010
3) Numărul obținut după adunarea cu 1 este 11111011

Deci, reprezentarea numărului -5 pe 8 biți este:

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

OBS: bitul de semn este 1 ceea ce ne indică faptul că avem de a face cu un număr negativ.

Concluzie: numerele întregi care se poate reprezenta pe 8 biți sunt cuprinse între 10000000_2 și 01111111_2 , adică -128_{10}, 127_{10}

OBS: umerele întregi se pot reprezenta în cod complementar, având la dispoziție 16, 32 sau 64 de biți. Mecanismul este același, însă valorile numerelor cresc.

2. Operatori la nivel de bit

Operatorii pe biți se pot aplica datelor ce fac parte din tipurile întregi. Operațiile se efectuează asupra biților din reprezentarea internă a numerelor.

a) **Operatorul de negație ~ (C/C++)** este un operator unar care întoarce numărul întreg a cărui reprezentare internă se obține din reprezentarea internă a numărului inițial, prin complementarea față de 1 a fiecărui bit(1→0 și 0→1).

Exemplu: ~ 5 = -6

Reprezentarea lui 5 este 00000101. Complementând acest număr se obține reprezentarea 11111010, care corespunde numărului întreg -6.

Verificare: Numărul -6 se reprezintă intern astfel:

1) 6_2 = 00000110,
2) complementând obținem 11111001
3) adunând 1 (în baza 2) se obține în final 11111010, adică exact reprezentarea internă returnată a numărului returnat de operația (not 5) ~5.

b) **Operatorul de conjuncție &** este un operator binar care returnează numărul întreg a cărui reprezentare internă se obține prin conjuncția biților care apar în reprezentarea internă a operanzilor. Conjuncția se face cu toate perechile de biți situați pe aceeași poziție.

Exemplu: 5 & 3 = 1
Verificare:
Reprezentarea internă a lui 5 este 00000101, iar a lui 3 este 00000011.

```
00000101 &
00000011
--------
00000001
```

Această reprezentare este dată de numărul întreg 1.

c) **Operatorul de disjuncție |** este operator binar care returnează numărul întreg a cărui reprezentare internă se obține prin disjuncția biților care apar în reprezentarea internă a operanzilor. Disjuncția se face între biții situați pe aceeași poziție.

Exemplu: 15 | 3 == 15
Reprezentarea internă a lui 15 este 00001111 iar a lui 3 este 00000011.

```
00001111 |
00000011
--------
00001111
```

d) **Operatorul "sau exclusiv" (xor) ^** este un operator binar care returnează numărul întreg a cărui reprezentare internă se obține prin operația or exclusiv asupra biților care apar în reprezentarea internă a operanzilor. Operația se face între biții situați pe aceeași poziție.

Exemplu: 15 ^ 3=12

```
00001111 ^
00000011
--------
00001100
```

Proprietăți ale operatorului XOR:
(i) Comutativitate: a^b = b^a
(ii) Asociativitate: (a^b)^c = a^(b^c)
(iii) Element neutru 0: a^0 = 0^a = a
(iv) Inversare: a^a = 0

e) **Operatorul shift left << e**ste un operator binar care returnează numărul întreg a cărui reprezentare este obținută din reprezentarea internă a primului operand prin deplasare la stânga cu un număr de biți egal cu al doilea operand.

Exemplu: 4<<2 = 16
Explicație: Reprezentarea internă a numărului 4 este 00000100. Prin deplasare la stânga cu doi biți se obține 00010000, care este reprezentarea internă a lui 16.

OBS: Acest operator poate fi folosit pentru calculul numerelor întregi de forma 2^n prin efectuarea operațiilor de forma 1<<n.

f) **Operatorul shift right >>** este un operator binar care returnează numărul întreg a cărui reprezentare este obținută din reprezentarea internă a primului operand prin deplasare la dreapta cu un număr de biți egal cu al doilea operand. Prin deplasarea la dreapta, primii biți din reprezentarea internă a numărului (pozițiile 1,2, ș.a.m.d.) se pierd iar ultimii se completează cu zero.

Exempliu: 4 >> 2= 3
Explicație: Reprezentarea internă a numărului 14 este 000001110. Prin deplasare la dreapta cu doi biți a reprezentării lui 14 se obține 00000011, care este reprezentarea internă a lui 3.

OBS: Operația n>>1 este echivalentă cu împărțirea întreagă la 2.