Lucrarea de laborator nr. 1

I. Scopul lucrării

Reprezentarea numerelor întregi în calculator. Erori.

II. Conținutul lucrării

1. Reprezentarea numerelor întregi fără semn
2. Reprezentarea numerelor întregi cu semn
3. Erori care apar ca urmare a limitelor de reprezentare.

III. Prezentarea lucrării

III.1. Reprezentarea internă a numerelor întregi fără semn (pozitive)

Reprezentarea în memoria unui calculator a numerelor întregi depinde de lungimea cuvântului utilizat (numărul de biți). Pentru a reprezenta un număr întreg pozitiv x pe k biți se face conversia numărului respectiv la baza 2, iar configurația binară obținută se completează la stânga cu zerouri până se obțin k cifre. Dacă numărul de cifre binare obținute prin conversie este mai mare decât k, atunci reprezentarea se trunchiază.

Cel mai mare număr întreg reprezentabil pe k biți este

$$\underbrace{11\ldots1}_{\text{k ori}}{}_2 = 2^k + 2^{k-1} + \ldots + 2^0 = \frac{2^k - 1}{2 - 1} = 2^k - 1$$

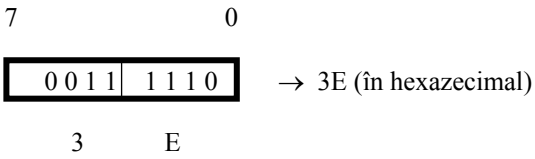
Deci pe k biți se pot reprezenta numerele întregi cuprinse între 0 şi $2^k - 1$, în total 2^k numere.

Exemple:

K	Domeniul de valori
8 biți = 1 B (byte)	0…255
16 biți = 2 B	0….65 535
32 biți = 4 B	0….2 147 483 647

Reprezentarea numărului 62_{10} pe k = 8 biți se face astfel:

$62 = 2^5 + 2^4 + 2^3 + 2^2 + 2 = 111110_2 = 00111110_2$



În reprezentarea binară a numerelor, ponderile cifrelor binare cresc de la dreapta la stânga, numerotarea lor corespunzând puterilor crescătoare ale bazei de numerație 2. Astfel, prima cifră binară din dreapta reprezintă ponderea 2^0 şi este ***bitul cel mai puțin semnificativ***. Primul bit din stânga este ***bitul cel mai semnificativ***.

Notăm cu $\prec$ relația de ordine lexicografică pe mulțimea şirurilor formate din 0 şi 1, de lungime k. Din punct de vedere al ordinii lexicografice, reprezentările numerelor întregi fără semn satisfac:

$$0 \prec 1 \prec \ldots \prec 2^k - 1$$

În consecință, relația de ordine lexicografică este compatibilă cu relația de ordine numerică.

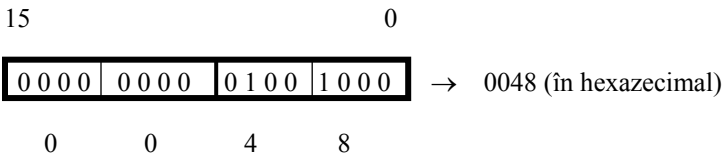
III.2. Reprezentarea internă a numerelor întregi (cu semn)

Numerele întregi pot fi codificate prin trei metode:

- ***cod direct*** (***semn şi valoare absolută***)
- ***cod invers*** (***complement față de 1***)
- ***cod complementar*** (***complement față de 2***)

Să presupunem că se rezervă k biți pentru reprezentarea unui număr întreg.

Prin toate cele metode numerele întregi pozitive se codifică prin conversie în baza 2 pe k-1 biți şi completarea primului bit cu zero. Astfel, dacă k = 16, atunci reprezentare internă a numărului $x = 72_{10} = 2^6 + 2^3 = 1001000_2$ este :



Numerele întregi pozitive reprezentabile pe k biți (prin oricare din cele trei metode) sunt cuprinse în intervalul $[0, 2^{k-1}-1]$ (deoarece primul bit este rezervat, iar reprezentarea se face doar pe restul de k-1 biți).

Prezentăm mai departe modul în care se face codificarea numerelor întregi negative prin fiecare din cele trei metode.

Prin ***metoda semn şi valoare*** (***cod direct***) se codifică valoarea absolută a numărului întreg negativ pe k-1 biți şi se completează primul bit cu 1. În consecință, numerele întregi negative care se pot reprezenta prin această metodă sunt cuprinse în intervalul $[-2^{k-1}+1, -1]$. Această metodă de reprezentare prezintă unele inconveniente:

- zero are două reprezentări distincte (00…0 şi 10…0), ca +0 şi –0
- tabelele de adunare şi înmulțire sunt complicate datorită bitului de semn care trebuie tratat separat

Din punct de vedere al ordinii lexicografice avem:

$$+0 \prec 1 \prec \ldots \prec 2^{k-1}-1 \prec -0 \prec -1 \prec \ldots \prec -(2^{k-1}-1)$$

Pentru a obține ***codul invers***:

1) se obține reprezentarea valorii absolute a numărului întreg negativ pe k biți
2) în reprezentarea obținută (la pasul 1) se înlocuieşte fiecare bit 0 cu 1 şi 1 cu 0

Intervalul în care se găsesc numerele întregi care pot fi reprezentate prin această metodă este acelaşi ca pentru metoda precedentă ($[-2^{k-1}+1, 2^{k-1}-1]$). Dacă notăm cu $\overline{x}$ complementul față de 1 al numărului x, atunci

$$x + \overline{x} = \underbrace{11....1}_{k}{}_2 = 2^k - 1$$

În consecință, complementul față de 1 poate fi folosit pe post de opus față de operația de adunare. Din acest motiv operațiile aritmetice relativ la această reprezentare sunt avantajoase, deoarece operația de scădere se realizează prin adunarea complementului față de 1.

Un dezavantaj al acestei reprezentări este recunoaşterea a două zerouri (11…1 şi 00..0).

Pentru a obține ***codul complementar***:

1) se obține codul invers pe k biți
2) se adună o unitate la valoarea obținută (la pasul 1)

Dacă $\overline{\overline{x}}$ este complementul față de doi al lui x, atunci

$$x + \overline{\overline{x}} = \underbrace{11....1}_{k}{}_2 + 1 = 2^k - 1 + 1 = 2^k$$

Şi în cazul acestei reprezentări operaţiile aritmetice sunt avantajoase, deoarece operaţia de scădere se realizează prin adunarea complementului faţă de 2.

Zero admite o singură reprezentare ($\underbrace{00...0}_{k}$).

Prin această metodă se pot reprezenta numerele întregi care se găsesc în intervalul $[-2^{k-1}, 2^{k-1}-1]$.

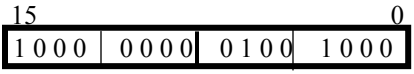
Din cele de mai sus rezultă că reprezentarea prin cod complementar este cea mai avantajoasă.

Exemple

Considerăm k = 16. Reprezentările pentru x = -72 prin cele trei metode se obţin după cum urmează:

$72_{10} = 1001000_2 = 1001000_2$

Cod direct (semn şi valoare absolută)



→ 1048 (în hexazecimal)

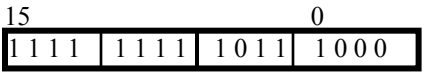
1 0 4 8

Cod invers (complement faţă de 1)

→ FFB7 (în hexazecimal)

F F B 7

Cod complementar(complement faţă de 2)



→ FFB8 (în hexazecimal)

F F B 8

(deoarece $1111111110110111_2 + 1 = 1111111110111000_2$)

Observaţie: Se poate remarca faptul că primul bit din stânga (bitul cel mai semnificativ) este întotdeauna 0 pentru numerele pozitive şi 1 pentru numerele negative şi aceasta pentru fiecare din c ele trei reprezentări. Acest bit se mai ***numeşte bit de semn***.

Dacă se utilizează codul complementar pe k biți pentru reprezentarea unui număr întreg x, atunci valoarea $\overset{=}{x}$ care corespunde reprezentării binare este dată din tabelul de mai jos

Semnul lui x	$\overset{=}{x}$
+	x
-	$2^k-\|x\|=2^k+x$

iar zero admite o unică reprezentare $\underbrace{00...0}_{k}$.

În tabelul din figura 1 sunt reprezentate, prin cele trei metode, pe k = 16 biți numere întregi cu semn:

Valoare Zecimală	cod direct (semn şi valoare absolută)	cod indirect (compl. față de 1)	cod complementar (compl. față de 2)
32767	$0111\ldots111=7FFF_{16}$	$0111\ldots111=7FFF_{16}$	$0111\ldots111=7FFF_{16}$
32766	$0111\ldots110=7FFE_{16}$	$0111\ldots110=7FFE_{16}$	$0111\ldots110=7FFE_{16}$
…	…	…	…
1	$0000\ldots001=0001_{16}$	$0000\ldots001=0001_{16}$	$0000\ldots001=0001_{16}$
+0	$0000\ldots000=0000_{16}$	$0000\ldots000=0000_{16}$	$0000\ldots000=0000_{16}$
-0	$1000\ldots000=8000_{16}$	$1111\ldots111=FFFF_{16}$	$0000\ldots000=0000_{16}$
-1	$1000\ldots001=0001_{16}$	$1111\ldots110=FFFE_{16}$	$1111\ldots111=FFFF_{16}$
…	…	…	…
-32766	$1111\ldots110=FFFE_{16}$	$1000\ldots001=8001_{16}$	$1000\ldots010=8002_{16}$
-32767	$1111\ldots111=FFFF_{16}$	$1000\ldots000=8000_{16}$	$1000\ldots001=8001_{16}$
$-32768=-2^{16}$	———	———	$1000\ldots000=8000_{16}$

figura 1

Probleme propuse

1. Găsiți reprezentarea pe 8 biți a numărului întreg fără semn x = 57.
2. Găsiți reprezentările pe 8 biți corespunzătoare celor trei metode (cod direct, cod invers, cod complementar) a numărului întreg x = -57.
3. Găsiți valoarea zecimală a numărului întreg cu semn C8 (în hexazecimal) codificat folosind cod complementar.

III.3. Erori care apar ca urmare a limitelor de reprezentare.

Utilizați următoarele programe în Pascal, respectiv in C pentru a pune în evidență câteva erori care apar ca urmare a limitelor de reprezentare. Explicați rezultatele afișate. (Reprezentarea utilizată pentru numerele întregi cu semn este codul complementar.)

```
var x,x1,x2,x3:byte;
    z:word;
    y:shortint;
    w:integer;
    begin
        writeln;
        write ('x= ');readln(x);
        writeln('x=',x);
        write ('y= ');readln(y);
        writeln('y=',y);
          write ('z= ');readln(z);
        writeln('z=',z);
          write ('w= ');readln(w);
         writeln('w=',w);
         x2:=200;x3:=67;
         x1:=x2+x3;
         writeln (x2,'+',x3,'=',x1);
         readln;
         end.
```

```
#include<stdio.h>
#include<conio.h>
void main(){
      unsigned char x,x1,x2,x3;
      unsigned int z;
      signed char y;
      int w;
      printf("x= ");scanf("%u",&x);
      printf("x=%d\n",x);
      printf("y= ");scanf("%d",&y);
      printf("y=%d\n",y);
      printf("z= ");scanf("%d",&z);
      printf("z=%d\n",z);
```

```
            printf("w= ");scanf("%d",&w);
            printf("w=%d\n",w);
            x2=200;x3=67;
            x1=x2+x3;
            printf("%u+%u=%u",x2,x3,x1);
            getch();
}
```