# BAZELE ARITMETICE ALE CALCULATOARELOR (I)

## 1. Scopul lucrării

Lucrarea prezintă sistemele de numerație, conversia dintr-o bază în alta pentru numere întregi şi fracționare, operațiile aritmetice elementare pentru numere fără semn, reprezentarea numerelor cu semn şi operații aritmetice cu numere reprezentate în complement față de 2.

## 2. Considerații teoretice

### 2.1. Sisteme de numerație

Prin *sistem de numerație* se înțelege totalitatea regulilor de reprezentare ale numerelor cu ajutorul simbolurilor denumite *cifre*.

Sistemele de numerație pot fi *poziționale* sau *nepoziționale*. Un exemplu de sistem pozițional este sistemul zecimal, iar un sistem nepozițional este cel roman. Într-un sistem pozițional, un număr $N$ cu parte întreagă şi parte fracționară, separate prin virgulă, se poate scrie sub oricare din următoarele forme:

$$N = a_{n-1}\, a_{n-2} \ldots a_1\, a_0\, ,\, a_{-1}\, a_{-2} \ldots a_{-(m-1)}\, a_{-m} \tag{1.1}$$

$$N = a_{n-1}\, q^{n-1} + a_{n-2}\, q^{n-2} + \ldots + a_1\, q^1 + a_0\, q^0 + a_{-1}\, q^{-1} + a_{-2}\, q^{-2} + \ldots + a_{-m}\, q^{-m} \tag{1.2}$$

$$N = \sum_{i=-m}^{n-1} a_i q^i \tag{1.3}$$

unde: $q$ este baza sistemului de numerație (întreg pozitiv);
$a_i$ reprezintă cifre: $0 \le a_i < q$, $i = n\text{-}1, n\text{-}2, \ldots, 1, 0, -1, -2, \ldots, -m$;
$n$ este numărul de cifre întregi ale numărului;
$m$ este numărul de cifre fracționare ale numărului.

Într-un sistem de numerație, cifra este un simbol care reprezintă o cantitate întreagă. Numărul de simboluri permise pentru reprezentarea cifrei într-un sistem de numerație se numeşte *baza* sau *rădăcina* sistemului de numerație. Cifrele $a_i$ reprezintă coeficienții cu care se înmulțesc puterile $q^i$ ale bazei $q$ în dezvoltarea polinomială a numărului pentru obținerea valorii sale.

Cifra $a_{n-1}$ din relațiile (1.1) şi (1.2) este cifra cea mai semnificativă (c.m.s.) a numărului, iar cifra $a_{-m}$ este cifra cea mai puțin semnificativă (c.m.p.s.). În cazul sistemului de numerație binar, pentru cifra binară se foloseşte prescurtarea de *bit*. Dacă $m = 0$, numărul $N$ este întreg. Dacă $n = 0$, numărul $N$ este fracționar şi subunitar. Dacă $m$ şi $n$ sunt întregi şi diferiți de zero, numărul $N$ este mixt.

#### Exemple

$465{,}75_{10} = 4 \times 10^2 + 6 \times 10^1 + 5 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}$

$110110{,}01_2 = 2^5 + 2^4 + 2^2 + 2^1 + 2^{-2} = 54{,}25$

Relația (1.2) explică de ce astfel de sisteme de numerație sunt denumite poziționale. Fiecare cifră $a_i$ contribuie la valoarea numărului respectiv cu o pondere dată de puterea $i$ a bazei $q$.

Pentru un sistem de numerație în baza $q$ trebuie să existe $q$ simboluri. La sistemele de numerație cu baza $q > 10$ se introduc simboluri noi. De exemplu, pentru sistemul hexazecimal se introduc literele de la A la F. Reprezentarea primelor 16 numere în sistemul zecimal, binar şi hexazecimal este dată în Tabelul 1.1.

**Tabelul 1.1.** Reprezentarea unor numere în sistemul zecimal, binar şi hexazecimal.

| $q$ = 10 | $q$ = 2 | $q$ = 16 |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

## 2.2. Conversia bazei de numerație

Pentru a deduce o metodă sistematică de conversie dintr-un sistem de numerație în altul, folosim rezultatul cunoscut că, dacă $N$ şi $q$ sunt numere întregi, există întotdeauna un singur întreg $r$ (care nu este negativ) mai mic decât $q$ şi un singur întreg $C$, astfel încât:

$$\frac{N}{q} = C + \frac{r}{q},\ 0 \le r < q \tag{1.4}$$

### 2.2.1. Conversia numerelor întregi

Aplicăm această regulă la conversia unui număr întreg $N_p$ din baza $p$ în echivalentul său în baza $q$, adică $N_q$. Trebuie să determinăm valorile întregi nenegative $a_0$, $a_1$, …, $a_{n-1}$, fiecare mai mic decât $q$, astfel ca:

$$N_p = a_{n-1}\, q^{n-1} + \ldots + a_2\, q^2 + a_1\, q^1 + a_0\, q^0 \tag{1.5}$$

Prin împărțire cu baza $q$ obținem:

$$\frac{N_p}{q} = \underbrace{a_{n-1}q^{n-2} + \ldots + a_2 q^1 + a_1 q^0}_{\text{parte întreaga}} + \underbrace{\frac{a_0}{q}}_{\text{parte fractionara}} = C_0 + \frac{r_0}{q} \tag{1.6}$$

Deoarece atât câtul, cât şi restul sunt unici, egalând părțile fracționare avem:

$$a_0 = r_0 \tag{1.7}$$

$$C_0 = a_{n-1}\, q^{n-2} + \ldots + a_2\, q^1 + a_1\, q^0 \tag{1.8}$$

Împărțind câtul $C_0$ la $q$ şi utilizând rezultatele de mai sus, obținem:

$$\frac{C_0}{q} = a_{n-1}q^{n-3} + ... + a_2 q^0 + \frac{a_1}{q} = C_1 + \frac{r_1}{q} \tag{1.9}$$

deci

$$a_1 = r_1 \tag{1.10}$$
$$C_1 = a_{n-1}\, q^{n-3} + \ldots + a_2\, q^0 \tag{1.11}$$

Operația se continuă până când se obține un cât egal cu zero.

Algoritmul de conversie a numerelor întregi este deci următorul:

1. Se împarte numărul inițial $N_p$ (în baza $p$) la noua bază $q$. Se obține câtul $C_0$ şi restul $a_0$.
2. Se împarte câtul $C_0$ la $q$. Se obține câtul $C_1$ şi restul $a_1$.
3. Se continuă până când se obține câtul $C_n = 0$. Resturile obținute reprezintă cifrele numărului convertit, $a_0$ fiind cifra c.m.p.s.

### Exemple

1) Conversie din zecimal în binar

```
35 : 2
17 : 2        a0 = 1
 8 : 2        a1 = 1
 4 : 2        a2 = 0
 2 : 2        a3 = 0
 1 : 2        a4 = 0
 0            a5 = 1
```

$35_{10} = 10\ 0011_2$

2) Conversie din zecimal în octal

```
35 : 8
 4 : 8        a0 = 3
 0            a1 = 4
```

$35_{10} = 43_8$

3) Conversie din zecimal în hexazecimal

```
35 : 16
 2 : 16       a0 = 3
 0            a1 = 2
```

$35_{10} = 23_{16}$

### 2.2.2. Conversia numerelor fracționare

Pentru conversia numerelor fracționare subunitare trebuie determinați coeficienții întregi nenegativi $a_{-1}$, $a_{-2}$, …, $a_{-m}$, fiecare mai mic decât baza $q$, astfel încât:

$$N_p = a_{-1}\, q^{-1} + a_{-2}\, q^{-2} + a_{-3}\, q^{-3} + \ldots + a_{-m}\, q^{-m} \tag{1.12}$$

Prin înmulțire cu baza $q$ obținem:

$$qN_p = a_{-1} + (a_{-2}\, q^{-1} + a_{-3}\, q^{-2} + \ldots + a_{-m}\, q^{-m+1}) = u_{-1} + F_1 \tag{1.13}$$

unde $u_{-1}$ este partea întreagă, iar $F_1$ este partea fracționară a valorii obținute. Deci,

$$a_{-1} = u_{-1} \tag{1.14}$$
$$F_1 = a_{-2}\, q^{-1} + a_{-3}\, q^{-2} + \ldots + a_{-m}\, q^{-m+1} \tag{1.15}$$

Înmulțind partea fracționară $F_1$ cu $q$ obținem:

$$qF_1 = a_{-2} + (a_{-3}\, q^{-1} + \ldots + a_{-m}\, q^{-m+2}) = u_{-2} + F_2 \tag{1.16}$$

deci:

$$a_{-2} = u_{-2} \tag{1.17}$$
$$F_2 = a_{-3}\, q^{-1} + \ldots + a_{-m}\, q^{-m+2} \tag{1.18}$$

Operația se continuă până când se obține o parte fracționară egală cu zero sau se ajunge la precizia cerută.

Algoritmul de conversie a numerelor fracționare este deci următorul:

1. Se înmulțește numărul inițial $N_p$ (în baza $p$) cu noua bază $q$. Se obține partea fracționară $F_1$ și partea întreagă $a_{-1}$.
2. Se înmulțește partea fracționară $F_1$ cu $q$. Se obține partea fracționară $F_2$ și partea întreagă $a_{-2}$.
3. Se continuă până când se obține partea fracționară $F_m = 0$ sau se ajunge la precizia cerută. Cifrele întregi obținute reprezintă cifrele numărului în baza $q$, $a_{-1}$ fiind cifra c.m.s.

### Exemple

1) Conversie din zecimal în binar

```
0,35 × 2
0,7  × 2      a-1 = 0
1,4  × 2      a-2 = 1
0,8  × 2      a-3 = 0
1,6  × 2      a-4 = 1
1,2  × 2      a-5 = 1
0,4           a-6 = 0
```

$0,35_{10} = 0,010110..._2$

Operația se continuă fără a se putea ajunge la $F_m = 0$. Deci, un număr fracționar finit într-un sistem de numerație nu poate fi reprezentat întotdeauna printr-un număr finit într-un alt sistem de numerație.

2) Conversie din zecimal în octal

```
0,3125 × 8
2,5    × 8    a-1 = 2
4,0           a-2 = 4
```

$0,3125_{10} = 0,24_8$

3) Conversie din zecimal în hexazecimal

```
0,25 × 16
4,0           a-1 = 4
```

$0,25_{10} = 0,4_{16}$

Pentru numere fracționare mai mari decât 1, partea întreagă și cea fracționară se obțin separat.

În cazul particular în care baza finală $q$ este o putere întreagă a bazei inițiale $p$:

$$q = p^r, r \in Z$$

se poate utiliza un procedeu mai simplu, deoarece unei cifre în baza $q$ îi corespund $r$ cifre în baza $p$. Numărul în baza $p$ se partiționează în grupe de câte $r$ cifre, și se înlocuiește fiecare grup de la dreapta și de la stânga virgulei cu echivalentul său în baza $q$.

Astfel, conversia *din binar în octal* se poate efectua prin înlocuirea fiecărui grup de 3 cifre binare prin echivalentul său octal, iar conversia *din binar în hexazecimal* se poate efectua prin înlocuirea fiecărui grup de 4 cifre binare prin echivalentul său hexazecimal. Dacă ultima grupă de cifre binare de la stânga sau de la dreapta virgulei nu este completă, se adaugă zerouri la stânga, respectiv la dreapta, până la completarea grupei.

### Exemple

$11100,110100_2 = 011\ 100,110\ 100_2 = 34,64_8$

$10100011,01_2 = 1010\ 0011,0100_2 = A3,4_{16}$

În cazul în care baza inițială $p$ este o putere a bazei finale $q$:

$$p = q^r, r \in Z$$

se înlocuieşte fiecare cifră în baza $p$ cu $r$ cifre în baza $q$.

Astfel, conversia *din octal în binar* şi cea *din hexazecimal în binar* se efectuează prin înlocuirea cifrelor octale, respectiv hexazecimale, printr-un grup de 3, respectiv 4 cifre binare.

**Exemple**

$123,4_8 = 001\ 010\ 011,100_2$

$53,C_{16} = 0101\ 0011,1100_2$

Conversia *din octal în hexazecimal* şi *din hexazecimal în octal* se poate efectua după ce s-a efectuat conversia în binar.

**Exemple**

$123,4_8 = 001\ 010\ 011,100_2 = 0\ 0101\ 0011,1000_2 = 53,8_{16}$

$A7,E_{16} = 1010\ 0111,1110_2 = 010\ 100\ 111,111_2 = 247,7_8$

### 2.2.3. Conversia binar-zecimală

Conversia unui număr binar întreg în echivalentul său zecimal se poate efectua prin metoda înmulțirii repetate cu 2, cunoscând faptul că operația se efectuează în sistemul zecimal. Se consideră numărul:

$$N = a_{n-1}\, q^{n-1} + a_{n-2}\, q^{n-2} + \ldots + a_1\, q^1 + a_0\, q^0 = [\ldots((a_{n-1}\, q + a_{n-2})\, q + a_{n-3})\, q + \ldots + a_1]\, q + a_0 \quad (1.19)$$

Pentru $q = 2$ rezultă următorul algoritm:

1. Se înmulțeşte cifra c.m.s. a numărului binar cu 2, şi se adună la rezultat următoarea cifră semnificativă.
2. Se înmulțeşte rezultatul cu 2 şi se adună următoarea cifră semnificativă.
3. Se continuă până când s-a prelucrat şi cifra c.m.p.s. a numărului binar. Rezultatul obținut este echivalentul zecimal al numărului binar dat.

**Exemplu**

$10111_2 = (((1\times2 + 0)\times2 + 1)\times2 + 1)\times2 + 1$
$= (5\times2 + 1)\times2 + 1 = 23_{10}$

Pentru numere binare fracționare (subunitare) există factorizarea:

$$N = a_{-1}q^{-1} + a_{-2}q^{-2} + \ldots + a_{-m}q^{-m} = \frac{1}{q}\left(a_{-1} + \frac{1}{q}\left(a_{-2} + \ldots + \frac{1}{q}\left(a_{-m+1} + a_{-m}\frac{1}{q}\right)..\right)\right) \quad (1.20)$$

Pentru $q = 2$ rezultă algoritmul:

1. Se împarte cifra c.m.p.s. a numărului binar cu 2, şi se adună următoarea cifră semnificativă.
2. Se împarte rezultatul cu 2 şi se adună următoarea cifră semnificativă.
3. Se continuă până când se efectuează împărțirea care corespunde cifrei c.m.s. a numărului fracționar.

**Exemplu**

$0,1011_2 = (((1/2 + 1)/2 + 0)/2 + 1)/2 = ((1,5/2 + 0)/2 + 1)/2$
$= (0,75/2 + 1)/2 = 1,375/2 = 0,6875_{10}$

În practică, aceste operații se efectuează mai simplu prin adunarea puterilor bazei 2.

**Exemple**

$10111_2 = 1\times2^4 + 1\times2^2 + 1\times2^1 + 1\times2^0 = 16 + 4 + 2 + 1 = 23_{10}$

$0,1011_2 = 2^{-1} + 2^{-3} + 2^{-4} = 0,5 + 0,125 + 0,0625 = 0,6875_{10}$

În mod similar se poate proceda pentru conversia din hexazecimal în zecimal.

**Exemplu**

$1A8_{16} = 1\times16^2 + 10\times16^1 + 8\times16^0 = 256 + 160 + 8 = 424_{10}$

## 2.3. Operații aritmetice cu numere fără semn

### 2.3.1. Adunarea

Adunarea a două cifre în baza *q* este o operație modulo *q*, deci cifra cu valoarea cea mai mare va fi *q*-1 (de exemplu, 9 în zecimal, 1 în binar, F în hexazecimal). Dacă rezultatul adunării a două cifre de rang *i* depășește această valoare, va apare un transport către rangul *i*+1, care se va aduna la suma cifrelor de rang *i*+1. Apariția unui transport de la cifra c.m.s. indică o depășire a capacității de reprezentare a rezultatului.

În cazul *adunării binare*, cifra sumei este 1 dacă unul din termenii adunării este 1. Cifra de transport este 1 numai dacă ambii termeni ai adunării sunt 1. În Tabelul 1.2 se prezintă regula de adunare a două cifre binare *x* şi *y*.

**Tabelul 1.2.** Adunarea a două cifre binare.

| *x* | *y* | Transportul | Suma |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Exemple**

$22_{10} = 10110_2 +$
$\underline{19_{10} = 10011_2}$
$41_{10} = 101001_2$

$6B32_{16} +$
$\underline{4DF1_{16}}$
$B923_{16}$

### 2.3.2. Scăderea

La scăderea a dacă cifre de rang *i*, dacă cifra descăzutului este mai mică decât cifra scăzătorului, apare un împrumut de la rangul *i*+1.

În cazul *scăderii binare*, diferența este 1 dacă fie descăzutul, fie scăzătorul este 1. Împrumutul apare numai dacă descăzutul este 0 şi scăzătorul este 1. În Tabelul 1.3 se prezintă regula de scădere a două cifre binare *x* şi *y*.

**Tabelul 1.3.** Scăderea a două cifre binare.

| *x* | *y* | Împrumutul | Diferența |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

### Exemple

$22_{10} = 10110_2$ -
$\underline{19_{10} = 10011_2}$
$3_{10} = 00011_2$

$6B32_{16}$ -
$\underline{4DF1_{16}}$
$1D41_{16}$

### 2.3.3. Înmulțirea

Înmulțirea se efectuează de obicei prin adunarea repetată a unor produse parțiale. La înmulțirea a două cifre binare, produsul este 1 numai dacă deînmulțitul şi înmulțitorul sunt 1. În Tabelul 1.4 se prezintă regula de înmulțire a două cifre binare *x* şi *y*.

**Tabelul 1.4.** Înmulțirea a două cifre binare.

| *x* | *y* | Produsul |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### Exemplu

$12 \times 6 = 72$

| | |
|---|---|
| $12 = 1100 \times$ | Deînmulțit |
| $6 = \underline{0110}$ | Înmulțitor |
| $\underline{0000}$ | |
| 1100 | Produse parțiale |
| 1100 | |
| $\underline{0000 \quad -}$ | |
| 1001000 | Produs (64+8) = 72 |

### 2.3.4. Împărțirea

Împărțirea a două numere nu se poate efectua dacă împărțitorul este egal cu zero. Fiind dat deîmpărțitul *X* şi împărțitorul *Y*, pentru operația de împărțire trebuie să se determine câtul *Q* şi restul *R*, astfel încât să fie satisfăcută relația:

$$X = Q \cdot Y + R$$

La împărțirea zecimală se determină cifrele câtului prin alegerea unei cifre şi scăderea din restul parțial (care este inițial o parte a deîmpărțitului) a produsului dintre această cifră şi împărțitor. Dacă rezultatul scăderii este un număr pozitiv mai mic decât împărțitorul, cifra aleasă este corectă. În caz contrar, se alege o altă cifră şi operația se repetă. În fiecare etapă a operației se obține o cifră a câtului.

În cazul împărțirii binare, dacă se alege în mod eronat o cifră a câtului, o nouă alegere nu mai este necesară, existând numai două cifre. Operația de împărțire se va reduce la o serie de scăderi ale împărțitorului din restul parțial, care se efectuează numai dacă restul parțial este mai mare decât împărțitorul, caz în care cifra câtului este 1; în caz contrar, cifra corespunzătoare a câtului este 0.

**Exemplu**

147 : 11 = 13, rest 4

$147_{10} = 10010011_2$
$11_{10} = 1011_2$

```
10010011 : 1011 = 1101      ← Cât (13₁₀)
 1011
  1110       ← Rest parțial
  1011
    1111     ← Rest parțial
    1011
     100     ← Rest (4₁₀)
```

## 2.4. Reprezentarea numerelor în calculator

Există mai multe forme de reprezentare a numerelor în calculator, în funcție de soluția aleasă pentru a indica semnul numerelor sau poziția virgulei.

Numerele reprezentate în calculator pot fi *fără semn* sau *cu semn*. Numerele *fără semn* sunt reprezentate în binar sau într-un cod binar zecimal. În cazul numerelor *cu semn*, se utilizează o cifră de semn pentru indicarea semnului. Convențional, se atribuie cifra 0 pentru semnul plus şi cifra 1 pentru semnul minus. Cifra de semn este reprezentată pe poziția c.m.s. a numărului. Pentru reprezentarea unui număr binar cu semn de $n$ cifre binare, sunt necesare deci $n$+1 poziții.

În general, un număr are o parte întreagă şi o parte fracționară, separate prin virgula binară. Virgula nu se reprezintă fizic, dar trebuie cunoscută localizarea ei. După modul de amplasare a virgulei binare, există două forme de reprezentare a numerelor:

- Forma cu virgulă fixă
- Forma cu virgulă mobilă

În *forma cu virgulă fixă*, virgula care separă partea întreagă de cea fracționară este aşezată într-o poziție bine definită a cuvântului binar. Există două posibilități de poziționare. Dacă virgula este *aşezată după cifra de semn*, se operează cu numere fracționare, subunitare. Dacă virgula este aşezată *după cifra c.m.p.s.*, se operează cu numere întregi. În continuare se presupune această poziționare, considerând că se lucrează cu numere întregi.

În *forma cu virgulă mobilă*, fiecare număr este caracterizat prin două valori:

- *Mantisa*, care indică mărimea exactă a numărului într-un anumit domeniu;
- *Exponentul*, care indică ordinul de mărime a numărului, fiind puterea la care se ridică baza mantisei. Exponentul indică deci implicit poziția virgulei binare.

## 2.5. Reprezentarea numerelor în virgulă fixă

### 2.5.1. Reprezentarea numerelor cu semn

În continuare se vor nota cu $x$, $y$ numerele în reprezentarea binară obişnuită, la care se ataşează semnul. Un număr cu $n$ cifre de mărime se va scrie sub forma:

$$\pm x = \pm x_{n-1}\, x_{n-2} \ldots x_1\, x_0$$

De exemplu:

```
±6 = ±0110
```

Cu *X*, *Y* se vor nota numerele în reprezentarea din calculator, care conţin şi cifrele de semn:

$$X = x_n\, x_{n-1}\, x_{n-2}\, \ldots\, x_1\, x_0$$

După modul de exprimare a numerelor negative, există trei forme uzuale de reprezentare a numerelor cu semn în virgulă fixă:

- În mărime şi semn (`MS`)
- În complement faţă de 1 (`C1`)
- În complement faţă de 2 (`C2`)

Pentru toate formele, un număr pozitiv se exprimă în acelaşi fel:

$$X = 0 \cdot 2^n + \sum_{i=0}^{n-1} x_i 2^i = 0 \cdot 2^n + x \tag{1.21}$$

Numărul *pozitiv cu semn* se reprezintă deci adăugând cifra 0 de semn în faţa numărului fără semn:

$$X = 0\, x_{n-1}\, x_{n-2}\, \ldots\, x_1\, x_0$$

Considerăm pentru simplitate numere cu 4 biţi de mărime şi un bit de semn. De exemplu, numărul fără semn `5` se reprezintă prin:

```
 5       0101
```

iar numărul cu semn `+5` prin:

```
+5     0 0101
```

Cea mai simplă formă de reprezentare este cea *în mărime şi semn*. Un număr negativ reprezentat în mărime şi semn are expresia:

$$X = 1 \cdot 2^n + \sum_{i=0}^{n-1} x_i 2^i = 1 \cdot 2^n + x \tag{1.22}$$

Deci, numărul negativ se reprezintă prin adăugarea cifrei 1 de semn în faţa numărului fără semn:

$$X = 1\, x_{n-1}\, x_{n-2}\, \ldots\, x_1\, x_0$$

De exemplu, numărul `-5` se va reprezenta prin:

```
-5(MS) 1 0101
```

Există mai multe dezavantaje ale acestei reprezentări. Primul dezavantaj este că adunarea şi scăderea necesită circuite mai complexe. Al doilea dezavantaj este că există două reprezentări pentru valoarea `0`:

```
+0          0 0000
-0          1 0000
```

De aceea este mai dificil să se testeze dacă o valoare este 0 (o operaţie frecventă), decât în cazul în care ar exista o singură reprezentare.

În cazul reprezentării *în complement faţă de 1*, un număr negativ se reprezintă prin complementul faţă de 1 al numărului pozitiv cu aceeaşi valoare absolută. Complementul faţă de 1 al unui număr binar se obţine prin înlocuirea biţilor de 1 cu 0, şi a celor de 0 cu 1. Un număr negativ reprezentat în complement faţă de 1 are expresia:

$$X = 1 \cdot 2^n + \sum_{i=0}^{n-1} \overline{x_i} 2^i = 2^{n+1} - x - 1 \tag{1.23}$$

unde $\overline{x_i} = 1 - x_i$ reprezintă complementul faţă de 1 al cifrei $x_i$.

Complementul față de 1 al unui număr negativ se obține prin complementarea tuturor cifrelor numărului fără semn şi adăugarea cifrei de semn 1:

$$X = 1\,\overline{x}_{n-1}\,\overline{x}_{n-2}\ldots\overline{x}_1\,\overline{x}_0$$

De exemplu:

```
+5          0 0101
-5(C1)      1 1010
```

Există şi în acest caz două reprezentări pentru 0:

```
+0          0 0000
-0          1 1111
```

Un număr negativ reprezentat în *complement față de 2* are expresia:

$$X = 1 \cdot 2^n + \sum_{i=0}^{n-1} \overline{x}_i 2^i + 1 \cdot 2^0 = 2^{n+1} - x \qquad (1.24)$$

Complementul față de 2 al unui număr se poate obține în mai multe moduri. O posibilitate o reprezintă utilizarea relației de definiție (1.24). De exemplu, considerând $n = 4$, complementul față de 2 al numărului –5 va fi:

$2^{4+1} - 5 = 32 - 5 = 27 = 11011$

O altă posibilitate este obținerea complementului față de 2 în două etape:

1. Se obține complementul față de 1.
2. Se consideră rezultatul ca un întreg fără semn, la care se adună valoarea 1.

De exemplu:

```
+5          0 0101
-5(C1)      1 1010 +
                  1
            ------
-5(C2)      1 1011
```

Practic, complementul față de 2 al unui număr se poate determina pornind de la numărul pozitiv cu semn, astfel:

1. Se scriu cifrele numărului începând cu cifra c.m.p.s., neschimbate, până la primul 1 inclusiv.
2. Se complementează cifrele întâlnite în continuare.

Există o singură reprezentare pentru 0 în C2. În plus, operațiile de adunare şi scădere se efectuează cel mai simplu în această reprezentare. Se poate arăta că reprezentarea prin C2 conduce la aflarea valorii reale a numărului, dacă cifra de semn se consideră negativă.

În Tabelul 1.5 se prezintă reprezentarea unor numere cu 4 biți de mărime şi un bit de semn în MS, C1 şi C2.

**Tabelul 1.5.** Diferite moduri de reprezentare a unor numere cu semn.

| X | MS | C1 | C2 |
|---|---|---|---|
| +15 | 0 1111 | 0 1111 | 0 1111 |
| +14 | 0 1110 | 0 1110 | 0 1110 |
| … | | | |
| +2 | 0 0010 | 0 0010 | 0 0010 |
| +1 | 0 0001 | 0 0001 | 0 0001 |
| 0 | 0 0000<br>1 0000 | 0 0000<br>1 1111 | 0 0000 |
| -1 | 1 0001 | 1 1110 | 1 1111 |
| -2 | 1 0010 | 1 1101 | 1 1110 |
| … | | | |
| -14 | 1 1110 | 1 0001 | 1 0010 |
| -15 | 1 1111 | 1 0000 | 1 0001 |
| -16 | - | - | 1 0000 |

În cazul reprezentării în MS şi C1, gama numerelor care pot fi exprimate prin *n* biţi de mărime este:

$$-(2^n - 1) \le x \le 2^n - 1$$

Pentru reprezentarea în C2, această gamă este:

$$-2^n \le x \le 2^n$$

De exemplu, pentru numere de un octet (7 cifre de mărime şi o cifră de semn), gama pentru reprezentarea în MS şi C1 este:

$$-(2^7 - 1) \le x \le 2^7 - 1$$

adică:

$$-127 \le x \le 127$$

iar în C2:

$$-128 \le x \le 127$$

### 2.5.2. Reguli de deplasare ale numerelor cu semn

De multe ori sunt necesare operaţii de deplasare a numerelor cu semn. Aceste deplasări trebuie efectuate astfel încât să se modifice numai valoarea numerelor, nu şi semnul. Deplasarea la stânga cu o poziţie este echivalentă înmulţirii cu 2, iar deplasarea la dreapta este echivalentă împărţirii cu 2 (înmulţirii cu $2^{-1}$).

Pentru stabilirea regulilor de deplasare, se consideră exemplele din Tabelul 1.6.

**Tabelul 1.6.** Stabilirea regulilor de deplasare ale numerelor cu semn.

| | | MS | C1 | C2 |
|---|---|---|---|---|
| X | + 6 | 0 0110 | 0 0110 | 0 0110 |
| X · 2 | +12 | 0 1100 | 0 1100 | 0 1100 |
| X · $2^{-1}$ | + 3 | 0 0011 | 0 0011 | 0 0011 |
| X | - 6 | 1 0110 | 1 1001 | 1 1010 |
| X · 2 | -12 | 1 110⓪ | 1 001① | 1 010⓪ |
| X · $2^{-1}$ | - 3 | 1 ⓪011 | 1 ①100 | 1 ①101 |

La deplasare participă numai cifrele de mărime ale numerelor.

- În cazul deplasării *numerelor pozitive*, în poziţiile rămase libere după deplasarea la stânga sau la dreapta se introduc cifre de 0.
- La numerele negative reprezentate în MS, în poziţiile rămase libere după o deplasare la stânga sau la dreapta se introduc cifre de 0.
- La numerele negative reprezentate în C1, în poziţiile rămase libere după o deplasare la stânga sau la dreapta se introduc cifre de 1.
- La numerele negative reprezentate în C2, în poziţiile rămase libere după o deplasare la stânga se introduc cifre de 0, iar după o deplasare la dreapta se introduc cifre de 1 (deci se repetă semnul numărului).

### 2.5.3. Operaţii cu numere reprezentate în virgulă fixă

#### 2.5.3.1. Adunarea numerelor reprezentate în C2

Metodele de adunare şi scădere a numerelor reprezentate în C2 demonstrează avantajele acestei reprezentări. Aceste operaţii se pot efectua ca şi în cazul numerelor fără semn.

Considerăm câteva exemple de adunare. În primele exemple, numerele sunt *de acelaşi semn*.

```
a)     + 9          0 1001
       + 5          0 0101
       +14          0 1110
```

Rezultatul este corect.

```
b)     + 9          0 1001
       +11          0 1011
       +20          1 0100
```

Se obţine un număr negativ, deci rezultatul este incorect. Deoarece $+20 > 2^4 = 16$, se depăşeşte capacitatea de reprezentare a rezultatului.

```
c)     - 9          1 0111
       - 5          1 1011
       -14        ①1 0010
```

Rezultatul este corect, deoarece $-14 > -2^4 = -16$. Se obţine un număr negativ reprezentat în C2. Apare un transport de la cifra de semn, care se neglijează.

```
d)     - 9          1 0111
       -11          1 0101
       -20        ①0 1100
```

Rezultatul este eronat, deoarece se obţine un număr pozitiv.

În exemplele următoare, numerele sunt *de semne contrare*.

```
e)     + 7          0 0111
       - 4          1 1100
       + 3        ①0 0011

f)     - 7          1 1001
       + 4          0 0100
       - 3          1 1101
```

În cazul numerelor de semne contrare, rezultatul este întotdeauna corect, deoarece nu poate apare depăşire de capacitate.

Din exemplele prezentate, se poate formula *regula generală de adunare* a două numere reprezentate în C2.

Se adună numerele bit cu bit, inclusiv biţii de semn, care sunt trataţi la fel cu biţii de mărime, şi se ignoră eventualul transport de la bitul de semn. Dacă rezultatul este negativ, apare ca un număr reprezentat în C2.

Dacă rezultatul este mai mare în valoare absolută decât valoarea maximă care poate fi reprezentată în registru, apare *depăşire* (exemplele *b* şi *d*). La apariţia depăşirii, UAL trebuie să semnaleze acest fapt, astfel încât rezultatul să nu fie utilizat.

Se observă că depăşirea poate apare indiferent dacă există sau nu transport de la cifra de semn. Pentru *detectarea apariţiei depăşirii*, se poate aplica următoarea regulă simplă:

> La adunarea a două numere de acelaşi semn, apare depăşire dacă şi numai dacă rezultatul are semn contrar semnului numerelor.

#### 2.5.3.2. Scăderea numerelor reprezentate în C2

Scăderea a două numere reprezentate în C2 se poate efectua fie prin metoda scăderii directe, dacă se dispune de scăzătoare elementare, fie prin metoda adunării complementului faţă de 2, dacă se dispune numai de sumatoare elementare. Se poate enunţa următoarea regulă de scădere:

> Pentru scăderea unui număr (scăzător) dintr-un altul (descăzut), se calculează complementul faţă de 2 al scăzătorului şi se efectuează adunarea acestuia la descăzut.

Se consideră următorul exemplu.

```
D      + 7          0 0111
S      - 4          1 1100
S'     -(-4)        0 0100

D      + 7          0 0111
S'     + 4          0 0100
       +11          0 1011
```

Regula pentru detectarea depăşirii poate fi enunţată astfel:

> La scăderea a două numere de semne contrare apare depăşire dacă şi numai dacă rezultatul are acelaşi semn cu scăzătorul.

## 3. Desfăşurarea lucrării

3.1. Se vor converti următoarele numere zecimale în binar, octal şi hexazecimal:

a) 78; b) 125; c) 125,34; d) 12,38

3.2. Se vor converti următoarele numere binare în octal şi hexazecimal:

a) 1010101; b) 1001,011; c) 1110011,10

3.3. Se vor converti următoarele numere în binar:

a) $125_8$; b) ; c) $470_8$; d) $3E_{16}$; e) $12A_{16}$; f) $45,E1_{16}$

3.4. Se vor converti următoarele numere binare în zecimal:

a) 1010101; b) 0,1101; c) 10,1011

3.5. Se vor efectua câte două exemple de adunare şi de scădere cu numere binare şi hexazecimale.

3.6. Se vor reprezenta următoarele numere negative în MS, C2 şi C1:

a) –114; b) –53,25; c) –75,18

3.7. Se vor efectua câte două exemple de adunare şi scădere cu numere reprezentate în C2.

3.8. Se va deduce regula de adunare a numerelor reprezentate în MS. Se vor considera cazuri similare cu cele prezentate la adunarea numerelor în C2.