IV.4. REPREZENTAREA INTERNĂ A NUMERELOR

IV.4.1.
SCRIEREA POZIȚIONALĂ

Scrierea poziţională

- Este tot o reprezentare!
  - 72018 nu este un număr, ci reprezentarea unui număr
- Inventată de arabi/indieni
  - Scrierea romană nu permite algoritmi eficienţi de calcul
- Factor implicit ataşat fiecărei poziţii din reprezentare
- Esenţială în arhitectura calculatoarelor
  - Exemplu: sumatorul serial din sumatoare complete

Baze de numerație

- Orice număr natural d>1
  - Cu un singur simbol nu se pot folosi factori impliciți, ci juxtapunere + numărare
- Mulțimea cifrelor în baza d: {0,1,…,d-1}
- Calculatorul lucrează în baza d=2
  - Estimări analitice şi probabiliste: bazele în care se pot face cel mai rapid calcule sunt 2 şi 3
  - Tehnic: 2 cifre cel mai uşor de realizat
  - Teoretic: baza 2 se poate "scufunda" în logica booleană
    - ca simboli şi ca operații

Limite

Dacă s-ar reprezenta numerele în baza 2 fără semn (pozitive), atunci:

- Numărul maxim reprezentabil pe un octet ar fi $255 = 2^8-1$
- Numărul maxim reprezentabil pe doi octeți ar fi $65535 = 2^{16}-1$
- Numărul maxim reprezentabil pe patru octeți ar fi $4294967295 = 2^{32}-1$

Scrierea poziţională

- Baza d, $d \in N^* - \{1\}$:

  pentru $a_i \in \{0,1,\ldots,d-1\}$, $i=-m,\ldots,n-1$

  $$\pm (a_{n-1}a_{n-2}\ldots a_1 a_0, a_{-1}\ldots.a_{-m})_{(d)} =$$
  $$\pm \sum_{i=-m}^{n-1} (a_i \times d^i)_{(10)}$$

  - a_i = valoarea celei de-a i+1[a] cifre de la stânga virgulei
    - » i=0..n-1
  - a_{-j} = valoarea celei de-a j[a] cifre de la dreapta virgulei (j>0)
    - » j=1..m
- d^i este factorul implicit pentru poziţia i
  - Se ridică la puterea i
    - d^{+1} pentru partea întreagă
    - d^{-1} pentru partea fracţionară

Baza d → baza 10

- Formula de mai sus este şi formula trecerii din baza d în baza 10
- Partea întreagă de n-1 cifre
- Partea fracționară de m cifre

Un exemplu

- $FA2,B_{(16)} =$

  $15 \times 16^2 + 10 \times 16^1 + 2 \times 16^0 + 11 \times 16^{-1}$

  $= 3840 + 60 + 2 + 11 / 16 =$

  $4002,6875_{(10)}$

Baza 10 → baza d

- $813,65_{(10)} = 1100101101,10(1001)_{(2)}$
  - » 813 / 2 = 406 + 1 / 2 → 1 (LSB_i)
  - » 406 / 2 = 203 + 0 / 2 → 0
  - » 203 / 2 = 101 + 1 / 2 → 1
  - » 101 / 2 = 50 + 1 / 2 → 1
  - » 50 / 2 = 25 + 0 / 2 → 0
  - » 25 / 2 = 12 + 1 / 2 → 1
  - » 12 / 2 = 6 + 0 / 2 → 0
  - » 6 / 2 = 3 + 0 / 2 → 0
  - » 3 / 2 = 1 + 1 / 2 → 1
  - » 1 / 2 = 0 + 1 / 2 → 1 (MSB_i)
  - » 0,65 / 2^{-1} = 1 + 0,3 → 1 (MSB_f)
  - » 0,3 / 2^{-1} = 0 + 0,6 → 0
  - » 0,6 / 2^{-1} = 1 + 0,2 → 1
  - » 0,2 / 2^{-1} = 0 + 0,4 → 0
  - » 0,4 / 2^{-1} = 0 + 0,8 → 0
  - » 0,8 / 2^{-1} = 1 + 0,6 → 1 (LSB_f)
  - » …. (perioadă)

Aproximarea reprezentării

- Dacă numărul are mai multe cifre la partea fracționară decât admite codificarea, atunci există o aproximare
  - de cel mult 2^{-k}, dacă m=k
  - dacă există la partea întreagă mai multe cifre decât se pot reprezenta, atunci se produce **depăşire**

Conversii între baze care sunt puteri ale aceluiaşi număr

- $d_1 = 8 = 2^3$; $d_2 = 16 = 2^4$
- $703{,}102_{(8)} =$

  $= 111\ 000\ 011\ ,\ 001\ 000\ 010\ _{(2)} =$

  $= 0001\ 1100\ 0011\ ,\ 0010\ 0001\ 0000\ _{(2)} =$

  $= 1C3{,}21_{(16)}$

IV.4.2. REPREZENTĂRILE BCD ŞI ÎN EXCES

- Coduri poziționale
  - Sisteme de numerație bazate pe scrierea pozițională
    - de exemplu, codul BCD
- Coduri non-poziționale
    - De exemplu, codul Excess-k
      - » k=3 etc.
      - » în general, $k=2^p-1$
- Pentru aplicații tip business, numerele se pot reprezenta ca şiruri de cifre în baza 10, fiecare cifră fiind reprezentată pe 4 biți
    - BCD, Excess

Codurile BCD şi Excess-3

Zecimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
...	...	...
7	0111	1010
8	1000	1011
9	1001	1100

- $1413_{(10)} = 0001\ 0100\ 0001\ 0011_{(BCD)}$

Adunarea BCD

$5 = 0101 +$

$3 = \underline{0011}$

$8_{(10)} = 1000 = 8_{(BCD)}$

$5 = 0101 +$

$8 = \underline{1000}$

$13_{(10)} = 1101$

$\neq 13_{(BCD)} =$

0001 0011

- Problemele apar atunci când suma cifrelor depăşeşte 9

Adunarea BCD

Soluție: se adună 6 (0110) atunci când suma depăşeşte 9.

Temă: De ce?

$5 = 0101 +$

$8 = 1000$

1101

$6 = 0110 +$

$1\ 0011 = 1\ 3_{(BCD)}$

$9 = 1001 +$

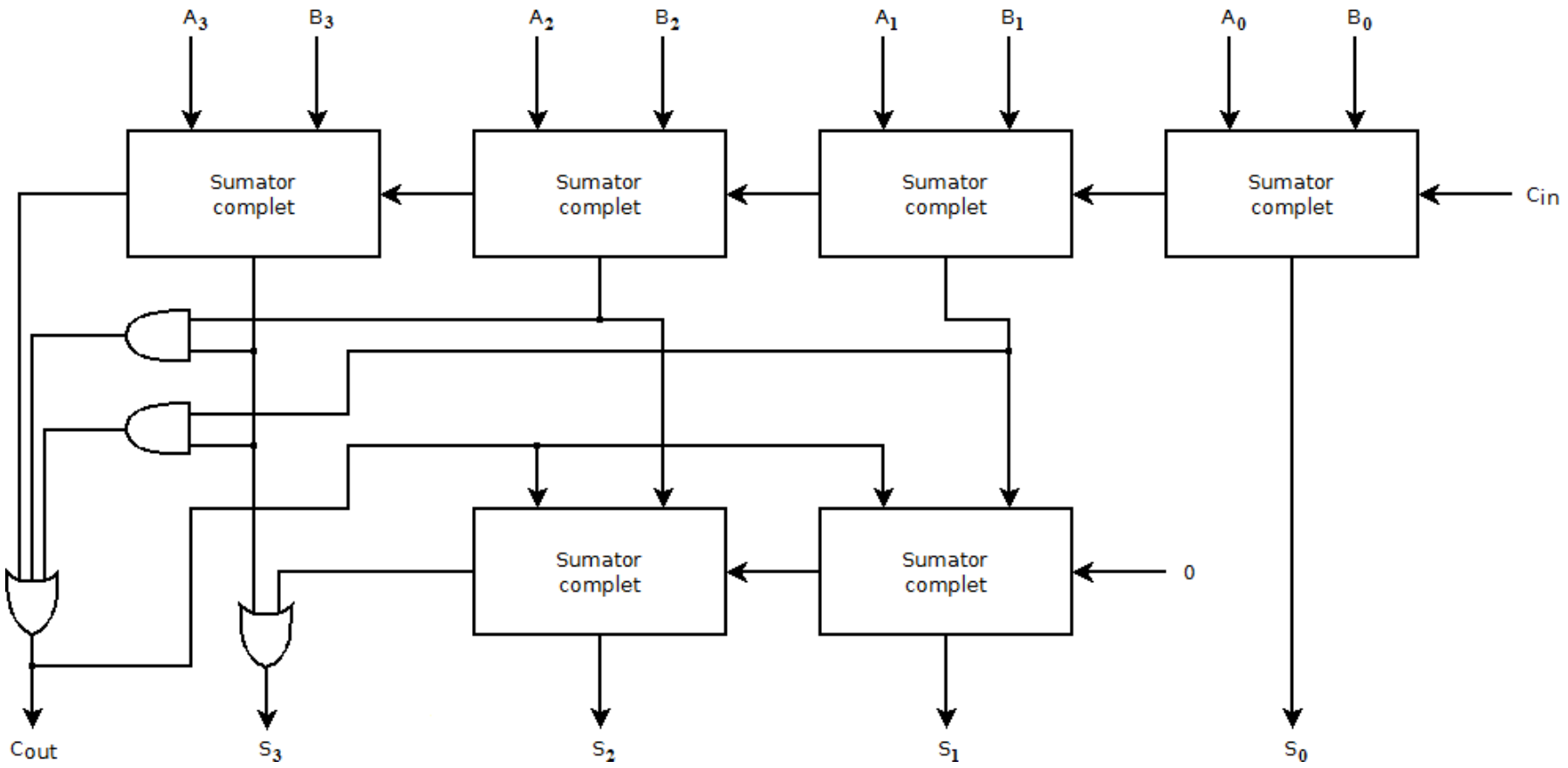
$7 = 0111$

$16_{(10)} = 1\ 0000_{(2)} \neq 16_{(BCD)}$

$6 = 0110$

$1\ 0110 = 1\ 6_{(BCD)}$

Sumator BCD



Se adună 0110 la sumă dacă ea depăşeşte 1001 (11XX sau 1X1X)

IV.4.3.
Reprezentarea numerelor întregi:
aritmetica în virgulă fixă

- Omogenitate
    - Nu se reprezintă semne speciale ("+", "-", ".", ",")
- Semnul este dat de **un bit**
    - nu de codificarea pe mai mulți biți a unui caracter special (+ sau -)
    - 0 pentru plus
    - 1 pentru minus
    - excepție: reprezentarea "cu exces"
- Pentru virgulă se ştie **poziția**
    - dar nu se reprezintă caracterul
    - aceeaşi poziție a virgulei pentru toate numerele: **virgulă fixă** ("aritmetică întreagă")
    - poziție diferită de la număr la număr: **virgulă mobilă** ("aritmetică flotantă")
- Esențial pentru portabilitate: standardizarea aritmeticii (atât întreagă, cât şi flotantă)
        - În toate implementările
            - » Funcțiile elementare definite şi calculate la fel
            - » Cazurile de excepție tratate la fel

Codificări în virgulă fixă

- $d = 2 \rightarrow a_i \in B = \{0,1\}$
- Obiectiv: eficiența calculului
- Codificare / decodificare facilă (omogenitate)
- Algoritmi eficienți
  - În particular, un singur algoritm pentru adunare şi scădere
- Codificările în virgulă fixă se fac pe **n+m** biți
  - $m \geq 0$; m=0 – numere întregi
  - $n \geq 1$; n=1 – numere subunitare
- Codificări în virgulă fixă
  - Cantitate şi semn
  - Complement față de 1
    - » în general, față de cifra maximă
  - Complement față de 2
    - » în general, față de bază

Codificări redundante

- Reprezentarea numerelor pozitive coincide la cele trei codificări.
- O codificare se numeşte *redundantă* dacă există numere care au două reprezentări diferite.
- În codificările în virgulă fixă folosite, singurul număr ce poate avea două reprezentări este 0.

IV.4.4.
Reprezentarea prin modul și semn

Reprezentarea prin modul şi semn

- A+S
- $Val_{A+S}^{n,m}(a_{n-1}a_{n-2}\ldots.a_1a_0a_{-1}\ldots..a_{-m}) =$

$$\begin{cases} a_{n-2} \times 2^{n-2} + \ldots + a_{-m} \times 2^{-m}, & \text{dacă } a_{n-1} = 0 \\ -(a_{n-2} \times 2^{n-2} + \ldots + a_{-m} \times 2^{-m}), & \text{dacă } a_{n-1} = 1 \end{cases}$$

- Coincide cu scrierea în baza 2
  - dar semnul este un bit şi virgula implicită

Reprezentarea prin modul şi semn

- Pe n+m biți există 2^{n+m} reprezentări diferite (şiruri diferite de biți)
- Ele corespund la 2^{n+m} -1 numere diferite
  - Redundantă, căci $0 = val_{A+S}^{n,m}(00\ldots0) = val_{A+S}^{n,m}(10\ldots0)$
- Cel mai mic număr reprezentabil este

  $$min_{A+S}^{n,m} = val_{A+S}^{n,m}(11\ldots1) = -(2^{n-1}-2^{-m})$$
- Cel mai mare număr reprezentabil este

  $$Max_{A+S}^{n,m} = val_{A+S}^{n,m}(01\ldots1) = 2^{n-1}-2^{-m}$$
- Intervalul pe care se află numerele reprezentabile este $[-(2^{n-1}-2^{-m}); +(2^{n-1}-2^{-m})]$

Reprezentarea prin modul şi semn

- Numerele reprezentabile **exact** sunt cele începând cu min = $-(2^{n-1}-2^{-m})$, cu pasul 2^{-m}
- Celelalte numere din interval se reprezintă aproximativ, cu eroare de cel mult 2^{-m}
- Precizia reprezentării este 2^{-m}
  - pentru numere întregi, precizia este 1
- Pentru n+m fixat
  - creşterea magnitudinii duce la aproximare mai slabă
  - precizie mai bună duce la magnitudine scăzută

Exemple

- $Val_{A+S}^{8,0}(00110011) = 51$

$00110011 \rightarrow + (2^0 + 2^1 + 2^4 + 2^5) = 51$

- $Val_{A+S}^{6,2}(00110011) = 12{,}75 = 51 : 2^2$

$00110011 \rightarrow + (2^{-2} + 2^{-1} + 2^2 + 2^3) = 12{,}75$

- $Val_{A+S}^{4,4}(00110011) = 3{,}1875 = 51 : 2^4$

$00110011 \rightarrow + (2^{-4} + 2^{-3} + 2^0 + 2^1) = 3{,}1875$

Exemple

- $Val_{A+S}^{8,0}(10110011) = -51$

$10110011 \rightarrow -(2^0 + 2^1 + 2^4 + 2^5) = -51$

- $min_{A+S}^{8,0} = val_{A+S}^{8,0}(11111111) = -(2^7-2^0) = -(128-1) = -127$
- $max_{A+S}^{8,0} = val_{A+S}^{8,0}(01111111) = 2^7-2^0 = 128-1 = 127$
- [-127; 127] → 255 numere, din 1 în 1
- $Val_{A+S}^{4,4}(10110011) = -3,1875$

  $10110011 \rightarrow -(2^{-4} + 2^{-3} + 2^0 + 2^1) = -3,1875$
- $min_{A+S}^{4,4} = val_{A+S}^{4,4}(11111111) = -(2^3-2^{-4}) = -7,9375$
- $max_{A+S}^{4,4} = val_{A+S}^{4,4}(01111111) = 2^3-2^{-4} = 8-0,0625 = 7,9375$
- [-7,9375; 7,9375] → 255 numere din 0,0625 în 0,0625

Operații A+S

- Algoritmi de complexitate relativ mare
  - Adunare / scădere
    - Stabilirea semnului rezultatului (comparație lexicografică)
    - Implementarea algoritmilor uzuali de adunare / scădere manuală
      - Incluzând "împrumuturi" etc.
  - Înmulțirea / împărțirea – analog celor manuale

IV.4.5.
Reprezentarea în complement față de 1

Reprezentarea în complement față de 1

- Complement față de 1: C_1
- $Val_{C1}^{n,m}(a_{n-1}a_{n-2}\ldots a_1a_0a_{-1}\ldots a_{-m}) =$

$$\begin{cases} a_{n-2} \times 2^{n-2} + \ldots + a_{-m} \times 2^{-m}, & \text{dacă } a_{n-1} = 0 \\ (a_{n-2} \times 2^{n-2} + \ldots + a_{-m} \times 2^{-m}) - (2^{n-1} - 2^{-m}), & \text{dacă } a_{n-1} = 1 \end{cases}$$

- Temă: negativ pentru $a_{n-1}=1$
  - Deci a_{n-1} reprezintă semnul

Reprezentarea în complement față de 1

- Cele 2^{n+m} reprezentări diferite (şiruri diferite de biți) corespund la $2^{n+m}-1$ numere diferite
  - Redundantă: 0 poate fi reprezentat şi ca număr negativ
- Cel mai mic număr reprezentabil este

$$\min_{C1}^{n,m} = val_{C1}^{n,m}(10\ldots0) = -(2^{n-1} - 2^{-m})$$

- Cel mai mare număr reprezentabil este

$$Max_{C1}^{n,m} = val_{C1}^{n,m}(01\ldots1) = 2^{n-1}-2^{-m}$$

- Intervalul pe care se află numerele reprezentabile este deci $[-(2^{n-1}-2^{-m}) ; +(2^{n-1}-2^{-m})]$

Reprezentarea în complement față de 1

- $Val_{C1}^{8,0}(00110011) = 51$

$00110011 \rightarrow + (2^0 + 2^1 + 2^4 + 2^5) = 51$

- $Val_{C1}^{6,2}(00110011) = 12{,}75 = 51 : 2^2$

$00110011 \rightarrow + (2^{-2} + 2^{-1} + 2^2 + 2^3) = 12{,}75$

- $Val_{C1}^{4,4}(00110011) = 3{,}1875 = 51 : 2^4$

$00110011 \rightarrow + (2^{-4} + 2^{-3} + 2^0 + 2^1) = 3{,}1875$

Reprezentarea în complement față de 1

- $Val_{C1}^{8,0}(10110011) = -76$

$10110011 \rightarrow (2^0 + 2^1 + 2^4 + 2^5) - (2^7 - 2^0) = 51 - 127 = -76$

- $min_{C1}^{8,0} = val_{C1}^{8,0}(10000000) = 0 - (2^7 - 2^0) = 0 - 127 = -127$
- $max_{C1}^{8,0} = val_{C1}^{8,0}(01111111) = 2^7 - 2^0 = 128 - 1 = 127$
- [-127; 127] → 255 numere, din 1 în 1.
- $Val_{C1}^{4,4}(10110011) = -4,75 = -76 : 2^4$

$10110011 \rightarrow (2^{-4} + 2^{-3} + 2^0 + 2^1) - (2^3 - 2^{-4}) = 3,1875 - 7,9375 = -4,75$

- $min_{C1}^{4,4} = val_{C1}^{4,4}(10000000) = 0 - (2^3 - 2^{-4}) = -7,9375 = -127 : 2^4$
- $max_{C1}^{4,4} = val_{C1}^{4,4}(01111111) = 2^3 - 2^{-4} = 8 - 0,0625 = 7,9375 = 127 : 2^4$
- [-7,9375; 7,9375] → 255 numere din 0,0625 în 0,0625

C_1 - complementare

- Dată reprezentarea lui q, se poate afla automat reprezentarea lui –q?
  - Dacă da, atunci scăderea p-q devine adunare, după generarea automată a reprezentării lui –q: p - q = p + (-q)
- Reprezentarea lui –q: complementul față de 1 al reprezentării lui q
- Exemplu: q = -76 = $Val_{C1}^{8,0}$ (10110011)

  q = 76 = $Val_{C1}^{8,0}$ (01001100) = 64+8+4
- Din cauza redundanței şi a adunării preciziei (în sumă algebrică, de două ori), algoritmii de calcul în C_1 sunt mai puțin eficienți decât cei în C_2
- De aceea, reprezentarea cvasi-general utilizată este C_2

IV.4.6.
Reprezentarea în complement față de 2

Complement

- Fie baza d > 1
- Complementul unei cifre:
  - Pentru o cifră a $\in \{0,1,\ldots d-1\}$

    $c_d(a) = (d - 1) - a$
  - Pentru d = 2 şi b $\in \{0,1\}$:

    $c_2(b) = (2 - 1) - b = 1 - b \rightarrow \overline{b}$
- Dar complementul unui şir de biți?

Complement față de bază şi față de cifra maximă

- Extinderea definiției complementului la un şir de biți se poate face în două moduri:
  - Conform definiției pentru un bit (complement față de cifra maximă)

    $C_1(1011) = 0100$ $C_1(0100) = 1011$
  - Adaptând definiția pentru şiruri (complement față de bază)

    $C_2(1011) = 0100 + 0001 = 0101$

    $C_2(0101) = 1010 + 0001 = 1011$

Reprezentarea în complement față de 2

- Cel mai frecvent utilizată
- Adunarea şi scăderea cu acelaşi algoritm / circuit
- Testarea automată a depăşirilor

C_2 - definiție

- Complement față de 2: C_2
- $Val_{C2}^{n,m}(a_{n-1}a_{n-2}\ldots a_1a_0a_{-1}\ldots..a_{-m}) =$

$$\begin{cases} a_{n-2} \times 2^{n-2} + \ldots + a_{-m} \times 2^{-m}, & \text{dacă } a_{n-1} = 0 \\ (a_{n-2} \times 2^{n-2} + \ldots + a_{-m} \times 2^{-m}) - 2^{n-1}, & \text{dacă } a_{n-1} = 1 \end{cases}$$

- Temă: strict negativ pentru $a_{n-1}=1$
  - Deci a_{n-1} reprezintă semnul

Reprezentarea în complement față de 2

- Cele 2^{n+m} reprezentări diferite (şiruri diferite de biți) corespund la 2^{n+m} numere diferite
  - Neredundantă: $0 = Val_{C2}^{n+m}(00\ldots0)$
  - Temă: 0 nu poate fi reprezentat ca număr negativ
- Cel mai mic număr reprezentabil este

  $min_{C2}^{n,m} = val_{C2}^{n,m}(10\ldots0) = -\ 2^{n-1}$
- Cel mai mare număr reprezentabil este

  $max_{C2}^{n,m} = val_{C2}^{n,m}(01\ldots1) = 2^{n-1} - 2^{-m}$
- Intervalul pe care se află numerele reprezentabile este deci $[-\ 2^{n-1}; +\ (2^{n-1} - 2^{-m})]$

Reprezentarea în complement față de 2

- Numerele reprezentabile **exact** sunt cele începând cu min = - 2^{n-1}, cu pasul 2^{-m}
- Celelalte numere din interval se reprezintă aproximativ, cu eroare de cel mult 2^{-m}
- **Precizia** reprezentării este 2^{-m}
  - pentru numere întregi, m=0, deci precizia este 1
- Pentru n+m fixat
  - creşterea magnitudinii duce la aproximare mai slabă
  - precizie mai bună duce la magnitudine scăzută

Reprezentarea în complement față de 2

- $Val_{C2}^{8,0}(00110011) = 51$

$00110011 \rightarrow + (2^0 + 2^1 + 2^4 + 2^5) = 51$

- $Val_{C2}^{6,2}(00110011) = 12{,}75 = 51 : 2^2$

$00110011 \rightarrow + (2^{-2} + 2^{-1} + 2^2 + 2^3) = 12{,}75$

- $Val_{C2}^{4,4}(00110011) = 3{,}1875 = 51 : 2^4$

$00110011 \rightarrow + (2^{-4} + 2^{-3} + 2^0 + 2^1) = 3{,}1875$

Reprezentarea în complement față de 2

- $Val_{C2}^{8,0}(10110011) = -77$

10110011 → $(2^0 + 2^1 + 2^4 + 2^5) - 2^7 = 51 - 128 = -77$

- $min_{C2}^{8,0} = val_{C2}^{8,0}(10000000) = 0 - 2^7 = 0 - 128 = -128$
- $max_{C2}^{8,0} = val_{C2}^{8,0}(01111111) = 2^7 - 2^0 = 128 - 1 = 127$
- [-128; 127] → 256 numere, din 1 în 1
- $Val_{C2}^{4,4}(10110011) = -4,8125 = -77 : 2^4$
- 10110011 → $(2^{-4} + 2^{-3} + 2^0 + 2^1) - 2^3 = 3,1875 - 8 = -4,8125$
- $min_{C2}^{4,4} = val_{C2}^{4,4}(10000000) = 0 - 2^3 = -8 = -128 : 2^4$
- $max_{C2}^{4,4} = val_{C2}^{4,4}(01111111) = 2^3 - 2^{-4} = 8 - 0,0625 = 7,9375 = 127 : 2^4$
- [-8; 7,9375] → 256 numere din 0,0625 în 0,0625

C_2 - complementare

- Dată reprezentarea lui q, se poate afla automat reprezentarea lui -q?
- Dacă da, atunci scăderea p-q devine adunare, după generarea automată a reprezentării lui -q: p - q = p + (-q)
- Reprezentarea lui -q: complementul față de 2 al reprezentării lui q
- Exemplu: q = -77 = $Val_{C2}^{8,0}(10110011)$

  -q = 77 = $Val_{C2}^{8,0}(01001100 + 00000001)$

  $Val_{C2}^{8,0}(01001101) = 64 + 8 + 4 + 1$

Temă

- Reprezentarea în C_2 pe N biți a numărului întreg negativ q este de fapt reprezentarea pe N biți a numărului $q + 2^N = 2^N - |q|$
- $a + \overline{a} = 11\ldots.11 \rightarrow -1$, deci $(-a) = \overline{a} + 1$
  - $\overline{a}$ notează negația bit cu bit a reprezentării numărului a
  - a notează atât numărul, cât şi reprezentarea sa; se foloseşte aici implicit faptul că numerele pozitive se reprezintă ca în baza 2

Reprezentări în virgulă fixă, ^{4,0}

şirul de biți	**A+S**	**C_1**	**C_2**	**XS-7**
0000	+0	+0	0	-7
0001	+1	+1	+1	-6
0010	+2	+2	+2	-5
0011	+3	+3	+3	-4
0100	+4	+4	+4	-3
0101	+5	+5	+5	-2
0110	+6	+6	+6	-1
0111	+7	+7	+7	0
1000	-0	-7	-8	+1
1001	-1	-6	-7	+2
1010	-2	-5	-6	+3
1011	-3	-4	-5	+4
1100	-4	-3	-4	+5
1101	-5	-2	-3	+6
1110	-6	-1	-2	+7
1111	-7	-0	-1	+8