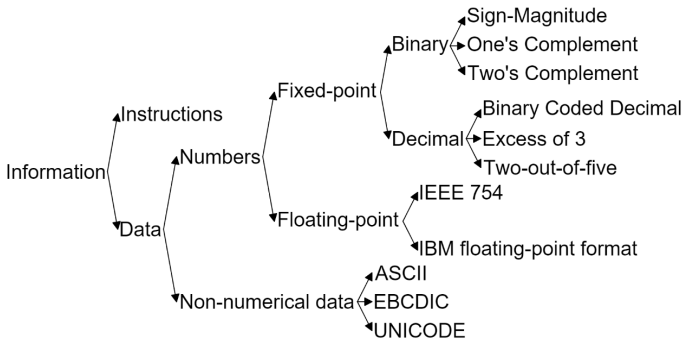
09.10.2024

Arhitectura calculatoarelor (curs 2 - S2)

1. Reprezentarea numerelor în sistemele de calcul

1.1 - Clasificarea informațiilor

Clasificarea informației:



Bit: binar digit

- byte
- cuvinte

Coduri pentru date non-numerice

- American Standard Code for Information Interchange (ASCII)
- Extended Binary Coded Decimal Interchange Code (EBCDIC)
- UNICODE

Numere de virgulă fixă

- întregi
- fracționare

Numere de virgulă mobilă:

- reprezentare aproximativă
  - Se consideră valoarea $1e20$
    - $(3.14 + 1e20) - 1e20 = 0$
    - $3.14 + (1e20 - 1e20) = 3.14$
  - $\Rightarrow$ Adunarea numerelor de virgulă mobilă: nu este asociativă!

1.2. Reprezentarea numerelor de virgulă fixă

Numărul X reprezentat în baza r:

- $X = \underline{x_{n-1}x_{n-2}\cdots x_1x_0}.\underline{x_{-1}x_{-2}\cdots x_{-m}}$
  - parte întreagă: $x_{n-1}x_{n-2}\cdots x_1x_0$
  - parte fracționară: $x_{-1}x_{-2}\cdots x_{-m}$
  - Most Significant Bit (MSB) (cel mai semnificativ bit): x_{n-1}
  - Least Significant Bit (LSB) (cel mai puțin semnificativ bit): x_{-m}

- parte reală/întreagă
- parte fracționară

Valoarea lui X reprezentată în baza r:

- $X = \sum_{j=-m}^{n-1} x_j * r^j$, $0 \leq x_j < r$
  - r^j: ponderea cifrei x_j
  - reprezentare **pozițională**

$\vartheta = 2$ ⟶ sistem de reprezentare binar

ex:

	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
$103_{(10)}$	1	1	0	0	1	1	1	$_{(2)}$
$68_{(10)}$	1	0	0	0	1	0	0	$_{(2)}$

Poziția virgulei binare:

- convenție pentru reprezentarea numerelor întregi și fracționare
  - evită codificarea poziției virgulei în reprezentarea numerelor
  - rămân mai mulți biți pentru precizie

Pentru nr. întregi, virgula binară se află la dreapta celui mai puțin semnificativ bit (LSB)

$$X = x_{n-1}x_{n-2}\dots x_1x_0. = \sum_{i=0}^{n-1} x_i \cdot 2^i \quad , x \text{ pe } n \text{ biți}$$

Pentru nr. fracționare, punctul binar se află la stânga celui mai semnificativ bit (MSB)

$$X = .x_{n-1}x_{n-2}\dots x_1x_0 = \sum_{i=0}^{n-1} x_i \cdot 2^{i-n} \quad , x \text{ pe } n \text{ biți}$$

ex: $.1101 = \frac{8}{\frac{1}{2}} + \frac{4}{\frac{1}{4}} + \frac{1}{16} = \frac{13}{16} = \frac{1101_{(2)}}{2^4}$

$$\frac{103}{128}_{(10)} = .1\,1\,0\,0\,1\,1\,1_{(2)}$$

weights: $2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ 2^{-6}\ 2^{-7}$

$$\frac{17}{32}_{(10)} = .1\,0\,0\,0\,1_{(2)}$$

Valoarea unui numar fractionar de virgula fixa se obtine impartind valoarea intreaga a aceleasi configuratii la 2^(nr de poz binare a partii fractionare).

ex: $.1100_{(2)} = \frac{1100_{(2)}}{2^4} = \frac{12}{16} = \frac{3}{4}$

!! $.123000 = .123$

1.2.1. Semn-mărime

MSB codifică semnul numărului. Convenția de semn:

- numerele pozitive au bitul de semn 0
- numerele negative au bitul de semn 1

⚠ adunare doar cu numere pozitive (rezultat eronat altfel)
⚠ facilitează înmulțirea → $x - y = x + (-y)$
⚠ complexitate hardware moderată

$$X = \underbrace{x_{n-1}}_{\text{semnul lui } x} \quad \underbrace{x_{n-2}x_{n-3}\dots x_1x_0}_{\text{magnitudinea lui } x}$$

ex: $+103_{(10)} = 0\ 1100111_{(SM)}$

$-103_{(10)} = 1\ 1100111_{(SM)}$

Interval valoric

Cel mai mare număr întreg pe n biți în S.M.

$MAXINT_{SM}$: $0\ \underbrace{1\,1\dots1\,1}_{n-1 \text{ biți}} = 2^{n-1} - 1$

$\Rightarrow$ intervalul: $[1 - 2^{n-1};\ 2^{n-1} - 1]$

Cel mai mare număr fracționar n biți în S.M.

$MAXFRA_{SM}$: $0.\underbrace{1\,1\dots1\,1\,1}_{n-1 \text{ biți}} = 1 - 2^{-n+1}$

$\Rightarrow$ intervalul: $[2^{-n+1} - 1;\ 1 - 2^{-n+1}]$

Precizie

- luăm în considerare numerele Semn-Mărime pe n biți
- câte cifre zecimale sunt necesare pentru a reprezenta oricare dintre numerele Semn-Mărime pe n biți?

răspuns: $2^{n-1} - 1 = 10^p$, unde p - precizia

$$\Rightarrow p = \lceil \log_{10}(2^{n-1} - 1) \rceil \leq \lceil \lg(2^{n-1}) \rceil = \lceil (n-1) \cdot 0{,}3 \rceil$$

ex: avem un nr S.M. pe 10 biți

$p \approx \lceil 9 \cdot 0{,}3 \rceil = \lceil 2{,}7 \rceil = 3$

→ cel mai mare nr. SM pe 10 biți este +511 pe 3 cifre zecimale

Dezavantaje

- există **două** configurații binare pentru 0 în Semn-Mărime
  - +0 : 0 00···000
  - −0 : 1 00···000
- Adunarea în Semn-Mărime
  - se consideră operanzii $X = 5$ și $Y = 2$, pe 4 biți
  - cele patru posibile configurații de semn pentru adunare

X=+5: $0\,1\,0\,1_{SM}$ + Y=+2: $0\,0\,1\,0_{SM}$ = $0\,1\,1\,1_{SM}$ $=+7_{SM}$

X=+5: $0\,1\,0\,1_{SM}$ + Y=-2: $1\,0\,1\,0_{SM}$ = $1\,1\,1\,1_{SM}$ $=-7_{SM}$

X=-5: $1\,1\,0\,1_{SM}$ + Y=+2: $0\,0\,1\,0_{SM}$ = $1\,1\,1\,1_{SM}$ $=+7_{SM}$

X=-5: $1\,1\,0\,1_{SM}$ + Y=-2: $1\,0\,1\,0_{SM}$ = $0\,1\,1\,1_{SM}$ $=7_{SM}$

1.2.2. Complementul de 1

→ MSB codifică semnul (la fel ca la SM)

Un nr X, pe n biți, va fi reprezentat astfel în C1:

$$\overline{X} = \begin{cases} 0\ x_{n-2}x_{n-3}\dots x_1x_0, & X \geq 0 \\ 1\ \overline{x_{n-2}}\,\overline{x_{n-3}}\dots\overline{x_1}\,\overline{x_0}, & X \leq 0 \end{cases}$$

, unde $\overline{x_i} = 1 - x_i$

→ Reprezentare non-pozițională (non-ponderată)

ex: a) $+103_{(10)} = 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(C1)}$

$-103_{(10)} = 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(C1)}$

b) $+68_{(10)} = 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0_{(C1)}$

$-68_{(10)} = 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1_{(C1)}$

Interval de valori

→ la fel ca la S.M.

Cel mai mare număr întreg pe n biți în S.M.

$MAXINT_{SM}$: $0\ \underbrace{1\,1\dots1\,1}_{n-1\text{ biți}} = 2^{n-1} - 1$

=> intervalul : $[1 - 2^{n-1};\ 2^{n-1} - 1]$

Cel mai mare număr fracționar n biți în S.M.

$MAXFRA_{SM}$: $0.\underbrace{1\,1\dots1\,1\,1}_{n-1\text{ biți}} = 1 - 2^{-n+1}$

=> intervalul : $[2^{-n+1} - 1;\ 1 - 2^{-n+1}]$

Precizie

——→ la fel ca pentru SM

$2^{n-1} - 1 = 10^p$, unde p - precizia

$$\Rightarrow p = \lceil \log_{10}(2^{n-1}-1) \rceil \leq \lceil \lg(2^{n-1}) \rceil = \lceil (n-1) \cdot 0{,}3 \rceil$$

Complexitate hardware

——→ mai mare pentru C1 decât pt S.M.

⇒ nu mai este favorabil pentru înmulțire

Dezavantaje

- există **două** configurații binare pentru 0 în Complementul de 1
  - $+0$: $0\,00\cdots000$
  - -0 : $1\,11\cdots111$
- Adunarea în Complementul de 1
  - se consideră aceiași operanzi $X = 5$ și $Y = 2$, pe 4 biți
  - cele patru posibile configurații de semn pentru adunare

X=+5: $0\,1\,0\,1_{C1}$ |+
Y=+2: $0\,0\,1\,0_{C1}$
$0\,1\,1\,1_{C1}$ $=+7_{C1}$

X=+5: $0\,1\,0\,1_{C1}$ |+
Y=-2: $1\,1\,0\,1_{C1}$
end around carry → $1\,0\,0\,1\,0_{C1}$ |+ 1
$0\,0\,1\,1_{C1}$ $=+3_{C1}$ ~~$=+2_{C1}$~~

X=-5: $1\,0\,1\,0_{C1}$ |+
Y=+2: $0\,0\,1\,0_{C1}$
$1\,1\,0\,0_{C1}$
→ $1\,0\,1\,1_{SM}$ $=-3_{SM}$

X=-5: $1\,0\,1\,0_{C1}$ |+
Y=-2: $1\,1\,0\,1_{C1}$
$1\,0\,1\,1\,1_{C1}$ |+ 1 $=*7_{C1}$
$1\,0\,0\,0_{C1}$
→ $1\,1\,1\,1_{SM}$ $=-7_{SM}$

1.2.3. Complementul de 2

——→ codifică semnul (la fel ca la SM)

Un nr. pe n biți este reprezentat astfel în C2:

Întreg:

$$-x = \begin{cases} 0\ x_{n-2}x_{n-3}\dots x_1x_0 & , x \geq 0 \\ (1\ \overline{x_{n-2}}\,\overline{x_{n-3}}\dots\overline{x_1}\,\overline{x_0} + 1) \bmod 2^n & , x \leq 0 \end{cases}$$

Fracționar:

$$-x = \begin{cases} 0.x_{n-2}x_{n-3}\dots x_1x_0 & , x \geq 0 \\ (1.\overline{x_{n-2}}\,\overline{x_{n-3}}\dots\overline{x_1}\,\overline{x_0} + 0.0\dots01) & , x \leq 0 \end{cases}$$

Operatiile mod 2^n si mod 2 pentru numerele intregi, respectiv fractionare, asigura ca transportul generat dinspre MSB este ignorat

ex: $+103_{(10)} = 0\ 1100111_{(C2)}$

$-103_{(10)} = 1\ 1100111$ (SM)

$= 1\ 0011000$ (C1) $\mid +1$

1

$1\ 0011001$ (C2)

Regulă practică de conversie SM $\longleftrightarrow$ C2

$\longrightarrow$ se păstrează bitul de semn

$\longrightarrow$ începând de la stânga spre dreapta, se complementează fiecare bit, cu excepția celui mai din dreapta bit de 1 și a tuturor zerourilor care îl urmează

ex: $-103_{(10)} = 1\ 1100111$ (SM)

$= 1\ 0011000$ (C2)

$-68_{(10)} = 1\ 1000100$ (SM)

$1\ 0111100$ (C2)

$-4_{(10)} = 1\ 0100$ (SM)

$= 1\ 1100$ (C2)

$-10_{(10)} = 1\ 0110$ (C2)

$10_{(10)} = 0\ 1010$ (C2)

$1\ 000 \rightarrow -2^{4-1} = -8$ (C2)

$\longrightarrow$ reprezentare non-pozițională

Interval valoric

a) Pentru nr. întregi pe n biți: $[-2^{n-1}; 2^{n-1}-1]$

b) Pentru nr. fracționare pe n biți: $[-1; 1-2^{-n+1}]$

Precizia

$$p = \lceil (n-1)\log_{10} 2 \rceil$$

Complexitate hardware

1. Adunarea si scaderea sunt mai simple decat in SM sau C1 (SM nu poate efectua corect adunari independent de semnele operanzilor)

2. inmultirea este mai complexa decat in cazul SM

Configuratia pentru -0 vs +0

$-0 = 1\ 00\ldots00$ (SM)

$= 1\ 11\ldots11$ (C1)

$+\ 1$

$1\ 00\ldots00$

Este ignorat transportul din MSB(cel mai din stanga bit de 1) deoarece adunarea unitatii se efectueaza modulo 2^n, conform definitiei Complementului de 2

Pentru numere fractionare, se poate construi reprezentarea -0 intr-un mod similar

Adunarea binară în C2

- se consideră aceiași operanzi $X = 5$ și $Y = 2$, pe 4 biți
- cele patru configurații posibile ale semnelor pentru adunare

X=+5: $0\,1\,0\,1_{C2}$ +
Y=+2: $0\,0\,1\,0_{C2}$
$0\,1\,1\,1_{C2}$ $=+7_{C2}$

X=+5: $0\,1\,0\,1_{C2}$ +
Y=-2: $1\,1\,1\,0_{C2}$
$0\,0\,1\,1_{C2}$ $=+3_{C2}$

X=-5: $1\,0\,1\,1_{C2}$ +
Y=+2: $0\,0\,1\,0_{C2}$
$1\,1\,0\,1_{C2}$
$1\,0\,1\,1_{SM}$ $=-3_{SM}$

X=-5: $1\,0\,1\,1_{C2}$ +
Y=-2: $1\,1\,1\,0_{C2}$
$1\,0\,0\,1_{C2}$
$1\,1\,1\,1_{SM}$ $=-7_{SM}$

ex:

6	1 1 0	+
5	1 0 1	
1	0 1 1	

6	0 1 10	+ C_2
5	0 1 0 1	
	1 0 1 1	

-6	1 0 1 0	+ C_2
-5	1 0 1 1	
	0 1 0 1	$= +5$ (C2)

-> 1 011 (SM) -11

Avantajele aritmeticii în Complementul de 2:

- operație corectă indiferent de semnele operandelor
  - facilitează implementarea scăderii: $X - Y = X + (-Y)$
- carry-out din MSB este ignorat
- bitul de semn este tratat ca oricare alt bit de magnitudine

Comparație a codurilor pentru numere întregi pe 5 biți:

Număr zecimal	Coduri binare de virgulă fixă		
	SM	C1	C2
+15	01111	01111	01111
+14	01110	01110	01110
⋮	⋮	⋮	⋮
+2	00010	00010	00010
+1	00001	00001	00001
+0	00000	00000	00000
-0	10000	11111	00000
-1	10001	11110	11111
-2	10010	11101	11110
⋮	⋮	⋮	⋮
-14	11110	10001	10010
-15	11111	10000	10001

Comparație a codurilor pentru numere întregi pe 5 biți:

Număr zecimal	Coduri binare de virgulă fixă		
	SM	C1	C2
+15	01111	01111	01111
+14	01110	01110	01110
⋮	⋮	⋮	⋮
+2	00010	00010	00010
+1	00001	00001	00001
+0	00000	00000	00000
-0	10000	11111	00000
-1	10001	11110	11111
-2	10010	11101	11110
⋮	⋮	⋮	⋮
-14	11110	10001	10010
-15	11111	10000	10001
-16	—	—	10000

Anomalia Complementului de doi:

- Prin convenție, configurația $1\,00\cdots000_{C2}$ codifică:

$$1\,00\cdots000_{C2} \begin{cases} -2^{n-1} & \text{pentru numere întregi} \\ -1 & \text{pentru numere fractionare} \end{cases}$$

- Pentru numerele fără semn, aceeași configurație codifică:

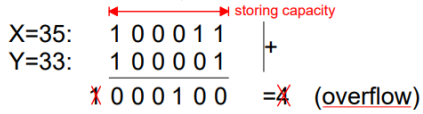
$$1\,00\cdots000 = +2^{n-1}$$

Overflow aritmetic

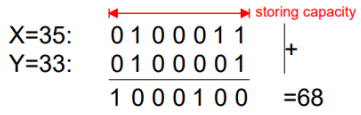
Rezultatul unei operatii aritmetice depaseste capacitatea de stocare.

Overflow aritmetic pentru numere fără semn:

- Se consideră $X = 35$, $Y = 33$ numere fără semn, pe 6 biți



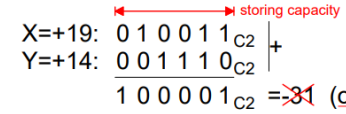
  - Dacă X și Y ar fi fost fără semn pe 7 biți:



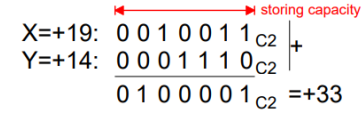
Notă: Overflow-ul la operațiile cu operanzi fără semn apare atunci când se generează un transport din MSB.

Overflow aritmetic pentru operanzi cu semn (C2):

- Se consideră $X = +19$, $Y = +14$ reprezentate în C2, pe 6 biți



  - Dacă X și Y ar fi fost fără semn pe 7 biți:



Notă: Overflow-ul pentru operanzii cu semn (C2) apare atunci când adunarea a doi operanzi de același semn produce un rezultat de semn opus.

1.2.4. Interpretare alternativă a Complementului de doi

Interpretarea lui Robertson

→ facilitează înmulțirea operanzilor în C2

X - nr. întreg negativ, în C2, pe n biți:

$$X = 1\ x^*_{n-2}\ x^*_{n-3}\ \cdots\ x^*_1\ x^*_0$$

$$= (1\ x^*_{n-2}\ x^*_{n-3}\ \cdots\ x^*_1\ x^*_0) \bmod 2^n$$

$$= \left(\begin{matrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ + & 0 & x^*_{n-2} & x^*_{n-3} & \cdots & x^*_1 & x^*_0 \end{matrix} \right) \bmod 2^n$$

$$= (-2^{n-1} + 0\ x^*_{n-2}\ x^*_{n-3}\ \cdots\ x^*_1\ x^*_0) \bmod 2^n$$

$0\ x^*_{n-2}\ x^*_{n-3}\ \cdots\ x^*_1\ x^*_0$ ↔ nr > 0 în C2

$$= \underbrace{(-1)}_{\text{semn}} \cdot 2^{n-1} + 0\ x^*_{n-2}\ x^*_{n-3}\ \cdots\ x^*_1\ x^*_0$$

Interpretarea lui Robertson : valoarea unui numar negativ in C2 este egala cu valoarea numarului pozitiv obtinut prin stergerea bitului de semn (din 0 -> 1) din care se scade ponderea asociata bitului de semn

Exemplu:

-103_{10} = 1 0 0 1 1 0 0 1 (C2)

⇓

= $-1*2^7+$ 0 0 0 1 1 0 0 1 (C2)

= −128+ 25

= −103

-68_{10} = 1 0 1 1 1 1 0 0 (C2)

⇓

= $-1*2^7+$ 0 0 1 1 1 1 0 0 (C2)

= −128+ 60

= −68

1 101 (C2) = $-1 \cdot 2^3$ + 0 101 = −8 + 5 = −3

1 011 = −3 (SM)

Interpretarea lui Robertson se aplică și numerelOR pozitive:

$+103_{10}$ = 0 1 1 0 0 1 1 1 (C2)

= $-0*2^7+$ 0 1 1 0 0 1 1 1 (C2)

= 0+ 103

= +103

În general:

- SE consideră X, întreg reprezentat în C2, pe n biți, cu $X = x_{n-1}\, x_{n-2}x_{n-3}\cdots x_1x_0$

Poate fi exprimat ca:

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

Consideratii similare pot fi construite si pentru numerele fractionare in C2.

ex: 0,2 în baza 2

0,2	· 2
0,4	· 2
0,8	· 2
1,6	· 2
1,2	...

$0,\overline{0011}\ldots$