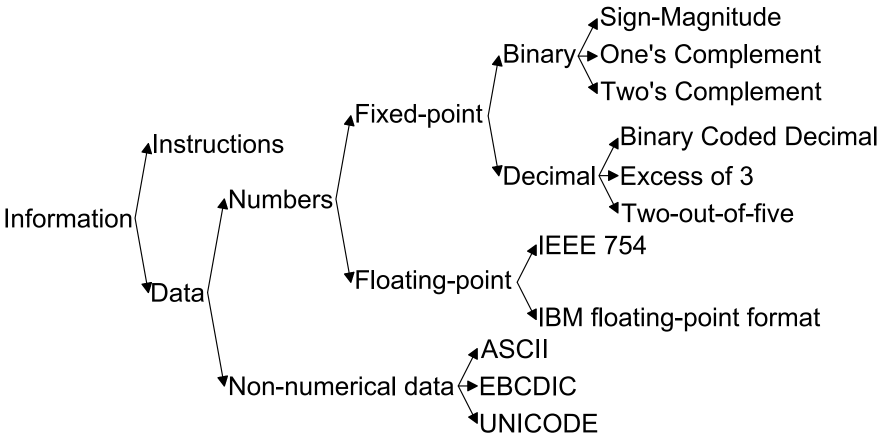
Arhitectura Calculatoarelor

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

4 Octombrie 2023
11 Octombrie 2023
18 Octombrie 2023

Cap. 1 Reprezentarea numerelor în sistemele de calcul - Recapitulare

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!

Numărul X reprezentat în baza $\boldsymbol{r}$:

- $X = x_{n-1}x_{n-2}\cdots x_1x_0.x_{-1}x_{-2}\cdots x_{-m}$
  - 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}

Valoarea lui X reprezentată în baza $\boldsymbol{r}$:

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

Atunci când baza $r = 2 \rightarrow$ sistem de reprezentare binar.

Exemplu:

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

weights $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$$68_{(10)} = 1\ 0\ 0\ 0\ 1\ 0\ 0\ ._{(2)}$$

Poziția virgulei binare:

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

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

- $X = x_{n-1}x_{n-2}\cdots x_1x_0. = \sum_{i=0}^{n-1} x_i * 2^i$, pentru X pe n biți

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

- $X = .x_{n-1}x_{n-2}\cdots x_1x_0 = \sum_{i=0}^{n-1} x_i * 2^{i-n}$, pentru X pe n biți

Exemplu:

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

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

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

MSB codifică semnul numărului. Convenția de semn:

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

Numărul X, pe n biți, este reprezentat în forma Semn-Mărime astfel:

- $X = x_{n-1} \ \ x_{n-2}x_{n-3} \cdots x_1x_0$, unde
  - x_{n-1} fiind semnul lui X
  - $x_{n-2}x_{n-3} \cdots x_1x_0$ fiind magnitudinea lui X

Exemplu:

$$+103_{10} \ = \ 0\ 1100111_{(SM)}$$
$$-103_{10} \ = \ 1\ 1100111_{(SM)}$$

Interval valoric:

- cel mai mare număr întreg pe n biți, în Semn-Mărime:
  - $MAXINT_{SM}$: $0\,\underbrace{11\cdots111}_{n-1\text{ biți}}$.
  - valoarea lui $MAXINT_{SM} = 2^{n-1} - 1$
- $\Rightarrow$ interval de valori pentru numerele întregi pe n biți: $[1 - 2^{n-1}; 2^{n-1} - 1]$

- cel mai mare număr fracționar pe n biți, în Semn-Mărime:
  - $MAXFRA_{SM}$: $0\,.\underbrace{11\cdots111}_{n-1\text{ biți}}$
  - valoarea lui $MAXFRA_{SM} = 1 - 2^{-n+1}$
- $\Rightarrow$ interval de valori pentru numerele fracționare pe n biți: $[2^{-n+1} - 1; 1 - 2^{-n+1}]$

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?
  - $2^{n-1} - 1 = 10^p$, unde p este *precizia*
  - rezultă că $p = \lceil log_{10}(2^{n-1} - 1) \rceil$
  - astfel, $p \leq \lceil log_{10}(2^{n-1}) \rceil =$
  - în final, $p \leq \lceil (n-1) * 0.3 \rceil$

Exemplu: se consideră numere Semn-Mărime pe 10 biți.

- precizia $p \approx \lceil 9 * 0.3 \rceil = \lceil 2.7 \rceil = 3$
- cel mai mare număr întreg în Semn-Mărime pe 10 biți este $+511$, a cărui reprezentare zecimală are 3 cifre

Complexitatea hardware a reprezentării Semn-Mărime:

- complexitate hardware moderată
- favorabilă operației de înmulțire

Dezavantaje:

- există **două** configurații binare pentru 0 în Semn-Mărime
  - $+0 : \;\; 0\,00\cdots000$
  - $-0 : \;\; 1\,00\cdots000$

Dezavantaje:

- Adunarea în Semn-Mărime
  - se consideră operanzii $X = 5$ și $Y = 2$, pe 4 biți
  - cele patru posibile configurații de semn pentru adunare

$$
\begin{array}{lrl}
X{=}{+}5: & 0\,1\,0\,1_{SM} & \\
Y{=}{+}2: & 0\,0\,1\,0_{SM} & + \\
\hline
 & 0\,1\,1\,1_{SM} & ={+}7_{SM}
\end{array}
\qquad
\begin{array}{lrl}
X{=}{+}5: & 0\,1\,0\,1_{SM} & \\
Y{=}{-}2: & 1\,0\,1\,0_{SM} & + \\
\hline
 & 1\,1\,1\,1_{SM} & =\cancel{-7_{SM}}
\end{array}
$$

$$
\begin{array}{lrl}
X{=}{-}5: & 1\,1\,0\,1_{SM} & \\
Y{=}{+}2: & 0\,0\,1\,0_{SM} & + \\
\hline
 & 1\,1\,1\,1_{SM} & =\cancel{+7_{SM}}
\end{array}
\qquad
\begin{array}{lrl}
X{=}{-}5: & 1\,1\,0\,1_{SM} & \\
Y{=}{-}2: & 1\,0\,1\,0_{SM} & + \\
\hline
 & \cancel{1}\;0\,1\,1\,1_{SM} & =\cancel{+}7_{SM}
\end{array}
$$

MSB codifică semnul (aceeași convenție ca pentru SM).

Numărul X, pe n biți, este reprezentat în Complementul de 1 astfel:

- $\overline{X} = \begin{cases} 0 \;\; x_{n-2}x_{n-3}\cdots x_1x_0 & X \geq 0 \\ 1 \;\; \overline{x_{n-2}}\,\overline{x_{n-3}}\cdots\overline{x_1}\,\overline{x_0} & X \leq 0 \end{cases}$, unde
  - $\overline{x_i}$ reprezentând complementul lui x_i: $\overline{x_i} = 1 - x_i$
- Reprezentare non-pozitionala (non-ponderata)

Exemplu:

$$+103_{10} = 0\ 1100111_{(C1)}$$
$$-103_{10} = 1\ 0011000_{(C1)}$$

$$+68_{10} = 0\ 1000100_{(C1)}$$
$$-68_{10} = 1\ 0111011_{(C1)}$$

1.2.2 - Complementul de 1 (cont.)

Interval de valori:

- la fel ca pentru Semn-Mărime
  - pentru un număr dat pe *n* biți, Semn-Mărime și Complementul de 1 codifică același număr de valori

Precizie:

- la fel ca pentru Semn-Mărime
  - deoarece codifică același număr de valori, Complementul de 1 are aceeași precizie ca Semn-Mărime

Complexitate hardware:

- complexitatea hardware mai mare pentru Complementul de 1
  - nu mai este favorabil operației de înmulțire

Dezavantaje:

- există **două** configurații binare pentru 0 în Complementul de 1
  - $+0: \;\; 0\,00\cdots000$
  - $-0: \;\; 1\,11\cdots111$

Dezavantaje:

- 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

$$
\begin{array}{rl|l}
X=+5: & 0\,1\,0\,1_{C1} & + \\
Y=+2: & 0\,0\,1\,0_{C1} & \\
\hline
 & 0\,1\,1\,1_{C1} & =+7_{C1}
\end{array}
$$

$$
\begin{array}{rrl}
X=+5: & & 0\,1\,0\,1_{C1} \quad + \\
Y=-2: & & 1\,1\,0\,1_{C1} \\
\hline
\text{end around carry} & 1 & 0\,0\,1\,0_{C1} \quad + \qquad =\cancel{+2_{C1}} \\
 & & 1 \\
\hline
 & & 0\,0\,1\,1_{C1} \qquad =+3_{C1}
\end{array}
$$

$$
\begin{array}{rl}
X=-5: & 1\,0\,1\,0_{C1} \quad + \\
Y=+2: & 0\,0\,1\,0_{C1} \\
\hline
 & 1\,1\,0\,0_{C1} \\
 & 1\,0\,1\,1_{SM} \qquad =-3_{SM}
\end{array}
$$

$$
\begin{array}{rrl}
X=-5: & & 1\,0\,1\,0_{C1} \quad + \\
Y=-2: & & 1\,1\,0\,1_{C1} \\
\hline
 & 1 & 0\,1\,1\,1_{C1} \quad + \qquad =\cancel{+}7_{C1} \\
 & & 1 \\
\hline
 & & 1\,0\,0\,0_{C1} \\
 & & 1\,1\,1\,1_{SM} \qquad =-7_{SM}
\end{array}
$$

Există dezavantaje legate de adunarea în Complementul de 1?

MSB codifică semnul (aceeași convenție ca pentru SM).

Numărul întreg X, pe n biți, este reprezentat în C2 astfel:

- $$-X = \begin{cases} 0\ x_{n-2}x_{n-3}\cdots x_1x_0. & X \geq 0 \\ (1\ \overline{x_{n-2}}\,\overline{x_{n-3}}\cdots\overline{x_1}\,\overline{x_0}.+1) \mod 2^n & X < 0 \end{cases}$$

Numărul fracționar X, pe n biți, este reprezentat în C2 astfel:

- $$-X = \begin{cases} 0\ .x_{n-2}x_{n-3}\cdots x_1x_0 & X \geq 0 \\ (1\ .\overline{x_{n-2}}\,\overline{x_{n-3}}\cdots\overline{x_1}\,\overline{x_0}+0.0\cdots01) \mod 2 & X < 0 \end{cases}$$

Operațiile $\mod 2^n$ și $\mod 2$ pentru numerele întregi, respectiv, fracționare, asigură că transportul generat dinspre MSB este ignorat

$$+103_{10} = \begin{array}{cccccccc} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{array} \quad (C2)$$

$$\begin{array}{rcccccccccl} -103_{10} = & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & (SM) & \\ = & 1 & \color{red}{0} & \color{red}{0} & \color{red}{1} & \color{red}{1} & \color{red}{0} & \color{red}{0} & \color{red}{0} & (C1) & \\ & & & & & & & & 1 & & + \\ \hline = & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & (C2) & \end{array}$$

Regulă practică de conversie SM $\leftrightarrow$ C2:

- se păstrează bitul de semn
- 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

Exemplu: se consideră numere pe 10 biți:

$$\begin{array}{rcccccccccl} -103_{10} = & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & (SM) \\ & & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & & \\ = & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & (C2) \end{array}$$

$$\begin{array}{rcccccccccl} -68_{10} = & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & (SM) \\ & & \Downarrow & \Downarrow & \Downarrow & \Downarrow & & & & \\ = & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & (C2) \end{array}$$

Reprezentare non-pozitionala (non-ponderata)

Intervalul valoric:

- pentru numere întregi pe n biți: $[-2^{n-1}; 2^{n-1} - 1]$
- pentru numere fracționare pe n biți: $[-1; 1 - 2^{-n+1}]$

Precizia:

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

Complexitate hardware:

- adunarea și scăderea sunt mai simple decât în cazul SM/C1 (SM nu poate efectua corect adunări independent de semnele operanzilor)
- înmulțirea este mai complexă decât în cazul SM

Configurația binară pentru -0 este diferită de cea pentru $+0$?

- se consideră operanzi întregi pe n biți

$$
\begin{array}{rcccccccccl}
-0 = & & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & (SM) & \\
= & & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & (C1) & \\
 & & & & & & & & 1 & & + \\
\hline
= & \cancel{1} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & (C2) &
\end{array}
$$

- observații:
  - Este ignorat transportul din MSB (cel mai din stânga bit de 1) deoarece adunarea unității se efectuează modulo 2^n, conform definiției Complementului de doi
  - Pentru numere fracționare, se poate construi reprezentarea -0 într-un mod similar

Adunarea binară în Complementul de 2:

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

$$
\begin{array}{lrl}
X=+5: & 0\,1\,0\,1_{C2} & + \\
Y=+2: & 0\,0\,1\,0_{C2} & \\
\hline
 & 0\,1\,1\,1_{C2} & =+7_{C2}
\end{array}
\qquad
\begin{array}{lrl}
X=+5: & 0\,1\,0\,1_{C2} & + \\
Y=-2: & 1\,1\,1\,0_{C2} & \\
\hline
 & \cancel{1}\,0\,0\,1\,1_{C2} & =+3_{C2}
\end{array}
$$

$$
\begin{array}{lrl}
X=-5: & 1\,0\,1\,1_{C2} & + \\
Y=+2: & 0\,0\,1\,0_{C2} & \\
\hline
 & 1\,1\,0\,1_{C2} & \\
 & 1\,0\,1\,1_{SM} & =-3_{SM}
\end{array}
\qquad
\begin{array}{lrl}
X=-5: & 1\,0\,1\,1_{C2} & + \\
Y=-2: & 1\,1\,1\,0_{C2} & \\
\hline
 & \cancel{1}\,1\,0\,0\,1_{C2} & \\
 & 1\,1\,1\,1_{SM} & =-7_{SM}
\end{array}
$$

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 operații aritmetice depășește capacitatea de stocare.

Overflow aritmetic pentru numere fără semn:

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

storing capacity

```
X=35:    1 0 0 0 1 1  |
Y=33:    1 0 0 0 0 1  |+
        -------------
       1 0 0 0 1 0 0     =4   (overflow)
```

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

storing capacity

```
X=35:    0 1 0 0 0 1 1  |
Y=33:    0 1 0 0 0 0 1  |+
        ---------------
         1 0 0 0 1 0 0     =68
```

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

storing capacity

$$
\begin{array}{rl}
X{=}{+}19{:} & 0\,1\,0\,0\,1\,1_{C2} \\
Y{=}{+}14{:} & 0\,0\,1\,1\,1\,0_{C2} \quad + \\
\hline
 & 1\,0\,0\,0\,0\,1_{C2} \;= \cancel{-31}\; (\underline{\text{overflow}})
\end{array}
$$

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

storing capacity

$$
\begin{array}{rl}
X{=}{+}19{:} & 0\,0\,1\,0\,0\,1\,1_{C2} \\
Y{=}{+}14{:} & 0\,0\,0\,1\,1\,1\,0_{C2} \quad + \\
\hline
 & 0\,1\,0\,0\,0\,0\,1_{C2} \;= +33
\end{array}
$$

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

Interpretarea lui Robertson:

- Facilitează înmulțirea operanzilor în C2

Fie X un număr întreg negativ, reprezentat în C2, pe n biți:

$$
\begin{array}{rclcccccccl}
X & = & & 1 & x^\star_{n-2} & x^\star_{n-3} & \cdots & x^\star_1 & x^\star_0 & & \\
 & = & (& 1 & x^\star_{n-2} & x^\star_{n-3} & \cdots & x^\star_1 & x^\star_0 &) & \mod 2^n \\
 & = & \Big(& 1 & 0 & 0 & \cdots & 0 & 0 & & \\
 & & + & 0 & x^\star_{n-2} & x^\star_{n-3} & \cdots & x^\star_1 & x^\star_0 & \Big) & \mod 2^n \\
 & = & (\; -2^{n-1} + & 0 & x^\star_{n-2} & x^\star_{n-3} & \cdots & x^\star_1 & x^\star_0 &) & \mod 2^n \\
 & & & \multicolumn{6}{c}{\longleftarrow \text{Număr pozitiv în C2} \longrightarrow} & & \\
 & = & -2^{n-1} + & 0 & x^\star_{n-2} & x^\star_{n-3} & \cdots & x^\star_1 & x^\star_0 & &
\end{array}
$$

1.2.4 - Interpretare alternativă a Complementului de doi (contd.)

Interpretarea lui Robertson: valoarea unui număr negativ în C2 este egală cu valoarea numărului pozitiv obținut prin *ștergerea* bitului de semn din care se scade ponderea asociată bitului de semn.

Exemplu:

$$
\begin{array}{rcccccccccc}
-103_{10} = & & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & (C2) \\
 & & \Downarrow & & & & & & & & \\
= & -1 * 2^7 + & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & (C2) \\
= & -128 + & 25 & & & & & & & & \\
= & -103 & & & & & & & & &
\end{array}
$$

$$
\begin{array}{rcccccccccc}
-68_{10} = & & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & (C2) \\
 & & \Downarrow & & & & & & & & \\
= & -1 * 2^7 + & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & (C2) \\
= & -128 + & 60 & & & & & & & & \\
= & -68 & & & & & & & & &
\end{array}
$$

1.2.4 - Interpretare alternativă a Complementului de doi (contd.)

Interpretarea lui Robertson se aplică și numerelOR pozitive:

$$
\begin{array}{rcccccccccc}
+103_{10} = & & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & (C2) \\
= & -0 * 2^7 + & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & (C2) \\
= & 0+ & 103 & & & & & & & & \\
= & +103 & & & & & & & & &
\end{array}
$$

În general:

- SE consideră X, întreg reprezentat în C2, pe n biți, cu $X = x_{n-1}\ x_{n-2} x_{n-3} \cdots x_1 x_0$
- valoarea lui X poate fi exprimată ca:

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

Considerații similare pot fi construite și pentru numerele fracționare în C2.

Comparație coduri de reprezentare zecimală:

Cifra zecimală	Coduri zecimale de virgulă fixă		
	BCD8421	Exces de 3	Doi-din-cinci
0	0000	0011	11000
1	0001	0100	00011
2	0010	0101	00101
3	0011	0110	00110
4	0100	0111	01001
5	0101	1000	01010
6	0110	1001	01100
7	0111	1010	10001
8	1000	1011	10010
9	1001	1100	10100

Notă: codificarea *doi-din-cinci* nu este unică.

Reprezintă o cifră zecimală pe o tetradă (grup de patru biți)

- tetradă $\equiv$ nibble $\equiv$ 4 biți consecutivi
- de obicei, referit prin mai simplul Binary Coded Decimal (BCD)

Exemplu:

$$297_{10} \quad = \quad 0010 \quad 1001 \quad 0111 \quad _{BCD}$$

- ponderea unui bit în reprezentare este $10^i * 2^j$, unde
  - i - poziția cifrei zecimale
  - j - poziția bitului în tetradă
- ponderi fixe $\rightarrow$ reprezentare pozițională

Reprezintă o cifră zecimală pe o tetradă

- Excesul $\equiv$ bias
  - Valoare adăugată la toate configurațiile codificării

Excesul de 3 (E3) adaugă 3 unități la cifra zecimală BCD corespunzătoare.

- Nu este pozițional
- Facilitează adunarea

Exemplu:

$$297_{10} \quad = \quad 0101 \quad 1100 \quad 1010 \quad {}_{E3}$$

Reprezintă o cifră zecimală folosind 5 biți

- 2 biți din cei 5 utilizați pentru reprezentarea unei cifre sunt 1
- 3 biți din cei 5 utilizați pentru reprezentarea unei cifre sunt 0

Doi-din-cinci (2-o-o-5) facilitează detectarea erorilor:

- folosind coduri de paritate: modificarea oricăruia dintre cei 5 biți schimbă numărul de biți de 1 și 0 în reprezentarea fiecărei cifre

Exemplu:

$$297_{10} \quad = \quad 00101 \quad 10100 \quad 10001 \quad {}_{2-o-o-5}$$

Notație științifică: $X = X_M * B^{X_E}$

- X_M - mantisa, reprezentată ca un număr de virgulă fixă, fracționar
- X_E - exponentul, reprezentat ca un număr de virgulă fixă, întreg
- B - baza reprezentării; în mod obișnuit, B este 2 sau o putere a lui 2

Atât X_M, cât și X_E pot fi reprezentate în SM sau în C2.

1.4.1 - Considerații generale

Formatul de virgulă mobilă:

sign	exponent	fractional part of mantissa
S	X_E	X_M^*
<1>	<*e*>	<*m*>

Mantisa folosește 1 bit pentru semn și m biți pentru partea fracționară:

$$X_M = S.X_M^\star$$

unde

- X_M - mantisă
- S - semnul numărului de virgulă mobilă
- $X_M^\star$ - partea fracționară a mantisei

Constrângeri de reprezentarea numerelor de virgulă mobilă

A Reprezentarea valorii 0

- X_E egal cu 0: ar trebui să fie cea mai mică valoare posibilă
  - comparație rapidă cu 0
  - erori de rotunjire

$$X_E \begin{cases} SM: & \text{cea mai mică valoare: } 11\cdots1 & = -2^{e-1}+1 \\ C2: & \text{cea mai mică valoare: } 10\cdots0 & = -2^{e-1} \end{cases}$$

- X_M egal cu 0: ar trebui să fie configurația all-0, indiferent de codul de reprezentare folosit, SM sau C2

Asamblând toate câmpurile împreună:

$$0 \begin{cases} SM: & S|11\cdots1|00\cdots0 \\ C2: & S|10\cdots0|00\cdots0 \end{cases}$$

Constrângeri de reprezentarea numerelor de virgulă mobilă

A Reprezentarea valorii 0

- Compararea rapidă cu 0 a numerelor de virgulă mobilă
  - necesită ca X_E să fie all-0
  - $\Rightarrow$ utilizarea unui cod bias (sau exces) pentru X_E
    - fiecărei configurații binară a noului cod exces i se asociază o valoare egală cu valoarea binară a codului la care se adaugă valoarea excesului

Excesul: cea mai mare valoare reprezentabilă în câmpul X_E:

$$\text{exces} \begin{cases} SM: & 2^{e-1} - 1 \\ C2: & 2^{e-1} \end{cases}$$

Noua reprezentare a lui 0:

$$0 \longrightarrow S|00\cdots0|00\cdots0$$

Observație: reprezentarea lui 0 este cu semn, semnul S fiind codificat în MSB-ul formatului.

Constrângeri de reprezentarea numerelor de virgulă mobilă

B Reprezentarea mantisei

- Inerent redundantă
  - În zecimal, următoarele expresii se referă la aceeași valoare: $0.237 = 0.0237 * 10^1 = 0.000237 * 10^3 = 23.7 * 10^{-2} = \cdots$
- Utilizarea *mantisei normalizate*: reprezentare unică

MSB-ul părții fracționare a mantisei :
- 1 dacă mantisa este în SM
- $\overline{S}$ dacă mantisa este în C2

Exemplu:

$$0.\underbrace{1}_{\text{msb}=1}\,1\,1_{SM} = \frac{7}{2^3} \qquad 1.\underbrace{1}_{\text{msb}=1}\,1\,1_{SM} = \frac{-7}{2^3}$$

$$\underbrace{0}_{S}.\underbrace{1}_{\text{msb}=\overline{S}}\,1\,1_{C2} = \frac{7}{2^3} \qquad \underbrace{1}_{S}.\underbrace{0}_{\text{msb}=\overline{S}}\,0\,1_{C2} = \frac{-7}{2^3}$$

Intervalul de valori al mantisei:

$$\frac{1}{2} \leq \|X_M\| < 1$$

1.4.2 - IEEE 754

Format portabil :
- → 32 biți (precizie simplă)
- → 64 biți (precizie dublă)
- → 128 biți (precizie cvadruplă)

Format de schimb de date:
- pe 16 biți (precizie înjumătățită)
  - folosit pentru accelerarea Deep Neural Networks

Formatul de precizie simplă:

sign	exponent	fractional part of significand
S	X_E	X_S^*
<1>	<8>	<23>

Câmpul exponentului, X_E, reprezintă exponentul real într-un cod exces de $127(= 2^{8-1} - 1)$.

Mantisa, pentru standardul IEEE 754, constă din bitul de semn și significand. Significandul este fără semn și are reprezentarea:

$$X_S = 1.X_S^\star$$

unde

- X_S - significand
- 1 - poziția supraunitară; **ascunsă** în reprezentare
- $X_S^\star$ - partea fracționară a significandului

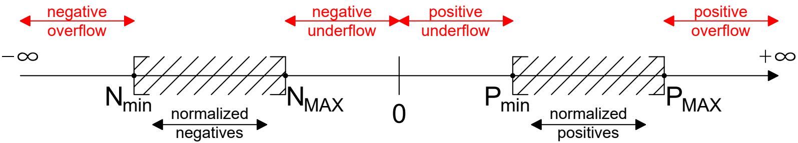
Intervalul de valori pentru significand este:

$$1 \leq X_S < 2$$

Valoarea unei reprezentări IEEE 754 de precizie simplă este dată de relația:

$$X = (-1)^S * 2^{X_E - exces} * (1.X_S^\star)$$

Limitele standardului IEEE 754 de precizie simplă:



Determinarea valorii P_{MAX}:

$$X_E = X_{E_{max}} - 1 = 255 - 1 = 254$$
$$X_S^\star = .\ \underbrace{1\ 1\ \cdots\ 1\ 1\ 1}_{\text{23 de biți}} = 1 - 2^{-23}$$
$$P_{MAX} = (-1)^0 * 2^{254-127} * (1.11\cdots1) = 2^{127} * (2 - 2^{-23}) \approx 3.4 * 10^{38}$$

Determinarea valorii P_{min}:

$$X_E = X_{E_{min}} + 1 = 0 + 1 = 1$$
$$X_S^\star = .\ \underbrace{0\ 0\ \cdots\ 0\ 0\ 0}_{\text{23 de biți}} = 0$$
$$P_{min} = (-1)^0 * 2^{1-127} * (1.00\cdots0) = 2^{-126} \approx 1.18 * 10^{-38}$$

Valori speciale în IEEE 754

(A) Not a Number (NaN)

- rezultatul unor operații precum $\infty - \infty$, $0 * \infty$, $\frac{\infty}{\infty}$, $\frac{0}{0}$, X mod 0, $\sqrt{\text{valoare negativă}}$
- câmpul exponentului, $X_E = X_{E_{max}} = 255$
- câmpul de parte fracționară a significandului, $X_S^\star \neq 0$
  - valoarea părții fracționare a significandului indică operația care a produs NaN

(B) Infinit

- indică un rezultat care depășește capacitatea de reprezentare
  - preferabil operației de trunchiere la P_{MAX}
  - **Q**: de ce?
- rezultatul unor operații precum $X \div 0$, $X \pm \infty$, $\sqrt{\infty}$
- câmpul exponentului, $X_E = X_{E_{max}} = 255$
- câmpul de parte fracționară a significandului, $X_S^\star = 0$

Valori speciale în IEEE 754

(C) Zero

- câmpul exponentului, $X_E = X_{E_{min}} = 0$
- câmpul părții fracționare a significandului, $X_S^\star = 0$

(D) Numere denormalizate (Denormals)

- indică un rezultat, în valoare absolută, mai mic decât P_{min}, dar cu $X_S^\star$ nenul
  - preferabil trunchierii la P_{min}
- câmpul exponentului, $X_E = X_{E_{min}} = 0$
- câmpul părții fracționare a significandului, $X_S^\star \neq 0$

Valoarea unui denormal este dată de:

$$X_D = (-1)^S * 2^{1-exces} * (0.X_S^\star)$$

unde

- X_D - număr denormalizat
- 0 - poziția supraunitară; **ascunsă** în reprezentare
- $X_S^\star$ - parte fracționară a significandului

1.4.2 - IEEE 754 (contin.)

Exemplu: Reprezentați $X = 612.8046875_2$ în formatul IEEE 754 de precizie simplă

$612.8046875_{10} = 1001100100.1100111_2$

$$
\begin{array}{rl}
612 & - \\
\underline{512} & (2^9) \\
100 & - \\
\underline{64} & (2^6) \\
36 & - \\
\underline{32} & (2^5) \\
4 & (2^2)
\end{array}
\qquad\qquad
\begin{array}{l|l}
.8046875_{10} & \times 2 \\
1.609375_{10} & \times 2 \\
1.21875_{10} & \times 2 \\
0.4375_{10} & \times 2 \\
0.875_{10} & \times 2 \\
1.75_{10} & \times 2 \\
1.5_{10} & \times 2 \\
1.0_{10} &
\end{array}
$$

$612.8046875_{10} = 1.001100100110011 1_2 \times 2^9$

$X_E = (9 + \text{bias})_{10} = (9 + 127)_{10} = 136_{10} = 10001000_2$

$X_S^* = .0011001001100111_2$

	S	X_E								X_S^*																						
X:	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0
			4				4				1				9				3				3				8				0_{16}	

$X = 44193380_{16}$

Exemplu: Dându-se configurația $412A0000_{16}$ care reprezintă un număr de virgulă mobilă în formatul IEEE 754 de precizie simplă, determinați corespondentul său zecimal.

	4	1	2	A	0	0	0	0_{16}
X:	0 1 0 0	0 0 0 1	0 0 1 0	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

S | X_E | X_S^*

$X_E = 10000010_2 = (128+2)_{10} = 130_{10}$

$X_S^* = .010101_2$

$X = (-1)^S \times 2^{X_E\text{-bias}} \times (1.X_S^*) = (-1)^0 \times 2^{130-127} \times (1.010101_2) = 2^3 * 1.010101_2$

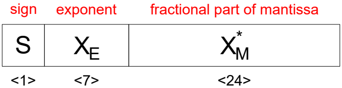
$X = 1010.101_2 = (10 + 2^{-1} + 2^{-3})_{10} = (10 + 0.5 + 0.125)_{10} = 10.625_{10}$

1.4.3 Formatul de virgulă mobilă IBM

Specific sistemelor IBM S/360, S/370

- Formate de 32, 64 și 128 de biți
- Baza 16 ($= 2^4$)
- Fără bit ascuns
- Normalizarea mantisei:
  - Toți biții mantisei sunt la dreapta virgulei binare
  - Pot exista cel mult 3 leading-zeroes
- Un singur caz special - zero:
  - $X_E = 0$
  - $X_M^\star = 0$

Formatul de 32 de biți:



- Valoarea lui X poate fi exprimată ca:
  $X = (-1)^S * 16^{X_E - bias} * (0.X_M^\star)$
- Exponent reprezentat în exces de 64 ($= 2^{7-1}$)

Exemplu: Reprezentați valoarea reală $X = 612.8046875_2$ în formatul de virgulă mobilă IBM pe 32 de biți

$612.8046875_{10} = 1001100100.1100111_2$

$612.8046875_{10} = 0.0010011001001100111_2 \times 16^3$

$X_E = (3+bias)_{10} = (3+64)_{10} = 67_{10} = 1000011_2$

$X_M^* = .0010011001001100111_2$

S	X_E	X_M^*
0	1 0 0 0 0 1 1	0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 0 0

X: 0 1 0 0 | 0 0 1 1 | 0 0 1 0 | 0 1 1 0 | 0 1 0 0 | 1 1 0 0 | 1 1 1 0 | 0 0 0 0

4 | 3 | 2 | 6 | 4 | C | E | 0_{16}

$X = 43264CE0_{16}$

Exemplu: Determinați valoarea zecimală reală care corespunde configurației hexazecimale $412A0000_{16}$, având în vedere că valoarea hexazecimală reprezintă un număr de virgulă mobilă în formatul IBM pe 32 de biți.

	4	1	2	A	0	0	0	0_{16}
X:	0 | 1 0 0	0 0 0 1	0 0 1 0	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
	S | X_E		| X_M^*					

$$X_E = 1000001_2 = (64+1)_{10} = 65_{10}$$

$$X_M^* = .0010101_2$$

$$X = (-1)^S \times 16^{X_E\text{-bias}} \times (0.X_M^*) = (-1)^0 \times 16^{65-64} \times (0.0010101_2) = 2^4 \times 0.0010101$$

$$X = 10.101_2 = (2+2^{-1}+2^{-3})_{10} = (2+0.5+0.125)_{10} = 2.625_{10}$$