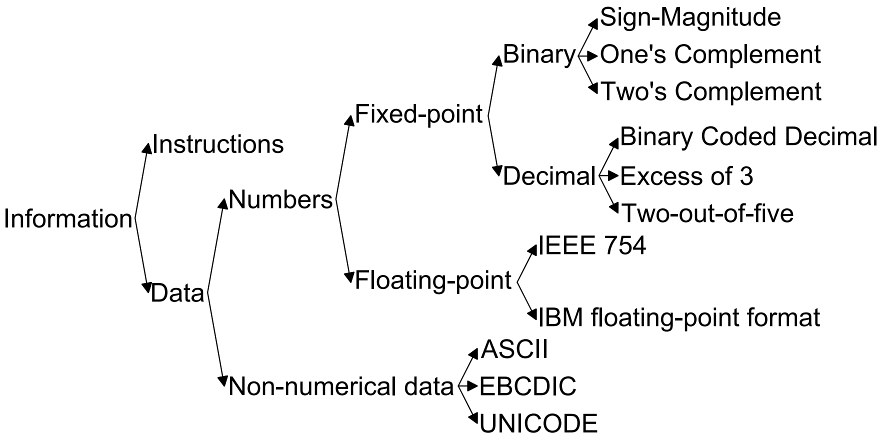
Arhitectura Calculatoarelor

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

9 Octombrie 2024

Cap. 1 Reprezentarea numerelor în sistemele de calcul - Recapitulare

Clasificarea informației:



Bit: <u>b</u>inar digi<u>t</u>

- 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 $\boldsymbol{r}$:

- $X = \sum_{j=-m}^{n-1} x_j * r^j$, $0 \leq x_j < r$
  - 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 $\quad 2^6\;\; 2^5\;\; 2^4\;\; 2^3\;\; 2^2\;\; 2^1\;\; 2^0$

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

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
  - $+0: \quad 0\,00\cdots000$
  - $-0: \quad 1\,11\cdots111$

1.2.2 - Complementul de 1 (cont.)

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

$$
\begin{array}{rll}
X=+5: & 0\,1\,0\,1_{C1} & + \\
Y=+2: & 0\,0\,1\,0_{C1} & \\
\hline
 & 0\,1\,1\,1_{C1} & =+7_{C1}
\end{array}
\qquad
\begin{array}{rll}
X=+5: & 0\,1\,0\,1_{C1} & + \\
Y=-2: & 1\,1\,0\,1_{C1} & \\
\hline
\text{end around carry} \; 1 & 0\,0\,1\,0_{C1} & =\cancel{+2_{C1}} \\
 & 1 & + \\
\hline
 & 0\,0\,1\,1_{C1} & =+3_{C1}
\end{array}
$$

$$
\begin{array}{rll}
X=-5: & 1\,0\,1\,0_{C1} & + \\
Y=+2: & 0\,0\,1\,0_{C1} & \\
\hline
 & 1\,1\,0\,0_{C1} & \\
 & 1\,0\,1\,1_{SM} & =-3_{SM}
\end{array}
\qquad
\begin{array}{rll}
X=-5: & 1\,0\,1\,0_{C1} & + \\
Y=-2: & 1\,1\,0\,1_{C1} & \\
\hline
1 & 0\,1\,1\,1_{C1} & =\cancel{+}7_{C1} \\
 & 1 & + \\
\hline
 & 1\,0\,0\,0_{C1} & \\
 & 1\,1\,1\,1_{SM} & =-7_{SM}
\end{array}
$$

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

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

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

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:

$$
\begin{array}{rcccccccccl}
-103_{10} = & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & (SM) \\
 & & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & & \\
= & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & (C2) \\
\\
-68_{10} = & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & (SM) \\
 & & \Downarrow & \Downarrow & \Downarrow & \Downarrow & & & & \\
= & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & (C2)
\end{array}
$$

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

1.2.3 - Complementul de 2 (cont.)

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

1.2.3 - Complementul de 2 (cont.)

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

1.2.3 - Complementul de 2 (cont.)

Anomalia Complementului de doi:

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

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

- 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.

1.2.3 - Complementul de 2 (cont.)

Overflow aritmetic pentru operanzi cu semn (C2):

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

storing capacity

$$
\begin{array}{ll|l}
X=+19: & 0\ 1\ 0\ 0\ 1\ 1_{C2} & \\
Y=+14: & 0\ 0\ 1\ 1\ 1\ 0_{C2} & + \\
\hline
 & 1\ 0\ 0\ 0\ 0\ 1_{C2} & = \cancel{-31}\ \text{(overflow)}
\end{array}
$$

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

storing capacity

$$
\begin{array}{ll|l}
X=+19: & 0\ 0\ 1\ 0\ 0\ 1\ 1_{C2} & \\
Y=+14: & 0\ 0\ 0\ 1\ 1\ 1\ 0_{C2} & + \\
\hline
 & 0\ 1\ 0\ 0\ 0\ 0\ 1_{C2} & =+33
\end{array}
$$

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:

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

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:

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

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

Interpretarea lui Robertson se aplică și numerelOR pozitive:

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

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:

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

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.