Bazele Tehnologiei Informaţiei
Curs 3

Prof. dr. Răzvan Daniel Zota
Facultatea de Cibernetică, Statistică şi Informatică Economică
ASE Bucureşti

https://zota.ase.ro/bti

Reprezentarea numerelor în calculator

Reprezentarea în:

- virgulă fixă
- virgulă mobilă

Reprezentarea numerelor în virgula fixă

Reprezentarea în virgulă fixă

- cod direct (CD)
- cod invers (CI)
- cod complementar (CC)

Reprezentarea numerelor în cod direct (CD)

$$R^{CD} = a_n \bullet 2^n + \sum_{i=-m}^{n-1} a_i \bullet 2^i$$

$$a_n = \begin{cases} 0\,, dacă\ R \geq 0 \\ 1, dacă\ R < 0 \end{cases}$$

$$a_n - \text{bit de semn}$$

Exemplu de reprezentare pe 8 biți:

10=**0**0001010

2^8=256 valori

Domeniul de valori reprezentabile: [-128;127] (numere cu semn)

Numere fără semn [0;255]

Reprezentarea numerelor în cod invers (CI)

$$R^{CI} = \begin{cases} 0 \bullet 2^n + \sum_{i=-m}^{n-1} a_i \bullet 2^i \ , dacă \ R \geq 0 \\ 1 \bullet 2^n + \sum_{i=-m}^{n-1} \overline{a}_i \bullet 2^i , dacă \ R < 0 \end{cases}$$

$\overline{a}_i = 1 - a_i, \forall i = \overline{-m, n-1}$, unde a_i sunt cifrele binare ale lui $|R|$ în cod direct

Modalitate de calcul:

$$R^{CI} = 2^{n+1} - |R|_{CD} - 2^{-m}$$

Reprezentarea numerelor în cod complementar (CC)

$$R^{CC} = \begin{cases} 0 \bullet 2^n + \sum_{i=-m}^{n-1} a_i \bullet 2^i , \ dacă \ R \geq 0 \\ 1 \bullet 2^n + \sum_{i=-m}^{n-1} \overline{a}_i \bullet 2^i , dacă \ R < 0 \end{cases}$$

$\sum_{i=-m}^{n-1} \overline{a}_i \bullet 2^i = \sum_{i=-m}^{n-1} \overline{a}_i \bullet 2^i + 2^{-m}$, unde $\overline{a}_i = a_i - 1$ şi a_i sunt cifrele binare ale lui $|R|$ în cod direct

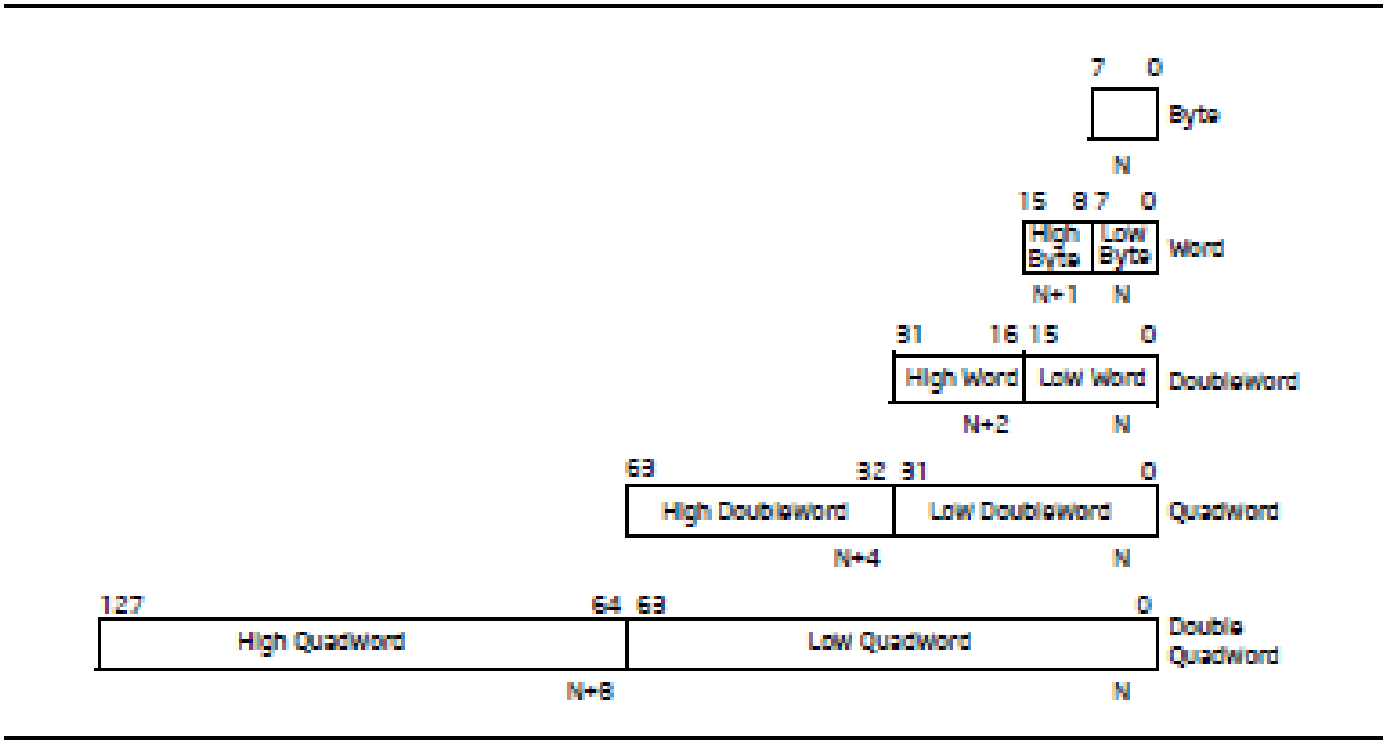
Modalitate de calcul:

$$R^{CC} = 2^{n+1} - |R|_{CD}$$

Adunarea/scăderea în virgulă fixă

- Adunarea în CD, CI şi CC (Ex. 93-27 în CI)
- Scăderea în CD, CI şi CC

Tipurile fundamentale de date



Lungimea, precizia şi domeniul datelor în virgulă fixă

Tipul	Lungime	Precizie	Domeniul de valori (binar)	Domeniul de valori (zecimal)
Format word	16	15	$-2^{15} – 2^{15}-1$	[-32768 ; 32767]
Format scurt	32	31	$-2^{31} – 2^{31}-1$	$[-2.14*10^{9} ; 2.14*10^{9}]$
Format lung	64	63	$-2^{63} – 2^{63}-1$	$[-9.22*10^{18} ; 9.22*10^{18}]$

Reprezentarea în virgulă mobilă

- În această reprezentare un număr are 3 părţi:
  - Bit de semn
  - Exponent (caracteristică)
  - Fracţie (mantisă)(significand=eng.)
- Standardul internaţional **IEEE (Institute of Electrical and Electronics Engineers) 754** –1985

Numere normalizate

- În majoritatea cazurilor, numerele sunt reprezentate în *formă normalizată*. Cu excepţia lui zero, numărul este reprezentat sub forma:

$$+/-1,fff\ldots fff * 2^{exp}$$

S=0 sau S=1

CAR = exp + K (K=constantă pozitivă)

Fracţie = fff…fff

Numere şi valori speciale

- **Zerouri cu semn** – valoarea 0 poate fi reprezentată ca +0 sau –0 în funcţie de bitul de semn. Ambele reprezentări sunt egale ca valoare. Semnul unui rezultat cu valoare 0 depinde de operaţia efectuată şi de modalitatea de rotunjire.
- **Numere finite normalizate şi denormalizate**.
- **+∞, -∞** reprezintă valoarea maximă pozitivă, respectiv negativă pentru numere reale ce poate fi reprezentată în virgulă mobilă. Valoarea *infinit* este totdeauna reprezentată de o fracţie 0 şi de exponentul maxim permis de formatul respectiv (de ex. 255 în format simplă precizie). Sunt generate excepţii atunci când utilizarea unei valori infinite ca operand sursă conduce la o operaţie invalidă.
- **Valori NaN** (Not a Number) – nu fac parte din mulţimea numerelor reale. Reprezentarea lor se face prin intermediul unui exponent maxim acceptat şi a unei fracţii non-zero. Bitul de semn este ignorat.

Numere normalizate si denormalizate

- Numerele diferite de zero finite. Numerele normalizate reprezintă numerele ce pot fi codificate într-o formă normalizată între 0 şi ∞. În tabelul următor, acest grup include toate numerele cu exponenţi modificaţi între 1 şi 254 (între –126 şi 127)
- Atunci când exponentul modificat este 0, numerele mai mici pot fi reprezentate făcând bitul părţii întregi zero. Numerele din acest domeniu se numesc numere *denormalizate*. Acest lucru duce la scăderea preciziei (numărul de biţi semnificativi ai fracţiei este redus datorită apariţiei zerourilor de la început).
- În momentul normalizării calculelor în virgulă mobilă, unitatea în virgulă mobilă operează cu numere normalizate şi produce rezultate normalizate. Numerele denormalizate reprezintă o condiţie de *underflow*. Un număr denormalizat este calculat prin intermediul unei tehnici denumită *gradual underflow*.

Valori reale şi NaN

-0	1	0	0
+0	0	0	0
- Denormalizat finit	1	0	0.fff
+ Denormalizat finit	0	0	0.fff
- Normalizat finit	1	1…254	Orice valoare
+ Normalizat finit	0	1…254	Orice valoare

Valori reale şi NaN (cont.)

- ∞	1	255	0
+ ∞	0	255	0
- SNaN	x	255	1.0ff
+ SNaN	x	255	1.0ff
- QNaN	x	255	1.1ff
+ QNaN	x	255	1.1ff

Procesul de denormalizare

Operatia	Semn	Exponent	Fractie
Rezultat real	0	-129	1.010111000...000
Denormalizare	0	-128	0.1010111000...000
Denormalizare	0	-127	0.01010111000...000
Denormalizare	0	-126	0.001010111000...000
Rezultat in forma denormalizata	**0**	**-126**	**0.001010111000...000**

Valori NaN

- Standardul IEEE defineşte două clase de NaN:
  - QNaN (quiet NaN) – are bitul CMS setat
  - SNaN (signaling NaN) – are bitul CMS zero.
- Valorile QNaN se propagă prin operaţiile aritmetice fără a indica o excepţie
- Valorile SNaN semnalizează în general o excepţie (operaţie invalidă) atunci când apar ca operanzi în operaţii aritmetice

Operaţii speciale

Operaţia	**Rezultat**
n / ±∞	0
± ∞ * ± ∞	± ∞
±Val.nonzero / 0	± ∞
∞ + ∞	∞
±0 / ±0	*NaN*
∞ - ∞	*NaN*
± ∞ / ± ∞	*NaN*
± ∞ * 0	*NaN*

Tipuri de date reale

Simplă precizie

CAR = exp + 127

Dublă precizie

CAR = exp + 1023

Tipuri de date reale (cont.)

Format real extins

Dacă notăm cu *n* numărul de biţi alocaţi caracteristicii, atunci putem utiliza formula:

CAR = exp + 2^{n-1} - 1

Lungimea, precizia şi domeniul datelor reale

Tipul	Lungime	Precizie	Domeniul de valori (binar)	Domeniul de valori (zecimal)
Simpla precizie	32	24	2^{-126} - 2^{127}	$1.18*10^{-38}$ – $3.40*10^{38}$
Dubla precizie	64	53	2^{-1022} - 2^{1023}	$2.23*10^{-308}$ – $1.79*10^{308}$
Real extins	80	64	2^{-16382} - 2^{16383}	$3.37*10^{-4932}$ – $1.18*10^{4932}$

Exemplu

1. Ce valoare are numărul (în format simplă precizie) reprezentat astfel:
1 10000001 01000000000000000000000

Caracteristica este 129, deci exponentul real este 129 – 127 = 2

Partea fracţionară este $.01_2$ = .25, deci vom avea valoarea 1,25.

Numărul este negativ (bitul de semn este 1)

Rezultatul: $-1{,}25 \times 2^2 = -5$

Exemplu

2. Care este reprezentarea numărului 16,625 în format simplă precizie?

Aducem numărul în forma normalizată:

$16 = 10000_2$

$0,625 = 0,101_2$

$16,625 = 10000,101 = 1,0000101 * 2^4$

$CAR = 4 + 127 = 131 = 128+3; (128=2^7)$

128=10000000+

3= 11

În concluzie, reprezentarea este:

0 10000011 00001010000000000000000

Reprezentarea numerelor în format BCD

BCD (Binary Coded Decimal) – zecimal codificat binar

Format:

- împachetat (packed BCD)
- despachetat (unpacked BCD)

În format **împachetat** se reprezintă 2 cifre zecimale pe un octet (cifra cmps pe biţii 0-3 şi cifra zecimală cms pe biţii 4-7):

96 = 1 0 0 1 0 1 1 0

7 4 3 0

În format **despachetat** se reprezintă o cifră zecimală pe un octet memorată pe biţii 0-3, iar biţii 4-7 conţin informaţia F_h:

6 = 1 1 1 1 0 1 1 0

7 4 3 0

Reprezentarea numerelor în format BCD la Intel

Tipul	Lungime	Precizie	Domeniul de valori (zecimal)
BCD împachetat	80	18 (cifre zecimale)	$(-10^{18}+1) – (10^{18}-1)$

Adunarea numerelor în BCD

- Adunarea în BCD – adunare obişnuită în binar, pentru fiecare grup de câte 4 cifre binare, avându-se în vedere următoarele cazuri. Dacă a şi b sunt cele două cifre zecimale codificate în binar, rezultatul c=a+b este:
  - Corect, dacă 0000 < c <=1001
  - Incorect, şi se adună 0110 astfel:
    - 1010 <= c <=1111 – nu corespunde unei cifre zecimale (adunarea lui 0110 va determina transport la rangul următor)
    - 0000 <= c < 1001, cu apariţia celei de-a 5-a cifre binare, 1, care reprezintă transport pentru tetrada superioară

Exemplu de adunare în BCD

Exemplu: Să se efectueze în BCD suma 5683 + 2794.

5683 +	0101 +	0110 +	1000 +	0011 +
2794	0010	0111	1001	0100
8477	0001 ←	0001 ←	←1 0001 +	0111
	1000	1110 +	0110	7
	8	0110	0111	
		←1 0100	7	
		4		

Mai sus transportul apărut din tetrada anterioară este evidențiat prin „←”.

Scăderea numerelor în BCD

- Scăderea în BCD – scădere obişnuită în binar, pentru fiecare grup de câte 4 cifre binare, avându-se în vedere următoarele cazuri:
- Dacă a şi b sunt cele două cifre zecimale codificate în binar, rezultatul **c = a - b** este:
  - corect, dacă c > 0
  - Dacă c < 0 atunci se face împrumut de la tetrada superioară, se face scăderea, apoi se scade valoarea de corecţie 0110