Universitatea Politehnica Timişoara
P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Logică digitală

-Curs 2-

REPREZENTAREA DATELOR ÎN SISTEME DE CALCUL

Reprezentarea numerelor în sistemele de calcul

- Sisteme de numerație poziționale (binar, octal, hexazecimal);
- Reprezentarea numerelor în virgulă fixă (SM, C1, C2);
- Reprezentarea numerelor de virgulă flotantă;
- Coduri binare pentru numere zecimale;

Sisteme de numere poziționale

- **Sistem poziţional** - un număr este reprezentat printr-un şir de cifre, unde fiecare poziţie a unei cifre este asociată o anumită greutate.

- Valoarea numărului este o sumă a cifrelor înmulţită cu greutatea aferentă:

  Ex1: $1734 = 1*10^3+7*10^2+3*10^1+4*10^0$

- Fiecare greutate e o putere a lui 10 corespunzătoare poziţiei numărului. Pentru numere cu virgulă folosim puteri negative a lui 10.

Ex2: $5186.67=5*10^3+1*10^2+8*10^1+6*10^0+6*10^{-1}+7*10^{-2}$

Sisteme de numere poziționale

- **Sistem poziţional** - un număr este reprezentat printr-un şir de cifre, unde fiecare poziţie a unei cifre este asociată o anumită greutate (pondere).

$$D = d_{m-1}\, d_{m-2} \ldots d_1\, d_0 \,.\, d_{-1}\, d_{-2} \ldots d_{-n}$$

MSD — Virgula fixă — LSD

$$D = d_{m-1} \cdot 10^{m-1} + \ldots + d_0 \cdot 10^0 + d_{-1} \cdot 10^{-1} + \ldots + d_{-n} \cdot 10^{-n}$$

Sisteme de numere binare

- Formă generală a unui număr binar

$$D = b_{m-1} b_{m-2} \ldots b_1 b_0 . b_{-1} b_{-2} \ldots b_{-n}$$

MSB — Virgula fixă — LSB

Valoarea:

$$D = \sum_{i=-n}^{p-1} b_i * r^i \quad , \text{ unde } r = 2 \text{ radix(bază)}$$

Sisteme de numere binare

Ex.: $N=11001.011_2$

$N= 1*2^4+1*2^3+0*2^2+0*2^1+1*2^0+ 0*2^{-1}+1*2^{-2}+1*2^{-3}=25.375_{10}$

- Baza 8 corespunde sistemului octal. cifre {0,1,2,3,4,5,6,7}

- Baza 16 corespunde sistemului hexazecimal.

  cifre {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Binar, octal, hexazecimal ...

Zecimal	Binar	Octal	Hexazecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversia numerelor

- Conversia din binar în hexazecimal

  Ex: 010011110111.101001010

  - Partiţionarea numărului binar în grupuri de 4 pornind de la virgulă şi inaintand spre dreapta sau stanga :

    0100_1111_0111 . 1010_0101_0000

  - Fiecare grup corespunde unei singure cifre hexazecimale. Folosind Tabelul ant. obţinem:

    4F7.D50

Aplicaţie: Convertiţi numărul din binar în hexazecimal:

11111001010.01111111

Raspuns : FCA.7F

Conversia numerelor

Conversia din binar în octal

- Partiţionarea numărului binar în grupuri de 3 pornind de la virgulă şi inaintand spre dreapta sau stanga.
- Fiecare grup corespunde unei singure cifre din octal.
- Aplicaţie: 111010.11

  111_010.110

  Raspuns: 72.6

Conversia numerelor

- Conversia din zecimal în binar

  - Conversia unui număr întreg din zecimal în binar se face prin algoritmul împărţire succesive ale lui N (nr de convertit) la r (noua bază).

  - Procesul se repetă pânã se obţine câtul **0**.

Conversia numerelor

- Conversia din zecimal în binar

Ex: Conversia nr zecimal 119:

119 : 2 = 59 rest 1 (LSB)
59 : 2 = 29 rest 1
29 : 2 = 14 rest 1
14 : 2 = 7 rest 0
7 : 2 = 3 rest 1
3 : 2 = 1 rest 1
1 : 2 = **0** rest 1 (MSB)

Condiția de oprire

$119_{10} = 1110111_2$

Conversia numerelor

Conversia numărului 73 din baza 10 în baza 2

$N9 = 73_{10} = 1001001_2$

$73/2 = 36$	rest	1	b_0
$36/2 = 18$	rest	0	b_1
$18/2 = 9$	rest	0	b_2
$9/2 = 4$	rest	1	b_3
$4/2 = 2$	rest	0	b_4
$2/2 = 1$	rest	0	b_5
$1/2 = 0$	rest	1	b_6

Conversia numerelor

- Conversia din zecimal în binar
  - Pentru conversia numerelor fracţionale, se înmulţeşte numărul cu noua bază în care se converteşte numărul.
  - Partea întreagă a rezultatului devine bit al şirului care reprezintă rezultatul conversiei.
  - Înmulţirea se face până rezultatul devine 0.

Ex: Conversia lui 0.75 în baza 2.

0.75*2= 1.5 parte întreagă 1 (MSB) si fracţionară 0.5

0.5 * 2= 1.0 parte întreagă 1 si fracţionară 0

0.0 * 2= 0.0 parte întreaga 0 (LSB)

Conversia numerelor

Conversia numărului 0.8125 din baza 10 în baza 2

$$N10 = 0.8125_{10} = 0.1101_2$$

$0.8125 * 2 = 1.625$	parte întreagă	1	b_{-1}
$0.625 * 2 = 1.25$	parte întreagă	1	b_{-2}
$0.25 * 2 = 0.5$	parte întreagă	0	b_{-3}
$0.5 * 2 = 1$	parte întreagă	1	b_{-4}

Reprezentarea numerelor în virgulă fixă

- Sistemele digitale sunt realizate din circuite ce procesează cifrele binare „0” şi „1”
- **Numerele fără semn**:
  - un şir de „0” şi „1”.
  - Fiecare cifră binară poartă denumirea de bit (**b**inary dig**it**).
  - Val. şirului binar $N = b_{n-1}b_{n-2}...b_1b_0.b_{-1}b_{-2}...b_{m-1}$
  - este dată de formula: $$N = \sum_{i=-m}^{n-1} b_i * 2^i$$

Reprezentarea numerelor în virgulă fixă

- Pentru reprezentarea numerelor cu semn, se alocă bitul cel mai semnificativ (cel mai din stânga – most significant bit - MSB) semnului:
  - „0” - numere pozitive,
  - „1” - numere negative.
- Uzual în sistemele de calcul se operează fie cu numere întregi, fie cu numere fracţionare;
- Din acest motiv, poziţia virgulei este considerată implicit după cum urmează:
  - numere întregi poziţia virgulei este la dreapta bitului cel mai puţin semnificativ: $b_{n-1}b_{n-2}...b_1b_0.$
  - numere fracţionare poziţia virgulei este la dreapta bitului cel mai semnificativ, care este şi bitul de semn:

$$b_0.b_{-1}...b_{-n+1}b_{-n}$$

Reprezentarea numerelor în virgulă fixă

- Pentru reprezentarea numerelor cu semn există trei formate de reprezentare:
  - semn-mărime - **SM**,
  - complement de 1 – **C1**,
  - complement de 2 – **C2**.

Reprezentarea numerelor în virgulă fixă

- Reprezentarea în semn-mărime/sign-magnitude (SM)
  - 1 bit semn, n biți mărime (valoare absolută)
  - Valoarea bitului de semn determină semnul
    - 0 – numere pozitive
    - 1 – numere negative
  - domeniul valoric a reprezentării în formatul semn-mărime acesta este între $-2^{n-1}+1$ şi $2^{n-1}-1$
  - Exemplu:

    +85= 0 1010101

    -85= 1 1010101

Reprezentarea numerelor în virgulă fixă

- Semn-mărime/sign-magnitude (SM)
  - Avantaje
    - simplitate
    - negare simpla prin schimbarea bitului de semn
    - implementare facilă a operaţiei de înmulţire
  - Dezavantaje
    - Dublă reprezentare pentru 0 (+0 și -0)
    - Implementare dificilă pentru adunare
      - Exercițiu: (-19) + (+12)

Reprezentarea numerelor în virgulă fixă – SM

- Ex.:

			Semn	Mărime				
+	25	=	0	1	1	0	0	1
-	25	=	1	1	1	0	0	1

+	2	=	0	0	1	0	+			
+	3	=	0	0	1	1				
			0	1	0	1	=	+5		

a.

-	2	=	1	0	1	0	+			
-	3	=	1	0	1	1				
			0	1	0	1	=	+5	(!)	

b.

-	2	=	1	0	1	0	+			
+	3	=	0	0	1	1				
			1	1	0	1	=	-5	(!)	

c.

+	2	=	0	0	1	0	+			
-	3	=	1	0	1	1				
			1	1	0	1	=	-5	(!)	

d.

Reprezentarea numerelor în virgulă fixă

- Complement de 1/One's complement (C1)
  - 1 bit semn, *n* biți pentru mărime
  - Numerele pozitive – identic cu SM
  - Numerele negative – complementarea/ negarea valorii pozitive
  - Exemplu:

    +85= 0 1010101

    -85= 1 0101010

Reprezentarea numerelor în virgulă fixă – **C1**

- Dezavantaje:
  - C1 nu este un format ponderat în conformitate cu notaţia poziţională
  - există **două** reprezentări pentru numărul zero (pentru un numar reprezentat pe 6 biţi avem 0 00000, respectiv 1 11111), deci testarea pentru zero se va face de două ori
- Avantaje:
  - o implementare mai facilă a operaţiei de adunare comparativ cu SM
- domeniul valoric pentru numere întregi:

$$-2^{n-1}-1 \text{ și } 2^{n-1}-1$$

Reprezentarea numerelor în virgulă fixă – **C1**

+	2	=		0		0	1	0	+		-	2	=		1		1	0	1	+	
+	3	=		0		0	1	1			+	3	=		0		0	1	1		
				0		1	0	1	=	+5				**1**	0		0	0	0	+	
														EAC					1		
															0		0	0	1	=	+1
			a.											**c.**							

-	2	=		1		1	0	1	+		+	2	=		0		0	1	0	+	
-	3	=		1		1	0	0			-	3	=		1		1	0	0		
			1	1		0	0	1	+						1		1	1	0	=	-1
			EAC					1													
				1		0	1	0	=	-5											
			b.											**d.**							

Reprezentarea numerelor în virgulă fixă

- Complement de 2/ Two's complement (C2)
  - Numerele pozitive – identic cu SM
  - Numerele negative – negarea valorii pozitive la care se adaugă 1
  - Întregi: $1\overline{b_{n-2}}...\overline{b_1}\,\overline{b_0} + 0.0...01$
  - Fracționare: $1.\overline{b_{-1}}...\overline{b_{-n+1}}\,\overline{b_{-n}} + 0.0...01$
  - Exemplu:

    +85= 0 1010101

    -85= 1 0101011

Reprezentarea numerelor în virgulă fixă – **C2**

- Dezavantaje:
  - Mai dificil de obținut decât SM și C1;
  - nu este un format ponderat în conformitate cu notaţia poziţională
  - anomalia complementului de doi
- Avantaje:
  - O singura reprezentare pentru 0 !
    - 0000000
  - Implementarea facilă a operației de adunare
    - Exercițiu (-19) + (+12)

Reprezentarea numerelor în virgulă fixă – **C2**

Număr zecimal	Format SM	Format C1	Format C2
+3	0 11	0 11	0 11
+2	0 10	0 10	0 10
+1	0 01	0 01	0 01
+0	0 00	0 00	0 00
-0	1 00	1 11	
-1	1 01	1 10	1 11
-2	1 10	1 01	1 10
-3	1 11	1 00	1 01
-4	----	----	1 00

Reprezentarea numerelor în virgulă fixă – **C2**

+	2	=	0	0	1	0	+		
+	3	=	0	0	1	1			
			0	1	0	1	=	+	5

a.

-	2	=	1	1	1	0	+		
-	3	=	1	1	0	1			
			1	0	1	1	=	-	5

b.

-	2	=	1	1	1	0	+		
+	3	=	0	0	1	1			
			0	0	0	1	=	+	1

c.

+	2	=	0	0	1	0	+		
-	3	=	1	1	0	1			
			1	1	1	1	=	-	1

d.

- Domeniul valoric pentru numere întregi:
  - -2^{n-1} și $2^{n-1}-1$

Aplicație:

- Se dau următoarele perechi de numere întregi: +23 şi +18, +23 şi -18, -23 şi +18, respectiv -23 şi -18. Se cere:
  - Să se convertească numerele în formatele semn-mărime, complement de 1, respectiv complement de 2.
  - Să se efectueze adunarea celor două numere.

Reprezentarea numerelor în virgulă flotantă (mobilă)

- reprezentate folosind notaţia ştiinţifică (care nu este poziţională) → un domeniu valoric foarte mare.
- Pentru a reprezenta un număr in virgulă flotantă folosim trei numere conform relaţiei:

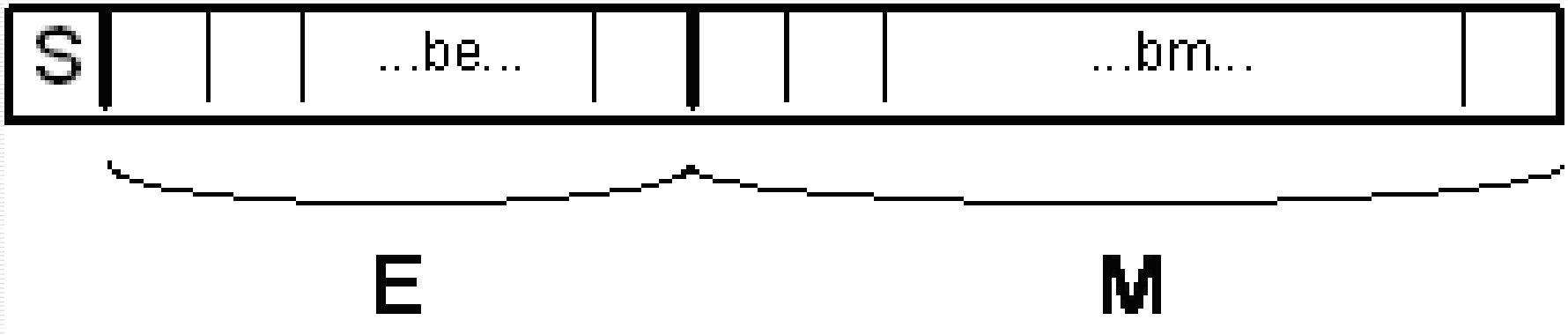
$$N = M * B^{E}$$

M - mantisa. (*M* poate fi reprezentată în SM sau C2)

B - baza (de obicei e 2 sau o putere a lui 2)

E - exponent. (E este reprezentat în SM sau cod exces)

Reprezentarea numerelor în virgulă flotantă



M - mantisa. (*M* poate fi reprezentată în SM sau C2)

B - baza (de obicei e 2 sau o putere a lui 2)

E - exponent. (E este reprezentat în SM sau cod exces)

Reprezentarea numerelor în virgulă flotantă

- Reprezentarea mantisei:
  - Reprezentarea lui 18:

$$18*10^0 = 1.8*10^1 = 0.18*10^2 = \ldots = 0.0\ldots018*10^n$$

  - *Obs.: M*, *B* şi *E* au o infinitate de valori posibile
  - Pentru o tratare unitară şi eficientă prin prisma a procesării în sistemele de calcul → o reprezentare **unică** → normalizarea mantisei *M*

Reprezentarea numerelor în virgulă flotantă – Normalizarea mantisei

- *M* în SM primul bit din dreapta virgulei trebuie să fie 1.
- *M* în C2 şi M conține:
  - valoare negativă atunci primul bit din dreapta virgulei trebuie să fie 0.
  - valoare pozitivă, atunci folosim regula de la SM.

Reprezentarea numerelor în virgulă flotantă – Normalizarea mantisei

Operaţia de normalizare presupune:

- mutări succesive ale virgulei la dreapta, urmată de incrementarea exponentului;
- Primul bit (bitul de semn) este 1, asadar numărul este negativ → primul bit din dreapta virgulei trebuie să fie 0

$$1.11100011_{C2} \rightarrow 11.1100011_{C2} * 2^1 \rightarrow 111.100011_{C2} * 2^2$$

$$\rightarrow 1111.00011_{C2} * 2^3$$

Reprezentarea numerelor în virgulă flotantă – Normalizarea mantisei

Operaţia de normalizare presupune:

- mutări succesive ale virgulei la stânga, urmată de decrementarea exponentului;

$$0.00011101_{SM} \rightarrow 00.0011101 * 2^{-1} \rightarrow 000.011101 * 2^{-2}$$

$$\rightarrow 0000.11101 * 2^{-3}$$

Reprezentarea numerelor în virgulă flotantă

- Reprezentarea exponentului:
  - Problema reprezentării lui 0 în virgulă mobilă.

$$0 = 0 * B^{E}$$

  - ... mai multe variante de reprezentare
  - zero trebuie să fie cât mai uşor de detectat şi testat → (?) şir de biţi de '0'
  - Dar... în calcule recurgem la aproximări
    - este posibil ca în urma unor calcule (FP), datorită acestor aproximări successive, să obţinem în loc de 0 un număr foarte mic (M≠0).

Reprezentarea numerelor în virgulă flotantă

- Pentru a minimiza eroarea→ exponentul aferent lui 0 să fie cel mai mic posibil.
- valoarea minimă a **orcărui** exponent să fie 0.
- Toate valorile negative reprezentabile pe N biţi sunt deplasate (devin pozitive) prin adunarea unui bias (unui surplus) = valoarea absolută a celui mai mic număr reprezentabil pe N numărul de biţi exponent.
- Pentru exponent reprezentat în:
  - SM pe 8 biți → valoarea bias-ului este 127
  - C2 pe 8 biți → valoarea bias-ului este 128

Reprezentarea numerelor în virgulă flotantă – reprezentarea exp.

Reprezentare binară	Valoare fără semn	Valoare cu semn (reală-cea a numărului reprezentat)	
		Bias = 127	Bias = 128
11111111	255	+128	+127
11111110	254	+127	+126
. . .	. . .	. . .	. . .
10000001	129	+2	+1
10000000	128	+1	0
01111111	127	0	-1
01111110	126	-1	-2
. . .	. . .	. . .	. . .
00000001	1	-126	-127
00000000	0	-127	-128

Reprezentarea numerelor în virgulă flotantă

- Cele mai reprezentative standarde pentru virgula mobilă sunt:
  - IEEE 754
  - IBM S360/370

Reprezentarea numerelor în virgulă flotantă

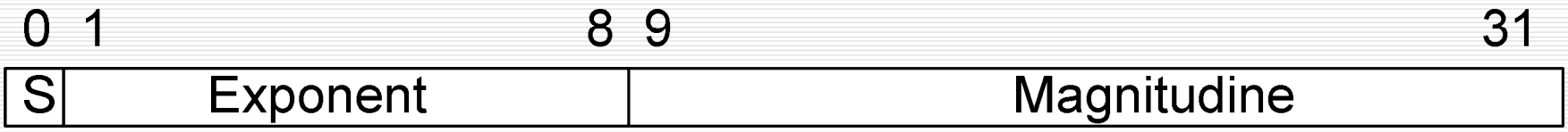
- Standardul IEEE 754/2008- formate:
  - Half precision
  - Simple precision
  - Double precision
  - Duble-extended

Reprezentarea numerelor în virgulă flotantă: IEEE 754

- Caracteristici:
  - E şi M – format SM
  - Exponentul este reprezentat în exces de:
    - 127 pentru simplă precizie
    - 1023 pentru dublă precizie.
  - *Hidden bit*.
    - Mantisa are un bit de 1 ascuns.
    - bitul la dreapta virgulei care trebuie să fie 1 (din condiţia de normalizare).
    - (S.**1**M) (unde S este semnul iar M este mantisa) → virgula a fost mutată la dreapta bit-ului de 1 cel mai semnificativ: S**1**.M

Reprezentarea numerelor în virgulă flotantă: IEEE 754 simplă precizie

- Simplă precizie: 32 biți
  - 1 bit de semn
  - 8 biţi de exponent; exponent reprezentat în exces de 127
  - 23 biţi de mantisă (significand)
- Formatul de număr este:



Ex.: Să se reprezinte în format IEEE 754 SP numărul 4.625

- Pasul 1: Se converteşte numărul în baza 2.

$$4.625 = 100.101 * \ 2^{0}$$

- Pasul 2: Normalizare → 1.M (mutarea virgulei cu ajustarea corespunzătoare a puterii lui 2)

$$100.101 = 1.00101 * \ 2^{2}$$

$$E = 2 + 127(exces) = 129 = 128 + 1 = 2^{7} + 2^{0} = 10000001$$

- Pasul 3:

S	E (8 biţi)	M (23 biţi)
0	10000001	0010 1000 0000 0000 0000 000

Reprezentarea numerelor în virgulă flotantă: IEEE 754 dublă precizie

- Reprezentare pe 64 de biţi:
  - 1 bit de semn
  - 11 biţi de exponent reprezentaţi în exces de 1023
  - 52 biţi de mantisă (significand)

Reprezentarea numerelor în virgulă flotantă: IEEE 754 valori speciale

Nr.	Exponent (E)	Mantisa (M)	Valoare speciala
1.	0	0	± 0
2.	0	$\neq 0$	Denormalized numbers
3.	255	0	$\pm \infty$
4.	255	$\neq 0$	NaN

- Nr. denormalizate: rezultat care este mai mic decât valoarea minimă reprezentabilă
- Infinit: situaţia în care rezultatul intermediar este infinit sau avem overflow
- 0/0 sau radical din nr. negativ

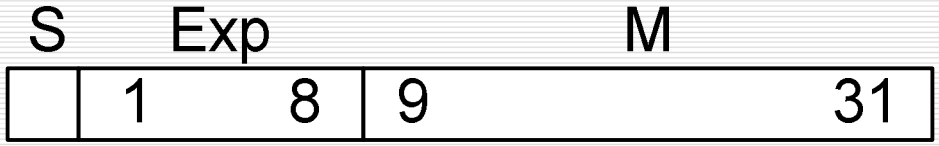
Reprezentarea numerelor în virgulă flotantă: IBM S360/370

- Mantisa - SM
- Baza este baza **16** →
  - domeniul valoric creşte foarte mult.
  - incrementarea/decrementarea exponentului atrage dupa sine mutarea virgulei la stânga sau la dreapta cu **patru** poziţii.
  - numărul se consideră normalizat atunci când prima cifră hexazecimală (adică primul grup de patru biţi) din dreapta virgulei este diferită de zero,
- Deosebiri față de IEEE 754 :
  - IBM nu prevede: valori speciale si excepţii, cum sunt cele definite de standardul IEEE 754 (NaN, overflow, numere denormalizate).
  - Zero este reprezentat ca un sir continuu de zerouri.
  - Nu exista hidden bit!
  - Baza este 16, şi nu 2!

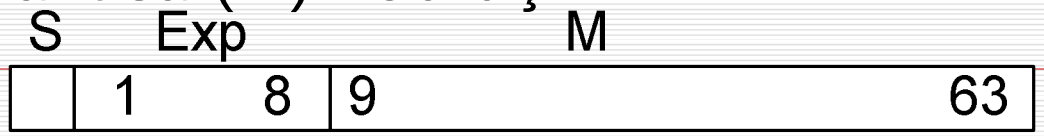
Reprezentarea numerelor în virgulă flotantă: IBM S360/370

Formatul de simplă precizie :

- Semn (S) – 1 bit
- Exponent (Exp) – 7 biţi (in exces de 64)
- Mantisa (M)– 24 biţi



- **Formatul de dubla precizie** are următoarele câmpuri:
  - Semn (S)– 1 bit
  - Exponent (Exp)– 7 biţi (in exces de 64)
  - Mantisa (M)– 56 biţi



Coduri binare pentru numere zecimale

(!) Cod (code),	O colecţie de şiruri diferite pe *n* biţi, iar fiecare dintre aceste şiruri are o semnificatie (reprezintă un număr, caracter, etc.) poartă denumirea de *cod* (code). Numărul maxim de cuvinte ale unui cod pe n biţi este 2^n. Nu întotdeauna însă, toate aceste combinaţii posibile pe *n* biţi sunt folosite (fac parte din colecţia de şiruri care alcătuiesc codul).
Cuvânt al unui cod (code word)	Un şir al colecţiei care reprezintă o combinaţie de *n* valori de *0* sau *1* se numeşte cuvânt de cod (code word).

Coduri binare pentru numere zecimale - BCD

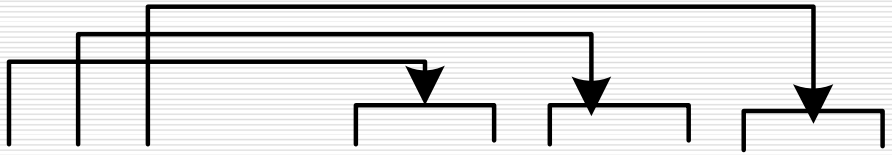
- există situaţii cand se doreşte afişarea rezultatelor de către interfeţele externe ale dispozitivului de calcul într-un format uşor de înţeles (decodificat) de către utilizator – şi anume mult întrebuinţatul format zecimal;
- cel mai la îndemână cod zecimal este BCD (binary-code decimal):
  - reprezentarea unei cifre BCD → înlocuirea cu reprezentarea în binar care îi corespunde→ cu un nr. pe 4 biţi

Coduri binare pentru numere zecimale - BCD

Cifră zecimală	Corespondent BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Coduri binare pentru numere zecimale - BCD

- conversia unui număr zecimal în BCD prin înlocuirea succesivă a cifrelor zecimale cu tetradele corespunzătoare



$$N_1 = 4\ 5\ 9_{10} = \quad 0100\ \ 0101\ \ 1001_{BCD}$$

- operaţia inversă de transformare a unui număr reprezentat în BCD în omologul zecimal

$$N_2 = 1000\ \ 0111\ \ 0000\ \ 0010_{\mathrm{BCD}} = \quad 8\ 7\ \ 0\ 2_{10}$$

Coduri binare pentru numere zecimale - BCD

- Avantaje:
  - simplitate
  - este un cod poziţional, ponderea fiecărei cifre fiind $10^j * 2^i$, unde j reprezintă poziţia tetradei zecimale, iar i poziţia bitului în cadrul tetradei
- Dezavantaje:
  - utlizează un număr mai mare de biţi faţă de reprezentarea binară a numărului respectiv;
    - Fol. 3 cifre BCD - 12 biţi (3 cifre zecimale * 4 biţi/cifră) se pot reprezenta 10^3 =1000 de numere;
    - folosind reprezentarea binară, pe 12 biţi se pot reprezenta 2^{12} =4096 de numere;
  - implementare anevoiasă a operaţiei de adunare

Coduri binare pentru numere zecimale – Exces de 3

- Exces de 3 se obţine din codul BCD astfel:
  - la fiecare cifră reprezentată în cod BCD se adună valoarea 3 (0011 în binar).
  - fiecare cifră zecimale se reprezintă cu ajutorul unei combinaţii de 4 biţi (o tetradă de biţi)

Coduri binare pentru numere zecimale – Exces de 3

Cifră zecimală	Corespondent exces de 3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Coduri binare pentru numere zecimale – Exces de 3

- Dezavantaje:
  - obţinerea reprezentării în excess de 3 este mai complicată comparativ cu reprezentarea BCD
  - nu este un cod poziţional
- Avantaje:
  - permite o implemenare mai facilă a operaţiei de adunare față de codul BCD

$b_3b_2b_1b_0$	$b_6b_5b_4$ 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

American Standard Code for Information Interchange (ASCII)

NUL	Null
SOH	Start of heading
STX	Start of text
ETX	End of text
EOT	End of transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horizontal tab
LF	Line feed
VT	Vertical tab
FF	Form feed
CR	Carriage return
SO	Shift out
SI	Shift in
SP	Space
DLE	Data link escape
DC1	Device control 1
DC2	Device control 2
DC3	Device control 3
DC4	Device control 4
NAK	Negative acknowledgement
SYN	Synchronize
ETB	End transmission block
CAN	Cancel
EM	End of medium
SUB	Substitute
ESC	Escape
FS	File separator
GS	Group separator
RS	Record separator
US	Unit separator
DEL	Delete or rubout

litera **A** are primele 3 pozitii (765) secventa 100, iar pe urmatoarele 4 poz

$b_3b_2b_1b_0$	$b_6b_5b_4$ 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011								
0100								
0101								
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

litera **A** are primele 3 pozitii (765) secventa 100, iar pe urmatoarele 4 pozitii (4321) secventa 0001. Deci A=(1000001) !

NUL	Null
SOH	Start of heading
STX	Start of text
ETX	End of text
EOT	End of transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horizontal tab
SP	Space
DLE	Data link escape
DC1	Device control 1
DC2	Device control 2
DC3	Device control 3
DC4	Device control 4
NAK	Negative acknowledgement
SYN	Synchronize
ETB	End transmission block
CAN	Cancel
EM	End of medium
SUB	Substitute
ESC	Escape
FS	File separator
GS	Group separator
RS	Record separator
US	Unit separator
DEL	Delete or rubout

American Standard Code for Information Interchange (ASCII)

Întrebări?

Enough Talking Let's Get To It
!!Brace Yourselves!!