

LB (curs 3 - S3)

Sisteme de numere pozitionale

- **Sistem pozițional** - un număr este reprezentat printr-un șir de cifre, unde fiecare poziție a unei cifre este asociată o anumită contribuție (pondere).

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

MSD

Virgula fixă

LSD

Sisteme de numere limitate

$$b = b_{m-1} b_{m-2} \dots b_1 b_0 . b_{-1} b_{-2} \dots b_{-n}$$

MSB

virgula
fixă

LSB

$$b = \sum_{i=-n}^{m-1} b_i \cdot r^i, \text{ unde } r = 2 \text{ radian (baza)}$$

Ex.: $N = 11001.011_2$

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

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

- Baza 16 corespunde sistemului hexazecimal.
cifre $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

- Conversia din binar în hexazecimal

Ex: 010011110111.101001010

- Partiționarea numărului binar în grupuri de 4 pornind de la virgulă și înaintând spre dreapta sau stanga :

0100_1111_0111 . 1010_0101_0000

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

4F7.A50

Aplicație: Convertiți numărul din binar în hexazecimal:

111 1100 1010.0111 1111

Raspuns : 7CA.7F

Conversia din binar în octal

- Partiționarea numărului binar în grupuri de 3 pornind de la virgulă și înaintând spre dreapta sau stanga.
- Fiecare grup corespunde unei singure cifre din octal.
- Aplicație: 111010.11
111_010.110
Răspuns: 72.6

☐ Conversia din zecimal în binar

- Pentru conversia numerelor fracționale, se **înmulțește** numărul cu noua bază în care se convertește numărul.
- Partea întreagă a rezultatului devine bit al șirului care reprezintă rezultatul conversiei.
- Înmulțirea se face până rezultatul devine 0.

Ex: Conversia lui 0.75 în baza 2.

$0.75 \cdot 2 = 1.5$ parte întreagă 1 (MSB) și fracționară 0.5

$0.5 \cdot 2 = 1.0$ parte întreagă 1 și fracționară 0

$0.0 \cdot 2 = 0.0$ parte întreagă 0 (LSB)

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 al reprezentării în formatul semn-mărime acesta este între $-2^{n-1} + 1$ și $2^{n-1} - 1$
- Exemplu:
 $+85 = 0 \ 1010101$
 $-85 = 1 \ 1010101$

☐ Semn-mărime/sign-magnitude (SM)

- Avantaje
 - ☐ simplitate
 - ☐ negare simplă prin schimbarea bitului de semn
 - ☐ implementare facilă a operației de înmulțire
- Dezavantaje
 - Dublă reprezentare pentru 0 (+0 și -0)
 - Implementare dificilă pentru adunare
 - Exercițiu: $(-19) + (+12)$

Complement de 1

- 1 bit semn, n biți pentru mărime
- Numerele pozitive – identic cu SM
- Numerele negative – complementarea/negarea mărimei

■ Exemplu:

$$+85 = 0\ 1010101$$

$$-85 = 1\ 0101010$$

□ Dezavantaje:

- C1 nu este un format ponderat în conformitate cu notația pozițională
- există două reprezentări pentru numărul zero (pentru un număr reprezentat pe 6 biți avem 0 00000, respectiv 1 11111), deci testarea pentru zero se va face de două ori

□ Avantaje:

- o implementare mai facilă a operației de adunare comparativ cu SM

□ domeniul valoric pentru numere întregi:

$$-(2^{n-1} - 1) \text{ și } 2^{n-1} - 1$$

+	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																			

Complement de 2

- Numerele pozitive – identic cu SM
- Numerele negative – negarea valorii pozitive la care se adaugă 1
- Întregi: $1\overline{b_{n-2}...b_1b_0} + 0.0...01$
- Fraționare: $1.\overline{b_{-1}...b_{-n+1}b_{-n}} + 0.0...01$
- Exemplu:
 - +85 = 0 1010101
 - 85 = 1 0101011

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

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

$$\begin{array}{r}
 + 2 = 0 \quad 0 \quad 1 \quad 0 \quad + \\
 + 3 = 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 0 \quad 1 \quad 0 \quad 1 = + 5
 \end{array}$$

a.

$$\begin{array}{r}
 - 2 = 1 \quad 1 \quad 1 \quad 0 \quad + \\
 + 3 = 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 0 \quad 0 \quad 0 \quad 1 = +
 \end{array}$$

c.

$$\begin{array}{r}
 - 2 = 1 \quad 1 \quad 1 \quad 0 \quad + \\
 - 3 = 1 \quad 1 \quad 0 \quad 1 \\
 \hline
 1 \quad 0 \quad 1 \quad 1 = - 5
 \end{array}$$

b.

$$\begin{array}{r}
 + 2 = 0 \quad 0 \quad 1 \quad 0 \quad + \\
 - 3 = 1 \quad 1 \quad 0 \quad 1 \\
 \hline
 1 \quad 1 \quad 1 \quad 1 = -
 \end{array}$$

d.

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

□ Se dau următoarele perechi de numere întregi: +23 și +18, +23 și -18, -23 și +18, respectiv -23 și -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.

N_n	SM	C_1	C_2
23	010111	010111	010111
18	010010	010010	010010
-18	110010	101101	101110
-23	110111	101000	101001

$$\begin{array}{r|l}
 \begin{array}{r} +23 \\ +18 \end{array} & \begin{array}{l} S \\ 0 \\ 0 \end{array} \quad \begin{array}{r} \overset{\wedge}{10111} \\ \overset{\wedge}{10010} \end{array} \\
 \hline
 41 & 0 \quad 101001 = 41
 \end{array}$$

SM - de obicei nu se poate dacă e mai mult de 1 nr < 0

$$\begin{array}{r|l}
 \begin{array}{r} +23 \\ -18 \end{array} & \begin{array}{l} \overset{\wedge}{0} \\ 1 \end{array} \quad \begin{array}{r} \overset{\wedge}{10111} \\ \overset{\wedge}{01101} \end{array} \quad \begin{array}{l} C_1 \\ C_2 \end{array} \\
 \hline
 5 & 0 \quad 00100 \\
 & + \quad 1 \\
 \hline
 & 0 \quad 00101 = 5 \Rightarrow -5 = 111010
 \end{array}$$

$$\begin{array}{r|l}
 \begin{array}{r} -23 \\ +18 \end{array} & \begin{array}{l} 101000 \\ 010010 \end{array} \quad \begin{array}{l} C_1 \\ C_2 \end{array} \\
 \hline
 -5 & 111010 \quad 111011
 \end{array}$$

$$\begin{array}{r|l}
 \begin{array}{r} -18 \\ -23 \end{array} & \begin{array}{l} 101101 \\ 101000 \end{array} \quad \begin{array}{l} C_1 \\ C_2 \end{array} \\
 \hline
 -41 & 1010101 \quad 1010111 \\
 & \underline{\quad \quad \quad} \\
 & 1010110 \quad 1010110
 \end{array}$$

Reprezentarea numerelor în virgulă flotantă

- reprezentate folosind notația științifică (care nu este pozițională) → un domeniu valoric foarte mare.
- Pentru a reprezenta un număr în virgulă flotantă folosim trei numere conform relației:

$$N = M * B^E$$

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)



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 mantisei:

■ Reprezentarea lui 18:

$$18 * 10^0 = 1.8 * 10^1 = 0.18 * 10^2 = \dots = 0.0\dots018 * 10^n$$

■ Obs.: M , B și E au o infinitate de valori posibile

■ Pentru o tratare unitară și eficientă prin prisma procesării în sistemele de calcul → o reprezentare unică → normalizarea mantisei M

□ M în SM primul bit din dreapta virgulei trebuie să fie 1.

□ M în C2 și M corespunde la o:

■ valoare negativă atunci primul bit din dreapta virgulei trebuie să fie 0.

■ valoare pozitivă, atunci folosim regula de la SM.

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

$$0 = 0 * B^E$$

- ... mai multe variante de reprezentare
- zero trebuie să fie cât mai ușor de detectat și testat → (?) șir de biți de '0'
- Dar... în calcule recurgem la aproximări
 - este posibil ca în urma unor calcule (FP), datorită acestor aproximări successive, să obținem în loc de 0 un număr foarte mic ($M \neq 0$).
- Pentru a minimiza eroarea → exponentul aferent lui 0 să fie cel mai mic posibil.
- valoarea min. a oricărui exponent să fie 0.
- Toate valorile negative reprezentabile pe N biți sunt deplasate (devin pozitive) prin adunarea unui bias (unui surplus) = valoarea absolută a celui mai mic număr reprezentabil pe N numărul de biți exponent.
- Pentru exponent reprezentat în:
 - SM pe 8 biți → valoarea bias-ului este 127
 - C2 pe 8 biți → valoarea bias-ului este 128

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

Cele mai reprezentative standarde pentru virgula mobilă sunt

1. IEEE 754

2. IBM 5360/370

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

❑ **Caratteristiche:**

- E și 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 condiția de normalizare).
 - (S.1M) (unde S este semnul iar M este mantisa)
→ virgula a fost mutată la dreapta bit-ului de 1 cel mai semnificativ: S1.M

- ❑ Simplă precizie: 32 biți

- 1 bit de semn
- 8 biți de exponent; exponent reprezentat în exces de 127
- 23 biți de mantisă (significand)

❑ Formatul de număr este:

0 1	8 9	31
S	Exponent	Magnitudine

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

- ❑ Pasul 1: Se convertește numărul în baza 2.

$$4.625 = 100.101 * 2^0$$

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

$$100.101 = 1.00101 * 2^2$$

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

- Pasul 3:

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

Reprezentarea numerelor în virgulă flotantă: IEEE 754 dublă precizie

- Reprezentare pe 64 de biți:

- 1 bit de semn
- 11 biți de exponent reprezentați în exces de 1023
- 52 biți de mantisă (significand)

0 1	12 13	63
S	Exponent	Magnitudine

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: situația în care rezultatul intermediar este infinit sau avem overflow
- 0/0 sau radical din nr. negativ

(!)	O colecție de șiruri diferite pe n biți, iar fiecare dintre aceste șiruri are o semnificație (reprezintă un număr, caracter, etc.) poartă denumirea de <i>cod</i> (code). Numărul maxim de cuvinte ale unui cod pe n biți este 2^n . Nu întotdeauna însă, toate aceste combinații posibile pe n biți sunt folosite (fac parte din colecția de șiruri care alcătuiesc codul).
Cuvânt al unui cod (code word)	Un șir al colecției care reprezintă o combinație de n valori de 0 sau 1 se numește cuvânt de cod (code word).

Coduri limitate pentru numere zecimale - BCD

- există situații când se dorește afișarea rezultatelor de către interfețele externe ale dispozitivului de calcul într-un format ușor de înțeles (decodificat) de către utilizator - și anume mult întrebuițatul format zecimal;
- cel mai la îndemână cod zecimal este BCD (binary-code decimal):
 - reprezentarea unei cifre BCD \rightarrow înlocuirea cu reprezentarea în binar care îi corespunde \rightarrow cu un nr. pe 4 biți

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

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

$$N_1 = 459_{10} = \begin{array}{|c|c|c|} \hline 4 & 5 & 9 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0100 & 0101 & 1001 \\ \hline \end{array}_{BCD}$$

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

$$N_2 = \begin{array}{|c|c|c|c|} \hline 1000 & 0111 & 0000 & 0010 \\ \hline \end{array}_{BCD} = \begin{array}{|c|c|c|c|} \hline 8 & 7 & 0 & 2 \\ \hline \end{array}_{10}$$

- Avantaje:
 - simplitate
 - este un cod pozițional, ponderea fiecărei cifre fiind $10^j \cdot 2^i$, unde j reprezintă poziția tetradiei zecimale, iar i poziția bitului în cadrul tetradiei
- Dezavantaje:
 - utilizează un număr mai mare de biți față de reprezentarea binară a numărului respectiv;
 - Fol. 3 cifre BCD - 12 biți (3 cifre zecimale * 4 biți/cifră) se pot reprezenta $10^3 = 1000$ de numere;
 - folosind reprezentarea binară, pe 12 biți se pot reprezenta $2^{12} = 4096$ de numere;
 - implementare anevoioasă a operației de adunare

Exces de 3

- Exces de 3 se obține din codul BCD astfel:
 - la fiecare cifră reprezentată în cod BCD se adună valoarea 3 (0011 în binar).
 - fiecare cifră zecimală se reprezintă cu ajutorul unei combinații de 4 biți (o tetradă de biți)

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

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

American Standard Code for Information Interchange (ASCII)

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