

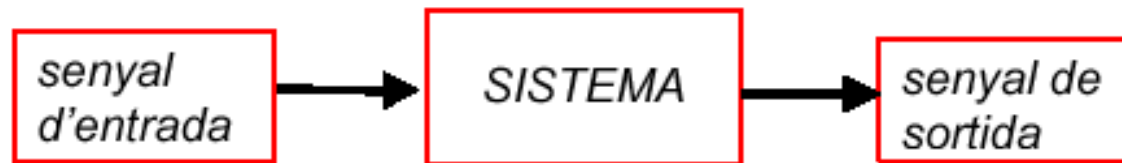
INTRODUCCIÓ ALS SISTEMES ELECTRÒNICS DIGITALS

Índex de conceptes

- **Tipus de sistemes digitals**
- **Numeració i bases**
- **Codis**
- **Aritmètica binària**
- **Representació de nombres negatius**

Característiques dels sistemes digitals

Sistemes electrònics



Sistema electrònic: sistema de transmissió o processament de la informació, on la informació es representa mitjançant magnituds elèctriques

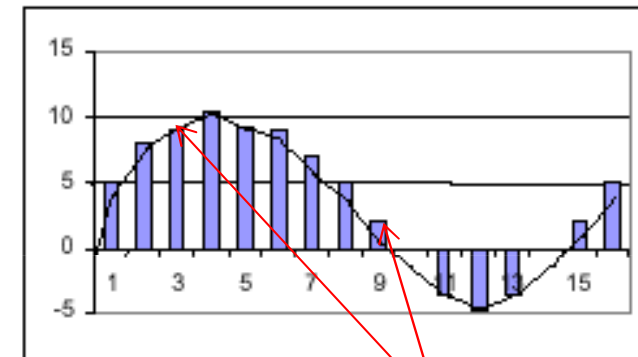
Sistema analògic: sistema en el qual les magnituds prenen valors continus

Sistema digital: sistema en el qual les magnituds només prenen valors discrets (nombre finit de valors diferents)

Digitalització d'un senyal = conversió a números

Sistema binari: només hi ha dos valors possibles (0, 1)

Variable binària: variable que pot prendre només dos valors diferents



Prenc mostres del senyal analògic i li assignem números

Digitalització del senyal

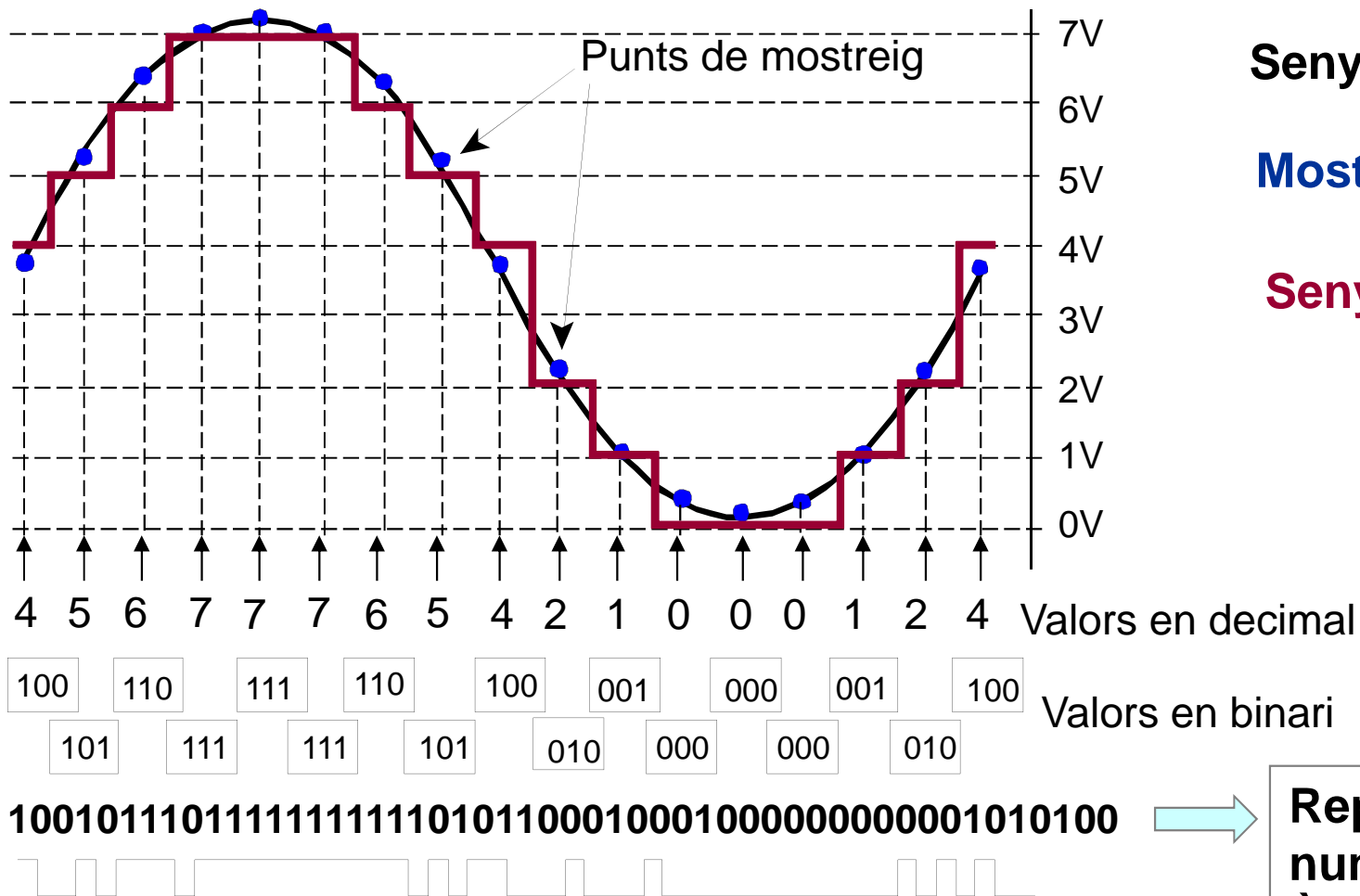
+ nivells:

+ rang dinàmic

+ precisió

+ bits

+ espai memòria



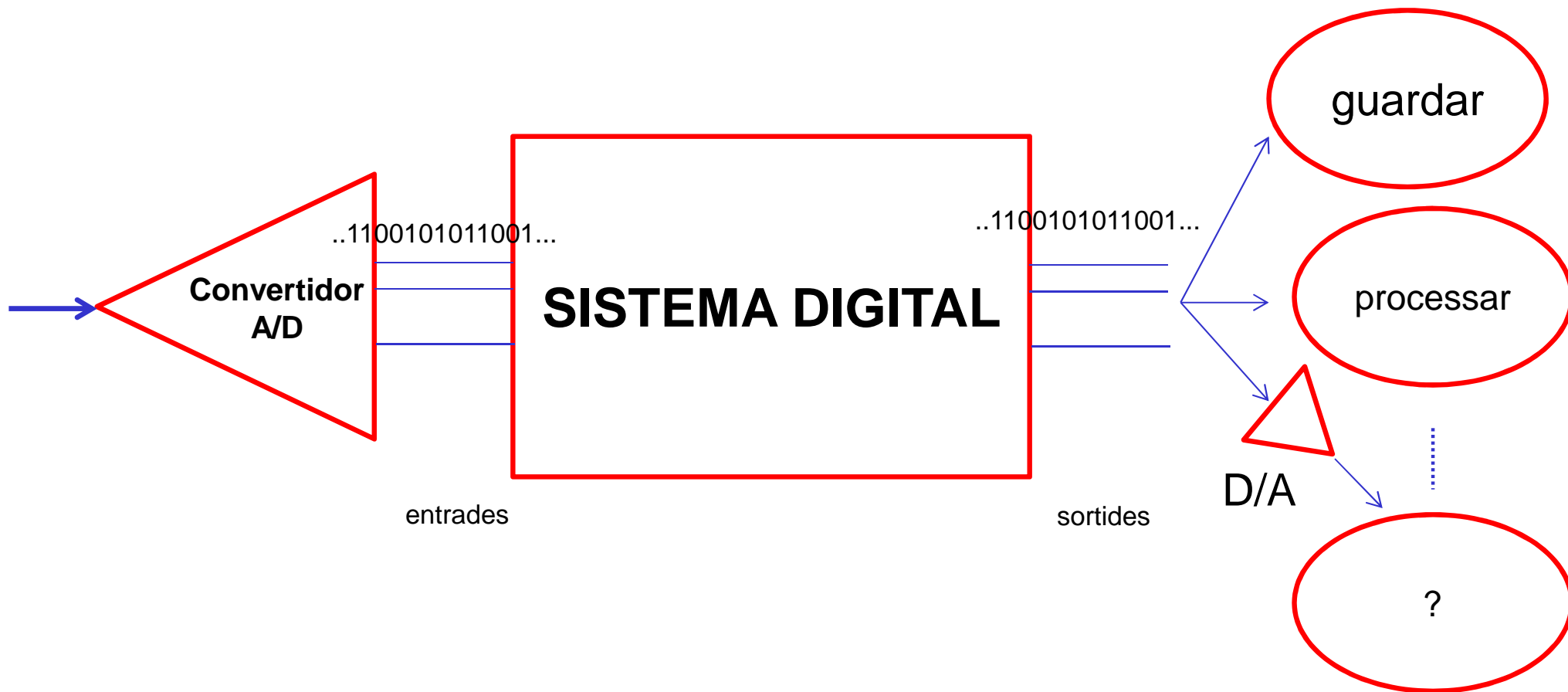
Senyal analògic

Mostres

Senyal digitalitzat

+ freqüència: + precisió; + espai memòria; + complicat

Representació numèrica:
Àmplies possibilitats



Tipus de sistemes digitals

Sistemes combinacionals

la sortida només depèn de l'entrada actual

$$y(t_i) = F[x(t_i)]$$

x = entrada
y = sortida

Sistemes seqüencials

la sortida depèn de l'entrada actual i de la història passada

$$y(t_i) = F[x(t_i), x(-\infty, t_i)]$$

Descripció de la història en termes d'estats

Estat: identifica
qualsevol situació
diferenciada

$$y(t_i) = G[x(t_i), s(t_i)]$$

Funció de sortida

$$s(t_{i+1}) = H[x(t_i), s(t_i)]$$

Funció de transició a estat següent

Sistemes de representació numèrica

Base b

$$N = \underbrace{a_n b^n + a_{n-1} b^{n-1} + \dots + a_0 b^0}_{\text{part entera}} + \underbrace{a_{-1} b^{-1} + \dots + a_{-p} b^{-p}}_{\text{part fraccionària}}$$

part entera

part fraccionària

on $0 \leq a_i < b$

$n + 1$ és el nombre de dígitos enters

p és el nombre de dígitos fraccionaris

Bases més habituals:

binària	(2)
octal	(8)
decimal	(10)
hexadecimal	(16)

Per què aquestes bases són especials?

Exemples

$$(11100001)_2 = (1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1)_2 = (225)_{10}$$

$$(341)_8 = (3 \times 8^2 + 4 \times 8 + 1)_{10} = (225)_{10}$$

$$(E1)_{16} = (E \times 16 + 1)_{10} = (14 \times 16 + 1)_{10} = (225)_{10}$$

$$87.54_{10} = 8 \times 10^1 + 7 \times 10^0 + 5 \times 10^{-1} + 4 \times 10^{-2}$$

$$1101.11_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 13.75_{10}$$

Els termes o coeficients a_i al sistema binari poden valer 0 o 1 i reben el nom de **bit** (unitat mínima d'informació)

byte = grup de 8 bits (2^3)

Kilobit = 2^{10} bits = 1024 $\approx 10^3$ bits

Megabit = 2^{20} bits = 1.048.576 $\approx 10^6$ bits

Gigabit = 2^{30} = 1.073.741.824 $\approx 10^9$ bits

Decimal	Binari	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	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
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
32	100000	40	20
50	110010	62	32
60	111100	74	3C
64	1000000	100	40
100	1100100	144	64
255	11111111	377	FF
1000	1111101000	1750	3E8

Conversió entre sistemes

a) De qualsevol base a base 10

Representar el polinomi característic i operar-lo en base 10

veure exemples anteriors

b) De base 10 a qualsevol base

Tractem separatament les parts **sencera** i fraccionària

- **Part sencera** representada a la nova base b:

$$N_E = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \dots + a_2b^2 + a_1b^1 + a_0b^0$$

Si dividim per b ens quedarà com a reste el terme a_0 (bit menys significatiu, LSB)

$$N'_E = N_E / b = (a_{n-1}b^{n-2} + a_{n-2}b^{n-3} + \dots + a_2b^1 + a_1) + a_0 / b$$

Si tornem a dividim per b ens quedarà a_1

$$N'_E / b = (a_{n-1}b^{n-3} + a_{n-2}b^{n-4} + \dots + a_2b) + a_1 / b$$

I així successivament fins al final

Exemple

quocient			resta		coeficient
524 : 2	=	262	0	=	a ₀
262 : 2	=	131	0	=	a ₁
131 : 2	=	65	1	=	a ₂
65 : 2	=	32	1	=	a ₃
32 : 2	=	16	0	=	a ₄
16 : 2	=	8	0	=	a ₅
8 : 2	=	4	0	=	a ₆
4 : 2	=	2	0	=	a ₇
2 : 2	=	1	0	=	a ₈

$$524_{10} = 1000001100_2$$

Comprovació: passem-ho a base 10

$$2^9 + 2^3 + 2^2 = 512 + 18 + 4$$

b) De base 10 a qualsevol base

Tractem separatament les parts sencera i **fraccionària**

- **Part fraccionària** representada a la nova base b:

$$N_F = a_{-1}b^{-1} + a_{-2}b^{-2} + \dots + a_{-p}b^{-p}$$

Si multipliquem per b, la part sencera del resultat correspondrà al terme a_{-1} (el més significatiu)

$$N'_F = N_F \cdot b = a_{-1} + a_{-2}b^{-1} + \dots + a_{-p}b^{-p+1}$$

Si tornem a multiplicar per b la part sencera serà a_{-2}

$$N'_F \cdot b = a_{-2} + \dots + a_{-p}b^{-p+2}$$

Exemple

	resultat	part entera
$0.825 \times 2 = 1.65$		1
$0.650 \times 2 = 1.30$		1
$0.300 \times 2 = 0.60$		0
$0.600 \times 2 = 1.20$		1
$0.200 \times 2 = 0.40$		0
$0.400 \times 2 = 0.80$		0
$0.800 \times 2 = 1.60$		1

$$0.825_{10} = 0.1101001_{..2}$$

Així, la conversió de $524,825_{10}$ a base 2 és:

$$524.825_{10} = 1000001100.1101001_{..2}$$

Exemple

Conversió de base 10 a base 16

Conversió : $58506.8435_{10} \Rightarrow N_{16}$

Part entera

58506 :	16	= 3656	resta	10 (A)
3656 :	16	= 228		8
228 :	16	= 14 (E)		4

Part fraccionària

0.8435 x	16	= 13.496	resta	D	$58506.8435_{10} = E48A.D7EF_{16}$
0.496 x	16	= 7.936		7	
0.936 x	16	= 14.976		E	
0.976 x	16	= 15.616		F	

c) De base 2 a base 2^k

Exemple $k=3$:

$$\begin{aligned} & (1010110001101011)_2 \\ &= (1\ 010\ 110\ 001\ 101\ 011)_2 \\ &= (126153)_8 \end{aligned}$$

Exemple $k=4$:

$$\begin{aligned} & (1010110001101011)_2 \\ &= (1010\ 1100\ 0110\ 1011)_2 \\ &= (AC6B)_{16} \quad \text{deu/dotze/sis/onze} \end{aligned}$$

$k=3$: s'agrupen els bits en grups de 3, començant pel menys significatiu i es converteixen a base 8 (0 a 7)

$k=4$: s'agrupen de 4 en 4 i es converteixen a base 16 (0 a 15)

d) De base 2^k a base 2

Es fa el procés invers (convertir cada dígit a base 2)

$$\begin{array}{ccccccccc} \text{(base 2 a base 8)} & 11011001.101100 & & 011 & 011 & 001 & . & 101 & 100 \\ & & & 3 & 3 & 1 & & 5 & 4 \end{array} = 331.54$$

$$\text{(hexadecimal a binari)} \quad FE91 = 1111 \ 1110 \ 1001 \ 0001$$

**Així es veu la facilitat de conversió
entre les bases 2, 8 i 16.**

Si el nostre sistema treballa sempre amb un número fix de n bits, direm que treballa amb paraules de n bits.

Codificacions més habituals

Codi binari natural

Exemple: *amb 3 bits*

<i>decimal</i>	<i>binari</i>
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Amb n bits podem codificar fins el nombre $2^n - 1$.

Codificació BCD

Codificació d'un dígit decimal:

<i>decimal</i>	<i>BCD</i>	<i>decimal</i>	<i>BCD</i>
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Codificació de més d'un dígit decimal:

exemple: $(84)_{10} = (1000\ 0100)_{\text{BCD}}$

És molt útil per representar números en displays (pantalles).

Inconvenient: amb el mateix número de bits es representen menys números que amb el binari natural

Codi BCD excés 3

<i>decimal</i>	<i>BCD excés 3</i>	<i>decimal</i>	<i>BCD excés 3</i>
0	0011	5	1000
1	0100	6	1001
2	0101	7	1010
3	0110	8	1011
4	0111	9	1100

Codi antisimètric

Codi de Gray

És un codi cíclic.

Construcció:

a) D'un bit:

decimal	Gray
0	0
1	1

b) De dos bits:

decimal	Gray
0	00
1	01
2	11
3	10

bit reflectit
bit afegit

c) De tres bits:

decimal	Gray
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

bits reflectits
bit afegit

Els números consecutius sempre difereixen en un sol bit

Codis alfanumèrics: ASCII de 7 bits

Exemples:

<i>Caràcter</i>	<i>Codi</i>	<i>Caràcter</i>	<i>Codi</i>
A	100 0001	0	011 0000
B	100 0010	1	011 0001
C	100 0011
...	9	011 1001
Z	101 1010	blank	010 0000
a	110 0001	!	010 0001
b	110 0010
...	>	011 1110
z	111 1010
		end text	000 0011

ASCII: American Standard Code for Information Interchange

Codis detectors d'errors

Exemple: *incorporació d'un bit de paritat*

<i>Codi binari</i>	<i>Bit de paritat</i>	<i>Codificació detectora</i>
000	0	0000
001	1	1001
010	1	1010
011	0	0011
100	1	1100
101	0	0101
110	0	0110
111	1	1111

Aritmètica Binària

En sistemes digitals les operacions es fan en codi binari, perquè el disseny de circuits per a operacions binàries és molt més senzill que per altres representacions

Suma Binària

La taula de sumar en binari és:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ i en portem 1 a la següent posició (carry)}$$

Exemple

n=5

$$\begin{array}{r} 13 \\ + 06 \\ \hline 19 \end{array}$$

$$\begin{array}{r} 11 \quad \leftarrow \text{carrys} \\ 01101 \\ 00110 \\ \hline 10011 \end{array}$$

Aritmètica Binària

Resta Binària

La taula de restar en binari és:

$$0 - 0 = 0$$

$$0 - 1 = 1 \quad \text{i en restem 1 a la següent posició (*carry*)}$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

Exemple

n=5

		1 ← <i>carry negatiu</i>
21	10101	
-19	10011	
<hr/>		
2	00010	

	111 ← <i>carrys negatius</i>
17	10001
-07	00111
<hr/>	
10	01010

Multiplicació Binària

La taula de multiplicar en binari és:

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

Si multipliquem nums. d'n bits cal anar desplaçant, com en decimal

11	1011
x9	1001
<hr/>	
	1011
	0000
	0000
	1011
<hr/>	
99	01100011

Observem que el resultat té una longitud més llarga:

Si multipliquem un nombre d'**n** bits per un d'**m** bits el resultat té **n+m** bits

Divisió Binària

La divisió és com la decimal però els coeficients només són 0 o 1

Exemple: dividim $145 : 11$ Quocient = 13 i el Reste = 2

10010001	quocient
1011	01
001110	
1011	1
001101	01
1011	
0010	

Es va comparant el divisor amb els 4 bits de més pes del dividend. Si és menor (hi cap) es resta i es posa un 1 al quocient. Si no hi cap es posa un 0 al quocient i es baixa el següent dígit i es torna a comparar....

Resultat $Q=01101$ (13_{10}); $R=0010$ (2_{10})

Representació de nombres negatius

Hi ha 3 mètodes per representar el signe de manera que sigui manejable per als sistemes digitals:

- Representació **signe-mòdul**
- Representació en **complement a 1**
- Representació en **complement a 2**

L'avantatge de representar els nombres negatius en Complements, és, com veurem, que **la resta es converteix en una suma**, per tant només cal implementar una operació (= **estalvi de cost**)

a) Representació signe-mòdul

Com que els ordinadors només utilitzen 0 i 1 lògic (2 nivells de tensió), cal indicar el signe amb un bit.

El més senzill és utilitzar el primer bit (MSB) com a bit de signe: 0 indica número positiu, 1 indica número negatiu:

$$N_{10} = b_{n-1} \cdot b_{n-2} \cdot \dots \cdot b_0 = (-1)^{b_{n-1}} (b_{n-2} \cdot 2^{n-2} + \dots + b_0)$$

El zero es pot representar com 0000 o 1000.

El marge dels nombres que es poden representar és:

$$(1 - 2^{n-1}) \leq N_{10} \leq (2^{n-1} - 1) \quad \text{on } n = \text{núm. de bits}$$

a) Representació signe-mòdul

$$N_{10} = b_{n-1} \cdot b_{n-2} \cdot \dots \cdot b_0 = (-1)^{b_{n-1}} (b_{n-2} \cdot 2^{n-2} + \dots + b_0)$$

Desavantatge: Amb signe-mòdul no es poden fer operacions, com la resta, de manera fàcil

$$\begin{array}{r} 4=0100 \\ -3=1011 \\ \hline \end{array} \quad \begin{array}{r} 4 \\ + -3 \\ \hline 1 \end{array} \longrightarrow \begin{array}{r} 0100 \\ +1011 \\ \hline 1111 \end{array} \quad \begin{array}{l} = -7 \\ ??? \end{array}$$

b) Representació amb complement

Consisteix en representar un enter negatiu amb un número complementari. Així **la resta de dos nombres es transforma, directament, en la suma d'un amb el complement a l'altre.**

Consisteix a representar un enter negatiu ($-Z$) amb un altre número format a partir d'una constant K i el mòdul del número, Z . El mòdul del complement es calcula com $K-Z$.

K depèn del nombre de bits i del tipus de complement.

b1) Representació de nombres negatius en complement a 1 (Ca1)

La constant K val: $K = 2^n - 1$

n = nombre de bits que s'utilitzen (incloent-hi el de signe)

El mòdul del número complementat val: $K - Z = (2^n - 1) - Z$

El bit de signe indica si és **negatiu (=1)** (està complementat)

El marge de valors que es poden representar amb n bits és:

$$-(2^{n-1} - 1) \leq N_{10} \leq (2^{n-1} - 1)$$

Un problema és que el **zero té dues representacions**:

exemple amb n=7 zero = 0000000 i zero = 1111111

Exemple

Representació de -7_{10} en Ca1 amb $n=4$ (4 bits)

$$\text{Ca1}(Z) = 2^n - 1 - Z$$

$$\begin{array}{rcl} 2^n & = & 10000 \\ -1 & & \underline{00001} \\ & & 01111 \\ -7 & & \underline{0111} \\ = & & 01000 = \text{Ca1}(7) \end{array}$$

El bit sobrant s'elimina

El bit de més pes indica el signe:
(1) Indica que el num. és negatiu

Representació de -12_{10} en Ca1 amb $n=5$

$$\begin{array}{rcl} 2^n & = & 100000 \\ -1 & & \underline{000001} \\ & & 011111 \\ -12 & & \underline{01100} \\ = & & 010011 = \text{Ca1}(12) \end{array}$$

Exemple

Mètode fàcil de complementar números a 1

Si comparem els números complementats i sense complementar

$$\begin{array}{lcl} 7 & = & 0111 \\ \text{Ca1}(7) & = & 1000 \end{array}$$

$$\begin{array}{lcl} 12 & = & 01100 \\ \text{Ca1}(12) & = & 10011 \end{array}$$

Què s'observa?

El Ca1 d'un nombre es pot obtenir intercanviant els 0 i els 1

b2) Representació de nombres negatius en complement a 2 (Ca2)

La constant K val: $K = 2^n$

n = nombre de bits que s'utilitzen (incloent-hi el de signe)

El mòdul del número complementat val: $K - Z = 2^n - Z$

El bit de signe indica si és **negatiu (=1)** (està complementat)

Aquí ja no hi ha problema amb el zero, doncs té una **única representació**: *exemple amb n=7* zero = 0000000

El marge de valors que es poden representar amb n bits és:

$$-\left(2^{n-1}\right) \leq N_{10} \leq \left(2^{n-1} - 1\right)$$

Exemple

Representació de -7_{10} en Ca2 amb $n=4$

$$\text{Ca2}(Z) = 2^n - Z$$

$$\begin{array}{rcl} 2^n & = & 10000 \\ -7 & & 0111 \\ \hline & = & \underline{11001} = \text{Ca2}(7) \end{array}$$

El bit sobrant de carry
s'elimina

El bit de més pes indica el signe:
(1) Indica que el num. és negatiu

Representació de -12_{10} en Ca2 amb $n=5$

$$\begin{array}{rcl} 2^n & = & 100000 \\ -12 & & 01100 \\ \hline & = & \underline{010100} = \text{Ca2}(12) \end{array}$$

Exemple

Mètode fàcil de complementar números a 2

Comparant els dos complements és evident que :

$$\text{Ca2}(N) = \text{Ca1}(N) + 1$$

Per tant una manera senzilla de complementar a 2 consisteix en intercanviar els 0 i 1 i sumar-li 1

Una altra manera: comparem els números amb els seus complements

$$\begin{array}{lcl} 7 & = & 0111 \\ \text{Ca2}(7) & = & 1001 \end{array}$$

$$\begin{array}{lcl} 12 & = & 01100 \\ \text{Ca1}(12) & = & 10100 \end{array}$$

Què s'observa?

exemple

0	1	0	1	1	0
↓	↓	↓	↓	↓	↓
1	0	1	0	1	0

El Ca2 d'un nombre es pot obtenir, començant per la dreta es deixen els bits igual fins a trobar el primer 1, a partir d'allà s'intercanvien els 0 per 1 i viceversa

Conversió de nombres en els diversos sistemes per a n=4

Enters positius		Enters negatius			
+N		-N	Signe-mòdul	Ca1	Ca2
+0	0000	-0	1000	1111	----
+1	0001	-1	1001	1110	1111
+2	0010	-2	1010	1101	1110
+3	0011	-3	1011	1100	1101
+4	0100	-4	1100	1011	1100
+5	0101	-5	1101	1010	1011
+6	0110	-6	1110	1001	1010
+7	0111	-7	1111	1000	1001
		-8	-----	-----	1000

Aritmètica Binària amb complements

Veiem com són les operacions amb nombres complementats

Resta en complement a 1

25	011001	minuend
-13 (001101)	110010	Ca1(substraend)
<hr/>		
	1001011	resultat parcial
	1	
12	<hr/>	
	001100	Resultat

Si es produeix un carry a la posició n+1, aquest es suma al resultat parcial.
End-around-carry

15	001111	minuend
-22 (010110)	101001	Ca1(substraend)
<hr/>		
-7	0111000	Ca1(resultat)

4	0100	-4	1011
+3	+0011	+(-3)	+1100
----	-----	----	-----
7	0111	-7	10111
			<u>1</u>
			1000 _{ca1} = -7 ₁₀

4	0100	-4	1011
+(-3)	1100	+3	0011
----	-----	----	-----
1	10000	-1	1110 _{ca1} = -1 ₁₀
	<u>1</u>		
	0001		

Resta en complement a 2

$$\begin{array}{rcl} 25 & = & 011001 \quad \text{Minuend} \\ -13 & = & 110011 \quad \text{Ca2(substraend)} \\ \hline 12 & = & (1)001100 \quad \text{Resultat} \end{array}$$

En aquest cas es produeix un desbordament quan el resultat és positiu, però **no** cal sumar-lo; simplement, s'ignora.

$$\begin{array}{rcl} 14 & = & 001110 \\ -22 & = & 101010 \quad \text{Ca2(22)} \\ \hline -08 & = & 111000 \quad \text{Ca2(8)} \end{array}$$

$$\begin{array}{r}
 4 \\
 +3 \\
 \hline
 7
 \end{array}
 \qquad
 \begin{array}{r}
 0100 \\
 +0011 \\
 \hline
 0111
 \end{array}$$

$$\begin{array}{r}
 -4 \\
 +(-3) \\
 \hline
 -7
 \end{array}
 \qquad
 \begin{array}{r}
 1100 \\
 +1101 \\
 \hline
 11001
 \end{array}$$

\swarrow \downarrow
 $1001_{ca2} = -7_{10}$

$$\begin{array}{r}
 4 \\
 +(-3) \\
 \hline
 1
 \end{array}
 \qquad
 \begin{array}{r}
 0100 \\
 1101 \\
 \hline
 10001
 \end{array}$$

\swarrow \downarrow
 0001

$$\begin{array}{r}
 -4 \\
 +3 \\
 \hline
 -1
 \end{array}
 \qquad
 \begin{array}{r}
 1100 \\
 0011 \\
 \hline
 1111
 \end{array}$$

$1111_{ca2} = -1_{10}$