# INTRODUCCIÓ ALS SISTEMES ELECTRÒNICS DIGITALS

## Índex de concepts

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

# Característiques dels sistemes digitals

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

Senyal d'entrada → Sistema electrònic → Senyal de sortida

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

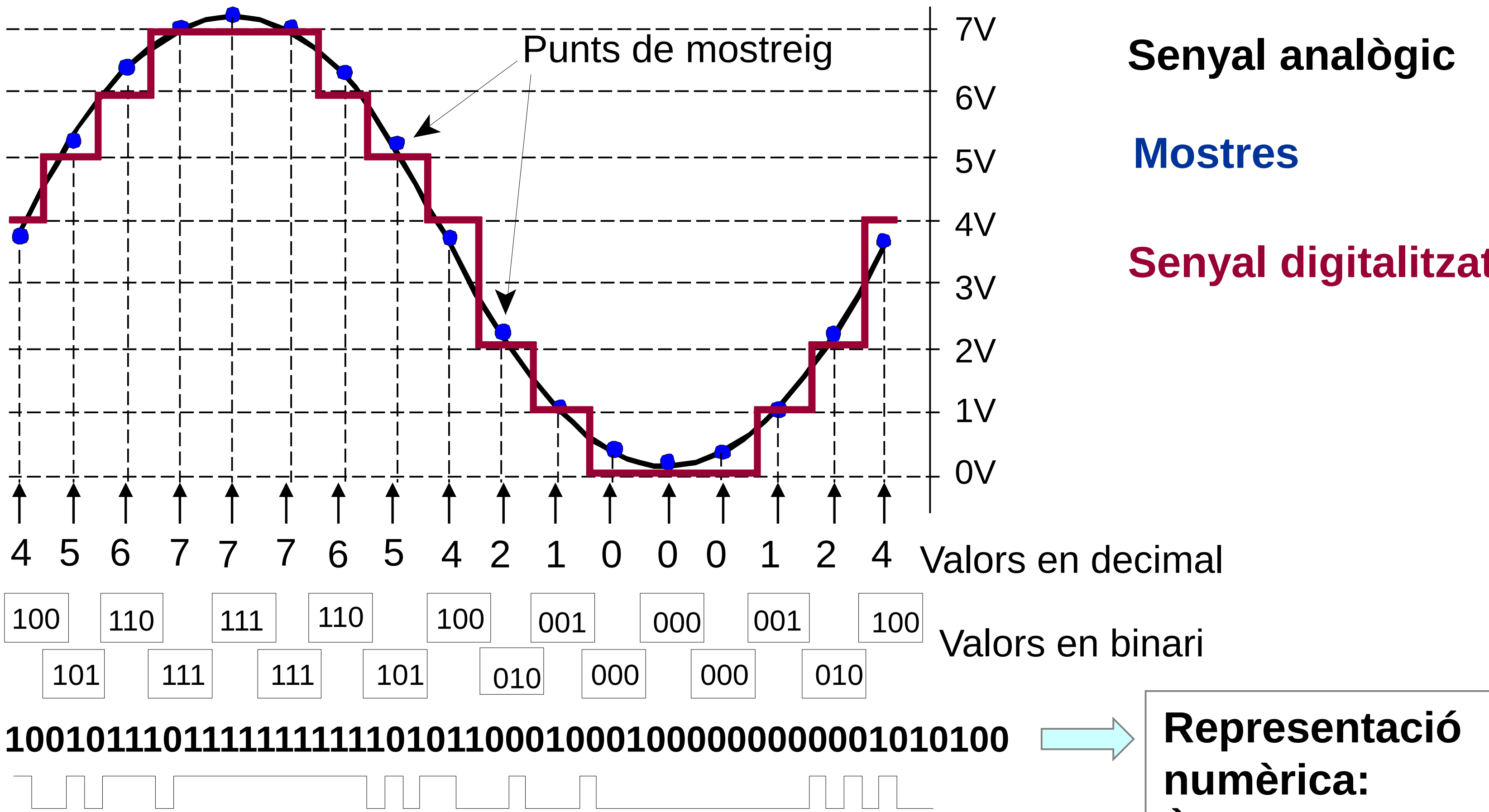
**Sistema digital:** sistema en el qual les magnituds només prenen valors discrets (nombre finit de valors diferents)

Digitalització d'un senyal = conversió a números

**Sistema binari:** només hi ha dos valors possibles, exemple: 0, 1 (la variable es binaria)

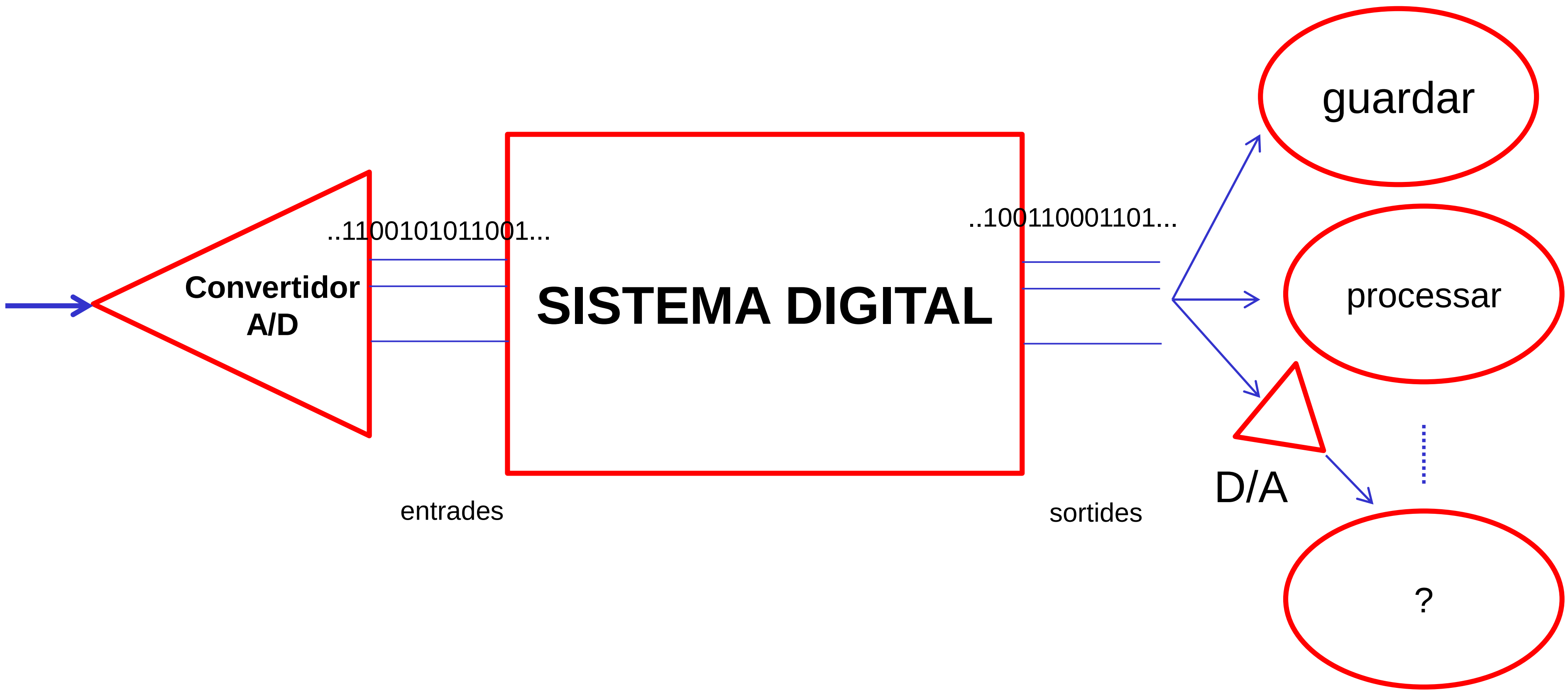
# Digitalització del senyal

**+ nivells:**
+ rang dinàmic
+ precisió
+ bits
+ espai memòria




**Representació numèrica: Àmplies possibilitats**

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


guardar
..1100101011001...
..100110001101...
Convertidor
A/D
SISTEMA DIGITAL
processar
D/A
entrades
sortides
?

# 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

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

**Estat:** identifica qualsevol situació diferenciada

**Descripció de la història en termes d'estats**

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

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

on $0 \leq a_i < b$

n + 1 és el nombre de dígits enters

p és el nombre de dígits fraccionaris

Bases més habituals:

| | |
|---|---|
| **binària** | (2) |
| octal | (8) |
| decimal | (10) |
| hexadecimal | (16) |

**Per què aquestes bases són especials?**

## Exemples

$$(11100001)_2=(1\times2^7 + 1\times2^6 + 1\times2^5 + 1)_{10}= (225)_{10}$$

$$(341)_8 = (3\times8^2 + 4\times8 + 1)_{10} = (225)_{10}$$

$$(E1)_{16} = (E\times16 + 1)_{10} = (14\times16 + 1)_{10} = (225)_{10}$$

$$87.54_{10} = 8x10^1 + 7x10^0 + 5x10^{-1} + 4x10^{-2}$$

$$1101.11_2 = 1.2^3 + 1.2^2 + 1.2^0 + 1.2^{-1} + 1.2^{-2} = 13.75_{10}$$

**Els termes o coeficients $a_i$ al <u>sistema binari</u> 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 ≈ $10^3$ bits**
**Megabit = $2^{20}$ bits = 1.048.576 ≈ $10^6$ bits**
**Gigabit = $2^{30}$ = 1.073.741.824 ≈ $10^9$ bits**

| | |
|---|---|
| 0 | 00000 |
| 1 | 00001 |
| 2 | 00010 |
| 3 | 00011 |
| 4 | 00100 |
| 5 | 00101 |
| 6 | 00110 |
| 7 | 00111 |
| 8 | 01000 |
| 9 | 01001 |
| 10 | 01010 |
| 11 | 01011 |
| 12 | 01100 |
| 13 | 01101 |
| 14 | 01110 |
| 15 | 01111 |

| | |
|---|---|
| 16 | 10000 |
| 17 | 10001 |
| 18 | 10010 |
| 19 | 10011 |
| 20 | 10100 |
| 21 | 10101 |
| 22 | 10110 |
| 23 | 10111 |
| 24 | 11000 |
| 25 | 11001 |
| 26 | 11010 |
| 27 | 11011 |
| 28 | 11100 |
| 29 | 11101 |
| 30 | 11110 |
| 31 | 11111 |

| 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 separadament les parts **sencera** i fraccionària

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

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

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

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

Si tornem a dividim per b ens quedarà $a_1$

$$N'_E / b = \left(a_{n-1}b^{n-3} + a_{n-2}b^{n-4} + \ldots + a_2\right) + a_1 / b$$

I així successivament fins al final

## Exemple conversió de $524{,}825_{10}$ a base 2

| | | | 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** = $a_9$ | **0** | = | $a_8$ |

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

Comprovació: passem-ho a base 10

$$2^9+2^3+2^2 = 512+8+4$$

## b) De base 10 a qualsevol base

Tractem separadament les parts sencera i **fraccionària**

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

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

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

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

Si trèiem $a_{-1}$ i tornem a multiplicar per b la part sencera serà $a_{-2}$

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

**Exemple** **conversió de $524,825_{10}$ a base 2**

| | 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

| | | | | |
|---|---|---|---|---|
| 0.843 x | 16 | = 13.496 | resta | **D** |
| 0.496 x | 16 | = 7.936 | | **7** |
| 0.936 x | 16 | = 14.976 | | **E** |
| 0.976 x | 16 | = 15.616 | | **F** |

$58506.8435_{10} = E48A.D7EF_{16}$

## c) De base 2 a base $2^k$

Exemple k=3 (octal):

$$(1010110001101011)_2$$
$$= (1\ 010\ 110\ 001\ 101\ 011)_2$$
$$= (126153)_8$$

Exemple k=4 (Hexadecimal):

$$(1010110001101011)_2$$
$$= (1010\ 1100\ 0110\ 1011)_2$$
$$= (AC6B)_{16}$$ *(deu/dotze/sis/onze)*

**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 a15)**

**d) De base $2^k$ a base 2**
**Es fa el procés invers (convertir cada dígit a base 2)**

3 3 1 5 4

octal a binari: $331{,}54_8 = 011\ 011\ 001,\ 101\ 100 = 11011011{,}1011_2$

F E 9 1

hexadecimal a binari: $FE91_{16} = 1111\ 1110\ 1001\ 0001 = 1111111010010001_2$

**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**.

# 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   i en portem 1 a la següent posició (**carry**)

---

**Exemple**

n=5

```
            11       <------      carrys
   13       01101
 + 06       00110
 ----       -----
   19       10011
```

# Aritmètica Binària

## Resta Binària

La taula de restar en binari és:

0 - 0 = 0
0 - 1 = 1   i en restem 1 a la següent posició (***carry***)
1 - 0 = 1
1 - 1 = 0

---

**Exemple**

n=5

| | | | | | |
|---|---|---|---|---|---|
| | 1 ← *carry* negatiu | | | 111 ← *carrys* negatius |
| 21 | 10101 | | 17 | 10001 |
| -19 | 10011 | | -07 | 00111 |
| 2 | 00010 | | 10 | 01010 |

# Multiplicació Binària

La taula de multiplicar en binari és:

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

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

```
11          1011
x9          1001
----------------
            1011
           0000
          0000
         1011
----------------
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 resta = 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**

  **-Z = Ca1(Z)**

- Representació en **complement a 2**

  **-Z = Ca2(Z)**

L'avantatge de representar els nombres negatius en Complements és que **la resta es converteix en una suma** i per tant només cal implementar un dispositiu que faci una única operació (= estalvi de cost)

## a) Representació signe-mòdul (SM)

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**:

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

El zero es pot representar com 0000... o 1000.... El marge dels nombres que es poden representar és:

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

Desavantatge: amb SM no es poden fer correctament operacions com la resta

4=0100

-3=1011

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

## 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 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 número complementat val: $K - Z = \left(2^n - 1\right) - Z$

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

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

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

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)

$$Ca1(Z) = 2^n - 1 - Z$$

```
2ⁿ  ------>  10000
-1         - 00001
             -----
             01111
-7         -  0111
             -----
=            01000 = Ca1(7)
```

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

```
2ⁿ  ------>  100000
-1         - 000001
             ------
             011111
-12        -  01100
             ------
=            010011 = Ca1(12)
```

**Exemple**

# Mètode fàcil de complementar números a 1

Si comparem els números complementats i sense complementar

| | | | | | |
|---|---|---|---|---|---|
| 7 | = | 0111 | 12 | = | 01100 |
| Ca1(7) | = | 1000 | Ca1(12) | = | 10011 |

**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 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

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

```
2^n ----------> 10000
-7            - 0111
                -----
=              X1001  = Ca2(7)
```

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

```
2^n ----------> 100000
-12      -       01100
                ------
=              X10100  = Ca2(12)
```

## Exemple

## Mètode fàcil de complementar números a 2

Comparant els dos complements és evident que :

$$Ca2(N) = Ca1(N) + 1$$

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

| | | | | | |
|---|---|---|---|---|---|
| 7 | = | 0111 | 12 | = | 01100 |
| Ca2(7) | = | 1001 | Ca2(12) | = | 10100 |

exemple

0 1 0 1 1 0

1 0 1 0 1 0

**Què s'observa?**

**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 negatius en els diversos sistemes per a n=4

| Enters negatius | | | |
|---|---|---|---|
| -N | Signe-mòdul | Ca1 | Ca2 |
| -0 | 1000 | 1111 | ---- |
| -1 | 1001 | 1110 | 1111 |
| -2 | 1010 | 1101 | 1110 |
| -3 | 1011 | 1100 | 1101 |
| -4 | 1100 | 1011 | 1100 |
| -5 | 1101 | 1010 | 1011 |
| -6 | 1110 | 1001 | 1010 |
| -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) |
| | 1001011 | resultat parcial |
| | +1 | |
| 12 | **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) |
| −7 | **111000** | resultat |

| | | | |
|---|---|---|---|
| 4 | 0100 | -4 | 1011 |
| +3 | +0011 | +(-3) | +1100 |
| ---- | --------- | ----- | -------- |
| 7 | 0111 | -7 | 10111 |
| | | | 1 |
| | | | $1000_{ca1} = -7_{10}$ |

| | | | |
|---|---|---|---|
| 4 | 0100 | -4 | 1011 |
| +(-3) | 1100 | +3 | 0011 |
| ------ | --------- | ---- | ------ |
| 1 | 10000 | -1 | $1110_{ca1} = -1_{10}$ |
| | 1 | | |
| | 0001 | | |

## Resta en complement a 2

| | | | |
|---|---|---|---|
| 25 | = | 011001 | minuend |
| -13 | = | +110011 | Ca2(substraend) |
| 12 | = | ($\cancel{1}$) 001100 | resultat |

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

| | | | |
|---|---|---|---|
| 14 | = | 001110 | minuend |
| -22 | = | +101010 | Ca2(substraend) |
| -08 | = | 111000 | resultat |

| | | | |
|---|---|---|---|
| 4 | 0100 | -4 | 1100 |
| +3 | +0011 | +(-3) | +1101 |
| ---- | --------- | ----- | -------- |
| 7 | 0111 | -7 | 11001 |
| | | | $1001_{ca2} = -7_{10}$ |

| | | | |
|---|---|---|---|
| 4 | 0100 | -4 | 1100 |
| +(-3) | 1101 | +3 | 0011 |
| ------ | --------- | ---- | ------- |
| 1 | 10001 | -1 | $1111_{ca2} = -1_{10}$ |
| | 0001 | | |

En general treballarem sempre en **Ca2** quan fem **operacions aritmètiques**

# Codificacions més habituals

## Codi binari natural

Exemple: *amb 3 bits*

| *decimal* | *binari* |
|---|---|
| ***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

BCD: Binary-Coded Decimal

Codificació d'un dígit decimal:

| decimal | BCD | decimal | BCD |
|---|---|---|---|
| *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:* $(84)_{10} = (1000\ \ 0100)_{BCD}$

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 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:

| *Caràcter* | *Codi* | *Caràcter* | *Codi* |
|---|---|---|---|
| ***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 |

<u>ASCII:</u> *American Standard Code for Information Interchange*

# Codis detectors d'errors

**Exemple:** *incorporació d'un bit de paritat*

| *Codi binari* | *Bit de paritat* | *Codificació detectora* |
|---|---|---|
| *000* | *0* | *0000* |
| *001* | *1* | *1001* |
| *010* | *1* | *1010* |
| *011* | *0* | *0011* |
| *100* | *1* | *1100* |
| *101* | *0* | *0101* |
| *110* | *0* | *0110* |
| *111* | *1* | *1111* |

## Codificacions SM, Ca1 i Ca2

*amb 4 bits*

| decimal | SM | Ca1 | Ca2 |
|---|---|---|---|
| 0 | 0000 | 0000 | 0000 |
| 1 | 0001 | 0001 | 0001 |
| 2 | 0010 | 0010 | 0010 |
| 3 | 0011 | 0011 | 0011 |
| 4 | 0100 | 0100 | 0100 |
| 5 | 0101 | 0101 | 0101 |
| 6 | 0110 | 0110 | 0110 |
| 7 | 0111 | 0111 | 0111 |
| -1 | 1001 | 1110 | 1111 |
| -2 | 1010 | 1101 | 1110 |
| -3 | 1011 | 1100 | 1101 |
| -4 | 1100 | 1011 | 1100 |
| -5 | 1101 | 1010 | 1011 |
| -6 | 1110 | 1001 | 1010 |
| -7 | 1111 | 1000 | 1001 |

# 10010011 ??

$10010011_2 =$ $2^7+2^4+2^1+2^0 = 128+16+2+1 = 147_{10}$

$10010011_{BCD} =$ $93_{10}$

$10010011_{16} =$ $16^7+16^4+16^1+16^0 = 268501009_{10}$

$10010011_8 =$ $8^7+8^4+8^1+8^0 = 2101257_{10}$

$10010011_{errores} =$ $2^4+2^1+2^0 = 19_{10}$ tres 1's = correcto

$10010011_{SM} =$ $-1\cdot(2^4+2^1+2^0) = -19_{10}$

$10010011_{Ca1} =$ $11101100_{SM} = -1\cdot(2^6+2^5+2^3+2^2) = -108_{10}$

$10010011_{Ca2} =$ $11101101_{SM} = -1\cdot(2^6+2^5+2^3+2^2+2^1) = -109_{10}$