2 · EL LLENGUATGE DELS ORDINADORS

2.1 Bases de Numeració i Còmput.

- Una base de numeració és el conjunt de símbols que utilitzam per a representar els valors numèrics.
- La més comuna, que tothom utilitza amb naturalitat, és la <u>decimal</u>, on hi ha 10 símbols. $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. $\#S = 10$
- Cada símbol, en un dígit, ocupa un lloc que li indica quina és la potència de la base per la que es multiplica, començant per zero i creixent cap a ~~la dreta~~ l'esquerra:

$$543 \rightarrow 5 \cdot 10^2 + 4 \cdot 10^1 + 3 \cdot 10^0$$

dígit de "més pes" — dígit de "menys pes"

- Una altra dada a tenir en compte: la <u>longitud màxima</u> dels números amb els que treballam. Això limita el nº de valors a representar.

Exemple: $L = 2 \rightarrow$ 00 → 1r, 01 → 2n, 02, 03, ..., 98 → nº 99, 99 → nº 100 = 10^2

$L = 3$ 000 → nº 1, 001 → nº 2, 002 → nº 3, 003 → nº 4, ..., 998 → nº 999, 999 → nº 4000 = 10^3

Valors diferents representables: Base ^ Longitud màxima.

Exemple pràctic: El DNI espanyol utilitza 8 dígits decimals. A quantes persones distintes pot identificar de forma unívoca?

$10^8 = 100.000.000$

Des de → 00.000.000

Fins a → 99.999.999

- Altres bases, (que en el nostre camp d'estudi són interessants):

  Binària: $S = \{0, 1\}$ $\#S = 2$, "Base 2"

  Octal: $S = \{0, 1, 2, 3, 4, 5, 6, 7\}$ $\#S = 8$ "Base 8"

  Hexadecimal. $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ $\#S = 16$ "Base 16"

Exemples de numeració:

Decimal	Binari	Octal	Hexadecimal
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	01000	10	8
09	01001	11	9
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16

- Per a indicar que treballam amb una longitud donada es poden passar "zeros a l'esquerra".

→ El llenguatge més natural per a naltros és el <u>decimal</u>.

→ El llenguatge amb que realment operen ds ordinadors és el <u>binari</u>.

→ Per a què s'utilitzen ds altres dos?

⇝ A nivell humà, el llenguatge binari és molt "farragós". Un ordinador ens pot voler dir, per exemple,

> Tenc la dada numèrica "..." a la posició de memòria
>
> 11111110 : 10111000

I això és dificil d'entendre.

- El que ens dirà és:

> Tenc la dada numèrica "..." a la posició de memòria
>
> FE : B8

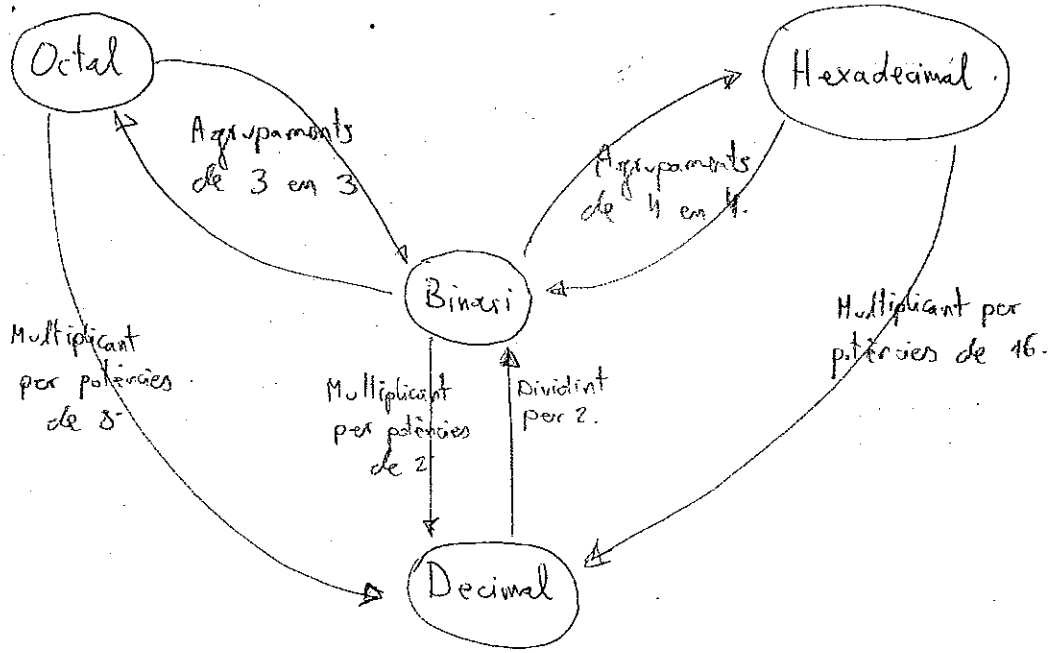
que és més senzill de recordar i manipular. (També de regles, es requereix introduir dades numèriques en aquest format).

Octal no s'utilitza tant, però té el mateix ús.

. Longitud de paraula. Màxim nº representable: L = longitud; màxim = base^L - 1 (nº de números).

Eg: $10^2 = 100$, $2^{10} = 1024$, $16^2 = 256$.

2.2 Conversions d'una base a una altra.



2.2.1 Decimal → Binari

Exemple: Convertir 314 decimal a binari

```
314 | 2
11  157 | 2
14  17  78 | 2
 0   1  18  39 | 2
     0  19  19 | 2
         1   9 | 2
         1   4 | 2
             0   2 | 2
                 0   1
```

⇒ 100111010

→ L'últim dividend és sempre (2) o 3.

9 digits.

2.2.2 Binari a decimal.

Exemple: Convertir 100111010 binari a decimal

$$1 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

$$256 + 32 + 16 + 8 + 2 = 314$$

256 · 53 = 314

Més senzill:

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1
↓			↓	↓	↓		↓	
256	+		32	+ 16	+ 8		+ 2	= 314
1	0	0	1	1	1	0	1	0

2.2.3. Binari → Hexadecimal.

Exemple: Convertir 100111010 binari a hexadecimal

Longitud 4	Dígits Hex
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

Agrupar de 4 en 4:

0001 0011 1010

1 3 A → 13A H

2.2.4 Hexadecimal → Binari

Exemple, convertir 13A Hexadecimal a binari.

Agrupar igualment: 1 3 A

0001 0011 1010 → 100111010.

2.2.5 Hexadecimal, a decimal.

Exemple: convertir 13A Hexadecimal a Decimal

- Es pot fer el pas per binari, i després passar a decimal (punt 2.2.2)
- o be fer la conversió directament:

Potències de 16 →

16^2	16^1	16^0
256	16	1
× 1	× 3	× 10
256 +	48 +	10 = 314

2.2.6. Binari → Octal.

Exemple: Convertir 100111010 binari a octal.

longitud 3	dígit octal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Agrupar de 3 en 3:

100 | 111 | 010

4 7 2 → 472 ϕ

2.2.7. Octal → Binari

Exemple: Convertir 472 octal a binari.

Agrupar igualment: 4 7 2

↓

100 111 010 → 100111010

2.2.8. Octal → Decimal

Exemple: Convertir 472 octal a decimal

- Es pot fer el pas per binari, i després passar a decimal (punt 2.2.2) o bé fer la conversió directament:

Potències de 8 → 8^2 8^1 8^0

↓ ↓ ↓

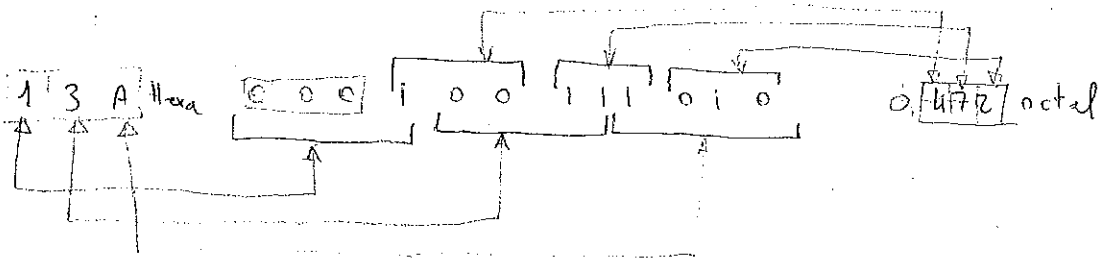
64 8 1

× 4 × 7 × 2

256 + 56 + 2 = 314

2.2.9. Octal → Hexadecimal i viceversa.

Amb el mateix exemple, pas per binari:



3.3 Aritmètica binària.

3.3.1. Operació suma.

✱ Dos operands → 2 resultats !!

Taula de sumes: d'1 sol dígit

a+b	b=0	b=1
a=0	0	1
a=1	1	0 → "m'emporto 1", CARRY

a+b	b=0	b=1
a=0	00	01
a=1	01	10

Forma de representar-ho:

a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

C: és l'AND.

Dos operands d'1 dígit → 1 Resultat (de 2 dígits)

✱ Operacions amb més dígits

	x_4	x_3	x_2	x_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1

$$\begin{array}{r} 3 \rightarrow 0\ 0\ 1\ 1 \\ +\ 4 \rightarrow 0\ 1\ 0\ 0 \\ \hline 7 \leftarrow 0\ 1\ 1\ 1 \end{array} \rightarrow \text{4 dígits}$$

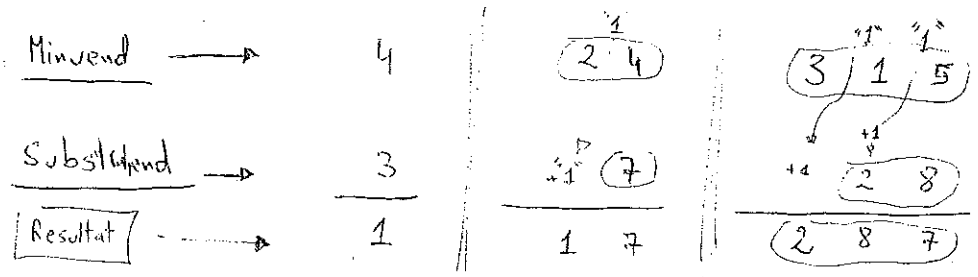
$$\begin{array}{r} 5 \rightarrow 0\ 1\ 0\ 1 \\ +\ 12 \quad\ 1\ 1\ 0\ 0 \\ \hline 17 \leftarrow 1\ 0\ 0\ 0\ 1 \end{array} \rightarrow \text{5 dígits}$$

2^4 2^0

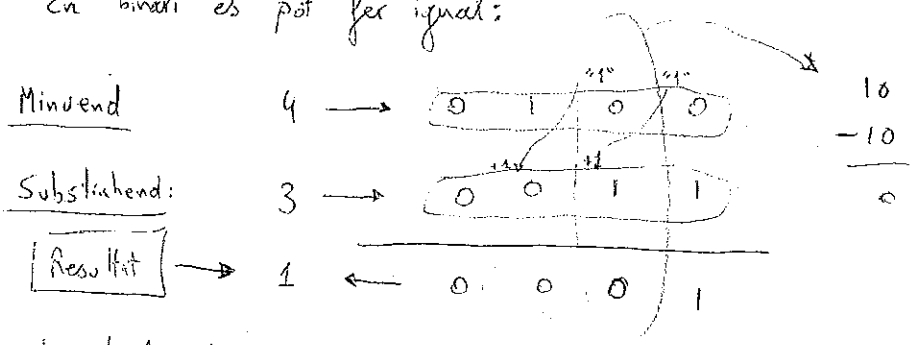
"OverFlow" (SobreFluxe)

2.3.2 Operació resta

- Com es fa la resta decimal? S'utilitza la "unitat prestada", que es suma al dígit de pes següent del substrahend:



En binari es pot fer igual:



La taula d'operació seria:

a	b	Resta	UP
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

Com abans:

Dos operands (1 dígit) x2 → Dos resultats d'1 dígit cada un

Problema:

- Què passa si el substrahend és més gran que el minuend?
→ No hem dit res dels nombres negatius.

⇝ En decimal, el signe és un "senzill" símbol, ... però, hi ha un altre mètode de restar, tant en decimal com en binari, que es basa en el "complement a base".

⇝ En el complement a base, la resta $a-b$ es converteix en l'operació $a + (B^l - b)$, on B és la base i l la longitud dels operands, (del substrahend). Del resultat s'en desprecia el 1r dígit.

Exemples base 10.

$4 - 3 = 1$

Minuend = 4
Substrahend = 3.
Base = 10
longitud = 1

Complement a 10 de 3 = $10^1 - 3 = 7$

4 + 7 = 11, sense el primer dígit = (1).

$24 - 7 = 17$

Minuend = 24
Substrahend = 7
Base = 10
Longitud = 2, (07).

Complement a 10 de 07 = $10^2 - 7 = 100 - 7 = 93$

$$\begin{array}{r} 24 \\ +\ 93 \\ \hline 117 \end{array}$$

→ sense el 1r dígit = (17)

Exemples base 2

Exemple 315 - 28, demanem.

$4 - 3 = 1$

Minuend = 4 → 100
Substrahend = 3 → 011
Base = 2
longitud = 3

Complement a 2 de 3 = $2^3 - 3 = 8 - 3 = (5) \rightarrow 101$

$$\begin{array}{rl} 4 & \text{Binari} \quad 100 \\ +\ 5 & \Rightarrow \quad +\ 101 \\ \hline & \quad 1001 \end{array}$$

→ sense el 1r dígit = 001 = (1)

$24 - 7 = 17$

Minuend = 24 → 11000
Substrahend = 7 → 00111
Base = 2
longitud = 5

Complement a 2 di 7 = $2^5 - 7 = 32 - 7 = 25$

25 → 11001

$$\begin{array}{rcl} 24 & \rightarrow & 24 \\ -\ 7 & & +\ 25 \\ \hline & & \end{array} \rightarrow \begin{array}{r} 11000 \\ +\ 11001 \\ \hline 1\,10001 \end{array}$$

→ Sense el 1r dígit

2^4 2^0 → (17).

Exemple (315 - 28)

Nota: En la pràctica, el complement a 2 d'un numero és:

a) Canviar 1 per 0, 0 per 1
b) Afegir 1.

- Exemple 1: 3, de 011 a 101
- Exemple 2: 7 de 00111 a 11001

- Com es tracten els nombres negatius?

No ho veurem amb molt de detall, però ho comentarem:

→ s'utilitza un bit de signe, i el resultat surt "complementat" i s'ha de descomplementar. El bit de signe es tracta com un dígit normal.

si és negatiu

Exemple:

22 - 37 = -15

Minuend : 22 = 10110
Substrahend : 37 = 100101
Base : 2
longitud 6

Complement a 2 de 37: 011010 + 1 = 011011

Suma, BS

	BS	
22	0	0 1 0 1 1 0
-37	1	0 1 1 0 1 1
0	1	1 0 0 0 1

Complement a 2 de: ... 0 0 1 1 1 0 + 1 = 0 0 1 1 1 1 (per descomplementar, inversió i suma d'1)

Desprecial. menys. → -15

37 - 22 = 15

Minuend = 37 → 1 0 0 1 0 1
Substrahend = 22 → 0 1 0 1 1 0
Base 2
l = 6;

Complement a 2 de 22: 1 0 1 0 0 1 + 1 = 1 0 1 0 1 0

Suma

	BS	
37	0	1 0 0 1 0 1
-22	1	1 0 1 0 1 0
1	0	0 0 1 1 1 1

Donat que el resultat és positiu, no es descomplementa.

Desprecial. més

Observació: Tenim 2 formes de restar.

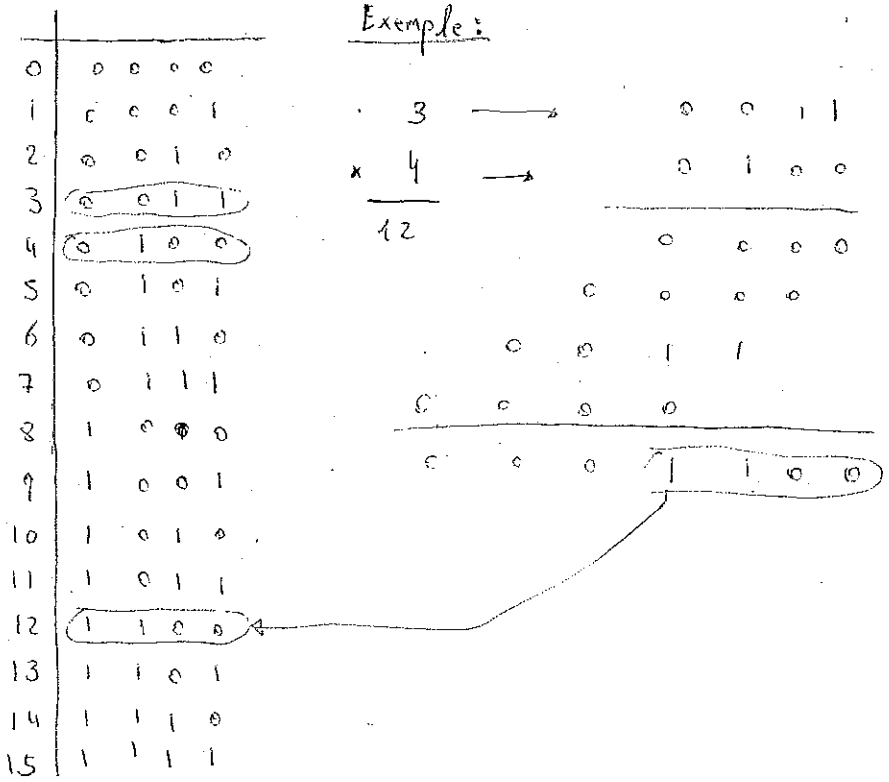
1.) Mètode directe, amb unitats prestades
2.) Mètode indirecte, complementant i executant sumes

2.3.3: Operació Producte.

- Dos dígits en generen un, sense carry.

a	b	a·b
0	0	0
0	1	0
1	0	0
1	1	1

(És l'AND).



Altre mètode: $3 \times 4 = 3+3+3+3$, suma 4 vegades:

3 → 0 0 1 1 → UNA

+3 + 0 0 1 1 → DOS

0 1 1 0

+3 0 0 1 1 → TRES

1 0 0 1

+3 0 0 1 1 → QUATRE.

1 1 0 0 → (12)

Observació: Dos mètodes
1. Mètode de multiplicació directa
2. Mètode de sumes múltiples, (comptant les vegades).

2.3.4. Operació Divisió

12 |4
0 3

~~Quocient / Dividend~~
Dividend / Divisor → Quocient, Residu.
12 4 → 3 0
Dos operands — Dos resultats.

Mètode directe:

Coneguent les taules de multiplicar, "cap a ...", producte parcial, resta, agrupament de dígits del dividend...

212 |25
12 8

→ Es pot reproduir en binari !!

Mètode de restes múltiples:

12 ≥ 4? Sí.

	12	
Restem	4	1 vegada, 8 ≥ 4? Sí!
	8	
Restem	4	2 vegades 4 ≥ 4? Sí!
	4	
Restem	4	3 vegades, fins que resultat parcial < divisor. 0 ≥ 4? No!
	0	

Aleshores: quocient = 3
residu = 0

Observació:

Dos mètodes:

1) Mètode directe
2) Mètode de restes, que complementant, es pot portar a terme efectuant sumes.

Exemple:

12	5
2	2

(12/5 ?)

12
− 5 → 1 vegada → Quocient = (2)
7
− 5 → 2 vegades
2 → (Residu < 5) = (2)

decimal	binari
12	1100
5	0101

→ Complement a 2 de 5: 1011

A lletres: 12 : 5 es Fa

12 → 1100 > 0101 ? (Sí)
− 5 → 1011
7 → 1111 > 0101 ? (Sí)
− 5 → 1011
2 → 0010 > 0101 (No!)

2 vegades: Q = 2

→ Residu = (2)

Un altre exemple:

25	7
4	3

25 → 11001
7 → 00111
C_2 7 → 11001

divisor = 0 0 1 1 1

25 → 1 1 0 0 1

+C2 7 → ~~0 0 0 0 1~~

~~0 0 0 0 0~~ > 0 0 1 1 1 ? → (SI)

1 1 0 0 1

~~0 1 0 1 1~~ > 0 0 1 1 1 ? → (SI)

1 1 0 0 1

~~0 0 1 0 0~~ > 0 0 1 1 1 ? → (NO)

Q = 3 repetes.

R = 0 0 1 0 0 = (4)

Exemple proposat

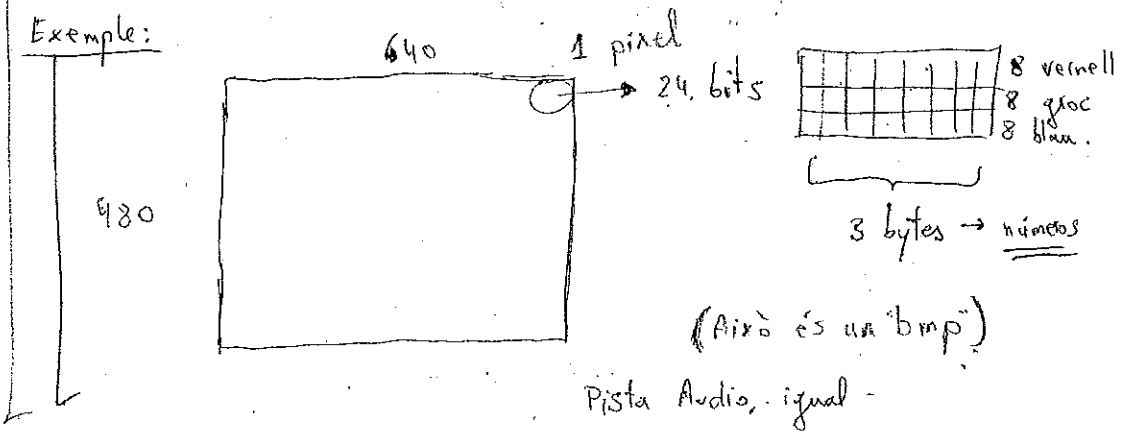
212 | 25
12 | 8

2.4 Codis Binaris

En una computadora captam com a input, emmagatzemant i processam i generam un output de dades tipus:

- numèric
- alfabètic
- alfanumèric estructurat (fitxes/registres) → números + lletres
- Altres: imatge, so, (multimèdia) Es converteixen en seqüències numèriques (digitalització) que després activen o desactiven els "circuits" corresponents.

Exemple:



(Això és un "bmp")

Pista Audio, igual.

2.4.1. Codis numèrics.

Donada una longitud de paraula, (1 byte = 8 bits és la més normal), hem de saber quants numeros diferents podem guardar segons el mètode:

Exemples

- 1 byte → màxim $2^8 = 256$ números → de 0 a 255, de -127 a 127.
- 2 bytes → màxim $2^{16} = 65.536$ números → { de 0 a 65.535 / de -32.768 a 32.767.
- ...

→ La representació dels negatius es pot fer de vàries maneres:

* amb un bit de signe
* de forma pareguda al Complement a 2, de manera que "l'aritmètica tengui sentit".

(Nombres sencers...) → No ho explicam.

Aritmètica distinta en Complement.

* La representació dels **decimals** té un problema: dígits [illegible] de cada costat del punt decimal? Perdem molt d'espai

Massa del sol: 2 000000000000000000000000000000000000,0000000000000000000000000000 grams (39 dígits ... 28 dígits)

Massa de l'electró: 000000000000000000000000000000000000,00000000000000000000000000009 grams

(mantissa → exponent o característica)

Massa del sol: $2.00 \cdot 10^{34} = 0.2 \cdot 10^{35}$

Massa de l'electró: $9.00 \cdot 10^{-28} = 0.9 \cdot 10^{-27}$

Numero $\pi = 3.14 = 3.14 \times 10^0$

$1941 = 1.941 \times 10^3$

$25.2 = 2.52 \times 10^1$

$0,000001 = 1.00 \times 10^{-6}$

El rang a representar depen ara del nº de dígits de mantissa i d'exponent, (dígits significatius)   x.xx E xx

Exemple: Mantissa = 3 → Desde 000 fins a 999
Exponent = 2 → Desde 00 fins a 99
(Amb positius)

$0.00^{00} = 1$ $9.99^{99} = 9.$

$1000 = 1 \cdot 10^3$

$55000 = 5.5 \times 10^4$ (nº es pot.)

S'han de tenir en compte també les magnituds negatives (signe menys del nº) i menors que 1 en valor absolut (signe − a exponent)

Això és la notació de **COMA FLOTANT** Notació Científica.

~~[illegible]~~ Aritmètica distinta: $10^3 \times 10^2 = 10^5$

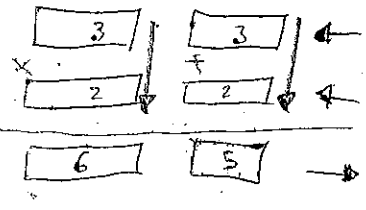
~~els nombres negatius i decimals (N, Z, R, naturals, sencers, reals)~~
~~com s'emmagatzemen i manipulen els nombres en un sistema digital?~~

*Observar que el producte de nºs en coma flotant és, en realitat, producte de mantisses i suma d'exponents:

$3 \cdot 10^3 = 3.000$

$2 \cdot 10^2 = 200$

$3 \cdot 10^3 \times 2 \cdot 10^2 = 6 \cdot 10^5$



3000
200
———
600.000

2.4.1.1. Binari Natural.

* És el binari $\mathbb{N}$, sense signes ni part decimal. Ja sabem com es representa i com s'opera aritmèticament amb aquest tipus de nºs.
* Ja hem comentat que existeix també la forma de representar els nºs binaris dins $\mathbb{Z}$ i dins $\mathbb{R}$, amb signe i amb part decimal respectivament. (Nosaltres els deixarem de banda)

2.4.1.2 Codis BCD

BCD = Binary Codified Decimal = Decimal Codificat en Binari.

* Serveix per a passar un numero decimal a binari. S'utilitza sobretot en els dispositius d'entrada de dades.

Dígits Decimals	Valor BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

En necessitam 10.

- Si $l=3$, $2^3 = 8 < 10$, no serveix
- Si $l=4$, $2^4 = 16 > 10$, serveix i sobren 6 combinacions.

Hi ha també codis BCD en els quals la correspondència no és equivalent al valor natural del nº binari (No en parlam)

Exemples d'utilització:

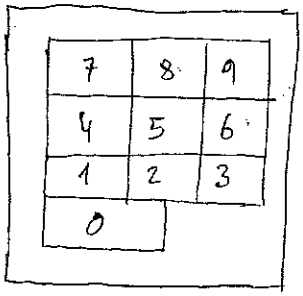
El nº 79 decimal en BCD és:

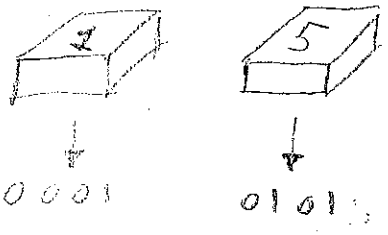
0 1 1 1	1 0 0 1

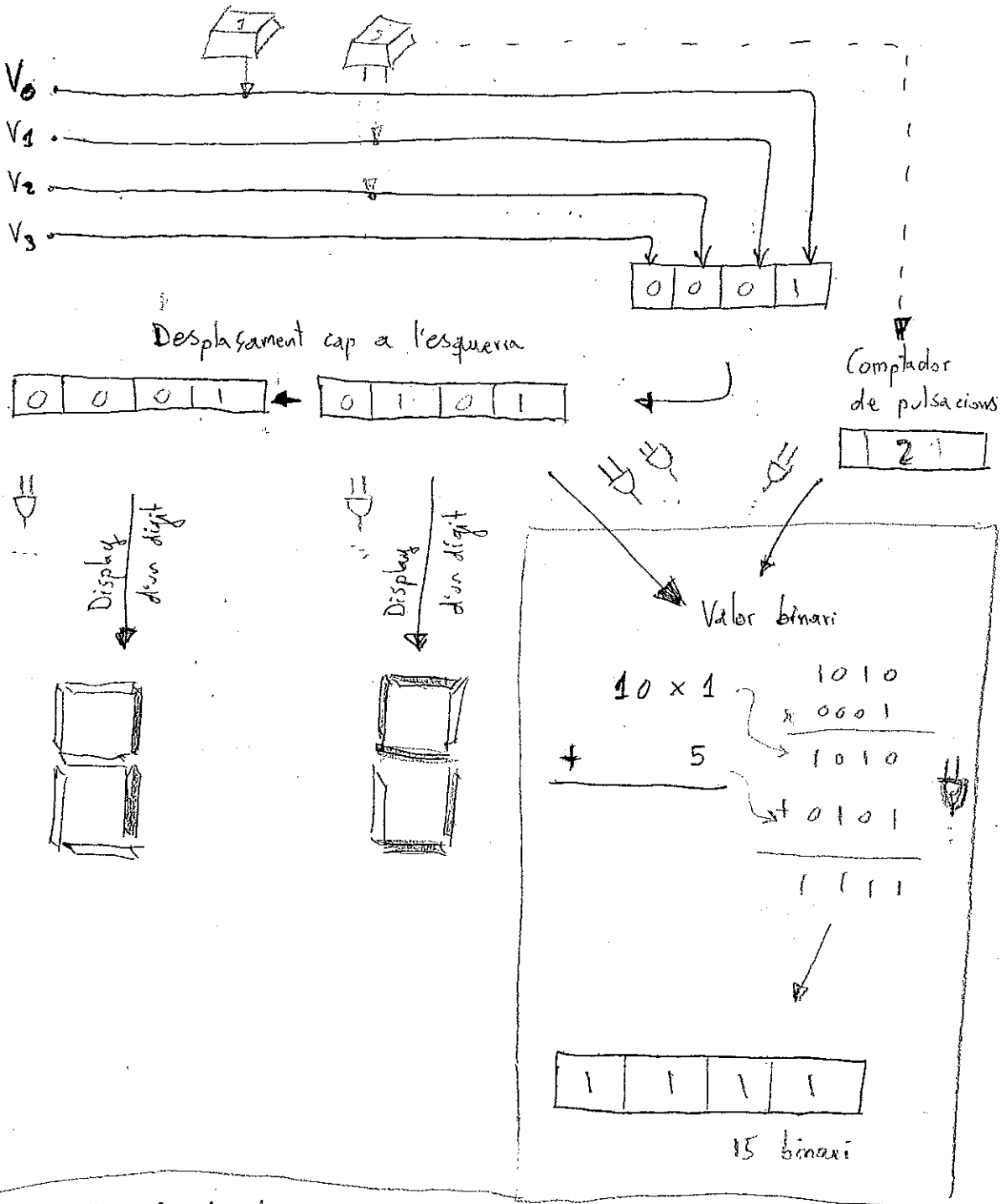
Aritmèticament no és viable, però ja no s'utilitza.

Entrades per teclat:

- Captura de BCDs, conversió a binari i display dels digits decimals en LCD de 7 segments.



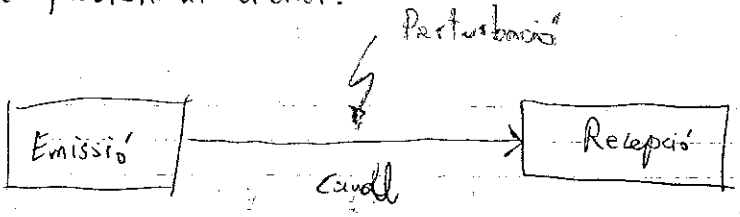




Exercici → fer la lectura del rimini 377

2.4.1.3. Codi Hamming

- En transmissió i emmagatzemament de dades digitals sempre existeix la possibilitat d'error:



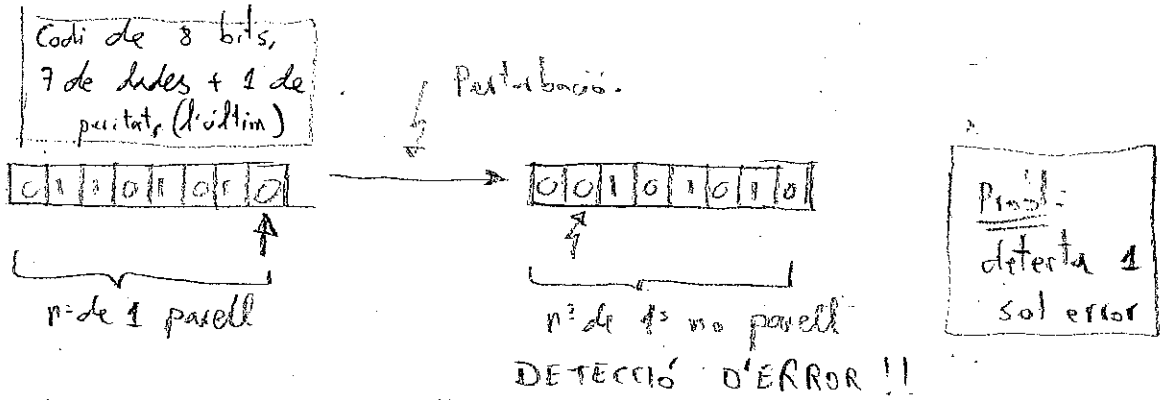
1 1 1 1 (15 binari)

1 0 1 1 (no és el 15 binari!!)

Una forma de prevenir això és introduir alguna informació extra (redundància) que ens indiqui si la informació rebuda és correcte o no.

Exemple: el bit de paritat.

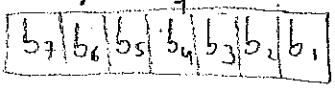
- Consisteix en afegir un "bit extra" que fa que el nº de 1 sigui, per exemple, parell.



Aquest tipus de codis detecten errors, però res més. Codis detectors.

Hi ha codis que van més enllà i que no tan sols detecten l'error, sinó que també el poden corregir. Són els codis autocorrectors. N'hi ha de molts de tipus, el forament algebràic és molt sòlid. Els més senzills són els codis Hamming.

Codi Hamming → codi de 7 bits, 4 que corresponen al BCD natural i 3 de redundància.



b_7, b_6, b_5, b_3 → BCD. b_1, b_2, b_4 → es dedueixen així:

Funció OREX $A \oplus B$.

A	B	A orex B
0	0	0
0	1	1
1	0	1
1	1	0

(Hi. té una implementació amb transistors...)

$b_1 = b_3 \oplus b_5 \oplus b_7$

$b_2 = b_3 \oplus b_6 \oplus b_7$

$b_4 = b_5 \oplus b_6 \oplus b_7$

Aleshores, el codi és:

	b_7	b_6	b_5	b_4	b_3	b_2	b_1
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0

El mètode de detecció d'errors és el següent:

→ L'error s'haurà produit a una posició entre 1 i 7, que en binari natural necessita 3 dígits per ser explicada: entre 000 i 111. Els tres dígits binaris que identifiquen la posició on s'ha produït l'error surten de les següents fórmules:

$C_3 C_2 C_1 \rightarrow xxx$

Si $xxx = 000$, no hi ha error.

Si $xxx = 001$, error a la posició 1

Si $xxx = 111$, error a la posició 7.

$C_1 = b_1 \oplus b_3 \oplus b_5 \oplus b_7$

$C_2 = b_2 \oplus b_3 \oplus b_6 \oplus b_7$

$C_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7$

Exemple de detecció d'error.

7 binari → 1 1 1

en BCD → 0 1 1 1

en codi Hamming 0 1 1 0 1 0 0 → Transmissió → error → 0 1 1 0 0 0 0 (posició 3 errònia)

(No ho sabem!!)

b_7	b_6	b_5	b_4	b_3	b_2	b_1
0	1	1	0	0	0	0

$C_1 = b_1 \oplus b_3 \oplus b_5 \oplus b_7 = 1$

$C_2 = b_2 \oplus b_3 \oplus b_6 \oplus b_7 = 1$

$C_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 0$

→ 0 1 1 → 3 decimal.

Exercici enviar 6 Hamming, amb error a la posició 4.

Observació: corregeix 1 sol error sobre 7. Si hi ha més d'un error no funciona.

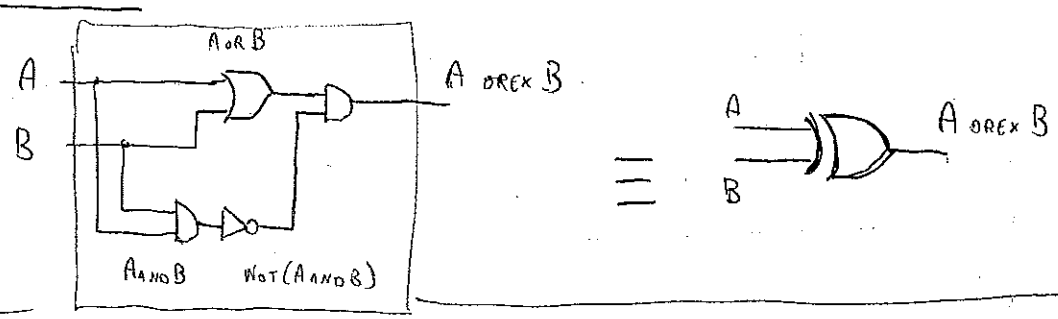
- En codis numèrics també es poden aplicar tècniques de compressió.
- Altres tècniques → xifrat (codis numèrics i lb. alfabètics).

Incís: la porta OREX (OR EXCLUSSIU, EXOR, $\oplus$)

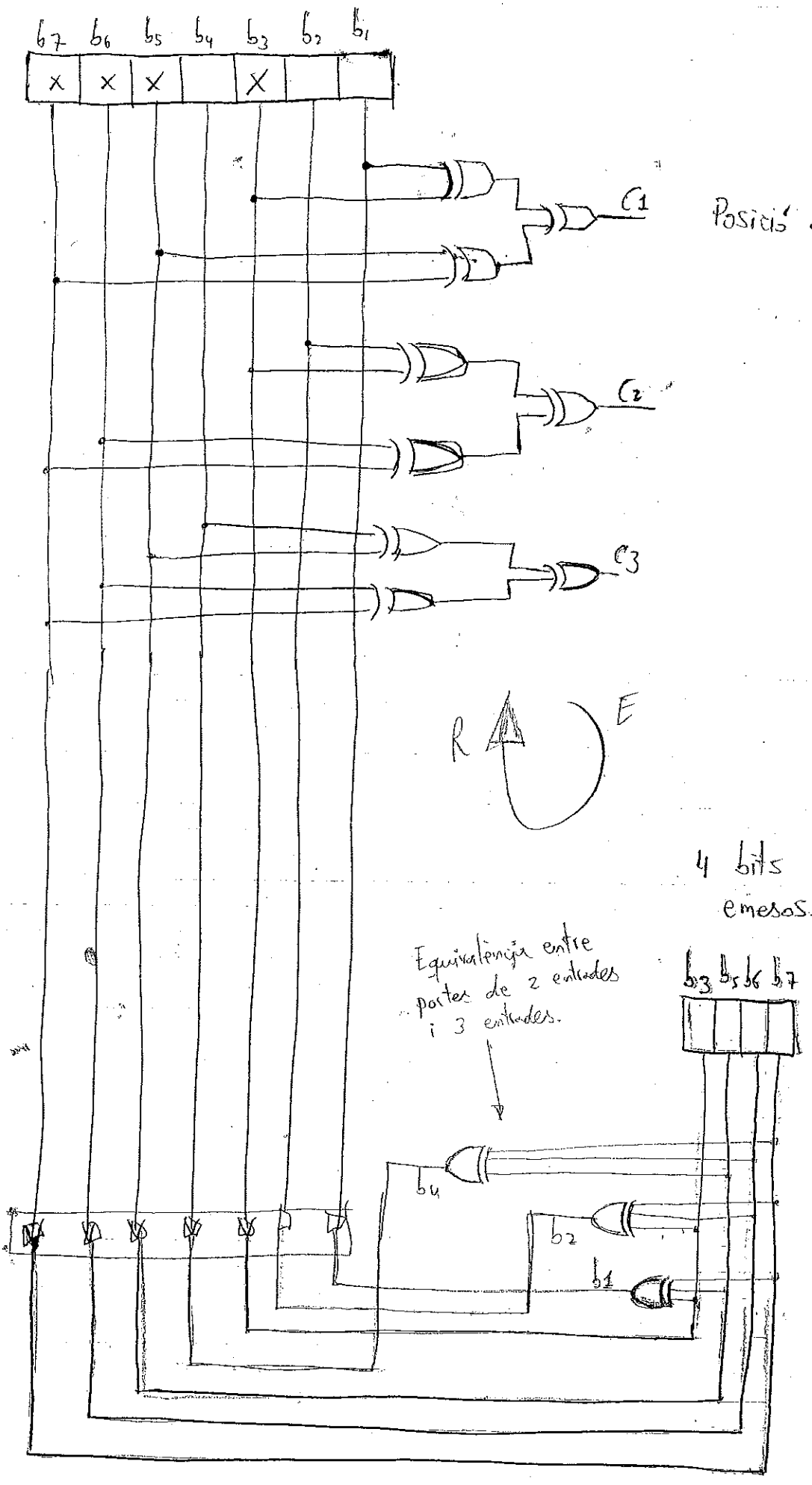
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

A OR B	A NAND B	NOT(A NAND B)	(A OR B) AND NOT(A AND B)
0	0	1	0
1	0	1	1
1	0	1	1
0	1	0	0

Aleshores:



Aleshores:



Posició de l'error.

2.4.2 Codis Alfabètics.

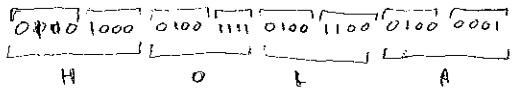
- El que es representa no són números, sino símbols.
  - Lletres a ... z, A ... Z
  - Dígits, no el valor numèric.
  - Caràcters gràfics # $ > ? ...
  - Caràcters de control: activa una funció en un dispositiu: (pantalla, impressora) FF, LF, CR, ... Poden tenir un símbol gràfic associat.

2.4.2.1. Codi ASCII. American Standard Code for Information Interchange.

Codi de 8 bits. → $2^8 = 256$ símbols. ANSI

Exemples de representació: (taula).

HOLA → 48H, 4FH, 4CH, 41H



Caràcters de retorn de carro i nova línia: CR, LF 13, 10 → 0DH on H
En pantalla o impressora

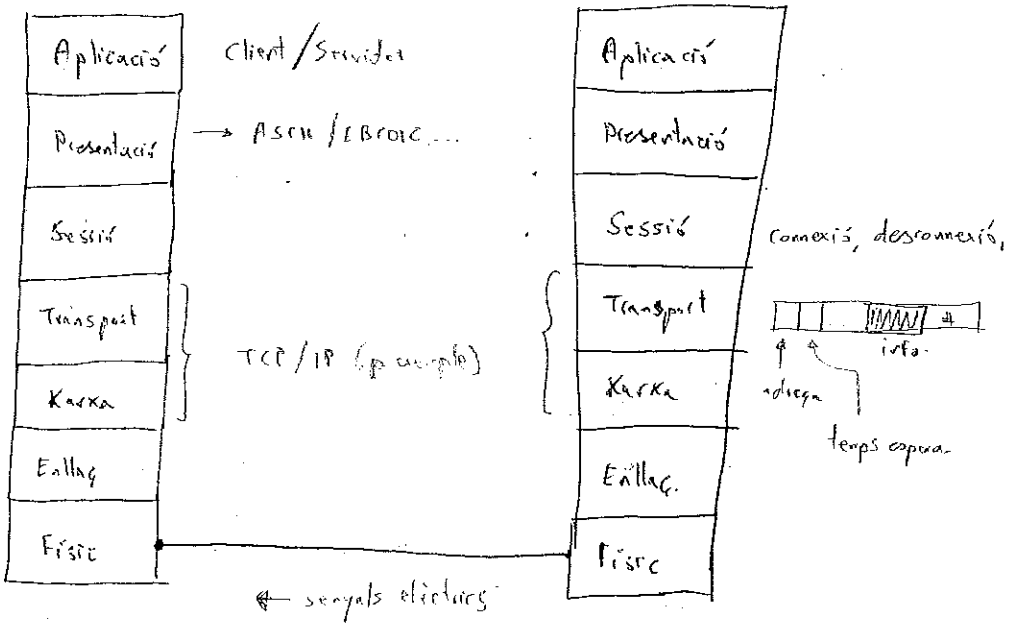
* El codi ASCII és el que utilitzen tots els PCs, per a la representació interna dels caràcters no numèrics.
* Obtenció d'un caràcter ASCII en un PC (via hardware):
  - tecla Alt, sense amollar, codi decimal al teclat numèric.
    Ex: Alt 65 → (A).
* Què és un fitxer ASCII? Fitxer que exclussivament conté caràcters ASCII. En un proc. de textes es pot generar un fitxer amb ASCII + caràcters propis per a representar: gràfics, taules, hipervincles, control d'ubicació de text (alineació, paràgraf), presentació (negreta, grandària ...).
  - Fitxer ASCII → fitxer .txt Importacions.
  - Editar ASCII → Wordpad.

2.4.2.2. Altres.

El més digne de ser comentat és el codi EBCDIC, equivalent al codi ASCII, (altres valors binaris per als mateixos símbols) en quant a Funció.

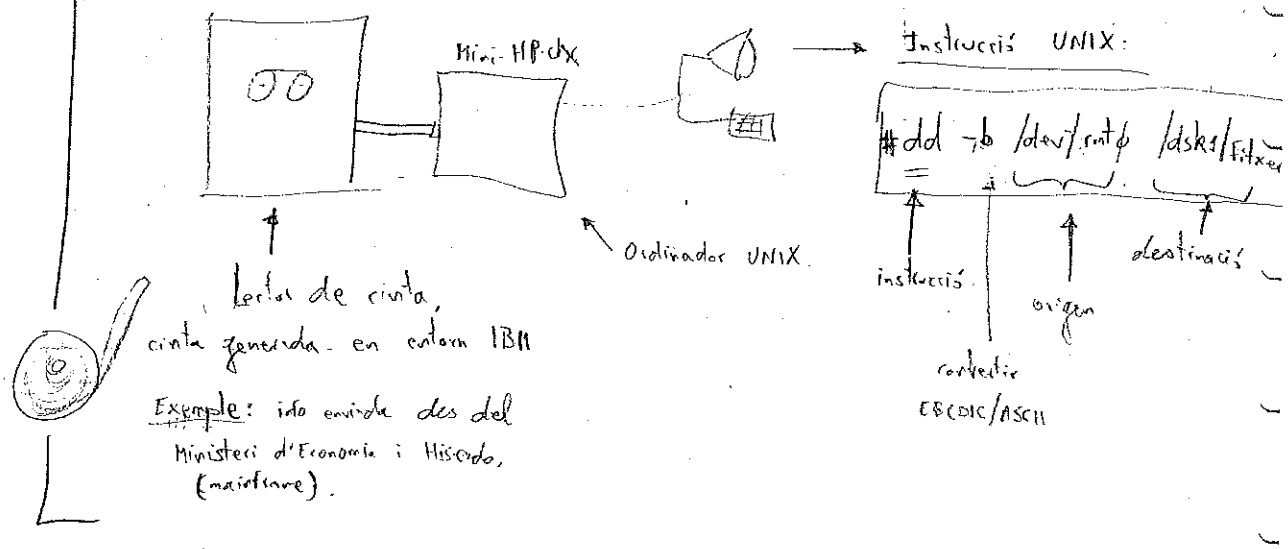
→ L'utilitzen els grans sistemes IBM.

→ Pbbl: com es connecten en xarxa ordinadors que utilitzen codis de representació alfanumèrica distints?
Això es resol en el nivell de Presentació de la pila OSI.



També pot haver-hi utilitats manuals. per a convertir codis.

Exemple:



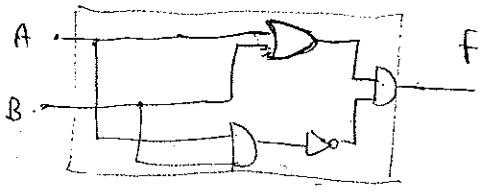
Exemple: info enviada des del Ministeri d'Economia i Hisenda, (mainframe).

2.5 | Lògica Binària. Algebra de Boole

→ Ja hem vist que hi ha una completa equivalència entre la "descripció matemàtica" d'una funció lògica, de l'estil

$$F(A,B) = (A \text{ OR } B) \text{ AND NOT } (A \text{ AND } B)$$

i la seva implementació en forma de circuit digital:



(porta OREX).

→ Ara ens interessa tenir un cert "domini" sobre la descripció matemàtica d'una d'aquestes funcions, de manera que hi puguem "manipular"

Definició: Donat un conjunt $S = \{A, B, C \ldots\}$, on $\forall x \in S$, $x=1 \cdot x=0$, i amb dues operacions internes, que anomenam OR, AND, o $+$, $\cdot$, o $\vee$, $\wedge$, es diu que S és un àlgebra de Boole si compleix aquestes 11 propietats:

Sistema axiomàtic de l'algebra de Boole, sobre S, ($\vee$, $\wedge$) ($+$, $\cdot$)			
(1.) Idempotència	$\forall x \in S$	$x \vee x = x$	$x + x = x$
		$x \wedge x = x$	$x \cdot x = x$
(2.) Commutativitat	$\forall x, y \in S$	$x \vee y = y \vee x$	$x + y = y + x$
		$x \wedge y = y \wedge x$	$x \cdot y = y \cdot x$
(3.) Associativitat	$\forall x, y, z \in S$	$x \vee (y \vee z) = (x \vee y) \vee z$	$x + (y + z) = (x + y) + z$
		$x \wedge (y \wedge z) = (x \wedge y) \wedge z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
(4.) Distributivitat	$\forall x, y, z \in S$	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	
		$x \vee (y \wedge z) = (x \vee y) \wedge (x \wedge z)$	
		$x \cdot (y + z) = xy + xz$	$x + (y \cdot z) = (x + y) \cdot (x + z)$ → comprovar amb taules
(5.) Inclusió (operacions internes)	$\forall x, y \in S$	$x \vee y \in S$	$x + y \in S$
		$x \wedge y \in S$	$x \cdot y \in S$

⑥ Absorció $\forall x,y \in S$ $(x \vee y) \wedge x = x$ $(x+y)\cdot x = x$

$(x \wedge y) \vee x = x$ $(x \cdot y) + x = x$

⑦ Element Universal $\exists\, I$ t.q. $x \vee I = I$ $x + I = I$

$x \wedge I = x$ $x \cdot I = x$

En el cas dels valors 0,1, $I = 1$

⑧ Element nul $\exists\, \phi \in S$ t.q. $x \vee \phi = x$ $x + \phi = x$

$x \wedge \phi = \phi$ $x \cdot \phi = \phi$

En el cas dels valors 0,1 $\phi = 0$

⑨ Complementarietat $\forall x \in S$, $\exists\, \bar{x}$ t.q. $x \vee \bar{x} = I$

$x \wedge \bar{x} = \phi$

⑩ Involució $\forall x \in S$, $\bar{\bar{x}} = x$

⑪ Lleis de De Morgan $\forall x,y \in S$ $\overline{(x \vee y)} = \bar{x} \wedge \bar{y}$ (*) Demostrar per taules.

$\overline{(x \wedge y)} = \bar{x} \vee \bar{y}$

2.6 Simplificació de Funcions lògiques

* Perquè serveix això que acabem de donar?

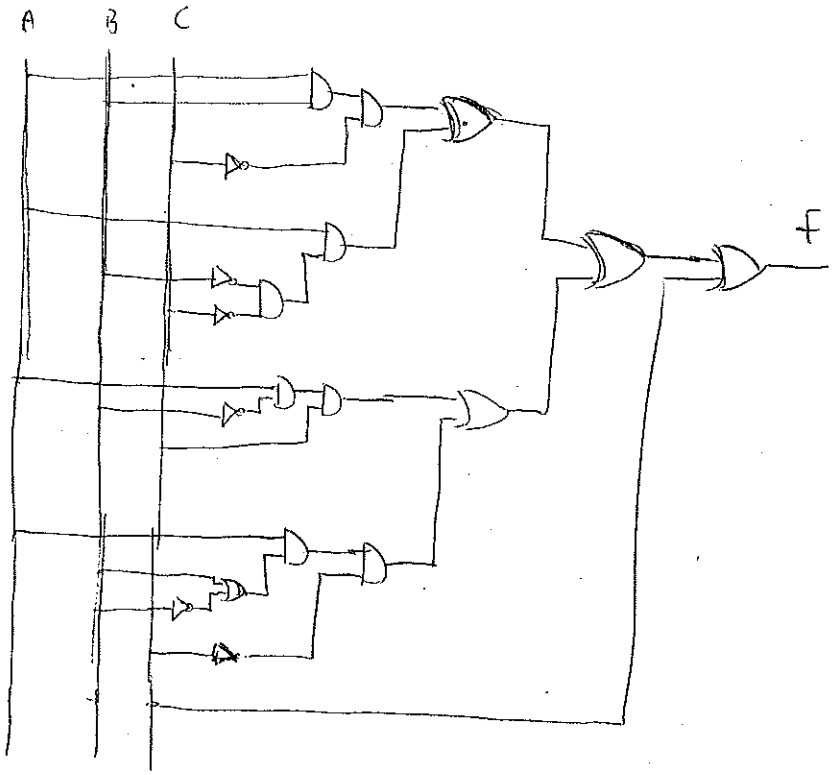
→ Serveix per a simplificar l'expressió matemàtica que defineix una Funció, de manera que quan la implementem utilitzem menys portes lògiques. El mètode que utilitza els axiomes de l'àlgebra de Boole directament és diu mètode algebraic, (2.6.1). Existeix, a més una altra forma de simplificar Funcions, el mètode de Karnaugh, (2.6.2) que s'elabora a partir de l'anterior, i que és més sistemàtic.

2.6.1 Mètode algebràic

Suposem la següent funció:

$$f(A,B,C) = AB\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + A(B+\bar{B})\bar{C} + C$$

Això s'implementaria així:



I els seus resultats serien:

A	B	C	$AB\bar{C}$	$A\bar{B}\bar{C}$	$A\bar{B}C$	$A(B+\bar{B})\bar{C}$	C	F
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	1	1
1	0	0	0	1	0	1	0	1
1	0	1	0	0	1	0	0	1
1	1	0	1	0	0	1	0	1
1	1	1	0	0	0	0	1	1

Una forma més senzilla d'implementar f passa per simplificar la seva expressió matemàtica:

$$F = AB\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}C + A(B+\bar{B})\bar{C} + C$$

$(B+\bar{B}) = 1$

$A\bar{C}$

$A\bar{C}(B+1) + A\bar{B}(C+\bar{C}) + C$

$1 \qquad 1$

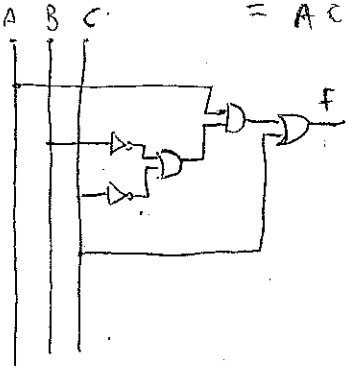
$A\bar{C} + A\bar{B} + C$

$\boxed{A(\bar{B}+\bar{C})+C}$

$AB\bar{C} + A\bar{C} =$

$A\bar{C}\cdot(B+1) =$

$= A\bar{C}$



A	B	C	$(\bar{B}+\bar{C})$	$F = A(\bar{B}+\bar{C})+C$
0	0	0	1	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1

Aquest mètode, però, no assegura que la representació obtinguda sigui la mínima:

$$f = \underbrace{AB\bar{C} + A\bar{B}\bar{C}}_{A\bar{C}(B+\bar{B})} + A\bar{B}C + \underbrace{A(B+\bar{B})}_{1}\bar{C} + C$$

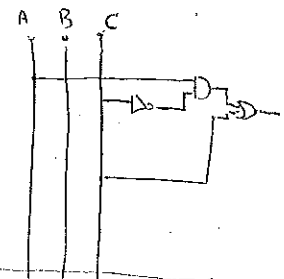
$A\bar{C}(B+\bar{B}) + A\bar{B}C + A\bar{C} + C$

1

$\underbrace{A\bar{C} + A\bar{C}}_{A\bar{C}} + \underbrace{A\bar{B}C + C}_{C(A\bar{B}+1)}$

1

$\boxed{A\bar{C}+C}$



A	B	C	$f = A\bar{C}+C$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

No falta!!

B no fa falta,

Equivalent = donat un nº de 6 dígits, diu si és <100

2.6.2 Mètode de Karnaugh

Es un mètode gràfic. Es basa en la següent propietat:
donada una funció digital tabulada,

A	B	C	F = $A\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + A(B+\bar{B})\bar{C} + C$ ①
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Sempre la podem expressar com a suma dels termes en què la funció val 1, multiplicant les variables:
(No demostrem perquè)

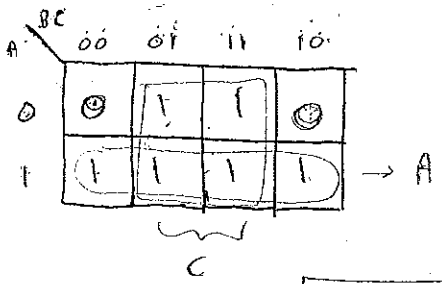
$f = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$ ②

Suma de (MINTERMS)

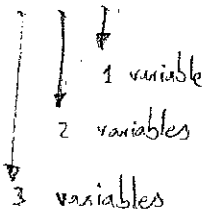
Es pot demostrar, per àlgebra, que sempre es pot passar de ① a ②. (No ho feim.)

② Es pot implementar com a circuit, però no és el la suma de que volem

Una vegada tenim els mintermes, en el cas d'una funció de 3 variables, es fa el següent gràfic:



Grups de 1, 2, 4 cas contigus.



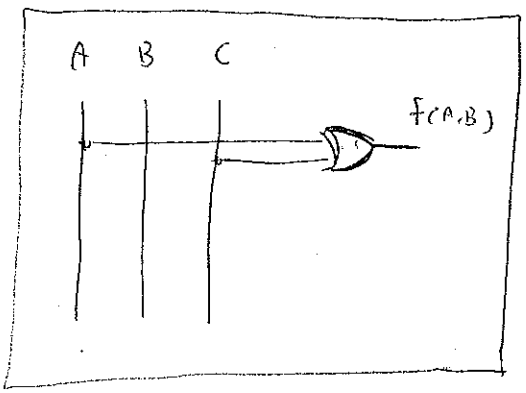
Aleshores, $f(A,B,C) = A + C$

Comprovació:

Taula

A	B	C	A+C
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Circuit



Quan A o C té 5 V, f(A,B,C) té 5 V, i es compleix la funció ①

Un altre exemple amb funció de 3 variables:

⊛ Donat un nº entre 0 i 7, representat per 3 dígits binaris en binari natural, implementar un circuit que ens digui si el nº és 0, 2, 5 o 6 (1 si ho és, 0 si no ho és)

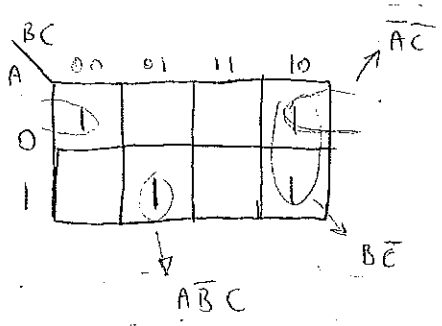
TAULA

	A	B	C	F(A,B,C)
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

PAS A FUNCIÓ EN FORMA DE SUMA DE MINTERMS.

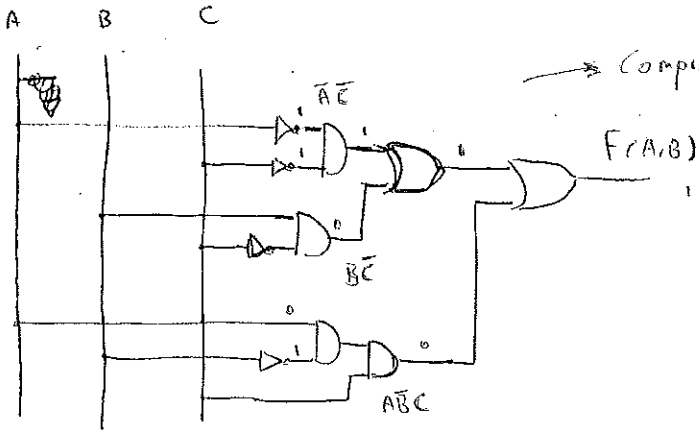
$$F(A,B,C) = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + AB\overline{C}$$

GRÀFIC



$$F(A,B,C) = \overline{A}\overline{C} + A\overline{B}C + B\overline{C}$$ Taula...

Circuit.



→ Comprovació de casos.

SIMULADOR (Workbench).

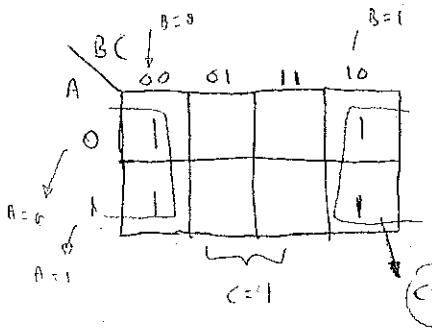
Un altre exemple amb 1 funció de 3 variables:

① Donat un número entre ϕ i 7, representat per 3 dígits binaris, implementar un circuit que ens digui si el número és parell (1 si ho és, ϕ si no ho és).

	A	B	C	F(A,B,C)
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

(Selectivitat !!!)

- En realitat l'expressió algebràica de la funció no ens ha falta per al gràfic del mètode de Karnaugh.
- Podem passar directament al gràfic

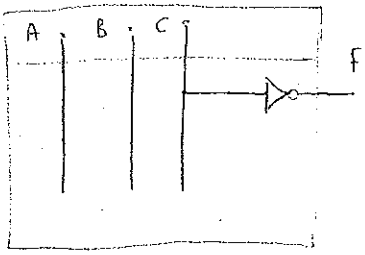


$\vec{C}$

$F(A,B,C) = \vec{C}$

Taula...

Circuit



$F(A,B,C) = \begin{cases} 1 \text{ si } ABC \text{ parell} \\ \phi \text{ si } ABC \text{ imparell.} \end{cases}$

Es veu en la pròpia taula !!

El mètode de Karnaugh ens garantitza, a més a més, que la simplificació del circuit és la màxima possible.

Exercici: // → divisible per (3)
Exemple:

Amb 4 variables, la ~~taula~~ gràfica canvia:

* Donat un nº entre 0 i 15, representat per 4 dígits binaris en binari natural, implementar un circuit que ens digui si el nº és divisible per 4

	A	B	C	D	divisible per 4
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	0	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

$F(A,B,C) = \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} +$ ~~$\bar{A}\bar{B}\bar{C}\bar{D}$~~

Això es podria implementar en forma de circuit, però 1r anem a simplificar.

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	0	0	0
10	1	0	0	0

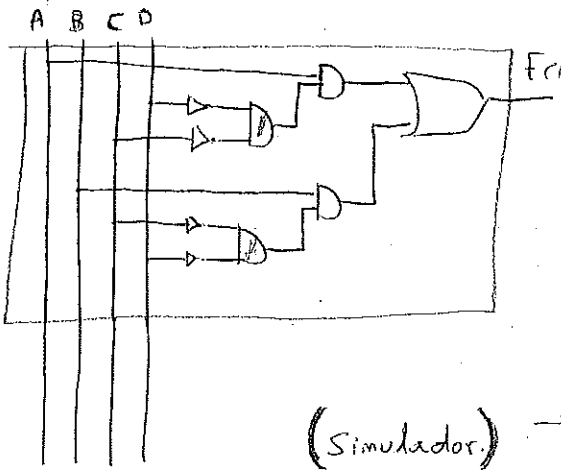
Grups de 8 → 1 var.
4 → 2 var
2 → 3 var
1 → 4 var.

B=1,0 → $A\bar{C}\bar{D}$ +

A=0,1 → $B\bar{C}\bar{D} = \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D}$

$F(A,B,C,D) = A\bar{C}\bar{D} + B\bar{C}\bar{D}$ Taula...

Circuit



$F(A,B,C,D) = \begin{cases} 1 & \text{si div. per 4.} \\ \phi & \text{si no div. per 4.} \end{cases}$

pas de Funció a taula.
pas de circuit a Funció automàtic.
(Simulador) → pas de Funció a circuit automàtic.
MINTERMS.

- Hi ha mètodes gràfics per funcions de 2, 3, 4 i 5 variables. Hem vist 3 i 4.
- Hi ha altres mètodes, per funcions de més de 5 variables.

Es pot donar el cas de què algun valor no estigui definit, i dona igual si l'incloim dins els grups de "1"s. Aquests valors se'ls anomena Termes d'Indiferència.

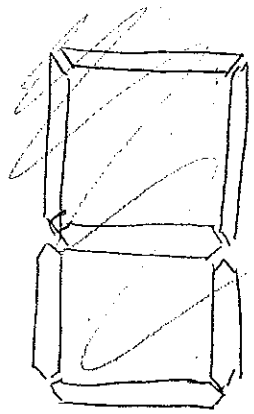
Anam a veure'n uns quants exemples de la següent manera:

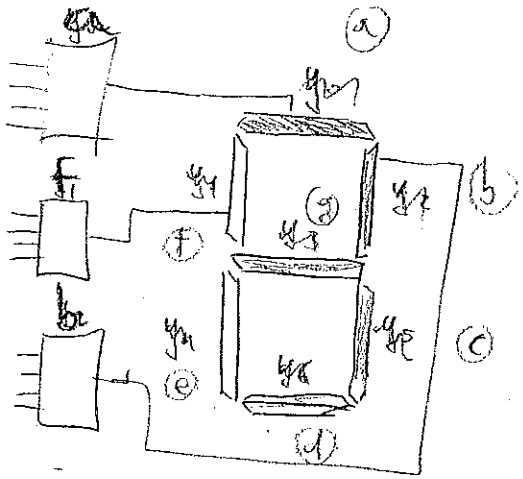
2.6.3 Exemples de display LCD 7 segments.

	A	B	C	D	a	F	b	g	e	c	d
0	0	0	0	0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	1	0	0	1	0
2	0	0	1	0	1	0	1	1	1	0	1
3	0	0	1	1	1	0	1	1	0	1	1
4	0	1	0	0	0	1	1	1	0	1	0
5	0	1	0	1	1	1	0	1	0	1	1
6	0	1	1	0	1	1	0	1	1	1	1
7	0	1	1	1	1	0	1	0	0	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	X	X	X	X	X	X	X
	1	0	1	1	X	X	X	X	X	X	X
	1	1	0	0	X	X	X	X	X	X	X
	1	1	0	1	X	X	X	X	X	X	X
	1	1	1	0	X	X	X	X	X	X	X
	1	1	1	1	X	X	X	X	X	X	X

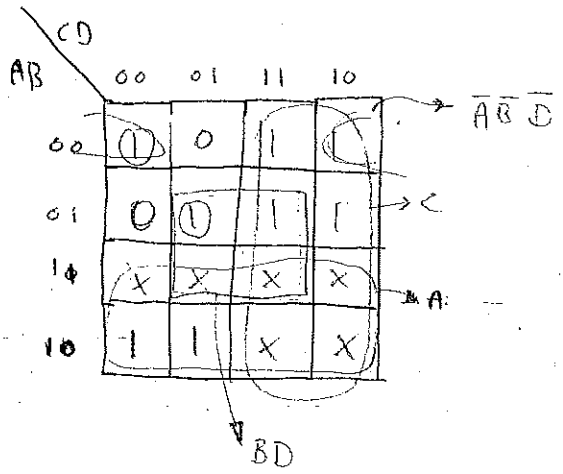
VALORS BCD

← Valors no definits





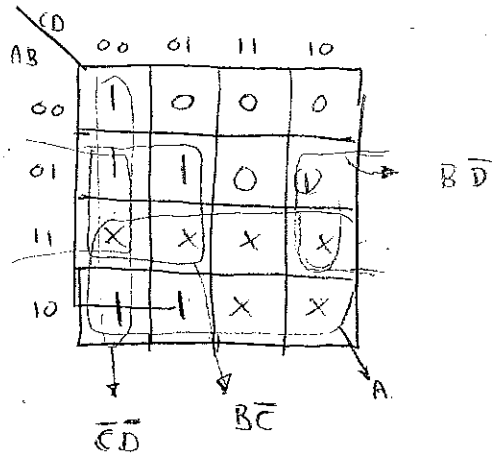
Funció $y_0 = f(A,B,C,D)$. (a)



$$F(A,B,C,D) = \underbrace{\bar{A}\bar{B}\bar{D}}_{x_3 x_2 x_0} + \underbrace{BD}_{x_2 x_0} + \underbrace{A}_{x_3} + \underbrace{C}_{x_1}$$

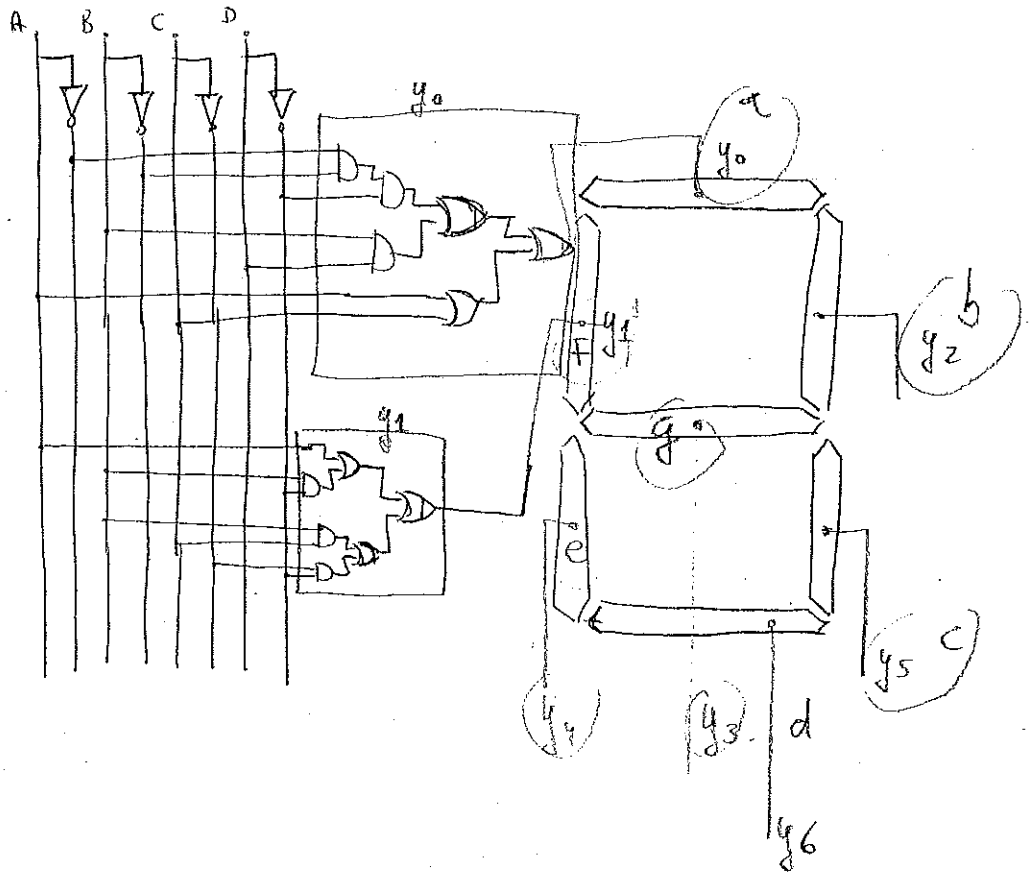
$\bar{B}\bar{D}$ (4 extrems)

Funció $y_1 = f(A,B,C,D)$ (f)



$$F(A,B,C,D) = \underbrace{A}_{x_3} + \underbrace{B\bar{D}}_{x_2 x_0} + \underbrace{B\bar{C}}_{x_2 x_1} + \underbrace{\bar{C}\bar{D}}_{x_1 x_0}$$

El que anam construint és:



y_2, y_3, y_4, y_5, y_6 Exercicis, + circuit complet.

Recordem el cas de: despertador, calculadora.