M1: SISTEMES INFORMÀTICSlínea horizontal  
FRANCESC SASAL

francesc.sasal@iesjoandaustria.org

<https://meet.google.com/iij-ehva-ydt>

Contraseña del moodle: M1DAM1A

**Estructuració del curs**

| UF 1. Instal·lació, configuració i explotació del sistema informàtic (49h) | 22/09-21/12 |
| --- | --- |
| UF 2. Gestió de la informació i de recursos en una xarxa (69h) | 08/01-20/05 |
| UF 3. Implantació de programa específic (14h) | 25/05-15/06 |

**Avaluació**

UF 1 30% x UF 2 55% x UF 3 15%

1. Pràctiques (Mitjana ponderada igual o superior a un 5)
2. Exàmens (Avisar mínim amb 5 dies d’antelació si no es pot assistir)

UF1. Instal·lació, configuració i explotació del sistema informàtic

header line

# Anàlisi i instal·lació de sistemes informàtics 60%

## 1.1 Sistemes numèric i portes lògiques

## 1.2 Estructura i components físics de sistema informàtic

## 1.3 Sistemes operatius

## 1.4 Xarxes

# Configuració i explotació de sistemes informàtics 40%

## 2.1 Administració i configuració de sistemes operatius propietaris i lliures

## 2.2 Gestionar usuaris i recursos compartits

## 2.3 Seguretat de comptes d’usuari i contrasenyes

## 2.4 Configuració de protocol TCP/IP Máscares de subxarxa. DNS

### 

## Informació, informàtica i sistemes informàtics

**Informàtica:** tractament automàtic i racional de la informació.

Però què és la informació?

* **Dades:**  Per si sol no significa res.
* **Informació:** Quan una dada li associem un significat.
* **Coneixement:** Quan s’interpreta. El conjunt de coneixements es saber.

## La informació i la seva mesura: bit i byte

La informació:

* Son dades.
* Representació de fets, objects, valors, idees, etc que permet la comunicació entre persones i l’adquisició del coneixement.
* Resultat de la manipulació de les dades amb la finalitat de produir un coneixement.

### Es pot mesurar la informació?

El **bit** (binary digit) es la mínima informació que podem comunicar (Si, No). És la unitat base de mesura de la informació. (Cert/ Fals, 0 i 1) anomenats **bits.**

El **byte** o octet son 8 bits.

## 

## 

## 

## 

## Múltiples del byte

L’ordinador funciona en binari, llavors 210=1024.

En 1998 la IEC (International Electrotechnical Commission) va desenvolupar un estàndard on es defineixen les unitats.

| kilobyte | kB | 103 bytes | kibibyte | KiB | 210 bytes |
| --- | --- | --- | --- | --- | --- |
| megabyte | MB | 106 bytes | mebibyte | MiB | 220 bytes |
| gigabyte | GB | 109 bytes | gibibyte | GiB | 230 bytes |
| terabytes | TB | 1012 bytes | tebibyte | TiB | 240 bytes |
| petabytes | PT | 1015 bytes | pebibyte | PiB | 250 bytes |

Al començaments es parlava de kibibytes o memibyte. Quan es va començar a vendre memoria, es va començar a dir megabytes.

*B: es bytes - b: son bits*

**Calcula: 640 MB i 640 MiB**

640 MB: 640.000.000

640 MiB: 671.088.640

## Grans quantitats d’informació

* Al 2017: el Laboratori de Física de Partícules (CERN) va superar els 200 petabytes.
* Google: 10 o 15 Hexabytes

## Codificació

Permet convertir la informació d’origen en dades codificades, per emmagatzemar o enviar.

Els ordinadors codifiquen de manera **binària**.

## Processament de la informació

Es tracta de dades d’entrada (informació) -> es **processen** -> per produir unes dades de sortida.

El tractament automàtic de la informació: **INFOR**mation auto**MATIQUE**, a l’anglès es coneix com **computer science.**

## Sistema informàtic

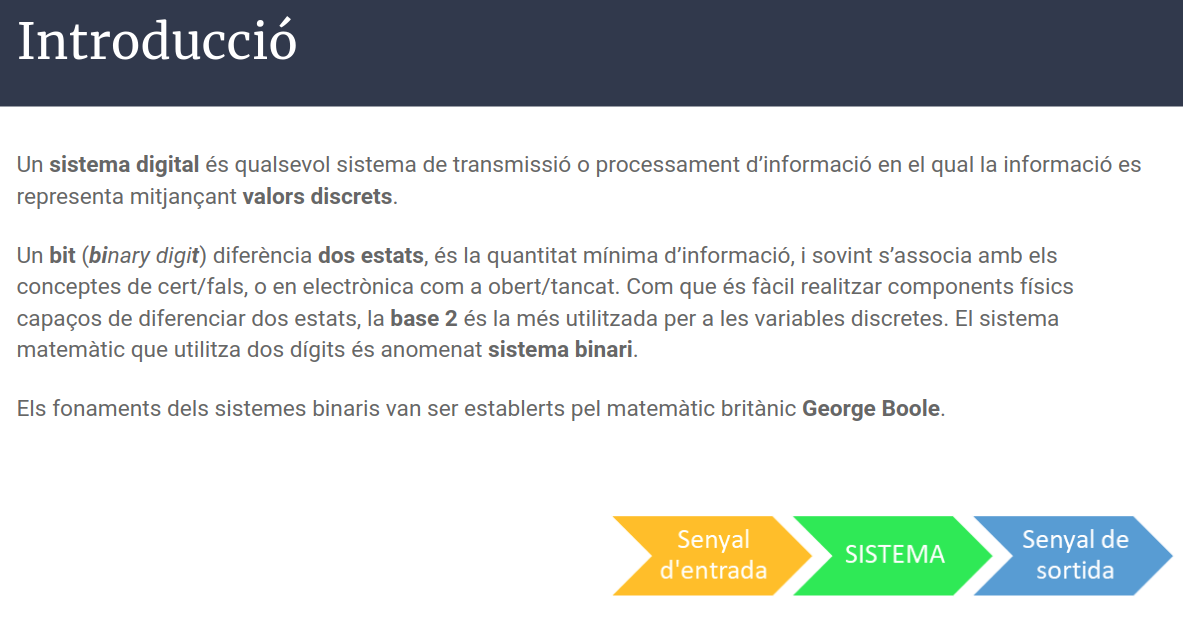
Un sistema informàtic està format per diferents coses.

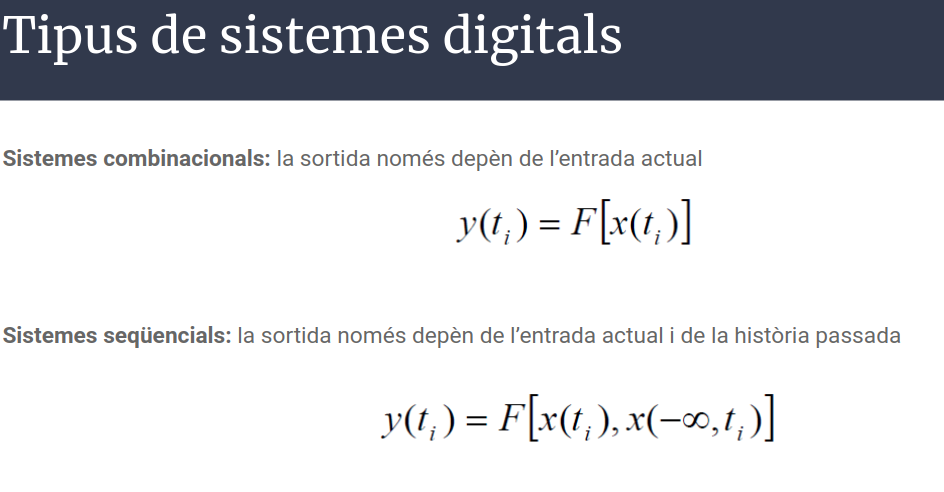
És un sistema format per **persones, dades, activitats,** en definitiva, el conjunt de recursos que **processen la informació** d’una organització.

* Maquinari: tot el que sigui físic.
* Programari: Software. Tots els programes i la seva documentació.
  + **Bàsic**: generalment el que està incorporat al sistema operatiu.
  + **Aplicació**: més específic, a una tasca concreta.
* Recursos Humans
  + **Usuari:** el que el fa servir.
  + **Personal informàtic:** 
    - Direcció: dirigeix a la resta de persones.
    - Anàlisis: busquen els problemes i solucions.
    - Programació: Traduixo del llenguatge.
    - Explotació: executen els programes, la part de sistemes. Tener el programari a punt.
    - Operadors: s’encarreguen del funcionament.

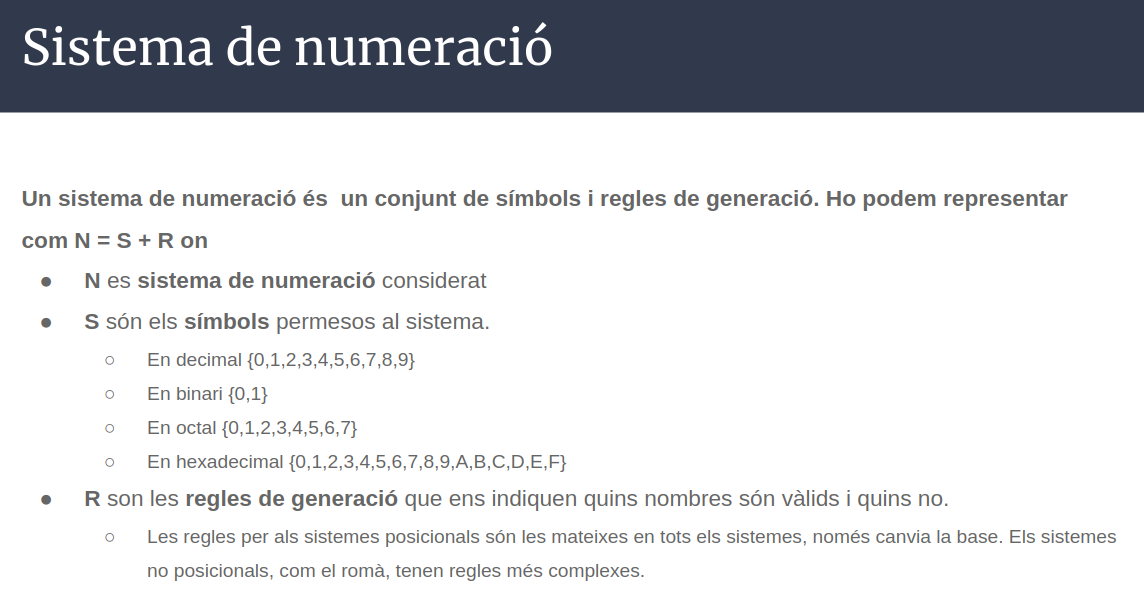
**(Ens ha de comentar el tema de la bibliografia)**

# Introducció

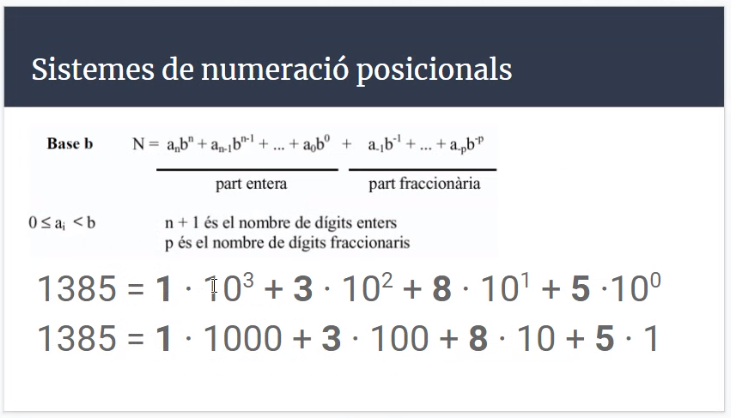
****

****

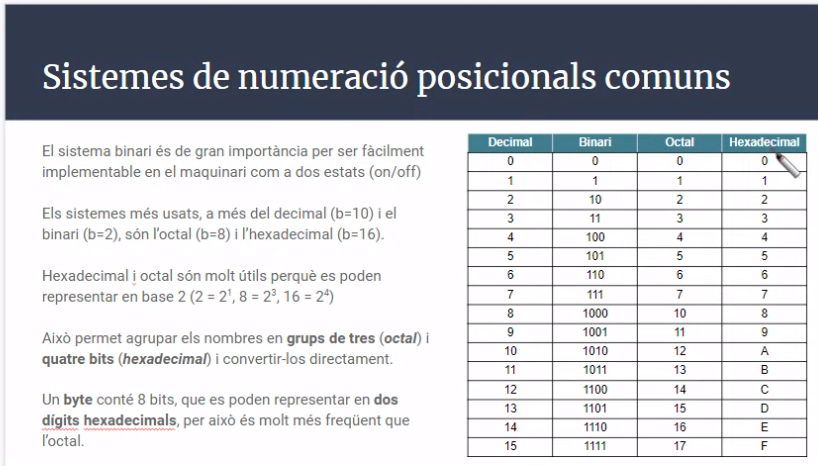
## Sistemes de númeració

****

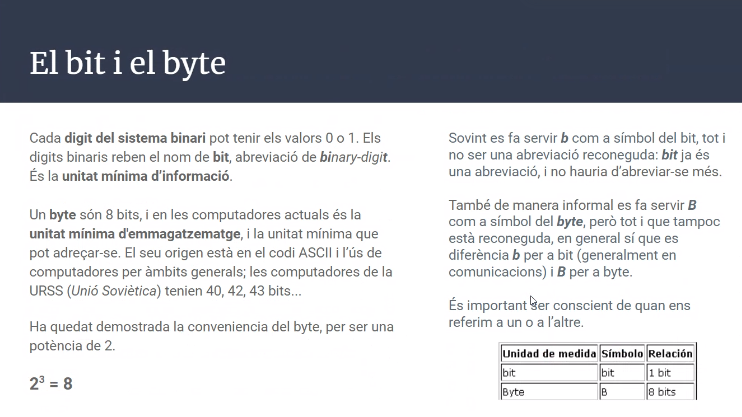
## Sistemes de numeració posicionals

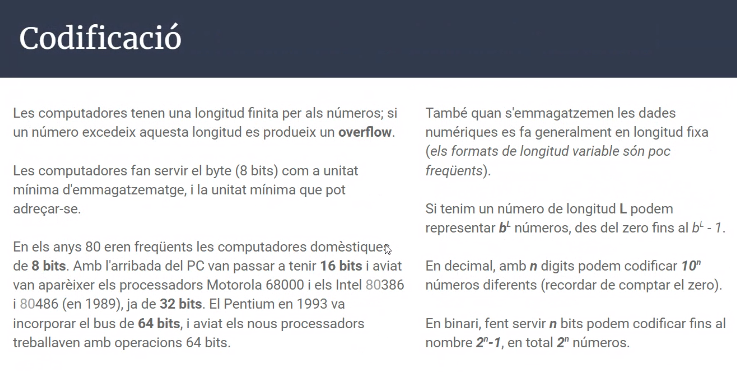
****

S’ha de cambiar la base. 2 en binari, 8 en octal,...   
La part fraccionària funciona igual.



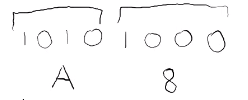
Amb aquesta taula es molt facil pasar de hexdecimal a binari, els més utilitzats.





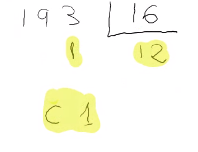
La mida de les computadores, van començar amb 8 bits, amb el PC, 16 bits i així creixent

Si tinc la longitud L en digits: son la base^Longitud → en un 1. 1000 del → 0 fins el 999



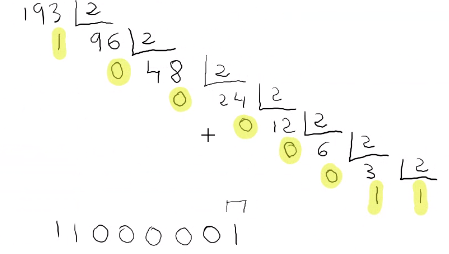
Això son 8 bits: 1 byte. (quatre i quatre). Cada 4 bits, representa un hexadecimal.

Si vull convertir 193 a hexadecimal:

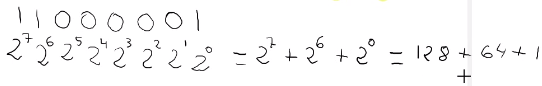
 → Es 12 1 → És a dir, C 1

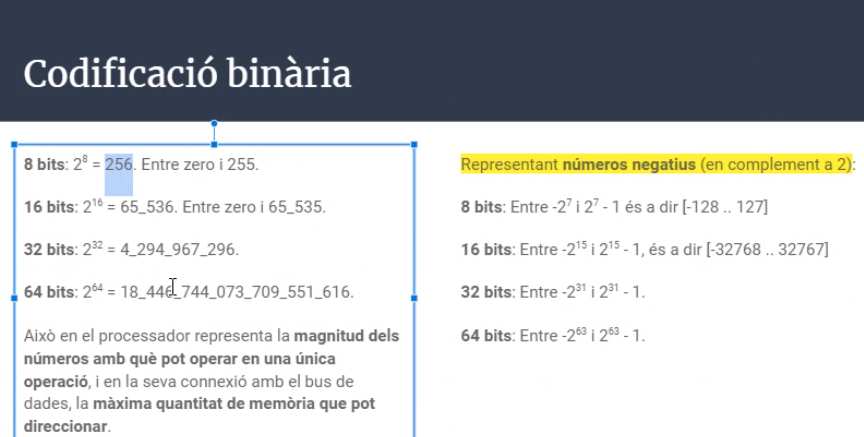
#### Com convertir a binari?

Tinc el número 193 i el vull convertir en binari. (Vas dividint entre 2). El número en binari es el residu. (De dreta a esquerra): **11000001**  *Truc: S****i un número es imparell acabará sempre en 1, i en parell en 0***



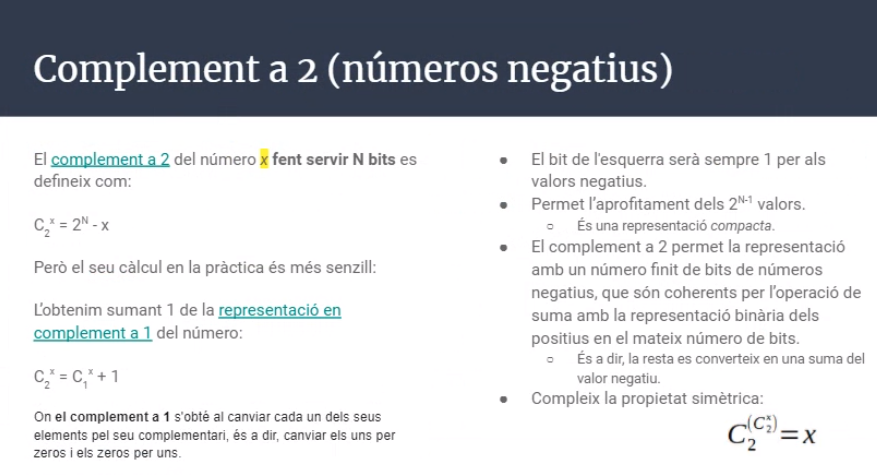
Per fer-ho al revès: (Agafes només els 1)

=193



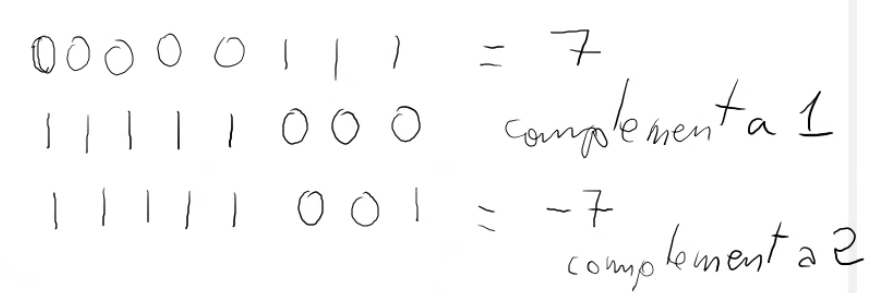
Una única instrucció pot fer les operacions amb aquesta quantitat de bits.  
Si tenim números negatius, en cabran la meitat. De -128 a 127.

## Complement a 2 (números negatius)

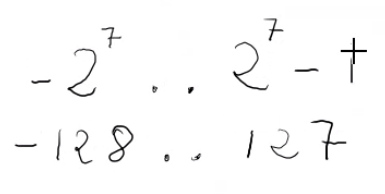


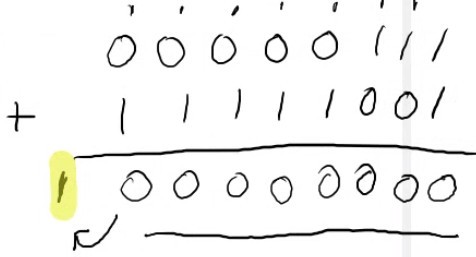
El complement a 2, es canviar de signe.

Cambiar els 0 per 1s. I afegir un 1.

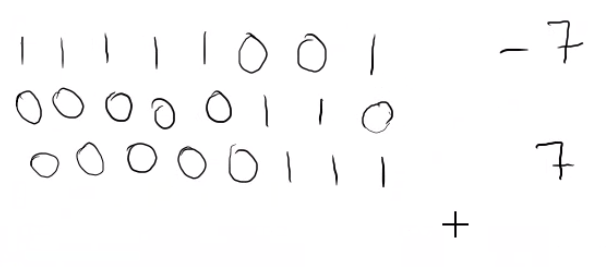


Si el 1(de l’esquerra del tot) és 1: Es negatiu.

  
Y es pot sumar. (1 y 1, ens emporten 1) Ignorem el 1 final.

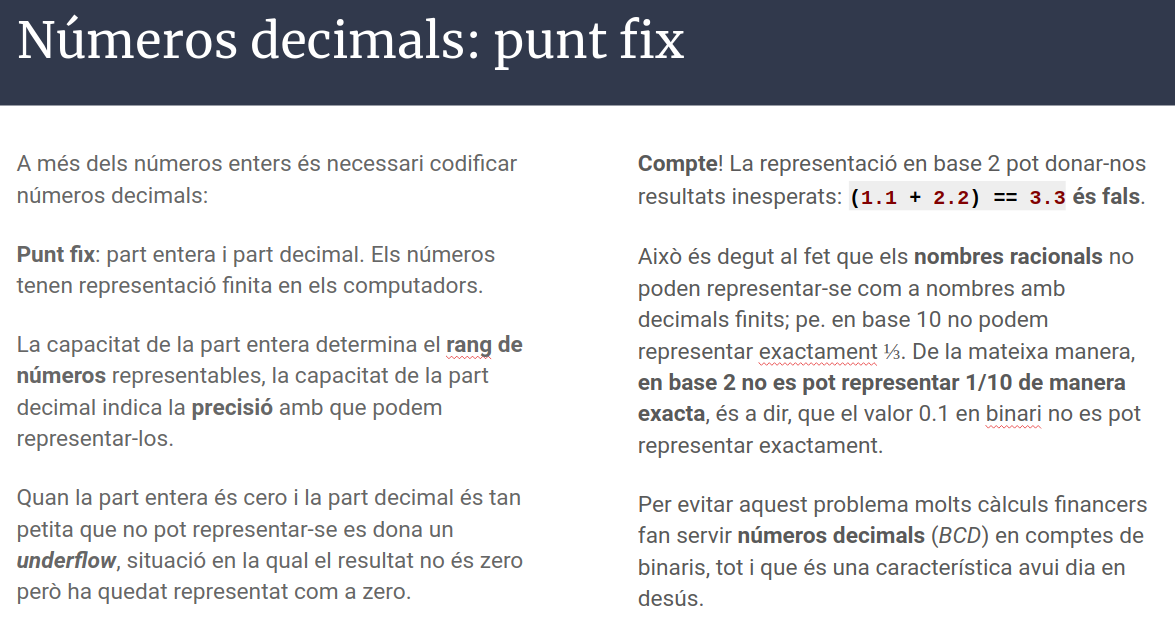


Complement a 2,



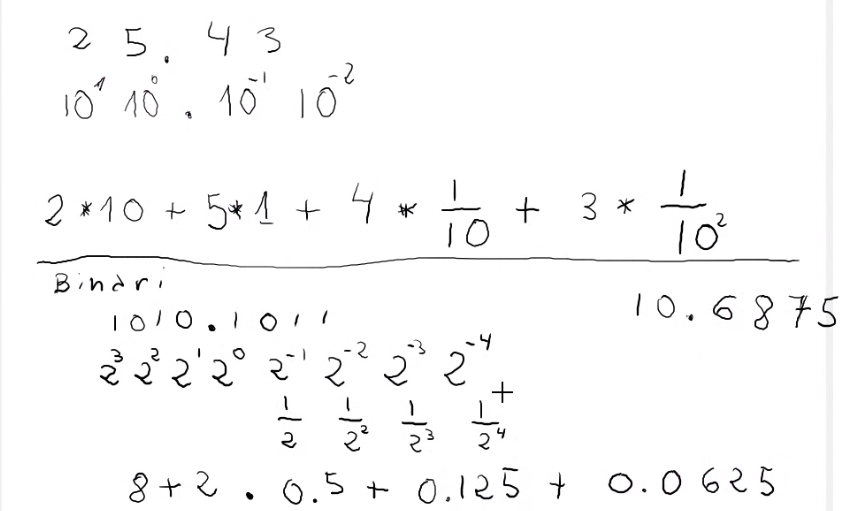
**El primer bit ens diu que es negatiu.**

## Números decimals: punt fix





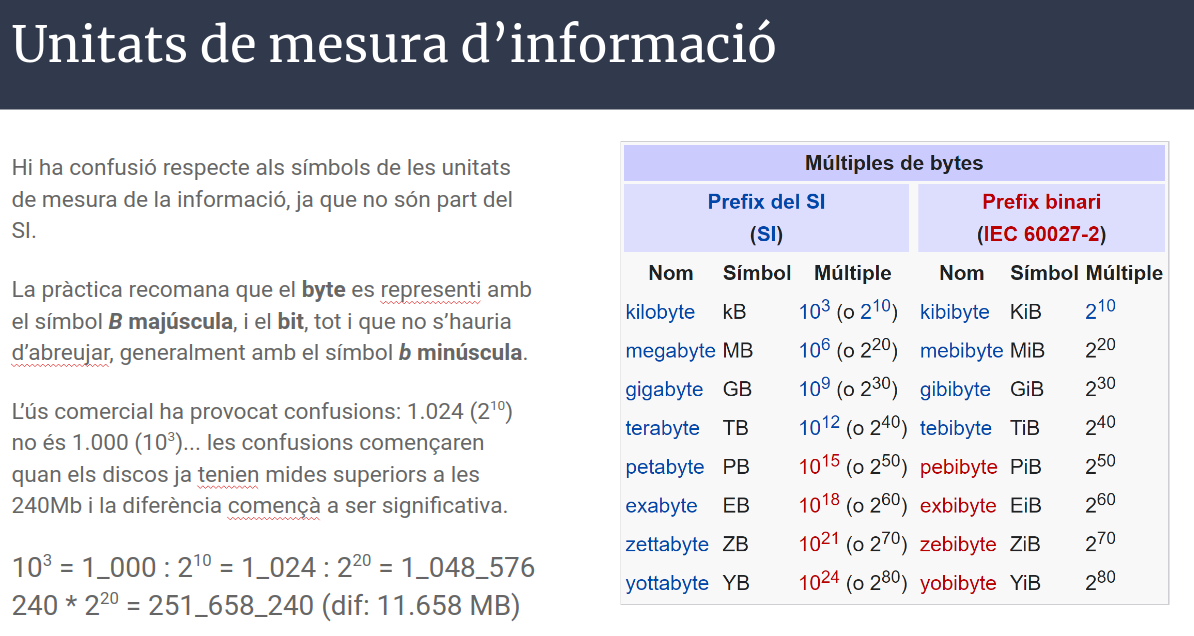
← Això sería la manera de fer de decimal a binari punt fix. pero dona errors.



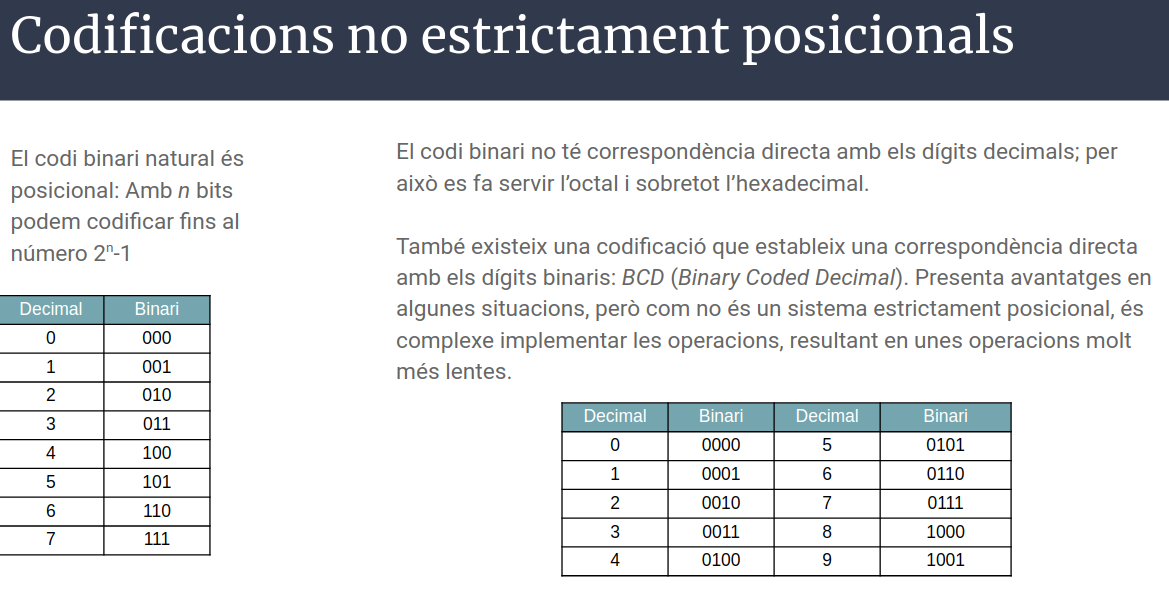


El punt fix(*manera de representar decimals en binari)* te aquests problemes. Això és fals perquè és en base 2. El 0.1 és periòdic. (S’ha de tenir en compte quan son números decimals)

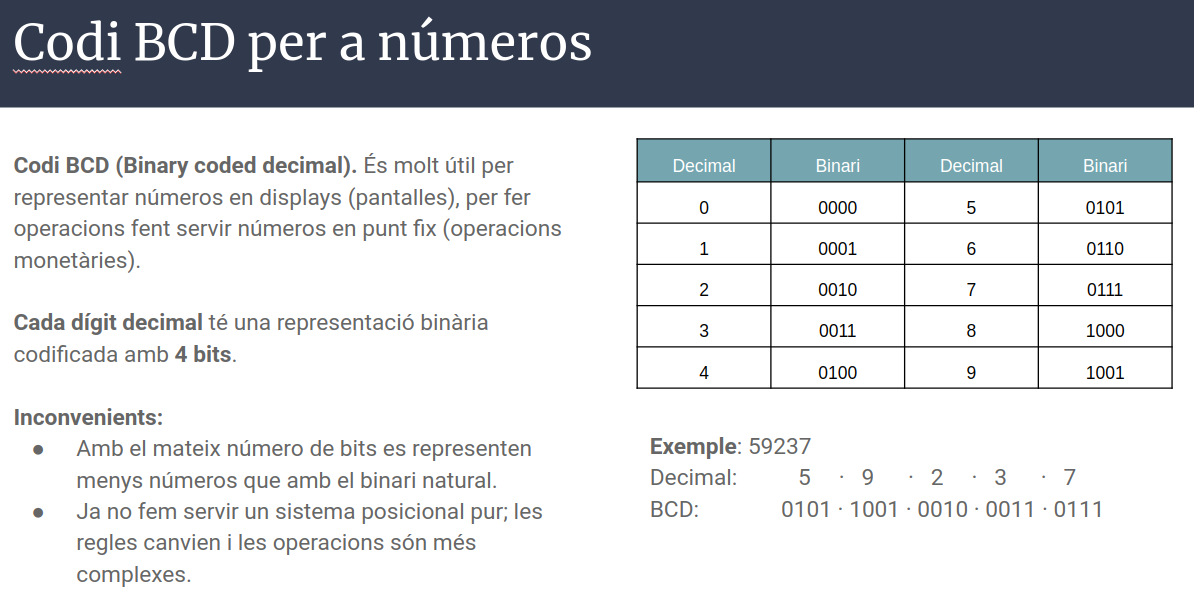




## Codificacions no estrictament posicionals

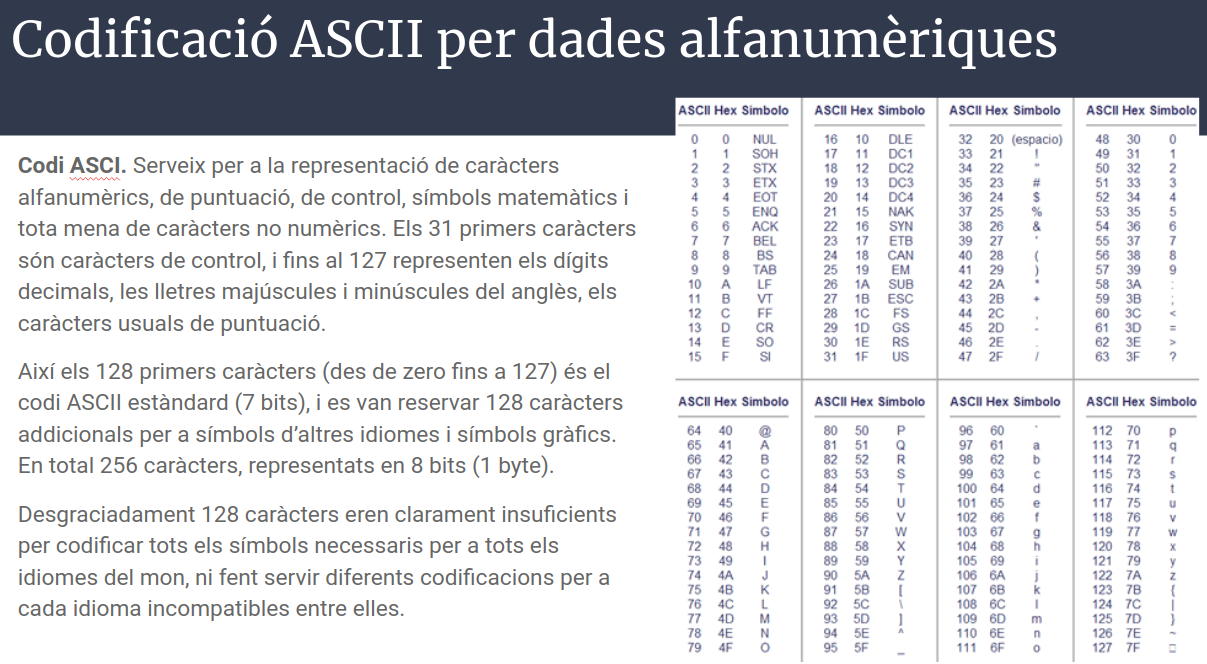


## Codi BCD per a números



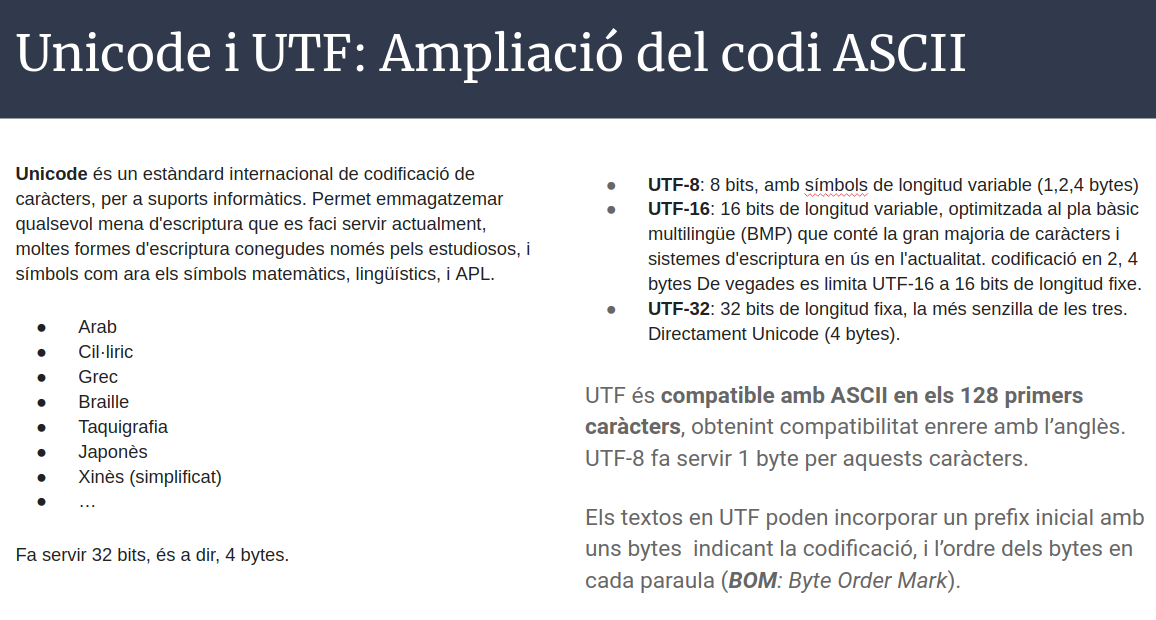
## 

## Codificació ASCII



L’ordinador treballa amb ASCII, però ens procesa la codificació per que ho veiem nosaltres com ho utilitzem al teclat (I viceversa). Quan escrivim, l’ordinador ho transofrma a ASCII.

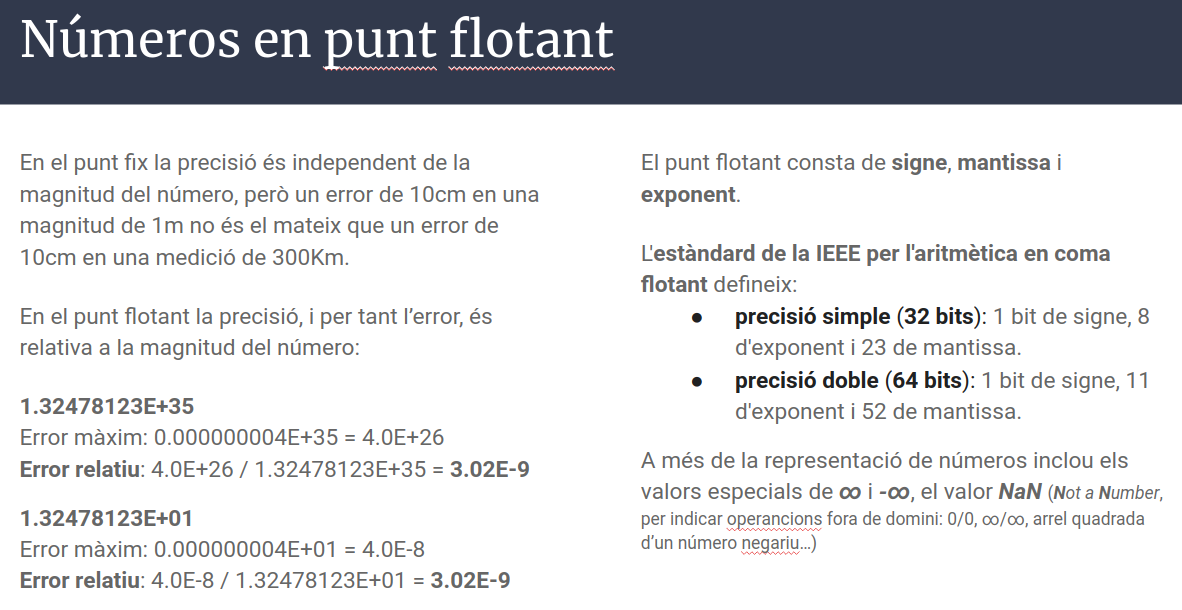
## Unicode i UTF: Ampliació del codi ASCII



Els primer caracteres de Unicode són d’ASCII (aixi son compatibles)

## 

## Números en punt flotant



Tenim 4 bits abans del punt, i quatre després. 0011.0001

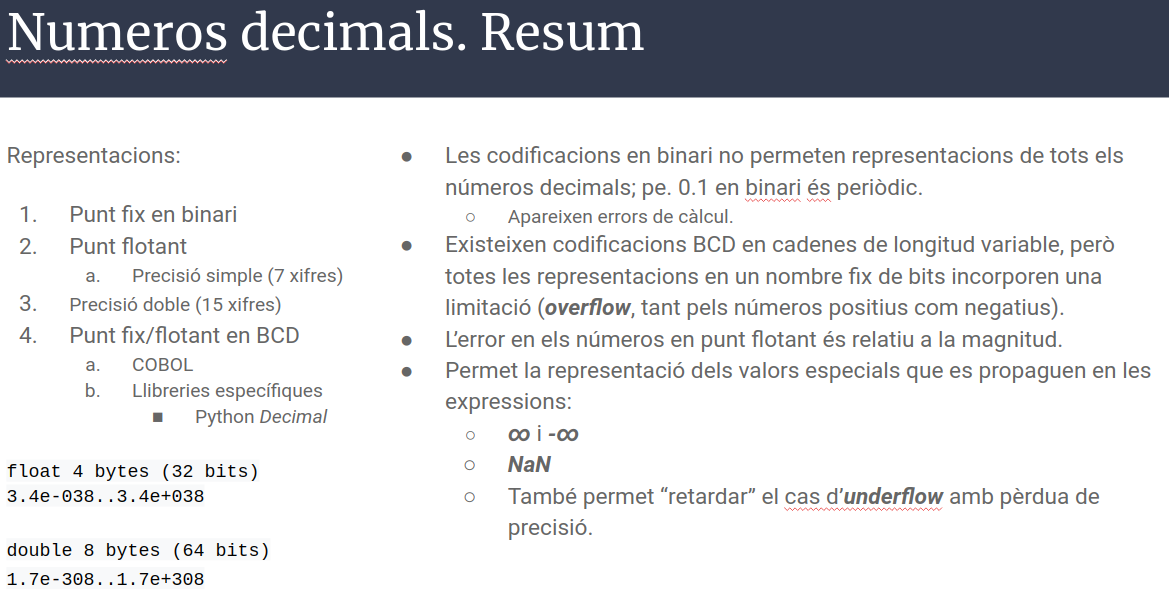
Si agafem un número molt gran i un molt petit, es perd informació.



Overflow: Que no cap per gran. / Underflow: no arriba a poder presentar-lo i diu que es 0.

Un error a un número petit, el error es deduiex (és més petit). Si el número és més gran, però serà relatiu al número. Al final el error es el mateix.

## Números en decimal. Resum



### Important, resumen de la primera unitat.

* base 2
* Hexadecimal
* Saber que existeix el octal
* Una idea de com va el punt fix i el punt flotant.
* Conceptes: ASCII, Unicode

(pasa el power point al de codificacio part 2 i despres tornarem al de la unitat 1)

Codificació Part 2

# 

# 1. Endianness

L’ordre dels bytes pot ser: (O per la part petita del ou, o per la gran)

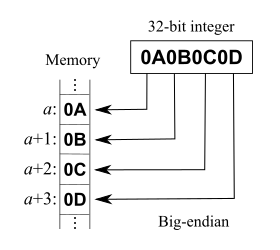
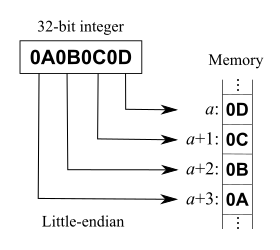
* Little-Endian (Little-End-In), Intel, AMD. → per la petita
* Big-Endian (Big-End-In), Network order, used in Internet comunications, IBM, SPARC, Motorola. → per la gran

Alguns processadors soporten Bi-endianness, com els nous ARM. (els dos)

Altres processadors incorporen funcions de conversió (Intel: bswap; ARM: rev).

## Ordre dels bytes: Big-Endian i Little-Endian

L’ordre dels bytes afecta tant a les dades com als mateixos programes i les seves instruccions.



L’ordre afecta, al Llitle-endian: començaria guardant a la memoria primer el menys significatiu (el 0D) i al final (0A). I el big-endian comença per la part més important del bite (0A).

## Endianness (ordre dels bytes)

Alguns processadors soporten bi-endianness (pe. els ARM més nous), però es donen diferents situacions de bi-endianness: (*¿Es cambia tot o només les dades? les instruccions per exemple. Si el canviem tot, si estem treballant a una forma canviar-ho a tota l’altre manera és molt delicat)*

* Només per dades, o per les dades i els programes.
* Configurable des de programari, o configuració del processador en la placa mare.

La característica d’ordre dels bytes d’un processador representa un problema quan s’han d’intercanviar arxius; el problema no es dona en ASCII, perquè és una codificació només en 1 byte.

*Quan canviem un arxiu com l’anomanem? big o little.*

## Solucions endianness per l’intercanvi d’arxius

1. Fixar l’ordre dels bytes com a part de l’especificació del format de l’arxiu.

* És el cas d’arxius **XSL**, que són Little-Endian i requereixen conversió en els Big-Endian.
* El sistema d’arxius **FAT** és Little-Endian, independentment del processador que el faci servir.

1. Indicar l’ordre dels bytes en l’arxiu. *dir quin format té.* 
   * *Ex:* Les imatges **TIFF** comencen amb **II** per a Little-Endian i per **MM** els Big-Endian; II es refereix a **Intel**, i MM a **Motorola**; el codi es palíndrom perquè sigui llegit igual en tots dos sistemes.
   * Els arxius en **UTF** incorporen un prefix anomenat **BOM** (Byte Order Mark), que indica l’ordre dels bytes.

## UTF BOM (Byte Order Mark)

*Una solució es indicar-ho.*

l primer caràcter d’un arxiu UTF és el BOM

UTF-16:

* Big-Endian: FE FF. (visualitzat com a þÿ)
* Little-Endian: FF FE. (visualitzat com a ÿþ)

UTF-32:

* Big-Endian: 00 00 FE FF.
* Little-Endian: FF FE 00 00.

| Cal anotar que ni FFFE ni FEFF són caràcters vàlids en UTF-16, però tampoc FF ni FE són caràcters vàlids en UTF-8, així que el BOM també permet detectar que l’arxiu és de codificació UTF-16.  El caràcter 00 és el caràcter *NULL*, que tampoc és un caràcter vàlid, així que també es pot detectar que l’arxiu està codificat en UTF-32.  ***En UTF-32 es pot ometre el BOM*** quan és Big-Endian, però també si són arxius que gestiona una aplicació, pe. arxius de configuració. |
| --- |

## 

## UTF-8 BOM

En UTF8 el BOM són 3 bytes amb el valor EF BB BF (es visualitza com a **ï»¿**).

No es requereix fer-ho servir, però actua com a [*magic-number*](https://en.wikipedia.org/wiki/Magic_number_(programming)#In_files), per reconèixer la codificació UTF-8, una pràctica habitual en els sistemes basats en Unix. *Es posa perque d’aquesta forma sabem que es no ASCII. A un programa: els dos primers, o quatre primers bites, es per idenfitificar i se anomena magic number. A java es CAFEBABE.*

La diferència amb un arxiu ASCII consistirà en els caràcters no codificables en els 128 primers caràcters ASCII (accents, dièresi, Ç, ela geminada, caràcters d’altres idiomes...).

*Tant se val, per que ha de cabre en 1 bite, i si no cap en un, es posaron dos bites.*

* Si un arxiu UTF8 s’interpreta com a ASCII aquests caràcters seran il·legibles.
* Si un arxiu de UTF8 té format preparat per processar, si és llegit com a ASCII probablement provocarà errors quan sigui processat.

# 2. Base64, Radix64 i UTF7: Codificació de binari a text

De vegades és necessari incorporar un arxiu binari en un arxiu de text, per exemple per incorporar una imatge en comptes de tenir-la en un altre arxiu.

La codificació Base64 permet **codificar arxius binaris com a text**. *(Ve del correu electrònic, que estava pensat només per text i a més en anglès, això ja està solucionat avui dia). (Exemple d’avui dia: el html i el css, son arxius de text*

El text resultant es divideix en línies de longitud fixe, separades amb els caràcters de final de línea (CR+LF).

*A windows el final de final es CR+LF (les impressores antigües hi havia que carregar el papel, i aquesta era la manera de representar-ho). En linux es LF. S’ha de tenir present a l’hora de canviar de Windows a linux i viceversa. (LF=10, CR=13)*

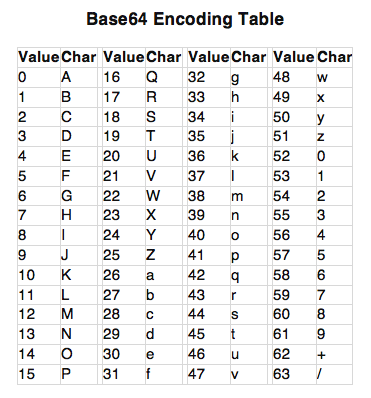
* Radix64 incorpora una verificació final (CRC, Control de redundància cíclica) *(per verificar un contingut binari, es fa una suma de tots els bits. i el resultat és el CRC)*
* UTF7 mai reconegut, en desús. No dividia en línies el text resultant.

## Codificació Base64

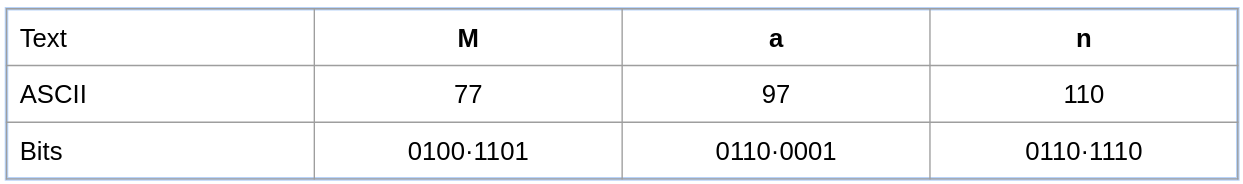
Ens limitem a 64 símbols, de forma que tots són caràcters imprimibles: 26 lletres majúscules i 26 de minúscules, 10 dígits i els símbols ‘+’ i ‘/’.

Cada 3 bytes (24 bits) generarem 4 caràcters, fent servir 6 bits per a cada caràcter, 64 valors diferents. *(En comptes de 8 en vuit)*

A més es fa servir el símbol “=” com a “*padding*”, per fer el nombre final de caràcters resultants divisible entre 3. *(Per que al ser de tres en tres, poden faltar i es posa == per acabar)*

25 lletres per mayúscules, 25 per minúscules, 

## Codificació



Amb representació de 3 bits: (Sis bits)

No hi haura cap número més gran del 64

Exemple de text codificat:

| TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz  IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg  dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu  dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo  ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4 = |
| --- |

[Base64 encode/decode online tool](https://www.base64decode.org/)

**Utilitat:**

* Actualment, per incorporar imatges en **HTML**, **CSS** i **SVG** (*Que no vagi adjunt, px: quan vols incorporar el logotip de l’empresa al correu)*
* El hash dels passwords dels sistemes tipus Unix es guarda en un arxiu de text ([/etc/passwd i /etc/shadow](https://en.wikipedia.org/wiki/Passwd#Password_file)) en Radix64. *(La carpeta de Unix de configuracions es la etc, el problema es que es binari (i linux no treballa en binari), llavors fa un Radix64 (Es igual que base 64 però amb un CF al final)*

*El hash es una funció que agafa la contrasenya i la codifica, i no es deuria descodificar. pero actualment, ja es pot descodificar.*

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

Introducció als sistemes combinacionals

(Segonda part del pwp UF1)

En els **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

Els dispositius electrònics amb funcions booleanes s’anomenen **comportes lògiques**. Les comportes lògiques són circuits electrònics que operen amb una o més senyals d’entrada per produir una senyal de sortida.

En els sistemes **combinacionals** la sortida és únicament f**unció de la entrada**, i per ser sistemes discrets, el seu comportament queda completament determinat definint la sortida resultant per a cada una de les combinacions de les dades d’entrada, en forma de les anomenades taules de veritat. ***(Els combinacionals SON FUNCIONS QUE DEPENEN NOMÉS DE LES ENTRADES)***

Altres sistemes digitals més complexos són els sistemes seqüencials, que tenen estats interns i la sortida no depèn únicament de les entrades, però només veurem sistemes combinacionals.

## 

## Algebra de Boole

Matemàtiques bàsiques per al disseny de sistemes digitals: Formulisme matemàtic per operar les funcions de commutació:

* Desenvolupada per Georges Boole al 1847 per problemes de lògica matemàtica
* Claude Shannon al 1939 l’aplica per primer cop a funcions de commutació *Ho va aplicar a l’electronica.*

Definicions

* **Variable lògica:** variable que pot assolir únicament dos valors {(0,1), (L,H), (F,V)}
* **Funció lògica:** funció definida amb variables lògiques el resultat de la qual només pot assolir dos valors {(0,1), (L,H), (F,V)}
* **Àlgebra:** conjunt d’elements, S, format, com a mínim, per dos elements diferents, amb dues operacions internes, suma (+) i producte (•) (que anomenarem suma lògica i producte lògic). Els elements satisfan el principi de substitució. *Es defineix pel producte i la suma.*

## Sistemes digitals seqüencials: Portes lògiques

[(El document de portes lògiques: més fàcil)](https://docs.google.com/document/d/1thFbUTfETSg43UhLogstfBXbP32MiGSERoA8xcnnPe4/edit?usp=sharing)

**NOT:** Realitza una inversió a nivell d'entrada. Si a l'entrada hi ha un nivell alt (1), a la sortida hi haurà un nivell baix (0), i viceversa.

**OR:** La porta OR implementa la suma lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 1, a la sortida hi haurà un 1. Només quan a les dues entrades hi hagi un 0 a la sortida hi haurà un 0 també.

**NOR:** La porta NOR implementa la negació de la suma lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 1, a la sortida hi haurà un 0. Només quan a les dues entrades hi hagi un 0 a la sortida hi haurà un 1.

**AND:** La porta AND implementa la multiplicació lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 0, a la sortida hi haurà un 0. Només quan a les dues entrades hi hagi un 1 a la sortida hi haurà un 1 també.

**NAND:** La porta NAND implementa la negació de la multiplicació lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 0, a la sortida hi haurà un 1. Només quan a les dues entrades hi hagi un 1 a la sortida hi haurà un 0.

**XOR:** La porta XOR funciona de forma que donarà 1 a la sortida sempre que el valor de les entrades sigui diferent, 0 en cas contrari. (Es el exclusiu, que son diferents)

**XNOR:** La porta XOR funciona de forma que donarà 1 a la sortida sempre que el valor de les entrades sigui iguals, 0 en cas contrari.



## 

## Taula de veritat

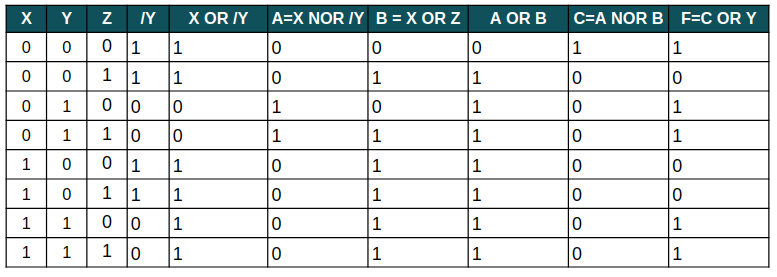
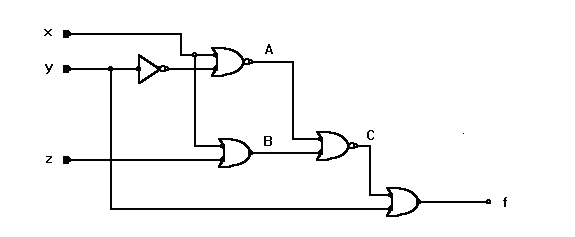
**Taula de veritat:** per una funció de n variables tenim una columna amb les 2n combinacions d’1 i 0 que es poden formar i un altre columna amb el valor de la funció per aquestes entrades).

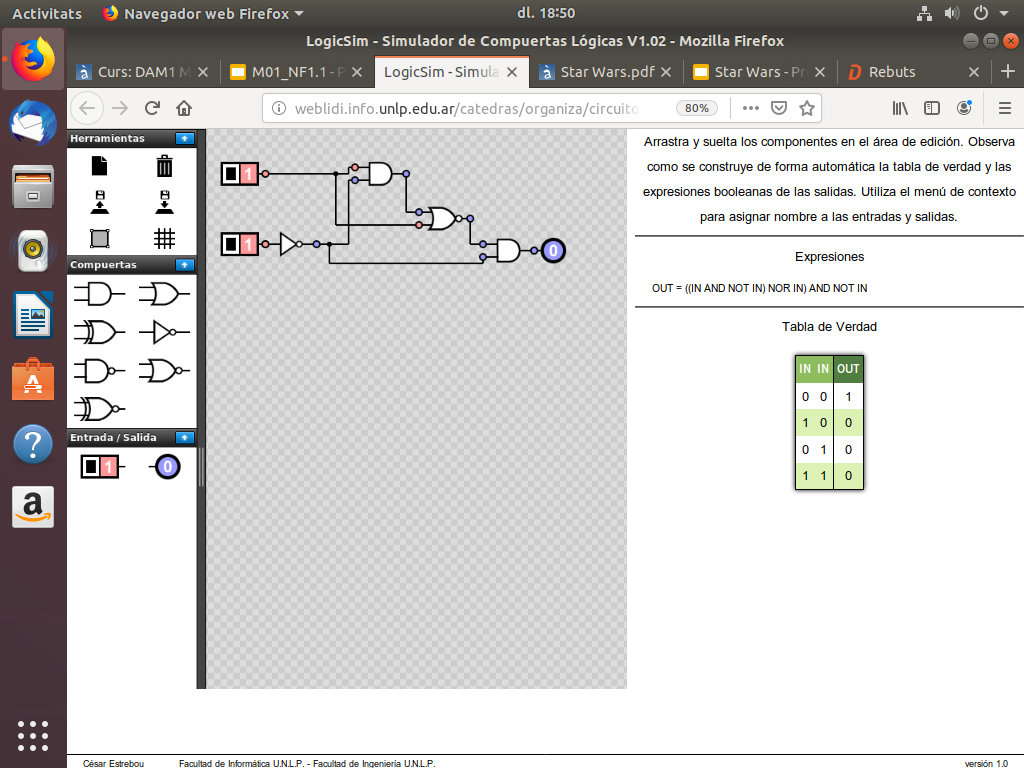
Dues funcions diferents tenen taules de veritat diferents.

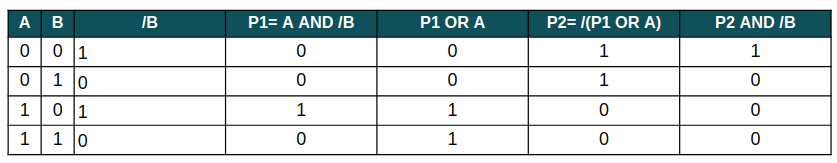
* Per N variables hi ha 22N funcions de commutació.
  + Així per a 1 variable tenim 4 funcions possibles
  + Per 2 variables, 16 funcions possibles
  + Per 3 variables, 256 funcions possibles

Aquest programari us servirà per simular taules i circuits lògics → [AQUÍ](http://weblidi.info.unlp.edu.ar/catedras/organiza/circuitos/editor_simple.html)

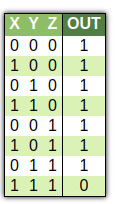
### Exemples



((A AND /B) NOR A) AND /B



### Exercici: NOT(X AND Y) OR NOT(X AND Z)



L’expressió que em dona: OUT = (Z NAND X) OR (X NAND Y)

## Lleis de morgan

1. NOT (A OR B) = NOT A AND NOT B

| **A** | **B** | **NOT A** | **NOT B** | **A O B** | **NOT (A OR B)** | **NOT A AND NOT B** |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |

1. NOT (A AND B) = NOT A OR NOT B

| **A** | **B** | **NOT A** | **NOT B** | **A O B** | **NOT (A OR B)** | **NOT A AND NOT B** |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |