1. Sistemes informàtics

1.1: informàtica

Informàtica: Ciència que estudia el tractament automàtic i racional de la informació.

Però què és la informació?

- **Dades**: Una dada pot ser numérica, un color o una paraula, però tota sola no significa res.
- **Informació**: Una dada es converteix en informació quan incorpora un significat associat, pe el teu nom, la teva edat o el teu color favorit. *Quan una dada li associem un significat.*
- **Coneixement**: La informació es converteix en coneixement quan s'interpreta. El **saber** és el conjunt de coneixements. *Quan s'interpreta. El conjunt de coneixements es saber.*

Podem definir la informació de diverses maneres:

- La informació és **tota forma de representació de fets, objectes, valors, idees, etc.**, que permet la comunicació entre persones i l'adquisició del coneixement de les coses.
- La informació és el **resultat de la manipulació de les dades**, treballant-les i ordenant-les amb la finalitat de produir un coneixement.

El **bit (*bi*nary digi*t*)** és la unitat base de mesura de la informació, que indica la quantitat mínima que forma la informació (cert/fals sí/no, obert/tancat...). Es representa mitjançant dos símbols, 0 i 1, anomenats **bits**.
Un grup de 8 bits s'anomena **byte**. També es coneix amb el nom d'**octet**.

Múltiples del byte.

kilobyte	**kB**	**10^3 bytes**	**kibibyte**	**KiB**	**2^{10} bytes**
megabyte	**MB**	**10^6 bytes**	**mebibyte**	**MiB**	**2^{20} bytes**
gigabyte	**GB**	**10^9 bytes**	**gibibyte**	**GiB**	**2^{30} bytes**
terabyte	**TB**	**10^{12} bytes**	**tebibyte**	**TiB**	**2^{40} bytes**
petabyte	**PB**	**10^{15} bytes**	**pebibyte**	**PiB**	**2^{50} bytes**

Codificació

El procés de ***codificació***, permet convertir la informació origen en unes dades codificades, que s'emmagatzemen o s'envien a un receptor. El procés invers és el de ***descodificació***, on es converteix aquests codis en informació interpretable per al destinatari.

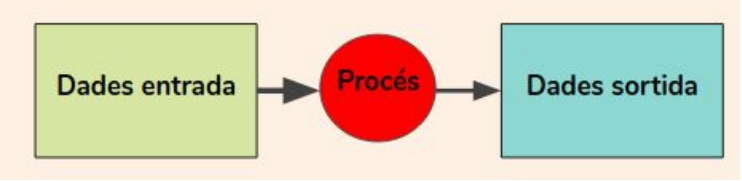
Les computadores emmagatzemen les dades codificades, fent servir **codificació binària** que és adequada per la seva manipulació. Per les dades que no són els nombres naturals calen altres codificacions; per als caràcters tenim el codi **ASCII** que codifica els caràcters en un número que es pot emmagatzemar en **un byte**.

El codi ASCII era insuficient per representar tots els caràcters de les diferents llengües del món. Es feia servir un codi **ASCII estès** per a cada llengua que requeria símbols diferents de l'anglès; calia conèixer quina versió d'ASCII s'havia fet servir i no es podien fer escrits incorporant símbols de diferents idiomes. La globalització d'internet va fer d'aquest sistema una gran barrera idiomàtica.

La codificació en **Unicode** inclou tots els idiomes coneguts (amb limitacions per al xinès) i fins i tot alguns d'imaginaris, com ara el Klingon o el Tengwar (èlfic), però calen **4 bytes** per codificar tots els caràcters possibles.

Processament de la informació

- Processament de la informació: unes **dades d'entrada** (*informació*) es **processen** per produir unes **dades de sortida** (*informació*).



El ***tractament automàtic de la informació*** neix al voltant dels anys quaranta quan surten al mercat les màquines automàtiques, que tracten la informació.
La paraula informàtica va aparèixer a França en 1962, i el seu origen són les paraules **INFOR**mation auto**MATIQUE**.

En els països de parla anglesa es coneix com a ***computer science***.

1.2 Components d'un sistema informàtic

Un **sistema d'informació** és un sistema format per **persones**, **dades**, **activitats**, i en definitiva, el conjunt de recursos que **processen la informació** d'una organització; un sistema d'informació habitualment té un component de tecnologies de la informació, un component informàtic.

La finalitat d'un **sistema informàtic** és aconseguir el millor **tractament automàtic** possible de la informació. Un sistema informàtic està format per un conjunt d'elements interrelacionats: **maquinari**, **programari** i **recursos humans**.

- El **maquinari** és **tot element físic**, material, del sistema informàtic com pot ser un ordinador, un teclat, una pantalla, suports d'emmagatzematge, cables de connexió i un llarg etcètera.
- El **programari** (*software*, en anglès) és el conjunt dels programes informàtics, procediments i

documentació que fan alguna tasca en un ordinador. *Software. Tots els programes i la seva documentació.*

- **Programari bàsic**: és el conjunt de programes que l'equip físic necessita per tenir capacitat de treballar. Aquests configuren el que s'anomena en un sistema informàtic el **sistema operatiu**. *generalment el que està incorporat al sistema operatiu.*
- **Programari d'aplicació**: són els programes que fan que l'ordinador desenvolupi una determinada tasca. *més específic, a una tasca concreta.*

- **Recursos Humans:** La estructura humana que treballa en un sistema informàtic la està formada per les parts següents:
  - **Usuari**: persona que utilitza la informàtica com a eina per desenvolupar el seu treball o ajudar-se en una activitat. *el que el fa servir*.
  - **Personal informàtic**: conjunt de persones que controlen i manipulen les màquines perquè donin el servei adequat a aquelles persones que necessiten utilitzar la informàtica per a les seves necessitats com a usuaris.
    - **Direcció**. Entre d'altres funcions, té la de coordinar i dirigir la part informàtica o algunes de les seves àrees (un departament, una àrea de programació, una àrea d'anàlisi, etc.). *dirigeix a la resta de persones.*
    - **Anàlisi**. El personal que pertany a aquest grup són els responsables d'intentar trobar solucions o millores informàtiques als problemes que es plantegin. *busquen els problemes i solucions.*
    - **Programació**. Tradueixen a llenguatge de programació les solucions proposades pels analistes. La seva funció també és la de fer la traducció de les diferents accions al llenguatge natiu de la màquina (llenguatge màquina). Per provar-lo utilitzen jocs d'assaigs que són proposats pels mateixos analistes. *Traduixo del llenguatge.*
    - **Explotació**. Són els responsables d'executar els programes o les aplicacions que hi ha i de comprovar el funcionament dels equips i dels sistemes que hi ha. *executen els programes, la part de sistemes. Tener el programari a punt.*
    - **Operadors**. S'encarreguen del funcionament, l'execució i els processos directes del sistema, la preparació dels suports, els perifèrics i el material informàtic. *s'encarreguen del funcionament.*

2. Sistemes digitals

Els sistemes base:

- decimal (b=10)
- el binari (b=2),
- l'octal (b=8)
- l'hexadecimal (b=16).

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

2.1. Bit y byte

Un **byte** són 8 bits (**$2^3 = 8$).**

b: bit / B: byte

Unidad de medida	Símbolo	Relación
bit	bit	1 bit
Byte	B	8 bits

–

Cada **digit del sistema binari** pot tenir els valors 0 o 1. Els digits binaris reben el nom de **bit**, abreviació de ***bi**nary-digi**t***. És la **unitat mínima d'informació**.

Un **byte** són 8 bits, i en les computadores actuals és la **unitat mínima d'emmagatzematge**, i la unitat mínima que pot adreçar-se. El seu origen està en el codi ASCII i l'ús de computadores per àmbits generals; les computadores de la URSS (*Unió Soviètica*) tenien 40, 42, 43 bits...

Ha quedat demostrada la conveniencia del byte, per ser una potència de 2.

$2^3 = 8$

Sovint es fa servir ***b*** com a símbol del bit, tot i no ser una abreviació reconeguda: ***bit*** ja és una abreviació, i no hauria d'abreviar-se més.

També de manera informal es fa servir ***B*** com a símbol del ***byte***, però tot i que tampoc està reconeguda, en general sí que es diferència ***b*** per a bit (generalment en comunicacions) i ***B*** per a byte.

És important ser conscient de quan ens referim a un o a l'altre.

–

2.2 Múltiples del byte prefixos sistema internacional o base 2

Sistema internacional: SI (Base 10)

Sistema binario (Base 2)

Prefijos en el uso convencional de la informática

Nombre	Símbolo	Potencias binarias y valores decimales	Valores en el SI	Hexa.	Nombre	Diferencia
unidad		$2^0 = 1$	$10^0 = 1$	16^0	un(o)	0 %
Kilo	K	$2^{10} = 1\,024$	$10^3 = 1\,000$	$16^{2,5}$	mil	2 %
Mega	M	$2^{20} = 1\,048\,576$	$10^6 = 1\,000\,000$	16^5	millón	5 %
Giga	G	$2^{30} = 1\,073\,741\,824$	$10^9 = 1\,000\,000\,000$	$16^{7,5}$	millardo	7 %
Tera	T	$2^{40} = 1\,099\,511\,627\,776$	$10^{12} = 1\,000\,000\,000\,000$	16^{10}	billón	10 %
Peta	P	$2^{50} = 1\,125\,899\,906\,842\,624$	$10^{15} = 1\,000\,000\,000\,000\,000$	$16^{12,5}$	billardo	13 %
Exa	E	$2^{60} = 1\,152\,921\,504\,606\,846\,976$	$10^{18} = 1\,000\,000\,000\,000\,000\,000$	16^{15}	trillón	15 %
Zetta	Z	$2^{70} = 1\,180\,591\,620\,717\,411\,303\,424$	$10^{21} = 1\,000\,000\,000\,000\,000\,000\,000$	$16^{17,5}$	trillardo	18 %
Yotta	Y	$2^{80} = 1\,208\,925\,819\,614\,629\,174\,706\,176$	$10^{24} = 1\,000\,000\,000\,000\,000\,000\,000\,000$	16^{20}	cuadrillón	21 %

2.3 Conversións entre prefixos SI i prefixos en base 2.

Prefijo: mega, kilo, etc

Hacer las multiplicaciones: ¿Cuanto es 250 MB más 240 GiB en bytes?

kilobyte	**kB**	**10^3 bytes**	**kibibyte**	**KiB**	**2^{10} bytes**
megabyte	**MB**	**10^6 bytes**	**mebibyte**	**MiB**	**2^{20} bytes**
gigabyte	**GB**	**10^9 bytes**	**gibibyte**	**GiB**	**2^{30} bytes**
terabyte	**TB**	**10^{12} bytes**	**tebibyte**	**TiB**	**2^{40} bytes**
petabyte	**PB**	**10^{15} bytes**	**pebibyte**	**PiB**	**2^{50} bytes**

Les computadores tenen una longitud finita per als números; si un número excedeix aquesta longitud es produeix un **overflow**.

3. conversions Binari.

3.1 binari - decimal

- Potencia de 2

1	0	0	**1**	.	**1**	0	**1**	**1**
128	64	32	**16**		**8**	4	**2**	**1**

128 + 16 + 8 +2 + 1 = 155 (Base 10)

3.2. binari (punt fix) a decimal *(Punt fix vol dir un binari amb decimals)*

- Representación en base 2

1	0	0	**1**	.	**1**	0	**1**	**1**
2^3	**2^2**	**2^1**	**2^0**		**2^{-1}**	**2^{-2}**	**2^{-3}**	**2^{-4}**

$2^3 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4} =$
8 + 1 + 0.5 + 0.125 + 0.0625 = 9.6875

3.3 decimal a binari

- El decimal entre 2

187 |_2__
1 93 |_2__
1 46 |_2_
0 23 |_2_
1 11 |_2_
1 5 |_2_
1 2 |_2_
0 **1** → **(De arriba abajo) 1011.1011**

3.4 Sumar números en binari

1			1	1			1	1	
	1	**0**	**0**	**1**	**.**	**1**	**0**	**1**	**1**
+	**1**	**0**	**0**	**1**	**.**	**1**	**0**	**1**	**1**
1	0	0	1	1		0	1	1	0

4. Conversion Hexadecimal

a. Conversió hexadecimal-binari

Binari	Hexadecimal	Binari	Hexadecimal
0	0	1010	A
1	1	1011	B
10	2	1100	C
11	3	1101	D
100	4	1110	E
101	5	1111	F
110	6		
111	7		
1000	8		
1001	9		

b. Conversió binari-hexadecimal

- Inmediat amb la taula:

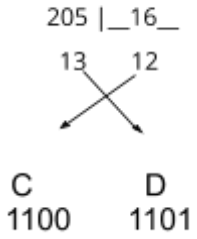
c. Conversió hexadecimal-decimal

Lo más simple sería pasar el hexadecimal a binari, y el binari a decimal.

9 B = 1001.1011 = 128+16+8+2+1= 155 Decimal

d. Conversió decimal-hexadecimal

- El número decimal entre 16



5. Números enters negatius

Teoria

8 bits: Entre -27 i 27 - 1 és a dir [-128 .. 127] (Será el mínim i el máxim)

16 bits: Entre -215 i 215 - 1, és a dir [-32768 .. 32767]

32 bits: Entre -231 i 231 - 1.

64 bits: Entre -263 i 263 - 1.

a. Complement a 2

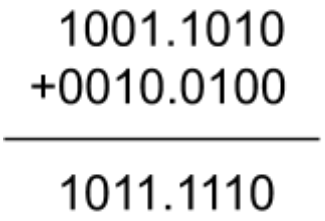
i. Complement 1

- El número lo convertimos en binario: -60. Cambiamos los 0 x 1 (viceversa)

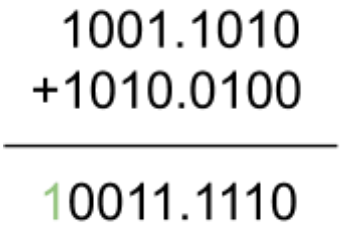
ii. Complement a 2

b. Suma de números en complement a 2

- Si son números positivos, se suma igual que los binarios



- Hay que tener en cuenta el overflow (situació en la qual es necessita un altre bit per representarla, per que si no quedaria en signe negatiu)



c. (ampliació) altres opcions per representar números negatius

i. Bit de signe

- Negatiu: comença en 1
- Positiu: comença en 0

ii. Excés a 2 N

El complement a 2 del número x fent servir N bits es defineix com:

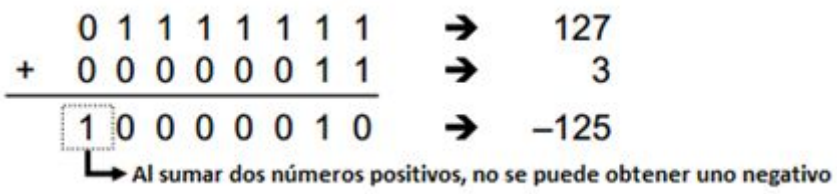
C2x = 2N - x

Però el seu càlcul en la pràctica és més senzill:
L'obtenim sumant 1 de la representació en complement a 1 del número:

C2x = C1x + 1

6. Problema de les representacions finites: Overflow

Dado que los números en un ordenador tienen una cantidad fija de bits, es posible que haya overflow, es decir, que el resultado tenga demasiados bits. Esto también, claro, nos puede pasar y nos pasa a nosotros mismos mientras operamos. Pero... ¿Cómo se cuando hay overflow? Podemos identificarlo a causa de un resultado incorrecto ya que no tendrá el signo que debería tener.



7. Representació de números reals

a. Números en punt fix

Punt fix: part entera i part decimal. Els números tenen representació finita en els computadors.

La capacitat de la part entera determina el rang de números representables, la capacitat de la part decimal indica la precisió amb que podem representar-los.

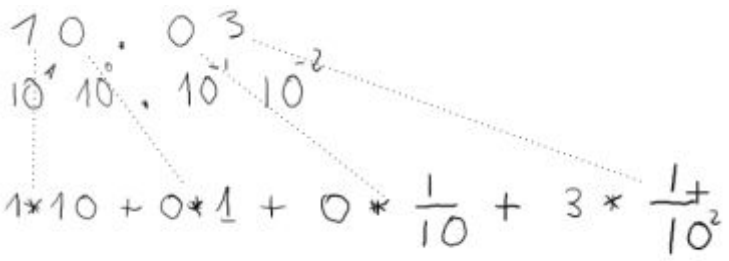
Quan la part entera és cero i la part decimal és tan petita que no pot representar-se es dona un **underflow**, situació en la qual el resultat no és zero però ha quedat representat com a zero.

Per evitar aquest problema molts càlculs financers fan servir números decimals (**BCD**) en comptes de binaris, tot i que és una característica avui dia en desús.

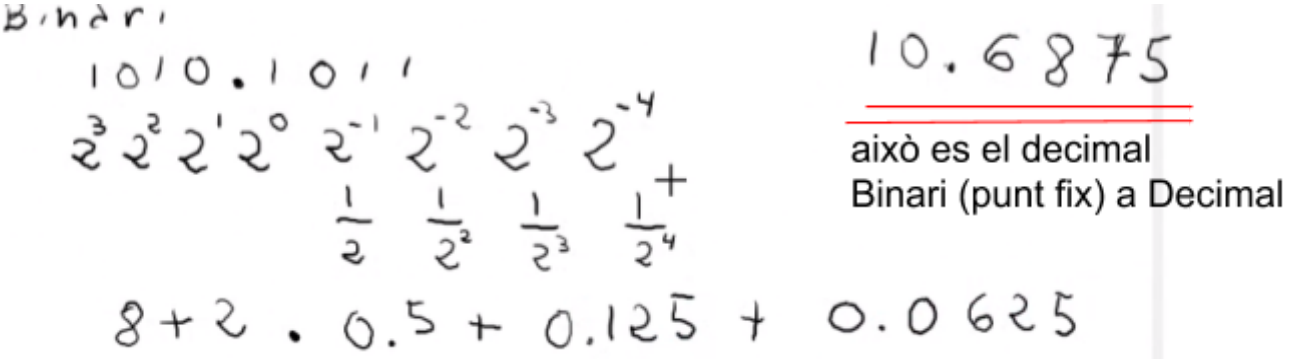
// punto fijo, es que el .(la coma) es fija. Si necesitamos sumar o restar se añadirán ceros.

Codi BCD (**Binary coded decimal**). És molt útil per representar números en displays (pantalles), per fer operacions fent servir números en punt fix (operacions monetàries).

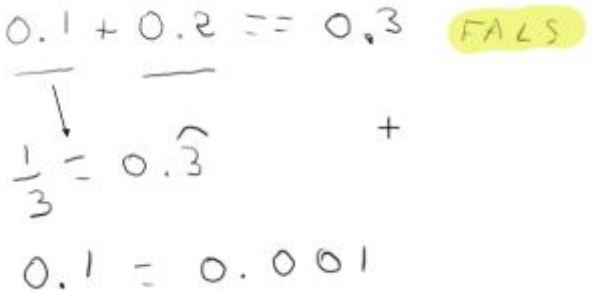
Cada dígit decimal té una representació binària codificada amb 4 bits.



← 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)



b. Números en punt flotant

En el punt fix la precisió és independent de la magnitud del número, però un error de 10cm en una magnitud de 1m no és el mateix que un error de 10cm en una medició de 300Km.

En el punt flotant la precisió, i per tant l'error, és relativa a la magnitud del número:

Como en el caso de números de base decimal, también es necesario en algunos casos trabajar con números muy grandes ó pequeños. Para ello resulta mas cómodo poder

expresarlos en un formato que permita operar con números de diferente posición de la coma. A esto se le denomina números de coma flotante.

El punt flotant consta de **signe, mantissa i exponent.**

L'estàndard de la IEEE per l'aritmètica en coma flotant defineix:

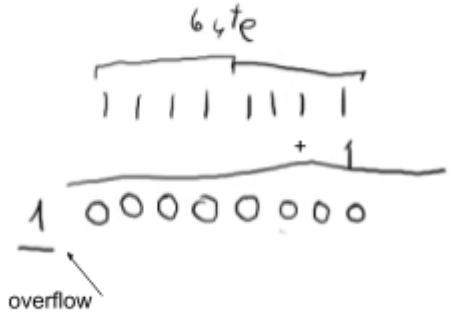
- **precisió simple** (32 bits): 1 bit de signe, 8 d'exponent i 23 de mantissa.
- **precisió doble** (64 bits): 1 bit de signe, 11 d'exponent i 52 de mantissa.

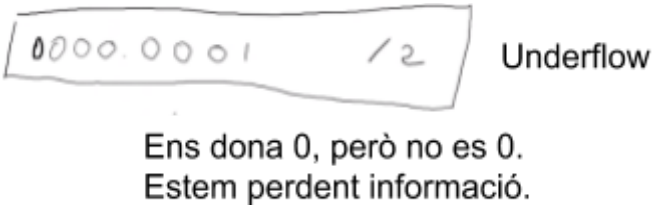
Ejemplo: +10E-07 Se leería, mas diez exponencial a menos 07.

- El primer signo es el número (+10)
- El signo después de la E, + (es positivo: más de 1) el - (es negativo, menos de 1)ç

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.

c. Problemes de les representacions dels reals

- Punt fix: Quan la part entera és cero i la part decimal és tan petita que no pot representar-se es dona un underflow, situació en la qual el resultat no és zero però ha quedat representat com a zero.

- Punt flotant: L'error en els números en punt flotant és relatiu a la magnitud.
- Les codificacions en binari no permeten representacions de tots els números decimals; pe. 0.1 en binari és periòdic. Apareixen errors de càlcul.

i. Codificació en base diferent: decimals periòdics per a valors representables en una altra base.

Al empezar a trabajar uno nota algo raro que ya aparecía en base 10, el problema es el número

0,99999999999.....

Este NO es "el número anterior a 1" ni nada que se le parezca. Este número ES 1.

0,99999999999.....=1

Para pasar un número entre 0 y 1 a otra base en vez de dividir varias veces hay que multiplicar.

Por ejemplo para escribir 0,89 en base 7

0,89*7=6,23 (me quedo con el 6, sigo con el 0,23)

Así que $0,89=0,61416141....{}_7$

ii. Precisió

1. Pèrdua de precisió i 2. Underflow

Quan la part entera és cero i la part decimal **és tan petita que no pot representar-se es dona un underflow**, situació en la qual el resultat no és zero però ha quedat representat com a zero.
La representació en base 2 pot donar-nos resultats inesperats: (1.1 + 2.2) == 3.3 és fals. Això és degut al fet que els nombres racionals no poden representar-se com a nombres amb decimals finits; pe. en base 10 no podem representar exactament ⅓. De la mateixa manera, en base 2 no es pot representar 1/10 de manera exacta, és a dir, que el valor 0.1 en binari no es pot representar exactament.
Per evitar aquest problema molts càlculs financers fan servir números decimals (BCD) en comptes de binaris, tot i que és una característica avui dia en desús.

d. Aventatges del punt flotant respecte el punt fix

En el punt fix la precisió és independent de la magnitud del número, però un error de 10cm en una magnitud de 1m no és el mateix que un error de 10cm en una medició de 300Km.
En el punt flotant la precisió, i per tant l'error, és relativa a la magnitud del número.

8. Representacions no posicionals

a. BCD

Codificacions no estrictament posicionals.

Presenta avantatges en algunes situacions, però com no és un sistema estrictament posicional, és complexe implementar les operacions, resultant en unes operacions molt més lentes.

Codi BCD (Binary coded decimal). **És molt útil per representar números en displays** (pantalles), per fer operacions fent servir números en punt fix (operacions monetàries).

Cada dígit decimal té una representació binària codificada amb **4 bits.**

Important:

Decimal	Binari	Decimal	Binari
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

- **Evita errors de reprensetanció**
- **Les operacions es fan byte a byte**
- Exemple: 59237

  Decimal: 5 · 9 · 2 · 3 · 7

  BCD: 0000 · 0101 · 1001 · 0010 · 0011 · 0111
- Aplicació en displays
- Llibreries de programació (pe. Python)

Inconvenients:

- Amb el mateix número de bits es representen menys números que amb el binari natural.
- Ja no fem servir un sistema posicional pur; les regles canvien i les operacions són més complexes.

- Llargues cadenes de números

c. BCD en la pràctica

- Representació de punt fix de COBOL (històric)
- Representació de números decimals en algunes bases de dades (punt flotant BCD)

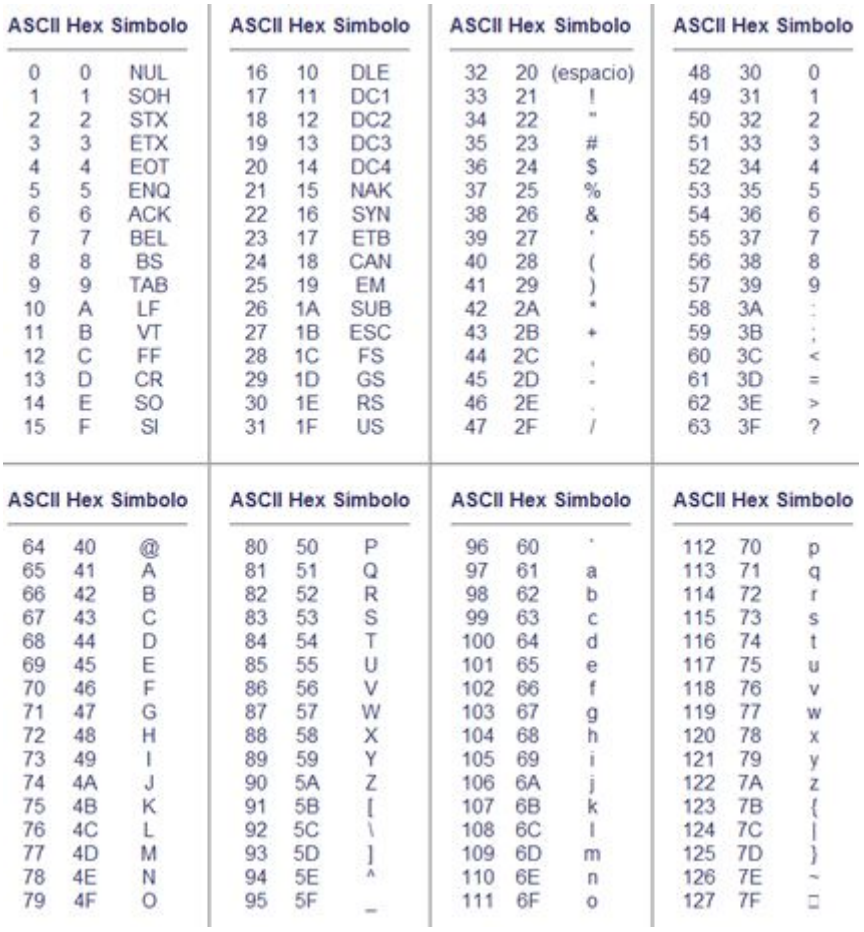
9. Representació de caràcters

a. ASCII

Serveix per a la representació de caràcters **alfanumèrics, de puntuació, de control, símbols matemàtics i tota mena de caràcters no numèrics**. Els 31 primers caràcters són caràcters de control, i fins al 127 representen els dígits decimals, les lletres majúscules i minúscules del anglès, els caràcters usuals de puntuació.

Així els 128 primers caràcters (des de zero fins a 127) és el codi ASCII estàndard (7 bits), i es van reservar **128 caràcters addicionals per a símbols d'altres idiomes i símbols gràfics**. En total 256 caràcters, representats en 8 bits (1 byte).

Desgraciadament 128 caràcters eren clarament insuficients per codificar tots els símbols necessaris per a tots els idiomes del mon, ni fent servir diferents codificacions per a cada idioma incompatibles entre elles.



ASCII	Hex	Simbolo
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US
32	20	(espacio)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	□

b. ASCII extés i taules de codi per a cada idioma diferent

Es van reservar **128 caràcters addicionals per a símbols d'altres idiomes i símbols gràfics**. En total 256 caràcters, representats en 8 bits (1 byte).

Desgraciadament 128 caràcters eren clarament insuficients per codificar tots els símbols necessaris per a tots els idiomes del mon, ni fent servir diferents codificacions per a cada idioma incompatibles entre elles.

c. Unicode

Unicode és un estàndard internacional de codificació de caràcters, per a suports informàtics. Permet emmagatzemar qualsevol mena d'escriptura que es faci servir actualment, moltes formes d'escriptura conegudes només pels estudiosos, i símbols com ara els símbols matemàtics, lingüístics, i APL.

i. UTF8

UTF-8: 8 bits, amb símbols de longitud variable (1,2,4 bytes)

ii. UTF16

UTF-16: 16 bits de longitud variable, optimitzada al pla bàsic multilingüe (BMP) que conté la gran majoria de caràcters i sistemes d'escriptura en ús en l'actualitat. codificació en 2, 4 bytes De vegades es limita UTF-16 a 16 bits de longitud fixe.

iii. UTF16

UTF-32: 32 bits de longitud fixa, la més senzilla de les tres. Directament Unicode (4 bytes).

iv. Compatibilitat ASCII d'UTF8 v. El BOM en UTF16 i UTF32

UTF és compatible amb ASCII en els 128 primers caràcters, obtenint compatibilitat enrere amb l'anglès. UTF-8 fa servir 1 byte per aquests caràcters.

Els textos en UTF poden incorporar un **prefix inicial amb uns bytes** indicant la codificació, i l'ordre dels bytes en cada paraula **(BOM: Byte Order Mark).**

(anex) 1. Endianness

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

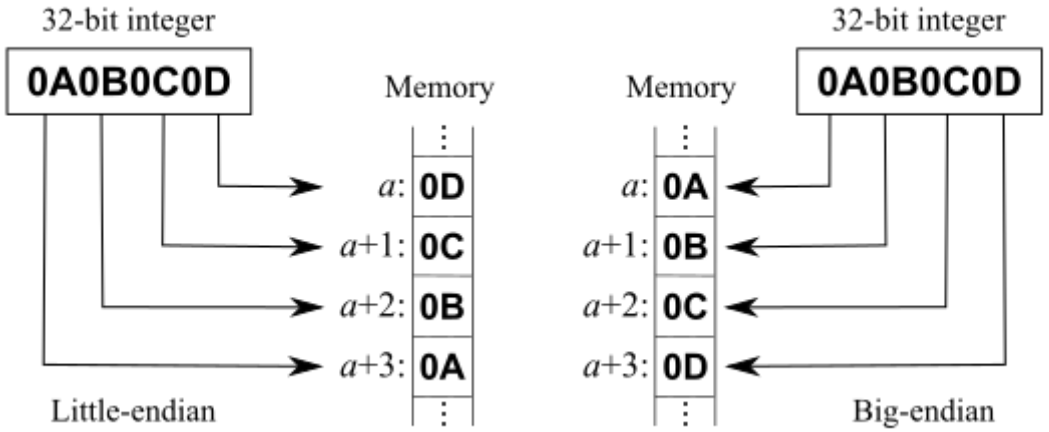
- Little-Endian (Little-End-In), Intel, AMD. → per la petita
- Big-Endian (Big-End-In), Network order, used in Internet comunications, IBM, SPARC, Motorola. → per la gran

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.

vi. BOM en UTF8

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*, 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.

UTF BOM (Byte Order Mark) 16-32 bits

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.

10. Representació de dades binaries en text: Base64

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.

a. 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)*

Base64 Encoding Table

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

25 lletres per mayúscules, 25 per minúscules,

b. Variant: Radix64 i. Incorporació de CRC (Control de Redundància Cíclica)

- 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)*

c. Utilitat i casos d'ús

- 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) 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.

ANNEX: CONVERSIÓ DE NOMBRES

Taules de conversió del binari

Binari a decimal (8 bits, valor màxim: 255)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Binari a hexadecimal (4 bits, valor màxim: F)

2^3	2^2	2^1	2^0
8	**4**	**2**	**1**

Recordatori dels valors decimals del digits hexadecimals

A	B	C	D	E	F
10	**11**	**12**	**13**	**14**	**15**