

# FACULTAD DE INGENIERIA Y CIENCIAS EXACTAS

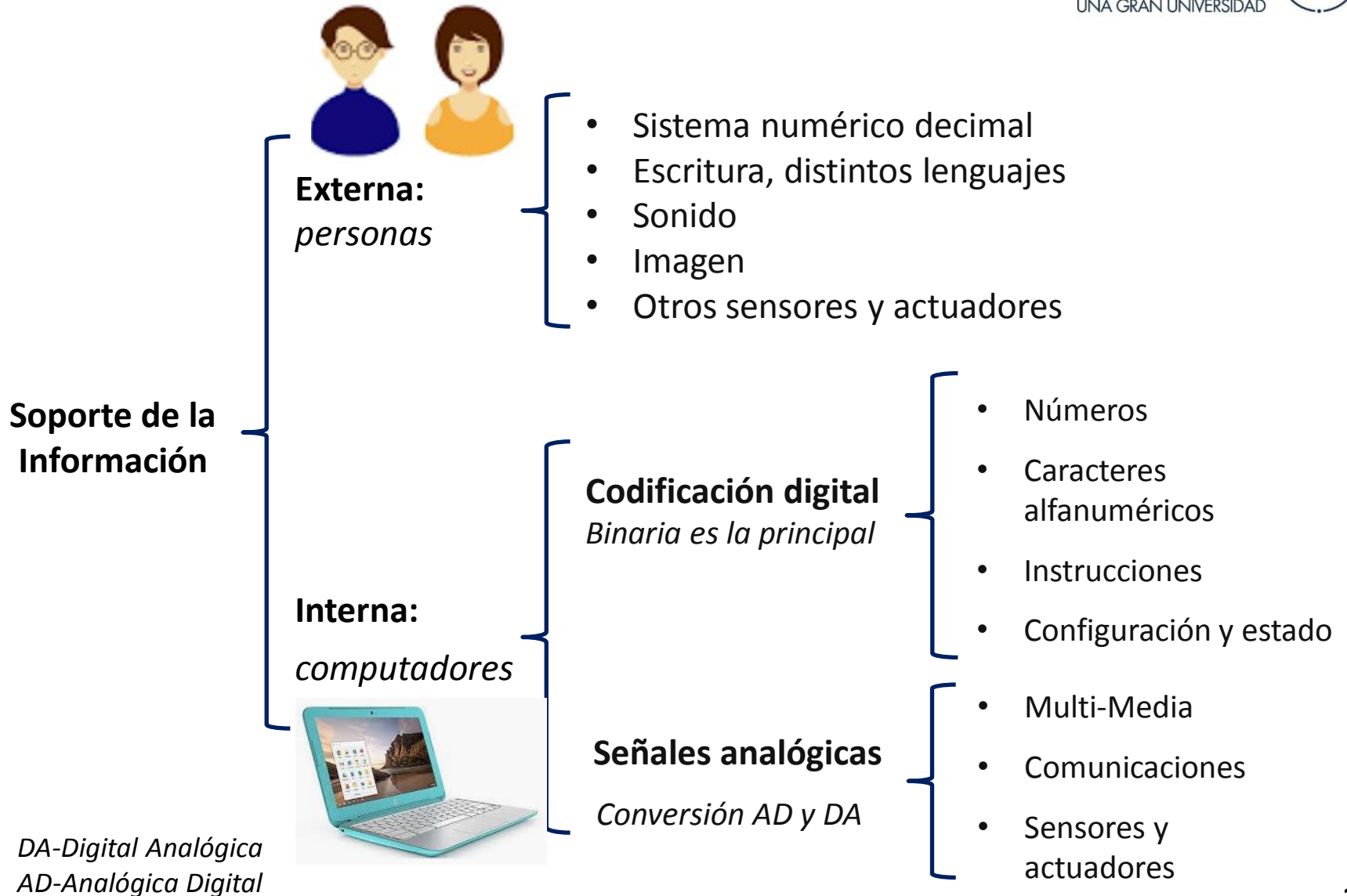
MATERIA: ARQUITECTURA DE COMPUTADORES (3.4.072)

Codificación interna en el computador



Ciudad de Buenos Aires, Argentina.

# Arquitectura de Computadores



## ● Sistemas Numéricos lineales:

Un sistema lineal permite la representación continua de los infinitos valores posibles de números expresados en cualquier base mediante el polinomio del ***teorema fundamental de la numeración*** con la sumatoria de infinitos términos.

Cualquier valor de una magnitud ANALÓGICA puede ser expresado.

## ● Sistemas Numéricos Modulares:

Podemos definir al módulo como la ***máxima cantidad de símbolos*** que puede representar el sistema. Este es el caso usado mayoritariamente dentro de un computador actual. El almacenamiento es ***modular y binario*** solo puede representar una cierta cantidad de dígitos binarios o bits en el bloque asignado.

# Arquitectura de Computadores

## ● Sistema Binario:

- El rango de representación viene dado por el intervalo de números representables en un formato.
- Con  **$n$  bits** se puede representar el rango de **0** a  **$2^n - 1$**  valores  
o sea, un total de  **$2^n$**  valores.

Si  **$n=4$**  se pueden representar

**$2^n = 2^4 = 16$  valores con 4 símbolos**

Si  **$n=2$**  se pueden representar

**$2^n = 2^2 = 4$  valores con 2 símbolos**



$b^3$	$b^2$	$b^1$	$b^0$	Dec.
8	4	2	1	
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	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Tabla Binaria de 4 bits:

- **Sistema Binario Modular:** Lo utilizan internamente los circuitos electrónicos (digitales). Cada número se representa en este sistema por un conjunto de elementos (símbolos), denominados **bit** (**b**inary **dig**it).

valor lógico del **bit** → “0” o “1”

valor físico del **bit** → Estados Eléctricos de  $\neq$  Valor

- El **byte** es el conjunto de **8 bits**, y es la **unidad elemental** de medida de la información representada mediante este sistema. Un byte es un **bloque** de 8 bits.

- Ej.: N° Binario de 8 dígitos =  $10110110_2 = 182_{10}$

- Byte 

<b>10110110</b>
-----------------

 = **Depende de la codificación !!!**

MSB

LSB

(Most Significant Bit)

(Least Significant Bit)



# Arquitectura de Computadores

Son bloques comunes utilizados por MS-Windows ®

● **Byte** **B** (8 bits)

10110110

● **Word** **W** (16 bits)

10110110 10110110

● **Double Word** **DW** (32 bits)

10110110 10110110 10110110 10110110

● **Quadruple Word** **QW** (64 bits)

10110110 10110110 10110110 10110110 10110110 10110110 10110110 10110110

*Estos bloques son los más utilizados por las codificaciones para los tipos de datos estándar de los diferentes lenguajes. Dependiendo del tipo de arquitectura o para tipos de datos definidos por el usuario se pueden ver otros bloques ; 80 bits, 128 bits 256 bits, etc*

## ● Sistema Digital Binario:

### ● Indicador luminoso

encendido (funciona)

apagado no funciona



### ● Juego de indicadores digitales binarios para el tablero de un automóvil

## ● Sistema Digital Decimal

- Odómetro digital de automóvil electrónico o mecánico



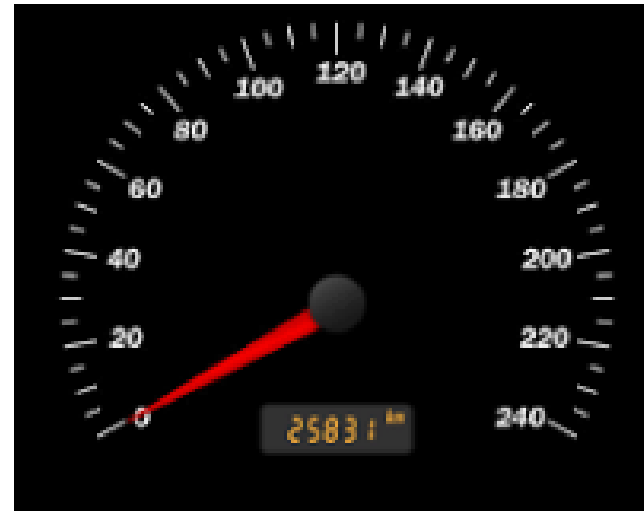
- Cada módulo corresponde a un dígito decimal





- **Sistema analógico**

- Velocímetro analógico de automóvil electrónico y mecánico



- **Códigos:**

- Es una representación de ciertos elementos a través de la asignación a cada una de ellos de una combinación determinada de símbolos.

**Ejemplos de código son:** El código postal, el código Morse, etc.

- Nos interesan los **códigos binarios de bloque**.

1. **Códigos *binarios*** son aquellos en que el alfabeto del código lo integran los dígitos binarios (0 y 1).
2. **Códigos *de bloque*** son aquellos en que las **distintas palabras** tienen todas el **mismo** número de símbolos. Pueden coincidir con un módulo para tratamiento de la información Ej: un **byte**

## ● Códigos:

- Hay algunos conjuntos de elementos que suelen necesitarse codificar con cierta frecuencia y que han sido estandarizados, caen en esta descripción los siguientes códigos:

1. Los códigos de **cambio único**. (1 bit entre códigos consecutivos)  
*Grey, Exceso 3, etc.*
2. Los códigos para representar los **caracteres alfanuméricos**.  
*Letras, números, signos de puntuación, control, gráficos, etc.*  
*ASCII, ANSI, EBCDIC, UNICODE*
3. Los códigos para representar los **números**.  
*Naturales, Enteros, Reales, Complejos.*
4. Los códigos **detectores (EDC) y/o correctores (ECC)** de errores.  
*Parity (P), Cyclic Redundancy Check (CRC), Hamming, etc.*

# Arquitectura de Computadores

## ● Representación de un número natural con bits: Código (8421) ->

- El rango de representación viene dado por el intervalo de números representables en un formato.
- Con  $n$  bits se puede representar el rango de  $0$  a  $2^n - 1$  o sea, un total de  $2^n$  valores.

Peso decimal  
del bit

$$8+2+1=11$$



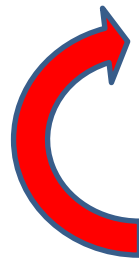
*Bloque de 4 bits:*

$b^3$	$b^2$	$b^1$	$b^0$	Dec.
8	4	2	1	
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	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

## ● Códigos de cambio único:

**Códigos Continuos:** son los códigos para los que la combinación que representa a un elemento no difiere más que en un bit de la que representa al elemento anterior

**Código Cíclico:** es un código continuo en el que tampoco difieren en más de un bit las combinaciones correspondientes al primer elemento y al último.



Elementos	$b_1$	$b_0$
A	0	0
B	0	1
C	1	1
D	1	0

← *Primer elemento*

← *Último elemento*

## ● Códigos de cambio único (Código Gray o Binario Reflejado):

Gray desarrolló una forma sistemática para diseñar códigos continuos y cíclicos para distintos números de elementos.

De la tabla surgen algunas conclusiones de interés:

1. Cualquier secuencia de palabras del código Gray forman un **código continuo**.
2. El código Gray es un **código de bloque**.
3. Tomando las primeras **2<sup>n</sup> palabras** del código, forman un **código cíclico**.
4. Dada una lista con las primeras 2<sup>n</sup> palabras del código Gray, las palabras ubicadas simétricamente con relación al eje que divide la lista en dos sólo difieren en 1 bit.
5. Como consecuencia de lo anterior, si de una lista de las primeras 2<sup>n</sup> palabras del código se suprimen simétricamente las primeras m palabras y las últimas m, la lista resultante es un código cíclico de  $(2^n - 2^m)$  elementos.

Binario				Gray			
b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



## ● Códigos de caracteres alfanuméricos:

**Se consideran caracteres a:**

Los códigos de control (CR, LF, TAB, SPACE)

Los dígitos ("0", ..., "9")

Las letras ("A", ..., "Z", "a", ..., "z") [1]

Los signos de puntuación (".", ",", ";", ...)

Los caracteres especiales ("\*", "&", "\$", "Ñ", "ü", "Á" ...)

Para representarlos se asignan códigos numéricos mediante una tabla  
El computador opera con los códigos, no con los símbolos gráficos.

**Las características de una representación son:**

Longitud en bits de los códigos. (Fija o Variable)

Número de caracteres representable.

La asignación de códigos a cada carácter (la tabla).

[1] Quedan excluidos caracteres en otros idiomas distintos del inglés como letras acentuadas o con diéresis y la letra ñ para el caso español. Estos caracteres pasan a ser tratados como símbolos especiales y no en la secuencia del alfabeto.

## ● Cuestiones asociadas con la codificación:

### **Internacionalización.**

- Usuarios de distintos países pueden interactuar con el sistema en su propia lengua.
- Proporciona las herramientas para la creación de aplicaciones internacionales.
- Creación de software multilingüe.
- Convenios nativos para la representación de datos.

### **Local: idioma, país y un conjunto de códigos**

- País
- Idioma
- Distribución de teclado
- Medida
- Separador de lista
- Formato de fecha
- Formato de hora
- Formato de moneda
- Formato numérico

## ● Cuestiones asociadas con la codificación:

**NLS** (*National Language Support*)

**El Soporte de Globalización asegura que:**

- los mensajes de errores de las diferentes aplicaciones y utilitarios
- clasificación ordenada
- Fecha
- Hora
- signo monetario
- convenciones numéricas
- convenciones de calendario
- automáticamente se adaptan al lenguaje nativo.

Un motor de base de datos actualmente soporta diferentes lenguajes y territorios.









Ordenamientos lingüísticos (mono-lingual y multi-lingual), y gran variedad de conjuntos de caracteres codificados.

## ● Cuestiones asociadas con la codificación:

### Distintos Tipos de Esquemas de Codificación

- Un esquema de codificación especifica los códigos que tanto un computador o terminal pueden mostrar y recibir.
- Se utilizan para interpretar los datos en símbolos significativos desde una terminal a una máquina host.
- Diferentes clases de esquemas de codificación:
  - ✓ Único byte.
  - ✓ Múltiples bytes.
  - ✓ Cantidad de Bits para la codificación variable.
  - ✓ Cantidad de Bits para la codificación fijo.

## Ejemplos de códigos:

-  Código Morse (Telégrafo eléctrico y versiones ópticas, 2 estados )
-  Código BAUDOT. (Teletipos cinco bits)
-  BCDIC
-  EBCDIC
-  Código ASCII y ASCII Extendido.
-  Código ANSI.
-  Código UNICODE.
-  Otras codificaciones

**Grisados** : No usados en informática actualmente

## ● Código Morse

Telégrafo eléctrico y óptico dos señales de distinta duración. Punto, menor duración y raya, mayor duración. (*dot and dash*)

## ● Código BAUDOT.

Longitud fija de **5 bits**.

Utilizado en Telex/Teletipo.

Codifica solo las letras mayúsculas, opera en forma contextual.

## ● Código BCDIC (Binary Coded Decimal Interchange Code).

Longitud fija de **6 bits**.

Utilizado por muchos sistemas de la firma IBM.

Codifica solo letras en mayúscula.

## ● Código EBCDIC (Extended Binary Coded Decimal Interchange Code).

Surgido en 1964 con el sistema IBM S360

Longitud fija de 8 bits

Utilizado en algunos sistemas *mainframe*.



## ● Código ASCII

*(American Standard Code for Information Interchange).*

- Se utiliza para la representación de los caracteres de texto (alfanuméricos) del alfabeto inglés, los signos de puntuación y algunos comandos.
- Propuesto en 1963 y finalizado en 1968.
- Esquema de longitud fija.
- **ASCII** original. Longitud de código de **7 bits** o sea 128 caracteres, se hizo estándar en los EEUU.
- **ASCII extendido (EASCII)**. Ampliación para caracteres internacionales, con una longitud de código de **8 bits**.
- Muy usado actualmente.

# Arquitectura de Computadores

Caracteres de control ASCII			
DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de línea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles								
DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	\$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_	<a href="http://elCodigoASCII.com.ar">elCodigoASCII.com.ar</a>		

# Arquitectura de Computadores

ASCII extendido											
DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
128	80h	Ç	160	A0h	á	192	C0h	Ł	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	ł	225	E1h	ô
130	82h	é	162	A2h	ó	194	C2h	Ť	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	Ŧ	227	E3h	Õ
132	84h	ä	164	A4h	ñ	196	C4h	—	228	E4h	ö
133	85h	à	165	A5h	Ñ	197	C5h	†	229	E5h	Õ
134	86h	á	166	A6h	ª	198	C6h	ä	230	E6h	μ
135	87h	ç	167	A7h	º	199	C7h	Ä	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	ℒ	232	E8h	ƒ
137	89h	ë	169	A9h	¿	201	C9h	ℓ	233	E9h	Û
138	8Ah	è	170	AAh	¬	202	CAh	ℓ	234	EAh	Ü
139	8Bh	ï	171	ABh	½	203	CBh	ℓ	235	EBh	Ù
140	8Ch	î	172	ACH	¼	204	CCh	ℓ	236	ECh	Ý
141	8Dh	ì	173	ADh	¡	205	CDh	ℓ	237	EDh	Ý
142	8Eh	Ā	174	A Eh	«	206	CEh	ℓ	238	EEh	—
143	8Fh	Ă	175	AFh	»	207	CFh	ℓ	239	EFh	·
144	90h	É	176	B0h	⋮	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	⋮	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	⋮	210	D2h	É	242	F2h	—
147	93h	ó	179	B3h	⋮	211	D3h	Ê	243	F3h	¾
148	94h	ò	180	B4h	⋮	212	D4h	È	244	F4h	¶
149	95h	ó	181	B5h	⋮	213	D5h	Ï	245	F5h	§
150	96h	û	182	B6h	⋮	214	D6h	İ	246	F6h	÷
151	97h	ù	183	B7h	⋮	215	D7h	İ	247	F7h	º
152	98h	ÿ	184	B8h	©	216	D8h	İ	248	F8h	º
153	99h	Ō	185	B9h	Ť	217	D9h	Ť	249	F9h	—
154	9Ah	Ů	186	BAh	Ť	218	DAh	Ť	250	FAh	·
155	9Bh	ø	187	BBh	Ť	219	DBh	Ť	251	FBh	·
156	9Ch	£	188	BCh	Ť	220	DCh	Ť	252	FBh	·
157	9Dh	Ø	189	BDh	¢	221	DDh	Ť	253	FDh	·
158	9Eh	×	190	BEh	¥	222	DEh	Ť	254	FEh	■
159	9Fh	f	191	BFh	¬	223	DFh	Ť	255	FFh	

# Arquitectura de Computadores

```
IMMMWIMMMWNNNNNNXXXXXXXXKKKKKKKK000KKKKK000000KKKKK
IMMMWIMWXK000000000000000000000000000000000000000000000
IMMMWIMWXK00kxxxxxxxxxxxxdddddoooooollllllllcc
IMMMWIMX00k1;,,, ''''''''''''''''''''''''''''''''''''''
IMMMWIMNK00d;'. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMNK00o;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMX00k1;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMX00kc;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMNK00x;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMX00d;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMX00o;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMX00k1;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMX00kc;.. ''''''''''''''''''''''''''''''''''''''''
IMMMWIMX00x;.. ''''''''''''''''''''''''''''''''''''''''
```

Banner Negativo de caracteres ASCII generado en línea a partir de una imagen.



<https://www.ascii-art-generator.org/es.html>



## ● Código ANSI

(American National Standards Institute).

- Últimas normas sobre el ASCII establecida a través ISO-14962-1997 and ANSI-X3.4-1986(R1997).
- **ANSI X3.64** (también denominado **ECMA-48**)
- Definido en 1991 Se utiliza para la representación de los caracteres de texto (alfanuméricos) del alfabeto inglés internacionalizado, los signos de puntuación, algunos comandos, mover el cursor y cambiar colores frente/fondo.
- Esquema de longitud fija.
- **ANSI x3.64**. longitud de código de **8 bits**.
- Muy usado actualmente (Windows® y otros SO).

Ecma International.org Standardization, Information and Communication Technology, Consumer Electronics, Industry association. Estándares ECMA.

## Código **UNICODE**.

Norma de codificación de caracteres que puede representar todos los caracteres utilizados por los computadores, incluyendo símbolos técnicos, caracteres utilizados en publicidad, etc.

Norma establecida por Apple y Xerox en **1988**

En **1991** se establece un comité con la participación de Adobe, Aldus, Apple, Borland, Digital, IBM, Microsoft, Novell, Sun, Unisys, Wordperfect y Xerox, Establecen la norma **Unicode**.

Longitud fija de **16 bits**. UTF-16

Existen codificaciones de 8 bits UTF-8 y 32 bits UTF-32



## ● Código **UNICODE**.

- La versión 2.0 representa hasta 38885 caracteres
- **UCS-2** (**U**niversal **C**haracter **S**et 2 byte form), es un código de 2 bytes con formato de longitud fija.
- **UTF-8** (**U**niversal Character Set **T**ransformation **F**ormat) es multibyte, con un formato de longitud variable.
- Tanto el UCS-2 como el UTF-8 codifican el mismo repertorio de caracteres tanto en Unicode 1.1 o Unicode 2.0
- El **Unicode 2.0** codificado como **UTF-8** incluye el **ASCII/ANSI** utilizando la codificación en un único byte.

- Código **UNICODE** Unicode UTF-16 ( 16 bits) Letra A mayúscula de la fuente Consolas mostrada por el mapa de caracteres de Windows 7®

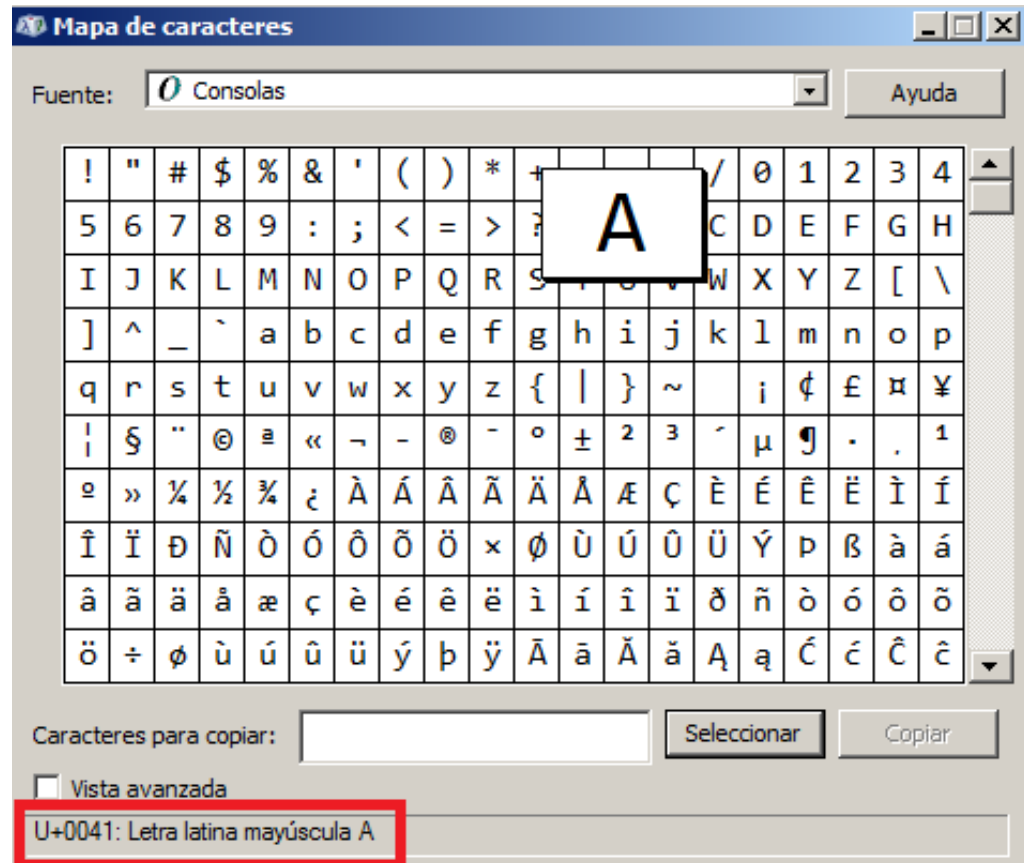
EL CODIGO ASCII, ANSI Y  
UTF-8 ES PARA "A":

HEX 41

Bin 0100 0001

HEX 0041

































BIN 0000 0000 0100 0001



- Código **UNICODE** actualmente
  - Dispone de la posibilidad de utilizar más de 65000 caracteres UTF-16/UCS-2 ( $2^{16}$ ), UTF-32/UCS-4 ( $2^{32}$ ).
  - Están definidos los alfabetos: árabe, chino, cirílico, griego, hebreo, japonés kana, coreano hangul, latín entre otros.
  - Símbolos de puntuación: matemáticos, técnicos, flechas, gráficos y otros caracteres.
  - Hay alrededor de 34000 caracteres utilizados.

🌐 Código **UNICODE** Unicode registra y clasifica todo tipo de gráficos e íconos para uso informático.

🌐 Ejemplo Unicode *Emoji data files*

face-hand														
Nº	Code	Browser	Appl	Goog	FB	Wind	Twtr	Joy	Sams	GMail	SB	DCM	KDDI	CLDR Short Name
29	<a href="#">U+1F917</a>									—	—	—	—	hugging face
30	<a href="#">U+1F92D</a>									—	—	—	—	face with hand over mouth
31	<a href="#">U+1F92B</a>									—	—	—	—	shushing face
32	<a href="#">U+1F914</a>									—	—	—	—	thinking face

<https://unicode.org/emoji/charts/full-emoji-list.html>

## ● Codificación de los Números.

● En las computadoras los números se usan para representar:

- Magnitudes binarias. *(Números naturales)*
- Enteros binarios. *(Números enteros)*
- Números reales binarios. *(Números reales)*
- Números Complejos binarios. *(Números complejos)*
- Decimales codificados en binario  
*(Decimales enteros y reales con coma fija)*
- Cadena de caracteres sólo para visualizarlos  
*(Todos los Números)*

## ● Clasificación de los números

● **N** (Naturales) 1, 2, 3 ....

*agregando los ( Negativos) -1,-2, -3*

● **Z** (Enteros) ... -3, -2, -1, 0, 1, 2, 3 ....

*agregando los fraccionarios -1.2 ... -0.4 ... 0.5 ... 2.8 ....*

● **Q** (Racionales)

*agregando los irracionales  $\pi$  (3.14159 26535...),  $e$  (2.7182818284...), ...*

● **R** (Reales)  $-\infty$  a  $+\infty$

*agregando los imaginarios  $R * (-1^{1/2})$*

● **C** (Complejos) (R , Ri)



## ● Codificación para Magnitudes Binarias:






- Se codifican normalmente usando el código binario “natural”.
  - Es la misma representación con la cual se escribe esa magnitud en binario.
  - Tratándose de un código de bloque, todas las magnitudes se representan con el **mismo número de bits**.

Ej.: Código de bloque de 8 bits, la magnitud binaria **11010**  
se representa como **0001 1010**

Las operaciones aritméticas se realizan con representaciones codificadas y el resultado se representa **igualmente codificado**.

Si el resultado de la operación no cae dentro del rango de representación del código, el mismo **no resultará válido**.

## Codificación para Números Enteros Binarios

-  Son códigos de bloque que utilizan  $n$  bits para la representación, difieren en la codificación.
-  Al utilizar  $n$  bits para la codificación, tienen  $2^n$  combinaciones disponibles para ser usadas como palabras del código.
-  Representan tanto a los números negativos como a los positivos.
-  Son capaces de codificar a los números enteros ubicados en el rango comprendido entre el  $-(2^{n-1}-1)$  y el  $+(2^{n-1}-1)$ .
-  Representan  $2^{n-1}-1$  números negativos, otros tantos números positivos, y el  $+/-0$ . Esto hace un total de  $2^n$  números

## ● Codificación para Números Enteros Binarios

Existen varios códigos de bloque que se utilizan para representar a los números enteros binarios.

- Código de **Signo y Magnitud**.
- Código de **Complemento a 1**.
- Código de **Complemento a 2**.
- Código **Binario Desplazado**.
- Código para **Números Binarios Reales**.

## ● Codificación en Signo y Magnitud:

- Se utiliza para representar enteros.
- Se reserva el bit mas significativo (**MSB**) para representar el signo.
- Se usa el 0 para los positivos y 1 para los negativos.
- Puede representar solo enteros comprendidos entre el  $-(2^{n-1}-1)$  y  $+(2^{n-1}-1)$ .
- Dado que cuenta con  $n-1$  bits para la magnitud solo puede representar números entre 0 y  $2^{n-1}-1$ .
- Doble representación del cero

$b_3$	$b_2$	$b_1$	$b_0$	S&M
0	1	1	1	+7
0	1	1	0	+6
0	1	0	1	+5
0	1	0	0	+4
0	0	1	1	+3
0	0	1	0	+2
0	0	0	1	+1
0	0	0	0	+0
1	0	0	0	-0
1	0	0	1	-1
1	0	1	0	-2
1	0	1	1	-3
1	1	0	0	-4
1	1	0	1	-5
1	1	1	0	-6
1	1	1	1	-7

## ● Codificación en Complemento a 1:

- Los números positivos se representan por su magnitud.
- Los números negativos se representan por el **complemento a 1** de su magnitud.
- **No** se reserva bit para el signo.
- Mantiene la doble representación del 0, pero no se considera ni positivo ni negativo.
- Puede representar solo enteros comprendidos entre el  $-(2^{n-1}-1)$  y el  $+(2^{n-1}-1)$ .

$b_3$	$b_2$	$b_1$	$b_0$	Ca1
0	1	1	1	+7
0	1	1	0	+6
0	1	0	1	+5
0	1	0	0	+4
0	0	1	1	+3
0	0	1	0	+2
0	0	0	1	+1
0	0	0	0	+0
1	1	1	1	-0
1	1	1	0	-1
1	1	0	1	-2
1	1	0	0	-3
1	0	1	1	-4
1	0	1	0	-5
1	0	0	1	-6
1	0	0	0	-7

## ● Codificación en Complemento a 2:

- Los números positivos se representan por su magnitud.
- Los números negativos se representan por el **complemento a 2** de su magnitud.
- **No** se reserva bit para el signo.
- Puede representar solo enteros comprendidos entre el  $-(2^{n-1})$  y el  $+(2^{n-1}-1)$ .
- Los números que empiezan con 1 dentro del módulo son negativos pero el primer bit **NO ES EL SIGNO !**

$b_3$	$b_2$	$b_1$	$b_0$	Ca2
0	1	1	1	+7
0	1	1	0	+6
0	1	0	1	+5
0	1	0	0	+4
0	0	1	1	+3
0	0	1	0	+2
0	0	0	1	+1
0	0	0	0	0
1	1	1	1	-1
1	1	1	0	-2
1	1	0	1	-3
1	1	0	0	-4
1	0	1	1	-5
1	0	1	0	-6
1	0	0	1	-7
1	0	0	0	-8

- **Complemento a 2** (complemento a la base):
- Se calcula para un módulo determinado como el valor del módulo menos el valor del número al que se le desea hallar el complemento.

Ej.: para mód.16  $\rightarrow$  Ca2 de 6  $\rightarrow$  10 o sea  $16-6=10$   
En este caso 10 es el complemento a 2 del número 6 en la base 16.

Forma gráfica de obtener el complemento a 2:

0110  $\rightarrow$  6  
 $\leftarrow$   
1010  $\rightarrow$  10

se repiten los símbolos de derecha a izquierda hasta encontrar el primer 1, luego se continúa pero invirtiéndolos.



## ● Codificación en Binario Desplazado:

- Representa los números enteros a partir del valor 7 (Exceso a 7)
- Posee representación única del 0.
- Se puede sumar a cada número a ser representado el valor  $2^{n-1}$ , el resultado de dicha suma no será negativo en ningún caso y será siempre una magnitud, la que podrá ser representada como tal.
- Este *desplazamiento* de los número en el valor 7 es lo que da origen a la denominación de este código.

$b_3$	$b_2$	$b_1$	$b_0$	BD
0	0	0	0	-7
0	0	0	1	-6
0	0	1	0	-5
0	0	1	1	-4
0	1	0	0	-3
0	1	0	1	-2
0	1	1	0	-1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	2
1	0	1	0	3
1	0	1	1	4
1	1	0	0	5
1	1	0	1	6
1	1	1	0	7
1	1	1	1	8

## ● Codificación de números reales binarios:

- Se suelen codificar en una forma denominada **coma flotante** (**punto flotante** o **notación científica**).
- Representar los números de la siguiente manera:

$$\pm S M * B^{\pm \text{exp}}$$

donde:

**S** es el signo del número, es decir + o –

**M** es la llamada **mantisa** del número. La forma correcta es parte significativa del número. En ingles **Significand**.

**B** es la **base** del sistema de numeración empleado.

**exp** es el **exponente**.

**M\*B<sup>exp</sup>** es la magnitud del número.

## ● Codificación de números reales base 10 en notación científica:

- Muy utilizada en cálculos científicos.

$$\text{Ej.: } -745,38 = -74538 \cdot 10^{-2}$$

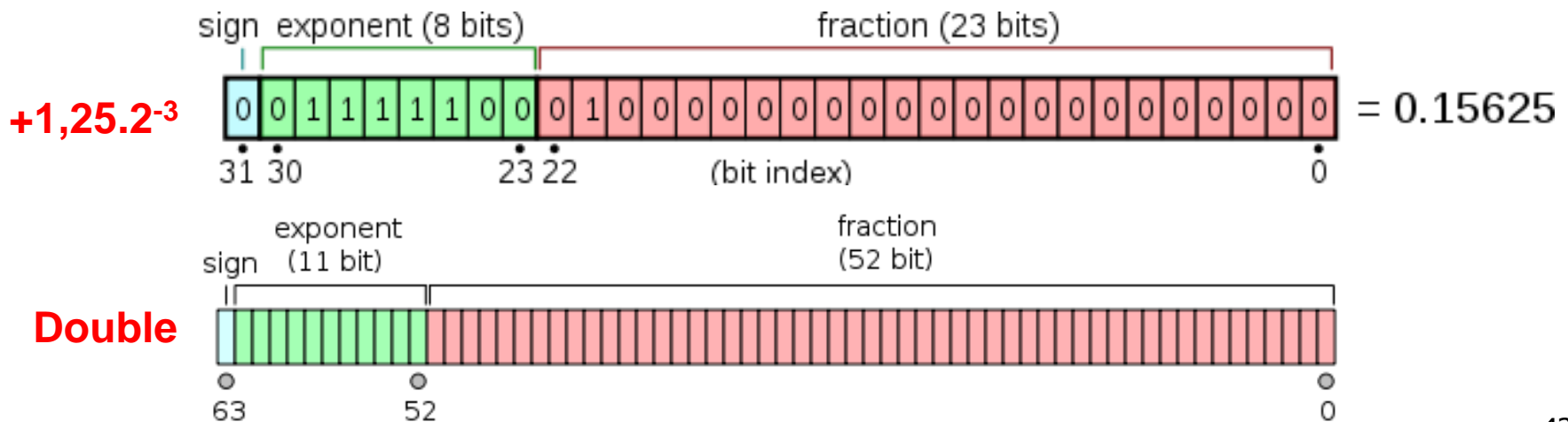
- Esta representación **no es única**.

$$\text{Ej.: } -745,38 = -7,4538 \cdot 10^2 = -0,74538 \cdot 10^3$$

- No es bueno tener múltiples representaciones, se usa solo una, llamada ***normalizada (IEEE 754 – 1985, 2008)***.
- Las representaciones normalizadas suelen ser, según el código empleado, aquella en que la mantisa es **totalmente fraccionaria** (como en  $-0,74538 \cdot 10^3$ ) o aquella en que sólo tiene **un dígito entero** (como en  $-7,4538 \cdot 10^2$ ).

## ● Codificación de números reales binarios:

- Los números, reales que se utilizan normalmente en los sistemas electrónicos son los binarios, y no los decimales, y suelen ser representados por conjuntos de 32 bits (simple) ó 64 bits (double)
- Este número de bits se reparte entre los tres elementos á representar, es decir: **signo**, **mantisa** (*significand*) y **exponente**, ya que la base es *implícitamente* 2.
- Esta repartición se denomina *división en campos*, y suele hacerse de la siguiente forma:

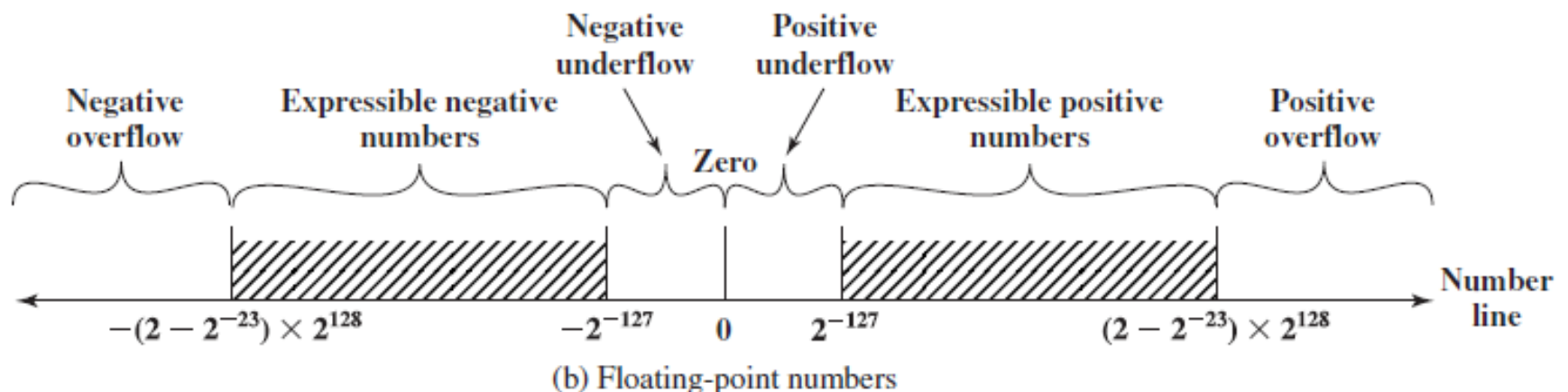
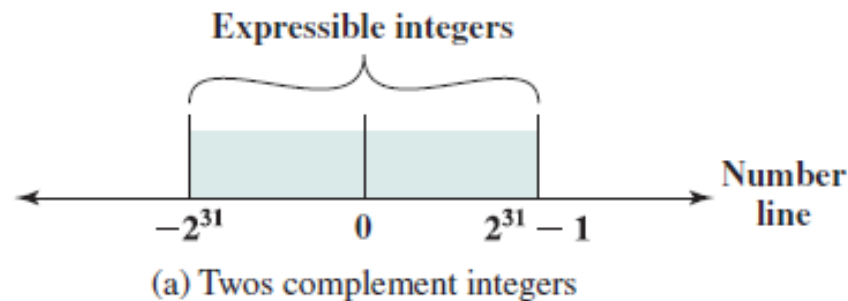


## ● Codificación de números reales binarios: FP- IEEE-754

- La mantisa (significand) se guarda en **binario <1**
- El exponente se utiliza en **código exceso a 127**
- Exceso a 127 significa
  - 8 bits en el campo del exponente.
  - Valor del rango puro entre 0-255
  - Restando 127 se obtiene el valor correcto.
  - Rango entre -126 a +127
- Para un número de 32 bits
  - 1 bit de signo.
  - 8 bits para el exponente.
  - 1 bit implícito que no se graba de valor 1
  - 23 bits de parte significativa (24 bits en total para 1+ la fracción).

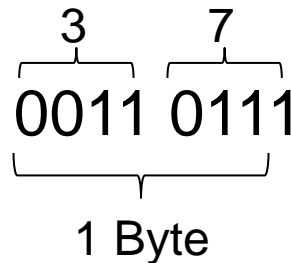
Descripción válida para 32 bits (Simple precisión) Normalizado

## ● Codificación de números reales binarios:








## ● Codificación Binaria para números Decimales:

- Los códigos para números decimales codificados en binario (códigos BCD o **B**inary **C**oded **D**ecimal) se basan en representar por separado en un cierto código binario a los diferentes dígitos que componen un número decimal.
- Ej.: el número decimal 37 se representa como decimal por dos combinaciones de dígitos binarios:
- la que represente al 3 y la que represente al 7.





## Codificación Binaria para números Decimales:

-  Existen varias formas de representar a los dígitos decimales mediante un código de alfabeto binario.
-  Todas ellas requieren un mínimo de 4 bits para ello, ya que 3 bits resultan insuficientes pues sólo pueden codificar a 8 elementos.
-  Los códigos BCD más usuales son los siguientes:
  -  BCD Natural
  -  BCD 7 segmentos

## ● Codificación Binaria para base 10:

### BCD Natural:

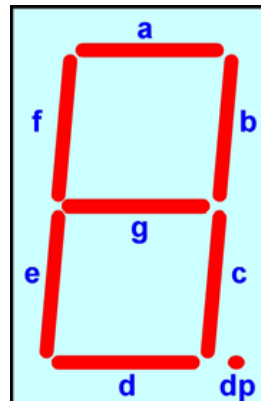
- En este código, cada uno de los diez dígitos decimales se representa directa y naturalmente por su número binario correspondiente expresado con 4 bits.
- Tiene todas las características del binario natural.
- Las combinaciones superiores no se utilizan. En algunos casos se utiliza para signo y separador decimal

$b_3$	$b_2$	$b_1$	$b_0$	Dec.
8	4	2	1	
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

## ● Codificación Binaria para base 10

### 7 Segmentos:

- Se vincula a los exhibidores de 7 segmentos con que habitualmente se exhiben los dígitos decimales en relojes digitales, calculadoras, etc.
- Este código no es pesado, ni continuo.



a	b	c	d	e	f	g	Dec.
1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	2
1	1	1	1	0	0	1	3
0	1	1	0	0	1	1	4
1	0	1	1	0	1	1	5
X	0	1	1	1	1	1	6
1	1	1	0	0	X	0	7
1	1	1	1	1	1	1	8
1	1	1	X	0	1	1	9

## ● Códigos Detectores y/o Correctores de Errores: Códigos Detectores de Errores:

- Ciertos códigos tienen la capacidad de permitir reconocer si es que en una palabra del mismo se ha introducido un error.
- La idea general que utilizan es hacer que los errores que pudieran llegar a producirse en una palabra del código la transformen en una **combinación inválida** del mismo, lo que denotaría la presencia del error.
- Estos códigos se denominan códigos detectores de errores o, también, códigos auto verificadores.

## ● Códigos Detectores de Errores:

### ● Códigos de paridad:

- Permiten detectar la aparición de hasta 1 error en una palabra del mismo.
- Estos se caracterizan en que todas las palabras de estos códigos tienen un número par (o impar) de unos.
- Cualquier código se puede convertir en código de paridad.
- Esto permite, al procesar una palabra, reconocer si se ha producido un error en un bit en la operación. Tanto si un 0 fue erróneamente interpretado como un 1, o si un 1 fue interpretado como un 0, la cantidad de unos de la palabra recibida o leída sería impar (o par), y por lo tanto evidentemente incorrecta.

## Códigos Detectores de Errores: Códigos de paridad:

Ej.: Se desea agregar un bit de paridad par al código ASCII; dicho bit adicional se colocará continuación de los 7 bits habituales del código:

- a) la letra M
- b) la letra m

Solución procediendo caso por caso:      M → 1001101

Esta palabra tiene un número par de unos, por lo que el bit de paridad a agregar es un 0, y la palabra queda ahora:      M → 1001101**0**

A su vez:      m → 1011101

Esta palabra tiene un número impar de unos, por lo que el bit de paridad a agregar es un 1, y la palabra queda ahora:

m → 1011101**1**

**NOTA:** El bit de paridad se colocó a continuación, pero vale aclarar que en este código en particular es más frecuente que el bit de paridad se coloque adelante.

*¿Preguntas?*