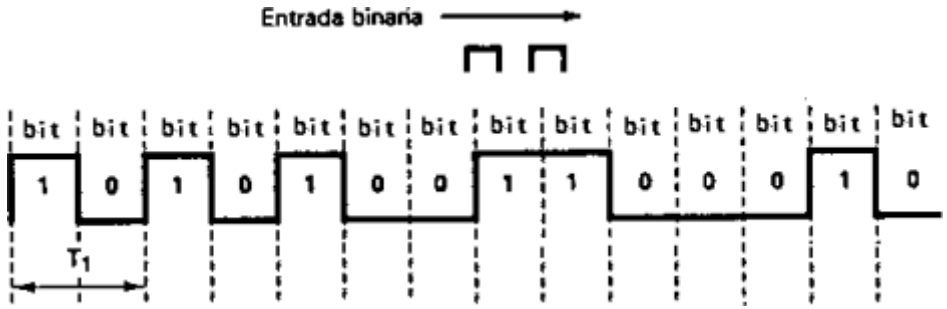
1. Codificación de la información en diferentes sistemas de representación

➢ 2.1. Sistemas de representación

Toda información que maneja el ordenador se representa mediante dos símbolos (0 y 1), los cuales corresponden a dos estados eléctricos, es decir, a los dos niveles de tensión que pueden llegar a tomar.



En definitiva, la información se mantiene utilizando dos valores de una magnitud física representable mediante ceros y unos.

Un **bit** es la unidad mínima de información. Puede tener dos valores, que serán sus estados. Puede estar *apagado* (su valor es 0) o *encendido* (su valor es 1).

A partir de estos símbolos básicos, el ordenador es capaz de construir, almacenar y representar distintos

tipos de información, pues puede codificarla. Además, hay distintos tipos de representaciones:

- Representación de **textos**
- Representación de **valores numéricos**
- Representación de **instrucciones**
- Representación de **sonidos**
- Representación de **imágenes y vídeos**

Para almacenar estos tipos de documentos, es necesario codificarlos, lo cual se hace mediante la transformación de la información al **sistema binario**.

En este tema, se van a ver los sistemas de codificación existentes y cómo se puede trabajar con ellos.

La **codificación** es la operación que permite convertir los datos de un sistema de información a otro.

➢ 2.2. Medidas de información

Palabra **Byte**

Además de los bits, en la información digital existen diferentes tipos de unidades de información:

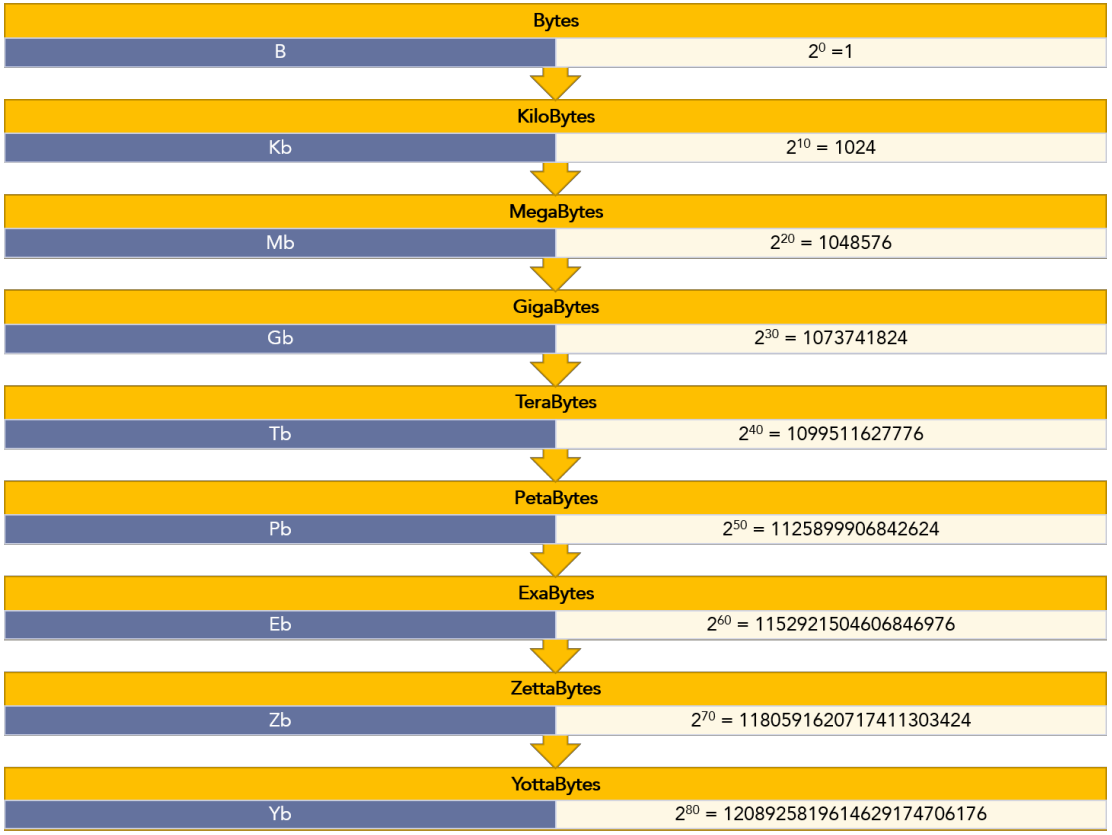
- **Palabra**: conjunto de *n* bits. La longitud de una palabra hace referencia al número de bits contenidos en ella. Además, su tamaño puede variar, pero en los ordenadores modernos suelen tener una longitud de **16, 32** o **64 bits**.
- **Byte**: conjunto de 8 bits.

Al almacenar la información
tenemos que tener en cuenta **dos factores**:

Unidad de medida **Capacidad de almacenamiento**

- **La unidad de medida**: los datos pueden ser de gran tamaño, por lo que, para simplificar su valor, se ha establecido una escala con diferentes unidades de medida.

- **La capacidad de almacenamiento**: se refiere a la cantidad de datos que pueden almacenarse en un dispositivo. Dependiendo del dispositivo, las unidades de medida más utilizadas pueden ser *megabytes* o *gigabytes*.



Dependiendo del tipo de información que vayamos a almacenar, podemos utilizar una u otra unidad. Si queremos medir el tamaño de un fichero de texto usaremos los kilobytes, *mientras que, si queremos medir el tamaño de una canción en formato MP3, utilizaremos los* megabytes.

➢ 2.3. Sistemas de codificación alfanumérica

ASCII | **EBCDIC (Unicode)**

Para la representación de los datos no numéricos o alfanuméricos se emplean códigos como el **ASCII**, el **EBCDIC** o el **Unicode**.

ASCII

El **código ASCII** es el más usado entre los sistemas informáticos actuales.

El **ASCII se utiliza para representar caracteres**. Se trata de un código estándar, independiente del lenguaje que usemos y de la computadora utilizada. Además, está formado por **8 bits**, de manera que cada carácter se expresa por un número comprendido entre 0 y 255.

Por otra parte, cabe mencionar que la información se guarda en 7 bits y el octavo se reserva para comprobar la paridad y prevenir errores.

En este sistema podemos distinguir **dos grupos**: los primeros 128 caracteres se denominan código **ASCII estándar** y representan los caracteres que aparecen en una máquina de escribir convencional. De estos, los primeros 32 son caracteres de control. Este tipo se refiere a aquellos códigos que no representan información imprimible. Por otro lado, los 128 restantes se denominan código **ASCII ampliado**, que son asociados a un número de caracteres que no aparecen en la máquina de escribir y que son muy

utilizados en la computadora, como pueden ser operadores matemáticos o caracteres gráficos.

Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

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

ASCII extendido							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

EBCDIC
(Unicode)

El **sistema de codificación Unicode** es un sistema de 16 bits que se utiliza en otros sistemas de escritura como, por ejemplo, el árabe, el griego o el japonés, entre otros. Esto se debe a su mayor capacidad con respecto a otros sistemas.

➢ 2.4. Sistemas de codificación numérica

DECIMAL | BINARIO | HEXADECIMAL | OCTAL

Un sistema de numeración es el **conjunto de símbolos y reglas** que se utilizan para representar datos numéricos.

Los sistemas más comunes son:

DECIMAL

- **Decimal**: se compone de 10 dígitos, por lo que decimos que tiene **base 10**. Su conjunto de dígitos va desde el 0 hasta el 9. D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  Es el sistema que utilizamos en la vida cotidiana.

BINARIO

- **Binario**: se compone únicamente de 2 dígitos, por lo que su base es 2.
- Los dígitos que la componen son 0 y 1.
  D = {0, 1}
  Como se ha comentado anteriormente, es el sistema que utilizan internamente los ordenadores.

- **HEXADECIMAL**
- **Hexadecimal**: se compone de 16 dígitos, donde se combinan números y letras. **Su base es 16**.
-

  D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

  Es el sistema de numeración que suelen utilizar las CPU.

OCTAL

- **Octal**: se compone de 8 dígitos, cuya numeración va del 0 al 7. **Su base es 8**.
-

  D = {0, 1, 2, 3, 4, 5, 6, 7}

  En algunos casos, se utiliza el sistema octal en vez del hexadecimal. Esto se debe a la ventaja de no tener que codificar letras. Para poder utilizar este sistema en vez del hexadecimal, es necesario añadir delante el prefijo 0x.

A(16 = 0x12(8

➢ 2.5. Conversión entre diferentes sistemas de numeración

En este apartamos, se van a explicar los diferentes métodos que tenemos para la conversión de los datos.

PASO DE DECIMAL A:
• Paso de decimal a cualquier otra base
• Paso de decimal a binario
• Paso de decimal a octal
• Paso de decimal a Hexadecimal

PASO de:
• Paso cualquier base a decimal
• Paso de binario a octal o hexadecimal
• Paso de octal a hexadecimal y viceversa
• Paso de hexadecimal a octal

➢ Sistemas de codificación numérica

CONVERSION **DECIMAL** a

Se compone de **10 dígitos**,
D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
Es el sistema que utilizamos en la vida cotidiana (10 digitos) (10 dedos)

OPERACIONES

Compuesto por	Representación: • Numérica • Alfanumérica	BASE	Codificación	Parte Entera	Parte Decimal
Se compone de **2 dígitos** (Los dígitos que la componen son 0 y 1).	D = **{0, 1}**	2 10101010_2	**BINARIO** Es el sistema que utilizan internamente los ordenadores.	(Pe)/2	(Pd)*2
Se compone de **8 dígitos** (Cuya numeración va del 0 al 7).	D = **{0, 1, 2, 3, 4, 5, 6, 7}**	8 $0x12_8$	**OCTAL** Es necesario añadir delante el prefijo **0x**.	(Pe)/8	(Pd)*8
Se compone de **16 dígitos** (Donde se combinan números y letras).	D = **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**	16 A_{16}	**HEXADECIMA** Es el sistema de numeración que suelen utilizar las CPU.	(Pe)/16	(Pd)*16

Pasar de → a DECIMAL

	BASE	Compuesto por	Representación • Numérica • Alfanumérica	Parte Entera	Parte Decimal
		Se compone de **10 dígitos**, D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} Es el sistema que utilizamos en la vida cotidiana (10 digitos) (10 dedos)		Multiplicar la parte decimal por la base a la que se va a transformar el número hasta que no haya parte decimal. Después, se escogen las cifras de la parte entera que resultan de cada operación.	
BINARIO Es el sistema que utilizan internamente los ordenadores.	2 10101010_2	Se compone de **2 dígitos.** (Los dígitos que la componen son 0 y 1).	D = **{0, 1}**	ejemplo: $10101010_2 = ?_{10}$ $1\cdot2^8+0\cdot2^7+1\cdot2^6+0\cdot2^5+1\cdot2^4+0\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0$ (Pe.)↑**2**	ejemplo: $0{,}101_2 = ?_{10}$ $1\cdot2^{-1}+0\cdot2^{-2}+1\cdot2^{-3}$ (.Pd) (-↑)2
OCTAL Es necesario añadir delante el prefijo 0x.	8 $0x12_8$	Se compone de **8 dígitos** (Cuya numeración va del 0 al 7).	D = **{0, 1, 2, 3, 4, 5, 6, 7}**	(Pe.) ↑8	(.Pd) (-↑)8
HEXADECIMAL Es el sistema de numeración que suelen utilizar las CPU.	16 A_{16}	Se compone de **16 dígitos** (Donde se combinan números y letras).	D = **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**	(Pe.) ↑16	(.Pd) (-↑)16

- # Paso de decimal a cualquier otra base

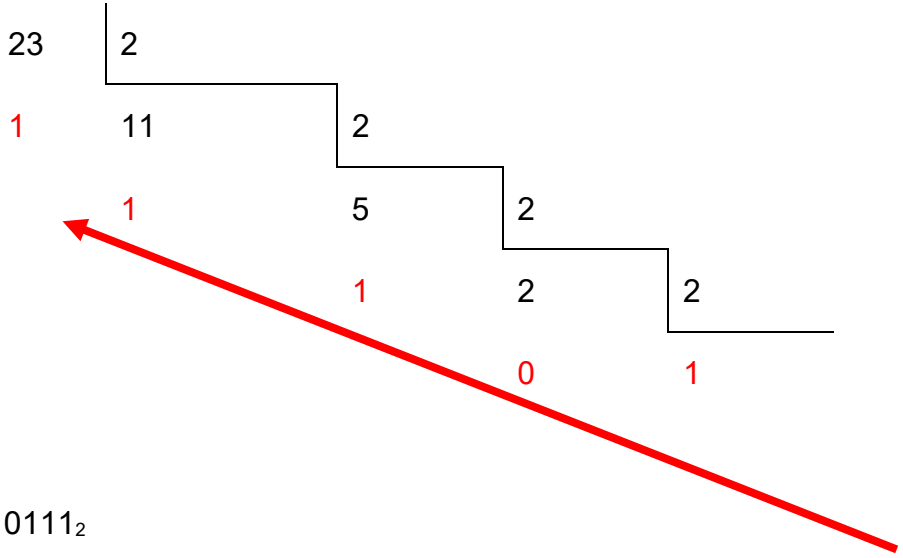
Para realizar este tipo de conversiones, es necesario separar la parte decimal de la parte entera del número, pues la operación se lleva a cabo de distinta forma.

- **Parte entera**: dividimos el número original (en decimal) entre la base del sistema de numeración de destino, hasta obtener un resto menor que el divisor, es decir, sin decimales. Después, formaremos el número en binario con el último cociente y los restos obtenidos.

- **Parte decimal**: al contrario que en la parte entera, aquí multiplicamos los dígitos. El procedimiento consiste en multiplicar la parte decimal por la base a la que se va a transformar el número hasta que no haya parte decimal. Después, se escogen las cifras de la parte entera que resultan de cada operación.

1.1.1. Paso de decimal a binario

- **Parte entera**: ejemplo: $23_{10} = ?_2$



$23_{10} = 10111_2$

- **Parte decimal**: ejemplo: $0{,}375_{10} = ?_2$

$0{,}375 \cdot 2 = 0{,}75$

$0{,}75 \cdot 2 = 1{,}5$

$0{,}5 \cdot 2 = 1{,}0$

$0{,}0 \cdot 2 = 0{,}0$

$0{,}375_{10} = 0{,}011_2$

1.1.2. Paso de decimal a octal

- **Parte entera**: ejemplo: $23_{10} = ?_8$

$23_{10} = 27_8$

- **Parte decimal**: ejemplo: $0{,}625_{10} = ?_8$

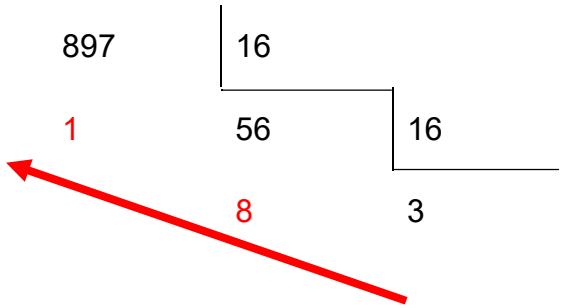
$0{,}625 \cdot 8 = 5{,}0$

$0{,}675_{10} = 0{,}5_8$

23	8
7	2

1.1.3. Paso de decimal a hexadecimal

- **Parte entera**: ejemplo: $897_{10} = ?_{16}$



$897_{10} = 381_{16}$

- **Parte decimal**: ejemplo: $0{,}625_{10} = ?_{16}$

$0{,}625 \cdot 16 = 10{,}0$

$0{,}675_{10} = 0{,}A_{16}$

1.1.4. Paso de cualquier base a decimal

Para realizar la conversión a decimal se debe multiplicar cada dígito por su base elevada a la posición en la que se encuentra. Para obtener la posición, se debe diferenciar la parte entera de la decimal:

- La **parte entera** comienza en la posición 0 en el dígito al lado de la coma.
- La **parte decimal** comienza en la posición -1 en el dígito al lado de la coma.

Paso de binario a decimal

- **Parte entera**: ejemplo: $10101010_{12} = ?_{10}\ 1\cdot2^8+0\cdot2^7+1\cdot2^6+0\cdot2^5+1\cdot2^4+0\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0$

  $1\cdot256+0\cdot128+1\cdot64+0\cdot32+1\cdot16+0\cdot8+1\cdot4+0\cdot2+1\cdot1$

  $101010101_2 = 341_{10}$

- **Parte decimal**: ejemplo: $0{,}101_2 = ?_{10}\ 1\cdot2^{-1}+0\cdot2^{-2}+1\cdot2^{-3}$

  $1\cdot0{,}5+0\cdot0{,}25+1\cdot0{,}125$

  $0.101_2 = 0{,}625_{10}$

1.1.5. Paso de octal a decimal

- **Parte entera**: ejemplo: $123_8 = ?_{10}$

  $1 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0$

  $1 \cdot 64 + 2 \cdot 8 + 3 \cdot 1$

  $123_8 = 83_{10}$

- **Parte decimal**: ejemplo: $0{,}458 = ?_{10}$

  $4 \cdot 8^{-1} + 5 \cdot 8^{-2}$

  $4 \cdot 0{,}125 + 5 \cdot 0{,}015625$

  $0{,}458 = 0{,}578125_{10}$

1.1.6. Paso de hexadecimal a decimal

- **Parte entera**: ejemplo: $A00116 = ?10$

  $A \cdot 16^3+0 \cdot 16^2+0 \cdot 16^1+1 \cdot 16^0$

  $10 \cdot 4096+0*256+0 \cdot 16+1*1$

  $A00116 = 409611_{10}$

- **Parte decimal**: ejemplo: $0,0716 = ?10$ $0 \cdot 16^{-}$

  $1+7 \cdot 16^{-2}$

  $0 \cdot 0,0625+7 \cdot 0,00390625$

  $0,0716 = 0,027343751_{0}$

➢ Sistemas de codificación numérica

CONVERSIÓN

8 **OCTAL**

$0x12_8$

a

Se compone de **08 dígitos**,
D = **{0, 1, 2, 3, 4, 5, 6, 7}**
Añadir delante el prefijo **0x**.

OPERACIONES

Compuesto por	Representación: • Numérica • Alfanumérica	BASE	Codificación	Parte Entera	Parte Decimal
Se compone de **2 dígitos** (Los dígitos que la componen son 0 y 1).	D = **{0, 1}**	2 10101010_2	**BINARIO** Es el sistema que utilizan internamente los ordenadores.	Grupos de 3 Según Tabla de Conversión	Grupos de 3 Según Tabla de Conversión
Se compone de **16 dígitos** (Donde se combinan números y letras).	D = **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**	16 A_{16}	**HEXADECIMA** Es el sistema de numeración que suelen utilizar las CPU.	Grupos de 4 Según Tabla de Conversión	Grupos de 4 Según Tabla de Conversión

➢ Sistemas de codificación numérica

2

10101010_2

CONVERSIÓN a **BINARIO**

Se compone de **2 dígitos**
(Los dígitos que la componen son 0 y 1).
D = **{0, 1}**
Es el sistema que utilizan internamente los ordenadores.

OPERACIONES

Compuesto por	Representación: • Numérica • Alfanumérica	BASE	Codificación	Parte Entera	Parte Decimal
OCTAL Se compone de **08 dígitos**, $0x12_8$	D = **{0, 1, 2, 3, 4, 5, 6, 7}**	8	Añadir delante el prefijo **0x**	Conversión **DIGITO A DÍGITO** Grupos de 3 Según Tabla de Conversión	Conversión **DIGITO A DÍGITO** Grupos de 3 Según Tabla de Conversión
HEXADECIMA Se compone de **16 dígitos** (Donde se combinan números y letras).	D = **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**	16 A_{16}	Es el sistema de numeración que suelen utilizar las CPU.	Conversión **DIGITO A DÍGITO** Grupos de 4 Según Tabla de Conversión	Conversión **DIGITO A DÍGITO** Grupos de 4 Según Tabla de Conversión

1.1.1. Paso de binario a octal o hexadecimal

Paso de binario a octal

Para realizar la conversión de binario a octal, debemos realizar grupos de 3, teniendo como referencia la coma. Cada uno de estos grupos de dígitos es un dígito en octal, por lo que se debe realizar dicha conversión.

Binario	Octal
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Ejemplo: $01011100,110012 = ?8$

001 011 10 , 110 01
0 0
1 3 4 , 6 2

$01011100,110012 = 134,628$

Paso de binario a hexadecimal

Para realizar la conversión de binario a hexadecimal, debemos realizar grupos de 4, teniendo como referencia la coma. Cada uno de estos grupos de dígitos es un dígito en hexadecimal, por lo que se debe realizar dicha conversión.

Binario	Hexadecimal
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Ejemplo: $11101100010101,11001_2 = ?_{16}$

0011 1011 0001 0101 , 1100 1000
3 B 1 5 , C 8

$01011100,11001_2 = 3B15,C4_{16}$

1.1.2. Paso de octal o hexadecimal a binario

Paso de octal a binario

Hay que realizar la conversión de todos los dígitos a binario. Cada uno de ellos se transformará en 3 dígitos binarios.

Ejemplo: $234{,}628 = ?2$

2	3	4	,	6	2
01	01	10	,	11	01
0	1	0		0	0

$234{,}628 = 010011100{,}110010_2$

Paso de hexadecimal a binario

Hay que realizar la conversión de todos los dígitos a binario. Cada uno de ellos se transformará en 4 dígitos binarios.

Ejemplo: $ABD30{,}C116 = ?2$

A	B	D	3	0	,	C	1
1010	1011	1101	0011	0000	,	1100	0001

$ABD30{,}C1 = 10101011110100110000{,}11000001_2$

1.1.3. Paso de octal a hexadecimal y vicecersa

Paso de octal a hexadecimal

Para realizar la conversión de octal a hexadecimal se debe hacer un paso intermedio, es decir, primero hay que transformar el número a binario para después convertirlo a hexadecimal.

Ejemplo: $3C6_{16} = ?_8$

1. **Pasar** a binario:

3	C	6
0011	1100	0110

2. **Agrupar** de tres en tres:

00	11	00	11
1	1	0	0
1	7	0	6

$3C6_{16} = 1706_8$

1.1.4. Paso de hexadecimal a octal

Para realizar la conversión de hexadecimal a octal se debe realizar un paso intermedio, es decir, primero se transforma el número a binario y después se realiza la conversión a octal.

Ejemplo: $17068 = ?16$

1. **Pasar** a binario:

1	7	0	6
00	11	000	110
1	1		

2. **Agrupar** de cuatro en cuatro:

0011	1100	0110
3	C	6

$17068 = 3C616$

>2.2. Aritmética binaria

Adicción + | **Sustración -**

Multiplicación * | **División /**

La aritmética binaria es el **conjunto de operaciones aritméticas y lógicas** que se realizan con variables representadas en el sistema binario.

Operaciones aritméticas

- **Adición (+)**:

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	0*

* Además, colocamos un 1 en la posición inmediata superior (me llevo una); es el digito de arrastres.

Ejemplo: suma los números binarios 11111,100 y 1001000,011

```
        1  1
     1  1  1  1  1  ,  1  0  0
+    1  0  0  1  0  0  0  ,  0  1  1
     1  1  0  0  1  1  1  ,  1  1  1
```

- **Sustracción (-)**:

A	B	A-B
0	0	0
0	1	1*
1	0	1
1	1	0

* Además, colocamos un -1 en la posición inmediata superior (y debo una); es el digito de arrastres.

Ejemplo: resta los números binarios 11111,100 y 1001000,011

-1	-1	-1	-1	-1	-1	-1				
1	0	0	1	0	0	0	,	0	1	1
		1	1	1	1	1	,	1	0	0
0	1	0	1	0	0	0	,	1	1	1

- **Multiplicación (*)**:

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

Ejemplo: multiplica los números binarios 11111,100 y 1001000,011

				1	1	1	1	1	,	1		
				X		1	0	1	,	0	1	
					1	1	1		1	1	1	
				0	0	0	0		0	0		
			1	1	1	1	1		1			
		0	0	0	0	0	0					
	1	1	1	1	1	1						
1	0	1	0	0	1	0	1	,	0	1	1	

- **División (/)**:

A	B	A/B
0	0	Indeterminado
0	1	0
1	0	Infinito
1	1	1

Ejemplo: divide los números binarios 1001 y 11

1001	11

11 | 11

0011

  11

  00

➢ 2.7. Métodos para representar números enteros

Signo y magnitud | **Complemento a 1** | **Complemento a 2**

Exceso a 2n-1

Los ordenadores tienen la necesidad de almacenar números y, al trabajar en binario, deben usar algún método para representar los números enteros, los positivos y los negativos.

En la actualidad, existen diferentes técnicas:

- **Signo y magnitud**: el bit que se encuentra más a la izquierda representa el signo. Si su valor es 0, entonces el número es positivo, mientras que si su valor es 1, entonces el número es negativo.

  El resto de bits representa el módulo del número.

  Ejemplo: representar el 12 y -12 en una palabra de 8 bits en signo y módulo:

  ```
  12   0   0001100
   -   1   0001100
  12
  ```

  Una de las desventajas de este método es la doble representación del 0(10). Tendríamos el conjunto 1000000(2) (-0(10)) y el conjunto 00000000(2)
  (+0(10).

- **Complemento a 1**: se realiza una diferenciación entre los números positivos y los números negativos. En los números positivos, el bit que se encuentra más a la izquierda representa el signo y el resto de los bits corresponden al módulo del número. No obstante, en los números negativos se parte del número positivo, pero después se debe cambiar cada uno de los dígitos. De esta forma, todos los ceros pasan a ser unos y viceversa.

  Ejemplo: representar 12 y -12 en una palabra de 8 bits en complemento a1:

  ```
   12   0   0001100
  -12   1   1110011
  ```

  La desventaja es la misma que en la técnica de signo y magnitud, es decir, tendríamos doble representación del $0_{(10)}$. Tendríamos el conjunto $00000000_{(2)}$ ($+0_{(10)}$) y $11111111_{(2)}$ ($-0_{(10)}$).

- **Complemento a 2**: al igual que en el complemento a 1, se realiza una diferenciación entre los números positivos y los números negativos. En los números positivos, el bit que se encuentra más a la izquierda representa el signo y el resto de los bits corresponden al módulo del número. No obstante, en los números negativos, primero se debe realizar la transformación al complemento a 1 y, después, sumar 1 a dicho resultado. Si el ultimo dígito de la suma tiene acarreo, se desprecia.

  Ejemplo: representar 12 y -12 en una palabra de 8 bits en complemento a2:

  Positivo: 12 0 0001100

  Negativo 1 1110011 Primer paso
  + 1 Segundo paso
  -12 1 1110100

  En esta técnica ya no se puede representar $0_{(10)}$ de formas diferentes. Su única representación será $00000000_{(2)}$.

  Ejemplo: representar 0 y -0 en una palabra de 8 bits en complemento a2:

  Positivo: 0 0 0000000

  Negativo 1 1111111 Primer paso
  + 1 Segundo paso
  -0 1|0 0000000

  Se descarta el acarreo (el 1).

- **Exceso a 2n-1**: si se usa este método, no habrá ningún bit para el signo, sino que todos los bits que componen el número tendrán peso en el valor total del mismo.

  Consiste en representar el cero como un valor intermedio, que para *n bits* está representado por 2^{n-1}, y colocar los números negativos antes de ese valor y los positivos después de él. De aquí proviene el nombre.

  Ej.: representar 12 y -12 en una palabra de 8 bits en exceso a 2^{n-1}:

  Para 8 bits el exceso es $2^{8-1} = 2^7 = 128$. Por tanto, para representar los números pedidos tendremos que sumarle esa cantidad:

  12 ➔ 12+128=140 10001100
  -12 ➔ -12+128=116 01110100

  El mayor inconveniente de esta técnica es su complejidad respecto a las otras, puesto que requiere operaciones intermedias. En la actualidad, los ordenadores utilizan la técnica del complemento a 2 para representar los números enteros.

➢ Sistemas de codificación numérica

CONVERSIÓN

8 **OCTAL**

$0x12_8$

a

Se compone de **08 dígitos**,
D = **{0, 1, 2, 3, 4, 5, 6, 7}**
Añadir delante el prefijo **0x**.

OPERACIONES

Compuesto por	Representación: • Numérica • Alfanumérica	BASE	Codificación	Parte Entera	Parte Decimal
Se compone de **2 dígitos** (Los dígitos que la componen son 0 y 1).	D = **{0, 1}**	2 10101010_2	**BINARIO** Es el sistema que utilizan internamente los ordenadores.	Grupos de 3 Según Tabla de Conversión	Grupos de 3 Según Tabla de Conversión
Se compone de **16 dígitos** (Donde se combinan números y letras).	D = **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**	16 A_{16}	**HEXADECIMA** Es el sistema de numeración que suelen utilizar las CPU.	Grupos de 4 Según Tabla de Conversión	Grupos de 4 Según Tabla de Conversión