Tipos de Datos y Representaciones

Circuitos Digitales,

2º de Ingeniero de Telecomunicación.

EITE — ULPGC.

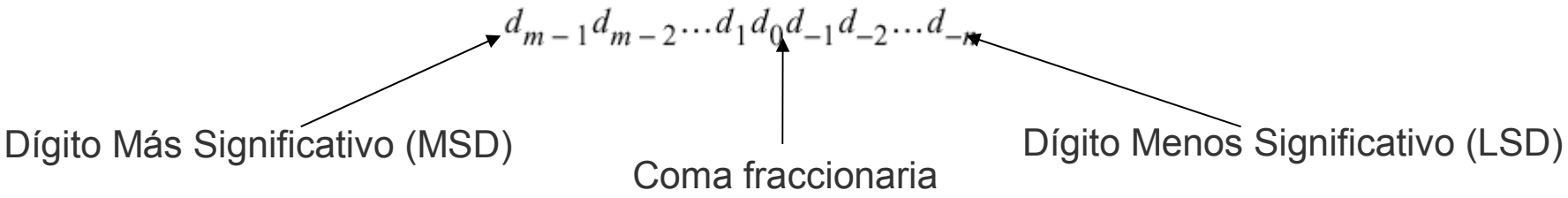
Índice

Objetivos

- Entender qué tipo de información se maneja en los Circuitos Digitales y cómo se representa:
  - Números
  - Caracteres alfanuméricos
  - Representaciones robustas frente a posibles fallos

Sistemas posicionales de numeración

- El sistema de numeración comúnmente empleado, el decimal, es un *sistema de numeración posicional.*
- En general, cualquier número decimal de la forma



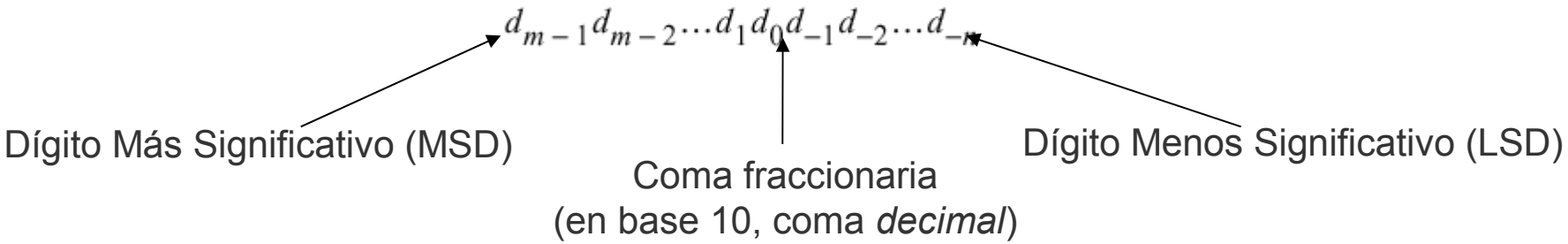
representa:

$$D = d_{m-1}\cdot 10^{m-1} + \ldots + d_0 \cdot 10^0 + d_{-1}\cdot 10^{-1} + \ldots + d_{-n}\cdot 10^{-n} = \sum_{i=-n}^{m-1} d_i \cdot r^i$$

Base (radix)

Sistemas posicionales de numeración

- El sistema de numeración comúnmente empleado, el decimal, es un *sistema de numeración posicional.*
- En general, cualquier número decimal de la forma:

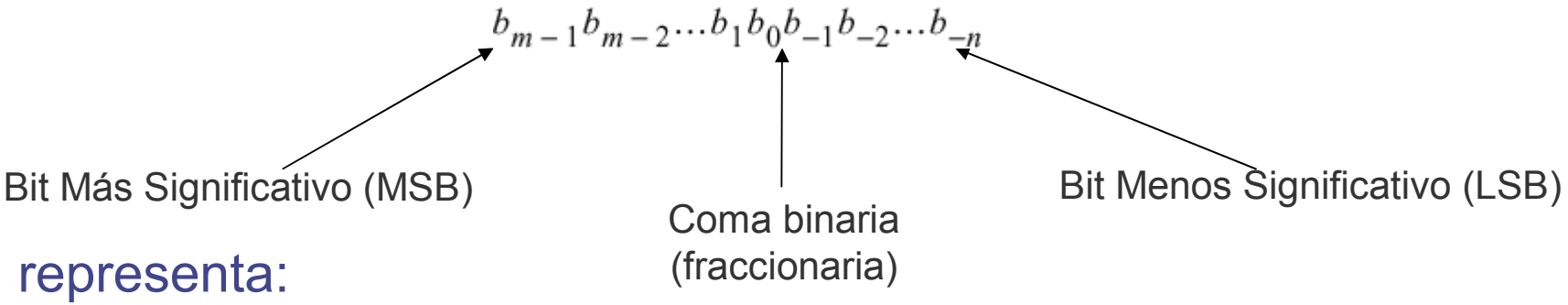


representa:

$$D = \sum_{i=-n}^{m-1} d_i \cdot r^i$$

Sistemas posicionales de numeración

- En los sistemas digitales se emplean sólo dos *símbolos*, que se representan en su diseño con dígitos binarios.
- Por esto, el sistema básico para representar cantidades en un sistema digital es el sistema de base binaria.
- La forma general de un número binario es:



representa:

$$B = \sum_{i=-n}^{m-1} b_i \cdot 2^i$$

¡¡¡ Base 2 (binaria) !!!

Números octales y hexadecimales

- La base 8 y especialmente la 16 se suelen emplear para representar números binarios empleando menos dígitos, evitando el uso de cadenas largas e indescifrables.

Representación	Base	Símbolos
Binaria	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7, 8,9,A,B,C,D,E,F

Números octales y hexadecimales

Binario	Octal	Decimal	Hexadecimal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10
10001	21	17	11

Conversiones entre sistemas numéricos

- El valor de un número en cualquier base viene dado por:

$$D = \sum_{i=-n}^{m-1} d_i \cdot r^i$$

donde *r* es la base de la representación del número, *m* indica el número de dígitos a la izquierda de la coma fraccionaria, y *n* indica el número de dígitos a la derecha.

Conversiones entre sistemas numéricos

- El valor decimal de un número en cualquier base se determina convirtiendo cada dígito del número en su equivalente en base 10, y luego se aplica la fórmula que representa el valor usando aritmética en base 10.

$$D = \sum_{i=-n}^{m-1} d_i \cdot r^i$$

Conversiones entre sistemas numéricos

- El valor decimal de un número en cualquier base se determina convirtiendo cada dígito del número en su equivalente en base 10, y luego se aplica la fórmula que representa el valor usando aritmética en base 10.

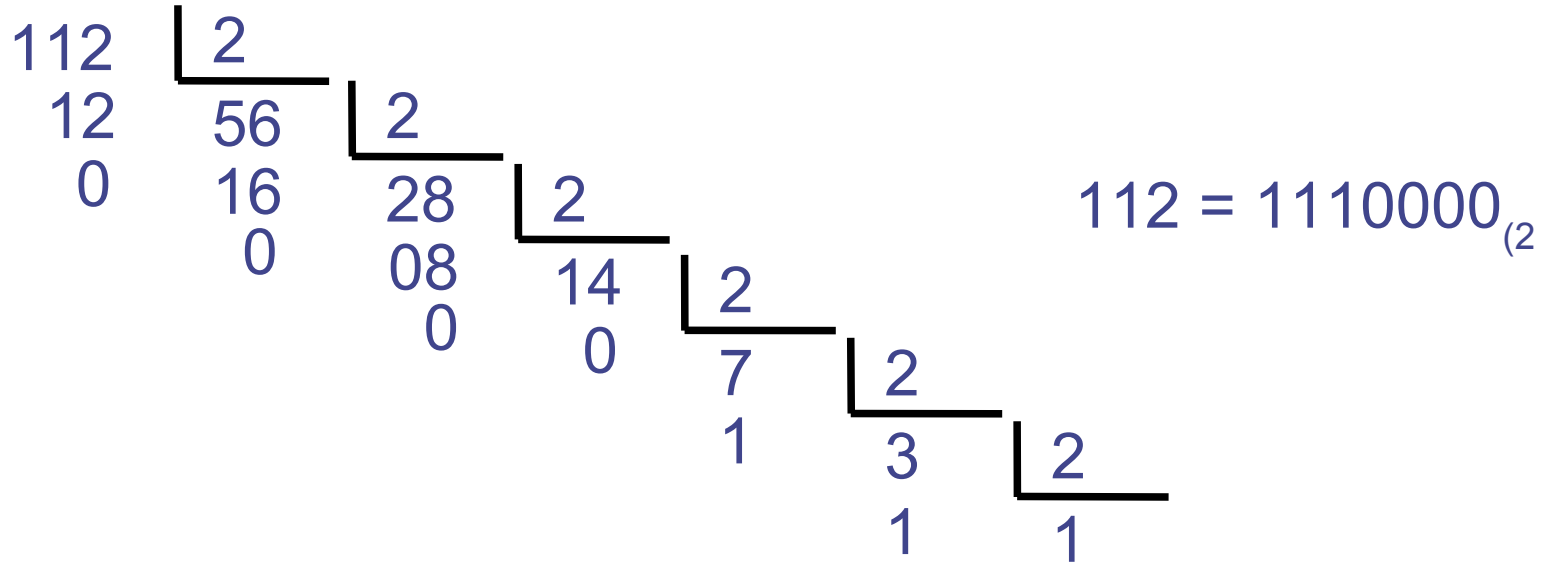
$$D = \sum_{i=-n}^{m-1} d_i \cdot r^i$$

- Hay un método alternativo que se basa en la modificación siguiente de esta expresión:

$$D = \sum_{i=0}^{m-1} d_i r^i = ((\ldots((d_{m-1})r + d_{m-2})r + \ldots)r + d_1)r + d_0$$

Conversiones entre sistemas numéricos

- Para convertir de base decimal a base *r* se emplea el método de la división sucesiva. La división sucesiva por *r* generará el número en base *r* equivalente formado por los restos de las divisiones. El primer resto que se genera es el dígito menos significativo (LSD).



Conversiones entre sistemas numéricos

El procedimiento de las divisiones sucesivas surge de la expresión ya vista:

$$D = \sum_{i=0}^{m-1} d_i r^i = ((\ldots ((d_{m-1}) r + d_{m-2}) r + \ldots) r + d_1) r + d_0$$

Si se divide D por la base r obtenemos d_0 en el resto y el cociente se puede volver a dividir por la base r para después obtener d_1... y así sucesivamente, hasta que el último cociente que quede no se pueda dividir por la base.

Conversiones entre sistemas numéricos

En realidad lo que se ha visto es para números sin parte fraccionaria... ¿Y si tiene parte fraccionaria?

Para la parte fraccionaria, en vez de hacer divisiones sucesivas, se hacen multiplicaciones y vamos *apartando* lo que salga en la parte entera de esas multiplicaciones:

0,6	0,2	0,4	0,8	0,6
x 2	x 2	x 2	x 2	x 2
1,2	0,4	0,8	1,6	1,2

$0,6 = 0,10011_{(2}$

Conversiones de octal a binario y visceversa

Del análisis de la expresión ya vista...

$$D = \sum_{i=0}^{m-1} d_i r^i = (((\ldots((d_{m-1})r + d_{m-2})r + \ldots)r + d_1)r + d_0$$

$$B = (((\cdots((b_{m-1}\cdot 2) + b_{m-2})\cdot 2 + \cdots)\cdot 2 + b_2)\cdot 2 + b_1)\cdot 2 + b_0$$

$$B = ((b_{m-1}\cdot 4 + b_{m-2}\cdot 2 + b_{m-3})\cdot 8 + \cdots)\cdot 8 + (b_2\cdot 4 + b_1\cdot 2 + b_0)$$

Comparando esta expresión con...

$$B = (o_{l-1}\cdot 8 + \cdots + o_1)\cdot 8 + o_0$$

se ve que:

$$o_0 = b_2\cdot 4 + b_1\cdot 2 + b_0$$

$$o_1 = b_5\cdot 4 + b_4\cdot 2 + b_3$$

Para la parte fraccionaria las ecuaciones son similares

Conversiones de hexadecimal a binario y visceversa

Este procedimiento es similar al anterior:

$$h_0 = b_3 \cdot 8 + b_2 \cdot 4 + b_1 \cdot 2 + b_0$$

$$h_1 = b_7 \cdot 8 + b_6 \cdot 4 + b_5 \cdot 2 + b_4$$

Conversión a números octales y hexadecimales

- Conversión de binario a octal: comenzando en la coma binaria, y hacia la izquierda, separar los bits en grupos de 3 y reemplazar por el correspondiente dígito octal.
- Conversión de binario a hexadecimal: comenzando en la coma binaria, y hacia la izquierda, separar los bits en grupos de 4 y reemplazar por el correspondiente dígito hexadecimal.
- Conversión de la parte fraccionaria: comenzando en la coma binaria, separar los bits en grupos de 3 (si es a octal) o 4 (si es a hexadecimal) hacia la derecha y reemplazar.
- Conversión a números binarios: Reemplazar cada octal o dígito hexadecimal con la correspondiente cadena binaria equivalente de cada dígito de 3 bits si el número original es octal o de 4 bits si era hexadecimal.

Suma de números binarios

Para sumar dos números binarios,

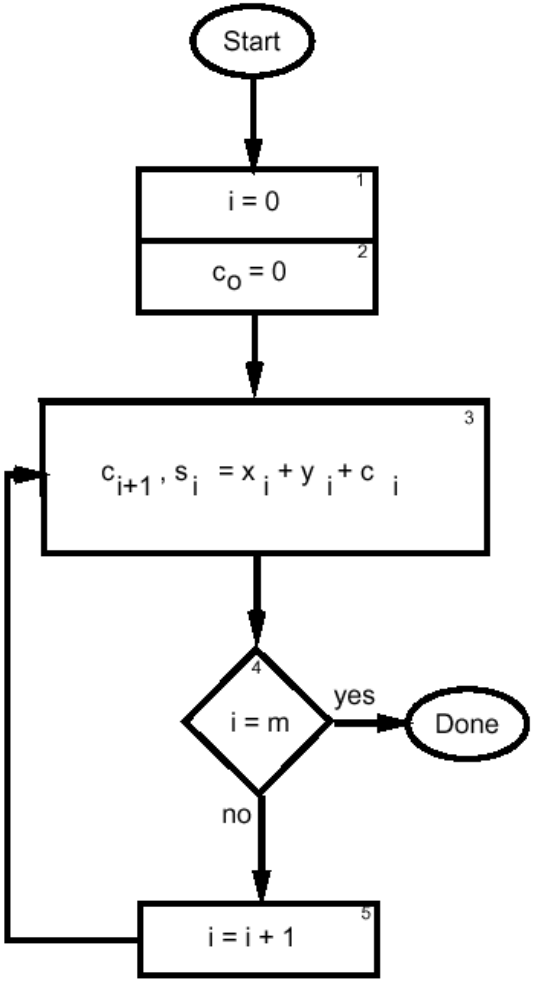
$$x = x_{m-1} \ldots x_0 \quad \text{e} \quad y = y_{m-1} \ldots y_0,$$

se sigue exactamente el mismo procedimiento que el empleado para números decimales. Es decir, se suman los dígitos (en este caso *bits*) de igual peso y se tiene en cuenta lo que "se lleva" (*acarreo*) para la suma de los dígitos de peso siguiente, comenzando este proceso desde los bits de menos peso, los LSBs.

Suma de números binarios

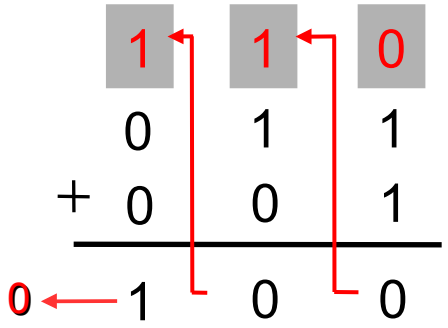
En la suma de los bits en la posición *i*...

$x_i + y_i + c_i$			c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Suma de números binarios

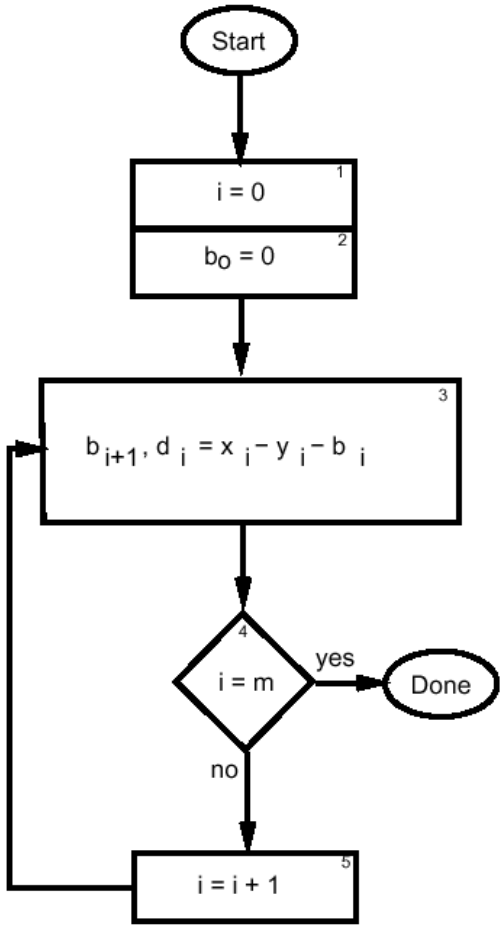
$x_i + y_i + c_i$	c_{i+1}	s_i
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	1	1



Resta de números binarios

- La resta binaria se realiza de forma similar, restando un par de bits cada vez, obteniendo en cada paso un bit de adeudo (borrow) y un bit de resta.

$x_i - y_i - b_i$			b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



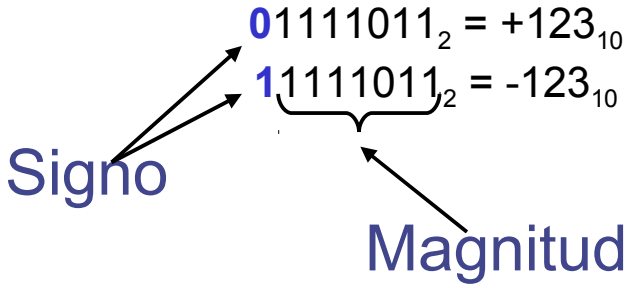
Representación de números negativos

- Los números negativos se pueden representar de varias formas:
  - Representación en signo y magnitud
  - Representación en complemento

Representación de números negativos

Representación en signo y magnitud.

El número está formado por dos partes, la magnitud y el signo, que puede ser + ó -. En los números binarios, el signo se representa mediante un único bit adicional (0 si es positivo y 1 si es negativo) cuya posición es *la que era* del MSB.



Representación de números negativos

Representación en signo y magnitud (cont.)

Al sumar con representación en signo y magnitud requiere que se comparen tanto los signos como las magnitudes de los operandos.

- Si el signo de ambos números es el mismo, se suman las magnitudes y se añade el mismo signo.

- Si los signos son diferentes, hay que comparar las magnitudes:

  - Si son iguales, el resultado es 0
  - Si son distintas, restamos a la magnitud mayor la más pequeña y el resultado hereda el signo de la mayor.

Representación de números negativos

Representación en signo y magnitud (cont.).

•La multiplicación y la división se pueden llevar a cabo realizando sumas y restas iterativas, operando con las magnitudes y dejando el resultado positivo si ambos signos son iguales y haciéndolo negativo si no.

•Hay dos posibles representaciones del cero.

•Un número con representación en signo y magnitud con n bits está dentro del rango $[-(2^{n-1}-1), (2^{n-1}-1)]$.

Representación de números negativos

Sistema de numeración en complemento.

• Se utiliza para realizar las sumas y restas de forma más sencilla sin la necesidad de comparadores.

• En el sistema binario natural, cualquier entero se puede representar por:

$$D = \sum_{i=0}^{m-1} d_i \cdot r^i$$

donde *m* es el número de dígitos y *r* la base.

Representación de números negativos

Sistema de numeración en complemento (cont.).

• En el sistema en complemento a la base *r*, se define el complemento de D con *m* dígitos como:

$$\overline{D} = r^m - D$$

Representación de números negativos

Sistema de numeración en complemento (cont.).

- En el sistema en complemento a la base *r*, se define el complemento de D con *m* dígitos como:

$$\overline{D} = r^m - D$$

- Alternativamente, se puede obtener el complemento a la base de la forma:

$$\bar{D} = D' + 1$$

Representación de números negativos

Radix-complement of a number $D = \sum_{i=0}^{m-1} d_i r_i$ is equal to:

$$\overline{D} = r^m - D = ((r^m - 1) - D) + 1$$

If digit complement $d' = (r-1) - d$ then

$$\begin{aligned}(r^m - 1) - D &= ((r-1)\,(r-1) \ldots (r-1) - (d_{m-1} d_{m-2} \ldots d_0)) \\ &= ((r-1) - d_{m-1})\,((r-1) - d_{m-2}) \ldots ((r-1) - d_0) \\ &= d'_{m-1} d'_{m-2} \ldots d'_0 = \sum_{i=0}^{m-1} d'_i = D'\end{aligned}$$

Representación de números negativos

Sistema de numeración en complemento (cont.).

- La ventaja del sistema en complemento a la base es que los números negativos se pueden representar mediante el complemento $\bar{D}$, ya que $D+\bar{D}=0$ cuando se guardan sólo los *m* bits menos significativos.
- El complemento a la base con números binarios se denomina complemento a 2, y la representación numérica que resulta de esta transformación se denomina representación en complemento a 2.
- Un número negativo se obtiene a partir de un número positivo complementando cada dígito binario, incluyendo el bit de signo, y luego sumándole 1. El acarreo resultante del MSB se descarta.

Representación de números negativos

En el sistema de representación en complemento a 2:

$$B = -b_{m-1} \cdot 2^{m-1} + \sum_{i=0}^{m-2} b_i \cdot 2^i$$

Representación de números negativos

El rango de números representables en complemento a 2 es $[-(2^{m-1}), +2^{m-1}-1]$.

Complementos de los dígitos en sus bases correspondientes

Digit	Binary	Octal	Decimal	Hexa-decimal
0	1	7	9	F
1	0	6	8	E
2	-	5	7	D
3	-	4	6	C
4	-	3	5	B
5	-	2	4	A
6	-	1	3	9
7	-	0	2	8
8	-	-	1	7
9	-	-	0	6
A	-	-	-	5
B	-	-	-	4
C	-	-	-	3
D	-	-	-	2
E	-	-	-	1
F	-		-	0

Representaciones en complemento a 2 y en signo/magnitud

Decimal	Two's Complement	Signed-Magnitude
-8	1000	-
-7	1001	1111
-6	1010	1110
-5	1011	1101
-4	1100	1100
-3	1101	1011
-2	1110	1010
-1	1111	1001
0	0000	1000 or 0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111

Representación de números negativos

A veces, mientras se realizan operaciones aritméticas es necesario pasar de números de *m* bits a representaciones con *n* bits:

- Si $n > m$, se repite $n-m$ veces el bit de signo por la izquierda (extensión de signo).
- Si $n < m$, se hace un truncamiento de signo, descartando los $m-n$ bits que siguen al bit de signo.

  Este número coincide en valor con el original solamente cuando todos los bits descartados son iguales al bit de signo.

Suma y resta en complemento a 2

- La suma se realiza exactamente igual que en el sistema binario natural.
- Existe riesgo de desbordamiento cuando el resultado de la suma debería ser un valor que en realidad no se puede representar con los bits disponibles.

Suma y resta en complemento a 2

$$\begin{array}{r} 0010\ (+2) \\ +\ 0100\ (+4) \\ \hline 0110\ (+6) \end{array} \qquad \begin{array}{r} 1110\ (-2) \\ +\ 1100\ (-4) \\ \hline 1010\ (-6) \end{array}$$

$$\begin{array}{r} 0010\ (+2) \\ +\ 1100\ (-4) \\ \hline 1110\ (-2) \end{array}$$

Suma y resta en complemento a 2

- Como norma, la suma de dos números de distinto signo nunca produce un desbordamiento.

- **Regla sencilla de desbordamiento**: “un desbordamiento en la suma ocurre siempre que el signo de la suma sea diferente a los signos de ambos sumandos”.

- ***Otra* regla de desbordamiento**: “Hay desbordamiento en la suma siempre que el acarreo que va hacia el bit de signo sea diferente del acarreo que produce el bit de signo”.

Suma y resta en complemento a 2

$$\begin{array}{r} 0100\ (+4) \\ +\ 0101\ (+5) \\ \hline 1001\ (-7) \end{array}$$

$$\begin{array}{rr} & 1100\ (-4) \\ +\ & 1011\ (-5) \\ \hline 1 & 0111\ (+7) \end{array}$$

Descartado

Suma y resta en complemento a 2

Para realizar la resta...

• En vez de emplearse una función específica de resta, se emplea la suma pero tomando el sustrendo cambiado de signo, o sea, su complemento a 2.

Multiplicación binaria

El método más común para multiplicar dos números consiste en sumar los productos parciales desplazados que resultan de multiplicar el multiplicando por cada uno de los dígitos del multiplicador.

```
   14   multiplicand
x  13   multiplier
   42   3 x multiplicand
  14    1 x multiplicand
  182   product
```

Con números binarios sin signo...

```
     1110   multiplicand (14)
x    1101   multiplier (13)
     1110
    0000
   1110
  1110
 10110110   product (182)
```

Multiplicación binaria

En los circuitos digitales este proceso se realiza por pasos, realizándose las multiplicaciones por cada dígito y sumando dichos productos desplazados en lo que se llama *producto parcial* que una vez completados los pasos contiene el producto.

	1110	multiplicand (14)
x	1101	multiplier (13)
	0000	first partial product
+	1110	shifted multiplicand
	1110	second partial product
+	0000	shifted zeros
	01110	third partial product
+	1110	shifted multiplicand
	1000110	fourth partial product
+	1110	shifted multiplicand
	10110110	product (182)

En general, multiplicar un número de *n* bits por un número de *m* bits da lugar a un producto de *n*+*m* bits.

Multiplicación binaria

Para multiplicar dos números con representación de signo y magnitud, sencillamente se multiplican sus magnitudes y se le da signo positivo si los operandos tienen el mismo signo y negativo si los signos son distintos.

Con la representación en complemento a 2 las sumas se hacen de forma normal, excepto que se debe tener cuidado cuando en las sumas parciales se suma un dato negativo, en cuyo caso hay que aumentar el número de dígitos con la oportuna extensión de signo.

Multiplicación binaria

Multiplicar (-14) por (-13) en complemento a 2.

```
    1 0 0 1 0
    1 0 0 1 1
  -------------
    0 0 0 0 0 0
    1 1 0 0 1 0
  -------------
  1 1 1 0 0 1 0
  1 1 0 0 1 0
-----------------
  1 1 0 1 0 1 1 0
  0 0 0 0 0 0
-----------------
  1 1 1 0 1 0 1 1 0
  0 0 0 0 0 0
-----------------
  1 1 1 1 0 1 0 1 1 0
  0 0 1 1 1 0
-------------------
  0 0 1 0 1 1 0 1 1 0
```

1. Introducir multiplicando y multiplicador
2. Introducir producto parcial inicial ampliado
3. Multiplicando ampliado
4. Producto parcial ampliado
5. Multiplicando desplazado y ampliado
6. Producto parcial ampliado (ignorar acarreo)
7. Todos ceros
8. Producto parcial ampliado
9. Todos ceros
10. Producto parcial ampliado
11. Multiplicando ampliado, desplazado, negado
12. Producto (182)

División binaria

Los operandos en una división son el *dividendo* y el *divisor* y el resultado es el *cociente*.

$$\frac{dividendo}{divisor} = cociente$$

En los circuitos digitales, la operación de la división se realiza, de forma similar a la multiplicación, con desplazamientos y restas.

Números en coma flotante

- La representación de los datos se deriva de una expresión equivalente al valor del tipo

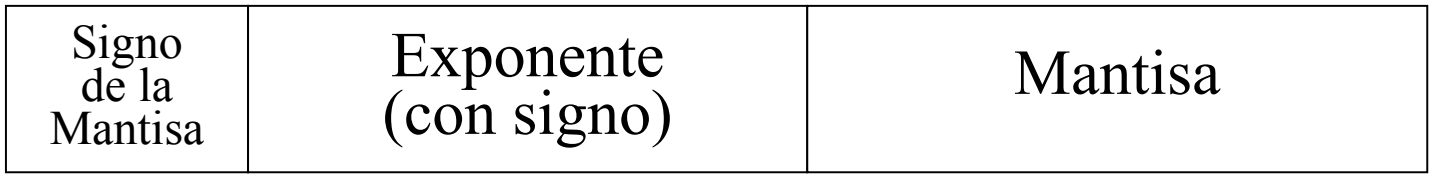
$$\text{mantisa} \times \text{base}^{\text{exponente}}$$

- En esta representación sólo se necesita representar la mantisa y el exponente:
  - *la base es conocida de antemano*.

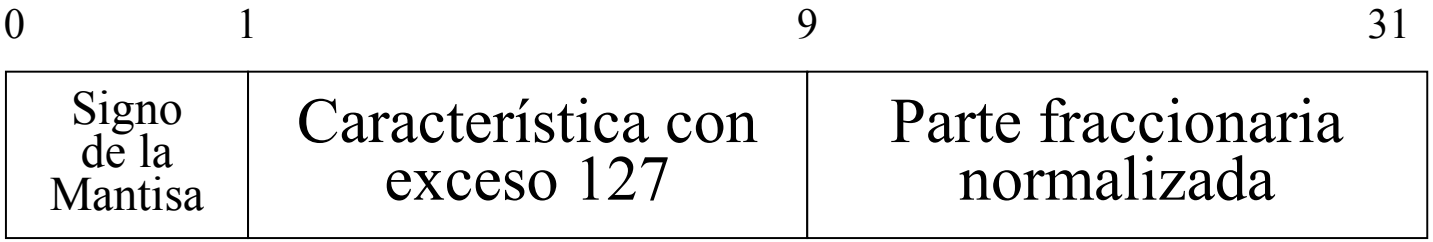
Números en coma flotante

- Al partir de expresiones normalizadas... sólo tienen un dígito en la parte entera distinto de 0
  - El único dígito en la parte entera es siempre el mismo, y por ello no se incluye en la representación
  - El *exponente* da idea de dónde está la coma fraccionaria
    - En realidad se pone el valor del exponente más un valor de exceso (*offset*), llamándose a este valor *característica*.

Números en coma flotante



Formato general



Formato estándar de 32 bits

Formato estándar de 64 bits

Números en coma flotante

- Rango:
  - Intervalo de valores representables
  - Cuantos más bits de la característica, mayor rango
- Precisión:
  - Número de valores representables en un intervalo
  - Cuantos más bits de la mantisa, mayor precisión

Códigos binarios para números decimales

Dígito decimal	BCD (8421)	2421	Exceso 3	Biquinario
0	0000	0000	0011	0100001
1	0001	0001	0100	0100010
2	0010	0010	0101	0100100
3	0011	0011	0110	0101000
4	0100	0100	0111	0110000
5	0101	1011	1000	1000001
6	0110	1100	1001	1000010
7	0111	1101	1010	1000100
8	1000	1110	1011	1001000
9	1001	1111	1100	1010000

Códigos decimales comunes

Códigos de caracteres

- Son realizados con tablas:
  - ASCII
  - EBCDIC
  - Unicode

Códigos de caracteres

Binario	Decimal	Hex	Abreviatura	Repr	AT	Nombre/Significado
0000 0000	0	00	NUL		^@	Caracter Nulo
0000 0001	1	01	SOH		^A	Inicio de Encabezado
0000 0010	2	02	STX		^B	Inicio de Texto
0000 0011	3	03	ETX		^C	Fin de Texto
0000 0100	4	04	EOT		^D	Fin de Transmisión
0000 0101	5	05	ENQ		^E	Enquiry
0000 0110	6	06	ACK		^F	Acknowledgement
0000 0111	7	07	BEL		^G	Timbre
0000 1000	8	08	BS		^H	Retroceso
0000 1001	9	09	HT		^I	Tabulación horizontal
0000 1010	10	0A	LF		^J	Line feed
0000 1011	11	0B	VT		^K	Tabulación Vertical
0000 1100	12	0C	FF		^L	Form feed
0000 1101	13	0D	CR		^M	Carriage return
0000 1110	14	0E	SO		^N	Shift Out
0000 1111	15	0F	SI		^O	Shift In
0001 0000	16	10	DLE		^P	Data Link Escape
0001 0001	17	11	DC1		^Q	Device Control 1 — oft. XON
0001 0010	18	12	DC2		^R	Device Control 2
0001 0011	19	13	DC3		^S	Device Control 3 — oft. XOFF
0001 0100	20	14	DC4		^T	Device Control 4
0001 0101	21	15	NAK		^U	Negative Acknowledgement
0001 0110	22	16	SYN		^V	Synchronous Idle
0001 0111	23	17	ETB		^W	End of Trans. Block
0001 1000	24	18	CAN		^X	Cancel
0001 1001	25	19	EM		^Y	End of Medium
0001 1010	26	1A	SUB		^Z	Substitute
0001 1011	27	1B	ESC		^[or ESC	Escape
0001 1100	28	1C	FS		^\	File Separator
0001 1101	29	1D	GS		^]	Group Separator
0001 1110	30	1E	RS		^^	Record Separator
0001 1111	31	1F	US		^_	Unit Separator
0111 1111	127	7F	DEL		^?, Delete, or Backspace	Delete

Código ASCII (1)

Códigos de caracteres

Binario	Dec	Hex	Representación
0010 0000	32	20	espacio ( )
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Binario	Dec	Hex	Representación
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

Binario	Dec	Hex	Representación
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

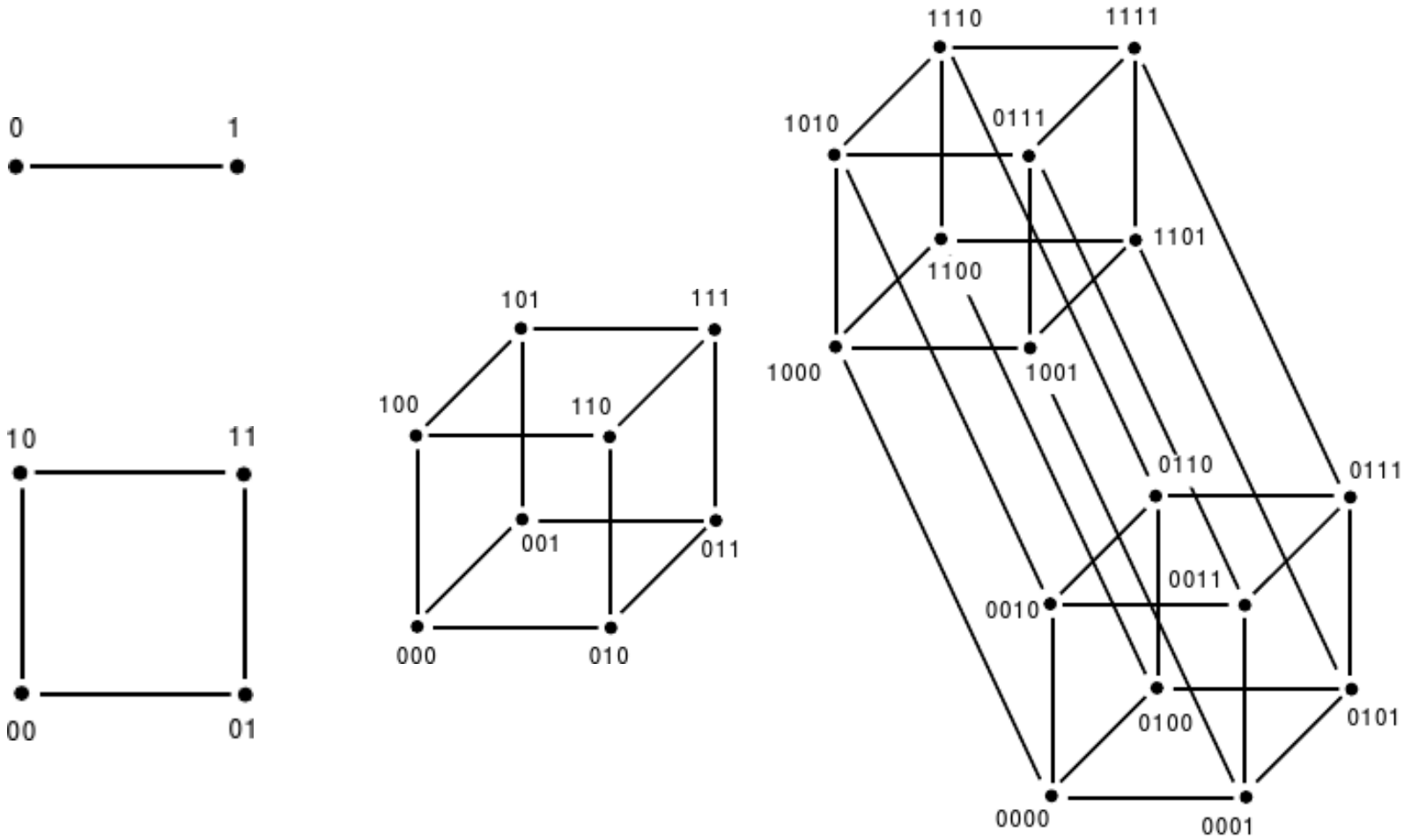
Código ASCII (y 2)

Códigos de caracteres

ISO-8859-15															
	x0	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Un código de tipo ASCII Extendido (ISO-8859-15)

Códigos para corrección y detección de errores



Cubo de orden *n* o hipercubo

Códigos para corrección y detección de errores

- La *distancia* entre dos vértices es el número de aristas que se recorren en el camino más corto desde uno hasta otro
  - Es igual a 1 más el número de vértices intermedios
  - Es igual al número de bits cuyos valores son distintos en ambas cadenas

Códigos para corrección y detección de errores

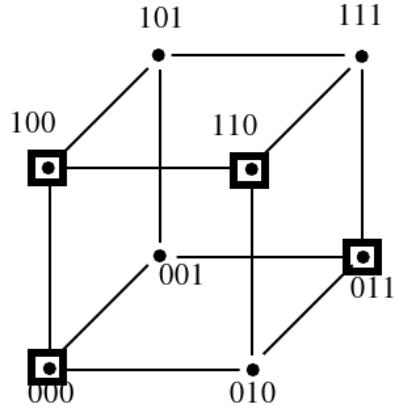
- Todas las cadenas posibles de *n* dígitos se pueden representar asociadas a los vértices de un cubo de orden *n*
  - Pónganse las cadenas que se distingan en el valor de un único bit unidas por una arista
- En un subcubo de orden *m* sus cadenas asociadas tienen $n - m$ bits iguales y los *m* restantes las distintas combinaciones posibles de valores

Códigos para corrección y detección de errores

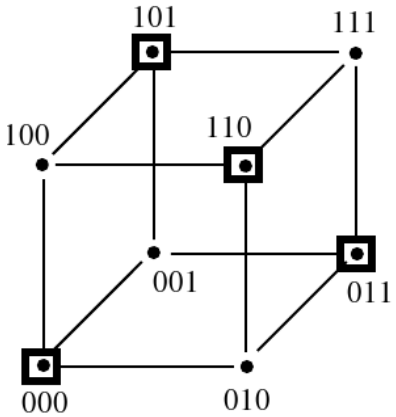
- Código:
  - Conjunto de determinadas secuencias de símbolos
- Palabras del código:
  - Una de dichas secuencias en ese conjunto
- Código de detección de errores:
  - Código con palabras que, en caso de producirse un error, transforma una palabra del código en otra que no lo es

Códigos para corrección y detección de errores

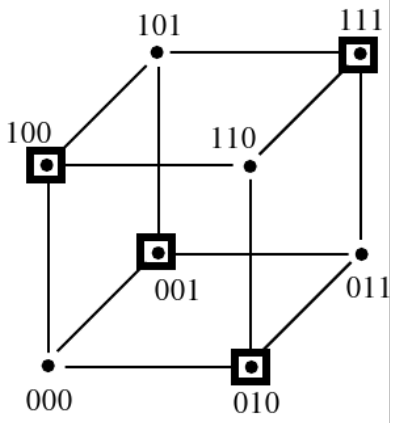
- Se puede detectar el error en un único bit eligiendo las palabras del código con distancia mayor o igual a 2



Código al azar



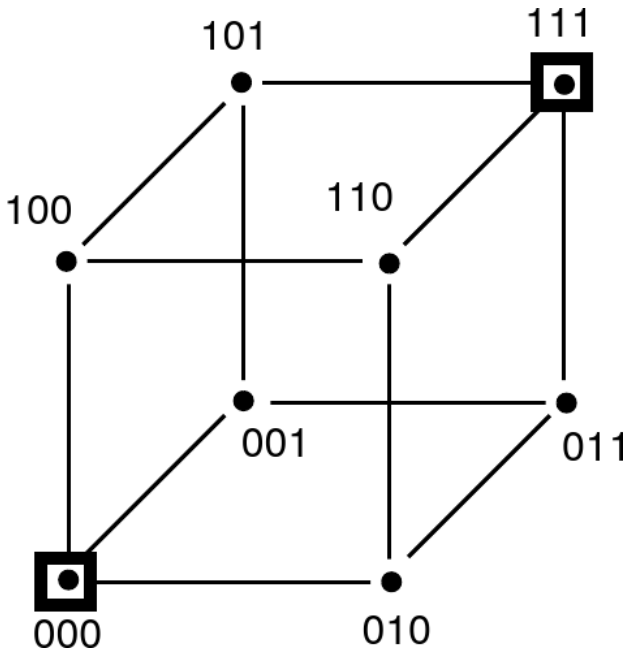
Código de paridad par



Código de paridad impar

Códigos para corrección y detección de errores

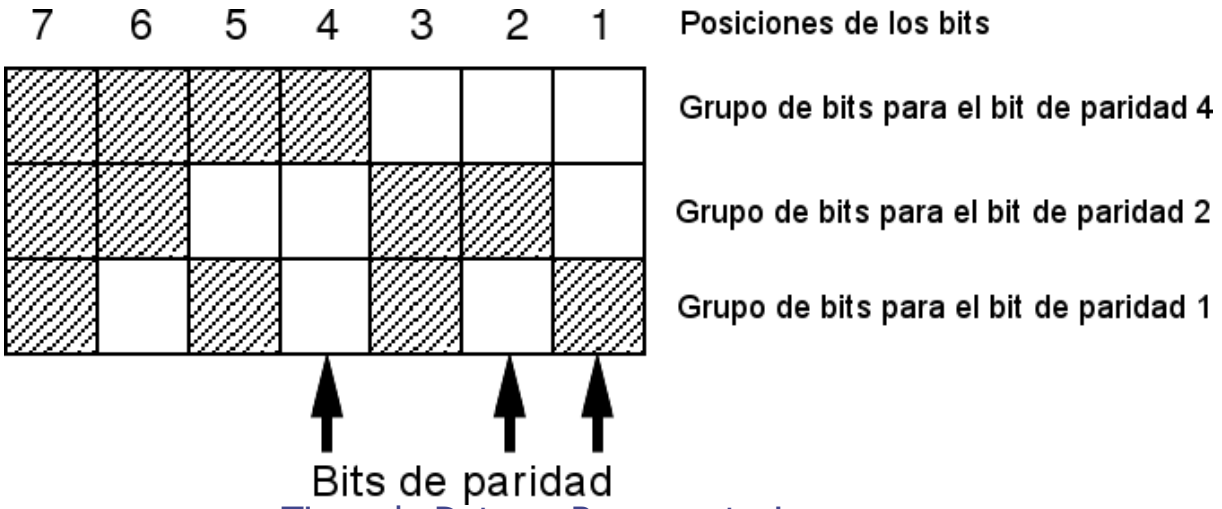
- Para corregir errores hace falta una distancia mayor
  - Además, se debe asumir que se puede producir un tope máximo predeterminado de errores.
  - Se pueden corregir *x* errores si la distancia mínima entre palabras del código es $2x + 1$



Códigos para corrección y detección de errores

- Código de Hamming:
  - Código de palabras de $2^m - 1$ bits con m bits de paridad y $2^m - 1 - m$ bits de información

Los bits de paridad se sitúan en las posiciones 1, 2, 4,…, 2^{m-1}

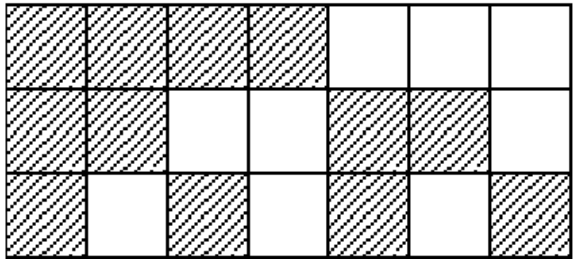


Códigos para corrección y detección de errores

- Código de Hamming:

7	6	5	4	3	2	1	Posiciones de los bits
1	0	1	0	1	0	1	Palabra del código correcta

7	6	5	4	3	2	1	Posiciones de los bits
1	1	1	0	1	0	1	Palabra del código corrupta



Error: 1
Error: 1
No error: 0

El bit corrupto es el de la posición 6