

SISTEMAS DIGITALES

Tema 2 - Representación de la información

José Luis Ávila Jiménez

Objetivos.

- Asimilar las características de los sistemas de numeración y códigos de representación empleados en los Sistemas Digitales.
- Comprender la metodología de conversión entre los diversos sistemas y códigos.
- Describir las características de los códigos detectores de errores más comunes.
- Resolver algunos supuestos prácticos de detección de errores en la transmisión de información.

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.

1. – Sistema binario.
2. – Sistema octal.
3. – Sistema hexadecimal.
4. – Conversión entre los sistemas de numeración.

2.– Definición de código.

2.3.– Códigos binarios.

1. – Códigos numéricos.
 - 1.– Con peso: BCD natural (8421) y BCD Aiken.
 - 2.3.1.2.– Sin peso: BCD exceso a 3, Gray y Johnson.

2. – Códigos alfanuméricos. 2.4.–

Códigos detectores de errores.

- 2.4.1.– Concepto de distancia y distancia mínima.
- 2.4.2.– Códigos de paridad constante.

2.1.– Sistemas de numeración posicional.

Un sistema numérico se define por sus símbolos básicos, llamados dígitos o cifras, y las formas de combinar los dígitos para representar todos los números requeridos.

En un sistema de numeración posicional, un número se define mediante una cadena de dígitos, de forma que **el valor de cada dígito depende de la posición** que ocupe en la cadena, es decir tiene un peso.

El **peso** se expresa en función de una **potencia de un número** concreto, que se denomina **base (b)**.

La **base da nombre** al sistema de numeración y define los valores que pueden tener los dígitos: **0 a $b - 1$** .

2.1.– Sistemas de numeración posicional.

Si consideramos un sistema de numeración con **n** dígitos para parte entera y **m** para la fraccionaria, un número **X** se representa:

$$X = (a_{n-1} a_{n-2} \dots a_2 a_1 a_0, a_{-1} a_{-2} \dots a_{-m})_b$$

Valor de los pesos:

$$P = b^{n-1} b^{n-2} \dots b^2 b^1, b^0, b^{-1} b^{-2} \dots b^{-m}$$

Valor de los dígitos:

$$0 \leq a_i < b$$

2.1.– Sistemas de numeración posicional.

Valor del número **X**:

$$V(X) = a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \dots + a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} + \dots + a_{-m} \cdot b^{-m})_b$$

$$V(X) = \sum_{i=-m}^{n-1} a_i \cdot b^i$$

Tamaño de palabra. En los sistemas digitales se trabaja con un nº finito de “n” dígitos llamado tamaño de palabra.

2.1.– Sistemas de numeración posicional.

Los sistemas de numeración posicionales utilizados en los sistemas digitales son **binario**, **octal** y **hexadecimal**, por ser potencias de dos.

El sistema de numeración **decimal** es el más utilizado por el ser humano.

Nombre	Base “b”	Dígitos utilizados
Binario	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

2.1.1.– Sistema decimal.

La base es **b=10**. Los dígitos son: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

Los pesos son una potencia de 10:

... 10^8 10^7 10^6 10^5 10^4 10^3 10^2 10^1 10^0 . 10^{-1} 10^{-2} 10^{-3} 10^{-4} ...

Ejemplos:

$$5684 = 5000 + 600 + 80 + 4 = 5 \cdot 1000 + 6 \cdot 100 + 8 \cdot 10 + 4 \cdot 1 = 5 \cdot 10^3 + 6 \cdot 10^2 + 8 \cdot 10^1 + 4 \cdot 10^0$$

$$5.643 = 5 + 0.6 + 0.04 + 0.003 = 5 \cdot 1 + 6 \cdot 0.1 + 4 \cdot 0.01 + 3 \cdot 0.001 = 5 \cdot 1 + 6 \cdot \frac{1}{10} + 4 \cdot \frac{1}{100} + 3 \cdot \frac{1}{1000} =$$

$$5 \cdot 10^0 + 6 \cdot 10^{-1} + 4 \cdot 10^{-2} + 3 \cdot 10^{-3}$$

2.1.2.– Sistema binario.

La base es **b=2**. Los dígitos son: 0 y 1. Se denomina bit (*Binary dIgit*)

Los pesos son una potencia de 2:

...	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}	2^{-4}	...
...	256	128	64	32	16	8	4	2	1	.	0.5	0.25	0.125	0.0625	...

Valor de un número binario:

$$101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 4 + 0 + 1 = 5$$

$$101.01_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 + 0 \cdot 0.5 + 1 \cdot 0.25 = 4 + 0 + 1 + 0 + 0.25 = 5.25$$

2.1.3.– Sistema octal.

La base es **b=8**. Los dígitos son: 0, 1, 2, 3, 4, 5, 6 y 7.

Los pesos son una potencia de 8:

$$\dots 8^8 \ 8^7 \ 8^6 \ 8^5 \ 8^4 \ 8^3 \ 8^2 \ 8^1 \ 8^0. \ 8^{-1} \ 8^{-2} \ 8^{-3} \ 8^{-4} \dots$$

Como $8 = 2^3$, cada dígito octal se forma agrupando 3 bits. Se utilizará para las conversiones entre ambos sistemas.

Valor de un número octal. Ejemplo:

$$\begin{aligned} 472.16_8 &= 4 \cdot 8^2 + 7 \cdot 8^1 + 2 \cdot 8^0 + 1 \cdot 8^{-1} + 6 \cdot 8^{-2} = 4 \cdot 64 + 7 \cdot 8 + 2 \cdot 1 + 1 \cdot 0.125 + 6 \cdot 0.015625 = \\ &256 + 56 + 2 + 0.125 + 0.09375 = 314.21875_{10} \end{aligned}$$

2.1.4.– Sistema hexadecimal.

La base es **b=16**. Los dígitos son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

Los pesos son una potencia de 16:

... 16^8 16^7 16^6 16^5 16^4 16^3 16^2 16^1 16^0 . 16^{-1} 16^{-2} 16^{-3} 16^{-4} ...

Como $16 = 2^4$, cada dígito octal se forma agrupando 4 bits. Se utilizará para las conversiones entre ambos sistemas.

Valor de un número hexadecimal. Ejemplo:

$$\mathbf{1A.2}_{16} = \mathbf{1} \cdot 16^1 + \mathbf{10} \cdot 16^0 + \mathbf{2} \cdot 16^{-1} = \mathbf{1} \cdot 16 + \mathbf{10} \cdot 1 + \mathbf{2} \cdot 0.0625 = 16 + 10 + 0.125 = \mathbf{26.125}_{10}$$

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.
 1. – Sistema binario.
 2. – Sistema octal.
 3. – Sistema hexadecimal.
 4. – **Conversión entre los sistemas de numeración.**
- 2.– Definición de código.
- 2.3.– Códigos binarios.
 1. – Códigos numéricos.
 - 1.– Con peso: BCD natural (8421) y BCD Aiken.
 - 2.3.1.2.– Sin peso: BCD exceso a 3, Gray y Johnson.
2. – Códigos alfanuméricos. 2.4.–
Códigos detectores de errores.
 - 2.4.1.– Concepto de distancia y distancia mínima.
 - 2.4.2.– Códigos de paridad constante.

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de binario, octal y hexadecimal a decimal.

Se aplica la fórmula que determina el valor de un número. Al operar en decimal se obtiene su equivalente decimal.

- Ejemplo de conversión de binario a decimal:

$$\mathbf{110010,011_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} =}$$
$$\mathbf{1 \cdot 32 + 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 + 0 \cdot 0,5 + 1 \cdot 0,25 + 1 \cdot 0,125 = 50,375_{10}}$$

- Ejemplo de conversión de octal a decimal:

$$\mathbf{567,45_8 = 5 \cdot 8^2 + 6 \cdot 8^1 + 7 \cdot 8^0 + 4 \cdot 8^{-1} + 5 \cdot 8^{-2} = 5 \cdot 64 + 6 \cdot 8 + 7 \cdot 1 + 4 \cdot 0,125 + 5 \cdot 0,015625 = 375,578125_{10}}$$

- Ejemplo de conversión de hexadecimal a decimal:

$$\mathbf{C8E,AB_{16} = C_{16} \cdot 16^2 + 8_{16} \cdot 16^1 + E_{16} \cdot 16^0 + A_{16} \cdot 16^{-1} + B_{16} \cdot 16^{-2} = 12 \cdot 256 + 8 \cdot 16 + 14 \cdot 1 + 10 \cdot 0,0625 +}$$
$$\mathbf{11 \cdot 0,00390625 = 3214,66796875_{10}}$$

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de decimal a binario, octal y hexadecimal.

Se convierte por separado la parte entera de la fraccionaria.

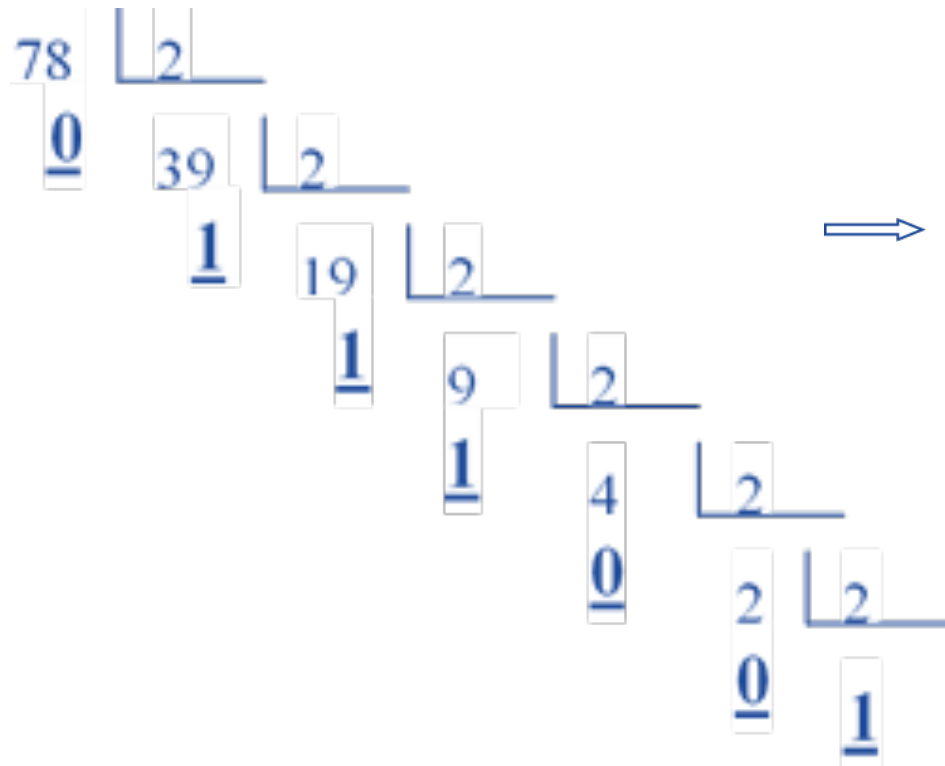
Vamos a utilizar el método de divisiones y multiplicaciones sucesivas por la base.

- **Parte entera:** Se divide sucesivamente la parte entera del número decimal por la base hasta obtener un cociente igual a 0.
 - Los restos forman los dígitos en orden creciente de peso. Es decir, el primer resto corresponde al dígito menos significativo y el último al más significativo.
- **Parte fraccionaria:** Se multiplica la parte fraccionaria del número decimal por la base. La parte entera obtenida corresponde al dígito más significativo. Se vuelve a repetir el proceso con la parte fraccionaria hasta que sea 0, se repita o se de por buena la aproximación. Se obtienen los dígitos en orden decreciente de peso.

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de decimal a binario, octal y hexadecimal.

- Ejemplo de conversión de decimal a binario: $78.59375_{10} = 1001110.10011_2$



$$\Rightarrow 78_{10} = 1001110_2$$

$$0.59375 \cdot 2 = 1.1875$$

$$0.1875 \cdot 2 = 0.375$$

$$0.375 \cdot 2 = 0.75$$

$$0.75 \cdot 2 = 1.5$$

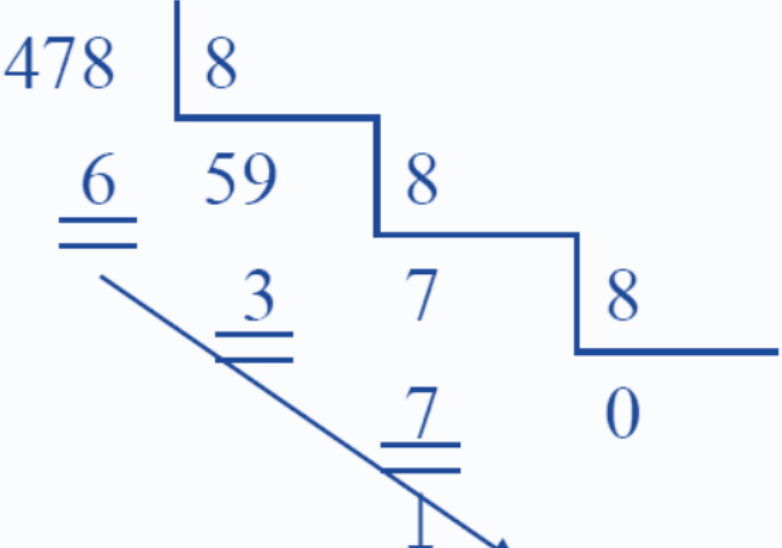
$$0.5 \cdot 2 = 1.0$$

$$0.59375_{10} = 0.10011_2$$

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de decimal a binario, octal y hexadecimal.

- Ejemplo de conversión de decimal a octal: $478,58_{10} = 736,4507_8$

	$0,58 \times 8 = 4,64 \Rightarrow 4 \quad + \text{ signif.}$
	$0,64 \times 8 = 5,12 \Rightarrow 5$
	$0,12 \times 8 = 0,96 \Rightarrow 0$
	$0,96 \times 8 = 7,68 \Rightarrow 7 \quad - \text{ signif.}$

$478,58 = 736,4507_8$

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de decimal a binario, octal y hexadecimal.

- Ejemplo de conversión de **decimal a hexadecimal**:

The diagram illustrates the conversion of the decimal number 478,58 to hexadecimal. It consists of two main parts: a division ladder for the integer part (478) and a multiplication ladder for the fractional part (0,58).

Integer Part Conversion (478 to hexadecimal):

Quotient	Remainder
478	14
29	13
1	1
0	0

The remainders, read from bottom to top, are 1, 13, and 14. These correspond to the hexadecimal digits 1, D, and E respectively. The integer part of the hexadecimal result is 1DE.

Fractional Part Conversion (0,58 to hexadecimal):

Fractional Part	Integer Part
0,58	9
0,28	4
0,48	7
0,68	10

The integer parts, read from top to bottom, are 9, 4, 7, and 10. These correspond to the hexadecimal digits 9, 4, 7, and A respectively. The fractional part of the hexadecimal result is 947A.

Final Result:

$$478,58 = 1DE,947A_{16}$$

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de binario a octal y viceversa:

- Como $8 = 2^3$, cada dígito octal consta de 3 bits y viceversa.
- Ejemplo de conversión de **octal a binario**:

$$257.46_8 = 010\ 101\ 111.100\ 110_2 = 10101111.10011_2$$

- Ejemplo de conversión de **binario a octal**:

$$11001.01101_2 = 011\ 001.011\ 010_2 = 31.32_8$$

b2	b1	b0	Octal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de binario a hexadecimal y viceversa:

- Como $16 = 2^4$, cada dígito hexadecimal consta de 4 bits y viceversa.
- Ejemplo de conversión de **hexadecimal a binario**:

$$17A.BC_{16} = 0001\ 0111\ 1010.1011\ 1100_2 = 101111010.101111_2$$

- Ejemplo de conversión de **binario a hexadecimal**:

$$110110101.011010101_2 = 0001\ 1011\ 0101.0110\ 1010\ 1000_2 = 1B5.6A8_{16}$$

b3	b2	b1	b0	Hex
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	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

2.1.5.– Conversión entre los sistemas de numeración.

Conversión de hexadecimal a octal y viceversa:

En estos casos lo más sencillo es pasarlo a binario y después al sistema deseado.

Hexadecimal \Rightarrow Binario \Rightarrow Octal

Octal \Rightarrow Binario \Rightarrow Hexadecimal

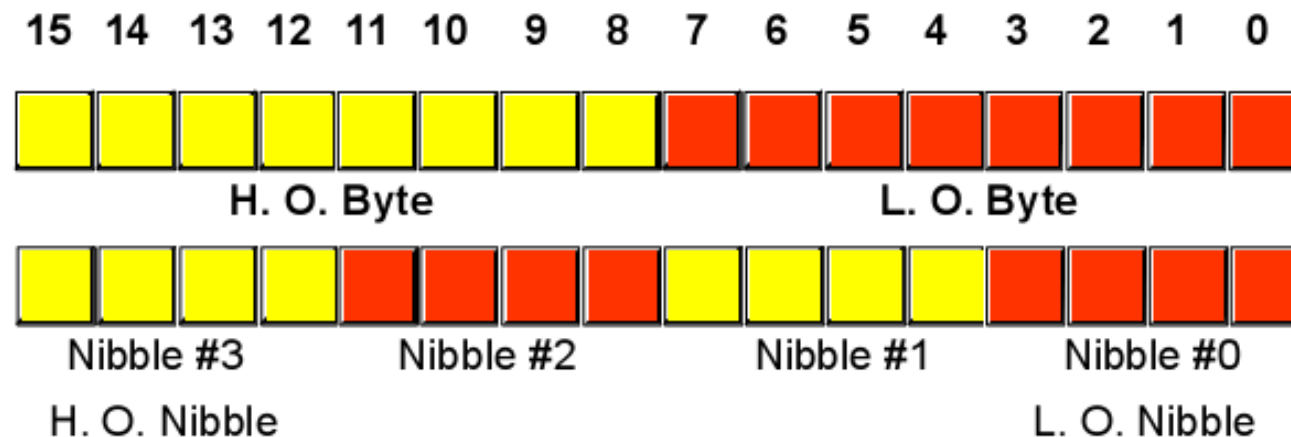
Unidades de almacenamiento de información

Bit: Dígito Binario (abreviado b)

Palabra: depende del sistema digital

Byte: 8 bits. Similar a octeto (Abreviado B)

Nibble 4 bytes



Unidades de almacenamiento de información Según las unidades del SI

Prefijo	Símbolo del prefijo	Nombre resultante del prefijo + <i>byte</i>	Símbolo del múltiplo del <i>byte</i>	Factor y valor en el SI
Valor de referencia		byte	B	$10^0 = 1$
kilo	k	kilobyte	kB	$10^3 = 1\ 000$
mega	M	megabyte	MB	$10^6 = 1\ 000\ 000$
giga	G	gigabyte	GB	$10^9 = 1\ 000\ 000\ 000$
tera	T	terabyte	TB	$10^{12} = 1\ 000\ 000\ 000\ 000$
peta	P	petabyte	PB	$10^{15} = 1\ 000\ 000\ 000\ 000\ 000$
exa	E	exabyte	EB	$10^{18} = 1\ 000\ 000\ 000\ 000\ 000\ 000$
zetta	Z	zettabyte	ZB	$10^{21} =$ 1 000 000 000 000 000 000 000
yotta	Y	yottabyte	YB	$10^{24} =$ 1 000 000 000 000 000 000 000 000

Unidades de almacenamiento de información según prefijos ISO/IEC 80000-1

Prefijo	Símbolo del prefijo	Nombre resultante del prefijo + byte	Símbolo del múltiplo del byte	Factor y valor en el ISO/IEC 80000-13
		byte	B	$2^0 = 1$
kibi	Ki	kibibyte	KiB	$2^{10} = 1024$
mebi	Mi	mebibyte	MiB	$2^{20} = 1\,048\,576$
gibi	Gi	gibibyte	GiB	$2^{30} = 1\,073\,741\,824$
tebi	Ti	tebibyte	TiB	$2^{40} = 1\,099\,511\,627\,776$
pebi	Pi	pebibyte	PiB	$2^{50} = 1\,125\,899\,906\,842\,624$
exbi	Ei	exbibyte	EiB	$2^{60} = 1\,152\,921\,504\,606\,846\,976$
zebi	Zi	zebibyte	ZiB	$2^{70} = 1\,180\,591\,620\,717\,411\,303\,424$
yobi	Yi	yobibyte	YiB	$2^{80} = 1\,208\,925\,819\,614\,629\,174\,706\,176$

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.
 1. – Sistema binario.
 2. – Sistema octal.
 3. – Sistema hexadecimal.
 4. – Conversión entre los sistemas de numeración.
2. – **Definición de código.**
3. – Códigos binarios.
 1. Códigos numéricos.
 2. Con peso: BCD natural (8421) y BCD Aiken.
 3. Sin peso: BCD exceso a 3, Gray y Johnson.
4. Códigos alfanuméricos.
5. Códigos detectores de errores.

Concepto de distancia y distancia mínima.

Códigos de paridad constante.

2.2.– Definición de código.

Cuando se representan números, letras o palabras por medio de un grupo especial de símbolos, se dice que se encuentran codificados, y al grupo de símbolos se le denomina código.

Se define código como la representación biunívoca de los números y caracteres alfanuméricos mediante una combinación de símbolos determinados.

Códigos universalmente conocidos son el código QR, código de barras, UPC (código universal de producto), código Morse, etc.

Los sistemas digitales generan, procesan y almacenan información utilizando señales binarias. Si la información que se está tratando no es estrictamente binaria, como es lo más normal, de alguna forma hay que representar esa información mediante magnitudes binarias: es lo que se conoce como codificación.

2.3.– Códigos binarios.

2.3.1.– Códigos numéricos.

Si los símbolos son los dígitos binarios, el código estará formado por combinaciones de bits o combinaciones binarias, y éste se denomina código binario.

Las combinaciones binarias pertenecientes al código se denominan palabras.

Se denomina codificación binaria al proceso para representar la información mediante códigos binarios.

Códigos binarios numéricos: se utilizan para representar los números del 0 al 9, y pueden tener en cuenta la posición del bit que contribuye al valor representado por el carácter con una determinada cantidad (o peso), o no.

2.3.1.– Códigos numéricos. Características de los códigos

Código **ponderado**: A cada dígito le corresponde un valor (peso) en función de la posición que ocupa

Código **autocomplementario**: El complemento a la base menos uno de cualquier número del código también pertenece al código.

Combinaciones **adyacentes**: Dos combinaciones binarias son adyacentes si difieren en un solo bit

Código **continuo**: Las combinaciones correspondientes a números decimales consecutivos son adyacentes

Código **cíclico**: Un código binario es cíclico si es continuo y la última combinación es adyacente a la primera.

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.
 1. – Sistema binario.
 2. – Sistema octal.
 3. – Sistema hexadecimal.
 4. – Conversión entre los sistemas de numeración.
- 2.– Definición de código.
- 2.3.– Códigos binarios.
 1. – Códigos numéricos.
 1. – **Con peso: BCD natural (8421) y BCD Aiken.**
 2. – Sin peso: BCD exceso a 3, Gray y Johnson.
2. – Códigos alfanuméricos. 2.4.–
Códigos detectores de errores.
 - 2.4.1.– Concepto de distancia y distancia mínima.
 - 2.4.2.– Códigos de paridad constante.

2.3.1.1.– Con peso: BCD natural (8421).

El código BCD (decimal codificado en binario) natural, BCDN, BCD estándar, BCD 8421, o simplemente BCD; es el código más sencillo. Consiste en que cada cifra se representa por su valor real en binario.

Ejemplos:

3 2 , 8 4
↓ ↓ ↓ ↓
0011 0010 , 1000 0100_{BCD}

01101001,00100101_{BCD}
↓ ↓ ↓ ↓
6 9 , 2 5

8	4	2	1	De ci mal
b ₃	b ₂	b ₁	b ₀	
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

2.3.1.1.– Con peso: BCD Aiken (8421).

Es un código ponderado (2421) autocomplementario, es decir, el complemento a 9 (C9) de cualquier dígito decimal se obtiene complementando todos los bits de su palabra del código.

Del 0 al 4 coincide con el BCD, del 5 al 9 se obtienen cambiando todos los bits de su complemento a 9.

Este código tenía la ventaja de que la realización de circuitos aritméticos eran más sencillos. Con el avance de las tecnologías de fabricación de circuitos integrados ha dejado de utilizarse

2	4	2	1	De ci m al
b_3	b_2	b_1	b_0	
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
1	0	1	1	5
1	1	0	0	6
1	1	0	1	7
1	1	1	0	8
1	1	1	1	9

2.3.1.1.– Con peso: Otros códigos BCD.

Se puede variar el peso de cada bit para obtener diferentes códigos, incluso con números negativos. Veamos dos ejemplos: BCD 5421 y BCD 441–2.

Dígito decimal	BCDN (8421)	BCD Aiken (2421)	BCD 5421	BCD 441–2
0	0000	0000	0000	0000
1	0001	0001	0001	0010
2	0010	0010	0010	0101
3	0011	0011	0011	0111
4	0100	0100	0100	0100
5	0101	1011	1000	1010
6	0110	1100	1001	1101
7	0111	1101	1010	1111
8	1000	1110	1011	1100
9	1001	1111	1100	1110

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.
 1. – Sistema binario.
 2. – Sistema octal.
 3. – Sistema hexadecimal.
 4. – Conversión entre los sistemas de numeración.
- 2.– Definición de código.
- 2.3.– Códigos binarios.
 1. – Códigos numéricos.
 1. – Con peso: BCD natural (8421) y BCD Aiken.
 2. – **Sin peso: BCD exceso a 3, Gray y Johnson.**
2. – Códigos alfanuméricos. 2.4.–
Códigos detectores de errores.
 - 2.4.1.– Concepto de distancia y distancia mínima.
 - 2.4.2.– Códigos de paridad constante.

2.3.1.2.– Sin peso: BCD exceso a 3.

Se suma 3 a cada dígito decimal, y el resultado se convierte a su equivalente binario de 4 bits.

Es autocomplementario lo cual hace que tenga ciertas ventajas en operaciones aritméticas.

Con el avance de las tecnologías de fabricación de circuitos integrados ha dejado de utilizarse.

Dec.	Dec. +3	b_3	b_2	b_1	b_0
0	3	0	0	1	1
1	4	0	1	0	0
2	5	0	1	0	1
3	6	0	1	1	0
4	7	0	1	1	1
5	8	1	0	0	0
6	9	1	0	0	1
7	10	1	0	1	0
8	11	1	0	1	1
9	12	1	1	0	0

Código Gray.

Es un código continuo y cíclico sin peso y reflejado.

El código Gray de n bits se forma a partir del de $n-1$ bits haciendo una reflexión especular y añadiendo por la izquierda un bit con valor 0 en las 2^{n-1} primeras combinaciones y un 1 en las 2^{n-1} siguientes.

Cada combinación es adyacente (se diferencian en un solo bit) con la que le sigue y la precede. La primera y la última también son adyacentes.

1 bit.
0
1

2 bits	
0	0
0	1
1	1
1	0

3 bits		
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

4 bits			
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

2.3.1.2.– Sin peso: Código Johnson.

Es un código continuo y cíclico donde se va desplazando un 1 hacia la izquierda y cuando se ha completado, lo que se desplaza es un 0.

Para un código de n bits la capacidad de codificación es de $2n$ números diferentes.

Dec	Código Johnson de 5 bits				
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	1	1
4	0	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	0
7	1	1	1	0	0
8	1	1	0	0	0
9	1	0	0	0	0

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.
 1. – Sistema binario.
 2. – Sistema octal.
 3. – Sistema hexadecimal.
 4. – Conversión entre los sistemas de numeración.
- 2.– Definición de código.
- 2.3.– Códigos binarios.
 1. – Códigos numéricos.
 2. – **Códigos alfanuméricos.**
4. – Códigos detectores de errores.
 - 1.– Concepto de distancia y distancia mínima.
 - 2.4.2.– Códigos de paridad constante.

2.3.2.– Códigos alfanuméricos.

En los computadores y otros sistemas digitales se procesa gran cantidad de información no numérica.

Conviene, por lo tanto, asignar un carácter o palabra binaria breve, de tamaño fijo, para representar estos datos llamados alfanuméricos.

Un código completo de este tipo puede incluir 26 letras minúsculas, 26 mayúsculas, 10 dígitos, siete signos de puntuación y entre 20 y 40 caracteres más como son + / # % *, y otros similares.

La codificación de los diferentes elementos que se vayan a procesar puede hacerse de forma arbitraria, pero está normalizada.

2.3.2.– Códigos alfanuméricos.

ASCII (*American Standard Code for Information Interchange*): Es el más utilizado casi universalmente. Este código comenzó con 6 bits (sin minúsculas), posteriormente con 7 bits y finalmente con 8 bits, utilizando un bit más para la detección de errores.

EBCDIC: (*Extended Binary Coded Decimal Interchange Code*) Desarrollado por IBM. Utiliza ocho bits y codifica un conjunto de 256 caracteres. Está en desuso.

Unicode (Universal Code):

- Utiliza dos formas de mapeo:
 - Codificación UTF (Codificación de longitud variable, P.Ej.: UTF–8, 8 bits)
 - Codificación UCS (Codificación de longitud fija, P.Ej.: UCS–2, 16 bits)
- Ámpliamente utilizado en la actualidad

Código ASCII de 8 bits

Carácter	Bits	Decimal	Hexadecimal	Carácter	Bits	Decimal	Hexadecimal
0	00110000	48	30
1	00110001	49	31	W	01010111	87	57
2	00110010	50	32	X	01011000	88	58
3	00110011	51	33	Y	01011001	89	59
4	00110100	52	34	Z	01011010	90	5A
5	00110101	53	35
6	00110110	54	36	a	01100001	97	61
7	00110111	55	37	b	01100010	98	62
8	00111000	56	38	c	01100011	99	63
9	00111001	57	39	d	01100100	100	64
...	e	01100101	101	65
A	01000001	65	41
B	01000010	66	42	w	01110111	119	77
C	01000011	67	43	x	01111000	120	78
D	01000100	68	44	y	01111001	121	79
E	01000101	69	45	z	01111010	122	7A

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.
 1. – Sistema binario.
 2. – Sistema octal.
 3. – Sistema hexadecimal.
 4. – Conversión entre los sistemas de numeración.
- 2.– Definición de código.
- 2.3.– Códigos binarios.
 1. – Códigos numéricos.
 - 1.– Con peso: BCD natural (8421) y BCD Aiken.
 - 2.3.1.2.– Sin peso: BCD exceso a 3, Gray y Johnson.
2. – Códigos alfanuméricos.
4. – **Códigos detectores de errores.**
 - 1.– Concepto de distancia y distancia mínima.
 - 2.4.2.– Códigos de paridad constante.

2.4.– Códigos detectores de errores.

Los sistemas digitales intercambian información entre sí mediante alguno de los códigos analizados: BCD, ASCII, etc.

El sistema que envía la información se denomina transmisor.

El sistema que recibe la información se denomina receptor.

Generalmente los sistemas pueden ser tanto transmisores como receptores.



2.4.– Códigos detectores de errores.

Se puede producir errores en la comunicación debido a las interferencias eléctricas o por avería de alguno de los componentes.

Se han desarrollados códigos que permitan detectar si la información recibida es errónea e incluso otros que pueden corregir el error.

Un código se dice que es un código de detección de errores si tiene la propiedad de detectar combinaciones binarias que no son del código.

Una primera condición sería no usar todas las 2^n posibles combinaciones de bits que se podrán obtener con n bits.

Por ejemplo, el código BCD no usa las combinaciones binarias del 10 al 15. Pero no es un código detector de errores, ya que si se envía el 0 (0000) y el receptor recibe el 1 (0001) no se puede detectar el error. Por tanto, es una condición necesaria, pero no suficiente.

TEMA 2: Representación de la información

1. – Sistemas de numeración posicional.
 1. – Sistema binario.
 2. – Sistema octal.
 3. – Sistema hexadecimal.
 4. – Conversión entre los sistemas de numeración.
- 2.– Definición de código.
- 2.3.– Códigos binarios.
 1. – Códigos numéricos.
 - 1.– Con peso: BCD natural (8421) y BCD Aiken.
 - 2.3.1.2.– Sin peso: BCD exceso a 3, Gray y Johnson.
2. – Códigos alfanuméricos. 2.4.–
Códigos detectores de errores.
 3. – **Concepto de distancia y distancia mínima.**
 4. – Códigos de paridad constante.

2.4.1– Concepto de distancia y distancia mínima.

La condición necesaria y suficiente para que un código pueda detectar errores viene dada por su distancia mínima.

Concepto de distancia: La distancia entre dos combinaciones binarias es el número de bits que hay que cambiar en una para obtener la otra.

$$D(0000, 0001) = 1$$

$$D(0000, 1111) = 4$$

Concepto de distancia mínima (D_m): La menor de las distancias de todas las posibles parejas de palabras de ese código.

Para que un código pueda detectar errores en un bit su D_m debe ser mayor que 1.

Para que un código pueda detectar errores en n bits su D_m debe ser mayor que $n+1$.

Códigos detectores de errores más comunes:

- Códigos de peso fijo.
- Códigos de paridad.

2.4.2.– Códigos de paridad constante.

Estos códigos se obtienen añadiendo a las combinaciones de los códigos de distancia unidad, anteriormente estudiados, un bit llamado bit de paridad.

Hay dos tipos de paridad:

- **Par.** El bit de paridad toma el valor necesario para que el número de bits con valor 1 en la palabra del nuevo código sea par. El número 0 se considera par.
- **Impar.** El bit de paridad toma el valor necesario para que el número de bits con valor 1 en la palabra del nuevo código sea impar.

El bit de paridad se puede colocar al final de la palabra del código, pero en general se coloca a la izquierda.

2.4.2.– Códigos de paridad constante.

Ejemplo: Código BCDN con paridad par e impar:

Digito	BCDN	Bit de paridad impar	Código paridad impar	Bit de paridad par	Código paridad par
0	0000	1	1 0000	0	0 0000
1	0001	0	0 0001	1	1 0001
2	0010	0	0 0010	1	1 0010
3	0011	1	1 0011	0	0 0011
4	0100	0	0 0100	1	1 0100
5	0101	1	1 0101	0	0 0101
6	0110	1	1 0110	0	0 0110
7	0111	0	0 0111	1	1 0111
8	1000	0	0 1000	1	1 1000
9	1001	1	1 1001	0	0 1001

2.4.2.– Códigos de paridad constante.

La detección de errores en estos códigos consiste en comprobar, mediante un circuito lógico, que el número de unos de cada combinación recibida es par (códigos de paridad par) o impar (códigos de paridad impar).

Con el bit de paridad se comprueba que los códigos con distancia mínima igual a 1 se transforman en códigos de distancia mínima igual a 2.

Normalmente, para detectar errores, basta con un código de distancia mínima igual a 2, ya que la probabilidad de error disminuye mucho.

Esta operación de añadir un bit de paridad es válida también para códigos alfanuméricos.