

# **ARQUITECTURA DE COMPUTADORAS**

---

Guía de Autoestudio

---

## **Codificación de la Información**

---

Ing. Gustavo Maurokefalidis  
CICLO LECTIVO: 2023

# **CODIFICACIÓN DE LA INFORMACIÓN**

## **1) INTRODUCCION**

Para poder transmitir información existen dos problemas fundamentales: el **HARDWARE** y el **LENGUAJE DE COMUNICACIÓN**. Haciendo una analogía entre las computadoras y la comunicación humana podemos decir que, si bien todas las personas del mundo tienen el mismo hardware para la comunicación hablada (labios, lengua, dientes y el resto del complejo aparato bucal) para transmitir y los oídos para recibir, la comunicación oral es posible solamente cuando dos personas conocen el mismo lenguaje, es decir la misma manera de **codificar la información**.

Así como el habla sería imposible sin lenguajes comunes, la comunicación entre computadoras sería imposible sin coordinación de *códigos de caracteres*.

Todas las computadoras digitales actuales usan un **lenguaje binario** para representar la información, internamente. Debido a que algunos de los dispositivos con los que se deben comunicar las computadoras están diseñados para uso humano (específicamente teclados, monitores e impresoras), es importante que estos **periféricos** utilicen un código de comunicación compatible con la comunicación humana.

Existen varios métodos para alcanzar dicha compatibilidad y cada uno utiliza un modo diferente de codificar los números y las letras, que conforman la base de la comunicación escrita entre las personas.

Actualmente es frecuente encontrar en los *sistemas de cómputo*, dispositivos provistos por distintos fabricantes. La posibilidad de conectar estos dispositivos existe únicamente si éstos utilizan un código común para la transmisión y recepción de la información.

Son claras las ventajas de conseguir que todas las computadoras utilicen el mismo código de comunicación. Aun cuando la calidad de los códigos varía enormemente, casi cualquier estándar universal sería mejor que ninguno. Si bien hay códigos que prácticamente son aceptados por todos los fabricantes, aún no existe un estándar que optimice el aprovechamiento tanto de los recursos del hardware como las nuevas teorías sobre codificación e información.

En sentido genérico **CÓDIGO** significa: Sistema de signos y de reglas que permite formular y comprender un mensaje.

La codificación consiste en establecer una ley de correspondencia, llamada **CÓDIGO**, entre las informaciones por representar y las posibles configuraciones binarias, de tal manera que a cada información corresponda una y generalmente solo una, configuración binaria.

El proceso de **CODIFICAR** significa: Transformar, mediante las reglas de un código, la formulación de un mensaje.

Llamamos **CODIFICACIÓN** al proceso de convertir un símbolo complejo en un grupo de símbolos más simples. Ejemplo: convertir una letra del alfabeto en un código de cinco bits.

**DECODIFICAR**: Aplicar inversamente las reglas de su código a un mensaje codificado para obtener la forma primitiva de este.

**DECODIFICACIÓN** es el proceso inverso al de codificación, se convierte a un código donde la cantidad de símbolos es menor, pero cada uno contiene más información.

**TRANSCODIFICACIÓN**: Aplicación de un cambio de código a una información ya codificada. Ejemplo: EBCDIC a ASCII.

Definimos como **CANAL DE COMUNICACIÓN** al medio utilizado para poder transmitir una información codificada, decodificada, o transcodificada, desde un emisor hacia un receptor.

**OTRAS DEFINICIONES:**

En comunicaciones, un código es una regla para convertir una información (por ejemplo, una letra, palabra o frase) en otro objeto o acción, no necesariamente del mismo tipo. Uno de los motivos es permitir la comunicación en lugares donde el lenguaje ordinario hablado o escrito es difícil o imposible. Por ejemplo, la codificación Q usada por los radioaficionados, permiten codificar frases utilizadas frecuentemente bajo un código alfabético corto, que permita ahorrar tiempo en la transmisión Ej: QRK: Su señal es ininteligible, QRL: Esta frecuencia está ocupada, Etc.

En los sistemas de comunicaciones, la codificación es la alteración de las características de una señal para hacerla más adecuada a una aplicación prevista, como optimizar la señal para la transmisión, mejorar la calidad y fidelidad de la transmisión, modificar el espectro de la señal, aumentar el contenido de la información, proporcionar detección y/o corrección de errores y proporcionar seguridad a los datos (Nota: Un único esquema de codificación no suele proporcionar más de una o dos capacidades específicas. Los distintos códigos tienen diferentes conjuntos de ventajas e inconvenientes).

En comunicaciones y procesamiento de la información, la descodificación es el proceso de convertir los datos de un mensaje, que han sido enviados por una fuente, en información comprensible por un receptor.

En telecomunicaciones, la transcodificación es la conversión directa de digital a digital de un esquema de codificación, como la voz LPC-10, a un esquema de codificación diferente sin devolver las señales a su forma analógica.

**BITS:** Teniendo en cuenta que las computadoras manejan un lenguaje binario analizaremos los **DÍGITOS BINARIOS** como caracteres para comunicación de datos.

La condición binaria es la que posee una calidad BIVALUADA. En el sistema binario de numeración esas condiciones están representadas por los dígitos **0** y **1**. Se denomina **BIT** (contracción de **BINARY DIGIT**) al dígito binario, independientemente del valor asignado (0 o 1).

Los dígitos binarios llevan, ambos, la misma cantidad de información, ya que la presencia de uno significa la ausencia del otro.

Un proceso fundamental en la codificación binaria es determinar la cantidad de BITS necesarios para representar las informaciones. De manera que podamos identificar una entre varias posibles.

Como un bit puede ser 1 o 0, podremos utilizarlos para seleccionar una información entre dos; con dos bits, una entre cuatro; tres bits una entre ocho; etc. Las posibilidades aumentan como potencias de dos:

Un bit	$2^1$	2 elecciones
Dos bits	$2^2$	4 elecciones
Tres bits	$2^3$	8 elecciones
..	..	..
n bits	$2^n$	$2^n$ elecciones

Si quisiéramos conocer cuántos bits necesitamos para una de ocho situaciones utilizamos logaritmos en base 2:

$$\log_2 8 = 3.-$$

En general el número de bits (**I**) que necesitaremos para poder codificar una determinada cantidad de informaciones (**N**) estará determinado por:

$$I = \log_2 N$$

Como ejemplo, para poder representar en forma binaria los 26 caracteres de nuestro alfabeto necesitaríamos:

$$I = \log_2 26 = 4,7 \text{ bits} \quad I = 5 \text{ bits}$$

## 2) CODIFICACIÓN DE LA INFORMACIÓN EN EL COMPUTADOR

Analizaremos los siguientes Puntos:

### 2.1. Codificación de la información numérica.

#### 2.1.1 Códigos ponderados.

#### 2.1.2 Códigos no ponderados.

### 2.2. Codificación de la información no numérica.

#### 2.2.1. Codificación de los caracteres.

#### 2.2.2 codificación de las instrucciones.

### 2.1 Codificación de la información numérica

Estudiaremos la representación de los símbolos del sistema decimal de numeración mediante símbolos binarios (BIT).

¿Cuántos bits necesitamos para representar los 10 símbolos del sistema decimal de numeración?

$$I = \log_2 10 = 3,162..... \cong 4$$

Esto nos indica que cualquier código que utilicemos para representar los números del sistema decimal precisará, como mínimo, 4 Dígitos Binarios (Bits). Cuando el código utilice más dígitos que los que necesita lo denominaremos **CODIGO REDUNDANTE**.

#### 2.1.1 Códigos Ponderados

Se denomina código ponderado a aquel que respeta, para la representación de cada dígito decimal, el “peso” que corresponde a cada dígito binario de acuerdo a la posición que ocupa. Ejemplos: BCD (8421), AIKEN(2421), 84-2-1.

**NOTA I:** En cualquiera de los códigos numéricos (ponderados y no ponderados) un número decimal se codificará cada dígito decimal por separado.

**Ejemplo:** el número 345 se codificará

	3	4	5		
en BCD	0011	0100	0101	$345_{(10)} =$	001101000101 <sub>(BCD)</sub>
en AIKEN	0011	0100	1011	$345_{(10)} =$	001101000111 <sub>(AIKEN)</sub>

**NOTA II:** si bien los símbolos utilizados se corresponden con los del sistema binario, la representación codificada **no tiene nada que ver** con el SISTEMA BINARIO DE NUMERACIÓN.

**Ejercicios propuestos**

**Codifique los siguientes números decimales utilizando los distintos códigos ponderados.**

**BCD (Pesos 8, 4, 2, 1**

El código **BCD** (DECIMAL CODIFICADO BINARIO) respeta para cada dígito decimal el “peso” que corresponde a cada dígito binario de acuerdo con la posición que ocupa, teniendo en cuenta el “peso” asignado en el sistema binario de numeración.

	<b>8421</b>	<b>8421</b>	<b>8421</b>	<b>8421</b>	<b>8421</b>
70922 <sub>(10) =</sub>	0111	0000	1001	0010	0010
12345 <sub>(10) =</sub>					
68284 <sub>(10) =</sub>					
95135 <sub>(10) =</sub>	1001	0101	0001	0011	0101
35746 <sub>(10) =</sub>					
87453 <sub>(10) =</sub>					
25846 <sub>(10) =</sub>					

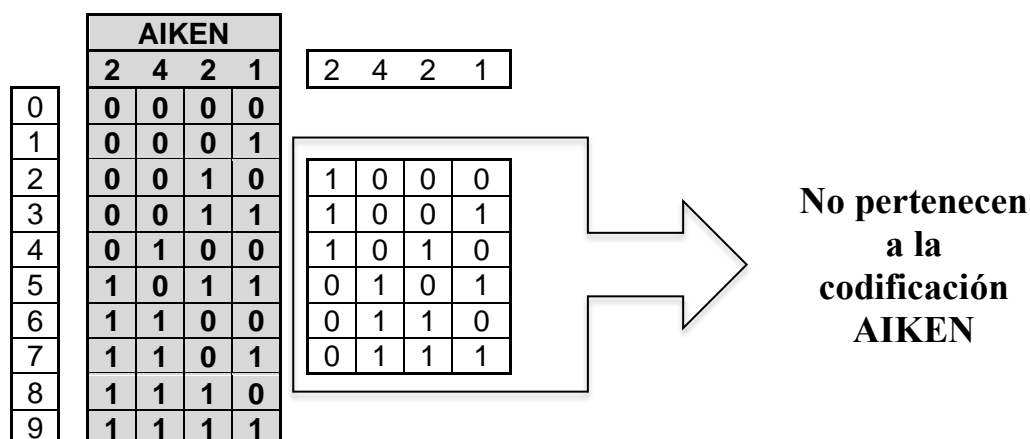
- 11  $\Rightarrow$  0001 0001
- 11  $\Rightarrow$  1011
- **2, 4, 2, 1**

	<b>2421</b>	<b>2421</b>	<b>2421</b>	<b>2421</b>	<b>2421</b>
70922 <sub>(10) =</sub>	0111	0000	1111	0010	1000
12345 <sub>(10) =</sub>		0010			
68284 <sub>(10) =</sub>			1000		
95135 <sub>(10) =</sub>					
35746 <sub>(10) =</sub>	1001	0101	1101	0100	1100
87453 <sub>(10) =</sub>					
25846 <sub>(10) =</sub>	0010				

Aclaración: El código 2, 4, 2, 1, permite dos combinaciones de los dígitos 2, 3, 4, 5, 6 y 7. Cualquiera es válida, más aún, en un mismo número de varios dígitos iguales pueden usarse codificaciones distintas. Ver ejemplo en la tabla anterior.

• **AIKEN (Pesos 2, 4, 2, 1)**

Construcción de la tabla de código AIKEN



	2421	2421	2421	2421	2421
AIKEN					
70922 <sub>(10)</sub> =	1101	0000	1111	0010	0010
12345 <sub>(10)</sub> =	0001	0010	0011	0100	1011
68284 <sub>(10)</sub> =			0010		
95135 <sub>(10)</sub> =					
35746 <sub>(10)</sub> =					
87453 <sub>(10)</sub> =					
25846 <sub>(10)</sub> =	0010				

• **8, 4, -2, -1**

	8 4-2-1	8 4-2-1	8 4-2-1	8 4-2-1	8 4-2-1
70922 <sub>(10)</sub> =	1 0 0 1	0 0 0 0	1 1 1 1	0 1 1 0	0 1 1 0
12345 <sub>(10)</sub> =					
68284 <sub>(10)</sub> =					
95135 <sub>(10)</sub> =					
35746 <sub>(10)</sub> =					
87453 <sub>(10)</sub> =					
25846 <sub>(10)</sub> =					

### • Exceso en tres

La representación en el código exceso de 3, cada dígito decimal se representa como en BCD pero excedido en 3. Ejemplo la representación del 2 es la BCD del 5. No existe representación de 0000

70922 <sub>(10)</sub> =	1010	0011	1100	0101	0101
12345 <sub>(10)</sub> =	0100	0101	0110	0111	1000
68284 <sub>(10)</sub> =		1011			
95135 <sub>(10)</sub> =					
35746 <sub>(10)</sub> =					
87453 <sub>(10)</sub> =					
25846 <sub>(10)</sub> =					

### • 2.1.2. Códigos no Ponderados

En estos códigos la representación de cada dígito decimal es en principio arbitraria o responde a características que no son el “peso” de acuerdo a la posición que ocupan. Ejemplos: Código de GRAY.

#### • Pasos para la construcción de la tabla del código de GRAY.

a. Se colocan los dos bits (0 y 1) y se traza una línea debajo de ellos (espejo), se copian los bits como si se reflejaran en dicho espejo.

		0
		1
		1
		0

b. Se completa la siguiente columna con 1 por debajo del espejo y con 0 por encima del mismo.

	0	0
	0	1
	1	1
	1	0

c. Se repite la operación de reflejado con los cuatro números obtenidos.


	0	0
	0	1
	1	1
	1	0
	1	0
	1	1
	0	1
	0	0

d. Se completa la tercera columna con 1 debajo del espejo y 0 por encima.

	0	0	0
	0	0	1
	0	1	1
	0	1	0
	1	1	0
	1	1	1
	1	0	1
	1	0	0

e. Se vuelve a realizar la operación hasta completar los diez dígitos.

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



0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1

### Ejercicios propuestos.

- Codifique los siguientes números decimales utilizando el código de GRAY.

70922 <sub>(10)</sub> =	0100	0000	1101	0011	0011
12345 <sub>(10)</sub> =					
68284 <sub>(10)</sub> =					
95135 <sub>(10)</sub> =					
35746 <sub>(10)</sub> =					
87453 <sub>(10)</sub> =					
25846 <sub>(10)</sub> =					

- Algunos códigos numéricos más usuales.

Dígito Decimal	B.C.D (8421)	Exceso de tres	A I K E N (2421)	8 4 -2 -1	Código de Gray
0	0000	0011	0000	0000	0000
1	0001	0100	0001	0111	0001
2	0010	0101	0010	0110	0011
3	0011	0110	0011	0101	0010
4	0100	0111	0100	0100	0110
5	0101	1000	1011	1011	0111
6	0110	1001	1100	1010	0101
7	0111	1010	1101	1001	0100
8	1000	1011	1110	1000	1100
9	1001	1100	1111	1111	1101



- **Algunos comentarios sobre los códigos.**

- a. Los códigos Exceso de 3, el 2421 y 84-2-1 son auto complementarios, o sea que el complemento a 9 del número decimal se obtiene cambiando los ceros por unos y los unos por ceros.
- b. El código de Gray o BINARIO REFLEJADO tiene la particularidad que de un dígito decimal al siguiente cambia siempre un solo dígito por vez.

## **2.2. Codificación de la información no numérica**

- **2.2.1. Codificación de los caracteres**

Luego de ver cómo pueden representarse los números, analizaremos ahora como extender el sistema de codificación al conjunto de signos de la máquina de escribir: letras, signos de puntuación, operadores y caracteres especiales.

### **Condiciones que se imponen para la codificación de caracteres**

- a. La representación debe englobar a las de las cifras en una de las formas descriptas (BCD, 2421, etc.) y permitir distinguir las cifras rápidamente de los otros caracteres.
- b. La representación debe permitir añadir nuevos caracteres específicos para una aplicación determinada.
- c. En el caso de las transmisiones, la representación debe incluir un sistema de redundancia que permita la detección de errores.

- **Carácter:**

El concepto de carácter aparece como la cantidad de BITS necesarios para representar uno de los diferentes símbolos del alfabeto (letras, cifras, signos de puntuación, caracteres especiales, etc.).

De acuerdo con el código utilizado cada carácter puede codificarse con un número variable de BITS, pero dentro de un sistema todos los caracteres se representan con el mismo número de bits.

El código **ASCII** se definió inicialmente con 6 bits, esto permitía representar  $2^6 = 64$  caracteres. Posteriormente la ANSI (Instituto Nacional Norteamericano de Normas) definió un nuevo **ASCII** (que se mantiene como norma) de 7 bits. Esta nueva definición permite codificar 128 caracteres; haciéndolo más apto, fundamentalmente para la transmisión donde parte de los "caracteres" codifican funciones de control.

El código **EBCDIC** creado por IBM utiliza 8 bits para representar cada carácter.

El **Unicode** es un código moderno que incluye todos los caracteres de uso común. Actualmente la versión 15.0 contiene 149.186 caracteres provenientes de alfabetos, sistemas ideográficos y colecciones de símbolos (matemáticos, técnicos, musicales, iconos, etc.). La cifra crece con cada nueva versión. Incluye sistemas de escritura modernos como: latino; escrituras históricas extintas, para propósitos académicos, como por ejemplo: cuneiforme, y rúnico. Entre los caracteres no alfabéticos incluidos en Unicode se encuentran símbolos musicales y matemáticos, fichas de juegos como el dominó, flechas, iconos etc.

## • **Palabra**

Una **palabra** es un conjunto de caracteres de longitud fija manejados por un ordenador como unidad de información. El tamaño o longitud de una palabra hace referencia al número de bits contenidos en ella, y es un aspecto muy importante al momento de diseñar una arquitectura de computadora.

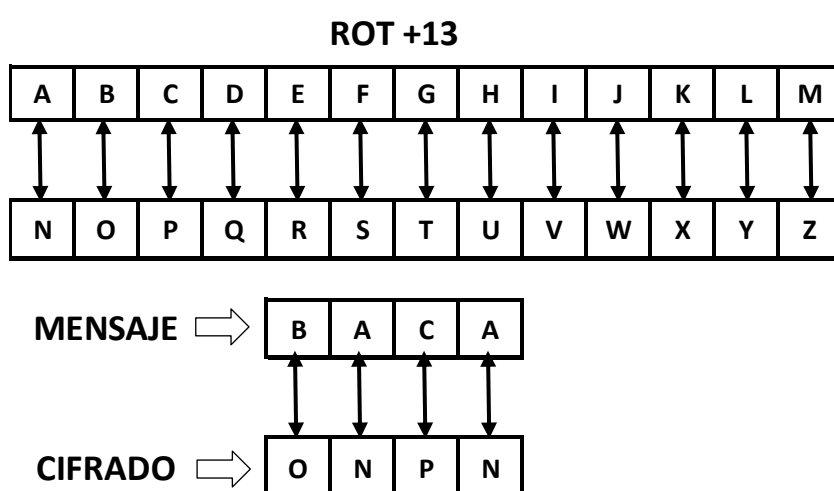
Los ordenadores modernos normalmente tienen un tamaño de palabra de 16, 32 o 64 bits. Muchos otros tamaños se han utilizado en el pasado, como 8, 9, 12, 18, 24, 36, 39, 40, 48 y 60 bits.

Algunos de los primeros ordenadores eran decimales en vez de binarios, típicamente teniendo un tamaño de palabra de 10 o 12 dígitos decimales y algunos de los primeros ordenadores no tenían una longitud de palabra fija.

Algunas veces, el tamaño de una palabra se define para tener un valor particular por compatibilidad con los ordenadores anteriores. Los microprocesadores utilizados en ordenadores personales (por ejemplo, los Intel Pentium y los AMD Athlon) son un ejemplo de esto. Su arquitectura IA-32 (conocido de manera genérica como x86, x86-32 o i386) es una extensión del diseño original del Intel 8086 que tenía un tamaño de palabra de 16 bits. Los procesadores IA-32 siguen soportando programas del 8086 (x86), así que el significado de "**palabra**" en el contexto IA-32 sigue siendo el mismo y se continúa diciendo que son 16 bits, a pesar del hecho de que en la actualidad puede (y especialmente cuando el tamaño del operando por defecto es 32-bit) opera más como una máquina con un tamaño de palabra de 32 bits. Similarmente en la nueva arquitectura x86-64, una "**palabra**" sigue siendo 16 bits, aunque los operandos de 64-bit ("cuádruple palabra") sean más comunes.

## • **El cifrado de Cesar**

En criptografía, el cifrado de César, también conocido como cifrado por desplazamiento, código de César o desplazamiento de César, es una de las técnicas de decodificación más simples y más usadas. Es un tipo de cifrado por sustitución en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto. Por ejemplo, con un desplazamiento de 3, la A sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. Este método debe su nombre a Julio César, quien lo usaba para proteger sus mensajes de miradas no autorizadas.



**•Byte:**

El término BYTE (octeto), se utiliza para describir un conjunto de 8 bits consecutivos que se toman como unidad.

El BYTE muchas veces no es práctico para manipular datos en problemas de programación, recurriéndose al concepto de PALABRA (WORD), de acuerdo a la siguiente relación:

1 BYTE	= 8 bits
1 PALABRA	= 2 Bytes
1 DOBLE PALABRA	= 4 Bytes
1 CUADRUPLE PALABRA	= 8 Bytes
1 DECABYTE	= 10 Bytes

**NOTA III:** El KILOBYTE (KB) es la unidad de capacidad de memoria que representa  $2^{10}$  unidades de información, por lo tanto una memoria de 1 KB podrá almacenar 1024 caracteres.

**•Código EBCDIC (expanded binary code decimal interchange code).**

Este código fue diseñado y utilizado exclusivamente por IBM. Su importancia radica en que sirvió como base para los códigos posteriores normalizados.

Utiliza 8 dígitos binarios para representar cada carácter. La correspondencia entre las informaciones por representar la correspondiente secuencia binaria se encuentra en tablas con distintos formatos.

Con respecto a los caracteres alfabéticos y signos de puntuación no tiene características que lo destaquen, salvo que podemos encontrar pequeñas alteraciones al ser utilizados en países con distintos alfabetos.

**•Representación de la información numérica (EBCDIC).**

Cada dígito decimal es representado internamente con 8 bits, distribuidos de la siguiente forma:

ZONA	DIGITO
4 Bits	4 Bits

**ZONA:** Ocupa los 4 bits de orden superior del Byte, tiene una secuencia binaria fija para cualquier número:  $1111_2 = F_{16}$

**DIGITO:** En este espacio se representa el número decimal codificado en **BCD**

Los datos numéricos codificados en **EBCDIC con zona**, no son técnicamente aptos para ser procesados aritméticamente. Si necesitamos realizar operaciones aritméticas con ellos, se debe eliminar la parte correspondiente a la ZONA de cada byte. Esta operación se llama **empaquetado** y la información resultante, información **empaquetada (empaquetada) o decimal sin zona**. Los datos numéricos con zona se denominan información **desempaquetada o zoneada**.

**EJEMPLO:** Representar el número 36045 en EBCDIC

F	3	F	6	F	0	F	4	F	5
---	---	---	---	---	---	---	---	---	---

Decimal con zona (zoneado)

0	0	0	0	3	6	0	4	5	F
---	---	---	---	---	---	---	---	---	---

Decimal sin zona (empacado)

**Medio byte correspondiente al signo**  
**F o C (positivo), B o D (negativo)**

0	0	0	0	3	6	0	4	5	D
---	---	---	---	---	---	---	---	---	---

- 36,045 (empacado)

En las representaciones internas de memoria, de los números, el punto decimal (coma) no se representa, queda implícitamente considerado en el lugar correspondiente, nunca forma parte física de la cifra, es el programador quien debe tenerlo en cuenta cuando efectúe la salida de los resultados.

• **Código ASCII (American Standard Code for Information Interchange).**

Se trata de un código que utiliza 7 bits para representar cada carácter. Parte de las configuraciones binarias son utilizadas para codificar funciones de control.

La mayor parte de las máquinas actuales utilizan este código normalizado, pero generalmente agregar un octavo bit, que les permite extender el código para representaciones no previstas o para utilizarlo como bit de control de paridad en las transmisiones.

El éxito de este código se basa en que cumple con todas las condiciones impuestas para la codificación de caracteres:

- Utiliza relaciones para establecer el código.
- Los valores correspondientes a las letras de alfabeto y a los restantes caracteres, siguen una secuencia binaria continua, la computadora no tiene que dejar su propio lenguaje binario para realizar operaciones secuenciales con esos caracteres.
- Agrupamiento de las funciones de control, con solo analizar los dos primeros bits de una combinación cualquiera codificada, la computadora puede determinar si se trata de una función de control (dos ceros) o de un carácter (uno de los dos no es cero).

• **Representación de los números.**

En este código, al igual en el EBCDIC, la representación de la información numérica contempla representar los símbolos del sistema decimal de numeración en forma binaria. Cada dígito se codifica con 7 bits que también se encuentran asociados en dos grupos ZONA Y DIGITO.

La zona es siempre 011 y dígito corresponde a la representación BCD del número.

Aquí surge el mismo inconveniente visto en el código EBCDIC, la dificultad para realizar operaciones matemáticas con los números representados con la ZONA, la solución es la misma que la indicada para el código anterior. Como la mayor parte de las máquinas utilizan 8 bits para cada dígito no surgen conflictos al producirse el EMPAQUE.

## Ejercicios propuestos

Completar el siguiente cuadro utilizando las tablas de ASCII y EBCDIC. Tenga en cuenta que la información codificada se refiere a formatos internos de memoria.

Caracteres	EBCDIC		ASCII	
	Zoneado	Emp. c/signo	Hexadecimal	Octal
U.T.N. F.R.Re.				
Arquitectura				
Computadoras				
17/08/2002				
user				
-1298				
+100,001				
	4D40D5F1405D			
	F3F2F7D3			
		007423C		
		0063346D		
			492E532E492E	
			3430313236	
				062063065060
				124145162155151156145

Nota: Tenga en cuenta que en la columna Caracteres la coma (del número +100,001) no se representa si se la trata como número.

### ● 2.2.2. Codificación de las instrucciones

Es el conjunto de códigos y especificaciones que indican como debe ser interpretado el patrón de bits de una instrucción de máquina para logra su ejecución dentro del ordenador. En la mayoría de las arquitecturas, está compuesto por el código de operación y las direcciones o los operandos que la instrucción específica.

Por tal motivo cada arquitectura deberá tener codificada a cada una de sus instrucciones conocidas como "Instrucciones de maquina" con el fin de los lenguajes de alto nivel, sean traducidos e interpretados a este formato, entendible por el ordenador.

<b>Código de Operación</b>	<b>Dirección Operando</b>	<b>Dirección Resultado</b>
----------------------------	---------------------------	----------------------------

### 3. Códigos redundantes

El movimiento de información dentro del computador desde y hacia los distintos órganos que lo componen están sujetos a posibles errores inducidos por el medio utilizado.

Así mismo una comunicación entre distintos ordenadores a través de una red wi fi puede introducir errores debido a interferencias producidas por factores físicos y climáticos.

Por tal motivo Los **códigos redundantes** son utilizados en la detección y corrección de errores en los procesos de transmisión de información binaria. En este contexto, la redundancia se refiere a la inclusión de bits adicionales en los datos transmitidos o almacenados con el fin de detectar y corregir errores.

En la detección de errores, los bits redundantes se utilizan para verificar si ha ocurrido algún error durante la transmisión o almacenamiento de los datos. Se aplican algoritmos específicos, como los códigos de detección de errores, para calcular estos bits de verificación. Si se detecta un error, se puede tomar alguna acción correctiva, como solicitar una retransmisión de los datos.

En la corrección de errores, los códigos redundantes no solo permiten detectar errores, sino también corregirlos. Se utilizan códigos de corrección de errores que agregan bits adicionales a los datos transmitidos o almacenados para permitir la recuperación de errores. Estos códigos tienen la capacidad de detectar y corregir errores hasta ciertos límites establecidos por la capacidad del código utilizado.

En ambos casos, los códigos redundantes desempeñan un papel crucial al brindar una capa adicional de protección contra errores en los datos. Al agregar redundancia, se mejora la fiabilidad y la integridad de la información transmitida o almacenada, lo que es especialmente importante en entornos propensos a errores, como las transmisiones inalámbricas o los medios de almacenamiento físico.

#### 3.1. Códigos autodetectores

Son aquellos Códigos en los que, mediante un determinado número de bits de redundancia, poseen la capacidad de detectar si la información recibida es correcta o no. El ejemplo más clásico de este tipo de códigos es de CONTROL DE PARIDAD.

##### Control de paridad

Este código permite detectar si un mensaje binario posee un error a la hora de su transmisión

Consiste en agregar a los bits de información transmitidos un bit más (generalmente el primero de la izquierda), que hace que la cantidad de unos transmitidos sea PAR (PARIDAD PAR) o IMPAR (PARIDAD IMPAR).

Si bien no permite detectar errores dobles, es el más utilizado debido a su simplicidad y a que en los ordenadores la probabilidad de que ocurra un error es muy pequeña, por lo tanto, que ocurran 2 es mucho menos probable.

#### **Ejercicios propuestos.**

Los siguientes mensajes se verifican mediante un bit de paridad par (de unos) que corresponde al primer bit de la izquierda. Indique si la información es correcta, o no, en cada caso:

	Correcta	Incorrecta
100010100100		
011110001000		
000001000100		
100000001010		
111110010111		
000000000000		
101010101010		
111111111111		
000111000111		

Agregue el bit de paridad impar (de ceros) en los siguientes mensajes. La ubicación del bit de paridad es al final de la cadena de bits.

	Bit Paridad
100010100100	
011110001000	
000001000100	
100000001010	
111110010111	
000000000000	
101010101010	
111111111111	
000111000111	

### 3.2. Códigos autocorrectores

Mediante el uso de estos códigos, el receptor puede determinar si la información recibida es correcta o no y en este caso corregir el error producido durante la transmisión.

#### Control 2 en 3

Para transmitir una información cualquiera de “n” bits, se envían 3 veces esos “n” bits, en forma sucesiva. El receptor de la información, al efectuar el análisis de la misma, pueden presentársele tres situaciones distintas:

- Las tres son idénticas. La información se toma como correcta.
- Dos son iguales y una distinta. El código se comporta como **AUTOCORRECTOR**, selecciona una de las dos iguales y la toma como correcta.
- Las tres son distintas. El código se comporta como **AUTODETECTOR**, la máquina detecta que hay error, pero no puede determinar cuál es la información correcta.

#### **Ejercicios propuestos**

Indicar en la siguiente tabla, el carácter ascii correspondiente a cada mensaje recibido en binario con codificación 2 en 3. En caso de no ser posible la corrección, indicarlo como retransmisión.

2 en 3	ASCII
100000110000011000001	
100110110011011001100	
100111110011111001111	
100111110011001001101	
100111110011101001111	
110011011001111100111	
111000111100001110000	
111101011110001111011	

**Control de paridad entrelazada**

Es un sistema de codificación que permite detectar y corregir uno o más errores de una cadena de caracteres, aplicando la paridad vertical (PV) y horizontal (PH) de la misma.

Del lado del receptor, se determina PV y PH agregando una nueva columna a la matriz binaria de decodificación de la palabra a transmitir

PH es la paridad par de unos a nivel de fila mientras que PV es la paridad par de unos a nivel de columna.

A continuación, se determina el bit de paridad par de unos a nivel de PV y PH, con el objeto de poder verificar que la cadena PH y PV haya llegado de manera correcta.

Del lado del receptor, se verifica el cumplimiento de la paridad par de unos a nivel Fila contra Columna. Aquellas intersecciones que no la cumplan serán interpretados como erróneos y cambiado su valor por el opuesto (corrección).

Si se detecta más de un error de paridad a nivel fila o columna, el error no podrá ser corregido por lo que será necesaria la retransmisión completa de la información.

Si bien este sistema tiene un alto grado de eficacia, consume muchísimo ancho de banda, siendo utilizado para aplicaciones especificados donde dicho impacto no implique un costo adicional al proceso.

D	o	l	a	r
1	1	1	1	1
0	1	1	1	1
0	0	0	0	1
0	1	1	0	0
1	1	1	0	0
0	1	0	0	1
0	1	0	1	0

ASCII

Se construye una matriz debajo de la cadena a transmitir, con el código ascii correspondiente a cada carácter

D	o	l	a	r	PH
1	1	1	1	1	1
0	1	1	1	1	0
0	0	0	0	1	1
0	1	1	0	0	0
1	1	1	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
PV	0	0	0	1	0

Se determina el bit de paridad par de unos para PV y PH

D	o	l	a	r	PH
1	1	1	1	1	1
0	1	1	1	1	0
0	0	0	0	1	1
0	1	1	0	0	0
1	1	1	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
PV	0	0	0	1	0

Se determina el bit de paridad par de unos para PV y PH

1	1	1	1	1	1
0	1	1	1	1	0
0	0	0	0	1	1
0	1	1	0	0	0
1	1	1	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	0	0	1	0	0

Se transmite el bloque completo serializándolo fila por fila.



**Del lado del receptor:**

Suponiendo un error de transmisión sobre los caracteres **o** y **a** de la palabra **Dolar**, la verificación de paridad par de unos a nivel fila y columna podría corregirlo como se observa a continuación:

	D	m	I	U	r	PH
1	1	1	1	1	1	1
0	1	1	0	1	1	0
0	0	0	1	1	1	1
0	1	1	0	0	1	0
1	1	1	1	1	1	1
0	0	0	0	1	1	0
0	1	0	1	0	1	0
PV	0	0	0	1	0	0

Se verifica que los bits de PH y PV cumplan con Paridad Par de unos, caso contrario los bits de control tienen errores y se requiere la retransmisión de todo el bloque

	D	m	I	U	r	PH
1	1	1	1	1	1	1
0	1	1	0	1	1	0
0	0	0	1	1	1	1
0	1	1	0	0	1	0
1	1	1	1	1	1	1
0	0	0	0	1	1	0
0	1	0	1	0	1	0
PV	0	0	0	1	0	0

Se barren filas y columnas verificando la paridad. Aquellas intersecciones de PV y PH que no lo cumplan, indican que el bit es erróneo y debe ser cambiado por su opuesto

	D	o	I	a	r	PH
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	0	0	1	1	1
0	1	1	0	0	1	0
1	1	1	0	0	1	1
0	1	0	0	1	1	0
0	1	0	1	0	1	0
PV	0	0	0	1	0	0

Finalmente la cadena transmitida que contenía dos errores (en distintas posiciones fila/columna) queda corregida

**Ejercicios propuestos**

Armar las matrices binarias con codificación entrelazada para los siguientes mensajes:

1) Circuito

2) Cálculos

3) Ascensor

4) Rebelión

Dadas las siguientes matrices, aplicar código entrelazado del lado del receptor y decodificar el mensaje ascii. En caso de error corregirlo. Indicar si es necesaria la transmisión de todo el bloque o es posible la corrección, según el caso.

1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	0
1	0	0	0	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	0	0	0	1	0
1	0	0	0	1	0	1	1
1	1	0	1	0	1	1	1
0	1	0	1	0	0	0	0

1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0
0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0
0	0	0	0	0	1	1	1
1	0	1	1	0	1	1	1
1	1	0	1	1	1	1	1
1	1	1	0	0	0	1	0

1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	0
1	0	1	0	0	0	1	1
0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	1
0	0	1	1	0	1	0	0
0	1	1	1	1	0	1	1
0	0	0	1	0	0	1	0

### **Código de hamming.**

Estos códigos permiten detectar y corregir un solo error producido durante la transmisión para palabras de cualquier número de bits.

Si se añaden junto al mensaje más bits detectores-correctores de error y si esos bits se pueden ordenar de modo que diferentes bits de error producen diferentes resultados, entonces los bits erróneos podrían ser identificados. En un conjunto de siete bits, hay solo siete posibles errores de bit, por lo que con tres bits de control de error se podría especificar, además de que ocurrió un error, en qué bit fue.

Hamming desarrolló una nomenclatura para describir el sistema, incluyendo el número de los bits de datos y el de los bits detectores-correctores de error en un bloque.

También estudió los problemas que surgían al cambiar dos o más bits a la vez y describió esto como "distancia" (ahora llamada distancia de Hamming en su honor). La paridad tiene una distancia de 2, dado que cualquier error en dos bits no será detectado. La repetición (3.1) tiene una distancia de 3, pues son necesarios el cambio simultáneo de tres bits para obtener otra palabra de código. La repetición (4.1) (cada bit se repite cuatro veces) tiene una distancia de 4, así que el cambio de dos bits en el mismo grupo quedará sin definir.

### **Proceso de Codificación**

Analizaremos el método para construir un código de Hamming para corregir un solo error. El primer paso consiste en determinar, para un código de "i" dígitos binarios de información, cuantos bits de control de paridad "p" son necesarios para detectar y corregir un error único.

Si consideramos que tenemos i dígitos de información y p bits de control y con la premisa que los casos que se pueden presentar son de ningún error o un solo error por vez, tendremos  $i + p + 1$  condiciones que deben identificarse usando los p bits de control de paridad.

Teniendo en cuenta que con p bits podemos formar  $2^p$  combinaciones, entonces la cantidad de bits de control de paridad debe ser tal que satisfaga que  $2^p \geq i + p + 1$

Mediante esta expresión podemos construir la siguiente tabla:

i	1	2	3	4	5	6	7	8	9	10	11	12	.....	Bits de información
p	2	3	3	3	4	4	4	4	4	4	4	5	.....	Bits de control
n	3	5	6	7	9	10	11	12	13	14	15	17	.....	Bits del mensaje

**La forma en que se distribuyen los bits de información y los de control, para conformar el mensaje, es en principio arbitraria.**

Analizaremos una distribución en el mensaje de la siguiente forma, considerando un mensaje de 4 bits de información y que por lo tanto necesitará 3 bits de control:

Posición mensaje	7	6	5	4	3	2	1
	$i_3$	$i_2$	$i_1$	$p_2$	$i_0$	$p_1$	$p_0$

**Los bits de control se colocan en las posiciones que corresponden a las potencias de dos**

Se desea que los dígitos de control (en nuestro ejemplo 3) que indican el resultado del test de paridad sobre los dígitos de paridad, den (en binario) la posición del dígito erróneo.

Para determinar el valor que tomará cada bit de paridad se construye una tabla de combinaciones binarias de tantas columnas como la cantidad de bits de paridad determinada para la codificación. Para el caso anterior serían 3 columnas ( $p_0$ ,  $p_1$  y  $p_2$ )

	$p_2$	$p_1$	$p_0$
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

**Cada bit de paridad controla la posición que coincide con un 1 en su columna en la tabla binaria. Es decir:**

$p_0$  controla las posiciones: 1, 3, 5 y 7

$p_1$  controla las posiciones: 2, 3, 6 y 7

$p_2$  controla las posiciones: 4, 5, 6 y 7

*Para mayor cantidad de bits de paridad se agregan más columnas a la tabla. El bit de paridad debe hacer que la cantidad de bit unos controlados sea par.*

**NOTA IV:** Para el caso de un byte (8bits) de información, corresponde un mensaje de 12 bits.

### • Control y corrección

Cuando el mensaje es enviado, el receptor calcula el valor de los bits de control de la siguiente forma:

$$c_0 = 1 \oplus 3 \oplus 5 \oplus 7$$

$$c_1 = 2 \oplus 3 \oplus 6 \oplus 7$$

$$c_2 = 4 \oplus 5 \oplus 6 \oplus 7$$

**NOTA V:** los números corresponden a la posición de cada bit utilizado para el cálculo de los bits de control (ej. "3" corresponde al bit ubicado en la posición 3) y el símbolo  $\oplus$  designa la operación OR exclusiva.

Si el bit de control es igual a 0 es correcto, si es 1 es incorrecto. La posición del error se determina según los valores obtenidos en los bits de control ordenados en forma decreciente, es decir  $c_2$ ,  $c_1$  y  $c_0$ . Si todos los bits de control son igual a cero no se detecta error en el mensaje.

**Ejemplo:**

Información a transmitir:

1 1 0 1

$$p = 3 \quad (2^p \geq i + p + 1)$$

7	6	5	4	3	2	1
1	1	0	—	1	—	—
$i_3$	$i_2$	$i_1$	$p_2$	$i_0$	$p_1$	$p_0$

 $p_0 : 1,3,5,7 \Rightarrow 0, 1, 0, 1$  (para mantener la paridad par toma el valor 0)

 $p_1 : 2,3,6,7 \Rightarrow 1, 1, 1, 1$  (para mantener la paridad par toma el valor 1)

 $p_2 : 4,5,6,7 \Rightarrow 0, 0, 1, 1$  (para mantener la paridad par toma el valor 0)

Mensaje transmitido:

7	6	5	4	3	2	1
1	1	0	0	1	1	0

Supongamos que al receptor llega el siguiente mensaje: 1 1 0 1 1 1 0

Al calcular, el receptor, los bit de control determina:

$$c_0 = 1 \oplus 3 \oplus 5 \oplus 7 = 0 \oplus 1 \oplus 0 \oplus 1 = 0 \quad \text{correcto}$$

$$c_1 = 2 \oplus 3 \oplus 6 \oplus 7 = 1 \oplus 1 \oplus 1 \oplus 1 = 0 \quad \text{correcto}$$

$$c_2 = 4 \oplus 5 \oplus 6 \oplus 7 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 \quad \text{incorrecto}$$

 $100_{(2)} = 4$  Indica la posición del error. $c_2 c_1 c_0$ 

7	6	5	4	3	2	1
1	1	0	0	1	1	0

Error en pos. 4 (se cambia el valor del bit)

**Ejercicios propuestos.**

Codifique utilizando el código de HAMMING.

	$i_8$	$i_7$	$i_6$	$i_5$	$i_4$	$p_3$	$i_3$	$i_2$	$i_1$	$p_2$	$i_0$	$p_1$	$p_0$
011010001													
101010101													
100000010													
111100010													
111000111													

Verifique si los siguientes mensajes codificados según el método de HAMMING, son correctos. En caso de que exista error corregirlo.

	<b>Posición Error</b>	<b>Mensaje Corregido</b>
<b>100010010001011</b>		
<b>011001101100110</b>		
<b>000001111000000</b>		
<b>110011001100110</b>		
<b>001111110001111</b>		