Arquitectura y Sistemas Operativos - Resumen de Codificación

Conceptos de Bit, Byte, Nibble, KByte, MByte, GByte y TeraByte

Bit: Es la menor porción de memoria que puede existir y consta sólo de un 0 ó un 1.
Byte: Grupo de 8 bits (representación de un carácter, número o signo de la tabla ASCII).
Nibble: es la expresión que equivale a la mitad de un byte.
KiloByte: 1024 bytes = 1 KByte
MegaByte: 10242 bytes = 1 MByte = 1.048.576 bytes.
GigaByte: 10243 bytes = 1 GByte = 1,0737418 x 109
TeraByte: 10244 bytes = 1 TByte = 1,0995116 x 1012

Métodos de codificación

BCD (Binary Coded Decimal)

Cada dígito se representa con un conjunto de bits.

Ponderados

Cada posición tiene un peso, y para convertir un decimal a una combinación binaria, se suman los pesos de las posiciones valor 1.

Número Decimal	Código BCD Natural 8421	Código BCD Aiken 2421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110
9	1001	1111

Por ejemplo:
El número decimal **59** se representa en BCD 8421 como:
$5 = 0101, 9 = 1001 \rightarrow$ 59 en BCD 8421 es **0101 1001**.

Y para representar **59** en código BCD 2421:
$5 = 1011, 9 = 1111 \rightarrow$ 59 en BCD 2421 es **1011 1111**.

No ponderados

Las posiciones binarias no tienen asignado un peso.

BCD Exceso-3

Se suma 3 a cada dígito decimal antes de convertirlo a binario en BCD Natural.

Decimal	BCD Exceso-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

BCD Gray

Solo un bit cambia entre números consecutivos.

DEC	BIN	GRAY
0	0000	0000
1	0001	0001
2	0010	0011

DEC	BIN	GRAY
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Pasaje de binario a Gray

1. Se copia el primer dígito (izquierda).
2. Se compara el primero con el segundo dígito (consecutivo a la derecha) si son iguales se pone un 0 y sino un 1 (XOR entre cada par de bits) en el bit con el que se comparó (a la derecha)
3. Se compara el segundo con el tercero y así sucesivamente.

Ejemplos:
Para el número 0010:
El resultado es 0011.

BIN	Comparación	GRAY
0	Se copia tal cual	0
0	0 XOR 0	0
1	0 XOR 1	1
0	1 XOR 0	1

Para el número 11010:
El resultado es 10111.

BIN	Comparación	GRAY
1	Se copia tal cual	1
1	1 XOR 1	0
0	1 XOR 0	1
1	0 XOR 1	1
0	1 XOR 0	1

Código Jhonson

Colo un bit cambia a la vez.

Estructura:

- Para n bits, el código Johnson tiene $2n$ estados.
- El primer estado es "000...0" (n ceros), y se recorre por un ciclo que va cambiando uno de los bits de cero a uno de derecha a izquierda hasta llegar a "111...1" (n unos), y luego se invierte el orden hasta llegar de nuevo a "000...0".

Por ejemplo:
Para $n = 3$:
Hay $2n$ estados, es decir, 6.

Decimal	Jhonson
0	000
1	001
2	011
3	111
4	110
5	100
6	000 (de nuevo)

Para $n = 9$:
Hay $2n$ estados, es decir, 18.

Decimal	Jhonson
0	000000000
1	000000001
2	000000011

Decimal	Jhonson
3	000000111
4	000001111
5	000011111
6	000111111
7	001111111
8	011111111
9	111111111
10	111111110
11	111111100
12	111111000
13	111110000
14	111100000
15	111000000
16	110000000
17	100000000
18	000000000 (de nuevo)

Bit de paridad

Bit de paridad par

Se agrega un bit de paridad par para verificar si el número de posiciones con valor `1` en el código BCD es **par** o **impar**, dependiendo de si se usa **paridad par** o **paridad impar**.

Ejemplo:
Para el dígito decimal 5, cuyo código BCD en 8421 es 0101.

- Número de bits 1 en el código 0101: hay **dos** bits 1, lo que ya es un número par.
- Entonces, el bit de paridad par será 0.
  Resultado final: **0101 0**

Para el dígito decimal 7, el código BCD en 8421 es 0111.

- Número de bits 1 en el código 0111: hay **tres** bits que valen 1, lo que es un número impar.
- Se agrega un bit de paridad 1 para hacer que el número total de bits 1 sea par.
  Resultado final: **0111 1**

Bit de paridad impar

Se agrega un bit de paridad impar para que el número total de bits 1 (incluyendo el bit de paridad) sea **impar**. Si el número de bits 1 en el código BCD es par, se agrega un bit de paridad 1 para convertirlo en impar. Si ya es impar, se agrega un bit de paridad 0.

Ejemplo:
Para el dígito decimal 5, cuyo código BCD en 8421 es 0101:
Número de bits 1 en el código 0101: hay **dos** bits 1, lo que es par.
Se agrega un bit de paridad impar 1 para hacer que el número total de bits 1 sea impar.
Resultado final: **0101 1**.

Para el dígito decimal 7, el código BCD en 8421 es 0111:
Número de bits 1 en el código 0111: hay **tres** bits 1, lo que ya es impar.
Se agrega un bit de paridad impar 0 para mantener el número total de bits 1 impar.
Resultado final: **0111 0**.

Numero Decimal	BCD Natural	Bit de Paridad par
0	0000	0
1	0001	1
2	0010	1
3	0011	0
4	0100	1
5	0101	0
6	0110	0
7	0111	1
8	1000	1
9	1001	0

Numero Decimal	BCD Exceso 3	Bit de Paridad impar
0	0011	1
1	0100	0
2	0101	1
3	0110	1
4	0111	0

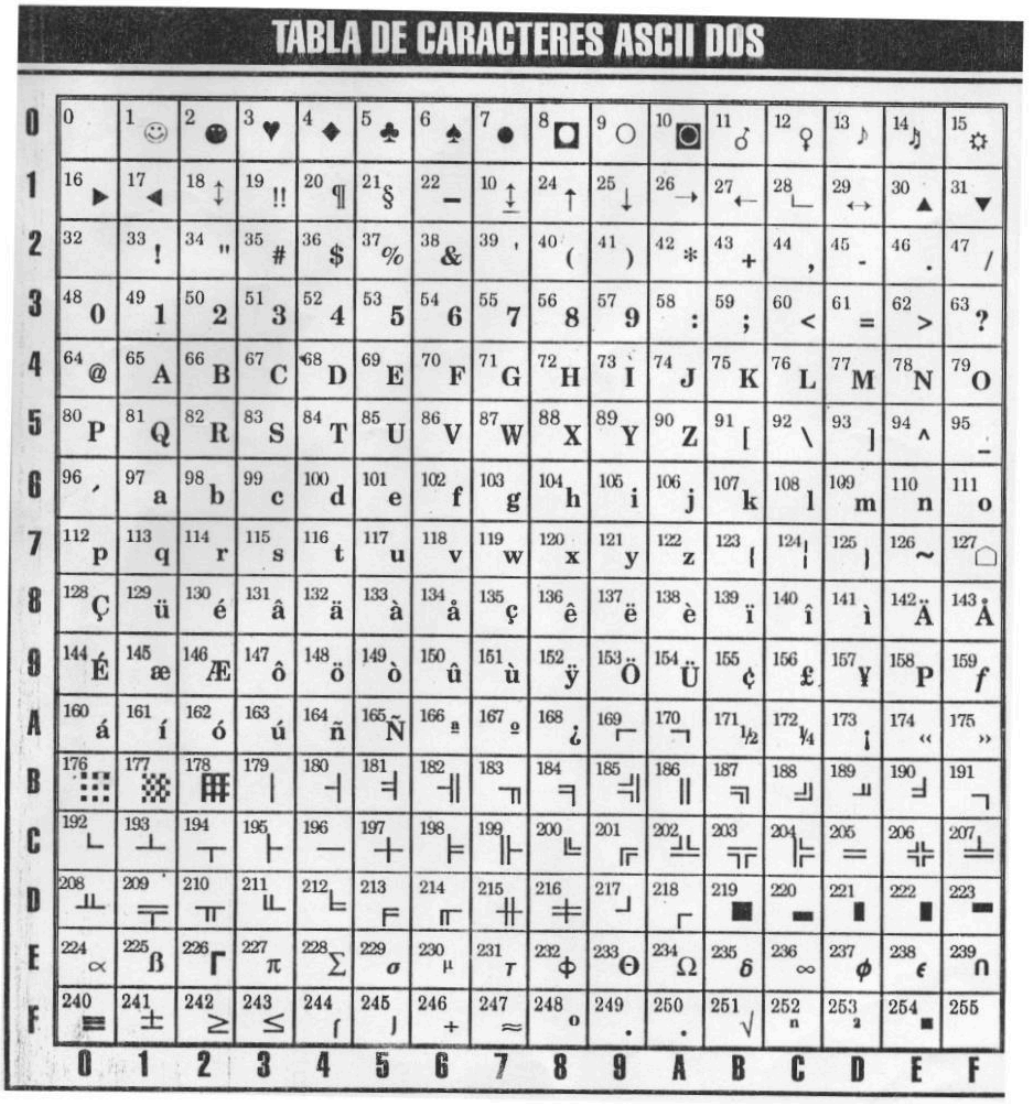
Numero Decimal	BCD Exceso 3	Bit de Paridad impar
5	1000	0
6	1001	1
7	1010	1
8	1011	0
9	1100	1

Códigos alfanuméricos

ASCII

Significados de algunos códigos de control

VALOR ASCII	SIGNIFICADO
0	Nulo
7	Sonido (bip)
9	Tabulación
10	Avance de línea
11	Cursor a inicio
12	Avance a página
13	Enter - Retorno de carro
28	Cursor a derecha
29	Cursor a izquierda
30	Cursor arriba
31	Cursor abajo
32	Espacio
255	Blanco “FF”



Ejemplo:
"Hola mundo" a ASCII es **01001000 01101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101111**

Letra	DEC	BIN ASCII
H	72	01001000
o	111	01101111
l	108	01101100
a	97	01100001
(espacio)	32	00100000
M	77	01001101
u	117	01110101
n	110	01101110
d	100	01100100
o	111	01101111

Código Hamming

Síndrome de Hamming

El **síndrome de Hamming** es el resultado que se obtiene al verificar los bits de paridad en un mensaje codificado con el **código de Hamming**. Este síndrome indica si ha ocurrido un error durante la transmisión de los datos y, en caso de que sea un error de un solo bit, la posición exacta de dicho error. Se dice que hay un **síndrome de error** cuando se detecta un error mediante los bits de paridad.

Codificación

Procedimiento para codificar

Siengo la expresión (n, d) donde n es **n**úmero total de bits y d es la cantidad de bits que son los **d**atos. Se calcula la cantidad de bits de paridad (o redundancia) restando $n - d$.

Para un mensaje de datos de longitud d, se añaden r bits de paridad de acuerdo con la relación:

$$2^r \geq d + r + 1$$

Según el número de bits que hayan en total (n) se creará una tabla con la misma cantidad de columnas y el número de la columna se representará en binario. Las filas de dicha tabla serán: una para indicar la posición, otra destinada al número original, las que sean necesarias para los bits de paridad $n - d$, y una última para el número codificado.

En las columnas, las que sean potencias de dos serán para los bits de paridad, mientras las que no lo sean, serán para los bits de datos.

Los bits de paridad se calculan comparando con XOR (los iguales dan 0 y los desiguales dan 1) entre todos los valores cuya posición en binario tenga el bit equivalente al número de bit de paridad en 1, por ejemplo, si fuera el bit de paridad P2, los bits que se compararían con XOR serían los de las posiciones en las cuales el segundo bit sea 1, como 0010, 0011, 0110 y 0111; y para P1 serían las posiciones 0001, 0011, 0101, 0111. **Se busca siempre la paridad par,** por lo que si en la comparación resulta que son impares, el bit de paridad deberá acomodarse para que siempre sea **par**. En otras palabras, siempre debe haber un número par de bits que valgan 1: **si la cantidad ya es par, el bit de paridad va a ser 0, en cambio, si no es par, el bit de paridad valdrá 1 para que entonces sea par.**

Al final, en la fila del número codificado, se bajan los valores que quedaron.

Ejemplo de codificación

Para $Hamming(7, 4)1101$:
Bits totales: 7.
Bits de datos: 4.
Bits de paridad: 3.

	p1	p2	d1	p3	d2	d3	d4
Posicion	(1) 0001	(2) 0010	(3) 0011	(4) 0100	(5) 0101	(6) 0110	(7) 0111
Número original			1		1	0	1
P1	1		1		1		1
P2		0	1			0	1
P3				1	1	0	1
Número codificado	1	0	1	1	1	0	1

El número codificado en Hamming(7,4) es 1011101

Decodificación y detección de errores

Procedimiento para detectar errores

Dado un número ya codificado, se procede de la forma similar que con la codificación, solo que se recalcula la paridad. Si la paridad calculada en la decodificación es igual a la del número codificado significa que no hay errores, y si es diferente entocnes es que hay un error.

La tabla que se realiza para la decodificación es la misma que para la codificación, pero con tres columnas más: **Cálculo de paridad, Paridad almacenada (del número codificado), y Comprobación.**

Luego de recalcularse la paridad, se colocan los valores en la columna correspondiente, y se añaden los bits de paridad obtenidos en el número codificado en la columna siguente.
Para la comprobación, se comparan el bit de paridad obtenido en el recálculo con el obtenido del número codificado: si son iguales se marca con un 0, que representa que es correcto, y si son distintos, se marca con un 1, que representa un error. Es decir, que se realiza un XOR entre dichos bits de paridad.

Si se detectan errores, se interpreta la columna de comparación como un número binario leído desde abajo hacia arriba $P_n, P_{n-1}, \ldots, P_1$, se trasnforma a decimal y ese número sería el valor del *síndrome de Hamming*.

- **Síndrome = 0**: No hay errores en el mensaje.
- **Síndrome ≠ 0**: Hay un error, y el valor del síndrome indica la posición exacta del bit que debe corregirse, siempre que sea un error de un solo bit. Dicha posición es leída desde el bit más significativo (de **izquierda a derecha**), y para corregirlo se invierte su valor.

Ejemplo de detección de errores

Para 1011101 pero con un error en el penúltimo bit: 1011111

Número codificado (Fila opcional)	1	0	1	1	1	1	1	**Cálculo de Paridad**	**Paridad Almacenada**	**Comprobación**
	p1	p2	d1	p3	d2	d3	d4			
Posicion	(1) 0001	(2) 0010	(3) 0011	(4) 0100	(5) 0101	(6) 0110	(7) 0111			
Número original			1		1	1	1			
P1	0		1		1		1	0	1	Error (1)
P2		1	1			1	1	1	0	Error (1)
P3				1	1	1	1	1	1	OK (0)

Si se leen los valores obtenidos en la comprobación como se mencionó anteriormente, se obtiene el número binario 011, que pasado a decimal es 3, con el cual se puede determiar que el error está en el tercer bit, contando desde el más significativo, 1111 quitando los bits de paridad que se encuentran en las posiciones que son potencias de dos. Y para corregir el error se invierte el valor del bit erróneo, quedando 1101 como el número decodificado.

Código Hamming Extendido

El código de Hamming estándar **no puede corregir errores de dos bits**, porque la información proporcionada por los bits de paridad es insuficiente para distinguir entre las diferentes combinaciones de dos bits erróneos.

Detección de errores de dos bits

Para corregir errores de dos bits, se puede utilizar un esquema de **código Hamming SECDED** (Single Error Correction, Double Error Detection). Este esquema añade un bit de paridad adicional a los datos codificados con Hamming. Este bit de paridad global cubre todos los bits de datos y paridad originales (XOR a todos los bits del número codificado).

Proceso para detectar y corregir errores

- **Detección de un solo error**: El síndrome de error indica la posición del bit erróneo, que puede corregirse. Pueden haber dos casos de esto teniendo en cuenta el bit de paridad adicional.
  - **Código de Hamming correcto pero falla el bit de paridad global (adicional)**:
    Por ejemplo: si se envía 11001100 y se recibe 11001101 solo se corrige el bit de praridad adicional.
  - **Código de Hamming y bit de paridad global incorrectos**: Debe corregirse el código por medio de la tabla.
    Por ejemplo: si se envía 11001100 pero se recibe 11001000
- **Detección de dos errores**: Si el bit de paridad adicional indica un error y el síndrome de Hamming es no nulo, hay dos errores, pero no pueden corregirse. Si el síndrome es 0 y el bit de paridad adicional indica un error, también se ha producido un error de dos bits.
  Por ejemplo: se envía 11001100 y se recibe 11000000
  - **Corrección**: Solo se corrige un error de un solo bit. Si se detectan dos errores, el sistema puede solicitar una retransmisión o indicar que no puede corregir el error.