Código de Hamming.

La transferencia de datos en la memoria puede en ocasiones tener errores que se producen por variaciones en la corriente eléctrica. Una forma de detectar esos errores es usando algún código de detección de errores, o bits de paridad, los cuales se agregan ala dato inicial para poder controlar alguna desviación. Es una forma de funcionamiento análogo al dígito de verificación que se utilizan en los legajos de los alumnos.

Primero se añaden tantos bits de paridad como sean necesarios para formar una palabra que sea igual a:

m + r = n bits

Siendo m la cantidad de bits dados por la palabra del sistema y r la cantidad de bits de paridad que se agregan y n la cantidad de bits resultante.

Los bits se numeran comenzando desde 1 en lugar de hacerlo desde 0 como habitualmente se hace en todas las funciones de computación, siendo por lo tanto el bit 1el de extrema derecha (también llamado bit de mayor significado especialmente en sistemas numéricos).

Todos los bits cuyo número es potencia de 2 son bits de paridad, los demás se utilizan para datos, por lo tanto los bits 1, 2, 4, 8, 16, 32,... etc. son bits de paridad y el resto son datos.

Es importante saber cual es el tamaño de los registros porque así sabremos cuantos bits debemos agregarle. De esta manera si la palabra es de 16 bits deberemos agregarle 5 bits de paridad con lo cual tendremos 21 bits en total, cumpliendo con la formula que dimos al principio:

16 + 5 = 21

Recordamos que podemos tener dos tipos de paridad: **par e impar**, sabiendo que cualquiera sea la que tomemos debemos sumar la cantidad de bits con valor 1 y obtener siempre un número par o impar de acuerdo a la forma elegida.

Los bits verifican los siguientes campos:

Bits	Verifica los siguientes bits	Noten que
1	1 3 5 7 9 11 13 15 17 19 21	1 si; 1 no
2	2 3 6 7 10 11 14 15 18 19	2 si; 2 no
4	4 5 6 7 12 13 1 4 15 20 21	4 si; 4 no
8	8 9 10 11 12 13 14 15	8 si; 8 no
16	16 17 18 19 20 21 22 23 24	16 si; 16 no

Cada bit de paridad comienza de su número para arriba.

En general un bit es verificado por todos aquellos que respondan a: **bp1 + bp2 + bp3 +.... + bpn = b** lo que da, por ejemplo que el bit 5 es verificado por el bit de paridad 1 y el bit de paridad 4 lo que cumple con la fórmula dada anteriormente **bp1 + bp4 = 5** y el bit 11 es verificado por el 1 + 2 + 8 = 11.

Veamos como se controla una palabra de 16 bits en memoria.

Tomemos la palabra en memoria 1100111000010111 y apliquemos el código de Hamming para paridad par.

Nro.Bits	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Cantidad
Hamming	H	H	1	H	1	0	0	H	1	1	1	0	0	0	0	H	1	0	1	1	1	
Para 16																0	1	0	1	1	1	4
Para 8								1	1	1	1	0	0	0	0							3
Para 4				1	1	0	0					0	0	0	0					1	1	3
Para 2		0	1			0	0			1	1			0	0			0	1			4
Para 1	1		1		1		0		1		1		0		0		1		1		1	7
Final	1	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1	12

Si se produce un cambio en el bit 5 deberá ser tomado por el control de los bits 1 y 4 donde se verificará el cambio o error producido en la transmisión o por una variación de corriente eléctrica.

No querramos controlar con los valores de 1 y 4 porque no es así como se debe hacer. Esto se debe hacer de la misma manera en que se construye cada bit. Iniciamos por el 16 y contamos la cantidad de unos que este bit controla y luego seguimos con el 8, luego el cuatro, el 2 y por último el 1. De la misma manera se hace con el error. Si da impar sabremos que ese control está mal.

Es importante indicar que el código de Hamming sólo puede detectar el cambio de 1 bit. Si fueran dos los que cambiaron sería casi imposible y si fueran 3 totalmente imposible más aún si se compensan.

Andrew S. Tanembaum[1] desarrolla el siguiente ejercicio, que traducimos en un desarrollo igual que el nuestro para que pueda ser seguido detalladamente

Nro.Bits	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Cantidad
Hamming	H	H	1	H	1	1	1	H	0	0	0	0	1	0	1	H	0	1	1	1	0	
Para 16																1	0	1	1	1	0	3
Para 8								0	0	0	0	0	1	0	1							2
Para 4				0	1	1	1					0	1	0	1					1	0	6
Para 2		0	1			1	1			0	0			0	1			1	1			6
Para 1	0		1		1		1		0		0		1		1		0		1		0	6
Final	0	0	1	0	1	1	1	0	0	0	0	0	1	0	1	1	0	1	1	1	0	10

Continuando

Códigos de Hamming.

Es un método general propuesto por R. W Hamming usando una distancia mínima *m*. Con este método, por cada entero *m* existe un código de hamming de 2^m-1 bits que contiene *m* bits de paridad

[1] Andrew S. Tanembaum en su libro **"Organización de Computadoras un Enfoque Estructurado"** Editorial Prentice Hall 3ra. Edición páginas 50 a 54. Aconsejamos leer.

y 2^m-1-m bits de información. En este código, los bits de paridad y los bits de paridad se encuentran entremezclados de la siguiente forma: Si se numeran las posiciones de los bits desde 1 hasta 2^m-1, los bits en la posición 2^k, donde $0 \leq k \leq m-1$, son los bits de paridad y los bits restantes son bits de información.

El valor de cada bit de paridad se escoge de modo que el total de unos en un número específico de bits sea par, y estos grupos se escogen de tal forma que ningún bit de información se cubra con la misma combinación de bits de paridad. Es lo anterior lo que proporciona al código su capacidad de corrección.

Para cada bit de paridad en la posición 2^k, su **grupo de bits de información correspondiente** incluye todos esos bits de información correspondiente cuya representación binaria tenga un uno en la posición 2^k. La siguiente tabla muestra los grupos de paridad para un código de hamming de 7 bits o sea de la forma 2^m-1 con *m* = 3. En este ejemplo, los bits de información son 4 y los bits de paridad son 3. Los bits de información están en las posiciones 7, 6, 5 ,3. Los bits de paridad están en las posiciones 1, 2, 4.

POSICIÓN DE LOS BITS

7	6	5	4	3	2	1
X	X	X	X			
X	X			X	X	
X		X		X		X

En la tabla anterior, el grupo de paridad del bit de paridad situado en la posición 4 son los bits de información situados en las posiciones 7, 6, 5 que contienen unos en la posición 2^k o sea 4 cuando k = 2.

El grupo de paridad del bit de paridad situado en la posición 2 son los bits de información situados en las posiciones 7, 6, 3 que contienen unos en la posición 2^k o sea 2 cuando k = 1.

El grupo de paridad del bit de paridad situado en la posición 1 son los bits de información situados en las posiciones 7, 5, 3 que contienen unos en la posición 2^k o sea 1 cuando K = 0.

Como 111 es la representación binaria de 7, el bit de información en la posición 7 se usa para calcular el valor de los tres bits de paridad. Similarmente, el bit de información en la posición 6 se usa para calcular el valor de los bits de paridad en las posiciones 4 y 2; el bit de información en la posición 5 se usa se usa para calcular el valor de los bits de paridad en las posiciones 4 y 1. Finalmente, el bit de información en la posición 3 se usa para calcular el valor de los bits de paridad en las posiciones 2 y 1.

De acuerdo con estos grupos de paridad, el valor del bit de paridad de la posición 1 tiene que elegirse de modo que el número de unos en las posiciones 7, 5, 3, 1 sea par, mientras el bit de paridad en la posición 2 hace el número de unos par 7, 6, 3, 2 y el valor del bit de paridad en la posición cuatro hace el número de unos par en las posiciones 7, 6, 5, 4.

Es fácil observar que, en estas condiciones, la distancia mínima es 3, o sea que tienen que haber al menos tres cambios de un bit para convertir una palabra de código en otra.

Para probar que un cambio de un bit siempre genera una palabra que no pertenece al código, hay que observar que un cambio de un bit en una palabra del código afecta al menos un bit de paridad.

Por otra parte, un cambio de dos bits en una palabra del código no cambia el valor del bit de paridad si ambos bits pertenecen al mismo grupo de paridad. Sin embargo ello no es posible ya que para dos posiciones cualquiera de una palabra del código siempre hay un grupo de paridad que no incluye ambas posiciones. En otras palabras, como dos bits cualquiera deben estar en distintas posiciones, sus números binarios deben diferir al menos en un bit, así que siempre hay al menos un grupo de paridad con un solo bit cambiado, lo cuál da lugar a una palabra que no pertenece al código con al menos un valor de paridad incorrecto.

Ejemplo 3.

Supóngase que se transmite una palabra de código y se recibe una palabra que no pertenece al código y que es 1110101 . Cuál fue la palabra correcta transmitida?

Posiciones de los bits.

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

En la tabla anterior se puede observar lo siguiente:

Cuando se cuenta el número de unos que hay en los bits, 7, 6, 5, 4 de la palabra del código recibida, se encuentra que este número es impar. De forma similar, se encuentra que los bits 7, 6, 3, 2 contienen un número0 impar de unos. Por tanto hay un error en los bits de paridad 4 y 2. Como la suma de los números en esas posiciones es 6, se sabe que el error se ha producido en el bit de posición 6 y por tanto la palabra transmitida fue 1010101.

EJERCICIOS

1) Defina un código de 4 bits para la representación de dígitos decimales, con la propiedad de que las palabras de código para dos dígitos cualquiera cuya diferencia sea uno, difieran sólo en una posición de bits, y que esto también se cumpla para los dígitos 0 y 9.

3) Determinar la distancia entre X y Y en:

a) X = 1100010 y Y = 1010001

b) X = 0100110 y Y = 0110010

c) X = 00111001, y Y = 10101001

4) Determine la distancia mínima de la función de decodificación

e= (000)= 00000000

e= (001)= 01110010

e= (010)= 10011100

e= (011)= 01110001

e= (100)= 01100101

e= (101)= 10110000

e= (111)= 00001111

Cuantos errores detectará e?

Cuantos errores corregirá e?

Bibliografia Youtube

https://www.youtube.com/watch?v=gQK9nROFX20
https://www.youtube.com/watch?v=Y5omFghds4U