I. SISTEMAS NUMÉRICOS

1.1. Introducción

Para el estudio de los sistemas numéricos se debe tener presente que existen diversas formas de registrar las cantidades, mediante diferentes notaciones.

Desde el punto de vista de lo que nosotros acostumbramos, o sea el sistema decimal, éste corresponde a una notación posicional, en la cual cada expresión numérica, es representada en forma única, según una cierta combinación de dígitos.

Existen otros sistemas, en los cuales la notación no es posicional, el significado depende de la convención adoptada.

Los sistemas numéricos posicionales (Los cuales permiten operar bajo un cierto esquema), se caracterizan por una base "r" y un conjunto de dígitos (0, 1, 2, 3, ..., r-1)

1.1.1 Sistema decimal

Base = 10

Conjunto de dígitos (0, 1, 2, 3, ..., 9)

Ejemplo 1:

$1999 = 1*10^3 + 9*10^2 + 9*10 + 9*10^0$

$N_r \geq 1 \rightarrow$ Entero

$N_r = a_1 r^{n-1} + a_2 r^{n-2} +,\ldots\ldots\ldots\ldots, + a_n r^0 = \sum_{i=1}^{n} a_i r^{n-i}$

n = Nro. de dígitos enteros

Ejemplo 2:

$0.1999 = 1*10^{-1} + 9*10^{-2} + 9*10^{-3} + 9*10^{-4}$

$0 < N_r < 1 \rightarrow$ fraccionario

$N_r = a_{n+1}r^{-1} + a_{n+2}r^{-2} +,\ldots\ldots\ldots\ldots, + a_{n+m}r^{-m} = \sum_{i=n+1}^{n+m} a_i r^{n-i}$

En general $N_r = \sum_{i=1}^{n+m} a_i r^{n-i}$

n = Nro. de dígitos enteros
m = Nro. de dígitos fraccionarios
a_i = {0, 1, 2, ..., r-1}

1.1.2 Sistema Binario

Base = 2

Conjunto de dígitos (0, 1)

Ejemplo

$1001_{(2)} = 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 9_{(10)}$

$0.1001_{(2)} = 1*2^{-1} + 0*2^{-2} + 0*2^{-3} + 1*2^{-4} = 0.5625_{(10)}$

1.1.3 Sistema Hexadecimal

Base = 16

Conjunto de dígitos (0, 1, ..., 15)

Para designar los números superiores a 9 se usan los caracteres A, B, C, D, F.

Ejemplo

$F63_{(16)} = 15*16^2 + 6*16^1 + 3*16^0 = 3939_{(10)}$

$0.1001_{(2)} = 1*2^{-1} + 0*2^{-2} + 0*2^{-3} + 1*2^{-4} = 0.5625_{(10)}$

1.2. Conversión entre bases

$N_{(10)} \rightarrow N_{(r)}$ Dos Casos

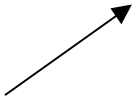
I) $N_r = a_n r^0 + a_{n-1}r^1 +,\ldots\ldots\ldots\ldots, a_1 r^{n-1}$ $N_r \geq 1$

II) $N_r = a_{n+1}r^{-1} + a_{n+2}r^{-2} +,\ldots\ldots\ldots\ldots, a_{n+m}r^{-m}$ $0 < N_r < 1$

Caso A : $N_r \geq 1$

$N_r = a_{n+} r(a_{n-1}+r(a_{n-2}+r(.....))))$.....

$N_r /r= a_n /r+ (an_{-1}+r(a_{n-2}+r(.....))))$.....)



Parte Entera

Con esto se conoce a_n

Si se desea conocer a_{n-1} , se divide la parte entera por r, etc.

Ejemplo

Convertir 1428 en base 10 a la base 2 y a la base 8

1428	2	
714	0	→ Dígito LSB
357	0	
178	1	

Luego,

1428_{10} ----------> 10110010100

Caso B : $0 < N_r < 1$

$N_r = a_{n+1}r^{-1} + a_{n+2}r^{-2} +$,....., $a_{n+m}r^{-m}$ $0 <N_r <1$

$N_r *r= a_{n+1} + a_{n+2}r^{--1} +$,....., $a_{n+m}r^{-m+1}$

Parte Entera conocida

Con esto se conoce a_{n+1}

Si se desea conocer a_{n+2} , se multiplica la parte fraccionaria por r, etc.

Ejemplo

Convertir 0.7895 en base 10 a la base 2 y a la base 8

0.	7895 * 2	
1	579	MSB
1	158	

Luego 0.7895 $_{10}$ ----------> 0.11001

N_r N_{10}

$$N_r = \sum_{i=1}^{n+m} a_i r^{n-i}$$

Ejercicios:

Convertir N_{16} = 37F.AB a bases 10, 8, 2.

1.3. Complemento.

Los complementos se usan en los computadores digitales para simplificar las operaciones de sustracción y para manipulaciones lógicas.

1.3.1. Complemento r de un número N.

Dado un número positivo N en base r con n dígitos enteros. El complemento r de un número N se define como:

$r^n - N$

para N distinto de 0 y 0 para N = 0.

Ejemplo:

Complemento 2 de 11010_2 ---- 26_{10}

$r^n = 2^5$

$100000 - 11010 = 00110_2 = 6_{10}$

Ejemplo

Complemento 10 de $(0.3267)_{10}$

$1-0.3267 = 0.6733_{10}$

Metodo para obtener el complemento 2 de N

- Dejar los ceros menos significativos iguales.
- Dejar el primer bit menos significativo distinto de cero sin cambiar y el resto reemplazar ceros por unos y unos por ceros.

1.3.2. Complemento (r-1) de un número N.

Dado un número N en base r, con n dígitos enteros y una fracción de m dígitos

El complemento r-1 de N se define como: $r^n-r^{-m} - N$

Ejemplo

Complemento 9 de 52520_{10}

$10^5 - 1 - 52520 = 47479$

complemento 1 de 1100_2

$2^4-2^0 -1100 = 10000-1-1100=0011$

Método para obtener el complemento 1 de N.

- Se cambian unos por ceros y ceros por unos
- El complemento r se puede obtener del complemento r-1 sumándole r^{-m}

1.3.3. Representación de los números negativos.

Existen tres formas de representar números negativos

- Signo Magnitud
- Complemento 2
- Complemento 1

- ***Representación en signo magnitud***

Un número tiene distintas representaciones dentro y fuera de una máquina.

En forma externa y en general

$$N = +/- \sum_{i=1}^{n+m} a_i r^{n-i}$$

En forma interna

a_o a_1 a_2 .....a_n a_{n+1},..... a_{n+m}

Signo Entero Fracción

La cantidad (n+m+1) de dígitos que pueden usarse dependerá de las características propias de la máquina digital de que se trate.

Luego

$$N = a_0 r^n + \sum_{i=1}^{n+m} a_i r^{n-i} \ /r^{-n}$$

$$N = a_0 r^o + \sum_{i=1}^{n+m} a_i r^{-i}$$

a_o = 0 para N positivos

a_o = 1 para N negativos

Ejemplo.

Representar el -9 en SM

0.1001 = 9, 1.1001 = -9

- ***Representación binaria en complemento 2***

Ejemplo :

+26 = 0.11010
-26 = 1.00110

- ***Representación binaria en complemento 1***

+26 = 0.11010
-26 = 1.00101

<u>Aritmética no decimal</u>

Base 2

+	0	1
0	0	1
1	1	10

Suma

*	0	1
0	0	0
1	0	1

Multiplicación

Ejercicios.

Construir la tabla en base 4, sumar y multiplicar 4*3

1.4. Operatoria

1.4.1. Aritmética en complemento 2

Ejercicios

1) Sumar dos números positivos

+8+3 = +11
0.1000 +0.0011 = 0.1011

2) Sumar un número positivo y uno negativo

a) +8 -3
+8 = 0.1000
+3 = 0.0011
-3 = 1.1101 8 -3 = 0.1000 + 1.1101 = 1|0.0101

b) -8 + 3

0.0011 + 1.1000 = 1.1011 $_{c2}$

3) Sumar dos números negativos

-3 -8 = 1.1101 + 1.1000 = 1| 1.0101$_{c2}$

1.4.2. Aritmética en complemento 1:

a) +8+3 = 0.1000 + 0.0011 = 0.1011

b) +8-3 = 0.1000 +1.1100 = 1|0.0100 +1 = 0.0101

c) -8 +3 = 1.0111 + 0.0011 = 1.1010$_{c1}$ = 1.0101

d) -8 -3 = 1.0111 + 1.1100 = 1|1.0011 +1 = 1.0100$_{c1}$ = 1.1011

1.5. Códigos

1.5.1. Definiciones

BIT : Dígito binario.

Palabra : Ordenamiento de BIT.

Distancia entre dos palabras: N° de dígitos que hay que cambiar de una palabra a otra.

Código :Un código es una forma sistemática de representar información.

Distancia de un Código : Es la distancia mínima entre dos palabras cualesquiera. En general servirán códigos con igual distancia.

Código cíclico : Es un código en el cual todo par de palabras consecutivas tienen la misma distancia (considerando también la primera y la última).

Código Reflejado : Es aquel en el cual dos palabras equidistantes de su eje de simetría, solo tienen distinto el dígito de mayor ponderación.

1.5.2. Tipos de códigos

a) Código Binario:

Es una forma directa de conversión de los números decimales a binarios.
En un código binario de n bits es posible representar 2^n mensajes

Código Decimal	Código Binario
0	000
1	001
2	010
...	...

b) Código BCD Ponderado (Decimal Codificado en Binario)

El código BCD ponderado utiliza las primeras palabras del sistema binario con distintos pesos. Los más usados son:

- BCD 8421,
- BCD 2421,
- BCD 5211.

Tabla: BCD $_{8421}$.

BCD	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Ejemplos:

$369_{10} \Rightarrow BDC_{8421}$

$0011\ 0110\ 1001_{BCD}$

c) Código Exceso 3:

El código exceso 3 suma tres al número decimal y luego lo convierte a binario. Este código es no ponderado debido a que los "1" no representan valores binarios.

La utilidad del código es que al menos un “1” esta presente en cada estado, dando la posibilidad de determinar un error.

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

Es además un código complementado, lo cual facilita las operaciones de sustracción.

d) Código Gray (Cíclico y Reflejado).

Este código tiene la propiedad de que entre una palabra y la adyacente solo cambia un BIT, lo cual es ventajoso para conversores A/D.

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

TABLAS DE CÓDIGOS

DECIMAL	BINARIO	BCD_{8421}	BCD_{2421}	BCD_{5211}	EX-3	GRAY
0	0000	0000	0000	0000	0011	0000
1	0001	0001	0001	0001	0100	0001
2	0010	0010	0010	0100	0101	0011
3	0011	0011	0011	0110	0110	0010
4	0100	0100	0100	0111	0111	0110
5	0101	0101	1011	1000	1000	0111
6	0110	0110	1100	1001	1001	0101
7	0111	0111	1101	1100	1010	0100
8	1000	1000	1110	1110	1011	1100
9	1001	1001	1111	1111	1100	1101

Conversión BCD_{8421} a Código Gray:

Comenzando desde el BIT menos significativo se realiza una operación OR-EX con el siguiente. Debe agregarse un cero delante de la palabra.

Ej.

56_{BCD} ⇒ Gray 0101 01**10**$_{BCD}$ ⇒ 0111 010**1**$_{CG}$

Conversión Código Gray a BCD_{8421}:

Se comienza por el BIT MSB. Los BIT de mayor orden no se cambian hasta que el primer uno es “pasado”. Se sigue colando “1” hasta que aparece el próximo “1” seguido por ceros hasta el próximo “1”.

Ej.

56_{CG} ⇒ BCD 0111 0101$_{CG}$ ⇒ 0101 0110$_{BCD}$

Códigos Alfanuméricos:

Son códigos utilizados en la representación de caracteres (letras, números, símbolos, caracteres de control, etc.).

- ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange).
- EBCDIC (IBM).
- UNICODE

1.6. Códigos para Detectar y Corregir Errores

1. **Paridad:**

Se trata de detectar un error para volver a retransmitir la información correcta. El sistema más simple es el chequeo de paridad. Se tienen dos casos: Paridad Par y Paridad Impar. Se debe agregar un BIT a la palabra tal que la paridad (Número de "1" de la palabra) sea Par o Impar.

Ejemplo: Codigo BCD_{8421} con paridad

Decimal	a_1 a_2 a_3 a_4	P. Par	P. Impa
0	0 0 0 0	0	1
1	0 0 0 1	1	0
2	0 0 1 0	1	0
3	0 0 1 1	0	1
4	0 1 0 0	1	0
5	0 1 0 1	0	1
6	0 1 1 0	0	1
7	0 1 1 1	1	0
8	1 0 0 0	1	0
9	1 0 0 1	0	1
...	...	...	...

Se ha utilizado el Código BCD_{8421} pero puede usarse cualquier otro. Para verificar, corresponde usar la función OR-EX entre los BIT de una palabra incluyendo el BIT de paridad.

Paridad Par : $a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus P = 0$
Paridad Impar : $a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus P = 1$

a b	a ⊕ b
0 0	0
0 1	1
1 0	1
1 1	0

2. **Por Distancia:**

Otra forma es usar código con una cierta distancia D.

Procedimiento:

Sean las palabras A y B con distancia d=3.

A = 00000 D=3
B = 11100

Para que se confunda una palabra con error de otra del código deben producirse 3 bits cambiados.

La probabilidad de que dos BIT estén errados es bastante pequeña y puede ser no considerada.

En general si d es la distancia del código, entonces pueden detectarse d-1 errores.

3. **Código Hamming:**

Un código Hamming esta formado por bits que dan paridad (ubicados en posiciones dadas por las potencias de 2^n), y el resto de los bits son de información. Los bits de paridad, entregan su paridad a los siguientes conjuntos, determinados como se indican:

	C B A
a_0	0 0 0
a_1	0 0 1
a_2	0 1 0
a_3	0 1 1
a_4	1 0 0
a_5	1 0 1
a_6	1 1 0
a_7	1 1 1

A={ a_1, a_3, a_5, a_7, ... }
B={ a_2, a_3, a_6, a_7, ... }
C={ a_4, a_5, a_6, a_7, ... }

Ejemplo.

Sea un mensaje de 7 bits. ¿Cuantos bit son de paridad y los cuantos de información?

$\mathbf{a_1}\ \mathbf{a_2}\ a_3\ \mathbf{a_4}\ a_5\ a_6\ a_7$

↳ bits de paridad, el resto es información.

Ejemplo.

Codificar el número 5_{10} en BCD_H

$$5_{10} \Rightarrow (0\ 1\ 0\ 1)_{BCD}$$

como:

$a_1 = 0$ $\qquad$ $a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 0$
$a_2 = 1$ $\qquad$ $a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 0$
$a_4 = 0$ $\qquad$ $a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 0$

El número en BCD_H seria 0 1 0 0 1 0 1 .

$$5_{10} \Rightarrow (0\ 1\ 0\ 1)_{BCD} \Rightarrow (0\ 1\ 0\ 0\ 1\ 0\ 1)_{BCD\text{-}H}$$

El proceso de detección de un error se realiza chequeando la paridad de los conjuntos indicados.

Supongamos que a_5 se recibió como 0.

Por lo tanto

a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	1	0	0	0	0	1

$A = a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 1$
$B = a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 0$
$C = a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 1$

El único bit que esta en A y C y no en B es a_5.

El código **H** para $(2^n - 1)$ bits con $(2^n - 1 - n)$ bits de información es posible tal que los bits ubicados en potencias de 2 dan paridad a los conjuntos indicados.

Donde:

$(2^n -1)$	n	$(2^n -1-n)$
7	3	4
15	4	11
31	5	26
	Bits de paridad	Bits de información

Ejercicio:

Codificar en Hamming el numero **1 0 1 1 0 1 1**, dado en cualquier código.

Se necesitan 4 bit de paridad:

	D C B A
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1

A={ a_1, a_3, a_5, a_7, a_9, a_{11} }
B={ a_2, a_3, a_6, a_7, a_{10}, a_{11} }
C={ a_4, a_5, a_6, a_7 }
D={ a_8, a_9, a_{10}, a_{11}}

Por lo tanto:
a_1=1
a_2=1
a_4=0
a_8=0

y **N** = $(11100110011)_{BCD-H}$