
UNIDAD 6

SISTEMAS DE NUMERACIÓN

INTRODUCCIÓN

En este capítulo nos ocuparemos de algunos Sistemas de numeración que nos serán de gran utilidad para iniciar el estudio del computador.

Antes de descubrir cada uno, de realizar operaciones y de encontrar equivalencias entre ellos, recordemos las características del Sistema Decimal, el sistema al cual estamos acostumbrados, dado que estas características son las que aparecerán en los restantes sistemas que aprenderemos.

Entendemos por Conjunto de Caracteres de un sistema de numeración a un conjunto finito y ordenado de elementos llamados dígitos. Estos dígitos, solos o combinados entre sí, forman los elementos de un sistema de numeración en particular. Así, el conjunto de caracteres del sistema decimal, que llamaremos C, es:

$$C = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

La base de un sistema de numeración nos indica la cantidad de dígitos con que cuenta el conjunto de caracteres. El elemento que indica la base del sistema en el cual estamos trabajando, se forma como combinación del segundo seguido del primer elemento de C. Señalaremos con β dicha base, teniendo entonces para el Sistema Decimal:

$$\beta = 10$$

Como sabemos, el sistema decimal es un sistema posicional, cada uno de sus elementos puede expresarse como suma ponderada de potencias de la base, donde los pesos de ponderación son siempre los dígitos de *C*. Por ello, para componer un número basta realizar las operaciones de potencia, producto y suma indicadas.

Como en adelante trabajaremos simultáneamente con más de un sistema a la vez, a efectos de identificar en que base estamos expresando un número, haremos la siguiente convención: encerraremos el número entre paréntesis y como subíndice indicaremos la base, expresando ésta en Sistema decimal, por ejemplo:

$$(314)_{10} = (3 \times 10^2 + 1 \times 10^1 + 4 \times 10^0)_{10}$$

$$(1094)_{10} = (1 \times 10^3 + 0 \times 10^2 + 9 \times 10^1 + 4 \times 10^0)_{10}$$

Tengamos en cuenta que, para efectuar operaciones en el sistema decimal, nos basta utilizar algo que tenemos memorizado: las tablas de las operaciones suma y producto.

Esto vale también para cualquier otro sistema y, por lo tanto, para definirlo bastará identificar su conjunto de caracteres y poseer las tablas de las operaciones.

EL SISTEMA BINARIO

El conjunto de caracteres es:

$C = \{0, 1\}$

La base por definición se representa: $\beta = 10$

Tengamos cuidado ahora con lo siguiente: la representación de β es idéntica en el sistema decimal y en el sistema binario, pero obviamente, no representan la misma magnitud. Utilizaremos la convención mencionada anteriormente y anotaremos para el Sistema Binario:

$$\beta = (10)_2$$

Podemos expresar un número del sistema binario como suma ponderada de potencias de la base $(10)_2$, por ejemplo:

$$(101)_2 = (1 \times 10^{10} + 0 \times 10^{1} + 1 \times 10^{0})_2$$

Esta expresión es matemáticamente correcta, sin embargo, dificulta la interpretación a simple vista, pues la base y los exponentes también están expresados en sistema binario. Por lo tanto, acordemos una nueva convención: cualquiera sea el sistema de numeración en que estemos trabajando, expresaremos la base y su exponente en sistema decimal. Entonces, podemos poner:

$$(101)_2 = (1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0})_2$$

Las tablas de las operaciones suma y producto para el sistema binario son las siguientes:

+	**0**	**1**
0	0	1
1	1	10

*	**0**	**1**
0	0	0
1	0	1

Ejemplos:
Sumas

```
  1 0 1 1 0          1 1 1 1          1 1 1
+     1 1 1      +     1 0 0      +     1 1 1
-------------    -------------    -------------
  1 1 1 0 1        1 0 0 1 1          1 1 1 0
```

Restas

```
  1 0 1 1 0          1 0 0 1 1
-     1 1 1      -       1 0 1
-----------      -------------
    1 1 1 1          1 1 1 0
```

Productos

```
  1 0 0 1 0          1 0 0 1
    x 1 1              x 1 0
-------------      -----------
  1 0 0 1 0          1 0 0 1
  1 0 0 1 0 -      + 1 0 0 1 -
-------------      -----------
  1 1 0 1 1 0        1 1 0 1 1
```

Conteo

Veamos cómo podemos realizar el conteo. Comenzaremos a reflexionar cómo contamos en el sistema decimal:

Tengamos presente el conjunto de caracteres para dicho sistema:

$$C = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Comencemos ahora a contar, los números naturales de un solo dígito, incluyendo al cero, en orden creciente:

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

Los naturales que continúan este orden poseen dos o más dígitos, el primero de ellos se compone por el segundo dígito del conjunto de caracteres seguido del primero de ellos:

C = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9} resultando **10** o lo que es lo mismo, sumando 9+1 = 10;

El siguiente número se compone por el segundo con el segundo:

C = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} resultando **11** o lo que es lo mismo, sumando 10+1 = 11;

El siguiente número se compone por el segundo con el tercero:

C = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} resultando **12** o lo que es lo mismo, sumando 11+1 = 12;

Los restantes números de la segunda decena se formarán siguiendo la misma secuencia, resultando los números **13, 14, 15, 16, 17, 18, 19**

La siguiente decena se formarán con el dígito 2 seguido de cada uno de los dígitos del conjunto de caracteres:

C = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} resultando **20, 21, 22, 23, 24, 25, 26, 27, 28, 29**

Este proceso es el efectuado para cada una de las decenas. La formación de las centenas continúa básicamente el mismo algoritmo:

99+1 = 100, o bien **C = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}** resultando **100**

100+1 = 101, o bien **C = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}** resultando **101**

Continuando con este proceso se construyen los números desde **100** hasta **109**. El **110** se forma como **109 + 1** o bien **fijando** el **1** como dígito más significativo y aplicando a los otros dos el algoritmo ya descrito para números de dos dígitos.

C = {0, 1 , 2, 3, 4, 5, 6, 7, 8, 9} resultando **110**

Esta misma ley de formación respeta cualquier secuencia de números naturales, cualquiera sea el sistema posicional del que se trate, por lo que, en el sistema binario, los naturales incluido el cero se formarán tal como en el sistema decimal. Para ello debemos tener presente el Conjunto de Caracteres:

C = {0, 1}

Los números de una sola cifra serán **0** y **1**

Los números de dos cifras serán **C = {0, 1} 10** y el **11** (Estos números se llaman “uno cero” y “uno uno”, ya que el nombre “diez” y “once” se reservan para los números en base decimal).

Continuando con la formación, los números naturales de tres cifras serán:
C = {0, 1} el **100**, el **101**

C = {0, 1 } y los naturales **110** y **111**

Los naturales que siguen serán de cuatro y más cifras: 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000...

Complemento Restringido y Complemento Auténtico

Veremos ahora dos definiciones:

Se define como *complemento restringido* o *complemento a uno* de un número binario negativo al número binario que se obtiene restando cada dígito de $(\beta - 1)$, es decir, se obtiene restando cada dígito de $(10 – 1 = 1)_2$

Por ejemplo, si el número es $(-100)_2$, tomamos cada uno de sus dígitos y lo restamos, en forma independiente, de $(10 – 1 = 1)_2$, comenzando desde el dígito menos significativo:
$(1 – 0 = 1)_2$ siendo lo mismo para
el siguiente:
$(1 – 0 = 1)_2$ y
para el último:
$(1 – 1 = 1)_2$

Por lo tanto, el complemento a uno de $(-100\ )_2$ es $(011)_2$ o como debe expresarse sin colocar el cero a la izquierda: $(11)_2$

Dos ejemplos más: el complemento a uno de $(-1101)_2$ es $(0010)_2 = (10)_2$ y el complemento a uno de $(-111)_2$ es $(000)_2 = (0)_2$

Se define como *complemento auténtico* o *complemento a dos* de un número binario negativo a aquel número binario que se obtiene sumando $(1)_2$ al complemento restringido del número dado.

Por ejemplo:

El complemento a uno de $(-11010)_2$ es $(101)_2$, por lo tanto, el complemento a dos de $(-11010)_2$ será: $(101 + 1 = 110)_2$

El complemento a uno de $(-111\ )_2$ es $(0)_2$, por lo tanto, el complemento a dos de $(-111\ )_2$ será: $(0 +1 = 1)_2$

En realidad, estos conceptos son los que se corresponden con los conocidos por nosotros como *complemento a nueve* y *complemento a diez* de un número en el sistema decimal. Sabemos también que podemos aplicarlos para transformar en suma la operación resta.

Veamos cómo proceder en base al siguiente ejemplo: digamos que deseamos hacer la resta 83 – 25, cuyo resultado, por supuesto conocemos y es 58. Digamos ahora que no deseamos restar “porque nos cuesta pedir prestado una unidad al dígito de la izquierda”,

entonces decidimos "complicar" nuestra expresión, para resolver el problema por un camino más largo, pero más sencillo:

$83 – 25 = 83 + (-25) = 83 + (100 – 25) – 100$

Pero lo mismo llegamos a la "resta complicada de tener que pedirle una unidad al dígito de la izquierda" para efectuar 100 - 25. Entonces, convertimos esa resta complicada en una expresión equivalente:

$83 + (100 – 25) – 100 = 83 + (99 – 25 + 1) – 100$

La resta 99 – 25 será siempre una resta sin dificultad porque nos hemos asegurado de restar cada dígito del sustraendo del dígito 9 del minuendo, que es el mayor dígito del sistema decimal y que –por lo tanto- "no necesita pedirle una unidad prestada a nadie". Continuando con las cuentas que resultaron:

$83 + (99 – 25 + 1) – 100 = 83 + (74 + 1) – 100 = 83 + 75 – 100 = 158 – 100$

Esta última resta tampoco tendrá dificultad porque siempre será una potencia de la base del sistema en el que estemos trabajando, o sea que será la unidad seguida de ceros...¡y los ceros son muy fáciles de restar! Finalmente:

$158 – 100 = 58$ que es el resultado esperado.

Reconozcamos los pasos que hemos seguido para desarrollar nuestro ejemplo y nuestros valores de interés: el complemento a nueve (o complemento restringido) y el complemento a diez (o complemento auténtico)

Paso 1: Convertimos la operación de resta, en suma, ya que sumamos 83 + (-25) en vez de restar 83 - 25. Ahora queda claro por qué definimos los complementos de números negativos. Para convertir una resta, en suma, el sustraendo positivo se convierte en un sumando negativo.

Paso 2: Obtenemos el Complemento Restringido del número negativo (-25) haciendo:

$99 - 25 = 74$

Paso 3: Obtenemos el Complemento Auténtico del número negativo (-25) haciendo:

$74 + 1 = 75.$

Observemos que también puede entenderse al número 75 como el complemento de 25, o sea, lo que le falta al 25 para ser una potencia de la base y 75 es lo que le falta a 25 para ser 100.

Paso 4: Sumamos 83 + 75

Paso 5: Restamos la potencia de la base en la que nos habíamos excedido 158 - 100 = 58

A continuación, expresamos la secuencia de pasos a seguir, válido para cualquier sistema de numeración:

1) Se completan con ceros a la izquierda tanto el minuendo como el sustraendo, hasta que ambos tengan la misma cantidad de dígitos que tiene el que posee mayor cantidad.

2) Se suma al minuendo el complemento auténtico del sustraendo

3) Si el último acarreo es cero, el signo del resultado de la resta es negativo y su valor absoluto es el complemento auténtico de la suma efectuada sin tener en cuenta el último acarreo.
   Si el último acarreo de la suma es uno, el signo del resultado de la resta es positivo y su valor absoluto es el obtenido en la suma efectuada sin tener en cuenta el último acarreo.

Ejemplos:

1) Efectuar: $(1111 - 100)_2$
   Completando con ceros a izquierda, efectuamos $(1111 - 0100)_2$
   El complemento a dos del sustraendo es $(1100)_2$, por lo tanto, la suma a realizar es: $(1111 + 1100)_2 = (1011)_2$ Siendo
   1 el último acarreo.
   Como el último acarreo es 1, el signo del resultado de la resta es positivo y su valor absoluto es $(1011)_2$ por lo tanto:
   $(1111 - 100)_2 = (1011)_2$

2) Efectuar: (101 – 1001)2
   Completando con ceros a la izquierda, efectuamos $(0101 - 1001)_2$.
   El complemento a dos del sustraendo es $(111)_2$, por lo tanto, la suma a realizar es: $(0101 + 111)_2 = (01100)_2$ siendo 0 el último acarreo.
   Como el último acarreo es 0, el signo del resultado de la resta es negativo y su valor absoluto es $(1100)_2$, cuyo complemento a dos es $(100)_2$ por lo tanto: $(101 - 1001)_2 = (-100)_2$.

EL SISTEMA OCTAL

El conjunto de caracteres es:

$C = \{0, 1, 2, 3, 4, 5, 6, 7\}$

Su base formalmente debe representarse $\beta = 10$ y la podemos poner, de acuerdo a las convenciones:

$\beta = (10)_8 \equiv (8)_{10;}$ esta expresión indica que $(10)_8$ es equivalente a $(8)_{10}$

Las tablas de las operaciones suma y producto son las siguientes:

+	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Ejemplos:

Producto

```
    3 4 5
    x 3 3
  -------
    1 2 5 7
+ 1 6 2 4
  -------
  1 7 5 1 7
```

Cociente

```
  1 7 5 1 | 4 3
- 1 5 1   | 3 4
  -------
    2 4 1
  - 2 1 4
  -------
      2 5
```

EL SISTEMA HEXADECIMAL

El conjunto de Caracteres es:

C = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Su base es: $\beta = (10)_{16} \equiv (16)_{10}$

Las tablas de las operaciones suma y resta son las siguientes:

+	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A

x	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	9	12	1B	24	2D	26	3F	48	51	5A	63	6C	75	7E	87
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96

Ejemplos:

Resta

```
  A 8 9
- E F
-------
  9 9 A
```

Producto

```
    A 9 F
    x 3 4
-----------
    2 A 6 C
+ 1 F D 1
-----------
  2 2 7 7 C
```

CONVERSIÓN DE SISTEMAS DE NUMERACIÓN

Dado un número expresado en un determinado sistema, siempre puede encontrarse el valor equivalente al mismo en cualquiera de los otros. A modo de ejemplo, mostramos la siguiente tabla elaborada para los sistemas de mayor interés:

β = **10**	β = **2**	β = **8**	β = **16**
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Esta tabla nos muestra que son equivalentes, por ejemplo: $(1010)_2$ y $(A)_{16}$, $(13)_{10}$ y $(15)_8$, $(10000)_2$ y $(20)_8$, $(7)_{10}$ y $(7)_{16}$.

Para construir esta pequeña tabla de equivalencias nos bastó escribir, en forma secuencial, los números en cada uno de los sistemas. Está claro que esto sería un método para encontrar cualquier equivalencia, pero bastante tedioso y muy propenso a errores si quisiésemos, por ejemplo, el equivalente en base 8 de $(A28CFA900FA)_{16}$ Veamos entonces algunos métodos que nos permitirán encontrar equivalencias.

Conversión de Números Enteros

Método de la expresión de un número como suma ponderada de potencias de su base

Vimos que para todo sistema de numeración de base β, se define un conjunto de caracteres al que llamamos *C*, que contiene los dígitos que, individualmente o en combinaciones entre ellos, representan todo número del sistema.

Sea entonces *C* el conjunto de caracteres de un sistema de numeración de base $\beta > 1$, y sean $\alpha_i \in C$ los dígitos de dicho sistema. Sean además α_j los dígitos elementos del conjunto *C*', conjunto de caracteres de la base de llegada β'.

Podemos expresar cualquier número de la base β, que simbolizamos como N_B , como suma ponderada de potencias de β, donde los pesos de ponderación son los elementos de *C*:

$$N_B = (\alpha_n \alpha_{n-1} \dots \alpha_1 \alpha_0)_B = (\alpha_n \beta_n + \alpha_{n-1}\beta_{n-1} + \dots + \alpha_1\beta_1 + \alpha_0\beta_0)_B$$

Cada dígito $\alpha_i \in C$ de la base de partida β tiene su equivalente γ_i en la base de llegada β':

$$(\alpha_0)_B \equiv (\gamma_0)_{B'} \quad ; \quad (\alpha_1)_B \equiv (\gamma_1)_{B'} \quad ; \dots\dots\dots ; \quad (\alpha_n)_B \equiv (\gamma_0)_{B'}$$

y además, β tiene también su equivalente en el sistema de base β'. Teniendo en cuenta que β se expresa en base 10, de acuerdo con la convención adoptada, simbolizamos:

$$(\beta)_{10} \equiv (\beta^*)_{B'}$$

entonces:

$$N_B = (\alpha_n \beta_n + \alpha_{n-1}\beta_{n-1} + \alpha_{n-2}\beta_{n-2} + \dots + \alpha_1\beta_1 + \alpha_0\beta_0)_B \equiv$$
$$\equiv (\gamma_n(\beta^*)_n + \gamma_{n-1}(\beta^*)_{n-1} + \gamma_{n-2}(\beta^*)_{n-2} + \dots + \gamma_1(\beta^*)_1 + \gamma_0(\beta^*)_0)_{B'}$$

y de esta manera, efectuando las operaciones correspondientes de potencia, producto y suma que quedaron expresadas en la aritmética de la base de llegada β', se compone el número $N_{B'}$, equivalente de N_B.

Ejemplos:

1) $(1A2)_{16}$ a base 8.
Descomponemos el número como suma de potencias de la base 16:
$(1A2)_{16} = (1 * 16^2 + A * 16^1 + 2 * 16^0)_{16}$

En nuestro caso: $(\alpha_0 = 2)_{16}$, $(\alpha_1 = A)_{16}$, $(\alpha_2 = 1)_{16}$. Buscamos ahora sus correspondientes equivalentes en la base 8:

$(1)_{16} \equiv (1)_8$, $(A)_{16} \equiv (12)_8$, $(2)_{16} \equiv (2)_8$.

Esto es: $(\gamma_0 = 2)_8$, $(\gamma_1 = 12)_8$, $(\gamma_2 = 1)_8$.

La base de partida es: $(\beta)_{10} = (16)_{10}$, por lo tanto, su equivalente en la base de llegada será:

$(\beta^*)_8 = (20)_8$, pues $(16)_{10} \equiv (20)_8$.
Reemplazando por los equivalentes y haciendo las operaciones con aritmética del sistema octal:
$(1A2)_{16} = (1 * 20^2 + 12 * 20^1 + 2 * 16^0)_{16} \equiv (1 * 400 + 12 * 20 + 2)_8 = (400 + 240 + 2) = (642)_8$
O sea: $(1A2)_{16} = (642)_8$

2) $(10011101)_2$ a base 16
$(10011101)_2 = (1 * 2^7 + 1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1)_2$.

Como: $(1)_2 \equiv (1)_{16}$ y $(2)_{10} \equiv (2)_{16}$
Utilizando aritmética del sistema hexadecimal para efectuar las operaciones, obtenemos:
$(10011101)_2 \equiv (1 * 2^7 + 1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1)_{16} = (80 + 10 + 8 + 4 + 1)_{16} = (9D)_{16}$

3) $(703)_8$ a base 10.

$(703)_8 = (7 * 8^2 + 4)_8 \equiv (7 \times 8^2 + 4)_{10} = (448 + 4)_{10} = (452)_{10}$.

Como hemos señalado al principio y mostrado con algunos ejemplos, el método es general: podemos convertir de cualquier base a de partida a cualquier base de llegada. Sin embargo, es muy conveniente notar que, este método en particular, resulta de mayor

practicidad cuando la base de llegada es $\beta' = 10$, pues utilizamos una aritmética que nos es conocida.

Método de Horner

El método de Horner se apoya en el principio de la regla de Ruffini, por todos conocida, que permite evaluar un polinomio de una variable para un valor dado de la misma.

Efectivamente, la expresión como suma ponderada de potencias de la base β de un número N_B, es un polinomio completo y ordenado en la variable β, el cual podemos factorear:

$$N_B = (\alpha_n \alpha_{n-1} \alpha_{n-2} \ldots \alpha_1 \alpha_0)_B = (\alpha_n \beta_n + \alpha_{n-1}\beta_{n-1} + \alpha_{n-2}\beta_{n-2} + \ldots + \alpha_1\beta_1 + \alpha_0)_B = (((\alpha_n \beta^n + \alpha_{n-1})\beta + \alpha_{n-2})\beta + \ldots + \alpha_1)\beta + \alpha_0$$

Como vimos en el método anterior, los $(\alpha_i)_B$ tienen sus equivalentes $(\gamma_j)_{B'}$ y, asimismo, $(\beta)_{10}$ tiene su equivalente $(\beta^*)_{B'}$. Si reemplazamos por los equivalentes, obtendremos un polinomio completo y ordenado en la variable β^*, también factoreado, al cual podemos evaluar en $(\beta^*)_{B'}$, obteniendo así el número $N_{B'}$ equivalente a N_B, obviamente efectuando operaciones con aritmética del sistema de la base de llegada.

	γ_n	γ_{n-1}	. . . γ_1	γ_0
β^*		$\gamma_n\beta^*$	. . . . .	. . . . .
	γ_n	$\gamma_n\beta^*$	$((\gamma_n \beta^* + \gamma_{n-1})\beta^* + \ldots)\beta^* + \gamma_1$	$N_{B'}$

Ejemplos:

1) $(10111)_2$ a base 10

Como $(1)_2 \equiv (1)_{10}$ y $(2_{10} \equiv (2)_{10}$, utilizando aritmética del sistema decimal:

	1	0	1	1	1
2		2	4	10	22
	1	2	5	11	23

Resulta: $(10111)_2 \equiv (23)_{10}$.

2) $(25)_8$ a base 2

Como $(2)_8 \equiv (10)_2$, $(5)_8 \equiv (101)_2$, $(8)_{10} \equiv (1000)_2$, utilizando aritmética del sistema binario:

	10	101
1000		10000
	10	10101

Resulta: $(25)_8 \equiv (10101)_2$.

3) $(1A2)_{16}$ a base 8

Como $(1)_{16} \equiv (1)_8$, $(A)_{16} \equiv (12)_8$, $(2)_{16} \equiv (2)_8$, $(16)_{10} \equiv (20)_8$, utilizando aritmética del sistema octal:

	1	12	2
20		20	640
	1	32	642

Resulta: $(1A2)_{16} = (642)_8$

Al igual que en el método anterior, aún cuando la aplicación es general para conversiones entre cualquier par de sistemas de numeración, su mayor practicidad ocurre cuando se trata de convertir un número expresado en cualquier sistema al sistema decimal.

Método de la División Reiterada

Antes de considerar este método, supongamos tener un número entero expresado en sistema decimal y desear separar los dígitos que lo componen. Tomemos, por ejemplo, el 197. Surge inmediatamente la posibilidad de dividirlo por 10, que es la base del sistema mientras se obtenga cociente entero, entonces, como el cociente entre 197 y 10 es 19 y el resto es 7, podemos poner:

197 = 19 * 10 + 7

donde justamente ese primer resto es el dígito menos significativo de 197. Si ahora dividimos por 10 al cociente anterior, como el resto es 9, tenemos:

$19 = 1 * 10 + 9$

siendo el 9 el dígito siguiente en significación.

Ahora no podemos seguir dividiendo por el cociente anterior, que es 1, ya que es menor que el divisor 10 pero, justamente, este último cociente, el 1, es el dígito más significativo de 197.

Podemos pensar en hacer lo mismo en cualquiera de las bases. Dado un número en la base β, N_B, lo dividimos por la base de llegada β', obviamente expresada por su equivalente en base β, efectuando el cociente utilizando aritmética de esta base:

Tenemos entonces que $(\beta')_{10} \equiv (\beta^*)_B$.

Efectuando el primer cociente podemos poner:

$$N_B = (C\beta^* + R)_B$$

Como $(R<\beta^*)_B$, entonces, el equivalente de $(R)_B$ en base β' deberá ser menor que β', y por la tanto, es un dígito de la base β' que denominaremos $(\alpha'_0)_{B'}$ por ser el dígito menos significativo del número $N_{B'}$.

Como hemos visto, el proceso de división se continúa hasta que el cociente sea menor que el divisor β^*. Pero este último cociente también tiene su equivalente en la base de llegada β', que es justamente el dígito más significativo de $N_{B'}$.

Ejemplos:

1) $(2816)_{10}$ a base 16
   Como la base de llegada ya está, por la convención, expresada en la base de partida, efectuamos: $(2816/16)_{10}$, cuyo cociente es $(176)_{10}$ y como resto tiene $(0)_{10}$.
   Efectuamos la siguiente división $(176/16)_{10}$, su cociente es $(11)_{10}$ y su resto es $(0)_{10}$.
   No podemos seguir dividiendo pues $(11<16)_{10}$; encontramos entonces las equivalencias:

   $(0)_{10} \equiv (0)_{16}$ y $(11)_{10} \equiv (B)_{16}$.

   Por lo tanto $(2816)_{10} \equiv (B00)_{16}$.

2) $(11101)_2$ a base 8.

$(\beta')_{10} = (8)_{10} \equiv (1000)_2 = (\beta^*)_B$

Efectuamos $(11101/1000)_2$, cuyo cociente es $(11)_2$ y como resto tiene $(101)_2$
Ahora debería dividirse $(11 / 1000)_2$ que no se puede efectuar pues el dividendo es menor que el divisor. Vemos entonces las equivalencias:

$(11)_2 \equiv (3)_8$, de modo que $(3)_8$ es el dígito más significativo de N_8 y $(101)_{10} \equiv (5)_8$, por lo tanto $(5)_8$ es el dígito menos significativo de N_8.

Por lo tanto $(11101)_2 \equiv (35)_8$

Destacando como siempre la generalidad del método, se observa que resulta de mayor practicidad para convertir números expresados en sistema decimal a cualquiera de los otros sistemas de numeración puesto que, además de utilizar aritmética decimal, la base de llegada, por la convención, ya está expresada en caracteres de la base de partida.

Método Directo

Hasta ahora hemos visto métodos de conversión de números enteros que nos permiten partir de cualquier base β para encontrar la correspondiente equivalencia en base β'.

Veremos a continuación un método que es restringido, sólo para convertir números enteros expresados en sistemas de numeración que posean la condición de que la base de uno de ellos es potencia de la base del otro. Así, será de utilidad para convertir del sistema binario al sistema octal o hexadecimal y viceversa, pues 8 y 16 son potencias de 2. También veremos que, reiterando el método, puede aplicarse en dos etapas para convertir números expresados en sistema octal a hexadecimal y viceversa.
Seguidamente haremos la justificación del método directo para convertir un número en sistema octal al sistema binario, siendo análogas las demás.

Sea N_8 el número entero dado en sistema Octal, que puede expresarse como suma ponderada de potencias de la base ocho:

$$N_8 = (\alpha_n \alpha_{n-1} \ldots \alpha_1 \alpha_0)_B = (\alpha_n 8_n + \alpha_{n-1} 8_{n-1} + \ldots + \alpha_1 8_1 + \alpha_0)_8$$

Por otra parte, para representar el mayor dígito en base 8, son necesarios tres dígitos binarios, pues $(7)_8 \equiv (111)_2$. Por esto diremos que a cada dígito α_i de la base 8 le corresponde, por su equivalencia, tres dígitos α'_j de la base dos, combinación que, a su vez, puede expresarse como suma ponderada de potencias de la base dos, lo que representamos del siguiente modo:

$$(\alpha_0)_8 \equiv (\alpha'_2 \alpha'_1 \alpha'_0) = (\alpha'_2 2^2 + \alpha'_1 2 \ldots \alpha'_0)_2$$

$(\alpha_1)_8 \equiv (\alpha'_5\alpha'_4\alpha'_3) = (\alpha'_5 2^2 + \alpha'_4 2 \ldots \alpha'_3)_2$

.....

$(\alpha_n)_8 \equiv (\alpha'_{3n+2}\alpha'_{3n+1}\alpha'_{3n}) = (\alpha'_{3n+2} 2^2 + \alpha'_{3n+1} 2 \ldots \alpha'_{3n})_2$

Además, $(8)_{10} = (2^3)_{10}$, entonces :

$N_8 = (\alpha_n \alpha_{n-1} \ldots \alpha_1 \alpha_0)_B = (\alpha_n 8_n + \alpha_{n-1} 8_{n-1} + \ldots + \alpha_1 8_1 + \alpha_0)_8 \equiv$

$((\alpha'_{3n+2} 2^2 + \alpha'_{3n+1} 2 + \alpha'_{3n}) 2^{3n} + \ldots + (\alpha'_5 2^2 + \alpha'_4 2 \ldots \alpha'_3) 2^3 + (\alpha'_2$

$2^2 + \alpha'_1 2 \ldots \alpha'_0))_2 =$

$((\alpha'_{3n+2} 2_{3n+2} + \alpha'_{3n+1} 2_{3n+1} + \alpha'_{3n} 2_{3n} + \ldots + \alpha'_5 2_5 + \alpha'_4 2_4 \ldots \alpha'_3 2_3 + \alpha'_2 2_2 + \alpha'_1 2 .$

$. . \alpha'_0))_2 = (\alpha'_{3n+2} \alpha'_{3n+1} \alpha'_{3n} \ldots \alpha'_5 \alpha'_4 \alpha'_3 \alpha'_2 \alpha'_1 \alpha'_0)_2 = N_2$

Ejemplos:

1) $(7204)_8$ a base 2
Buscamos las equivalencias de cada dígito octal tomando, para cada uno, tres dígitos del sistema de numeración de base dos:

$(7)_8 \equiv (111)_2$, $(2)_8 \equiv (010)_2$, $(0)_8 \equiv (000)_2$, $(4)_8 \equiv (100)_2$
entonces:

$(7204)_8 \equiv (111010000100)_2$

2) $(1100101)_2$ a base 8
Separamos de a tres dígitos comenzando desde la derecha, para agregar ceros a la izquierda si fuera necesario y buscamos las correspondientes equivalencias:

$(101)_2 \equiv (5)_8$, $(100)_2 \equiv (4)_8$, $(001)_2 \equiv (1)_8$ entonces:

$(1100101)_2 \equiv (145)_8$.

3) $(715)_8$ a base 16
Como primer paso convertiremos el número dado a base dos y luego, en una segunda etapa, de base dos a base dieciséis, teniendo en cuenta que para representar el mayor dígito del sistema hexadecimal son necesarios cuatro dígitos del sistema binario.

$(7)_8 \equiv (111)_2$, $(1)_8 \equiv (001)_2$, $(5)_8 \equiv (101)_2$ tenemos:

$(715)_8 \equiv (111001101)_2$ ahora:

$(1101)_2 \equiv (D)_{16}$, $(1100)_2 \equiv (C)_{16}$, $(0001)_2 \equiv (1)_{16}$ resultando,
en definitiva:

$(715)_8 \equiv (1CD)_{16}$.

Ya conocemos, entonces, métodos que nos permiten encontrar equivalencias entre números enteros. Analizaremos ahora un método que nos permite convertir de un sistema a otras fracciones propias. Con esto último, combinando ambos procedimientos, estaremos en condiciones de convertir cualquier número real, trabajando por un lado su parte entera y por otro su parte fraccionaria.

Conversión de Fracciones Propias

Los métodos de conversión de números enteros por descomposición en suma ponderada y también el método directo son de aplicación en el caso de las fracciones propias, siguiendo la misma justificación que la vista oportunamente.

Ejemplos:

1) $(0,101)_2$ a base 10

$$(0,101)_2 = (1 * 2^{-1} + 1 * 2^{-3})_2 = \frac{1+1}{2\ 8}_{10} = (0,5 + 0,125)_{10} = (0,625)_{10}$$

2) $(0,24)_8$ a base 2

Como $(2)_8 \equiv (010)_2$ y $(4)_8 \equiv (100)_2$ tenemos: $(0,24)_8 \equiv (0,010100)_2 = (0,0101)_2$

Tal como en el caso de la conversión de números enteros, el método de descomposición en suma ponderada resulta conveniente cuando la base de llegada es la base 10 y el método directo tiene las mismas restricciones ya vistas. Por lo tanto, analizaremos ahora un método que es general y que será de mayor utilidad cuando deseemos convertir a otro sistema, números fraccionarios expresados en sistema decimal.

<u>Método de la Multiplicación Reiterada</u>

Antes de formalizar el procedimiento, pensemos en la tarea de separar los dígitos de un dado número fraccionarios expresado en base 10. Tomando como ejemplo $(0,714)_{10}$ si multiplicamos el número por la base, tenemos:

$(0,714 * 10 = 7,14)_{10}$

y justamente, la parte entera del resultado del producto es el dígito más significativo del número dado. Si ahora hacemos la misma operación sobre la parte fraccionaria del resultado del producto, tenemos:

$(0{,}14 * 10 = 1{,}4)_{10}$

nuevamente, la parte entera del resultado del producto es el dígito que sigue en menor significación, debiendo efectuar, por último:

$(0{,}4 * 10 = 4)_{10}$

para obtener el dígito menos significativo.

Generalizando, sea F_B un número fraccionario puro expresado en la base de partida β, del cual deseamos su equivalencia $F_{B'}$ en la base de llegada β'. Para hacer los productos utilizando la aritmética de la base β previamente debemos contar con la equivalencia:

$(\beta')_{10} \equiv (\beta^*)_B$

Ahora podemos efectuar el producto cuyo resultado será un número en base β, al que llamaremos N:

$(F \times \beta^* = N)_B$

La parte entera de N_B, tiene su equivalente en la β', que llamaremos α'_i, y este será justamente el dígito más significativo de la parte fraccionaria del número en base β':

$[N_B] = (\gamma_1)_B \equiv (\alpha'_1)_{B'}$

Ahora transformaremos N_B en un número fraccionario puro, quitándole su parte entera:

$(F = N - [N])_B$

y repetimos el procedimiento hasta que ocurra alguna de las siguientes tres alternativas:

a. $(F = [F])_B$ o
b. Se obtiene la cantidad de dígitos decimales deseada, o
c. Se encuentra un período, en el caso de fracciones periódicas.

Ejemplos:

---

1) $(0,11)_2$ a base 10

Como $(10)_{10} \equiv (1010)_2$, tenemos:
$(F\ x\beta^{*})_B = (0,11 * 1010)_2 = N_2 = N_B$

Ahora como $[N_B] = (111)_2$ y como $(111)_2 \equiv (7_{10} = (\alpha'_i)$, $(7)_{10}$ será el dígito más significativo de la parte fraccionaria del número buscado en base 10.

Ahora calculamos:
$(F = N - [\ N\ ])_B$, o sea $(F = 111,10 - [\ 111,10\ ])_2$, de donde $(F = 111,10 - 111 = 0,10)_2$ y repetimos el procedimiento:
$(0,10 * 1010 = 101)_2$ como $([101] = 101)_2$ y $(101)_2 = (5)_{10}$ tenemos el dígito siguiente, deteniendo el proceso por haber llegado a $(F = [F])_B$ Por lo tanto: $(0,11)_2 \equiv (0,75)_{10}$

2) $(0,1272)_{10}$ a base 2

Como la base de llegada ya está, por la convención, expresada en caracteres de la base de partida podemos efectuar:

$(0,1272 * 2 = 0,2544; [0,2544] = 0)_{10}$ teniendo presente que $(0)_{10} \equiv (0)_2$ Ahora:

$(0,2544 - [0,2544] = 0,2544;\ 0,2544 * 2 = 0,5088\ ;\ [0,5088] = 0)_{10}$ y $(0)_{10} \equiv (0)_2$

$(0,5088 - [0,5088] = 0,5088; 0,5088 * 2 = 1,01176\ ;\ [1,01176] = 1)_{10}$ y $(1)_{10} \equiv (1)_2$

Si interrumpimos el cálculo contando con estos tres decimales, resulta:

$(0,1272)_{10} \equiv (0,001\ \ldots)_2$

CODIFICACIÓN

Dato: es una representación formalizada de entidades o hechos, adecuada para la comunicación, interpretación y procesamiento por medios humanos o automáticos.

Información: es el significado que una persona asigna a un dato.

DATO ≠ INFORMACIÓN

Un dato es un material de escaso valor o nulo para un individuo en una situación concreta. No reduce la dosis de ignorancia o el grado de incertidumbre de quién tiene que tomar una decisión.

La información es un dato o conjunto de datos evaluados en un momento determinado, sobre un problema específico, para alcanzar un objetivo dado.

Dato elemental o BIT: es uno de los dos posibles estados estables 0 o 1.

Cualquier combinación de bits representa nuevos datos, en particular:

Byte: combinación de 8 bits.

Word: combinación de 2 bytes.

Códigos Binarios. Decimales codificados a binario (BCD)

El sistema binario es el más natural para una computadora, pero las personas están acostumbradas al sistema decimal. Una manera de arreglar este conflicto es convertir en números binarios todos los números decimales que se proporcionen como entrada, permitir que la computadora efectúe todas las operaciones aritméticas en binario y después convierta los resultados binarios a decimales para que el usuario los comprenda.

Así, el número $2469)_{10}$ es, en sistema binario, equivalente al número $100110100101)_2$

Sin duda que la conversión es larga y tediosa, razón por la cual se han elaborado los **sistemas decimales codificados a binario** (abreviado BCD, del inglés *Binary Coded Decimals*). Cuando los números decimales se usan para cálculos aritméticos internos se representan en código binario con cuatro bits por dígito.

Es muy importante comprender la diferencia entre la *conversión de números decimales a binarios* y la *codificación binaria de números decimales*. Por ejemplo, cuando se *convierte* a número binario el número decimal 99 se representa con la serie de bits 1100011; pero cuando *se representa en uno de los códigos BCD*, se vuelve 1001 1001.

Un código binario es un grupo de n bits que adopta 2^n combinaciones distintas de números 1 y 0 ; donde cada combinación representa un elemento del conjunto que se codifica. Por ejemplo, un conjunto de cuatro elementos puede representarse mediante un código de 2 bits, de manera que a cada elemento del conjunto se le asigna una de las siguientes combinaciones de bits: 00, 01, 10 y 11. Un conjunto de ocho elementos requiere un código de 3 bits, un conjunto de dieciséis elementos requiere un código de 4 bits y así sucesivamente.

Un código binario tendrá algunas combinaciones de bits no asignadas si la cantidad de elementos del conjunto no es una potencia de 2. Justamente el conjunto de los diez dígitos decimales constituye uno de esos casos. Un código binario que distingue entre 10 elementos debe contener al menos cuatro bits, pero seis de sus combinaciones no se asignarán. Pueden obtenerse muchos códigos diferentes al arreglar cuatro bits en 10 combinaciones distintas. En adelante se considerarán algunos de esos códigos, atendiendo las ventajas prácticas que suponen en las aplicaciones corrientes.
Códigos Ponderados: Responden a una regla de formación que adjudica un cierto peso a los "unos" binarios según la posición que ocupan en el conjunto de los cuatro bits, debiéndose verificar que la suma algebraica de los pesos de cada combinación sea igual al número decimal representado.

Un ejemplo de este tipo de códigos es el **Código 8-4-2-1**, conocido también como BCD "natural" o "puro", porque sus pesos coinciden con los cuatro primeros pesos del sistema binario natural.

Código 8-4-2-1

	8	4	2	1
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

Códigos Libres: En los códigos no ponderados la correspondencia decimal-binaria es arbitraria, en el sentido de que, al no existir pesos, no se verifica ningún tipo de sumatoria, aunque por lo general las combinaciones se forman según ciertas reglas.

Un ejemplo de este tipo de códigos es el **Código Exceso-3**, en el cual, cada dígito D de un número decimal se representa como el dígito (D + 3) del código 8-4-2-1. Una característica importante del código Exceso-3 es la de ser **auto complementado**, es decir que se observa en él una simetría de complementación respecto de la línea punteada. Esta complementación que se obtiene por la simple inversión de unos y ceros lo constituye en un código de utilidad para construir circuitos sumadores-restadores decimales sencillos.

Código Exceso-3

0	0	0	1	1	(3)
1	0	1	0	0	(4)
2	0	1	0	1	(5)
3	0	1	1	0	(6)
4	0	1	1	1	(7)
	---	---	---	---	
5	1	0	0	0	(8)
6	1	0	0	1	(9)
7	1	0	1	0	(10)
8	1	0	1	1	(11)
9	1	1	0	0	(12)

Códigos Progresivos: Se caracterizan por el cambio de un sólo bit en el pasaje de una combinación binaria a la siguiente. Si se lo compara con el código 8-4-2-1 podemos apreciar que en este último, para pasar de 0111 a 1000, por ejemplo, deben cambiarse los cuatro bits. Teniendo en cuenta que durante el funcionamiento de una computadora, por causas atribuibles al usuario o a la misma máquina (ruidos, intermitencias, amplificador defectuoso, etc.), pueden producirse errores, cambiándose un 1 por un 0 y viceversa, es que deben existir diversos procedimientos para reducir, detectar y ocasionalmente-corregir tales errores.

Uno de estos métodos es el empleo de un **Código Binario de Forma Reflejada**, caracterizado por el hecho de que dos números adyacentes difieren entre sí en el valor de *uno sólo de sus dígitos.* Tal disposición reduce los errores que pueden producirse cuando cambian varios dígitos.

Un tipo de código binario reflejado es el **Código de Gray** (Frank Gray), que si bien no se utiliza mayormente en las operaciones aritméticas, tiene su campo de aplicación en los dispositivos de entrada-salida, convertidores analógico-digitales y otros campos.

La construcción de este código se hace "reflejando" a partir de una primera línea de simetría 2 bits y agregando otros no reflejados. A su vez, el conjunto así formado se vuelve a reflejar utilizando otra línea de simetría y así sucesivamente.

De esta forma se obtienen códigos reflejados de 4, 8, 16, ... combinaciones, según se muestra en la figura, donde cada rectángulo recuadrado puede generar otro mayor siguiendo la regla expuesta.

Código de Gray

0 0 0 0 0
1 0 0 0 1
---------- 1a. línea de simetría
2 0 0 1 1
3 0 0 1 0
---------- 2a. línea de simetría
4 0 1 1 0
5 0 1 1 1
6 0 1 0 1
7 0 1 0 0
---------- 3a. línea de simetría
8 1 1 0 0
9 1 1 0 1
10 1 1 1 1
11. 1 1 1 0
12. 1 0 1 0
13. 1 0 1 1

14 1 0 0 1
15. 1 0 0 0

Detección de error

Como se dijo antes, en algunos códigos BCD, tal como el 8-4-2-1, sólo se utilizan 10 de las 16 combinaciones de cuatro dígitos que es posible realizar. Las combinaciones prohibidas o redundancias son, para este código: 1010, 1011, 1100, 1101 y 1111. Si se quiere enviar por un cierto canal de transmisión la información 0**1**11 de dicho código y por algún ruido en la transmisión se recibe con un bit de error la combinación 0**0**11 del mismo código, el sistema receptor la admitirá como tal sin "darse cuenta" del error producido. Si aparece en cambio, una de las combinaciones redundantes en la memoria puede asegurarse que se ha cometido un error. Se deduce entonces que un mayor número de bits redundantes significan más combinaciones "ajenas" al código ya que no le pertenecen, o sea, mayor posibilidad de detectar errores y/o corregirlos.

El bit de paridad

El código de detección de error que se utiliza con mayor frecuencia es el **bit de paridad**. Un bit de paridad es un bit extra que se incluye con un mensaje binario para hacer *par* o *impar* la cantidad total de unos del mensaje. En cualquier aplicación particular, se adoptará uno u otro tipo de paridad. El esquema de paridad par tiene por desventaja la de contar con una combinación de bits en que todos son ceros, mientras que en la paridad impar siempre hay un bit de 1 (de los cuatro que conforman el mensaje y el bit de paridad P).

Durante la transferencia de información, el bit de paridad se maneja como sigue: en el extremo desde donde se envía el mensaje se aplica un *generador de paridad*, donde se genera el bit de paridad requerido. El mensaje, incluyendo el bit de paridad P, se transmite a su destino. En el extremo donde se reciben, todos los bits que llegan se aplican a un *comprobador de paridad*, que verifica la paridad correcta adoptada (impar o par). Se detecta un error si la paridad comprobada no se apega a la paridad adoptada. El método de paridad detecta uno, tres, o cualquier cantidad impar de errores. No se detecta un número par de errores.
Como ejemplo, veamos una tabla en la que se muestra un mensaje de tres bits (xyz) a los que se le agregará un bit de paridad P impar o par.

Mensaje xyz	(impar)	P (par)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1