SISTEMAS NUMERICOS

Un sistema numérico es un conjunto de números que se relacionan para expresar la relación existente entre la cantidad y la unidad. Debido a que un número es un símbolo, podemos encontrar diferentes representaciones para expresar una cantidad.

CLASIFICACION

Los sistemas de numeración se clasifican en: posicionales y no posicionales.

SISTEMAS POSICIONALES

En ellos, cada cifra de un valor numérico contribuye al valor final dependiendo de su valor y de la posición que ocupa dentro de él (valor relativo). En estos sistemas tenemos tantos símbolos como la base del sistema. Los números mayores que la base se representan por medio de varias cifras.
El valor final será la suma de una serie de potencias de la base del sistema (B):

$$N = A_n \cdot B^n + A_{n-1} \cdot B^{n-1} + ... + A_1 \cdot B^1 + A_0 \cdot B^0$$

Donde A_i son las distintas cifras del valor numérico e 'i' su posición.

SISTEMAS NO POSICIONALES

Al contrario que en el caso anterior, en este caso la contribución de cada cifra no depende del lugar que ocupa. Un ejemplo de este sistema serían los números romanos:

La combinación XXI equivale a 21. Podemos ver cómo la cifra X aparece dos veces y siempre tiene el mismo valor: 10 unidades, independientemente de su posición.

El inconveniente que tienen estos sistemas es que para escribir valores numéricos grandes son necesarios muchos símbolos, y además resulta difícil efectuar operaciones aritméticas con ellos, cosa que no sucede con los posicionales.

Los sistemas de numeración que veremos a continuación son todos sistemas posicionales.

A partir de ahora, para evitar confusiones, cuando expresemos un valor numérico pondremos un subíndice al final indicando la base en la que se expresa dicho valor, salvo que por el contexto quede suficientemente claro:

225_{10} = Base decimal
11011_2 = Base binaria.

SISTEMA DECIMAL

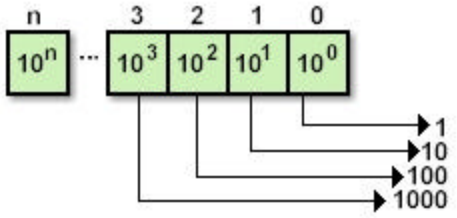
En el sistema de numeración decimal se utilizan diez símbolos, del 0 al 9 para representar una determinada cantidad. Los diez símbolos no se limitan a expresar solamente diez cantidades diferentes, ya que se utilizan varios dígitos en las posiciones adecuadas dentro de un número para indicar la magnitud de la cantidad.

Base: 10
Símbolos: 0,1,2,3,4,5,6,7,8,9

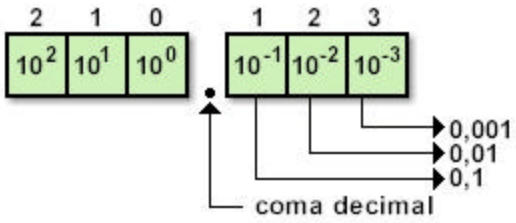
Cumple la fórmula anterior (B=10):

$$N = A_n \cdot B^n + A_{n-1} \cdot B^{n-1} + \ldots + A_1 \cdot B^1 + A_0 \cdot B^0$$

La posición de cada dígito en un número decimal indica la magnitud de la cantidad representada y se le puede asignar un peso. Los pesos para los números enteros son potencias de 10, que aumentan de derecha a izquierda, comenzando por $10^0 = 1$



Para números fraccionarios, los pesos son potencias negativas de diez que aumentan de izquierda a derecha comenzando por 10^{-1} .



Ejemplo:

$225_{10} = 2 \cdot 10^2 + 2 \cdot 10^1 + 5 \cdot 10^0 = 200 + 20 + 5$

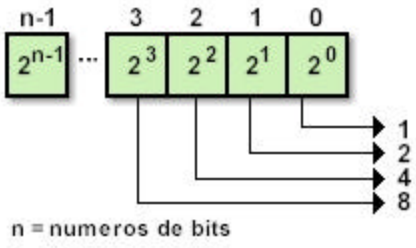
SISTEMA BINARIO

El sistema de numeración binario es simplemente otra forma de representar magnitudes. El sistema binario es menos complicado que el sistema decimal ya que sólo tiene dos dígitos. Al principio puede parecer más complicado por no ser familiar. El sistema decimal con sus diez dígitos es un sistema en base 10, el sistema binario con sus dos dígitos es un sistema en base dos. Los dos dígitos binarios son 0 y 1. La posición de un 1 o un 0 en un número binario indica su peso dentro del número, así como la posición de un dígito decimal determina el valor de ese dígito. Los pesos de un número binario están basados en las potencias de dos.

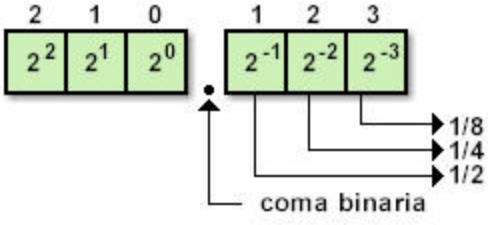
Base: 2
Símbolos: 0,1

El formato para números enteros y fraccionarios es similar al de los números binarios.

Para los números enteros



Para los números fraccionarios



Este sistema, presenta el inconveniente de que necesita muchas cifras para la representación de un número grande, y es muy engorroso para un humano.
Sin embargo, el sistema binario es el más adecuado a las máquinas electrónicas por varias razones:

1. La mayor parte de las computadoras existentes representan la información y la procesan mediante elementos y circuitos electrónicos de dos estados (relés, núcleos de ferrita, etc.).
2. Por la seguridad y la rapidez de respuesta de los elementos físicos de dos estados diferenciados (ON / OFF).
3. Las operaciones aritméticas son sencillas.

Los quince primeros números binarios se escriben:

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

CONVERSIÓN BINARIO – DECIMAL

La forma más sencilla de realizar esta conversión es desarrollando la fórmula que vimos para los sistemas posicionales (suma de potencias de la base).

$N = A_n \cdot B^n + A_{n-1} \cdot B^{n-1} + ... + A_1 \cdot B^1 + A_0 \cdot B^0$

Donde Ai son las distintas cifras del valor numérico e 'i' su posición.
B = 2

Ejemplo 1:

Dado el número binario: "1011_2", encontrar el equivalente decimal.

Si desarrollamos el número dado como potencias de 2 tendremos:

$1011_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 8 + 2 + 1 = 11_{10}$

Ejemplo 2:

Ahora vamos a realizar lo mismo pero con cifras decimales.

Dado el número binario: "$1011{,}011_2$", encontrar el equivalente decimal.

$1011{,}011_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 + 0 \cdot 1/2 + 1 \cdot ¼ + 1 \cdot 1/8 = 11{,}375_{10}$

CONVERSIÓN DECIMAL – BINARIO

Existen dos métodos, Suma de pesos y división sucesiva por 2.

1. Método de suma de pesos

Consiste en determinar el conjunto de pesos binarios, cuya suma es igual al numero decimal. Una forma sencilla de recordar los pesos binarios es que el peso más bajo es 1, es decir 2^0 y que duplicando cualquier peso se obtiene el peso superior así tendremos 1, 2, 4, 8, 16, 32, 64 ..... y así sucesivamente.

2. Método de la división sucesiva por 2.

Se divide sucesivamente el numero decimal entre 2. Cada cociente resultando se divide entre 2 hasta que se obtiene un cociente cuya parte entera es 0. Los restos generados en cada división forman el número binario. El primer resto es el bit menos significativo (LSB) del número binario, y el último resto es el bit más significativo (MSB).

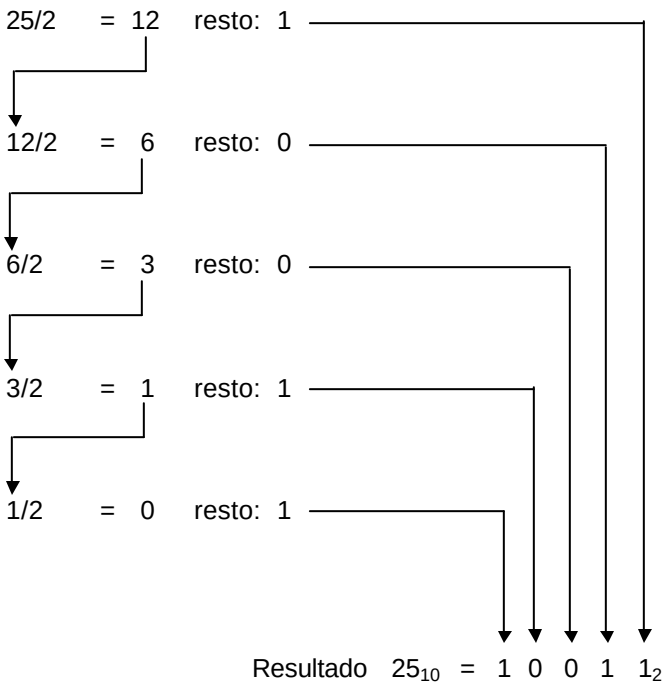
Ejemplo 1:

Vamos a obtener el equivalente binario del valor decimal: 25_{10}

1. Método de suma de pesos

$25_{10} = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 = 1\ 1\ 0\ 0\ 1$

2. Método de la división sucesiva por 2.



Si el número es decimal, se divide en parte entera y parte fraccionaria.

1. La **parte entera** se convierte utilizando uno de los dos métodos utilizados anteriormente.

2. La **parte fraccionaria** se puede convertirse utilizando dos métodos, suma de pesos o multiplicación sucesiva por 2. En este caso se multiplica la parte fraccionaria por 2 y después se multiplica cada parte fraccional resultante del producto por 2, hasta que el producto fraccionario sea 0 o hasta que se alcance el número deseado de posiciones decimales. Los dígitos acarreados ,o acarreos generados por la multiplicación dan lugar al número binario. El primer acarreo que se obtiene es el MSB y el último el LSB.

Ejemplo 2:

Expresar el número decimal $109{,}625_{10}$ en el sistema binario.

Parte entera

Podemos utilizar cualquiera de los dos métodos, pesos o división sucesiva. Para este caso utilizaremos el de los pesos.

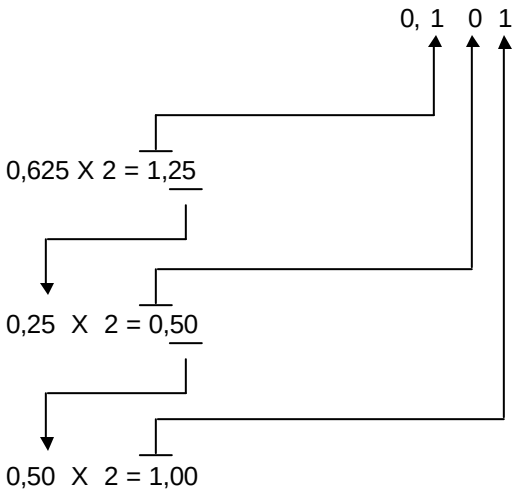
$109 = 64 + 32 + 8 + 4 + 1 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0 = 1101101_2$

Parte fraccionaria:

Método de suma de pesos:

$0,625 = 0,5 + 0,125 = 2^{-1} + 2^{-3} = 0,101_2$

Método de la multiplicación sucesiva:



El resultado final es la unión de ambos valores:

$109,625_{10} = 1101101,101_2$.

SISTEMA OCTAL

Este sistema tiene una base de ocho símbolos. La facilidad que existe en convertir entre el sistema binario y el octal, permite expresar los numeros binarios en un formato más compacto, ya que cada dígito octal equivale a 3 dígitos binarios.
Base: 8
Símbolos: 0,1,2,3,4,5,6,7

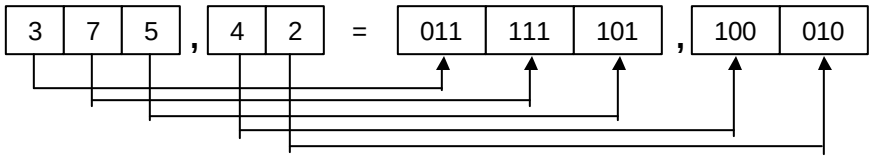
Los ocho primeros números octales se escriben:

Octal	Decimal	Binario
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

CONVERSIÓN OCTAL-BINARIA

Para convertir un número expresado en base 8 a base 2, simplemente sustituimos cada una de las cifras que lo forman por sus tres cifras binarias equivalentes.

Ejemplo: Convertir a Binario el número $375,42_8$

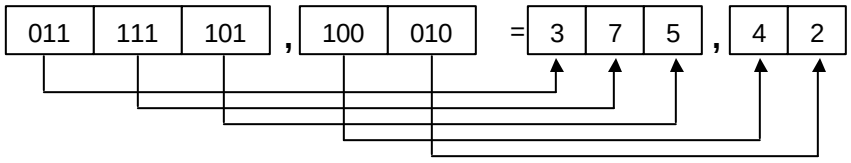


Con lo que tenemos que $375,42_8 = 011111101,100010_2$

CONVERSIÓN BINARIA-OCTAL

Se realiza a la inversa, comenzando desde la coma decimal hacia la izquierda para la parte entera, rellenando con 0's a la izquierda si fuera necesario; y desde la coma decimal hacia la derecha para la parte fraccionaria, rellenando con 0's a la derecha si fuera necesario.

Ejemplo: Convertir $11111101,100010_2$ a octal



Con lo que tenemos que $11111101,100010_2 = 375,42_8$

CONVERSIÓN OCTAL-DECIMAL

Se realiza del mismo modo que de binario a decimal, teniendo en cuenta que la base ahora es B=8.

Ejemplo: $345,5_8 = 3 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 + 5 \cdot 8^{-1} = 192 + 32 + 5 + 0,625 = 229,625_{10}$

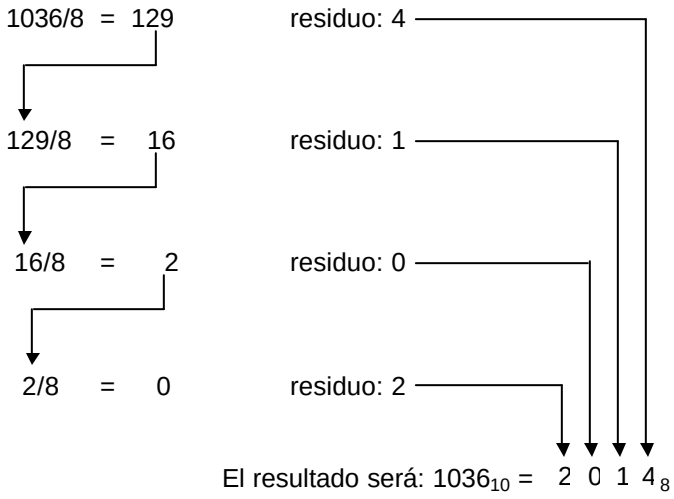
CONVERSIÓN DECIMAL - OCTAL

Se realiza del mismo modo que de decimal a binario, dividiendo la parte entera de forma sucesiva por la base B=8, y multiplicando la parte fraccionaria por la base.

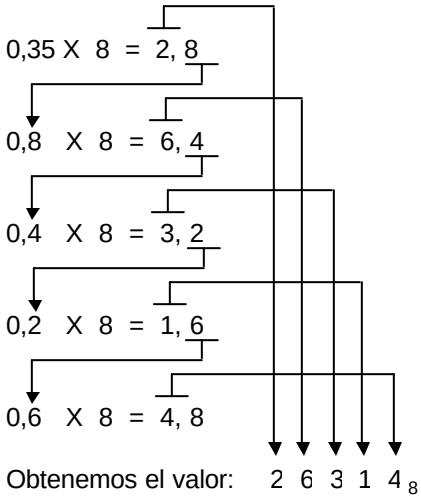
Ejemplo:

Expresar el número decimal $1036,35_{10}$ en octal.

Parte entera



Parte fraccionaria



El resultado final es la unión de ambos valores:

$1036,35_{10} = 2014,26314..._8$.

SISTEMA HEXADECIMAL

Al igual que el sistema octal este sistema da una forma mas compacta para representar los números binarios. Consta de 16 símbolos. Para indicar que el número se expresa en hexadecimal se suele colocar una H al final, por ejemplo $34AF_{16}$ puede indicarse como 34AFH

Base: 16
Símbolos: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

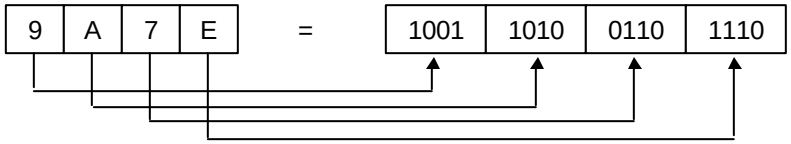
CONVERSIÓN HEXADECIMAL-BINARIO

Basta con sustituir cada símbolo hexadecimal por su equivalente en binario, según se indica en la tabla siguiente:

Hexadecimal	Decimal	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Ejemplo:

Hállese el equivalente binario del número $9A7E_{16}$



Con lo que tenemos que $9A7E_{16} = 1001101001101110_2$

CONVERSIÓN BINARIO-HEXADECIMAL

La conversión de un número binario a hexadecimal se realiza a la inversa: se forman grupos de cuatro cifras binarias a partir de la coma decimal, hacia la izquierda y hacia la derecha, y se sustituye cada grupo por su equivalente hexadecimal. Si el grupo final de la izquierda queda incompleto, se rellena con 0's por la izquierda. Del mismo modo, si el grupo final de la derecha queda incompleto, se rellena con 0's por la derecha.

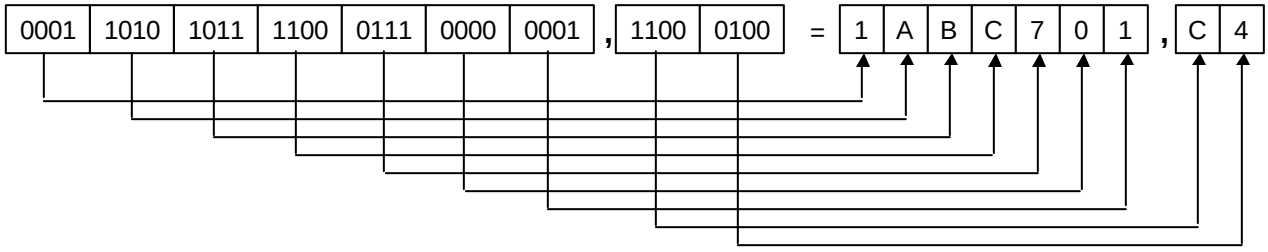
Ejemplo:

Calcúlese el equivalente hexadecimal del número binario :$11010101111000111000000001,110001_2$

Agrupamos y rellenamos con 0's:

$0001 \quad 1010 \quad 1011 \quad 1100 \quad 0111 \quad 0000 \quad 0001 \; , \; 1100 \quad 0100_2$

Sustituimos cada grupo de 4 por su equivalente hexadecimal:



Resultado: $1ABC701,C4_{16}$

CONVERSIÓN HEXADECIMAL-DECIMAL

La conversión se realiza siguiendo el mismo procedimiento que en las conversiones binario-decimal, pero considerando la base B=16. En este caso, además, deberemos sustituir los valores A, B, C, D, E, F por su equivalencia en el sistema decimal.

Ejemplo:

Hállese el equivalente decimal del valor hexadecimal $39,B8_{16}$.

$$39,B8_{16} = 3\cdot16^{1} + 9\cdot16^{0} + B\cdot16^{-1} + 8\cdot16^{-2} =$$
$$= 3\cdot16^{1} + 9\cdot16^{0} + 11\cdot16^{-1} + 8\cdot16^{-2} =$$
$$= 48 + 9 + 0.6875 + 0.03125 =$$
$$= 57,71875$$

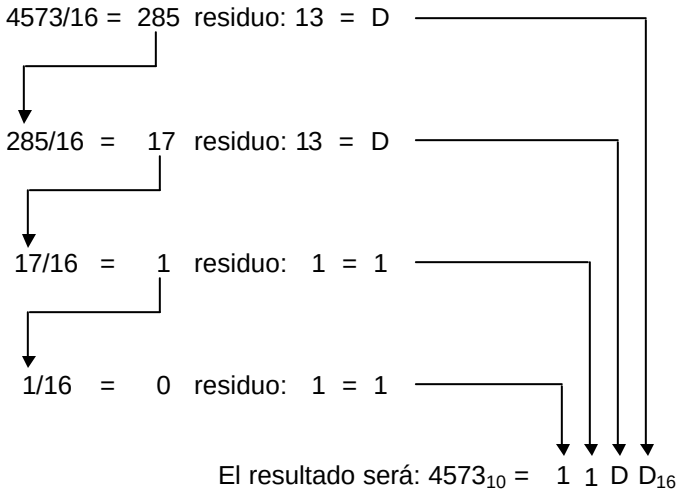
CONVERSIÓN DECIMAL- HEXADECIMAL

Procederemos del mismo modo que en la conversión decimal-binario, considerando B=16. Dividiremos la parte entera sucesivamente por la base, y la parte fraccionaria la multiplicaremos por la base.

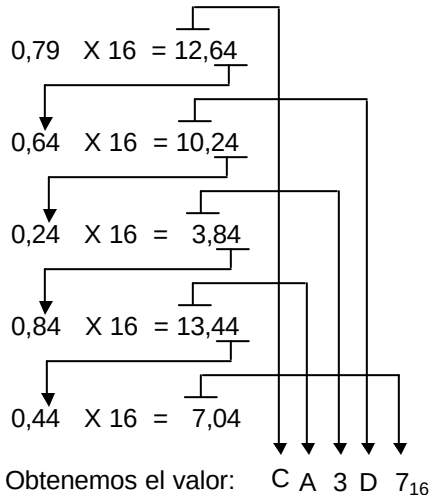
Ejemplo:

Hállese el equivalente hexadecimal del número $4573{,}79_{10}$.

Parte entera



Parte fraccionaria



El resultado final es la unión de ambos valores:

$11DD,CA3D7..._{16}$

ARITMÉTICA BINARIA

La aritmética binaria es básica en las computadoras digitales.

Las **operaciones aritméticas** que vamos a ver son las mismas que para el sistema decimal:

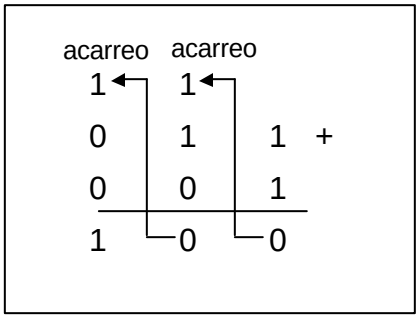
- suma
- resta
- multiplicación
- división, para la base binaria

SUMA BINARIA

La tabla de adición siguiente nos muestra las 4 reglas básicas para sumar dígitos binarios:

0 + 0 = 0	Suma = 0	Acarreo = 0
0 + 1 = 1	Suma = 1	Acarreo = 0
1 + 0 = 1	Suma = 1	Acarreo = 0
1 + 1 = 1 0	Suma = 0	Acarreo = 1

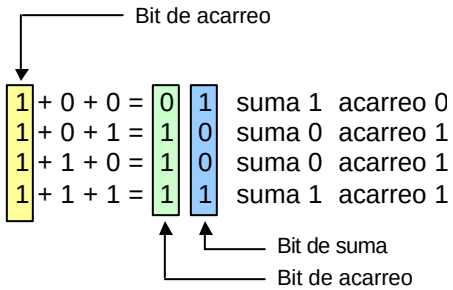
Puede verse que las primeras tres reglas dan lugar a un resultado de un solo bit, y la cuarta regla, la suma de dos unos, da lugar a 10(2 en binario). Cuando se suman números binarios, teniendo en cuenta la última regla se obtiene en la columna dada la suma 0 y un acarreo de 1 que pasa a la siguiente columna de la izquierda, como se muestra:



En la columna de la derecha 1 + 1 = 0 con acarreo 1, que pasa a la siguiente columna de la izquierda.

En la columna central, 1 + 1 + 0 = 0 con acarreo 1 que pasa a la siguiente columna de la izquierda. Y en la columna de la izquierda 1 + 0 + 0 = 1.

Cuando existe un acarreo igual a 1 se produce una situación en la que hay que sumar tres bits, un bit correspondiente a cada uno de los números y un bit de acarreo de esta forma tenemos:



La suma o adición es la operación aritmética de mayor importancia en los sistemas digitales. Como se verá más adelante, las operaciones de sustracción, multiplicación y división se realizan utilizando únicamente la adición como operación básica.

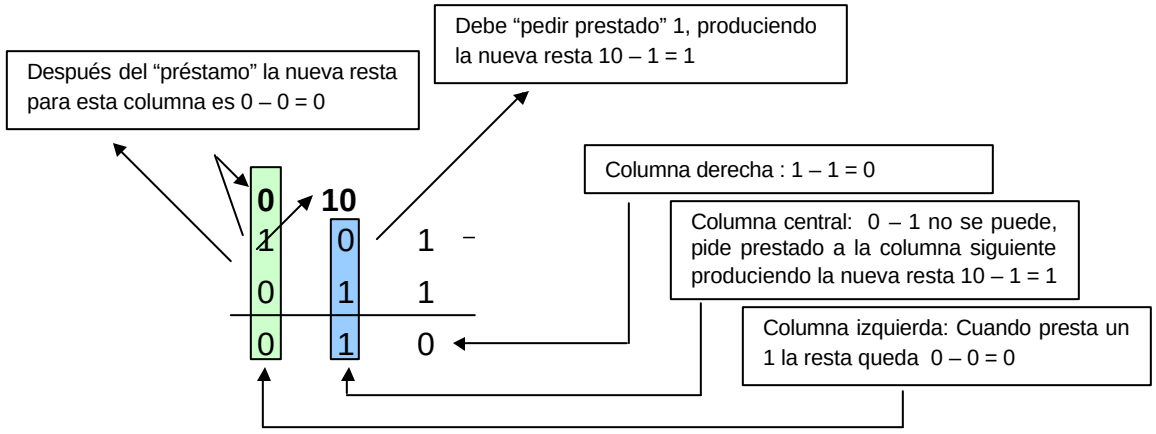
RESTA BINARIA

La tabla de siguiente nos muestra las 4 reglas básicas para restar dígitos binarios:

0 - 0 = 0
1 - 1 = 0
1 - 0 = 1
10 - 1 = 1 0 – 1 con acarreo negativo (prestamo) de 1

Cuando se restan números, algunas veces se genera un acarreo negativo que pasa a la siguiente columna de la izquierda. En binario esto sucede cuando se intenta restar 1 de 0. En este caso se pide prestado un 1 de la siguiente columna de la izquierda, y en la columna que se está restando se genera un 10. Veamos esto con un ejemplo:

Supongamos que queremos realizar la resta 5 – 3 = 2 en binario. Esto es 101 – 011



MULTIPLICACION BINARIA

La multiplicación de números binarios se realiza de la misma forma que con números decimales. Se realizan los productos parciales, desplazando cada producto parcial una posición a la izquierda, y luego se suman dichos productos. Es aun más sencilla que con números decimales, ya que en binario tan sólo tenemos dos dígitos: 0 y 1. Cuando multiplicamos por 0 obtenemos 0, y cuando multiplicamos por 1 obtenemos el mismo número.

Ejemplo: Calcular el producto 1100 x 1011.

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

DIVISION BINARIA

Se realiza del mismo modo que la división decimal. Por ejemplo dividir 110 entre 11

```
110 | 11
 11 | 10
 00
 00
```

El resultado será cociente 10 con resto 0

REPRESENTACIÓN DE NUMEROS CON SIGNO

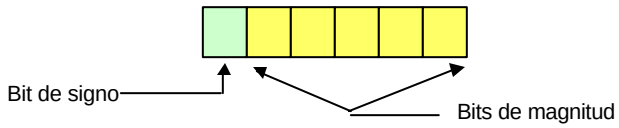
Los sistemas digitales deben ser capaces de manejar números positivos y negativos. Un número binario con signo queda determinado por su magnitud (valor) y su signo (positivo/negativo).
El símbolo "-" del sistema decimal no se puede representar en binario.

Debido a esto, existen 3 formatos de representación de números con signo:
- Signo y magnitud:
- Complemento a uno
- Complemento a dos

En todos ellos, el signo del número viene representado por un bit adicional, el "Bit de signo", que se coloca en el extremo izquierdo del número binario con signo. Se utiliza el siguiente convenio:
"0": signo positivo "1": signo negativo

Representación signo y magnitud

En el sistema de signo y magnitud, un número se compone de una magnitud y un símbolo que indica si la magnitud es positiva o negativa. Normalmente el bit del extremo izquierdo (MSB) como bit de signo, y los restantes representan el valor numérico del número en formato binario (magnitud).
Con n bits se podrán representar números que van desde $-(2^{n-1}-1)$ hasta $+(2^{n-1}-1)$ y existen dos representaciones posibles del cero.



Ejemplo:

- $01010101_2 = 85_{10}$
- $11010101_2 = -85_{10}$

Esta representación presenta ciertos inconvenientes:
- Pues para cualquier operación aritmética debemos comprobar primero el signo, para después sumar o restar en función del mismo.
- El diseño de circuitos lógicos que realicen operaciones aritméticas con números binarios en signo-magnitud no es fácil.
- Tenemos dos representaciones para el número 0:

Representación en complemento

La utilización de números complementados es un concepto especialmente útil para simplificar la resta, pues permite realizarla utilizando circuitos sumadores.
Los números positivos en este sistema se representan de la misma forma que los números positivos en formato signo-magnitud, simplemente se añade un 0 como bit de signo a la magnitud del número. El número 0 se identifica como positivo y tiene por tanto un bit de signo 0 y una magnitud de todos ceros.
Los números negativos son el complemento del correspondiente número positivo.

Existen dos variantes en la forma de complemento: el complemento a 1 y el complemento a 2

Complemento a uno

Como se dijo anteriormente, en notación complemento a 1 los números positivos se representan igual que en signo y magnitud. Los números negativos se obtienen cambiando todos los 0's por 1's y viceversa.
El rango de valores representables para un número de n bits va desde $-(2^{n-1}-1)$ hasta $+(2^{n-1}-1)$

Ejemplo:

- $01010101_2 = 85_{10}$
- $10101010_2 = -85_{10}$

Complemento a dos

Los números positivos se expresan igual que en signo y magnitud y en complemento a uno. Los números negativos se obtienen aplicando el complemento a 1 y sumándole 1

01010101	Numero
10101010 +	Complemento a 1
1	
10101011	Complemento a 2

El rango de valores posibles representables para un número de n bits va desde $-(2^{n-1})$ hasta $+(2^{n-1}-1)$

Ejemplo:

- $01010101_2 = 85_{10}$
- $10101011_2 = -85_{10}$

De los tres sistemas explicados para representar números con signo se prefiere el de complemento a 2 puesto que la circuitería relacionada a las operaciones aritméticas se hace mas sencilla.

En la tabla siguiente se muestran las diferentes representaciones para un número de 4 bits.

NUMERO	SIGNO Y MAGNITUD	COMPLEMENTO A 1	COMPLEMENTO A 2
0111	+ 7	+ 7	+ 7
0110	+ 6	+ 6	+ 6
0101	+ 5	+ 5	+ 5
0100	+ 4	+ 4	+ 4
0011	+ 3	+ 3	+ 3
0010	+ 2	+ 2	+ 2
0001	+ 1	+ 1	+ 1
0000	+ 0	+ 0	+ 0
1000	– 0	– 7	– 8
1001	– 1	– 6	– 7
1010	– 2	– 5	– 6
1011	– 3	– 4	– 5
1100	– 4	– 3	– 4
1101	– 5	– 2	– 3
1110	– 6	– 1	– 2
1111	– 7	– 0	– 1

Operaciones aritméticas de números con signo

ADICION

Cuando se suman dos números binarios A y B (A + B) pueden producirse cuatro casos:
- A y B son positivos ⇒ A + B ≥ 0
- A y B son negativos ⇒ A + B < 0
- A es positivo y B negativo, con A > |B| ⇒ A + B ≥ 0
- A es positivo y B negativo, con |B| > A ⇒ A + B < 0

En cualquier caso, el procedimiento de la suma es muy sencillo: sumar los dos números y descartar el bit de acarreo final (si lo hay).

Ejemplos:

Consideremos números con signo de 8 bits (n = 8).

- A y B son positivos (A = 7, B = 4)

7_{10} = 0000 0111
4_{10} = 0000 0100

```
0 0 0 0 0 1 1 1  +          7+
0 0 0 0 0 1 0 0             4
-----------------  ------>  ---
0 0 0 0 1 0 1 1             11
```

- A y B son negativos (A = – 5, B = – 9)

5_{10} = 0000 0101 : Complemento a 2 = 1111 1010 + 1 = 1111 1011
9_{10} = 0000 1001 : Complemento a 2 = 1111 0110 + 1 = 1111 0111

```
1 1 1 1 1 0 1 1  +          – 5+
1 1 1 1 0 1 1 1             – 9
-----------------  ------>  -----
1 1 1 1 1 0 0 1 0           – 14
```

Acarreo que se descarta → **1**

–14

Para comprobar el resultado se hace el proceso inverso al complemento a 2 →

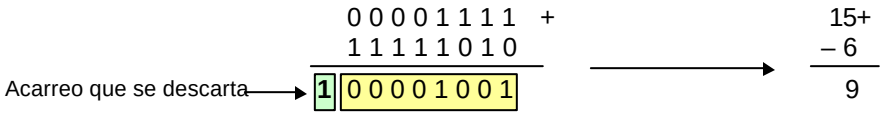
```
11110010  –
00000001
--------
11110001
00001110  → 14
```

En este caso tenemos acarreo, que descartamos. El resultado es 11110010. Como la suma es negativa, el resultado está en Complemento a 2.

Para comprobar el resultado se realiza el proceso inverso al complemento a 2 .(también se puede realizar el complemento a 2 del resultado negativo y se obtiene el equivalente positivo. Compruébelo)

• A es positivo y B negativo, con A > |B| (A = 15, B = – 6)

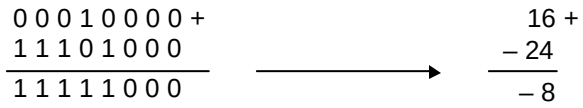
15_{10} = 0000 1111
6_{10} = 0000 0110: Complemento a 2 = 1111 1001 + 1 = 1111 1010



Como siempre, el bit de acarreo se desprecia. La suma es positiva

• A es positivo y B negativo, con |B| > A (A = 16, B = – 24)

16_{10} = 0001 0000
24_{10} = 0001 1000: Complemento a 2 = 1110 0111 + 1 = 1110 1000



La suma es negativa y por lo tanto está en complemento a 2.

Resumiendo: Para realizar la adición de dos números se suman los dos números representados en complemento a 2 en un sumador de n bits ignorando el acarreo del bit mas significativo. La suma será el valor algebraico correcto en la representación complemento a 2 siempre que la respuesta esté en el rango $-(2^{n-1})$ hasta $+(2^{n-1}-1)$

SUSTRACCION

La sustracción es un caso especial de la suma. Por ejemplo, restar + 6 (el sustraendo) de + 15 (minuendo) es equivalente a sumar – 6 a + 15. Básicamente la operación de sustracción cambia el signo del sustraendo y le suma el minuendo. El signo de un numero binario positivo o negativo se cambia calculándole su complemento a 2.

Para realizar la sustracción de dos números se obtiene el complemento a dos del sustraendo y se suman los dos números representados en complemento a 2 en un sumador de n bits ignorando el acarreo del bit mas significativo. Nuevamente el resultado será el valor algebraico correcto en la representación complemento a 2 siempre que la respuesta esté en el rango $-(2^{n-1})$ hasta $+(2^{n-1}-1)$

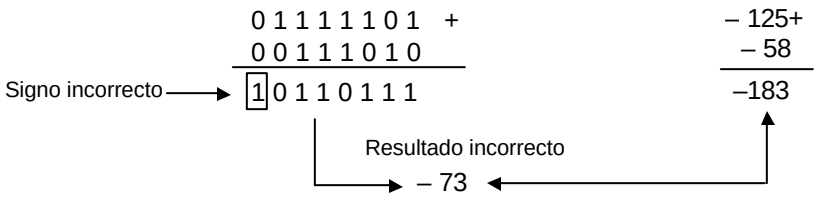
DESBORDAMIENTO

Hemos mencionado que el resultado será el valor algebraico correcto en la representación complemento a 2 siempre que la respuesta esté en el rango $-(2^{n-1})$ hasta $+(2^{n-1}-1)$.

Cuando se suman dos números y el número de bits requerido para representar la suma excede al número de bits de los dos números, se produce un desbordamiento (overflow), que se indica mediante un bit de signo incorrecto (p.e. negativo cuando A y B son positivos). Un desbordamiento puede producirse solo cuando ambos números son positivos o negativos. Veamos dos ejemplos con números de 8 bits. El rango representable es $-(2^{n-1})$ hasta $+(2^{n-1}-1)$ con n = 8.
Esto es de $-(2^{7})$ hasta $+(2^{7}-1)$ = de –128 hasta + 127.

• Suma de dos números positivos (A = – 125, B = – 58)

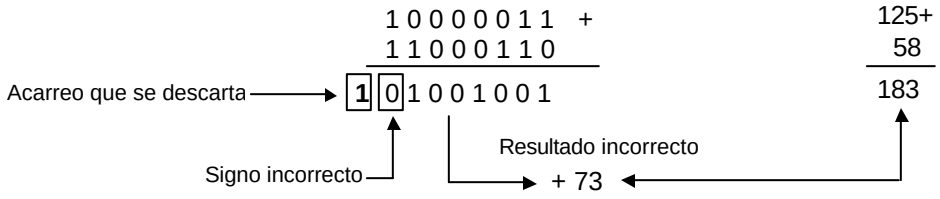
125_{10} = 0111 1101
58_{10} = 0011 1010



Note que al sumar dos números positivos cuyo resultado está fuera del rango representable el resultado es incorrecto, dando un número negativo.

• Suma de dos números negativos (A = 125, B = 58)

125_{10} = 0111 1101 Complemento a 2 = 1000 0010 + 1 = 1000 0011
58_{10} = 0011 1010 Complemento a 2 = 1100 0101 + 1 = 1100 0110



Note que al sumar dos números negativos cuyo resultado está fuera del rango representable el resultado es incorrecto, dando un número positivo.