CAPÍTULO 3

ARITMETICA Y FUNCIONAMIENTO DEL COMPUTADOR I

1. Sistemas de Numeración

El sistema decimal consiste en un sistema de numeración posicional que emplea el siguiente conjunto de símbolos: 0 1 2 3 4 5 6 7 8 9. Las cantidades numéricas que formamos con ellos vienen determinadas por los propios símbolos y la posición que ocupen los mismos. Las posiciones se identifican de forma relativa a donde se encuentre el punto decimal. Si no existiera punto decimal en la cantidad, se supone que está colocado de forma implícita a la derecha del todo de la cantidad.

Ejemplo de cómo se representan y se interpretan cantidades en el sistema decimal:

$$1325 = 1x103 + 3x102 + 2x101 + 5x100$$

$$1.235 = 1x100 + 2x10\text{-}1 + 3x10\text{-}2 + 5x10\text{-}3$$

El sistema binario es el sistema de numeración que emplea internamente el ordenador cuya base es el 2, por simplicidad en ocasiones optamos por el octal que equivale a 3 dígitos binarios o el hexadecimal que equivale a 4.

El sistema binario representa cantidades usando símbolos o dígitos 0 y 1 denominándose bits y Por encima del bit existen otras unidades que hay que conocer, sabiendo previamente que un byte (B) equivale a 8 bits:

Unidades de información digital[GC1] [CF2] [GC3]				
Almacenamiento	byte		**Transferencia (bit/s)**	
Prefijo + unidad	**Símbolo**	**Base 2**	**Prefijo + unidad**	**Símbolo**
byte	B	2 0	bit	b - bit
kilobyte	kB	2 10	kilobit	kb - kbit
megabyte	MB	2 20	megabit	Mb - Mbit
gigabyte	GB	2 30	gigabit	Gb - Gbit
terabyte	TB	2 40	terabit	Tb - Tbit
petabyte	PB	2 50	petabit	Pb - Pbit
exabyte	EB	2 60	exabit	Eb - Ebit
zettabyte	ZB	2 70	zettabit	Zb - Zbit
yottabyte	YB	2 80	yottabit	Yb - Ybit

El octal emplea los símbolos 0 1 2 3 4 5 6 7 y el hexadecimal 01 2 3 4 5 6 7 8 9 A B C D E F. Como se ha comentado se emplean mucho para trabajar de forma más cómoda con los binarios.

OCTAL	0	1	2	3	4	5	6	7
BINARIO	000	001	010	011	100	101	110	111

HEXADECIMAL	BINARIO
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

El teorema fundamental de la numeración relaciona la cantidad expresada con el sistema de numeración posicional cualquiera con su misma cantidad en decimal. En resumen, el teorema viene a decir que el valor decimal de una cantidad expresada en cualquier sistema de numeración posicional se obtiene mediante la fórmula:

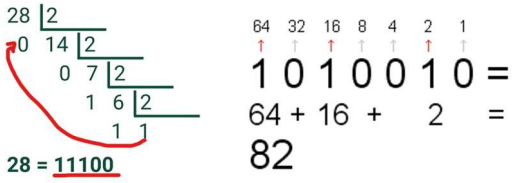
$$\sum X_i B^i$$

Por ejemplo: Imaginemos el número 543.1 expresado en un sistema de numeración con Base= 6. Por tanto, utiliza los dígitos 0 1 2 3 4 5 . Su valor decimal equivalente es:

543.3 en Base 6= $5 \times 6^2 + 4 \times 6^1 + 3 \times 6^0 + 3 \times 6\text{-}1 = 180 + 24 + 3 + 0.5 = 207.5$ en decimal.

2. Conversiones

Para convertir números enteros decimales a binarios, el proceso consiste en dividir sucesivamente el número decimal y los cocientes que se vayan obteniendo por 2, hasta que el cociente de alguna división sea 0. La unión de los diferentes restos obtenidos en sentido inverso nos da el número equivalente en binario.

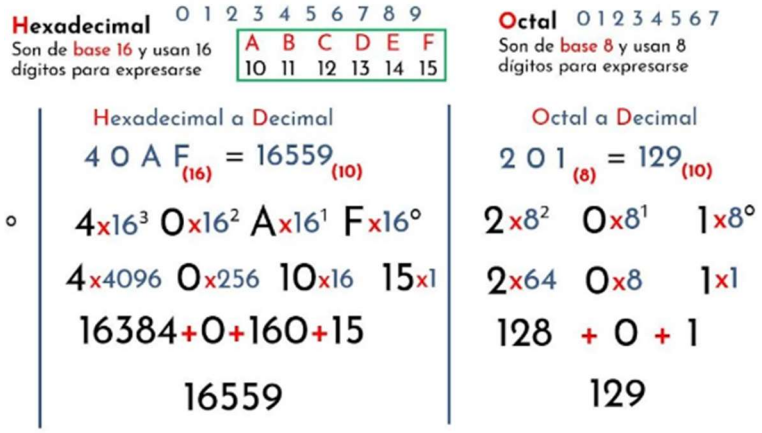


De binario a decimal básicamente hay que realizar las sumas de las potencias de 2 correspondientes a las posiciones (dígitos) que tienen un 1 en binario. Ejemplo:

Numero	1	0	1	0	1	0
Posición	5	4	3	2	1	0
Potencia de 2	32	16	8	4	2	1

101010= 32 + 0 + 8 + 0 + 2 + 0= 42 en decimal

Para pasar de decimal a octal/hexadecimal se usa el método explicado de divisiones sucesivas por 8 o 16 sucesivamente.



3. Operaciones aritméticas básicas en binario

La suma binaria es en cierto modo es semejante a la suma con números decimales con la salvedad de que en binario solo tenemos dos símbolos el 0 y el 1. Recordad que también existe el concepto de "acarreo".

Tabla del 0:

0+0 =0

0+1 =1

Tabla del 1:

1+0 =1

1+1 =10 (equivale a decir que a 0 con un acarreo de 1)

También es semejante a la resta con números. Solo tiene dos símbolos el 0 y el 1 y al ir realizando las restas parciales entre dos dígitos, si el sustraendo fuese mayor que el minuendo se sustrae una unidad del dígito o posición situado más a la izquierda en el minuendo. Es el concepto justamente opuesto al acarreo.

Tabla del 0:

0+0 =0

0 -1 -No cabe, habrá que sustraer a la izquierda

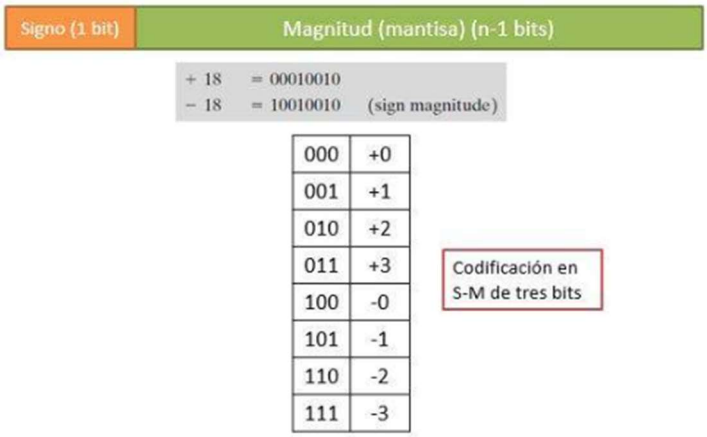
Tabla del 1:

1-0 =1

1-1 =0

4. Otras representaciones

Con el sistema binario, cualquier número puede representarse mediante 0 y 1, la coma (o punto) para los números decimales y un signo menos (-). No obstante, un procesador es únicamente capaz de manejar 0' s y 1' s. No es posible representar comas ni signos. Para solucionar este problema, se usan diferentes sistemas de representación que trabajan sobre los números en binario, añadiendo información de su signo y coma.



$$A = \begin{cases} \sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 1 \end{cases}$$

$4785 = 4x10^3 + 7x10^2 + 8x10^1 + 5x10^0$

$-4785 = -(4x10^3 + 7x10^2 + 8x10^1 + 5x10^0)$

Limitaciones que tiene esta representación

- El signo va por separado de la magnitud. Hace difíciles los cálculos en un computador.
- Hay dos representaciones para el número 0. Ej. Una representación de 8 bits:
  - + 0 = 0 0000000
  - - 0 = 1 0000000

Ventajas de este sistema

Como ventaja que presenta este sistema de representación frente a otros está la de poseer un rango de representación simétrico. El rango de representación es el conjunto de números representables. Así por ejemplo:

- Con 8 bits el rango va de -127 a + 127
- Para 16 bits de -32767 a 32767

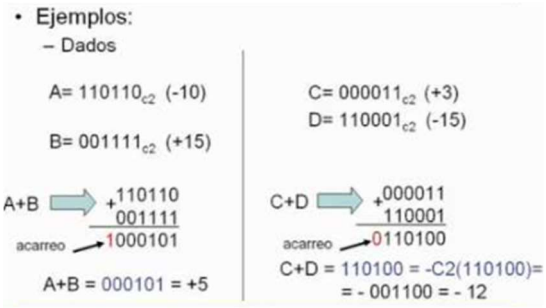
Toda la aritmética en los ordenadores es modular. Eso supone una limitación, Pero también tiene ventajas. En aritmética modular, la resta y la suma son la misma operación si se elige de una forma adecuada la representación de los números negativos. En aritmética modular, con módulo N, dos números A y B se dice que son equivalentes si se cumple que: A mod N = B mod N, y se escribe como (A=B) mod N. Si se cumple esta condición entonces existe un número entero tal que A= B + KN.

Complemento a 2

En la representación Complemento a 2 para representar números enteros con signo, al igual que en S-M, se utiliza el primer bit para codificar el signo. Varía la forma de codificar la magnitud: y con ello conseguimos ofrecer sólo una codificación para el 0.

Se acaba un número positivo, se dice que p es su número complemento a dos si se cumple que:

$$(k+p=0) \bmod 2^n$$



Complemento a 1

Sea k un número positivo, se dice que q es su complemento, a 1 si se cumple que:

$$(k+q=0) \bmod 2^n - 1$$

De forma resumida, el negativo de un número se obtiene cambiando (complementando) todos sus dígitos (ceros por unos y viceversa) incluido el bit de signo:

- Número 10 : 00001010 donde el cero de más ala izquierda indica el signo
- Número -10: 11110101

Este sistema de numeración tiene la ventaja de ser simétrico como el S-M pero también tiene la desventaja de tener 2 ceros (el +0 00000000 y el -0 11111111 para 8 bits por ejemplo). El rango es como en el caso del S-M.

5. Aritmética en enteros y sistemas de representación

Negación:

Esta operación es básicamente lo que ya hemos visto: Obtener el número negativo equivalente a otro dado. En S-M es trivial: basta con cambiar el bit del signo:

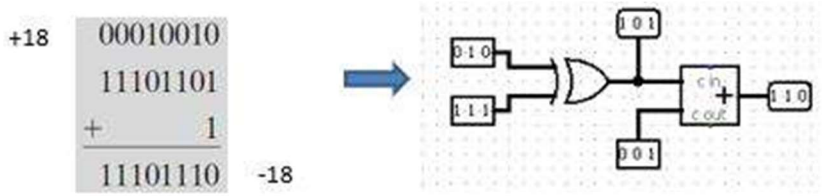
Número 10 : 00001010
Número -10: 10001010

En complemento a 1 hemos visto que hay que cambiar los 0 por 1 incluidos el del signo:

Número 10 : 00001010

Número -10: 11110101

En complemento 2 primero: Obtener el complemento (negación) de cada bit (0->1, 1->0).
Segundo: Sumar 1 al resultado.



Este proceso se llama transformación a complemento a dos u obtención del complemento a dos de un entero.

Es equivalente negar todos los dígitos haciendo XOR contra un número con la misma cantidad de dígitos binarios pero lleno de 1s y sumar 1 al resultado. Obtener el complemento a dos de -31 (100001).

Suma y Resta:

Primeramente, vamos a demostrar que la suma y resta pueden ser la misma operación en aritmética modular, si se escoge la representación adecuada de los números negativos. una vez lo hayamos demostrado procederemos a ver cómo se realiza dicha operación en cada una de las representaciones (complemento 1 y complemento a 2).

Para el caso de S-M se trata de una suma binaria o resta tradicional en la cual habrá que tener en cuenta el valor del signo.

Para demostrarlo volveremos a nuestro ejemplo del reloj módulo 16. queremos hacer la operación 6 – 3. primero encontramos la representación modular 16 del número -3:

$$p = 16 - 3 = 13$$

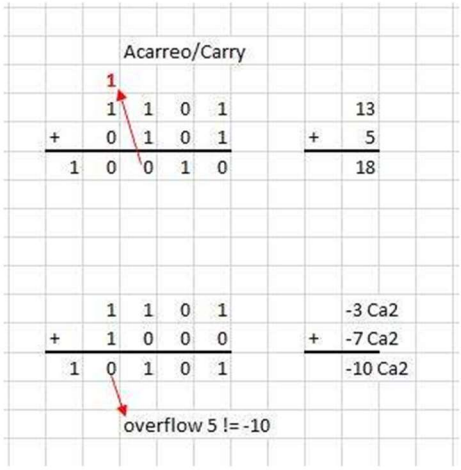
A continuación, sumamos modularmente:

$$(6+13) \bmod 16 = (19) \bmod 16 = 3$$

Hay que observar que 13 y -3 son componentes congruentes módulo 16.

Carry y overflow:

Carry es cuando hay desbordamiento del bit más significativo y trabajamos sin signo. overflow es cuando el signo queda es incoherente. Ejemplo:



Multiplicación sin signo:

Este algoritmo se conoce como suma-desplazamiento y comprende los siguientes pasos:

1. Se inicializan dos registros Q y M con el multiplicador y el multiplicando.
2. Se inicializa un registro A a 0 y un registro de un único bit C a 0.
3. Si Q0 es 1:
    1. Se suma A + M y se almacena en A. Si hay acarreo, se anota en C.
    2. Se desplazan todos los bits de C, A y Q una posición hacia la derecha.
        1. C -> A_{n-1}
        2. A0 -> Q_{n-1}
        3. Q_0 se pierde.
4. Si Q_0 es 0, sólo se realiza desplazamiento de C, A y Q.
5. Se vuelve a 3 hasta que se agoten los bits del multiplicador original.

El resultado del producto es la concatenación de A y Q.