Conversión Binario a Hexadecimal

1. **Busca una línea de hasta cuatro números binarios para hacer la conversión.** Los números binarios pueden ser 1 o 0. Los números hexadecimales pueden ser de 0 a 9 y de la A a la F, debido a que el sistema hexadecimal tiene base 16. Puedes convertir cualquier cadena de números binarios en hexadecimal (1, 01, 101101, etc.), pero necesitas cuatro números para hacer la conversión (0101→5; 1100→C, etc.).
   Si el número que quieres convertir tiene menos de 4 dígitos, agrega ceros a la izquierda hasta alcanzar los cuatro dígitos. Por ejemplo, 01 tendrías que convertirlo en 0001.

2. Escribe un "1" arriba del último dígito. Cada uno de los cuatro números significa otro número en el sistema decimal. El uno va en el último dígito. Ten en cuenta que no vas a elevar nada a ninguna potencia: esto es solo una pequeña ayuda para ver el valor de cada uno de los dígitos.
   Escribe un pequeño "2" arriba del tercer dígito, un "4" arriba del segundo y un "8" arriba del primero. Esos son los marcadores restantes. Estos números representan las distintas potencias de 2. El primero es 2^3, el segundo es 2^2, y así sucesivamente.
   1010 -> $1^8 0^4 1^2 0^1$

3. Una vez que tienes los cuatro números y ya sabes lo que significan, el proceso de conversión es fácil. Si tienes un 1 en el primer lugar, significa que tienes un 8. Si tienes un cero en la segunda columna, significa que no tienes ningún 4. La tercera columna indica cuántos 2 tienes y la cuarta cuántos 1. Así, por ejemplo:

   1010 -> $1^8 0^4 1^2 0^1$ -> 8 0 2 0

4. Ahora que tienes los números de cada marcador, debes sumarlos. El resultado que obtienes al hacer la suma es un número expresado en sistema decimal. Los números del 0 al 9 del sistema decimal y hexadecimal son exactamente iguales (por ejemplo, 1 [dec] = 1 [hex]). Sin embargo, si el número es mayor a 9, ya no cumple con esta propiedad y por lo tanto debes convertirlo de decimal a hexadecimal.
   $8 + 0 + 2 + 0 = 10_{10}$ -> A_{16}

   Por lo tanto el valor expresado en binario 1010_2 es igual a A_{16} representado en hexadecimal.

Complemento a 2

Este es un sistema que nos permite representar números binarios de forma negativa, en donde el MSB (Bit más Significativo) es el **bit del signo**.

¿Para qué sirve? Su utilidad principal se encuentra en las operaciones matemáticas con números binarios. En particular, la resta de números binarios se facilita enormemente utilizando el complemento a dos: la resta de dos números binarios puede obtenerse sumando al minuendo el complemento a dos del sustraendo.

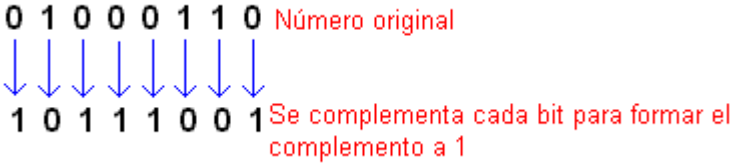
Si el bit de signo es 0 entonces el número binario es positivo (+), si el bit del signo es 1, entonces el número es negativo (-) los siete bits restantes del registro representan la magnitud del número **01000110**, para complementar mejor la explicación tendremos que dedicarle mucha atención a la explicación de conversiones donde interviene este tipo de numeración, que es bastante utilizado en los **microprocesadores**, ya que estos manejan tanto números positivos como números negativos.

Tener presente siempre que con un número binario de **8 bits sin bit de signo** podemos representar el rango de valores decimales desde el número 0 hasta el 255 inclusive.

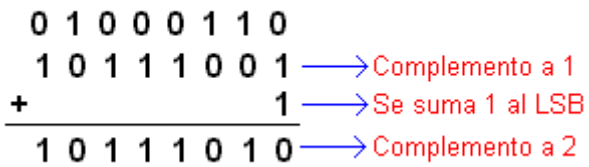
Aplicando este **sistema con bit de signo**, se reduce el rango para los números positivos, pero ganamos una forma de **representar números negativos**. El rango decimal que podemos representar con este nuevo sistema va desde el -128 al 127.

Para comprender mejor la conversión de este sistema de numeración, hay que tener en cuenta las siguientes definiciones:

El **complemento a 1** de un numero binario se obtiene cambiando cada 0 por 1 y viceversa. En otras palabras, se cambia cada bit del número por su complemento.



El **complemento a 2** de un numero binario se obtiene tomando el complemento a 1, y sumándole 1 al bit menos significativo. Continuando con el ejemplo anterior donde **01000110_2 = 70_{10}** :



Donde el nuevo obtenido del complemento a 2, **$1\,0\,1\,1\,1\,0\,1\,0_2$ = - 70_{10}** con el bit mas significativo (MSB) como bit de signo (**1**).

Binario (positivo) - Complemento a 2(negativo)	Decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Complemento a dos con enteros de 4 bits

Conceptos de acarreo y overflow

Desbordamiento u "Overflow": Cuando en una **suma con signo**, dos números positivos arrojan un resultado negativo, o dos números negativos arrojan un resultado positivo. En una **suma no signada** encontramos este bit de desbordamiento cuando el resultado no se encuentra dentro del rango establecido por la cantidad de bits para su representación.

Ejemplo

En el caso de utilizar 8 bits para representar el numero decimal en binario, sumamos los valores $250_{10} = 11111010_2$ con el valor $10_{10} = 00001010_2$. El valor resultante es 260_{10}, y el mayor al rango máximo que se puede representar en 8 bits (el valor máximo es 255_{10}).

Entonces $11111010_2 + 00001010_2 = 100000100$, donde 1 es el bit de overflow.

Acarreo de salida o “Carry out”: Cuando existe acarreo en el último bit de la suma

Ejemplo:

Suponemos la siguiente resta: el numero 89_{10} con el número utilizado anteriormente (70_{10}):

$$89 - 70 = 19$$

$$01011001 + 10111010 = 100010011$$

Observamos que el resultado posee un bit extra que puede ser tomado como desbordamiento, como vimos en el caso anterior. Pero la diferencia es que, en este caso, el valor resultado 19_{10} puede ser representado dentro del rango establecido por la representación binaria de 8 bits con signo (-128 a 127). Es decir, los primeros 8 bits menos significativos del resultado $00010011_2 = 19_{10}$. El bit MSB del resultado (1) es entonces descartado.

En resumen, cuando nos encontramos con un caso de overflow o desbordamiento diremos que también existe un acarreo (un número fuera de la cantidad de bits que se puede representar), pero si hay acarreo no necesariamente estamos en presencia de un overflow.

Estándar IEEE 754 single precision format

El **estándar del IEEE para aritmética en coma flotante** (**IEEE 754**) es la norma o estándar técnico para computación en coma flotante, establecida en 1985 por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE). La norma abordó muchos problemas encontrados en las diversas implementaciones de coma flotante que las hacían difíciles de usar de forma fiable y portátil. Muchas unidades de coma flotante de hardware utilizan ahora el estándar IEEE 754.

El estándar define:

- Formatos aritméticos: conjuntos de datos de coma flotante binarios y decimales, que consisten en números finitos, incluidos los ceros con signo y los números desnormalizados o subnormales, infinitos y valores especiales "no numéricos" (NaN).
- Formatos de intercambio: codificaciones (cadenas de bits) que se pueden utilizar para intercambiar datos de coma flotante de forma eficiente y compacta.
- Reglas de redondeo: propiedades que deben satisfacerse al redondear los números durante las operaciones aritméticas y las conversiones.
- Operaciones: operaciones aritméticas y otras (como funciones trigonométricas) en formatos aritméticos.
- Manejo de excepciones: indicaciones de condiciones excepcionales, tales como división por cero, desbordamiento, etc.

El estándar IEEE 754 especifica que un formato binary32 consta de:

- Bits de signo (S): 1 bit.
- Exponente desplazado (E): 8 bits.
- Mantisa (T): 24 bits (23 almacenados explícitamente).

El bit de signo (S) determina el signo del número, que también es el signo de la mantisa o significando. El exponente (E) es un entero sin signo (e) de 8 bits comprendido de 0 a 255 La mantisa real del formato incluye 23 bits de la fracción a la derecha de la coma binaria.

Ejemplo de representación

Represente el número -24,60 utilizando el estándar de coma flotante de simple precisión IEEE 754.

1. Representamos la parte entera del número en términos decimales:

   $24_{10} = 11000_2$

2. Para representar el valor decimal a binario realizamos lo siguiente:

   0.6 * 2 = 1.2
   0.2 * 2 = 0.4
   0.4 * 2 = 0.8
   0.8 * 2 = 1.6

Al valor 0.6 decimal del numero original se lo multiplica por el valor de la base al cual queremos convertir (en este caso 2). Tomamos el valor entero en rojo como parte de nuestro futuro valor decimal en binario. A la parte decimal del resultado se la vuelve a multiplicar por el valor base y así sucesivamente hasta llegar al valor de precisión deseado (4 dígitos es suficiente). Tomamos como MSB al primer numero obtenido de las multiplicaciones. Nuestro valor en binario es: 0.6_{10} = 0.1001

Por lo tanto 24.60 en binario es igual a 11000.1001_2

3. Ahora convertimos el valor binario en notación científica (el 1 más significativo queda como valor entero, el resto como decimal):
   1.10001001×2^4 -> El numero decimal formará parte de la mantisa del número a representar

4. El valor de la potencia puede ser negativo o positivo (en este caso positivo -> **4**) y tiene que ser sumado con el valor 127 que es la base para representar el exponente en IEEE 754. Luego de la suma representamos el resultado en binario:

   $127 + (+4) = 131_{10} = 10000011_2$ -> Este resultado representará al exponente.

5. Para finaliza observamos que el valor en decimal es negativo por lo tanto el bit de signo será 1.

6. Entonces nuestro valor representado es:
   -24.60_{10} = 1 10000011 10001001 00...0000
   S E T