# Codificación de números enteros y números reales.

# Contents

# Codificación de números enteros.

## Codificación con Signo y Magnitud.

La codificación de números enteros en binario se puede realizar de diferentes formas. La más sencilla se conoce con el nombre de “signo y magnitud”. Esta codificación está basada en la observación de que todo número entero se puede considerar como un número natural, su valor absoluto, y un signo positivo o negativo.

Así, para representar un número entero utilizando ”signo y magnitud”, se codifica el valor absoluto como un número natural y el signo utilizando un bit adicional, el de mayor peso, en el que el valor 0 representa el signo positivo y el valor 1 el signo negativo.

## Ejemplo de Signo y Magnitud.

Sea una representación en formato de Signo y Magnitud que nos permite codificar un número entero en binario con 8 bits (un byte). Esto nos otorga 1 bit para el signo y 7 bits para la magnitud.

Con 8 bits, podemos representar 256 números distintos. En este caso podemos representar del -127 al +127, en total solo 255 números diferentes, ya que con esta codificación el cero tiene dos maneras de representarse, como +0 y como -0, por lo que se desperdicia un posible número del espacio de representación.

Supongamos ahora, que tenemos que representar el número -57 (decimal). Procedemos a:

1. Tomar nota del signo del número: en el caso de -57, siendo negativo, llevará como bit de signo un 1.
2. Convertir el valor absoluto del número ab inario y representarlo con 7 bits: el valor absoluto de |-57| = 57, que en binario es 0111001.
3. Colocar todo junto: el número -57 en binario con formato de Signo y Magnitud es: 10111001.

## Suma de números naturales en binario.

Existe una codificación alternativa que se denomina “complemento a dos”. Para codificar los números de esta forma es necesario, como parte del proceso, realizar una operación de suma.

La suma números binarios se realiza de la misma forma que la de los números codificados en base 10, teniendo en cuenta que este caso cualquier cifra del número puede adquirir solamente dos valores (0 o 1). Así, al sumar dos dígitos el resultado es el que se indica a continuación:

```
0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0  (y nos llevamos 1, lo que se denomina acarreo)
```

## Complemento a dos (o a la base).

La codificación en complemento permite codificar $2^n$ números enteros consecutivos utilizando n bits, eliminado el problema del doble cero. El bit más significativo sigue indicando el signo del número con idéntica codificación que en el caso de signo y magnitud, los números positivos tienen este bit a cero y los negativos a uno.

Pero la propiedad más interesante de esta codificación es que las operaciones de suma y resta de números en complemento a dos siguen exactamente las mismas reglas que los números codificados en binario. Esta propiedad tiene una importancia enorme en el contexto de los circuitos digitales puesto que si se implementa un circuito que calcula la suma y resta de números naturales, dicho circuito se puede utilizar sin modificación alguna para sumar y restar enteros representados en complemento a 2.

La conversión de un número entero en base 10 a su representación con n bits en complemento a 2 se realiza mediante las siguientes reglas:

- Si el número mayor o igual que cero, la representación que le corresponde es directamente su traducción a base 2 con n bits.
- Si el número es negativo, su representación se obtiene mediante tres operaciones:
  - Obtener la representación del valor absoluto del número.
  - Reemplazar cada cero por un uno y cada uno por un cero. A esta operación también se le conoce como “negar” el número.
  - Sumar el valor 1 al número obtenido.

La traducción de un número N en complemento a dos con n bits a su representación en base 10 se realiza mediante las siguientes reglas:

- Si el número N es positivo, es decir, el bit de más peso es cero, el número en base 10 se obtiene igual que en el caso de los números naturales.
- Si el número N es negativo, es decir, el bit de más peso es uno, el número en base 10 se obtiene mediante la fórmula: $N - 2^n$, para lo que sería necesario realizar las siguientes operaciones:
  - Convertir el número a base 10 como si de un número natural se tratase.
  - Restar $2^n$ al número en base 10 (por ejemplo, para un número de 8 bits, hay que restar 256).

# Codificación de números reales.

La codificación de números reales utilizando lógica binaria es significativamente más compleja que el caso de los naturales o enteros. Parte de esta complejidad deriva del hecho de que si bien al utilizar un número finito de bits se representaba un intervalo concreto de números enteros o naturales, en el caso de los números reales esta técnica no es posible.

## Representación de números en coma flotante.

Los números reales se codifican en lógica binaria mediante la técnica conocida como “mantisa y exponente” que está basada en la siguiente observación: todo número real consta de una parte entera y una parte fraccionaria. La parte entera es aquella a la izquierda de la coma, y la parte fraccionaria a la derecha. A esta parte fraccionaria se le denomina “mantisa”. En el contexto de los números reales, la multiplicación y división por potencias de la base es equivalente al desplazamiento de la coma a lo largo de los diferentes dígitos que conforman el número.

Mediante operaciones de multiplicación y división por la base es posible representar todo número real por una mantisa en la que el primer dígito significativo está a la derecha de la coma.

La ventaja de la representación en coma flotante respecto a la representación en coma fija es que permite representar números muy pequeños o muy grandes de forma muy compacta. El número 0.0000000001 necesita 11 dígitos para representarse en punto fijo, sin embargo, en coma flotante utilizando base 10 tan sólo se precisa representar la mantisa 0.1 y el exponente -10.

### Elementos de la codificación: signo, mantisa y exponente.

La codificación de los números reales en base 2 consta de tres partes:

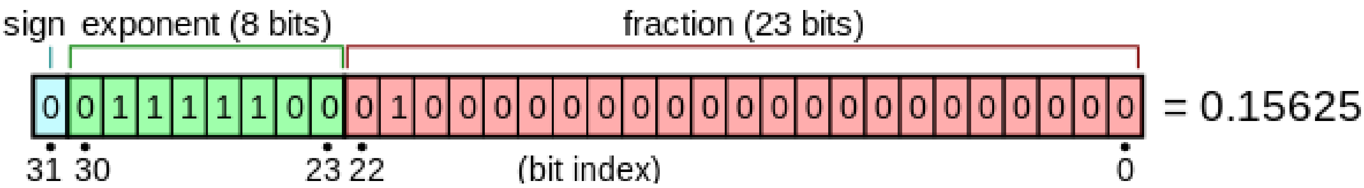
- Mantisa: representa la parte decimal del número y tiene una influencia directa sobre su precisión. Cuantos más bits se incluyan en la mantisa, mayor precisión tendrá la codificación.
- Signo de la mantisa: si es positivo se representa con un 0 y si es negativo con un 1.
- Exponente: es un número entero y por tanto se puede utilizar cualquiera de las técnicas de codificación presentadas anteriormente. La más común es complemento a 2.

### El estándar IEEE 754.

El Instituto de Ingeniería Eléctrica y Electrónica (en adelante IEEE) ha propuesto dos estándares para la representación de números reales, el IEEE 754 y el IEEE 854, de los cuales, el más utilizado es el IEEE 754. Esta representación tiene cuatro posibles precisiones: precisión simple, simple extendida, precisión doble, y doble extendida.

El estándar IEEE 754 de simple precisión especifica que un formato binario de 32 bits que consta de:

- Bit de signo: 1 bit.
- Exponente (E): 8 bits (desplazado: el valor real del exponente es E-127)
- Mantisa: 23 bits




## Fuente.

http://ocw.uc3m.es/ingenieria-telematica/arquitectura-de-ordenadores/lecturas/html/cod.html#cod:sec:inte
(CC-BY-NC-SA)

http://ocw.uc3m.es/ingenieria-telematica/arquitectura-de-ordenadores/lecturas/html/cod.html#cod:sec:floa
(CC-BY-NC-SA)

https://es.wikipedia.org/wiki/Representaci%C3%B3n_de_n%C3%BAmeros_con_signo (CC-BY-SA)

https://es.wikipedia.org/wiki/Formato_en_coma_flotante_de_simple_precisi%C3%B3n (CC-BY-SA)