# TEMA II. CODIFICACIÓN.

*Algoritmos de codificación, codificación de enteros, codificación de reales de coma fija, codificación de reales de coma flotante y el estándar IEEE 754.*

**Bibliografía:** Rafael Rico López "Codificación" Apuntes de Arquitectura de Computadores; Andrew S. Tanenbaum "Structured Computer Organization".

## Teorema fundamental de la numeración

Un número racional está compuesto por una ristra de dígitos, en donde, dependiendo del sistema posicional de numeración en el que esté representado el número (decimal, binario, hexadecimal…) cada uno de los dígitos tiene un cierto peso. Pongamos un ejemplo de ello:

Se tiene el siguiente número en sistema decimal: $\mathbf{6512,35_{10}}$

En donde cada dígito tiene los siguientes pesos:

| **6** | **5** | **1** | **2** | **3** | **5** |
|---|---|---|---|---|---|
| $10^3$ | $10^2$ | $10^1$ | $10^0$ | $10^{-1}$ | $10^{-2}$ |
| $d_3$ | $d_2$ | $d_1$ | $d_0$ | $d_{-1}$ | $d_{-2}$ |

El teorema fundamental de la numeración es una fórmula matemática que sirve para determinar el valor en decimal de un número racional representado en una cierta base numérica de sistema posicional. Formalmente se tiene que:

$$\boldsymbol{V_b = \sum_{i=-q}^{p-1} d_i \cdot b^i = \sum_{i=0}^{p-1} d_i \cdot b^i + \sum_{i=-q}^{-1} d_i \cdot b^i = [V], \{V\}}$$

En donde $\boldsymbol{p}$ es el número de dígitos de la parte entera del número y $\boldsymbol{q}$ es el número de dígitos de la parte fraccionaria del número. El valor entre corchetes $[\boldsymbol{V}]$ representa la parte entera del número, mientras que el valor entre llaves $\{\boldsymbol{V}\}$ es la parte fraccionaria del número.

Pongamos dos ejemplos del uso de esta fórmula, usando el número anteriormente descrito y un número representado en base 2 (en binario).

$$6512{,}35_{10} = \sum_{i=-2}^{3} d_i \cdot 10^i = 5 \cdot 10^{-2} + 3 \cdot 10^{-1} + 2 \cdot 10^0 + 1 \cdot 10^1 + 5 \cdot 10^2 + 6 \cdot 10^3 = \mathbf{6512,35}$$

$$101101{,}1_2 = \sum_{i=-1}^{5} d_i \cdot 2^i = 2^{-1} + 2^0 + 2^2 + 2^3 + 2^5 = 0{,}5 + 1 + 4 + 8 + 32 = \mathbf{45,5}$$

El valor $V_b$ es racional si puede expresarse como cociente de dos números enteros y es trivial demostrar que es así haciendo el siguiente cambio en el recorrido de los índices:

$$\boldsymbol{V_b = \frac{1}{b^q} \sum_{i=0}^{p+q-1} d_{i-q} \cdot b^i = \frac{A}{B}} \qquad \text{Entonces: } \boldsymbol{V \in \mathbb{Q}}$$

Pongamos dos ejemplos de ello:

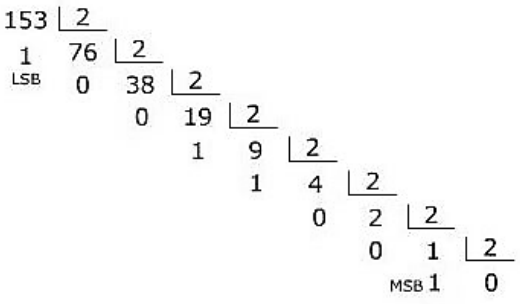
$$\boldsymbol{69,36_{10} = \frac{1}{10^2} \sum_{i=0}^{3} d_{i-2} \cdot 10^i = \frac{6 \cdot 10^0 + 3 \cdot 10^1 + 9 \cdot 10^2 + 6 \cdot 10^3}{10^2} = \frac{6936}{100}}$$

$$\boldsymbol{101101,1_2 = \frac{1}{2^1} \sum_{i=0}^{6} d_{i-2} \cdot 2^i = \frac{2^0 + 2^1 + 2^3 + 2^4 + 2^6}{2^1} = \frac{91}{2} = 45,5}$$

## Codificación de enteros

El algoritmo de codificación de números enteros consiste en dividir un valor entero $[\boldsymbol{V}]$ decimal entre la base $\boldsymbol{b}$ a la que se desee codificar. Esta operación genera un cociente $\boldsymbol{Q_0}$ y un resto $\boldsymbol{d_0}$. A continuación, se toma el cociente obtenido y se divide entre $\boldsymbol{b}$ obteniendo un nuevo cociente y resto. Este paso se repite sucesivamente hasta que el cociente sea cero. La codificación resultante será la ristra de dígitos $\boldsymbol{d_i}$ obtenidos, es decir, los restos obtenidos en cada división, siendo el primer resto el dígito menos significativo del número codificado.

Ejemplo de la codificación entera a binario (base 2) del número decimal $153_{10}$:




Siendo el número codificado en base 2 (binario) igual a: $10011001_2$

Se puede aplicar el teorema fundamental de la numeración para verificar si el número codificado es igual al número en su representación original:

$$V = 2^0 + 2^3 + 2^4 + 2^7 = 153 \quad \checkmark$$

## Codificación fraccionaria

El algoritmo de codificación de la parte fraccionaria consiste en multiplicar un valor fraccionario $\{\boldsymbol{V}\}$ decimal por la base $\boldsymbol{b}$ a la que se quiere codificar. Esta operación genera el producto $\boldsymbol{d_{-1}}$. A continuación, se toma la parte fraccionaria del producto obtenido $\{\boldsymbol{d_{-1}}\}$ y se multiplica por $\boldsymbol{b}$ obteniendo un nuevo producto. Este paso se repite sucesivamente hasta que la parte fraccionaria de algún producto sea cero o hasta que hayamos calculado el número $\boldsymbol{q}$ de dígitos fraccionarios especificado por el problema. La codificación resultante será la ristra de dígitos $[\boldsymbol{d_i}]$ obtenidos, es decir, la parte entera de los productos obtenidos tras cada multiplicación, siendo el primer producto el primer decimal del número codificado.

Ejemplo de la codificación fraccionaria a binario (base 2) del número decimal $0,75_{10}$:

$\boldsymbol{0,75 \times 2 = 1,5} \qquad \boldsymbol{d_{-1} = 1}$
$\boldsymbol{0,50 \times 2 = 1,0} \qquad \boldsymbol{d_{-2} = 1}$ Siendo la codificación igual a: $\boldsymbol{0,11_2}$

## Operador $TAM$

El operador $\boldsymbol{TAM(V, b)}$ devuelve el número de dígitos necesarios para representar un número $\boldsymbol{V}$ en un cierto sistema posicional de base $\boldsymbol{b}$.

$$\boldsymbol{TAM(V,b) = \lfloor \log_b V \rfloor + 1 = \left\lfloor \frac{\log_{10} V}{\log_{10} b} \right\rfloor + 1}$$

Ejemplo: Calcular el número de bits necesarios para representar el número 235 en binario.

$TAM(235, 2) = \lfloor \log_2 235 \rfloor + 1 = \lfloor 7{,}87 \rfloor + 1 = 7 + 1 = \mathbf{8\ bits}$

## Operador $BIN$

El operador $\boldsymbol{BIN(V, n)}$ realiza la conversión de un número decimal $\boldsymbol{V}$ a binario, representado en $\boldsymbol{n}$ bits. Para realizar la conversión se usa el algoritmo de codificación de enteros (divisiones sucesivas entre 2), obteniéndose así el número codificado $\boldsymbol{V_2}$. Si el número de bits necesarios para representar el número codificado en base 2 es mayor que el número de bits con el que trabajamos, entonces se producirá un desbordamiento ya que el número no cabrá en los $n$ bits.

$$\boldsymbol{BIN(V,n)} \begin{cases} \boldsymbol{V_{2,n}} & \textbf{con } \boldsymbol{d = 0} \quad \textbf{si } \boldsymbol{n \leq TAM(V,2)} \\ & \textbf{con } \boldsymbol{d = 1} \quad \textbf{si } \boldsymbol{n > TAM(V,2)} \end{cases}$$

Otra forma para determinar si se produce desbordamiento es comprobando si el número $V$ pertenece al rango de representación de los números en binario puro de $n$ bits. También se deduce el desbordamiento si el cociente tras hacer las $n$ divisiones sucesivas no es 0.

$$\textbf{Rango } \boldsymbol{BIN_n = [0\,, 2^n - 1]}$$

Ejemplo: Calcular el binario de 235 delimitado a 6 y 8 bits y determinar el desbordamiento.

$TAM(235, 2) = 8$
Rango $BIN_6 = [0\,, 63]$
Rango $BIN_8 = [0\,, 255]$

$BIN(235, 6) = \mathbf{10\ 1011\ con\ } \boldsymbol{d = 1}$ se produce desbordamiento (“no cabe en 6 bits”)
$BIN(235, 8) = \mathbf{1110\ 1011\ con\ } \boldsymbol{d = 0}$ no se produce desbordamiento

## Operador $VAL\text{-}BIN$

El operador $\boldsymbol{VAL\text{-}BIN(V_{2,n})}$ realiza la decodificación de un número binario $\boldsymbol{V_2}$ representado en $\boldsymbol{n}$ bits a decimal utilizando la fórmula del teorema fundamental de la numeración.

$$\boldsymbol{VAL\text{-}BIN(V_2, n) = \sum_{i=0}^{n-1} d_i \cdot 2^i}$$

Ejemplo: Decodificar el valor binario $11101011_{2,8}$ a decimal.

$$VAL\text{-}BIN\left(11101011_{2,8}\right) = \sum_{i=0}^{7} d_i \cdot 2^i = 2^0 + 2^1 + 2^3 + 2^5 + 2^6 + 2^7 = \mathbf{235}$$

## Operador $SM$

El operador $\boldsymbol{SM(Z,n)}$ codifica un número entero $\boldsymbol{Z}$ a binario en signo-magnitud representado sobre $\boldsymbol{n}$ bits, en donde el bit más significativo indica el signo y el resto de bits el valor absoluto del número codificado. Si el **MSB** es 0, entonces el número es positivo. En cambio, si es 1, el número es negativo.

$$\underbrace{\overbrace{\boxed{\mathbf{0}}}^{\textbf{signo}}\ \underbrace{\mathbf{0}\ \cdots\ \mathbf{0}\ \mathbf{0}\ \mathbf{0}\ \mathbf{0}\ \mathbf{0}}_{\textbf{magnitud}}}_{\textbf{n bits}}$$

$$\boldsymbol{SM(Z,n)} = \begin{cases} \boldsymbol{Z_{SM,n} = BIN(|Z|, n-1)} & \textbf{con signo} = \mathbf{0} \quad \textbf{si } \boldsymbol{Z \geq 0} \\ & \textbf{con signo} = \mathbf{1} \quad \textbf{si } \boldsymbol{Z < 0} \end{cases}$$

$$\textbf{Rango } \boldsymbol{SM_n} = [-(\mathbf{2^{n-1}-1}), \mathbf{2^{n-1}-1}] \text{ Es simétrico (tiene dos ceros } \{-0, +0\})$$

Ejemplo: Calcular los valores 114 y $-34$ en signo magnitud delimitado a 8 bits.

$$SM(114,8) = \begin{Bmatrix} BIN(114,7) \\ \text{con signo} = 0 \end{Bmatrix} = \boxed{\mathbf{0}}\ \mathbf{111\ 0010} \quad \textbf{con } \boldsymbol{d} = \mathbf{0}$$

$$SM(-34,8) = \begin{Bmatrix} BIN(34,7) \\ \text{con signo} = 1 \end{Bmatrix} = \boxed{\mathbf{1}}\ \mathbf{010\ 0010} \quad \textbf{con } \boldsymbol{d} = \mathbf{0}$$

## Operador $VAL\text{-}SM$

El operador $\boldsymbol{VAL\text{-}SM(Z_{SM,n})}$ decodifica un número binario en signo-magnitud $\boldsymbol{Z_{SM,n}}$ a decimal utilizando la fórmula del teorema fundamental de la numeración, teniendo en cuenta el signo.

$$\boldsymbol{VAL\text{-}SM\left(Z_{SM,n}\right) = (-1)^{d_{n-1}} \sum_{i=0}^{n-2} d_i \cdot 2^i = (-1)^{d_{n-1}} \cdot VAL\text{-}BIN\left(Z_{SM,n-1}\right)}$$

Ejemplo: Decodificar el valor $10100010_{SM,8}$ a decimal.

$$VAL\text{-}SM\left(10100010_{SM,8}\right) = (-1)^1 \sum_{i=0}^{6} d_i \cdot 2^i = -\left(2^1 + 2^5\right) = -(2+32) = \mathbf{-34}$$

## Operador $C2$

El operador $\boldsymbol{C2(Z,n)}$ codifica un número entero $\boldsymbol{Z}$ a binario en complemento a 2 representado sobre $n$ bits, en donde el MSB indica el signo. El complemento a 2 de un número $x$ en cierta base $b$ con $n$ dígitos (también llamado el complementario a la base) se calcula mediante la siguiente fórmula: $C2(x_b) = b^n - |x_b|$. Con lo que se demuestra que $|x_b| + C2(x_b) = b^n$

$$\boldsymbol{C2(Z,n)} = \begin{cases} \textbf{si } \boldsymbol{Z \geq 0} & \boldsymbol{Z_{C2} = BIN(Z,n)} \\ \textbf{si } \boldsymbol{Z < 0} & \boldsymbol{Z_{C2} = BIN(2^n - |Z|, n)} \end{cases}$$

$$\textbf{Rango } \boldsymbol{C2_n} = [-(\mathbf{2^{n-1}}), \mathbf{2^{n-1}-1}] \text{ No es simétrico (solo tiene un cero } \{+0\})$$

Ejemplo: Calcular el valor $-34$ en complemento a 2 delimitado a 8 bits.

$$C2(-34,8) = BIN(2^8 - 34, 8) = BIN(222,8) = \mathbf{11011110}$$

## Operador $\boldsymbol{VAL\text{-}C2}$

El operador $\boldsymbol{VAL\text{-}C2(Z_{C2,n})}$ decodifica un número binario en complemento a 2 $\boldsymbol{Z_{C2,n}}$ a decimal utilizando la fórmula del teorema fundamental de la numeración.

$$\boldsymbol{VAL\text{-}C2(Z_{C2,n}) = \sum_{i=0}^{n-2} d_i \cdot 2^i - d_{n-1} \cdot 2^{n-1}}$$

$$\boldsymbol{VAL\text{-}C2(Z_{C2,n}) = VAL\text{-}BIN(Z_{C2,n-1}) - d_{n-1} \cdot 2^{n-1}}$$

Ejemplo: Decodificar los valores $11011110_{C2,8}$ y $00011110_{C2,8}$ a decimal.

$$VAL\text{-}C2(11011110_{C2,8}) = \sum_{i=0}^{6} d_i \cdot 2^i - 2^7 = (2^1 + 2^2 + 2^3 + 2^4 + 2^6) - 2^7 = \mathbf{-34}$$

$$VAL\text{-}C2(00011110_{C2,8}) = \sum_{i=0}^{6} d_i \cdot 2^i = (2^1 + 2^2 + 2^3 + 2^4) = \mathbf{94}$$

## Operador $\boldsymbol{C1}$

El operador $C1(Z, n)$ codifica un número entero $Z$ a binario en complemento a 1 representado sobre $n$ bits, en donde el MSB indica el signo. El complemento a 1 de un número $x$ en cierta base $b$ con $n$ dígitos funciona complementando a la base menos 1 ($b^n - x - 1$).

$$\boldsymbol{C1(Z,n)} = \begin{cases} \textbf{si } \boldsymbol{Z \geq 0} & \boldsymbol{Z_{C1} = BIN(Z,n)} \\ \textbf{si } \boldsymbol{Z < 0} & \boldsymbol{Z_{C1} = BIN(2^n - 1 - |Z|, n)} \end{cases}$$

$$\textbf{Rango } \boldsymbol{C1_n} = [-(\boldsymbol{2^{n-1} - 1}), \boldsymbol{2^{n-1} - 1}] \text{ Es simétrico (tiene dos ceros } \{-0, +0\})$$

Ejemplo: Calcular el valor $-34$ en complemento a 1 delimitado a 8 bits.

$C1(-34, 8) = BIN(2^8 - 1 - 34, 8) = BIN(221, 8) = \mathbf{11011101}$

## Operador $\boldsymbol{VAL\text{-}C1}$

El operador $VAL\text{-}C1(Z_{C1,n})$ decodifica un número binario en complemento a 1 $Z_{C1}$ representado sobre $n$ bits a decimal utilizando la fórmula del teorema fundamental de la numeración.

$$\boldsymbol{VAL\text{-}C1(Z_{C1,n}) = \sum_{i=0}^{n-2} d_i \cdot 2^i - d_{n-1} \cdot 2^{n-1} + d_{n-1}}$$

$$\boldsymbol{VAL\text{-}C1(Z_{C1,n}) = VAL\text{-}BIN(Z_{C1,n-1}) - d_{n-1} \cdot 2^{n-1} + d_{n-1}}$$

Ejemplo: Decodificar los valores $11011101_{C1,8}$ y $01110_{C1,5}$ a decimal.

$$VAL\text{-}C1(11011101_{C1,8}) = \sum_{i=0}^{6} d_i \cdot 2^i - 1 \cdot 2^7 + 1 = 2^0 + 2^2 + 2^3 + 2^4 + 2^6 - 2^7 + 1 = \mathbf{-34}$$

$$VAL\text{-}C1(01110_{C1,5}) = \sum_{i=0}^{3} d_i \cdot 2^i - 0 \cdot 2^4 + 0 = (2^1 + 2^2 + 2^3) = \mathbf{14}$$

## Operador $EX2^{n-1}$

El operador $EX2^{n-1}(Z,n)$ codifica un número entero $Z$ a binario en formato de Exceso a $M$ $(2^{n-1})$ representado sobre $n$ bits, en donde el MSB indica el signo (ahora el 0 representa a los negativos y el 1 a los positivos). El exceso a $M$ de un número binario se consigue sumando $2^{n-1}$ al valor que se desea codificar.

$$\boldsymbol{EX2^{n-1}(Z,n) = BIN(2^{n-1} + Z, n)}$$

$$\textbf{Rango } \boldsymbol{EX2^{n-1}}_{\boldsymbol{n}} = [-(\boldsymbol{2^{n-1}}), \boldsymbol{2^{n-1} - 1}] \text{ No es simétrico (solo tiene un cero } \{+0\})$$

Ejemplo: Calcular el valores 32 y $-12$ en exceso a $2^{n-1}$ delimitado a 8 bits.

$EX2^{n-1}(32,8) = BIN(2^8 + 32, 8) = BIN(160,8) = \mathbf{10100000}$

## Operador $VAL\text{-}EX2^{n-1}$

El operador $VAL\text{-}EX2^{n-1}(Z_{EX2,n})$ decodifica un número binario en exceso $Z_{EX2}$ representado sobre $n$ bits a decimal utilizando la fórmula del teorema fundamental de la numeración.

$$\boldsymbol{VAL\text{-}EX2^{n-1}(Z_{EX2,n}) = \sum_{i=0}^{n-2} d_i \cdot 2^i - \overline{d_{n-1}} \cdot 2^{n-1}}$$

$$\boldsymbol{VAL\text{-}EX2^{n-1}(Z_{EX2,n}) = VAL\text{-}BIN(Z_{EX2}, n-1) - \overline{d_{n-1}} \cdot 2^{n-1}}$$

Ejemplo: Decodificar los valores $11011101_{EX2^{n-1},8}$ y $01110_{EX2^{n-1},5}$ a decimal.

$$VAL\text{-}EX2^{n-1}(11011101_{EX2,8}) = \sum_{i=0}^{6} d_i \cdot 2^i - \overline{1} \cdot 2^7 = (2^0 + 2^2 + 2^3 + 2^4 + 2^6) = \mathbf{93}$$

$$VAL\text{-}EX2^{n-1}(01110_{EX2,5}) = \sum_{i=0}^{3} d_i \cdot 2^i - \overline{0} \cdot 2^4 = (2^1 + 2^2 + 2^3) - 2^4 = 14 - 16 = \mathbf{-2}$$

## Números reales de coma fija

Para codificar un número real en las diferentes representaciones binarias se pueden aplicar varios métodos. Los operadores anteriormente vistos $BIN, SM, C1, C2$ y $EX2^{n-1}$ solo se pueden aplicar sobre números enteros $Z$, salvo el operador $BIN$ que también acepta reales, en donde se aplicaría el método de codificación fraccionaria para los decimales. En cambio, en el resto de operadores habría que realizar modificaciones para que puedan recibir números reales.

Como se había visto anteriormente, un número racional puede interpretarse como el cociente de dos enteros:

$$\boldsymbol{R = \frac{Z}{b^q}}$$

En donde $\boldsymbol{q}$ es el número de decimales del número real, $\boldsymbol{b}$ es la base en la que está representada el número y $\boldsymbol{Z}$ sería un número entero auxiliar que se obtiene de la siguiente manera:

$$\boldsymbol{Z = R \cdot b^q}$$

Por lo tanto, los operadores de codificación binaria quedarían definidos de la siguiente manera. El operador ***BIN*** recibe ahora un tercer parámetro $\boldsymbol{q}$, el cual indica el número de decimales:

$$BIN(R,n,q) = BIN\left(\frac{Z}{2^q},n,q\right) = \begin{cases} R_{2,n,q} & \text{con } d=0 \quad \text{si } n \le TAM([R],2) \\ & \text{con } d=1 \quad \text{si } n > TAM([R],2) \end{cases}$$

$$SM(R,n,q) = \begin{cases} BIN\left(\frac{|Z|}{2^q},n-1,q\right) = BIN(|R|,n-1,q) & \text{con signo} = 0 \quad \text{si } R \ge 0 \\ & \text{con signo} = 1 \quad \text{si } R < 0 \end{cases}$$

$$C2(R,n,q) = \begin{cases} \text{si } R \ge 0 & R_{C2,n,q} = BIN\left(\frac{Z}{2^q},n,q\right) = BIN(R,n,q) \\ \text{si } R < 0 & R_{C2,n,q} = BIN\left(\frac{2^n - |Z|}{2^q},n,q\right) = BIN(2^{n-q} - |R|) \end{cases}$$

$$C1(R,n,q) = \begin{cases} \text{si } R \ge 0 & R_{C1,n,q} = BIN\left(\frac{Z}{2^q},n,q\right) = BIN(R,n,q) \\ \text{si } R < 0 & R_{C1,n,q} = BIN\left(\frac{2^n - 1 - |Z|}{2^q},n,q\right) = BIN(2^{n-q} - 2^{-q} - |R|) \end{cases}$$

$$EX2^{n-1}(R,n,q) = \left\{BIN\left(\frac{2^{n-1}+Z}{2^q}\right) = BIN(2^{n-q-1}+R)\right.$$

Podemos optar por dos opciones a la hora de usar estos operandos.

1. Usar los operandos tal cual como están descritos y codificar tantos las cifras enteras y fraccionarios usando los métodos de codificación entera y fraccionaria.
   $BIN(R,n,q) = R_{2,n,q}$

2. Multiplicar por $2^q$ el número real y codificar la parte entera del producto obtenido, y después colocar la coma en su posición correspondiente.
   $\frac{1}{2^q} \cdot BIN([R \cdot 2^q],n,q) = \frac{1}{2^q} \cdot BIN(Z,n) = \frac{Z_{2,n}}{2^q} = R_{2,n,q}$

Cabe destacar que los rangos de cada representación binaria serán distintos dependiendo del número de decimales con los que se trabajen:

$$\text{Rango } BIN_{n,q} = \left[0, \frac{2^n - 1}{2^q}\right] = [0, 2^{n-q} - 2^{-q}]$$

$$\text{Rango } SM_{n,q} = \left[-\frac{2^{n-1}-1}{2^q}, \frac{2^{n-1}-1}{2^q}\right] = [-(2^{n-q-1} - 2^{-q}), 2^{n-q-1} - 2^{-q}]$$

$$\text{Rango } C2_{n,q} = \left[-\frac{2^{n-1}}{2^q}, \frac{2^{n-1}-1}{2^q}\right] = [-(2^{n-q-1}), 2^{n-q-1} - 2^{-q}]$$

$$\text{Rango } C1_{n,q} = \left[-\frac{2^{n-1}-1}{2^q}, \frac{2^{n-1}-1}{2^q}\right] = [-(2^{n-q-1} - 2^{-q}), 2^{n-q-1} - 2^{-q}]$$

$$\text{Rango } EX2^{n-1}{}_{n,q} = \left[-\frac{2^{n-1}}{2^q}, \frac{2^{n-1}-1}{2^q}\right] = [-(2^{n-q-1}), 2^{n-q-1} - 2^{-q}]$$

No obstante, la codificación de los reales en coma fija no es muy precisa, por lo que la codificación podría tener un error por defecto (menor que el valor original) o por exceso (mayor que el valor original). Para determinar el error producido debemos de comprobar cuál es el valor del número codificado y compararlo con el valor original.

$$\epsilon = VAL(R_{2,n,q}) - R \quad \begin{cases} \text{si } \epsilon = 0 & \text{no hay error} \\ \text{si } \epsilon > 0 & \text{error por exceso} \\ \text{si } \epsilon < 0 & \text{error por defecto} \end{cases}$$

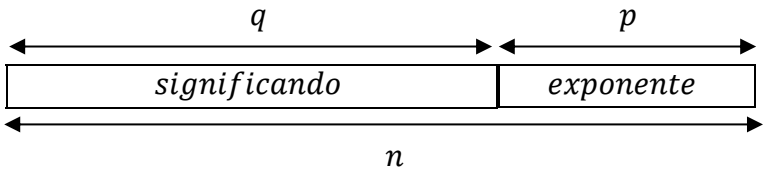
## Codificación de reales en coma flotante

La representación de reales en coma flotante nos permite codificar de manera más precisa valores muy grandes o pequeños. Esta representación es similar a la notación científica de un número, el cual está compuesto de una magnitud $M$ la cual está multiplicada por la base de representación $b$ del número real $R$ elevado a un exponente $E$.

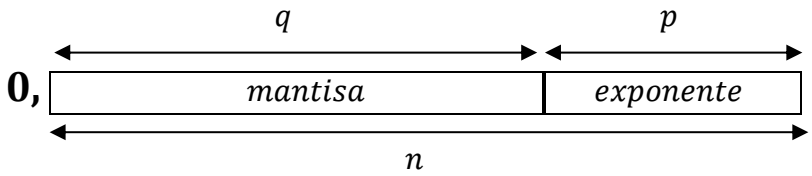
$$R = M \cdot b^E$$

Para la representación de números binarios en coma flotante, repartimos el tamaño de representación de $n$ bits en dos campos: uno expresa un número real en coma fija $M$ (a este campo se le suele llamar significando), y el otro expresa un exponente entero $E$ sobre una base 2 haciendo de factor de escala. Formalmente:

$$R = M \cdot 2^E$$

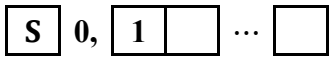



No obstante, si movemos la coma completamente a la izquierda (al peso $2^q$), representando sólo parte fraccionaria o mantisa, conseguimos codificaciones más precisas al disponer de más decimales.




Decimos que la mantisa está normalizada si a la derecha de la coma tenemos un dígito significativo. Para normalizar una mantisa en los diferentes sistemas de representación binaria debemos tener en cuenta lo siguiente:

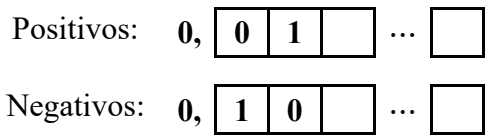
**Signo-magnitud:** La normalización consiste en tener un 1 a la derecha de la coma (sin tener en cuenta el signo, el cual viene representado por el primer bit de la mantisa).



**Complemento a 2 y Complemento a 1:** La normalización depende del valor del número. Si el número es positivo la mantisa empieza con un 0 para permanecer en el rango de los positivos pero sigue inmediatamente con el primer 1 del número.


Para los negativos es totalmente recíproco, la mantisa empieza por 1 y el siguiente bit es el primer 0 que haya después.


**Exceso a $2^{n-1}$:** La normalización es parecida que la de los sistemas complementarios, pero recordamos que los números representado en exceso el bit de signo para los positivos es 1 y para los negativos el 0 (al revés que en los sistemas complementarios).




## Precisión de la coma flotante

El rango de representación de un número de coma flotante es el siguiente:

$$\left[M_{inf} \cdot b^{E_{sup}}, M_{sup-} \cdot b^{E_{inf}}\right] \cup \left[M_{inf+} \cdot b^{E_{inf}}, M_{sup} \cdot b^{E_{sup}}\right]$$

| | |
|---|---|
| $E_{sup}$ | Valor más grande representable en el exponente |
| $E_{inf}$ | Valor más pequeño representable en el exponente |
| $M_{inf}$ | Valor normalizado más pequeño representable |
| $M_{sup-}$ | Valor normalizado más grande negativo representable |
| $M_{inf+}$ | Valor normalizado más pequeño positivo representable |
| $M_{sup}$ | Valor normalizado más grande representable |

La precisión del exponente se calcula con la fórmula: $\mathbf{2^{E-q}}$

## Estándar IEEE 754 de coma flotante

La estándar simple IEEE 754 para valores binarios de coma flotante usa 32 bits:

- El primer bit representa el signo del número.
- Los siguientes 8 bits son el exponente. Se expresa en exceso, sumándole 127.
- Los siguientes 24 bits es la mantisa del tipo 1,… en binario puro.

También se puede ver como que el significado (el valor) está expresado en signo-magnitud.