TEMA 1: REPRESENTACIÓN DE LA INFORMACIÓN

1. Sistemas de numeración y cambio de base
2. Aritmética binaria
3. Sistemas de codificación y representación numéricos
4. Representación de los números enteros
5. Representación de los números reales

TEMA 1: REPRESENTACIÓN DE LA INFORMACIÓN

Bibliografía:

- P. M. Anasagasti. Fundamentos de los Computadores.
  - Cap. 2 Representación de la Información.
- R. J. Tocci. Sistemas Digitales: Principios y Aplicaciones.
  - Cap. 2: Sistemas Numéricos y Códigos.
- T.L.Floyd. Fundamentos de Sistemas Digitales.
  - Cap. 2: Sistemas de Numeración, Operaciones y Códigos.
- C.Blanco. Fundamentos de Electrónica Digital.
  - Cap. 1: Sistemas y Códigos Numéricos.
- J.Mª Angulo y J. García. Sistemas Digitales y Tecnología de Computadores.
  - Cap. 2: Sistemas de Numeración y Códigos.
- W. Stallings. Organización y Arquitectura de Computadores.
  - Cap. 9: Aritmética del Computador. Representación en Coma Flotante.

1. Sistemas de Numeración y Cambio de Base

Un sistema de numeración en *base* ***b*** utiliza para representar los números un alfabeto compuesto por **b** símbolos, dígitos o cifras.

Ejemplos:
b = 10 (*decimal*) {0,1,2,3,4,5,6,7,8,9}
b = 8 (*octal*) {0,1,2,3,4,5,6,7}
b = 16 (*hexadecimal*) {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
b = 2 (*binario*) {0,1}

El número se expresa mediante una secuencia de cifras:

$$N \equiv \ldots n_4\ n_3\ n_2\ n_1\ n_0\ n_{-1}\ n_{-2}\ n_{-3} \ldots$$

El valor de cada cifra depende de la cifra en si y de la posición que ocupa en la secuencia. Todo número **N** sometido a un sistema *posicional* se puede descomponer como un polinomio de potencias de la base:

$$\mathbf{N} \equiv \ldots + n_3 \cdot b^3 + n_2 \cdot b^2 + n_1 \cdot b^1 + n_0 \cdot b^0 + n_{-1} \cdot b^{-1} \ldots$$

Es decir, $\mathbf{N} = \sum_i n_i b^i$

Ejemplos:

$3278{,}52_{10} = 3 \cdot 10^3 + 2 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0 + 5 \cdot 10^{-1} + 2 \cdot 10^{-2}$

$175{,}372_8 = 1 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 + 3 \cdot 8^{-1} + 7 \cdot 8^{-2} + 2 \cdot 8^{-3} = 125{,}48828125_{10}$

Conversión decimal - base b

Para obtener la representación de un número decimal entero *N* en una base *b* (con b<10) bastará con que dividamos sucesivamente *N* entre *b*. Agrupando los restos junto con el último cociente en orden inverso al obtenido, obtendremos el número buscado.

	Cociente	Resto
363 : 2	181	1
181 : 2	90	1
90 : 2	45	0
45 : 2	22	1
22 : 2	11	0
11 : 2	5	1
5 : 2	2	1
2 : 2	1	0

$$363_{10} = 1\,0\,1\,1\,0\,1\,0\,1\,1_2$$

Para obtener la representación de la parte fraccionaria bastará con que multipliquemos sucesivamente *N* por *b*. Agrupando las partes enteras en el mismo orden de su obtención obtendremos la representación buscada.

$0.67245 \times 2 = 1{,}3449 \rightarrow 1$
$0.34490 \times 2 = 0.6898 \rightarrow 0$
$0.68980 \times 2 = 1.3796 \rightarrow 1$
$0.37960 \times 2 = 0.7592 \rightarrow 0$
$0.75920 \times 2 = 1.5184 \rightarrow 1$
$0.51840 \times 2 = 1.0368 \rightarrow 1$

$$0.67245_{10} = 0.1010110_2$$

¿Cómo actuaríamos si tenemos que convertir un numero a una base mayor de 10?

Denominamos *Rango de Representación* al conjunto de valores representable en una determinada base. Con *n* cifras en la base *b* podemos formar b^n combinaciones distintas. [0..b^n-1].

El **sistema binario** es un sistema posicional. Utiliza únicamente dos símbolos: el "0" y el "1", a los que denominamos bits.

Binario	Decimal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Ejemplos:

$110100_2 = (1\cdot 2^5) + (1\cdot 2^4) + (1\cdot 2^2) = 2^5 + 2^4 + 2^2 = 32 + 16 + 4 = 52_{10}$

$0{,}10100_2 = 2^{-1} + 2^{-3} = (1/2) + (1/8) = 0{,}625_{10}$

$10100{,}001_2 = 2^4 + 2^2 + 2^{-3} = 16 + 4 + (1/8) = 20{,}125_{10}$

2. Aritmética Binaria

Las operaciones básicas que podemos realizar son las mismas que en cualquier otra base: suma, resta, multiplicación y división.

Suma binaria

0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	0 y llevo 1
1	+	1	+	1 = 1 y llevo 1

Ejemplos:

$$\begin{array}{r} 1110101 \\ +\ 1110110 \\ \hline 11101011 \end{array} \qquad \begin{array}{r} 10110110 \\ +\ 10011011 \\ \hline 101010001 \end{array} \qquad \begin{array}{r} 101101101 \\ +\ 010100011 \\ \hline 1000010000 \end{array}$$

Multiplicación binaria:

0	×	0	=	0
1	×	0	=	0
0	×	1	=	0
1	×	1	=	1

Ejemplos:

```
   1101010
        x 11
   -------
   1101010
  1101010
  --------
  100111110
```

```
   1010011
       x 101
   -------
   1010011
  1010011
  --------
  110011111
```

```
   1011011
      x 1011
   -------
   1011011
  1011011
 1011011
 ---------
 1111101001
```

Resta binaria:

0	-	0	=	0
1	-	0	=	1
1	-	1	=	0
0	-	1	=	1 y llevo 1

Ejemplos:

$$\begin{array}{r} 1101010 \\ -\ 1010111 \\ \hline 0010011 \end{array} \qquad \begin{array}{r} 10110110 \\ -\ 01111010 \\ \hline 00111100 \end{array} \qquad \begin{array}{r} 1011011000 \\ -\ 0101110011 \\ \hline 0101100101 \end{array}$$

División binaria:

0	÷ 0 =	Operación no definida
0	÷ 1 =	0
1	÷ 0 =	Operación no definida
1	÷ 1 =	1

Ejemplos:

```
 1101,011   |101
-101        |10,101
 0011 0
-  10 1
   00 111
-    101
      10
```

```
  1011011101   |1110
- 1110         |110100
  010001
-  1110
   0001111
-    1110
     000101
```

3. Sistemas de Codificación y Representación Numérica

Sistema Octal:

Puesto que necesitamos 8 símbolos diferentes se utilizan del 0 al 7. Su utilidad radica en la fácil conversión a sistema binario y viceversa.

Conversión decimal a octal.

	Cociente	Resto
2014 : 8 =	251	6
251 : 8 =	31	3
31 : 8 =	3	7

$$2014_{10} = 3736_8$$

La correspondencia con el sistema binario es inmediata. Puesto que $8 = 2^3$, una cifra en octal corresponde a 3 binarias:

$$10001101100.1101_2 = 010001101100.110100 = 2154.64_8$$

$$537.24_8 = 101011111.010100 = 101011111.0101_2$$

Sistema Hexadecimal:

Como ahora vamos a necesitar 16 símbolos diferentes utilizaremos los números del 0 al 9 junto con las 6 primeras letras, es decir de la 'A' a la 'F'.

Conversión decimal a hexadecimal.

	Cociente	Resto
4373 : 16 =	273	5
273 : 16 =	17	1
17 : 16 =	1	1

$$4373_{10} = 1115_{16}$$

La correspondencia con el sistema binario es inmediata. Puesto que $16 = 2^4$, una cifra en hexadecimal corresponde a 4 binarias:

$$1001011011111.1011101 = 0010010111011111.10111010 = 25DF.BA_{16}$$

$$\begin{matrix} 2 & 5 & D & F & B & A \end{matrix}$$

$$592.D8_{16} = 10110010010.11011_2$$

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

Código Gray:

Es un código no ponderado, continuo (los números decimales consecutivos se codifican con combinaciones adyacentes, es decir que solo difieren en un bit) y cíclico (la última combinación es adyacente a la primera). También se denomina reflejado debido a la forma de obtenerse.

2 bits	3 bits	4 bits	Decimal
0 0	0 0 0	0 0 0 0	0
0 1	0 0 1	0 0 0 1	1
1 1	0 1 1	0 0 1 1	2
1 0	0 1 0	0 0 1 0	3
	1 1 0	0 1 1 0	4
	1 1 1	0 1 1 1	5
	1 0 1	0 1 0 1	6
	1 0 0	0 1 0 0	7
		1 1 0 0	8
		1 1 0 1	9
		1 1 1 1	10
		1 1 1 0	11
		1 0 1 0	12
		1 0 1 1	13
		1 0 0 1	14
		1 0 0 0	15

Códigos BCD:

Los códigos BCD (*Binary-Coded Decimal*) codifican en binario de forma individual cada uno de los dígitos que componen un número decimal. Cada dígito decimal queda codificado con 4 binarios. El más utilizado es el BCD natural:

Decimal	BCD Natural
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Código BCD:

Las cadenas de 4 bits que no están incluidas en la columna se denominan cadenas vacías. No pertenecen a ese código y carecen de sentido

Los pesos del BCD natural coinciden con el binario natural: 8 4 2 1

Ejemplo:

$98325_{10} = 1001\ 1000\ 0011\ 0010\ 0101_{\text{BCD Natural}}$

4. Números enteros

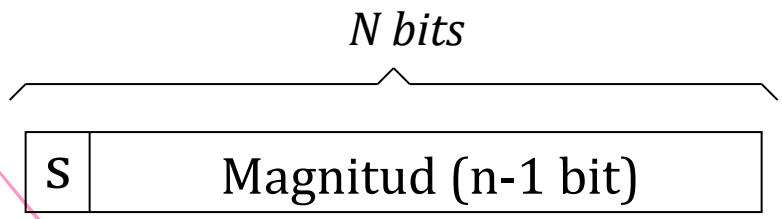
Para poder expresar números positivos y negativos, se hace necesaria la representación del signo. Existen diferentes métodos de representación de los números con signo.

Signo y Magnitud

El signo se representa en el bit más a la izquierda del dato. [Bit (n-1)], según el criterio:

0 : Número Positivo
1 : Número Negativo

En el resto de los bits se representa el valor del número en binario natural. [Bits (n-2)..0]



Signo y Magnitud

Ejemplos:

$$01101_2 = +13_{10}$$
$$11101_2 = -13_{10}$$

Rango de Representación:

$$-(2^{n-1}-1) \le x \le (2^{n-1}-1)$$

Existe duplicidad en la representación del cero.

Valor de un número cualquiera

$$\left.\begin{array}{l} \text{si } x_{n-1} = 0, \quad x = \sum_{i=0}^{n-2} x_i 2^i \\ \text{si } x_{n-1} = 1, \quad x = -\sum_{i=0}^{n-2} x_i 2^i \end{array}\right\} x = (1 - 2 \cdot x_{n-1}) \sum_{i=0}^{n-2} x_i 2^i$$

Complemento Restringido a la Base (a la base menos 1)

La representación de un número negativo (-A) la obtenemos a partir de la expresión:

$$-A \rightarrow B^n - 1 - A$$

donde *B* es la base en que estamos trabajando y *n* el número de dígitos de que disponemos

Ejemplos:

Base 10 (*Complemento a 9*)

n=2 $-26_{10} \rightarrow 10^2 - 1 - 26 = 73$ $(-26_{10} = 73_{C9})$
n=3 $-26_{10} \rightarrow 10^3 - 1 - 26 = 973$ $(-26_{10} = 973_{C9})$
n=4 $-1384_{10} \rightarrow 10^4 - 1 - 1384 = 8615$ $(-1384_{10} = 8615_{C9})$

Si particularizamos para la **Base 2**, hablaremos del **Complemento a 1.** La representación de los números negativos la obtendremos a partir de la expresión:

$$(-A)_{C1} \rightarrow 2^n - 1 - A$$

Ejemplo:
Representemos con 6 bits (N = 6) el número -18_{10}.

$$-18 \rightarrow 2^6 - 1 - 18 = 64 - 1 - 18 = 45$$
$$-18_{10} = 101101_{C1}$$

Como características citaremos:

- Los valores positivos tienen la misma representación que en SM.
- Los números positivos empiezan por cero
- Los números negativos empiezan por uno
- Presenta doble representación del 0.
- El complemento se hace y se deshace de la misma forma
- El Rango de Representación es: $[-2^{n-1} + 1, 2^{n-1} - 1]$

Ejemplo: ¿Qué número representa 101101_{C1}?

Empieza por 1 → Se trata de un número negativo:

$$-X_{10} = 101001_{C1} \quad \rightarrow \quad +X_{10} = 010110_{C1}$$

$$+X_{10} = +22_{10} \quad \rightarrow \quad 101001_{C1} = -22_{10}$$

Complemento a 1: Sumas y Restas. Ejemplos.

Realicemos la operación: $1000111_2 - 10010_2$

Si restamos en binario natural:

$$\begin{array}{r} 1000111_2 \\ -\ 0010010_2 \\ \hline 0110101_2 \end{array}$$

En Complemento a 1 (n=8) haremos:

(+1000111) + (- 10010)

$+1000111 = 01000111_{C1}$

$-\ 10010 = 11101101_{C1}$

$$\begin{array}{r} 01000111_{C1} \\ 11101101_{C1} \\ \hline (\cancel{1})00110100 \\ \qquad\qquad \rightarrow\ 1 \\ \hline 00110101_{C1} \end{array}$$

$(110010_2) - (1011101_2)$

$+ 110010_2 = 00110010_{C1}$

$- 1011100_2 = 10100011_{C1}$

$$\begin{array}{rr} & 00110010_{C1} \\ + & 10100011_{C1} \\ \hline & 11010101_{C1} \end{array}$$

$-(110010_2) - (1011101_2)$

$- 110010_2 = 11001101_{C1}$

$- 1011101_2 = 10100010_{C1}$

$$\begin{array}{rr} & 11001101_{C1} \\ + & 10100010_{C1} \\ \hline & (\cancel{1})01101111_{C1} \end{array}$$

Overflow

Complemento a la Base

La representación de un número negativo (-A) la obtenemos a partir de la expresión:

$$-A \rightarrow B^n - A$$

donde *B* es la base en que estamos trabajando y *n* el número de dígitos de que disponemos

Ejemplos:

Base 10 (*Complemento a 10*)

n=3 $-63_{10} \rightarrow 10^3 - 63 = 937$ $(-26_{10} = 937_{C10})$
n=3 $-16_{10} \rightarrow 10^3 - 16 = 984$ $(-16_{10} = 984_{C10})$
n=4 $-16_{10} \rightarrow 10^4 - 16 = 9984$ $(-16_{10} = 9984_{C10})$
n=4 $-1384_{10} \rightarrow 10^4 - 1384 = 8615$ $(-1384_{10} = 8615_{C10})$

Si particularizamos para la **Base 2**, hablaremos del **Complemento a 2.** La representación de los números negativos la obtendremos a partir de la expresión:

$$(-A)_{C2} \rightarrow 2^n - A$$

Ejemplo:
Representemos con 6 bits (N = 6) el número -21_{10}.

$$-21 \rightarrow 2^6 - 21 = 64 - 21 = 43$$
$$-21_{10} = 101011_{C2}$$

Como características citaremos:

- Los valores positivos tienen la misma representación que en SM y en C1.
- Todos los números positivos empiezan por cero
- Los números negativos empiezan por uno
- Tiene una representación única del 0.
- El complemento a 2 se hace y se deshace de la misma forma
- El Rango de Representación es: $[-2^{n-1}, 2^{n-1} - 1]$

Para obtener la representación de un número negativo en Complemento a 2 se intercambian ceros por unos y unos por ceros a la representación del número positivo y sumaremos 1.

Ejemplo: Representación del -21_{10} con 6 bits.

$$21_{10} = 10101_2 \rightarrow +21 = 010101_{C2}$$
$$-21_{10} \rightarrow 101010 + 1 = 101011_{C2}$$

Sumas y Restas en Complemento a 2. Ejemplo.

$(1011101_2) - (110010_2)$

$+1011101_2 = 01011101_{C2}$
$-\ 110010_2 = 11001110_{C2}$

El acarreo final **NO** forma parte del resultado

$$\begin{array}{rr} & 01011101_{C2} \\ + & 11001110_{C2} \\ \hline & (1)00101011_{C2} \end{array}$$

Representación sesgada

La representación se obtiene sumando un sesgo o cantidad al valor del número.
El sesgo suele ser: 2^{n-1}, en cuyo caso el rango de representación será $[-2^{n-1}, 2^{n-1} - 1]$.

Ejemplos:

$$n = 8 \Rightarrow \text{Sesgo} = 2^{8-1} = 128_{10} = 1000\ 0000_2$$

$$\begin{aligned} 11010_2 &= 10011010_S \\ -\,11010_2 &= 01100110_S \\ 0_2 &= 10000000_S \end{aligned}$$

$$n = 4 \Rightarrow \text{Sesgo} = 2^{4-1} = 8_{10} = 1000_2$$

$$\begin{aligned} 1_2 &= 1001_S \\ -1_2 &= 0111_S \end{aligned}$$

Representación en coma fija

El formato de coma fija reserva una cantidad de bits para la representación de la parte entera y otra para la parte fraccionaria. El número de bits de cada una de las partes es siempre el mismo independientemente del valor que se desee representar.

Ejemplo:
Disponemos de un formato en coma fija compuesto por 5 bits para la parte entera y 3 para la parte fraccionaria.

11,25 = 01011.010
16,5 = 10000.100
0,125 = 00000.001

También se suele incluir un bit para el signo.

Rango de Representación.

Si para la representación de *x* disponemos de 1 bit de signo, *n* bits para la parte entera y *p* para la parte fraccionaria, su rango será:

$$-[(2^n - 2^{-p}), 2^{-p}] \leq x \geq [2^{-p}, (2^n - 2^{-p})]$$

5. Números reales

Representación en coma flotante.

El formato de coma flotante utiliza el siguiente formato para representar cualquier número:

$$N = \pm M \cdot B^E$$

Donde:

N ≡ Valor numérico **M** ≡ Mantisa

B ≡ Base **E** ≡ Exponente

Ejemplo:

$$1.234535 \cdot 10^3 = 1234.535 \cdot 10^0 = 0.1234535 \cdot 10^4 = $$
$$= 123453.5 \cdot 10^{-2} = 0.0001234535 \cdot 10^7$$

Estándar IEEE754

$$N = (-1)^s M \cdot 2^E$$

Representación **S E M**

$$N = nS + nE + nM$$

Siendo *nS* la cantidad de bits para el signo, *nE* la cantidad de bits para el exponente y *nM* la cantidad de bits para la mantisa.

Estándar IEEE754 (simple precisión)

Consta de 32 bits, distribuidos de la siguiente forma:

Campo de signo : 1 bit

Campo del exponente: 8 bits con representación sesgada, siendo el sesgo:

$$S = 2^{nE-1}-1$$

Ejemplos:

Puesto que tenemos $nE = 8 \Rightarrow S = 2^{nE-1}-1 = 127 = 0111\ 1111$

(E)	E+S	(e)
0	127+0=127	0111 1111
+2	127+2=129	1000 0001
+127	127+127=254	1111 1110
-1	127-1=126	0111 1110
-126	127-126=1	0000 0001

Campo de mantisa : 23 bits, con formato normalizado: $1 \leq M < 2$
La mantisa siempre tendrá la forma M = [1.*m*] donde *m* es el valor que se almacena en el formato.

Ejemplos de normalización:

$$N1 = 1001.1100110 \cdot 2^{-5} = 1.\mathbf{0011100110} \cdot 2^{-2}$$
$$N2 = 0.000001101101 \cdot 2^{34} = 1.\mathbf{101101} \cdot 2^{28}$$

Ejercicio:
Supongamos un formato de las mismas características que el IEEE754, pero dotado solo de 16 bits, de los cuales 1 es para el signo y 8 para el exponente. Identifiquemos el número:

N = 1001 1111 0001 1101

$s = 1 \Rightarrow N<0$ $\quad$ $e = 0011\ 1110 \Rightarrow E = -65$

$m = 001\ 1101 \Rightarrow M = 1.0011101_2 = -1.2265625_{10}$

$N = -1.2265625 \cdot 2^{-65} = -3{,}32440346980633 \cdot 10^{-20}$

Situaciones especiales:

Si el en el campo del exponente encontramos e = 0, la mantisa está **desnormalizada.** El sesgo pasa a ser $2^{ne-1}-2$. Y el exponente del número será:

$$E = e - S = -2^{ne-1} + 2$$

- Si $(e = 0) \wedge (m = 0) \Rightarrow N = 0$
- Si $(e = 11...1) \wedge (m = 0) \Rightarrow N = \infty$
- Si $(e = 11...1) \wedge (m \neq 0) \Rightarrow N = NaN$

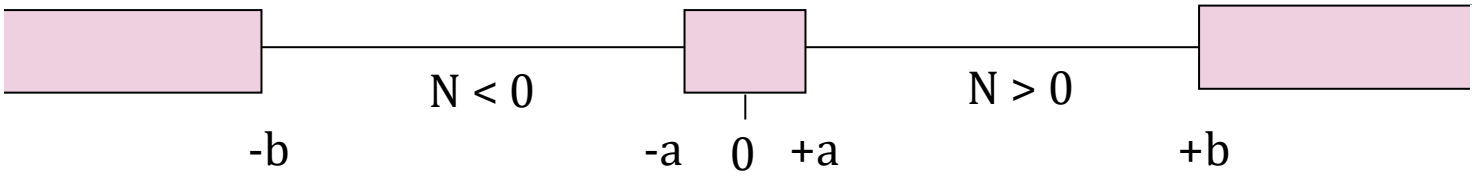
Redondeo:

El redondeo de un número lo podemos realizar por exceso, por defecto, al más cercano o al par. El formato IEEE754 emplea el redondeo al par.

Ejemplos:

Resultado	Acción	Mantisa
1.01101|10	Sumar 1	1.01110
1.01100|10	Truncar	1.01100
1.01100|11	Sumar 1	1.01101
1.01100|01	Truncar	1.01100
1.01100|00	Truncar	1.01100

Rango de Representación:



Números Normalizados:

$$|b| = M_{max}\, 2^{E\,max} \begin{cases} M_{max} = 2 - 2^{-nm} \\ E_{max} = 2^{ne-1} - 1 \end{cases} \qquad |a| = M_{min}\, 2^{E\,min} \begin{cases} M_{min} = 1 \\ E_{min} = -(2^{ne-1} - 2) \end{cases}$$

Números Desnormalizados:

$$|a'| = M'_{min}\, 2^{E'min} \begin{cases} M'_{min} = 2^{-nm} \\ E'_{min} = -(2^{ne-1} - 2) \end{cases}$$

Si particularizamos para el caso de simple precisión:

Números Normalizados:

$$\left.\begin{aligned} M_{max} &= 1.1111\cdots11 = 1+(1-2^{-23}) = 2-2^{-23} \\ E_{max} &= 11111110 = 254-127 = 127 \end{aligned}\right\} |b| = M_{max}\cdot 2^{E\,max} = (2-2^{-23})\cdot 2^{127} = 2^{128}-2^{-104}$$

$$\left.\begin{aligned} M_{min} &= 1.0000\cdots00 = 1 \\ E_{min} &= 00000001 = 1-127 = 126 \end{aligned}\right\} |a| = M_{min}\cdot 2^{E\,min} = 1\cdot 2^{-126}$$

Números Desnormalizados:

$$\left.\begin{aligned} M'_{min} &= 0.0000\cdots01 = 2^{-23} \\ E'_{min} &= 00000000 = -126 \end{aligned}\right\} |a'| = M'_{min}\cdot 2^{E'min} = 2^{-23}\cdot 2^{-126} = 2^{-149}$$

Y para doble precisión (52 bits para la mantisa y 11 para el exponente):

Números Normalizados:

$$\left.\begin{aligned} M_{max} &= 1.1111\cdots11 = 1+(1-2^{-52}) = 2-2^{-52} \\ E_{max} &= 11111110 = 2046-1023 = 1023 \end{aligned}\right\} |b| = M_{max}\cdot 2^{Emax} = (2-2^{-52})\cdot 2^{1023} = 2^{1024}-2^{-971}$$

$$\left.\begin{aligned} M_{min} &= 1.0000\cdots00 = 1 \\ E_{min} &= 00000001 = 1-1023 = 1022 \end{aligned}\right\} |a| = M_{min}\cdot 2^{Emin} = 1\cdot 2^{-1022}$$

Números Desnormalizados:

$$\left.\begin{aligned} M'_{min} &= 0.0000\cdots01 = 2^{-52} \\ E'_{min} &= 00000000 = -1022 \end{aligned}\right\} |a'| = M'_{min}\cdot 2^{E'min} = 2^{-52}\cdot 2^{-1022} = 2^{-1074}$$

- Valores límite
  - Si |N| > |b| ⇒ desbordamiento a infinito
    OVERFLOW
  - Si |N| < |a'| ⇒ desbordamiento a cero
    UNDERFLOW
- Consideraciones sobre la aritmética digital:
  - Casi siempre hay redondeo/truncamiento
  - Números excesivamente pequeños
  - Números excesivamente grandes
  - No siempre se cumple, en la práctica, la propiedad asociativa:
    $(a \cdot b) \cdot c \neq a \cdot (b \cdot c)$

Ejemplo en computador con precisión 10^{-10}:

$a=10^{7}$; $b=10^{-3}$; $c=10^{-8}$ $\quad (10^{7}\cdot 10^{-3})\cdot 10^{-8} \neq 10^{7}\cdot(10^{-3}\cdot 10^{-8})$

$$10^{4}\cdot 10^{-8} \neq 10^{7}\cdot 0$$

$$10^{-4} \neq 0$$

¡desbordamiento a 0!