Estructura de Computadores

Tema 4. Representación y Aritmética

REPRESENTACIÓN Y ARITMÉTICA

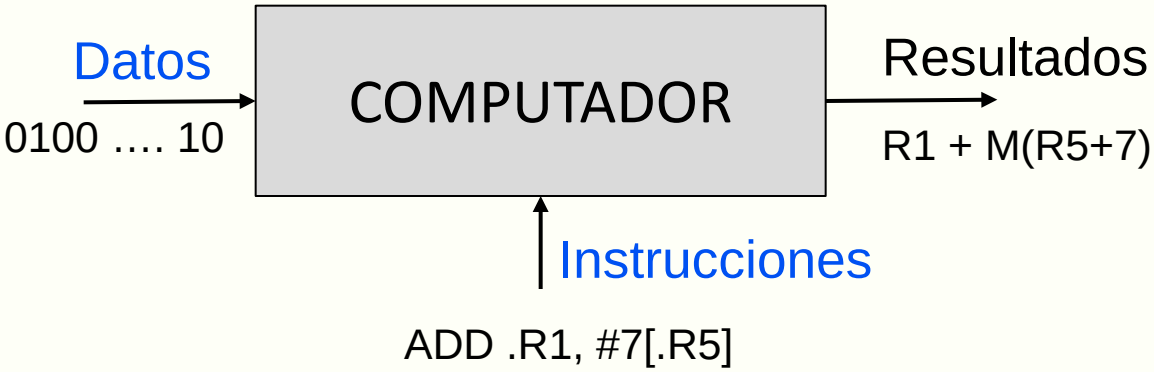
- Introducción
  - Representaciones alfanuméricas y numéricas
  - Operador y estructura de la ALU
- Representación en coma fija
  - Binario sin signo
  - Complemento a 2
  - Complemento a 1
  - Signo-magnitud
  - Exceso a M
- Representación en coma flotante
  - Definición, rango y resolución
  - Normalización y bit implícito
  - Suma y resta
  - Redondeo y bits de guarda
  - Estándar IEEE 754

BIBLIOGRAFÍA

- Fundamentos de los computadores. Pedro de Miguel. Editorial Paraninfo, 9ª edición, 2004.
- Estructura y diseño de computadores. Patterson-Hennessy. Editorial Reverté, 2000
- Organización y arquitectura de computadores. Stallings. Prentice Hall, 7ª edición, 2006
- Computer Arithmetic Systems. Omondi. Prentice Hall International, 1994
- Estructura de computadores: Problemas resueltos. García Clemente y otros. RAMA, 2006

REPRESENTACIÓN DE LA INFORMACIÓN (1)

INFORMACIÓN QUE LLEGA AL COMPUTADOR



REPRESENTACIÓN DE LA INFORMACIÓN (2)

- CONDICIONANTES DEL COMPUTADOR
  - Circuitos integrados del Computador:

    Utilización del **Sistema Binario**
  - El computador es Finito:

    Las representaciones son **Acotadas**
  - Diseño de sus unidades funcionales:

    Existen **Tamaños Privilegiados** (byte, palabra, ..)

- MODOS DE REPRESENTACIÓN
  - **Representaciones Alfanuméricas**
  - **Representaciones Numéricas**
  - Representaciones Gráficas
  - Representaciones Redundantes
  - Representaciones Etiquetadas

REPRESENTACIONES ALFANUMÉRICAS (1)

- REPRESENTAN:
  - Las 26 letras del alfabeto (Mayúsculas y minúsculas)
  - Los 10 dígitos decimales
  - Un conjunto de caracteres especiales (+ , - = < ...)
  - Un conjunto de caracteres de control (no visibles)

- CARACTERÍSTICAS:
  - Facilidad para comprobar un carácter numérico
    - ASCII: desde H'30 hasta H'39
  - Fácil equivalencia Mayúsculas y minúsculas
    - ASCII: desde H'41 (A) hasta H'5A (Z)
    - ASCII: desde H'61 (a) hasta H'7A (z)
  - Fácil comprobación si es carácter de control
    - ASCII: desde H'00 (NUL) hasta H'1F (US)
    - ASCII: excepción H'7F (DEL)

TABLA DE CÓDIGOS ASCII

Dígito hexadecimal más significativo

HEX	0	1	2	3	4	5	6	7
0	NUL	DLE	Space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	Bell	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Dígito hexadecimal menos significativo

REPRESENTACIONES ALFANUMÉRICAS (2)

- ASCII 8 bits:

  Norma ISO que añade la representación de caracteres no presentes en inglés, p.e.

  á es el 0xE1 **Á** es el 0xC1

  é es el 0xE9 **É** es el 0xC9.

- UTF-8:

  Codificación de un carácter con varios bytes.

  - Codificación de un byte: el carácter es ASCII de 7 bits.
  - Codificación de dos bytes: el carácter pertenece a lenguas romances y otras.

    La codificación es:

    110000xx 10xxxxxx donde xxxxxxxx es el carácter ISO de 8 bits.

    Ejemplo: Conversión del carácter ISO de 8 bits **á** (0xE1) a UTF-8

    11000011 10100001 0xC3A1
  - Codificación de tres bytes: el carácter pertenece a lenguas asiáticas.
  - Codificación es de cuatro bytes: otros.

REPRESENTACIONES NUMÉRICAS

- LIMITACIONES
  - Número finito de valores representables:

    RANGO DE REPRESENTACIÓN:

    Intervalo entre el mayor y el menor número representables
  - Número finito de bits para la representación:

    RESOLUCIÓN:

    Diferencia entre dos valores representables consecutivos
  - Operaciones con resultados no representables:

    DESBORDAMIENTO:

    Cuando un resultado está fuera del rango de representación

- SISTEMAS POSICIONALES CON BASE

  b = base = nº natural > 1

  Rep(X) = (... x_2 x_1 x_0 x_{-1} x_{-2} ...) con xi $\in$ {b-1, b-2, ... , 1, 0}

$$V(X) = \sum_{i=-\infty}^{i=\infty} x_i b^i = \sum_{i=0}^{i=\infty} x_i b^i + \sum_{i=1}^{i=\infty} x_{-i} b^{-i}$$

SISTEMA POSICIONAL: CAMBIO DE BASE (1)

- Conversión de base 10 a cualquier base *b*
  - La parte entera se convierte mediante divisiones sucesivas por *b* y
  - la parte fraccionaria mediante multiplicaciones sucesivas por *b*
- Conversión de base *b* a base 10

  Evaluando directamente la expresión Σ

  Ejs: $010101,1010_{(2}$ = ??

  $A27,8C_{(16}$ = ??
- Conversión de base *b1* a base *b2* = $(b1)^k$

  Cada k dígitos de la representación en *b1* constituyen un dígito en la representación en base *b2 (ej: b1 = 2 y b2 = 16 o b2 = 8)*

  Ej: $010101,1010_{(2}$ = $??_{(16}$ = $??_{(8}$
- Un número puede tener una representación exacta en una base y no tenerla en otra

  Ej: $13,2_{(10}$ ¿Representación en base 2?

SISTEMA POSICIONAL: CAMBIO DE BASE (1)

- Conversión de base 10 a cualquier base *b*
  - La parte entera se convierte mediante divisiones sucesivas por *b* y
  - la parte fraccionaria mediante multiplicaciones sucesivas por *b*
- Conversión de base *b* a base 10

  Evaluando directamente la expresión Σ

  Ejs: $010101{,}1010_{(2} = 2^4 + 2^2 + 2^0 + 2^{-1} + 2^{-3} = 21{,}625_{(10}$

  $A27{,}8C_{(16} = 10\times16^2+2\times16^1+7\times16^0+8\times16^{-1}+12\times16^{-2} = 2599{,}546875_{(10}$
- Conversión de base *b1* a base *b2* = $(b1)^k$

  Cada k dígitos de la representación en *b1* constituyen un dígito en la representación en base *b2 (ej: b1 = 2 y b2 = 16 o b2 = 8)*

  Ej: $010101{,}1010_{(2} = 15{,}A_{(16} = 25{,}5_{(8}$
- Un número puede tener una representación exacta en una base y no tenerla en otra

  Ej: $13{,}2_{(10} = {,}00110011\ldots0011_{(2}$

CAMBIO DE BASE (2)

Ejercicio: Expresar N = $2202{,}735_{(10}$ en base 16, 2 y 8

CAMBIO DE BASE (2)

*Ejercicio (**Solución**):* Expresar N = $2202{,}735_{(10}$ en base 16, 2 y 8.

$2202 = 16 \times 137 + 10 \rightarrow x_0 = 10$ (A)

$137 = 16 \times 8 + 9 \rightarrow x_1 = 9$ y $x_2 = 8$

$0{,}735 \times 16 = 11{,}760 \rightarrow x_{-1} = 11$ (B)

$0{,}760 \times 16 = 12{,}160 \rightarrow x_{-2} = 12$ (C)

Seguir hasta obtener el número de dígitos deseado

N = 89A,BC… $_{(16}$

Expandiendo cada dígito hexadecimal en 4 bits:

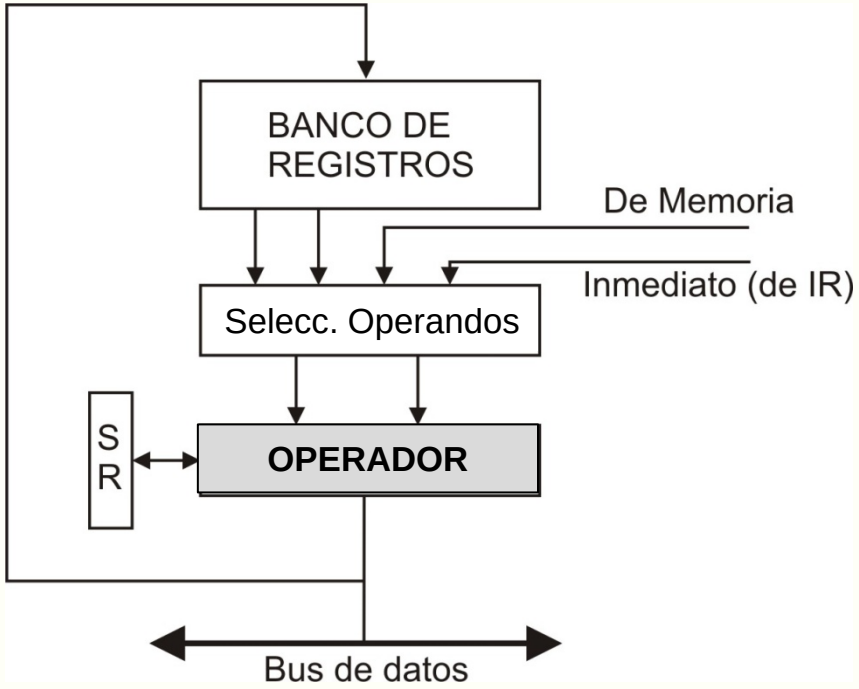
N = 1000 1001 1010, 1011 1100 … $_{(2}$

Agrupando cada 3 bits en un dígito octal:

N = 100 010 011 010, 101 111 00? $_{(2}$ = 4232,57 … $_{(8}$

OPERADOR Y ESTRUCTURA DE LA ALU

- Operador: circuito que realiza una o varias operaciones
- Estructura de la ALU (modelo de ejecución Registro-Memoria)



- Registro de estado (SR). Los flags más usuales son: Acarreo (C), Cero (Z), Signo (S), Desbordamiento (V), Paridad (P), Resta (N)

OPERACIONES BÁSICAS DE LA ALU (1)

- Operaciones lógicas (NOT, OR, AND, XOR, …)
  - Actúan sobre los operandos bit a bit:

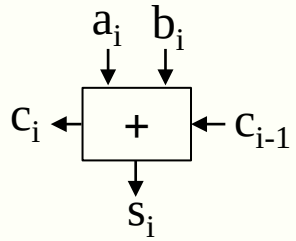
    Ej: (1001) XOR (0101) = 1100

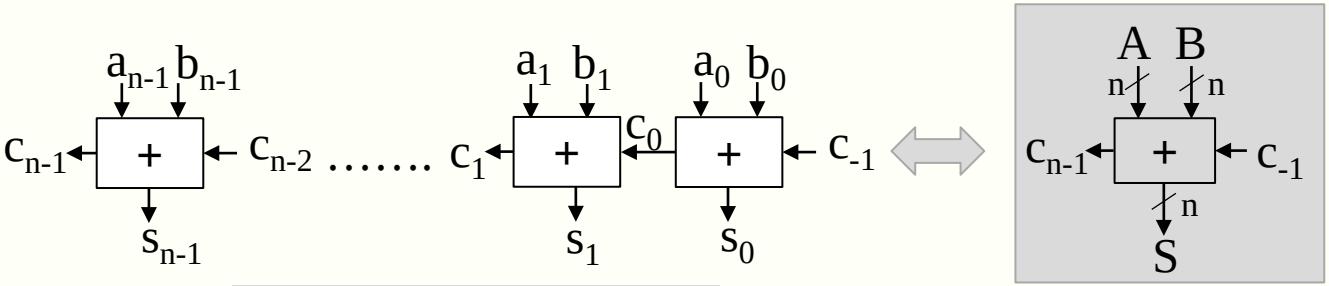
- Desplazamientos
  - **Lógicos:** se rellenan los huecos generados con ceros, ya sean a la derecha o a la izquierda
  - **Aritméticos**: se realizan sobre enteros con signo. Equivalen a multiplicar por 2 (a la izquierda) o dividir por 2 (a la derecha). Dependen del sistema de representación utilizado
  - **Concatenados**: entre registros y con biestables (acarreo)
  - **Circulares** o rotaciones
- Extensión de signo
  - Representar un dato de *n* bits en *m* bits, *m*>*n*
  - Depende del sistema de representación utilizado

OPERACIONES BÁSICAS DE LA ALU (2)

- Cambio de signo
  - Dado un número A, obtener –A
  - Depende del sistema de representación utilizado
- Suma/Resta
  - Depende del sistema de representación utilizado

Sumador elemental:





Sumador paralelo de n bits

REPRESENTACIONES NUMÉRICAS EN COMA FIJA

- Binario puro (sin signo)
- Complemento a 2
- Complemento a 1
- Signo-magnitud
- Exceso M

Para cada una de ellas, se estudiará

1) Representación y valor de un número
2) Rango y resolución
3) Cambio de signo
4) Desplazamiento aritmético
5) Extensión de signo
6) Cómo se hacen las operaciones (+ y -) y cómo se identifica el desbordamiento

BINARIO SIN SIGNO

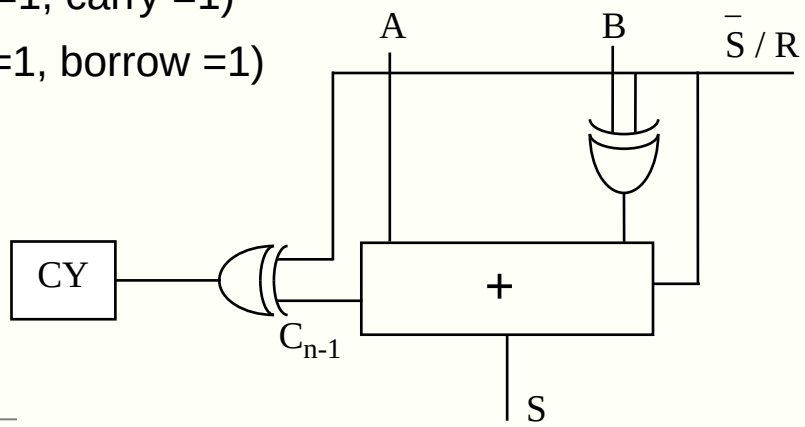
1) Rep(X) = (x_{n-1} x_{n-2} .... x_1 x_0)
2) Rango = [0, 2^n-1] Resolución = 1
3) Cambio de signo: N/A
4) Desplazamiento aritmético = desplazamiento lógico
5) Extensión de signo = añadir ceros a la izquierda
6) Operaciones + y -

$$V(X) = \sum_{i=0}^{n-1} x_i 2^i$$

A-B = A+ [(2^n-1-B)+1] -2^n = S + C_{n-1} × 2^n - 2^n

Desbordamiento (OVF) a partir de CY (biestable de acarreo)

- SUMA: C_{n-1} =1 y S/R =0 (CY=1, carry =1)
- RESTA: C_{n-1} =0 y S/R =1 (CY=1, borrow =1)



ENTEROS EN COMPLEMENTO A 2 (1)

1) Rep(X) = (x_{n-1} x_{n-2} .... x_1 x_0)
   - Si $x_{n-1} = 0 \rightarrow X \geq 0$, Igual que binario puro
   - Si $x_{n-1} = 1 \rightarrow X < 0$, Rep(X) = 2^n - |X|
   - Rep(X) + Rep(-X) = 2^n

$$V(X) = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

2) Rango = $[-2^{n-1}, -1] \cup [0, 2^{n-1}-1]$ Resolución = 1
   - Rango de representación asimétrico
   - Representación del cero única

Ejemplo: n = 6

– Representar A = $-7_{(10}$ en ca2, Calcular el valor de B = 101110

– Calcular los valores máximo y mínimo representables

ENTEROS EN COMPLEMENTO A 2 (1)

1) Rep(X) = (x_{n-1} x_{n-2} .... x_1 x_0)
   - Si x_{n-1} = 0 → X ≥ 0, Igual que binario puro
   - Si x_{n-1} = 1 → X < 0, Rep(X) = 2^n - |X|
   - Rep(X) + Rep(-X) = 2^n

$$V(X) = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

2) Rango = $[-2^{n-1}, -1] \cup [0, 2^{n-1}-1]$ Resolución = 1
   - Rango de representación asimétrico
   - Representación del cero única

Ejemplo (Solución): n = 6

– Representar A = $-7_{(10}$ en ca2, Calcular el valor de B = 101110

– Calcular los valores máximo y mínimo representables

A = 000111 -A = 1000000 – 000111 = 111001 = 111000 + 1

|B| = 1000000 – 101110 = 010010 = 18, B = -18

Valor máximo = 011111 = 2^5 -1 = 31

Valor mínimo = 100000 = -2^5 = -32

ENTEROS EN COMPLEMENTO A 2 (2)

3) Cambio de signo

   Aplicar el complemento a 2

4) Desplazamientos aritméticos

   - Izquierda (x 2): Se rellena el hueco con 0 y hay desbordamiento si cambia de signo.
   - Derecha (/ 2): Se rellena con 0 ó 1 dependiendo de si el número es positivo o negativo

5) Extensión de signo

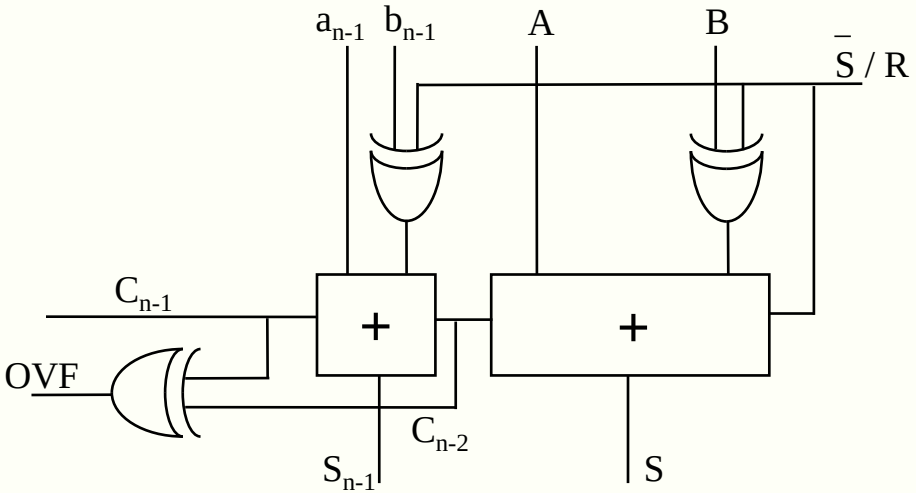
   Se rellenan con 0s ó 1s los *m-n* bits, dependiendo de si el número es positivo o negativo → con el valor de x_{n-1}

ENTEROS EN COMPLEMENTO A 2 (3)

6) Operaciones + y -
   - **Suma**

	A	B	A+B	C_{n-1}	OVF
A+ B+	**a**	**b**	**a+b**	**0**	S_{n-1}=1 C_{n-2}=1
A- B-	2^n- **a**	2^n- **b**	2^n+2^n - **(a+b)**	**1**	S_{n-1}=0 C_{n-2}=0
A+ B- (a>=b)	**a**	2^n- **b**	2^n+ **(a-b)**	**1**	**NO**
A+ B- (a<b)	**a**	2^n- **b**	2^n- **(b-a)**	**0**	**NO**

   - **Resta**: A - B = A + (-B) = A + [2^n - 1 - Rep(B)] + 1
   - Análisis de **OVF**



ENTEROS EN COMPLEMENTO A 1 (1)

1) Rep(X) = (x_{n-1} x_{n-2} .... x_1 x_0)
   - Si $x_{n-1} = 0 \rightarrow X \geq 0$, Igual que binario puro
   - Si $x_{n-1} = 1 \rightarrow X \leq 0$, Rep(X) = 2^n - 1 -|X|
   - Rep(X) + Rep(-X) = 2^n -1

$$V(X) = x_{n-1}(1-2^{n-1}) + \sum_{i=0}^{n-2} x_i 2^i$$

2) Rango = $[-(2^{n-1}-1), 0] \cup [0, 2^{n-1}-1]$ Resolución = 1
   - Rango de representación simétrico
   - Doble representación del cero: 000…000 y 111…111

Ejemplo: n = 6

– Representar A = $-7_{(10}$, Calcular el valor de B = 101110

– Calcular el valor máximo y mínimo representables

A = 000111 -A = 111111 – 000111 = 111000

|B| = 111111 – 101110 = 010001 = 17, B = -17

Valor máximo = 011111 = 2^5 -1 = 31

Valor mínimo = 100000 = -011111= -(2^5-1) = -31

ENTEROS EN COMPLEMENTO A 1 (2)

3) Cambio de signo

   Aplicar el complemento a 1

4) Desplazamientos aritméticos

   - Izquierda (x 2): Se recircula el acarreo y hay desbordamiento si el número cambia de signo.
   - Derecha (/ 2): Se rellena con 0 ó 1 dependiendo de si el número es positivo o negativo (Igual que en ca2)

5) Extensión de signo

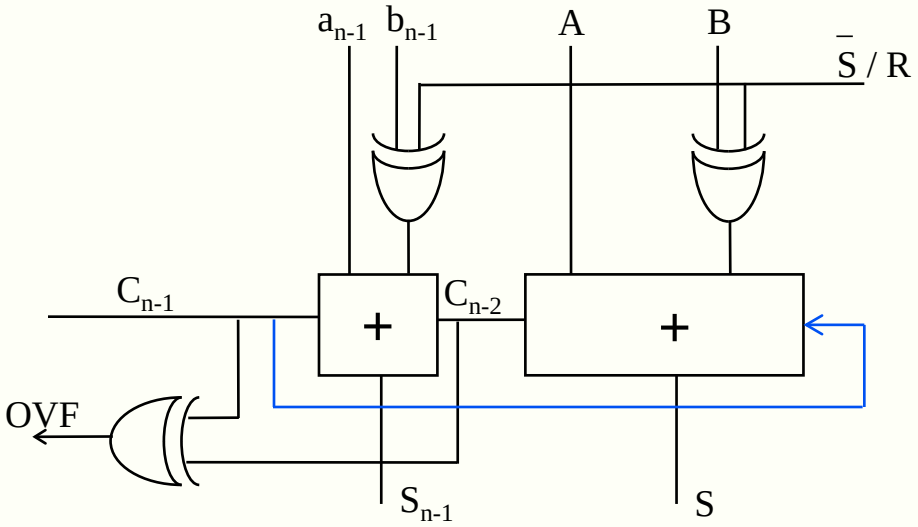
   Se rellenan con 0s ó 1s los *m-n* bits, dependiendo de si el número es positivo o negativo (Igual que en ca2)

ENTEROS EN COMPLEMENTO A 1 (3)

6) Operaciones + y -
   - **Suma**

	A	B	A+B	C_{n-1}	OVF
A+ B+	**a**	**b**	**a+b**	**0**	S_{n-1}=1 C_{n-2}=1
A- B-	**2^n-1- a**	**2^n-1- b**	2^n-1 + **2^n-1-(a+b)**	**1**	S_{n-1}=0 C_{n-2}=0
A+ B- (a>=b)	**a**	**2^n-1- b**	2^n-1 +**(a-b)**	**1**	**NO**
A+ B- (a<b)	**a**	**2^n-1- b**	**2^n-1-(b-a)**	**0**	**NO**

   - **Resta**: A - B = A + (-B) = A + [2^n - 1 - Rep(B)]
   - Análisis de **OVF**



ENTEROS EN SIGNO-MAGNITUD

1) Rep(X) = (x_{n-1} x_{n-2} .... x_1 x_0)
   - x_{n-1} = bit de signo
   - x_{n-1} = 0 → X ≥ 0 y x_{n-1} = 1 → X ≤ 0

$$V(X) = (1 - 2 \cdot x_{n-1}) \cdot \sum_{i=0}^{n-2} x_i 2^i$$

2) Rango y resolución: igual que en complemento a 1

Ejemplo: n = 6

– Representar A = $-7_{(10}$, Calcular el valor de B = 101110

– Calcular el valor máximo y mínimo representables

A = 000111 → -A = 100111 B = -14 → -B = 001110

3) Cambio de signo

4) Desplazamientos aritméticos

ENTEROS EN SIGNO-MAGNITUD (2)

5) Extensión de signo

6) Operaciones de suma y resta

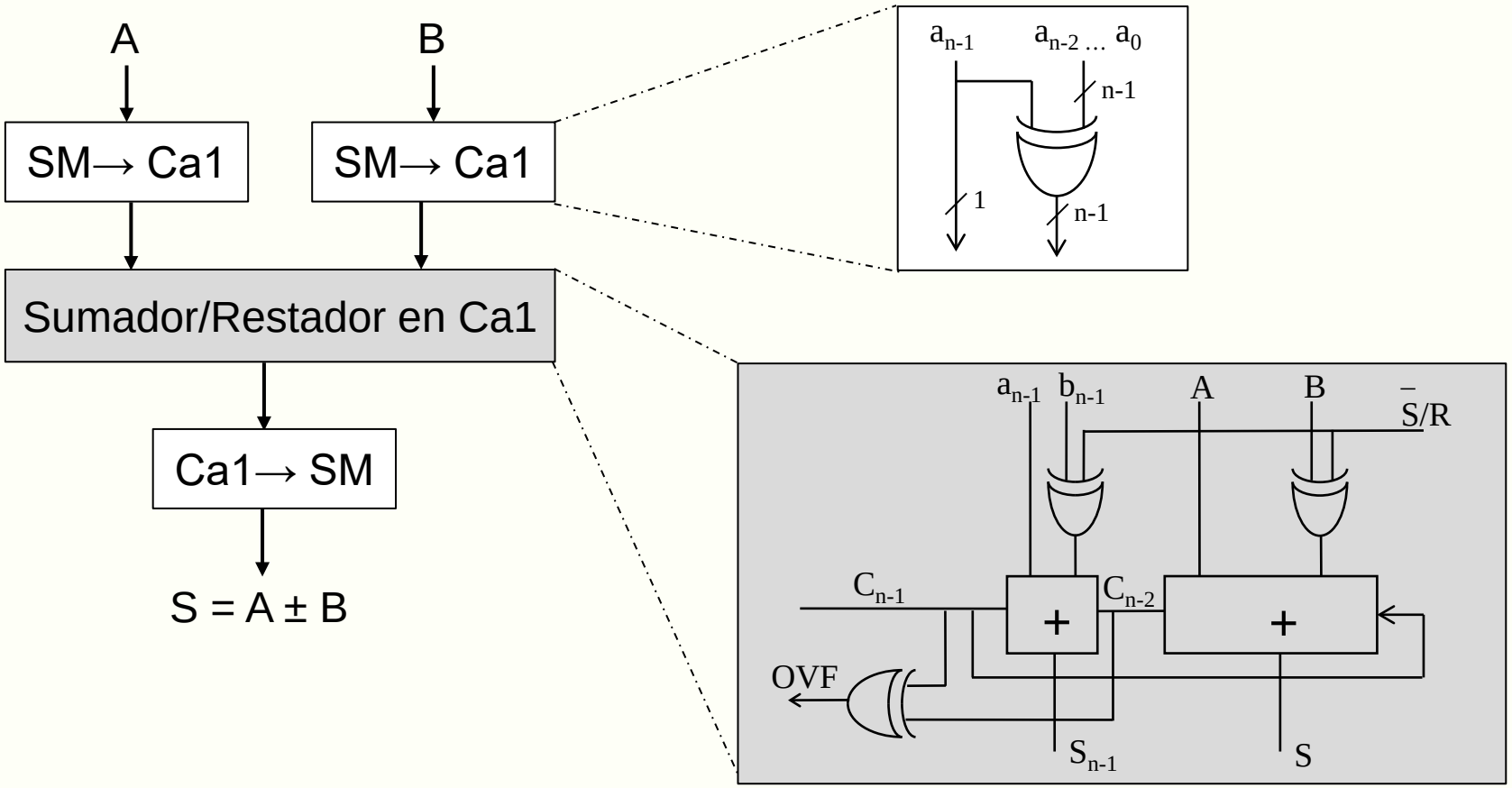
– **Suma** $A+B=(-1)^{S}\times M$, siendo $A=(-1)^{SA}\times MA$ y $B=(-1)^{SB}\times MB$, y utilizando un sumador en binario sin signo:

```
if S(A) =S(B) then
        S(R) = S(A) = S(B)
        M(R) = M(A) + M(B)
else
        if M(A) >= M(B) then
                S(R) = S(A)
                M(R) = M(A) – M(B)
        else
                S(R) = S(B)
                M(R) = M(B) – M(A)
```

– **Resta**: A - B = A + (-B)

ENTEROS EN SIGNO-MAGNITUD (3)

Operador de suma y resta utilizando un sumador/restador en Ca1



ENTEROS EN EXCESO A "M"

1) Rep(X) = (x_{n-1} x_{n-2} .... x_1 x_0)
   - Rep(X) = V(X) + M
   - Normalmente M=$\mathbf{2^{n-1}}$ o M=2^{n-1}-1 (el utilizado en el estándar IEEE)

$$V(X) = \sum_{i=0}^{n-1} x_i 2^i \text{ -M}$$

2) Rango = $[-M, -1] \cup [0, 2^n-1-M] = [-2^{n-1}, -1] \cup [0, 2^{n-1}-1]$ Resolución = 1

Ejemplo: n = 6, M = 2^{6-1} = 32

- Representar A = – $7_{(10}$, Calcular el valor de B = 101110
- Calcular el valor máximo y mínimo representables

A = – 7+32 = 25 = 011001 B = 101110 – 100000 = $14_{(10}$

Valor máximo = 111111 = 63 – 32 = 31

Valor mínimo = 000000 = 0 – 32 = – 32

3) Cambio de signo
4) Desplazamiento aritmético
5) Extensión de signo
6) Suma y resta

REPRESENTACIÓN EN COMA FLOTANTE (1)

- $V(X) = M \times r^{E}$ (notación científica)
  - M = mantisa o fracción (p bits)
  - r = base o radix
  - E = exponente (q bits)

- $Rep(X) = (e_{q-1}\ e_{q-2}\ ....\ e_1\ e_0\ m_{p-1}\ m_{p-2}\ ....\ m_1\ m_0)$

- CARACTERÍSTICAS:
  - Normalmente **r** = 2^k (r = 2, 8, 16)
  - Mantisa: coma fija con signo y base **r**
  - Exponente: Entero y base 2

REPRESENTACIÓN EN COMA FLOTANTE (2)

Ejemplo:

	Exceso 64		Signo-magnitud
S	EXPONENTE	,	MANTISA
1	7		8

- Rango Exponente = [-64, 63]
- Rango Mantisa:

  ,00000000 = 0

  ,00000001 = 2^{-8}

  .....

  ,11111111 = $1-2^{-8}$

- **Rango** = $\pm [2^{-8} \times 2^{-64}, (1-2^{-8}) \times 2^{63}] \cup 0$
- **Resolución** = $2^{-8} \times 2^{E}$
- A = H'C63C = 1 1000110 ,00111100

  $V(A) = - ,00111100 \times 2^{6} = -15_{(10}$

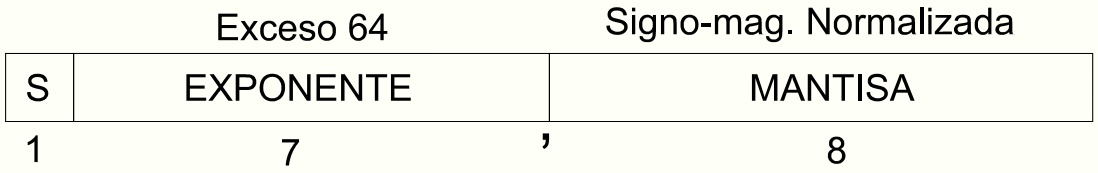
  $V(A) = - ,01111000 \times 2^{5} = -15_{(10} \rightarrow A = H' C578$

  $V(A) = - ,11110000 \times 2^{4} = -15_{(10} \rightarrow A = H' C4F0$

REPRESENTACIÓN EN COMA FLOTANTE (3)

Normalización

Un número en coma flotante está con su mantisa normalizada si al desplazar la mantisa un dígito a la izquierda y decrementar el exponente en 1 cambia el valor del número



- Rango Mantisa normalizada:

  $,10000000 = 2^{-1}$

  .....

  $,11111111 = 1-2^{-8}$

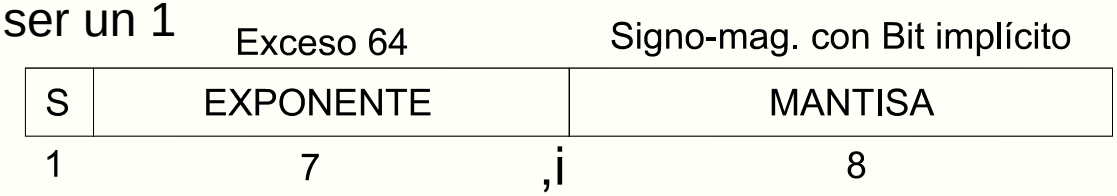
- **Rango** = $\pm [2^{-1} \times 2^{-64}, (1-2^{-8}) \times 2^{63}]$

- **Problemas de la normalización:**
  - Resultados de operaciones no normalizados
  - El cero no es representable

REPRESENTACIÓN EN COMA FLOTANTE (4)

Bit Implícito

A un número en coma flotante con r=2 y su mantisa en signo magnitud y normalizada, puede dejarse el bit más significativo como implícito ya que tiene que ser un 1



- Rango Mantisa:

  ,**1** 00000000 = 2^{-1}

  .....

  .....

  ,**1** 11111111 = $1-2^{-9}$

- **Rango** = $\pm [2^{-1} \times 2^{-64}, (1-2^{-9}) \times 2^{63}]$
- **Resolución** = $2^{-9} \times 2^{E}$

SUMA Y RESTA EN COMA FLOTANTE (1)

Solución analítica:

$A = MA \times r^{EA}$ $B = MB \times r^{EB}$

$r = 2^{k}$; Las mantisas MA y MB normalizadas

si EA ≥ EB siendo d = EA – EB

$$A \pm B = (MA \pm MB \times r^{-d}) \times r^{EA}$$

si EA < EB siendo d = EB – EA

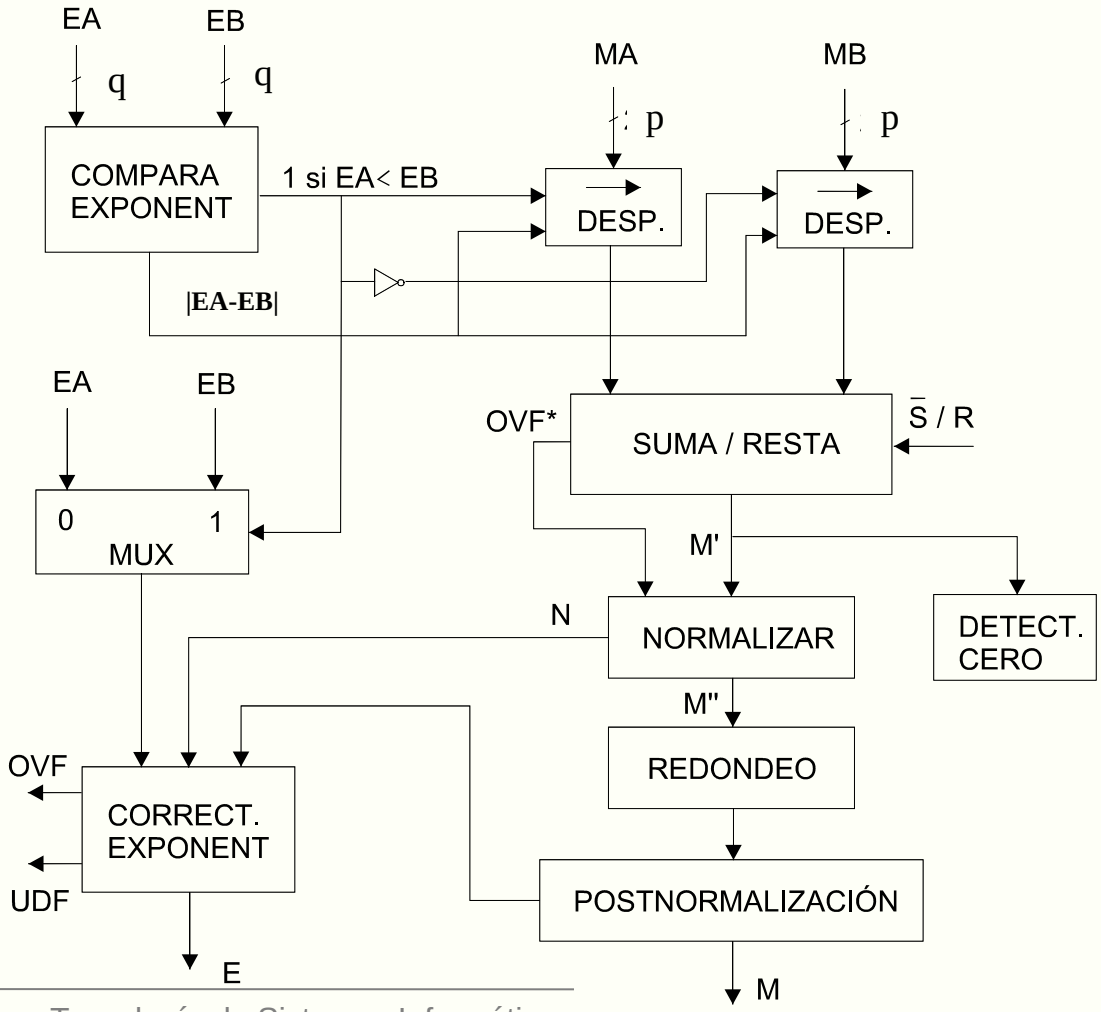
$$A \pm B = (MA \times r^{-d} \pm MB) \times r^{EB}$$

Pasos a seguir:

1. Comparar exponentes
2. Desplazar mantisa de exponente menor a la derecha
3. Sumar / Restar mantisas
4. Si el resultado es cero FIN
5. Normalizar mantisa (si redondeo postnormalizar)
6. Ajuste del exponente
7. Detectar desbordamiento

SUMA Y RESTA EN COMA FLOTANTE (2)

Esquema del sumador/restador en coma flotante:



SUMA Y RESTA EN COMA FLOTANTE (3)

1. Comparar exponentes
   - Identificar mantisa a desplazar
   - Determinar el número de desplazamientos = |EA-EB|
   - Utiliza un restador (puede haber OVF en esta resta)
2. Desplazar mantisa de exponente menor
   - Desplaza |EA-EB| dígitos
   - Desplazamientos aritméticos
3. Sumar / Restar mantisas
   - Depende de la representación de las mantisas
   - Depende del operador que se utilice
   - Puede haber OVF* → hay que normalizar
4. Detectar resultado cero
   - Se detecta con el flag, Z=1
   - Se devuelve la representación definida para el cero

SUMA Y RESTA EN COMA FLOTANTE (4)

5. Normalización (mantisa signo y p bits de magnitud)
   - OVF*=1: desplaza dcha. M' y E ← E+1
   - OVF*=0: desplaza izda. M' y E ← E-x (x=0,1,...,p-1)
   - N=1,0,-1,...,-(p-1)=cantidad a sumar al exponente mayor
   - Si redondeo y postnormalización : desplaza dcha. M' y E ← E+1
6. Corregir exponente
   - Seleccionar el exponente mayor
   - Sumar N (de la fase de normalización)
   - Sumar 1 si hay postnormalización tras el redondeo
7. Detectar desbordamiento
   - Si E > Exponente mayor, hay overflow (OVF)
   - Si E < Exponente menor, hay underflow (UDF)

REDONDEO

Ejemplo:

A= $\pm M \times 2^E$ donde M está representada por 6 bits. Se ha obtenido un resultado de 10 bits M=,100100 1011 que ha de ajustarse a 6 bits mediante **técnicas de redondeo:**

- **Truncamiento**: Suprimir los bits sobrantes. M=,100100. Error absoluto $\varepsilon_a < 2^{-6}$ siempre por defecto
- **Forzado a 1**: Truncamiento dejando siempre a 1 el bit menos significativo. M=100101. Mismo error absoluto que en truncamiento, pero por defecto y por exceso
- **Redondeo al más próximo**: Ajustar al valor M_{i-1}=,100100 ó al M_i=,100101 más próximo sumando la mitad del intervalo, $\frac{1}{2}(M_i - M_{i-1})$ = 000000 1000. M=,100101. Error absoluto $\varepsilon_a \leq 2^{-7}$ por defecto y por exceso
- **Redondeos a cero, a +∞ y a -∞** : Ajustar al extremo M_i ó M_{i-1} que corresponda en la dirección (M a 0), (M a +∞) y (M a -∞), respectivamente

DÍGITOS DE GUARDA Y BIT RETENEDOR (1)

- **Dígitos de guarda**: dígitos añadidos a la mantisa para obtener la precisión máxima. En el caso de mantisa en signo-magnitud se necesitarían dos bits de guarda, uno para normalizar el resultado y otro para redondeo

- **Bit retenedor**: bit que se añade para propagar el *borrow* en la resta. Al realizar los desplazamientos en la mantisa de menor exponente en la operación suma/resta, el bit retenedor se pone a 1 en el momento que pase un 1 y permanece ese valor independientemente de los bits que pasen después

Ejercicio: Mantisa normalizada en signo magnitud (1 bit de signo y 6 de magnitud) y exponente de 5 bits en exceso a 16.

$A = ,100001 \times 2^{7}$ $B = ,100101 \times 2^{3}$

Realizar A-B

1) Usando todos los bits necesarios para absorber todos los desplazamientos
2) Con dos bits de guarda y
3) Con los dos bits de guarda más el bit retenedor

DÍGITOS DE GUARDA Y BIT RETENEDOR (2)

SOLUCIÓN

- Resultado con 4 bits adicionales:

```
   A =              ,100001 0000 ×2^7
   B =              ,000010 0101 ×2^7
  A-B =             ,011110 1011 ×2^7
  Normalización     ,111101 011  ×2^6
  Redondeo       +           1
  A-B =             ,111101      ×2^6 = D'61
```

- Resultado con 2 bits de guarda:

```
   A =              ,100001 00 ×2^7
   B =              ,000010 01 ×2^7
  A-B =             ,011110 11 ×2^7
  Normalización     ,111101 1  ×2^6
  Redondeo       +          1
  A-B =             ,111110    ×2^6 = D'62
```

DÍGITOS DE GUARDA Y BIT RETENEDOR (3)

- Resultado con 2 bits de guarda y bit retenedor:

```
   A =              ,100001 00 0 ×2^7
   B =              ,000010 01 1 ×2^7
A-B =               ,011110 10 1 ×2^7
Normalización       ,111101 01   ×2^6
Redondeo         +           1
A-B =               ,111101      ×2^6 = D'61
```

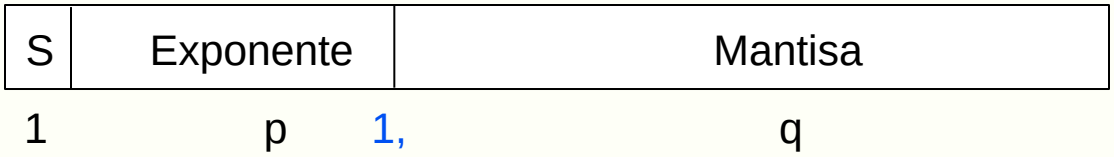
- Resultado exacto y errores:

A = ,100001 ×2^7 =D'**66** B = ,100101 ×2^3 =D'**4,625**

A-B = D'**61,375**

  - Error con 2 bits de guarda = |61,375 - 62| = 0,625
  - Error con bit retenedor = |61,375 - 61| = 0,375

ESTÁNDAR IEEE 754 DE COMA FLOTANTE (1)

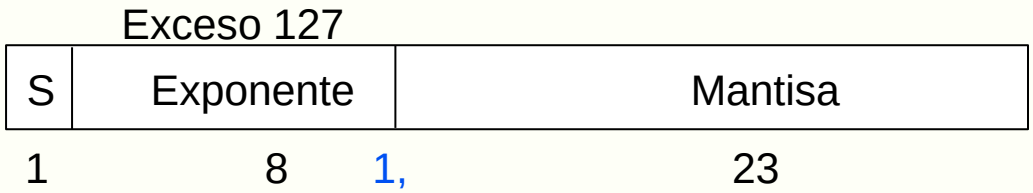


- Características generales (salvo casos especiales)
  - Exponente en exceso $2^{p-1}-1$
  - Mantisa en signo-magnitud, normalizada, con bit implícito y la coma a la derecha del bit implícito

- Diferentes formatos:
  - Simple precisión (32 bits) (p = 8, q = 23)
  - Doble precisión (64 bits) (p = 11, q = 52)
  - Cuádrupe (128 bits) (p = 15, q = 112)

ESTÁNDAR IEEE 754 DE COMA FLOTANTE (1)

- Formato en simple precisión



$$V(X) = (-1)^S \times 1,M \times 2^{E-127}$$

Combinaciones, valores de los exponentes y casos especiales:

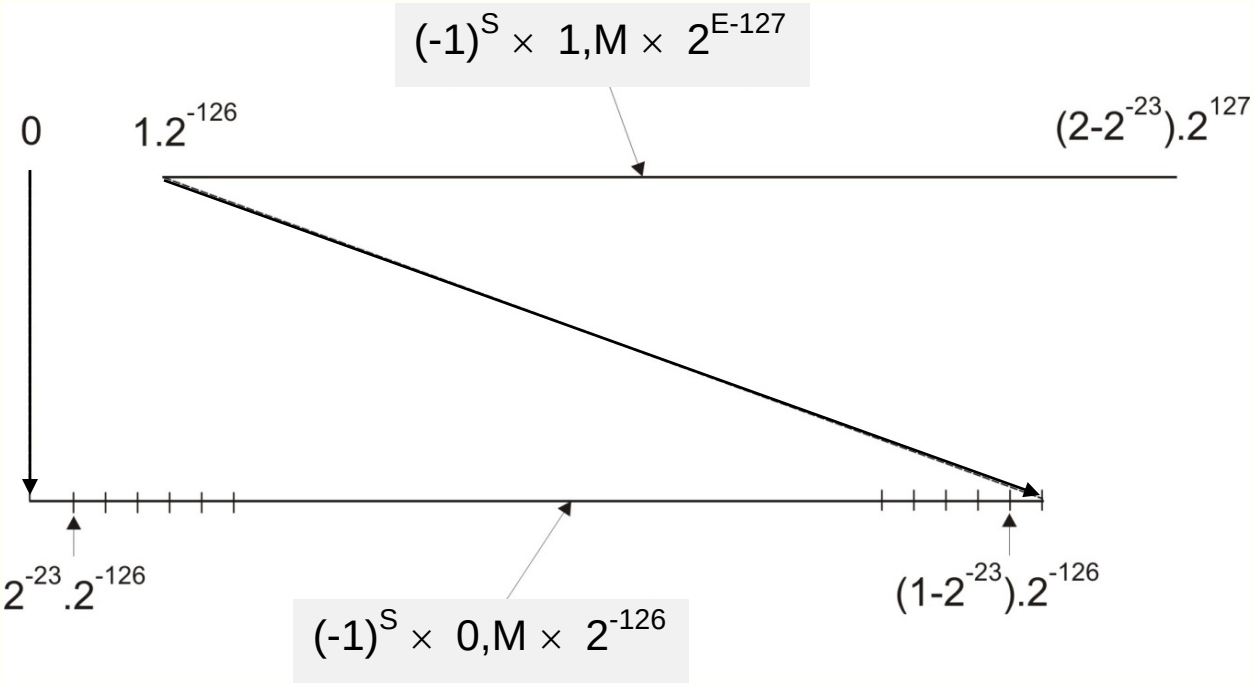
```
00000000 = -127
00000001 = -126
00000010 = -125
.............
01111111 = 0
10000000 = 1
.............
11111101 = 126
11111110 = 127
11111111 = 128
```

si M = 00 .. 00 → ± **Cero**
si M ≠ 00 .. 00 → **Número no normalizado**
$V(X) = (-1)^S \times 0,M \times 2^{-126}$

si M = 00 .. 00 → **± ∞**
si M ≠ 00 .. 00 → **Not a Number**

ESTÁNDAR IEEE 754 DE COMA FLOTANTE (2)

- Rango de representación:



- Precisión:
  - 3 Bits adicionales (2 de guarda y 1 retenedor)
  - Redondeos al más próximo, a +∞ , a –∞ y truncamiento