Arquitectura de Computadores

Aritmética Binaria y Lógica de Computadores

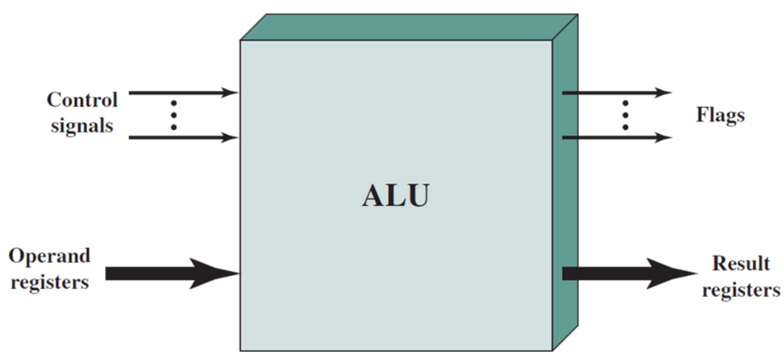
Daniel Maldonado-Ruiz

1

Aritmética Digital

2

ALU



Cada operación se realiza en la ALU, a través de los registros de memoria

Aritmética Binaria y Lógica de Computadores

2

3

Aritmética Digital

Representación de enteros

$-1101.0101_2 = -13.3125_{10}$

$00000000 =$	0
$00000001 =$	1
$00101001 =$	41
$10000000 =$	128
$11111111 =$	255

Sigue la lógica de $$A = \sum_{i=0}^{n-1} 2^i a_i$$

4

Aritmética Digital

Representación de enteros

$+18$	$= 00010010$	
-18	$= 10010010$	(sign magnitude)

$+0_{10}$	$= 00000000$	
-0_{10}	$= 10000000$	(sign magnitude)

Sigue la lógica de $$A = \begin{cases} \sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 1 \end{cases}$$

5

Aritmética Digital

Complemento a Dos

Rango	-2^{n-1} a $2^{n-1} - 1$
Número de representaciones del cero	Uno
Negación	Tomar el complemento booleano de cada bit del número positivo correspondiente, luego añadir 1 al patrón de bits resultante visto como un entero sin signo.
Expansión de la longitud del bit	Añadir posiciones de bit adicionales a la izquierda y rellene con el valor del bit de signo original.
Regla del desbordamiento	Si se suman dos números con el mismo signo (ambos positivos o ambos negativos), se produce un desbordamiento si y sólo si el resultado tiene el signo opuesto.
Regla de la resta	Para restar B a A, se toma el complemento a dos de B y se suma a A.

Sigue la lógica de:

$$A = \sum_{i=0}^{n-2} 2^i a_i \qquad \text{for } A \geq 0$$

$$A = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

6

Aritmética Digital

Complemento a Dos

Decimal Representation	Sign-Magnitude Representation	Twos Complement Representation	Biased Representation
+8	—	—	1111
+7	0111	0111	1110
+6	0110	0110	1101
+5	0101	0101	1100
+4	0100	0100	1011
+3	0011	0011	1010
+2	0010	0010	1001
+1	0001	0001	1000
+0	0000	0000	0111
−0	1000	—	—
−1	1001	1111	0110
−2	1010	1110	0101
−3	1011	1101	0100
−4	1100	1100	0011
−5	1101	1011	0010
−6	1110	1010	0001
−7	1111	1001	0000
−8	—	1000	—

7

Aritmética Digital

Complemento a Dos

−128	64	32	16	8	4	2	1

Criba de valores para un numero negativo de 8 bits

−128	64	32	16	8	4	2	1
1	0	0	0	0	0	1	1

−128 +2 +1 = −125

Conversión 10000011 a -125

−128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	0

−120 = −128 +8

Conversión 10001000 a -120

8

Aritmética Digital

Extensión de Rango

Representación de un numero de n bits como un numero de m bits, siempre que m>n

+18	=	00010010
+18	=	0000000000010010
−18	=	10010010
−18	=	1000000000010010

Extensión para magnitudes con signo

+18	=	00010010
+18	=	0000000000010010
−18	=	11101110
−32,658	=	1000000001101110

Extensión para magnitudes con complemento a 2

9

Aritmética Entera

Negación

- Toma el complemento booleano de cada bit del entero (incluido el bit de signo).
- Tratando el resultado como un entero binario sin signo, añada 1.

$$
\begin{array}{rcl}
+18 & = & 00010010 \quad \text{(twos complement)} \\
\text{bitwise complement} & = & 11101101 \\
 & & \underline{+ \qquad\quad 1} \\
 & & 11101110 = -18
\end{array}
$$

Funciona igual para ambas conversiones

10

Aritmética Entera

Suma y Resta

Desbordamiento: Si se suman dos números, y ambos son positivos o ambos negativos, entonces se produce el desbordamiento si y sólo si el resultado tiene el signo contrario.

1001 = −7 +0101 = 5 1110 = −2 (a) (−7) + (+5)	1100 = −4 +0100 = 4 10000 = 0 (b) (−4) + (+4)
0011 = 3 +0100 = 4 0111 = 7 (c) (+3) + (+4)	1100 = −4 +1111 = −1 11011 = −5 (d) (−4) + (−1)
0101 = 5 +0100 = 4 1001 = Overflow (e) (+5) + (+4)	1001 = −7 +1010 = −6 10011 = Overflow (f) (−7) + (−6)

11

Aritmética Entera

Suma y Resta

Resta: Para restar un número (sustraendo) de otro (minuendo), se toma el complemento a dos (negación) del sustraendo y se suma al minuendo.

```
     0010 =  2
    +1001 = –7
     1011 = –5

(a) M = 2 = 0010
    S = 7 = 0111
   –S =       1001
```

```
     0101 =  5
    +1110 = –2
    10011 =  3

(b) M = 5 = 0101
    S = 2 = 0010
   –S =       1110
```

```
     1011 = –5
    +1110 = –2
    11001 = –7

(c) M = –5 = 1011
    S =  2 = 0010
   –S =        1110
```

```
     0101 = 5
    +0010 = 2
     0111 = 7

(d) M =  5 = 0101
    S = –2 = 1110
   –S =        0010
```

```
     0111 = 7
    +0111 = 7
     1110 = Overflow

(e) M =  7 = 0111
    S = –7 = 1001
   –S =        0111
```

```
     1010 = –6
    +1100 = –4
    10110 = Overflow

(f) M = –6 = 1010
    S =  4 = 0100
   –S =        1100
```

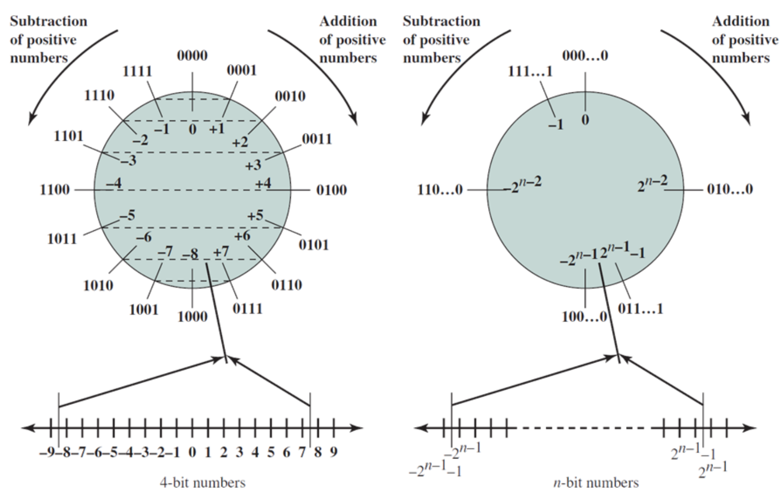
Aritmética Binaria y Lógica de Computadores

11

12

Aritmética Entera

Suma y Resta



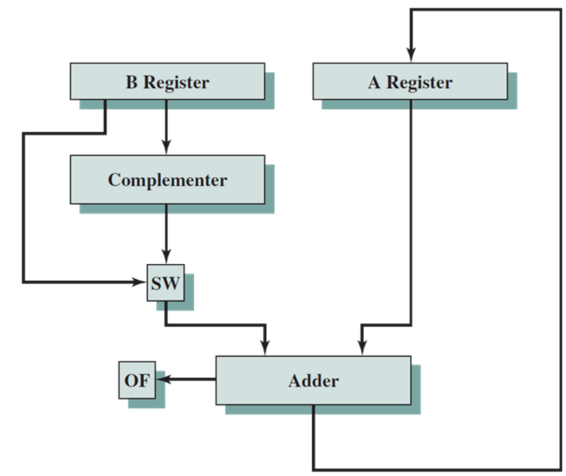
Aritmética Binaria y Lógica de Computadores

12

13

Aritmética Entera

Suma y Resta



OF = Overflow bit
SW = Switch (select addition or subtraction)

Diagrama de bloques del hardware para sumas y restas

13

14

Aritmética Entera

Multiplicación

1011	**Multiplicand (11)**
×1101	**Multiplier (13)**
1011	
0000	**Partial products**
1011	
1011	
10001111	**Product (143)**

Enteros sin signo:

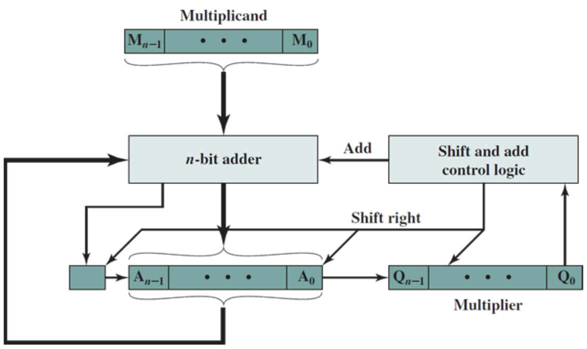
- La multiplicación implica la generación de productos parciales.
- Cuando el bit del multiplicador es 0, el producto parcial es 0. Cuando el multiplicador es 1, el producto parcial es el multiplicando.
- El producto total se obtiene sumando los productos parciales. Para esta operación, cada producto parcial sucesivo se desplaza una posición a la izquierda con respecto al producto parcial precedente.
- La multiplicación de dos enteros binarios de n bits da lugar a un producto de hasta 2n bits de longitud.

14

15

Aritmética Entera

Multiplicación



Algoritmo de multiplicación con pocos registros.

C	A	Q	M		
0	0000	1101	1011	**Initial values**	
0	1011	1101	1011	**Add**	**First cycle**
0	0101	1110	1011	**Shift**	
0	0010	1111	1011	**Shift**	**Second cycle**
0	1101	1111	1011	**Add**	**Third cycle**
0	0110	1111	1011	**Shift**	
1	0001	1111	1011	**Add**	**Fourth cycle**
0	1000	1111	1011	**Shift**	

Aritmética Binaria y Lógica de Computadores

15

16

Aritmética Entera

Multiplicación

1011	
× 1101	
00001011	$1011 \times 1 \times 2^0$
00000000	$1011 \times 0 \times 2^1$
00101100	$1011 \times 1 \times 2^2$
01011000	$1011 \times 1 \times 2^3$
10001111	

Multiplicación hecha con complemento a dos

1001	(9)
× 0011	(3)
00001001	1001×2^0
00010010	1001×2^1
00011011	(27)

Unsigned integers

1001	(−7)
× 0011	(3)
11111001	$(-7) \times 2^0 = (-7)$
11110010	$(-7) \times 2^1 = (-14)$
11101011	(−21)

Twos complement integers

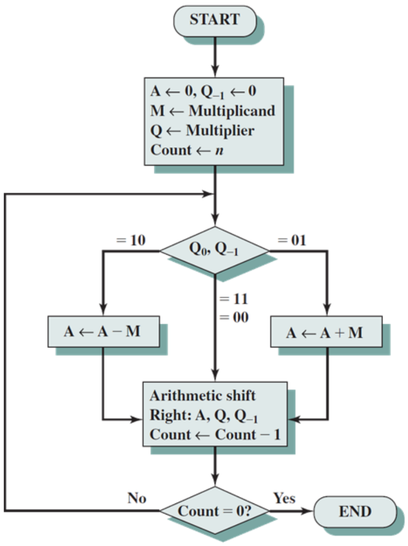
Aritmética Binaria y Lógica de Computadores

16

17

Aritmética Entera

Multiplicación



A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A←A – M	First cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second cycle
0101	0100	1	0111	A←A + M	Third cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth cycle

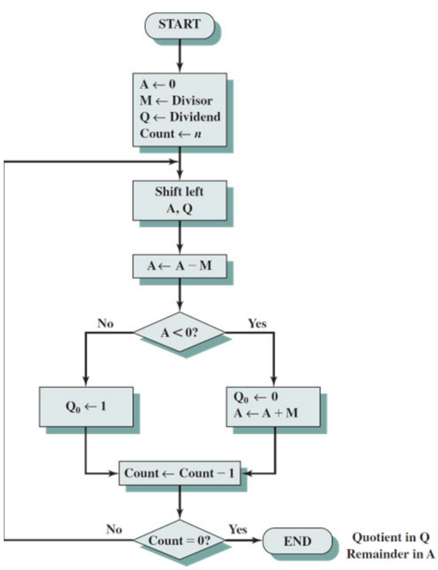
Multiplicación hecha con Algoritmo de Booth

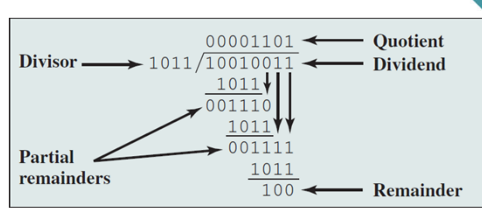
17

18

Aritmética Entera

División





División para enteros sin signo.

18

Aritmética Entera

19

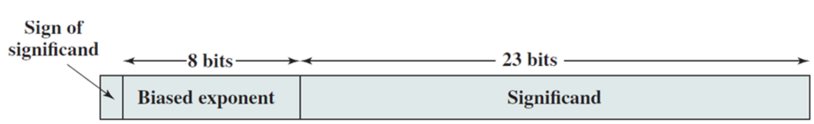
División

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set $Q_0 = 0$
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set $Q_0 = 0$
0011	1000	Shift
1101		
0000	1001	Subtract, set $Q_0 = 1$
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set $Q_0 = 0$

Algoritmo de división para enteros complemento a dos.

Aritmética de Punto Flotante

20



Format

```
 1.1010001 × 2^10100  = 0 10010011 10100010000000000000000 =  1.6328125 × 2^20
-1.1010001 × 2^10100  = 1 10010011 10100010000000000000000 = -1.6328125 × 2^20
 1.1010001 × 2^-10100 = 0 01101011 10100010000000000000000 =  1.6328125 × 2^-20
-1.1010001 × 2^-10100 = 1 01101011 10100010000000000000000 = -1.6328125 × 2^-20
```

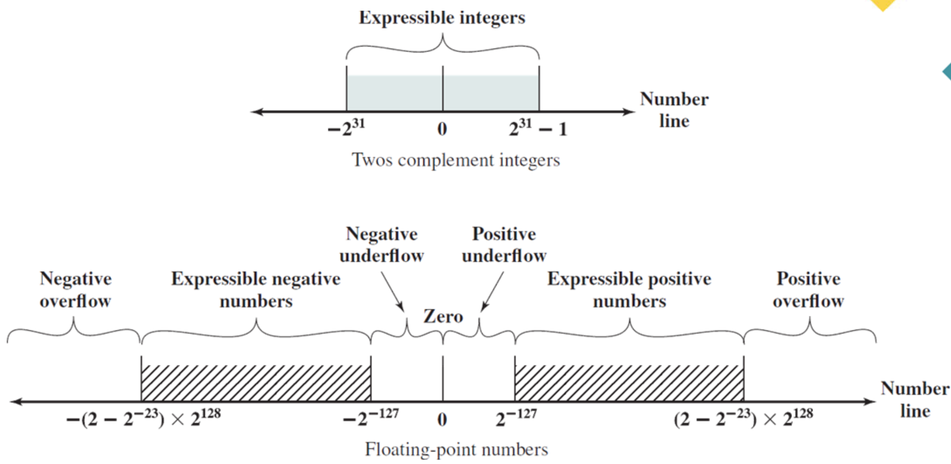
Examples

$$\pm S \times B^{\pm E}$$

Donde:
- S: Signo del numero
- B: Base
- E: Exponente

21

Aritmética de Punto Flotante

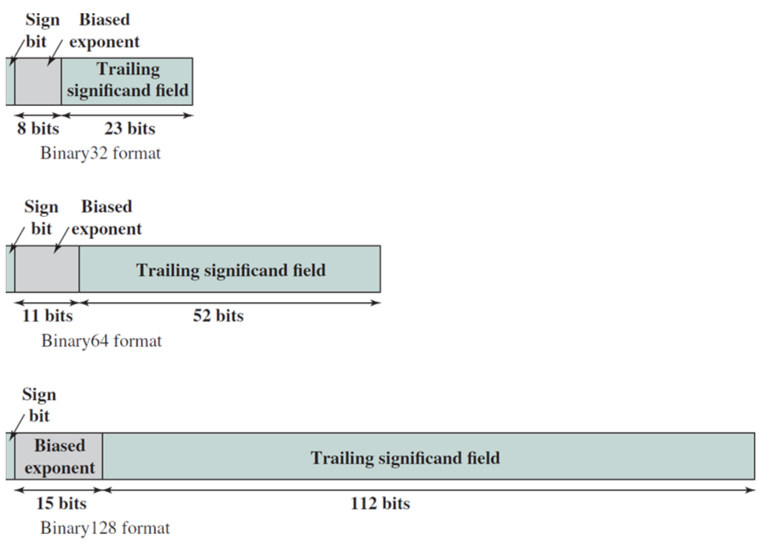


Expresión de un numero de 32 bits

22

Aritmética de Punto Flotante

Formatos IEEE 754



23

Aritmética de Punto Flotante

Formatos IEEE 754

Parameter	Format		
	Binary32	Binary64	Binary128
Storage width (bits)	32	64	128
Exponent width (bits)	8	11	15
Exponent bias	127	1023	16383
Maximum exponent	127	1023	16383
Minimum exponent	−126	−1022	−16382
Approx normal number range (base 10)	$10^{-38}, 10^{+38}$	$10^{-308}, 10^{+308}$	$10^{-4932}, 10^{+4932}$
Trailing significand width (bits)*	23	52	112
Number of exponents	254	2046	32766
Number of fractions	2^{23}	2^{52}	2^{112}
Number of values	1.98×2^{31}	1.99×2^{63}	1.99×2^{128}
Smallest positive normal number	2^{-126}	2^{-1022}	2^{-16362}
Largest positive normal number	$2^{128} - 2^{104}$	$2^{1024} - 2^{971}$	$2^{16384} - 2^{16271}$
Smallest subnormal magnitude	2^{-149}	2^{-1074}	2^{-16494}

24

Aritmética de Punto Flotante

Formatos IEEE 754

Format	Format Type		
	Arithmetic Format	Basic Format	Interchange Format
binary16			X
binary32	X	X	X
binary64	X	X	X
binary128	X	X	X
binary{k} ($k = n \times 32$ for $n > 4$)	X		X
decimal64	X	X	X
decimal128	X	X	X
decimal{k} ($k = n \times 32$ for $n > 4$)	X		X
extended precision	X		
extendable precision	X		

25

Aritmética de Punto Flotante

Formatos IEEE 754

	Sign	Biased Exponent	Fraction	Value
positive zero	0	0	0	0
negative zero	1	0	0	-0
plus infinity	0	all 1s	0	∞
minus infinity	1	all 1s	0	$-\infty$
quiet NaN	0 or 1	all 1s	$\neq 0$; first bit $= 1$	qNaN
signaling NaN	0 or 1	all 1s	$\neq 0$; first bit $= 0$	sNaN
positive normal nonzero	0	$0 < e < 225$	f	$2^{e-127}(1.f)$
negative normal nonzero	1	$0 < e < 225$	f	$-2^{e-127}(1.f)$
positive subnormal	0	0	$f \neq 0$	$2^{e-126}(0.f)$
negative subnormal	1	0	$f \neq 0$	$-2^{e-126}(0.f)$

Formato binary32

26

Aritmética de Punto Flotante

Formatos IEEE 754

	Sign	Biased Exponent	Fraction	Value
positive zero	0	0	0	0
negative zero	1	0	0	-0
plus infinity	0	all 1s	0	∞
minus infinity	1	all 1s	0	$-\infty$
quiet NaN	0 or 1	all 1s	$\neq 0$; first bit $= 1$	qNaN
signaling NaN	0 or 1	all 1s	$\neq 0$; first bit $= 0$	sNaN
positive normal nonzero	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$
negative normal nonzero	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$
positive subnormal	0	0	$f \neq 0$	$2^{e-1022}(0.f)$
negative subnormal	1	0	$f \neq 0$	$-2^{e-1022}(0.f)$

Formato binary64

27

Aritmética de Punto Flotante

Formatos IEEE 754

	Sign	Biased Exponent	Fraction	Value
positive zero	0	0	0	0
negative zero	1	0	0	-0
plus infinity	0	all 1s	0	∞
minus infinity	1	all 1s	0	$-\infty$
quiet NaN	0 or 1	all 1s	$\neq 0$; first bit $= 1$	qNaN
signaling NaN	0 or 1	all 1s	$\neq 0$; first bit $= 0$	sNaN
positive normal nonzero	0	all 1s	f	$2^{e-16383}(1.f)$
negative normal nonzero	1	all 1s	f	$-2^{e-16383}(1.f)$
positive subnormal	0	0	$f \neq 0$	$2^{e-16383}(0.f)$
negative subnormal	1	0	$f \neq 0$	$-2^{e-16383}(0.f)$

Formato binary128

28

Aritmética de Punto Flotante

Floating-Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{array}{l} X + Y = (X_S \times B^{X_E - Y_E} + Y_S) \times B^{Y_E} \\ X - Y = (X_S \times B^{X_E - Y_E} - Y_S) \times B^{Y_E} \end{array}\right\} X_E \leq Y_E$ $X \times Y = (X_S \times Y_S) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$

- **Desbordamiento de exponente:** Un exponente positivo excede el valor máximo de exponente posible. En algunos sistemas, esto puede designarse como +∞ o -∞ .
- **Subdesbordamiento de exponente:** Un exponente negativo es menor que el valor mínimo posible del exponente (por ejemplo, - 200 es menor que -127). Esto significa que el número es demasiado pequeño para ser representado, y puede ser reportado como 0.
- **Subdesbordamiento del significante:** En el proceso de alineación de significados, los dígitos pueden desbordarse por el extremo derecho del significando.
- **Desbordamiento del significante:** La suma de dos significados del mismo signo puede dar lugar a un desbordamiento del bit más significativo.

29

Aritmética de Punto Flotante

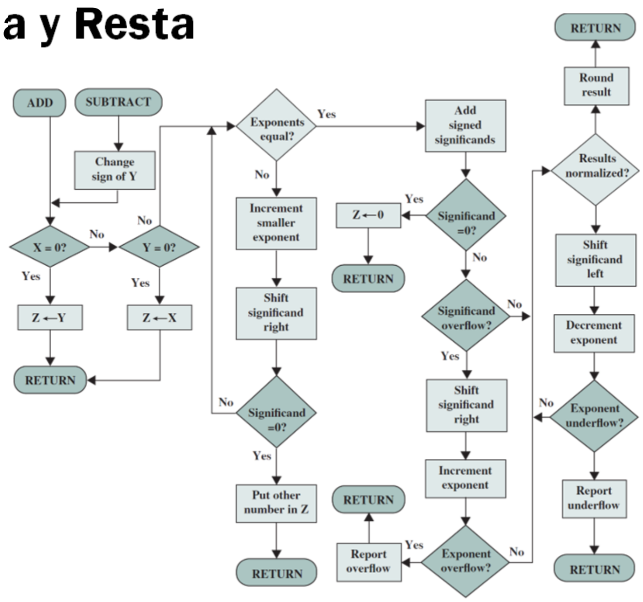
Suma y Resta

1. **Comprobación del cero:** El proceso comienza cambiando el signo del sustraendo si se trata de una operación de resta. A continuación, si alguno de los operandos es 0, se indica el otro como resultado.
2. **Alineación de significantes:** Manipular los números para que los dos exponentes sean iguales.
3. **Suma:** Suma de significantes, teniendo en cuenta sus signos. Como los signos pueden diferir, el resultado puede ser 0. También existe la posibilidad de que el significando se desborde en 1 dígito. En ese caso, el significando del resultado se desplaza a la derecha y se incrementa el exponente.
4. **Normalización:** Consiste en desplazar los dígitos del significante hacia la izquierda hasta que el dígito más significativo sea distinto de cero. Cada desplazamiento provoca una disminución del exponente y, por lo tanto, podría causar un desbordamiento por debajo del exponente.

30

Aritmética de Punto Flotante

Suma y Resta

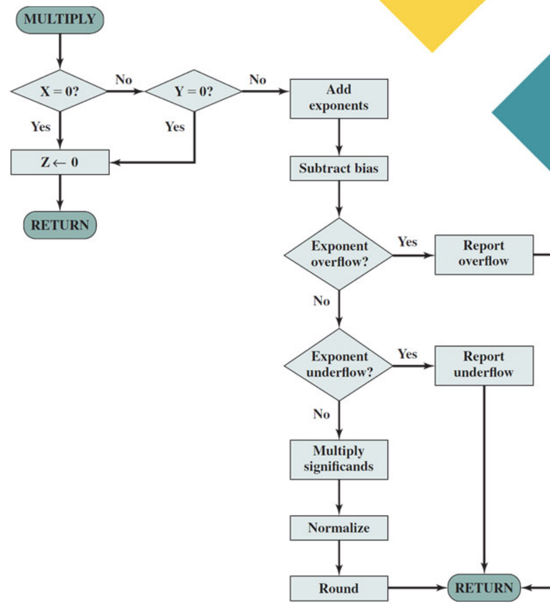


Proceso de suma/resta
de números no enteros

Aritmética de Punto Flotante

31

Multiplicación

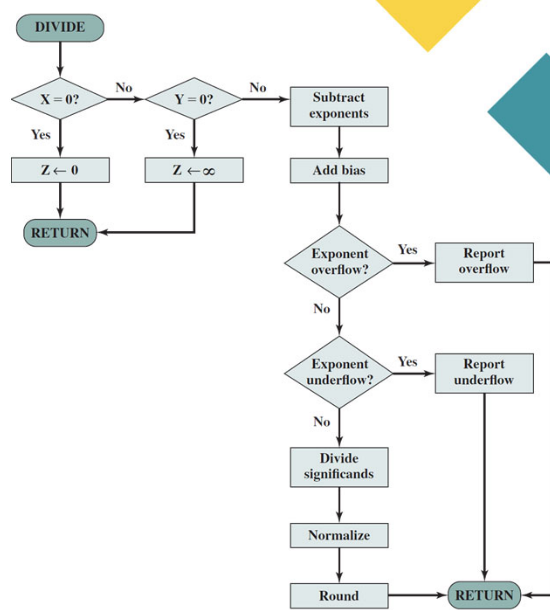


Multiplicación de números no enteros

Aritmética de Punto Flotante

32

División



División de números no enteros

Aritmética de Punto Flotante

33

Gestión del Infinito

Operation	Quiet NaN Produced By
Any	Any operation on a signaling NaN
Add or subtract	Magnitude subtraction of infinities: $(+\infty) + (-\infty)$ $(-\infty) + (+\infty)$ $(+\infty) - (+\infty)$ $(-\infty) - (-\infty)$
Multiply	$0 \times \infty$
Division	$\frac{0}{0}$ or $\frac{\infty}{\infty}$
Remainder	x REM 0 or ∞ REM y
Square root	$\sqrt{x}$, where $x < 0$

*NaN: Not a number (no es un número)

¿Preguntas?

daniel.maldonado02@epn.edu.ec