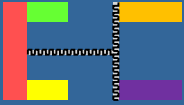


TEMA 2.

UNIDAD ARITMÉTICO-LÓGICA

dtic

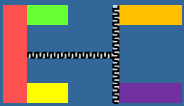




UNIDAD ARITMÉTICO-LÓGICA

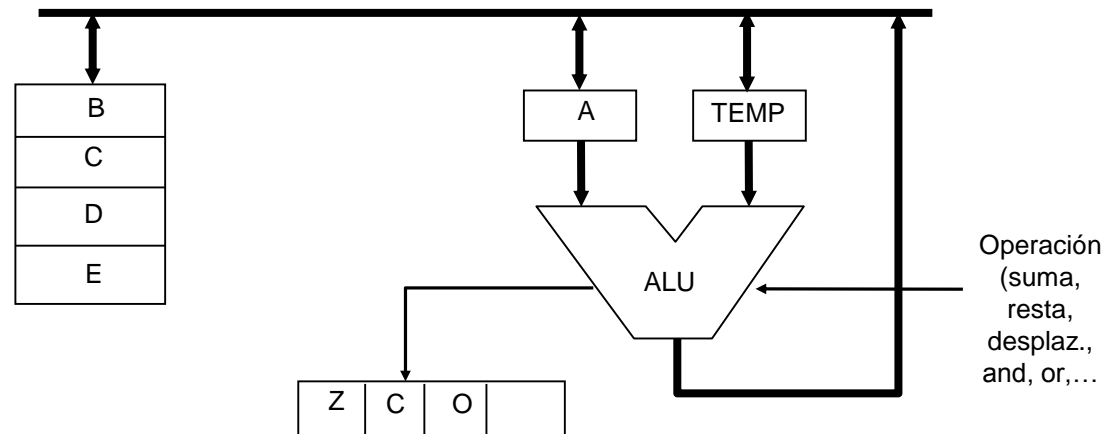
Índice

- ⊙ Introducción
- ⊙ Unidad lógica
- ⊙ Operadores de desplazamiento
- ⊙ Unidad aritmética entera
 - ⊙ Sumar y restar
 - ⊙ Multiplicar y dividir
- ⊙ Unidad aritmética flotante. IEEE 754
 - ⊙ Sumar y restar
 - ⊙ Multiplicar y dividir
 - ⊙ Técnicas de redondeo

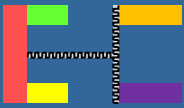


ESTRUCTURA GENERAL ALU

Introducción



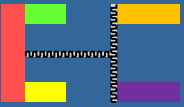
- ⊙ Operador aritmético, lógico, desplazamiento (uno o varios) (ALU)
- ⊙ El Acumulador
- ⊙ Uno o varios registros temporales
- ⊙ Indicadores de resultado
 - ⊙ Acarreo (C)
 - ⊙ Negativo (N)
 - ⊙ Desbordamiento (O)
 - ⊙ Cero (Z)



OPERACIONES TÍPICAS

Introducción

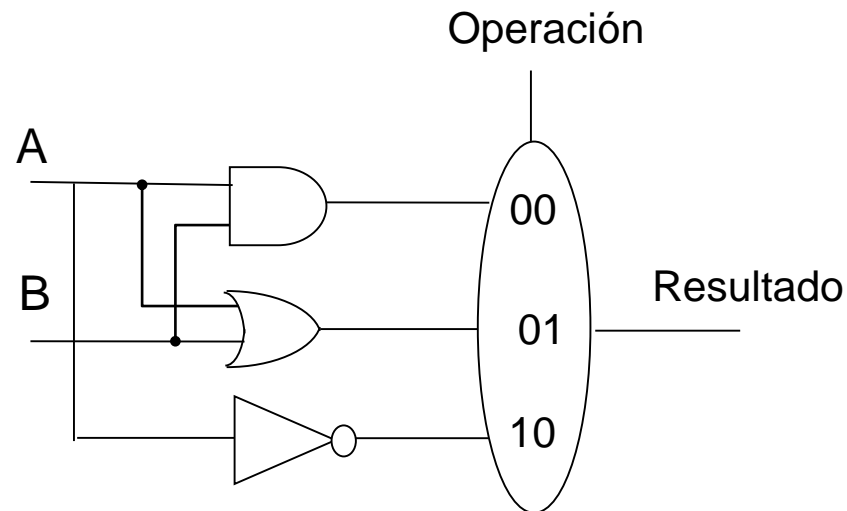
Operación	Potencia de cálculo			
	Mínima	Baja	Media	Alta
Suma/Resta en binario	Comb	Comb	Comb	Comb
Suma/Resta en coma flotante	Prg/Copr	Prg/UC	UC	Secu
Multiplicación en binario	Prg/Copr	Prg/UC	UC	Comb
Multiplicación en coma flotante	Prg/Copr	Prg/UC	UC	Secu
División en binario	Prg/Copr	Prg/UC	UC	Secu
División en coma flotante	Prg/Copr	Prg/UC	UC	Secu
Operaciones lógicas	Comb	Comb	Comb	Comb
Desplazamientos unitarios	Comb	Comb	Comb	Comb
Desplazamientos múltiples	Prg	Prg/UC	UC	Comb

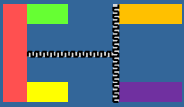


OPERACIONES LÓGICAS

Unidad
lógica

- ⊙ Fáciles de implementar \Rightarrow Correspondencia directa con Hardware.
- ⊙ Puertas lógicas AND, OR, OR-EXCLUSIVA, INVERSORES,...



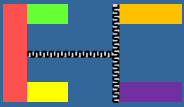


OPERACIONES DE DESPLAZAMIENTO

Operadores de desplazamiento

- ⊙ Consisten en trasladar los bits de una palabra hacia la izquierda o derecha.
- ⊙ Dependiendo de cómo se traten los extremos, se obtienen tres tipos de desplazamientos:
 - ⊙ Lógicos
 - ⊙ Circulares
 - ⊙ Aritméticos

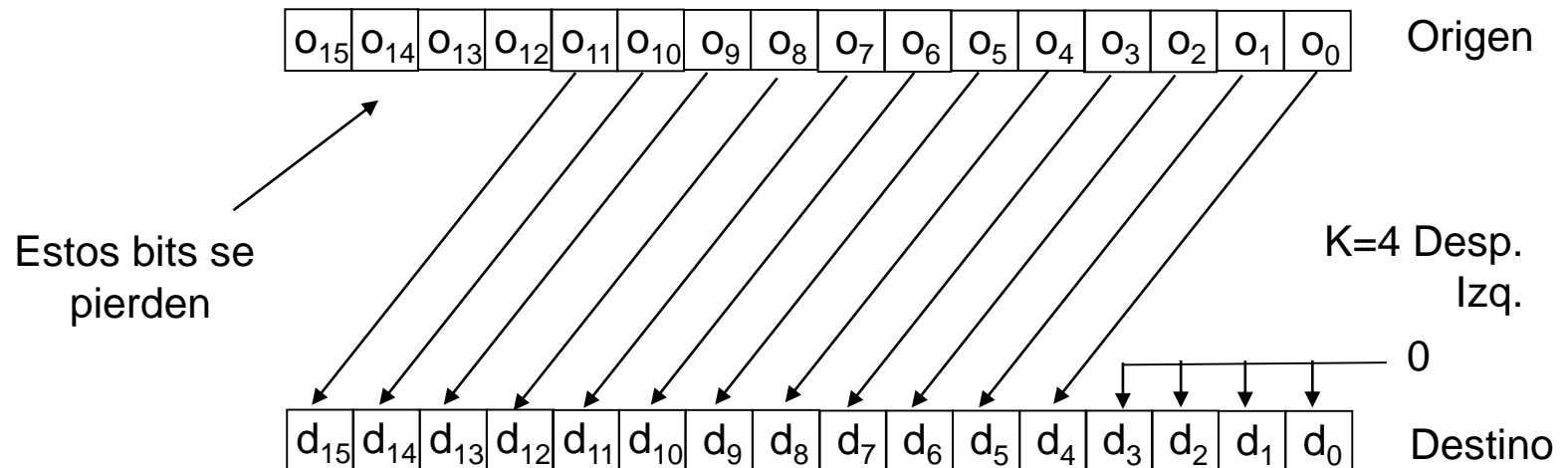




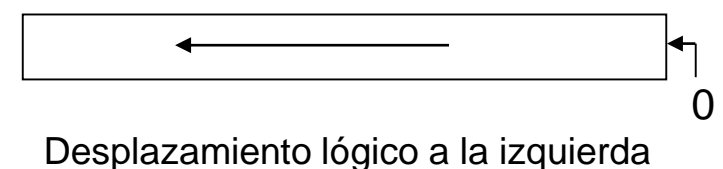
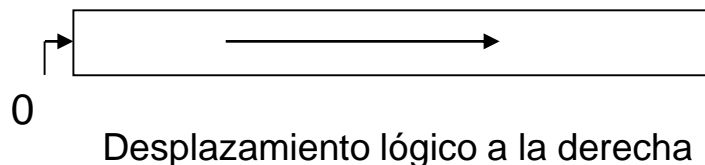
DESPLAZAMIENTOS LÓGICOS

Operadores de
desplazamiento

- Los valores extremos se completan con ceros, aunque se pueden plantear desplazamientos lógicos con inclusión de unos en lugar de ceros



Habitualmente, el origen y destino es la misma palabra.

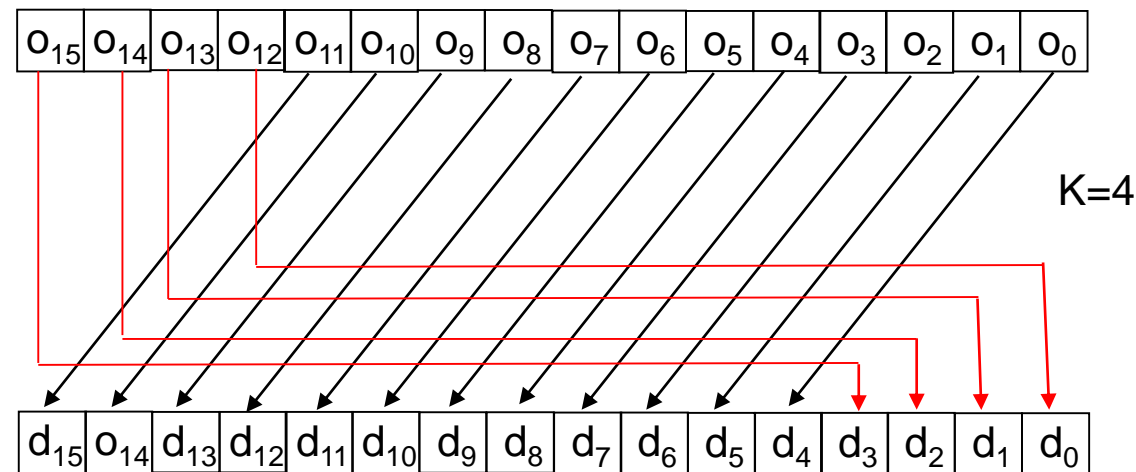




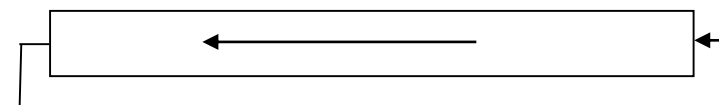
DESPLAZAMIENTOS CIRCULARES

Operadores de
desplazamiento

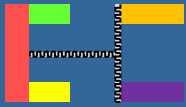
- Los bits del origen que sobran por un lado, se insertan en el destino por el otro, matemáticamente:



Desplazamiento circular a la derecha



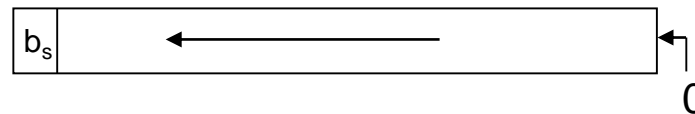
Desplazamiento circular a la izquierda



DESPLAZAMIENTOS ARITMÉTICOS

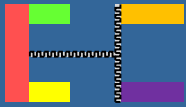
Operadores de
desplazamiento

- Se tiene en cuenta el bit de signo y se representa en complemento a 2.
- Desplazamiento a la **izquierda** (multiplicación por 2):



Desplazamiento aritmético a la izquierda

- Se van perdiendo bits de signo y hay que introducir ceros por la derecha para números positivos
- Para que no haya overflow hay que comprobar el bit de signo después de la operación:
 - Si el número es positivo, se desplaza y da negativo -> Overflow
 - Si el número es negativo, se desplaza y da positivo -> Overflow



DESPLAZAMIENTOS ARITMÉTICOS

Operadores de
desplazamiento

🎯 Ejemplos:

$O = 0000\ 1010 = 10_{10} \times 2 \rightarrow D = 0001\ 0100 = 20_{10} \rightarrow \text{correcto}$

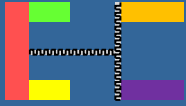
$O = 0100\ 0000 = 64_{10} \times 2 \rightarrow D = 1000\ 0000 = -128_{10} \rightarrow \text{incorrecto}$

$O = 1111\ 1100 = -4_{10} \times 2 \rightarrow D = 1111\ 1000 = -8_{10} \rightarrow \text{correcto}$

$O = 1000\ 0100 = -124_{10} \times 2 \rightarrow D = 0000\ 1000 = 8_{10} \rightarrow \text{incorrecto}$

Negativos en
complemento a 2:
 $1 \leftrightarrow 0$ y $+1$

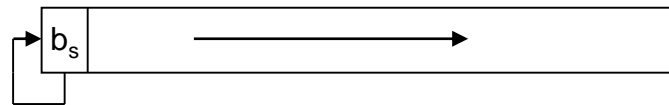
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1



DESPLAZAMIENTOS ARITMÉTICOS

Operadores de
desplazamiento

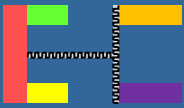
- ⊙ Desplazamiento a la derecha (división por 2):
 - ⊙ Hay que conservar el signo del dato, hay que recircularlo.
 - ⊙ Hay que introducir ceros por la izquierda para números positivos
 - ⊙ Para números negativos hay que introducir unos



Desplazamiento aritmético a la derecha

$$O = 0000\ 1010 = 10_{10} / 2 \rightarrow D = 0000\ 0101 = 5_{10}$$

$$O = 1111\ 1100 = -4_{10} / 2 \rightarrow D = 1111\ 1110 = -2_{10}$$



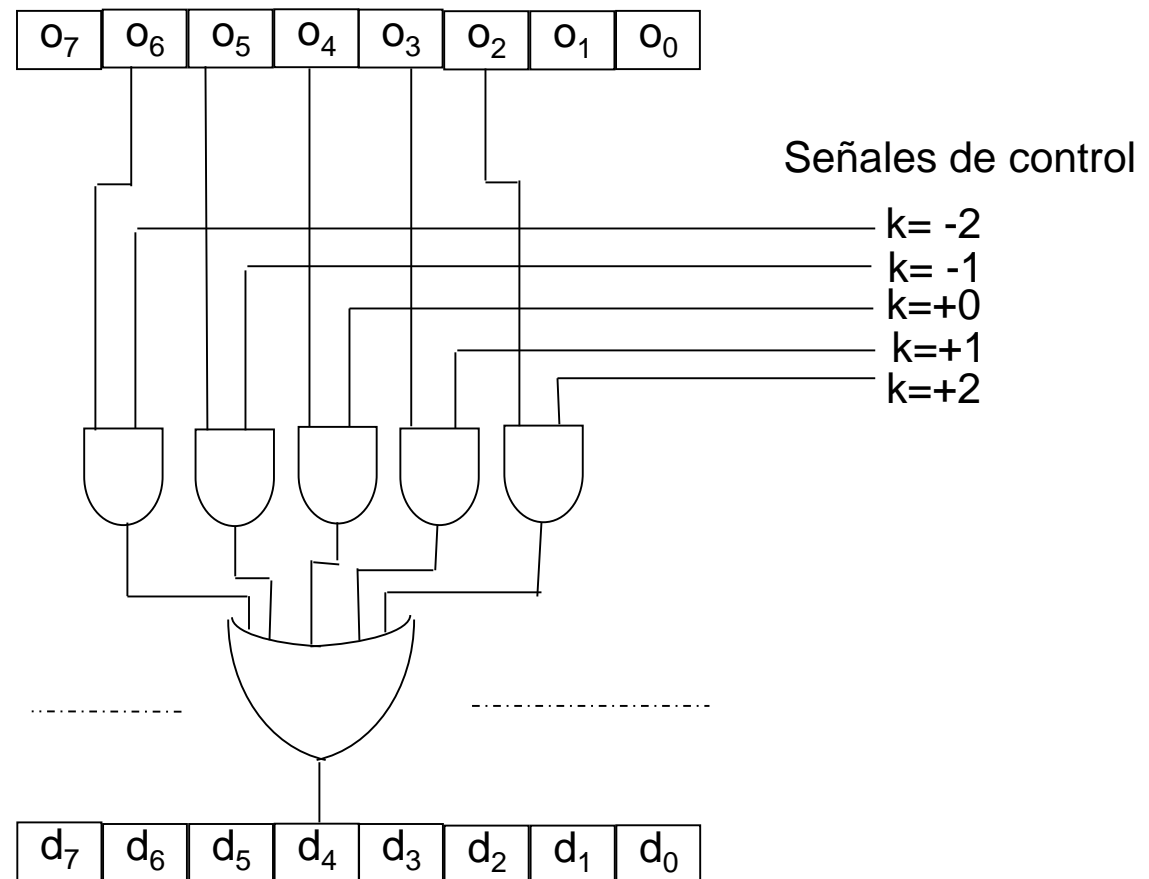
IMPLEMENTACIÓN OPERACIONES DE DESPLAZAMIENTO

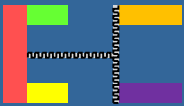
Operadores de desplazamiento

🎯 Puertas lógicas

- 🎯 La complejidad es elevada.
- 🎯 Las señales de control son las mismas para cada bit.

Unidad de desplazamiento de 2 bits para el bit 4

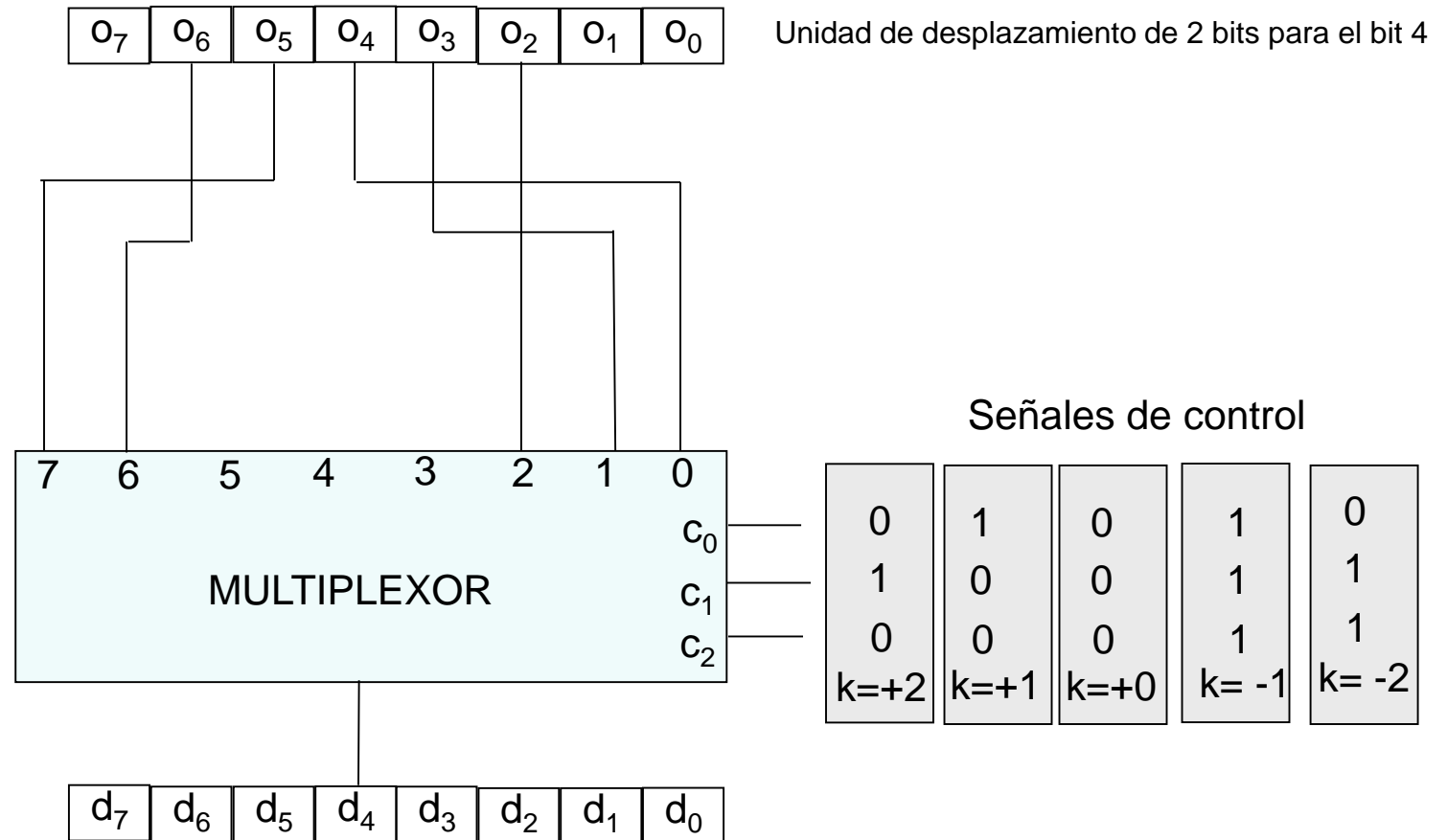


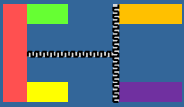


IMPLEMENTACIÓN OPERACIONES DE DESPLAZAMIENTO

Operadores de desplazamiento

🎯 Multiplexores





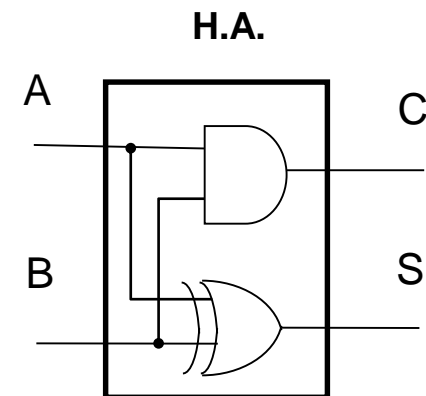
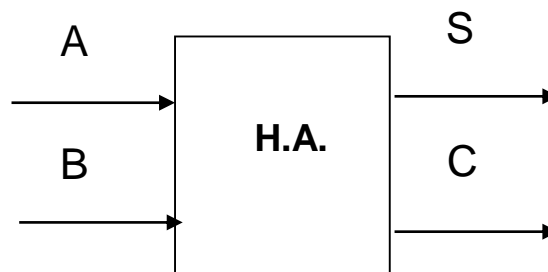
LA SUMA Y LA RESTA

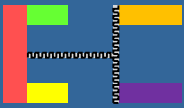
Unidad
aritmética
entera.
Sumar y
restar

⊙ Semisumador Binario (H.A.)

Entradas		Salidas	
X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{X} \cdot Y + X \cdot \overline{Y} = X \oplus Y$$
$$C = X \cdot Y$$

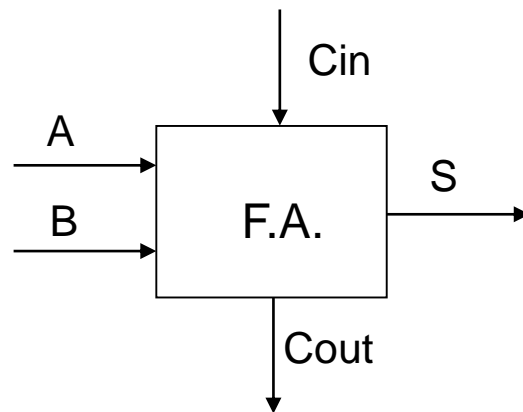




SUMADOR COMPLETO (F.A.)

Unidad
aritmética
entera.
Sumar y
restar

⊙ Semisumador Binario (H.A.)



Entradas			Salidas	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \bar{A} \cdot \bar{B} \cdot Cin + \bar{A} \cdot B \cdot \bar{Cin} + A \cdot \bar{B} \cdot \bar{Cin} + A \cdot B \cdot Cin$$

$$Cout = A \cdot B + A \cdot Cin + B \cdot Cin$$

$$S = (A \oplus B) \oplus Cin$$

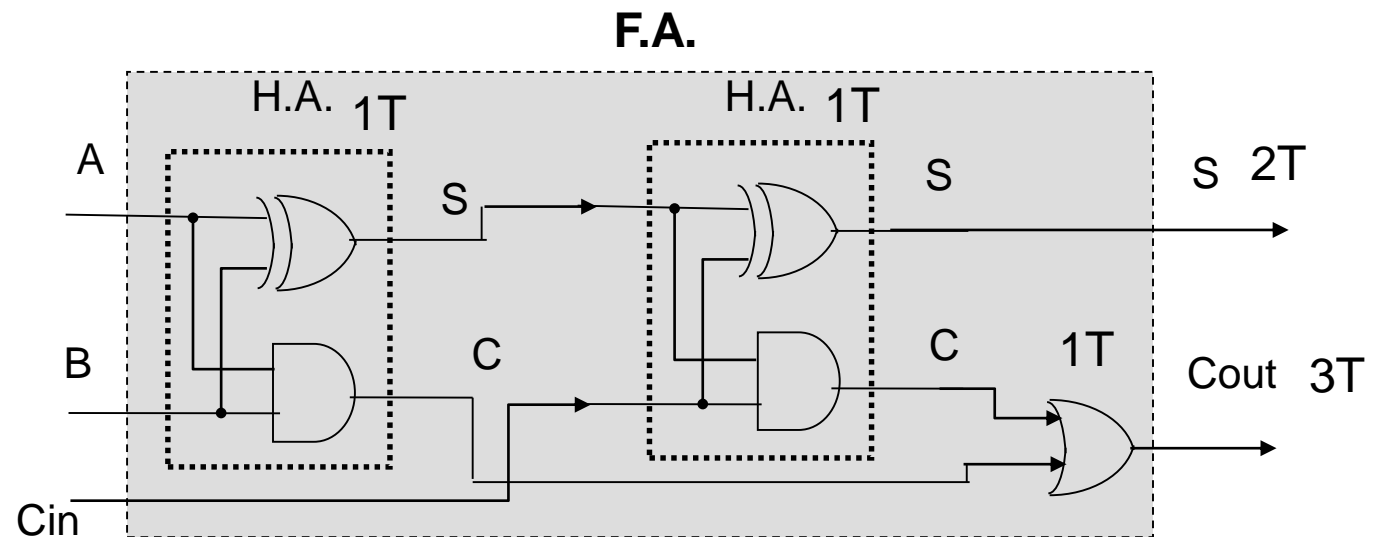
$$Cout = AB + Cin(A \oplus B)$$

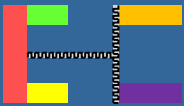


SUMADOR COMPLETO (F.A.)

Unidad
aritmética
entera.
Sumar y
restar

Sumador completo (FA) - Semisumadores

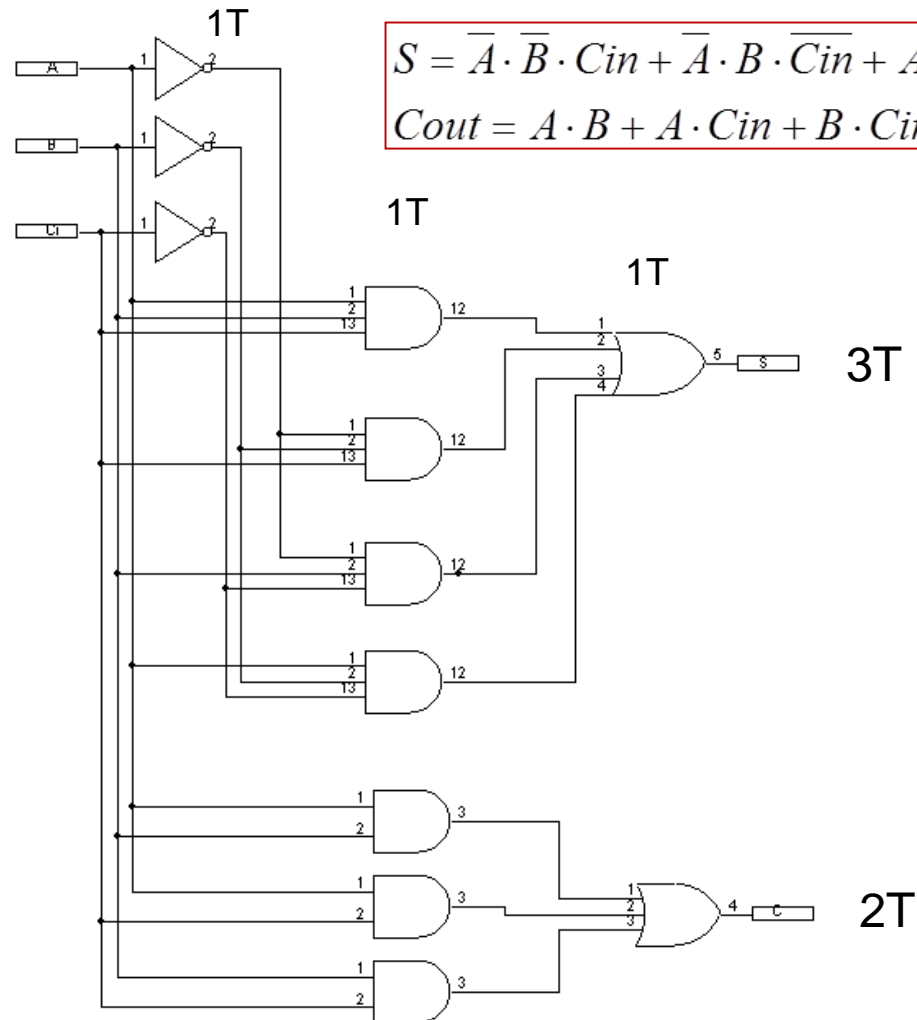


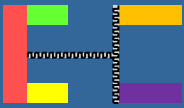


SUMADOR COMPLETO (F.A.)

Unidad
aritmética
entera.
Sumar y
restar

Sumador completo (FA) – Puertas lógicas (expresión booleana)

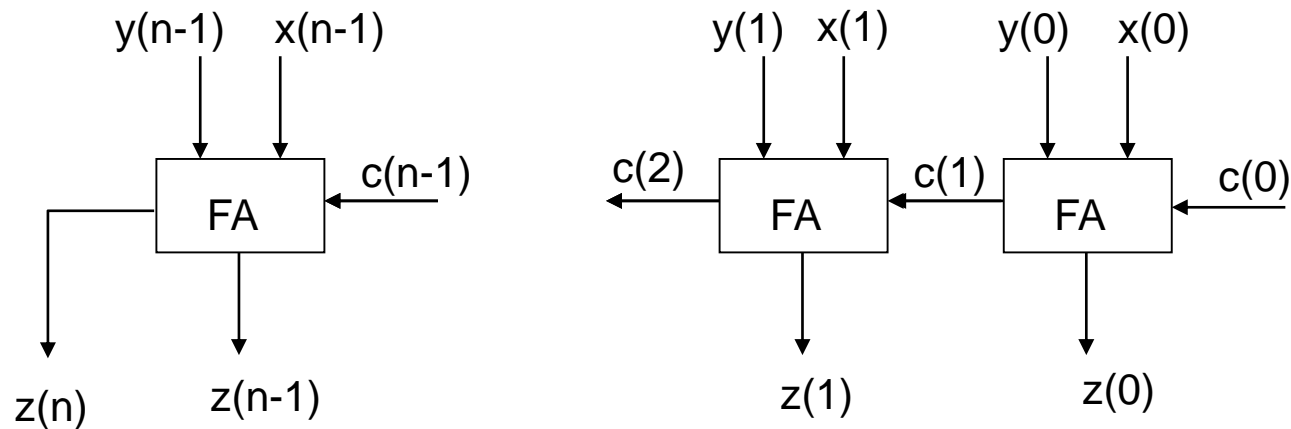


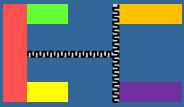


SUMADOR CON PROPAGACIÓN DE ACARREO (CPA)

Unidad
aritmética
entera.
Sumar y
restar

- La estructura para sumar dos números de n bits es colocar en cascada n sumadores completos.
- El acarreo se propaga de una etapa a la siguiente: Sumador con Propagación de Acarreo (Carry Propagated Adder)

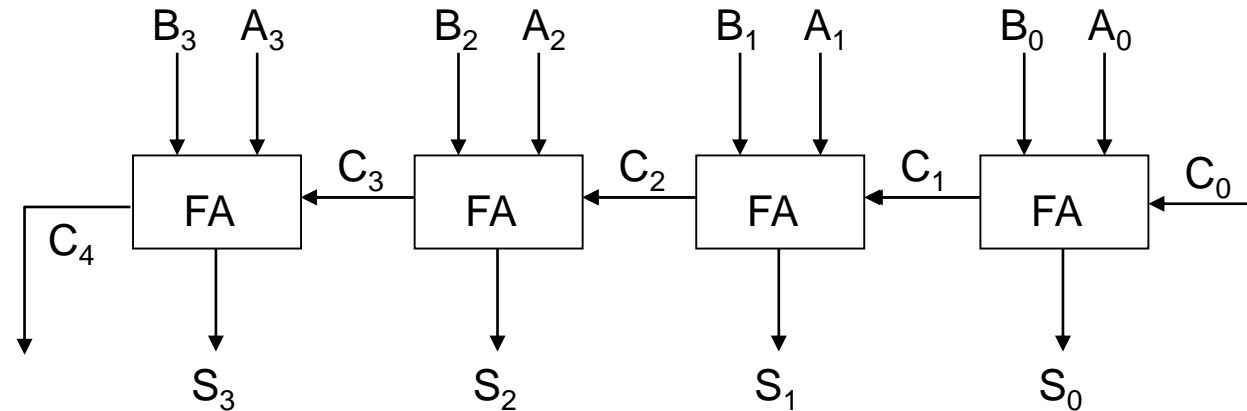


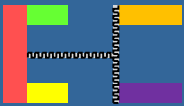


SUMADOR CON PROPAGACIÓN DE ACARREO (CPA)

Unidad
aritmética
entera.
Sumar y
restar

- Sumador con propagación de acarreo de 4 bits.



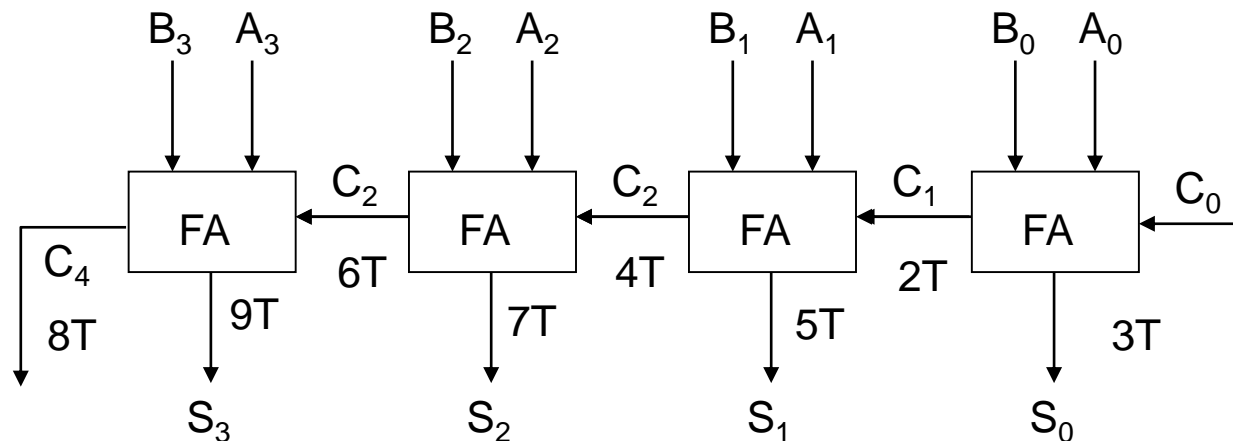
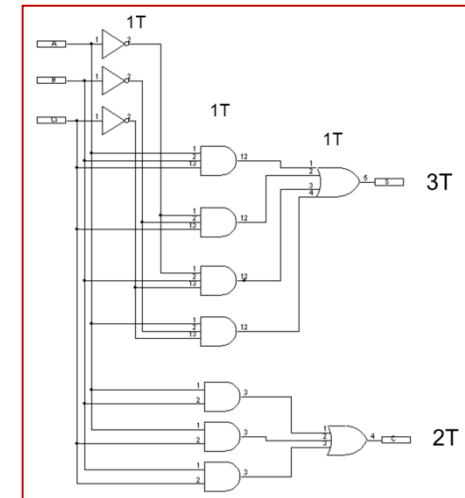


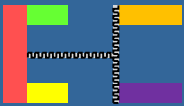
SUMADOR CON PROPAGACIÓN DE ACARREO (CPA)

Unidad
aritmética
entera.
Sumar y
restar

- 🎯 Sumadores contruidos con puertas lógicas a partir de la expresión:

$$S = \bar{A} \cdot \bar{B} \cdot Cin + \bar{A} \cdot B \cdot \bar{Cin} + A \cdot \bar{B} \cdot \bar{Cin} + A \cdot B \cdot Cin$$
$$Cout = A \cdot B + A \cdot Cin + B \cdot Cin$$

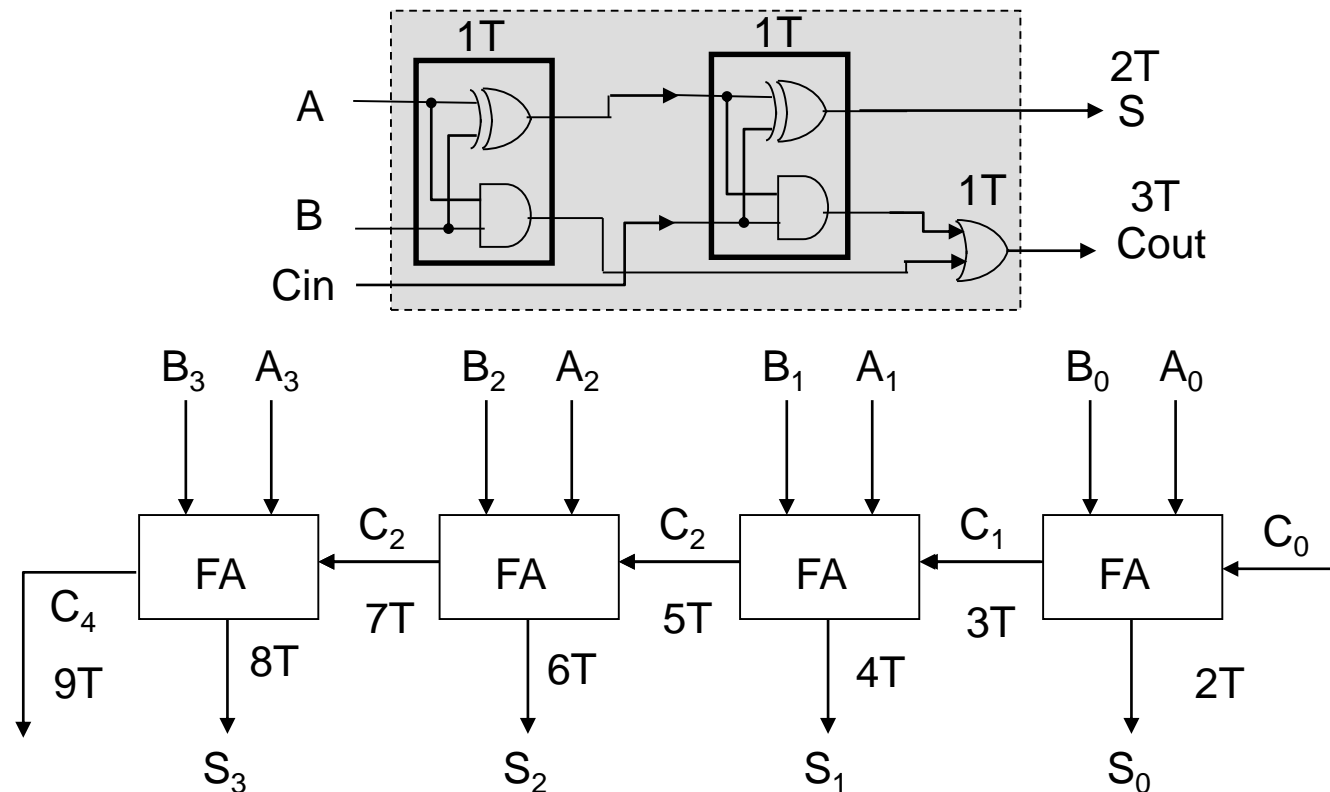




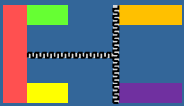
SUMADOR CON PROPAGACIÓN DE ACARREO (CPA)

Unidad
aritmética
entera.
Sumar y
restar

Sumadores completos contruidos con semisumadores:

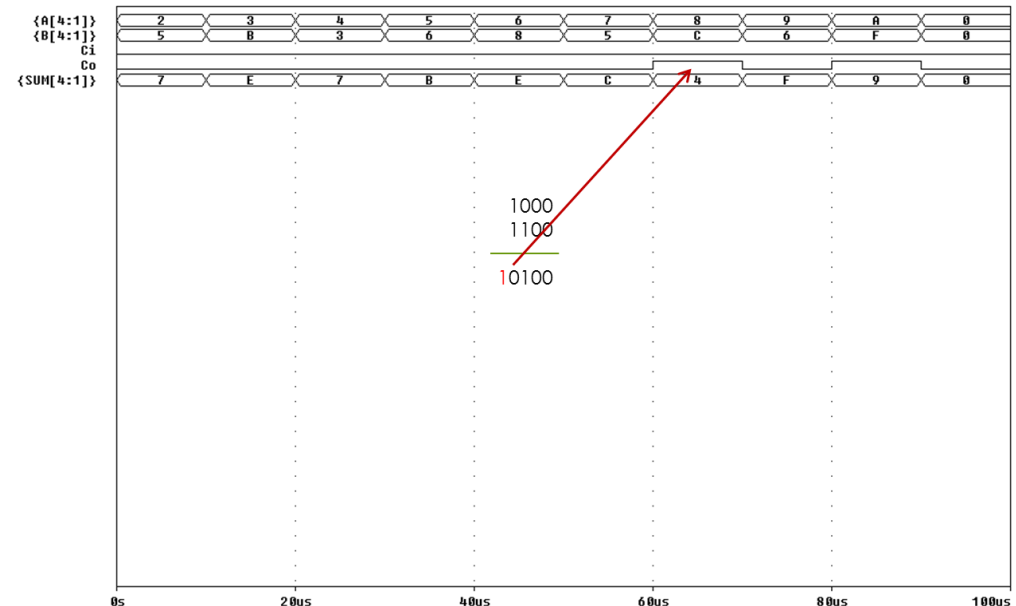
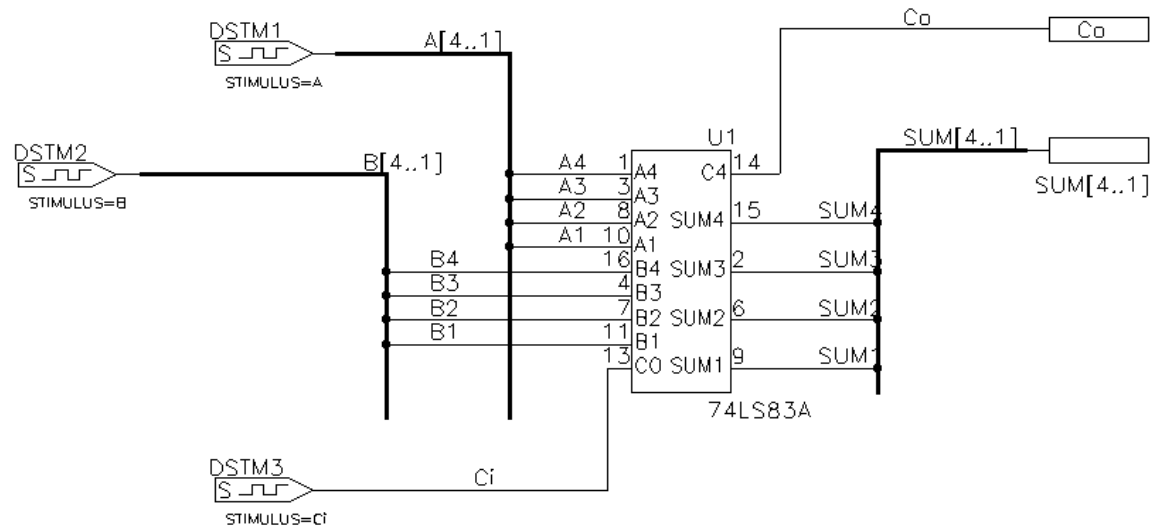


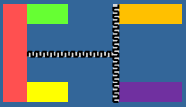
$$Tiempo_Total = (2 \cdot n + 1)T$$



CHIP SUMADOR 74LS83

Unidad
aritmética
entera.
Sumar y
restar

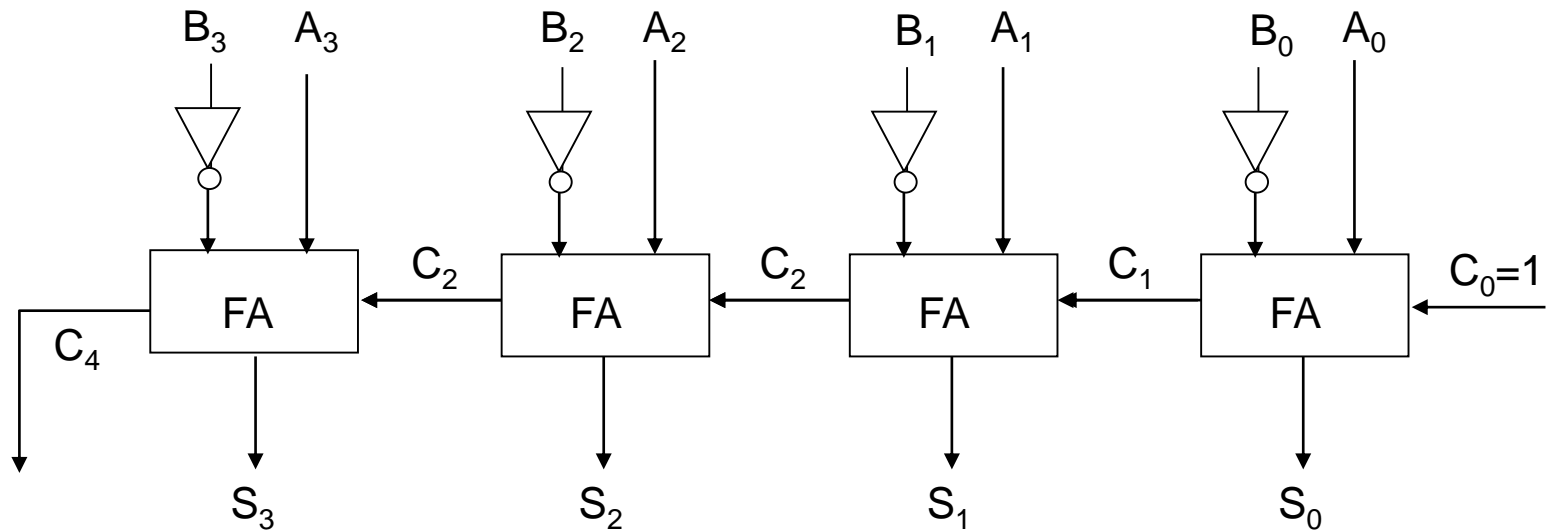


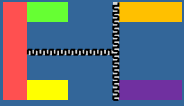


CIRCUITO RESTADOR

Unidad
aritmética
entera.
Sumar y
restar

- Suponer que se trabaja con números expresados en complemento a 2 ($C2$) $\rightarrow C2(B) = C1(B) + 1$.
- $A - B = A + (C1(B) + 1)$



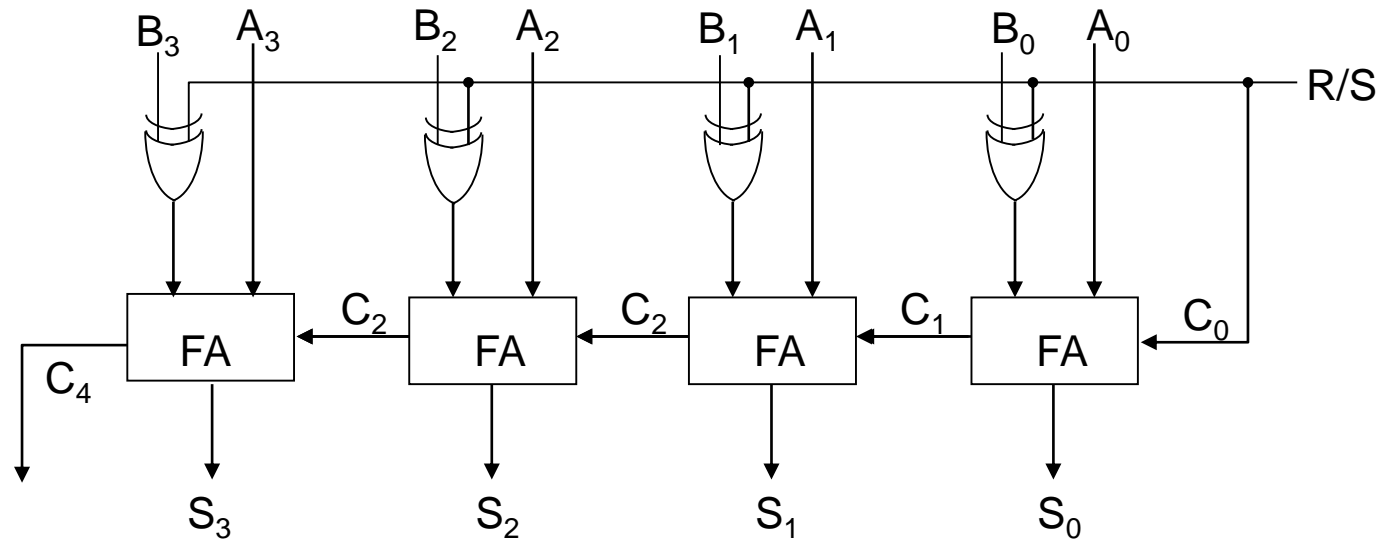


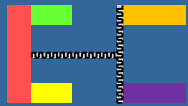
CIRCUITO SUMADOR-RESTADOR

Unidad
aritmética
entera.
Sumar y
restar

R/S	B _i	Entrada al FA
0	0	0
0	1	1
1	0	1
1	1	0

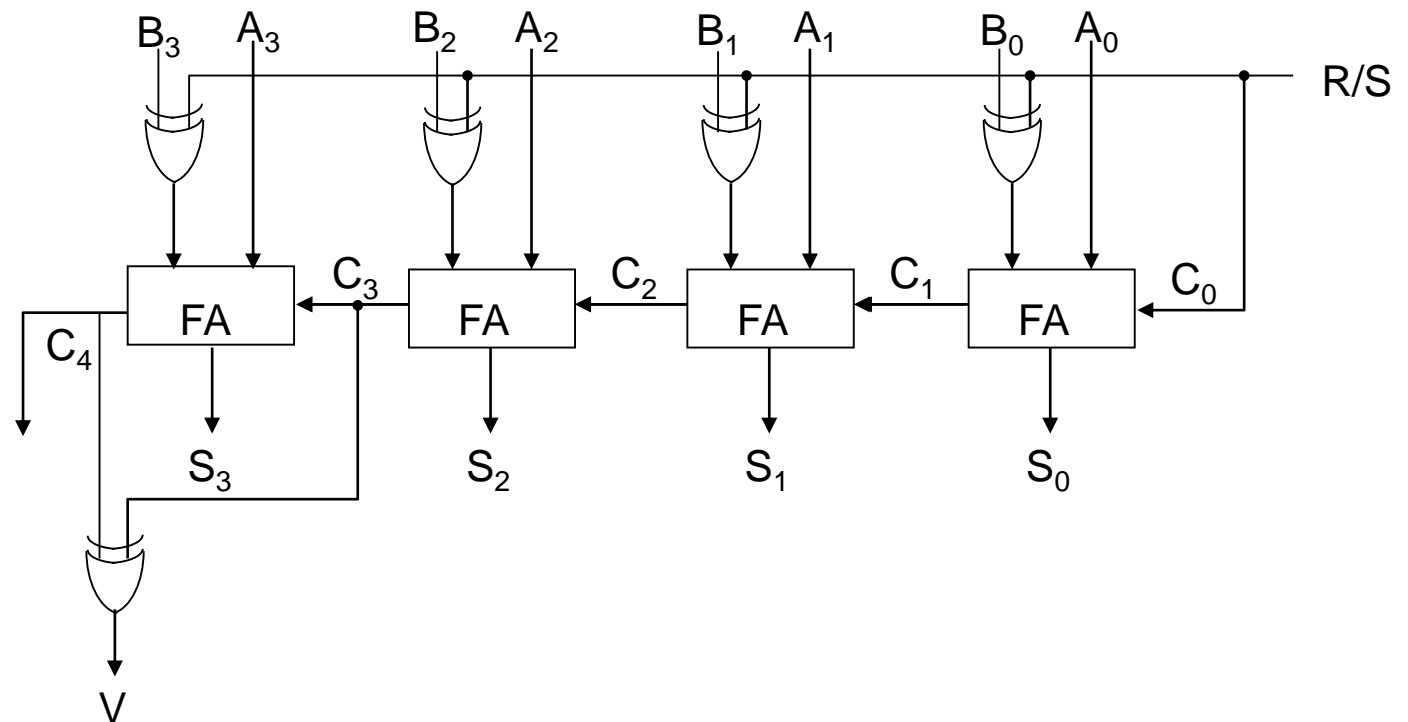
$$Tiempo_Total = 2(n+1)T$$

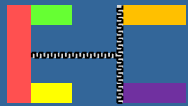




DETECCIÓN DE DESBORDAMIENTO

Sumador-Restador en C2 con detección de desbordamiento.





DETECCIÓN DE DESBORDAMIENTO

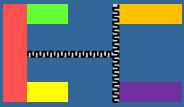
Unidad
aritmética
entera.
Sumar y
restar

1. Caso suma de dos positivos

S				
C4	C3	C2	C1	
0	1	1	1	
	0	1	1	1 (7)
	0	1	1	1 (7)
<hr/>				
1	1	1	0	OV → ¿ -2 ?

2. Caso suma de dos negativos

S				
C4	C3	C2	C1	
1	0	1	1	
	1	0	0	1 (-7)
	1	0	1	1 (-5)
<hr/>				
0	1	0	0	OV → ¿ 4 ?



SUMADOR CON ANTICIPACIÓN DE ACARREO (CLA)

Unidad
aritmética
entera.
Sumar y
restar

- ⊙ Carry Lookahead Adder (CLA)
- ⊙ Suponer A y B números de 4 bits

- ⊙ Señal generadora de acarreo : $G_i = a_i \cdot b_i$
- ⊙ Señal propagadora de acarreo:
$$\begin{cases} P_i = a_i \oplus b_i \\ P_i = a_i + b_i \end{cases}$$

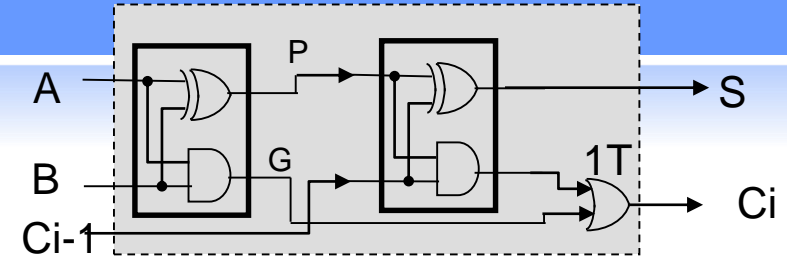
- ⊙ Acarreo de la etapa i: $C_i = G_i + P_i \cdot C_{i-1}$
- ⊙ Particularizando para A y B:

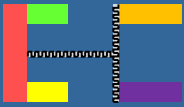
$$C_0 = G_0 + P_0 \cdot C_{-1}$$

$$C_1 = G_1 + P_1 \cdot C_0$$

$$C_2 = G_2 + P_2 \cdot C_1$$

$$C_3 = G_3 + P_3 \cdot C_2$$





SUMADOR CON ANTICIPACIÓN DE ACARREO (CLA)

Unidad
aritmética
entera.
Sumar y
restar

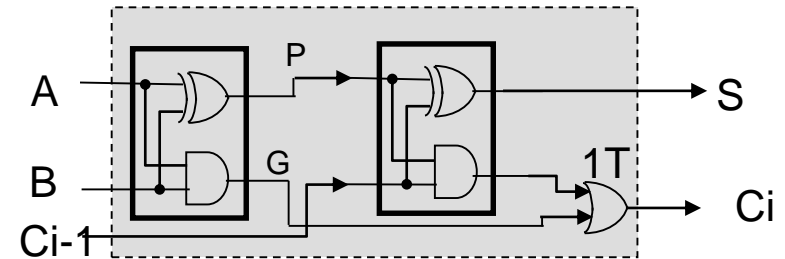
- Desarrollando las expresiones y poniéndolas en función de C_{-1} :

$$C_0 = G_0 + P_0 \cdot C_{-1}$$

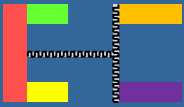
$$C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_{-1}$$

$$C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_{-1}$$

$$C_3 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_{-1}$$

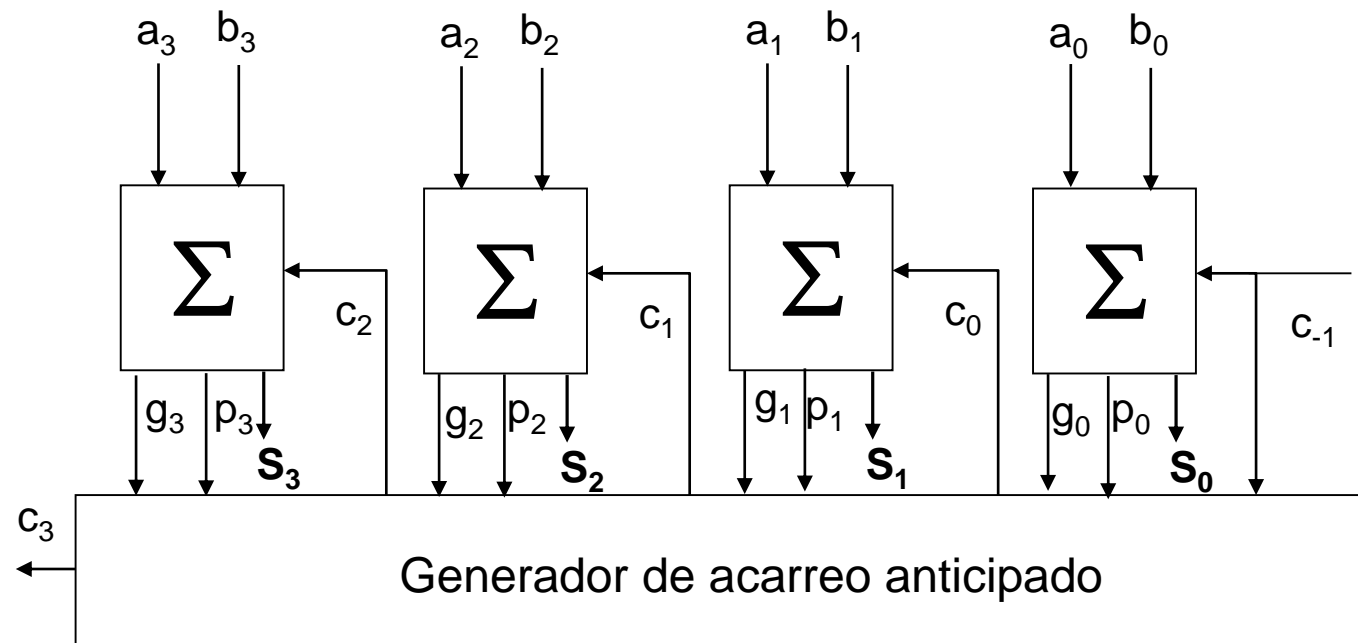


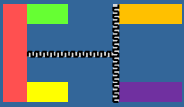
- Todos los acarrees dependen de a_i y b_i . (y de C_{-1})
- Estas expresiones se resuelven como suma de productos.
- Tres niveles de puertas lógicas para obtener cada uno de los acarrees.



SUMADOR CON ANTICIPACIÓN DE ACARREO (CLA)

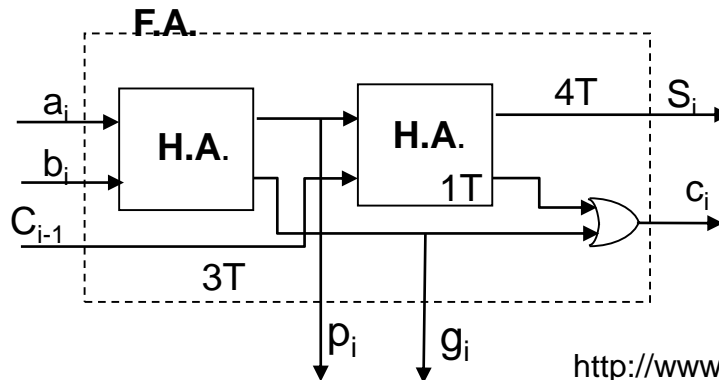
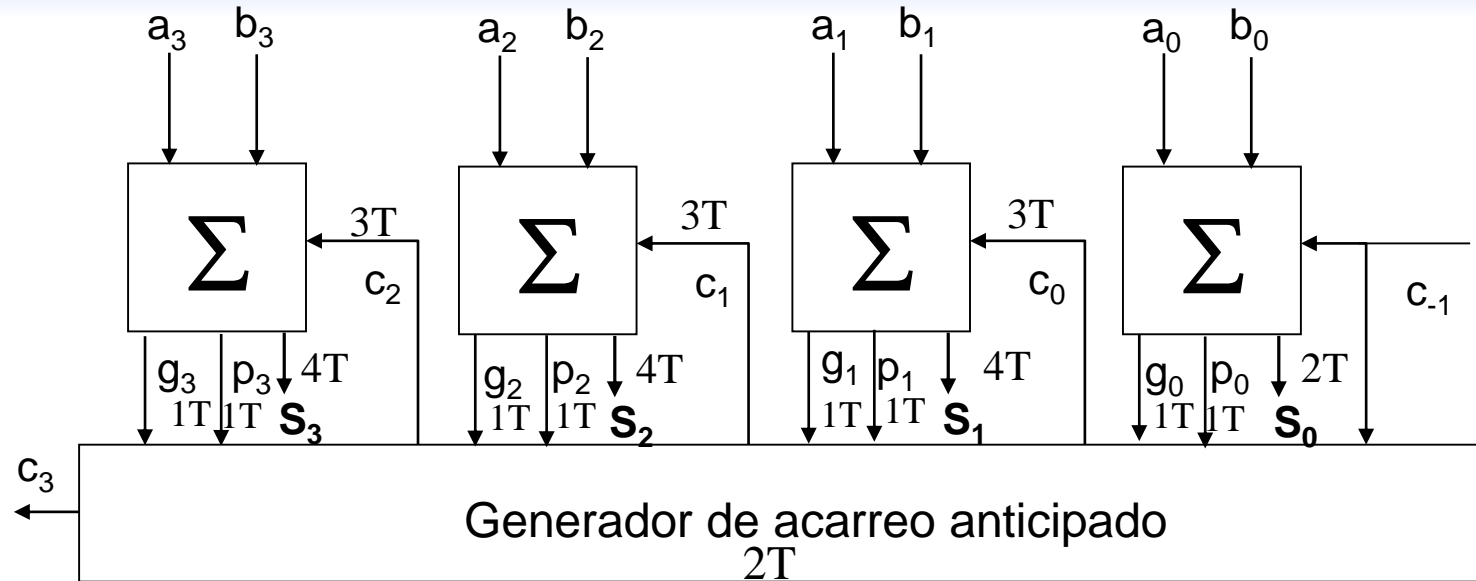
Unidad
aritmética
entera.
Sumar y
restar





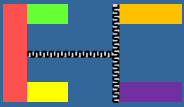
SUMADOR CON ANTICIPACIÓN DE ACARREO (CLA)

Unidad
aritmética
entera.
Sumar y
restar



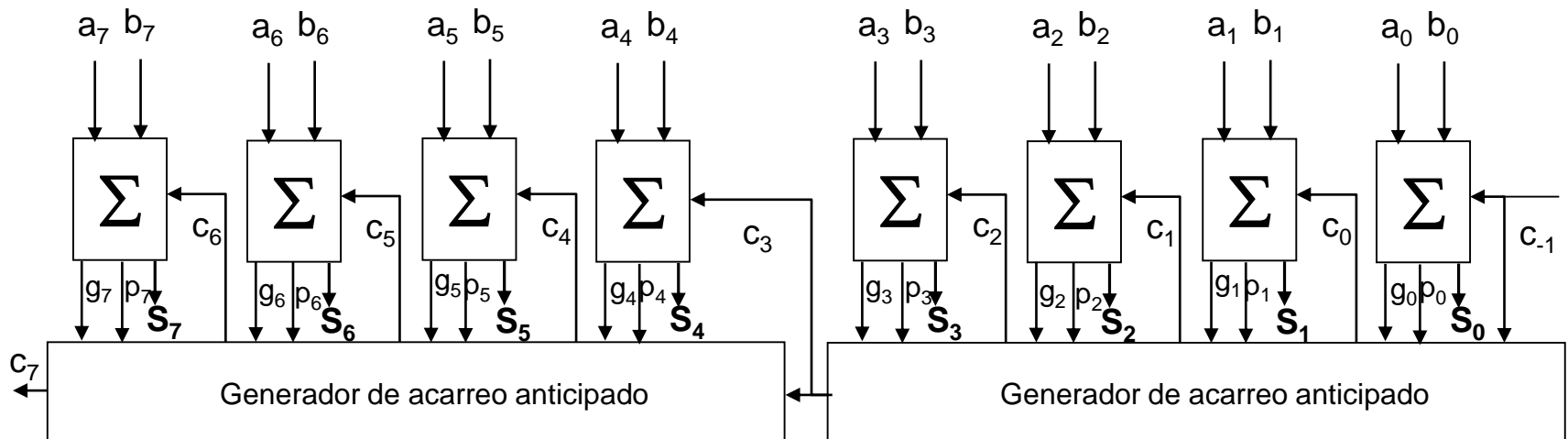
Sumadores contruidos
con semisumadores

<http://www.ecs.umass.edu/ece/koren/arith/simulator/Add/lookahead/>



SUMADOR CON ANTICIPACIÓN DE ACARREO (CLA): EJEMPLO DE 8 BITS

Unidad
aritmética
entera.
Sumar y
restar



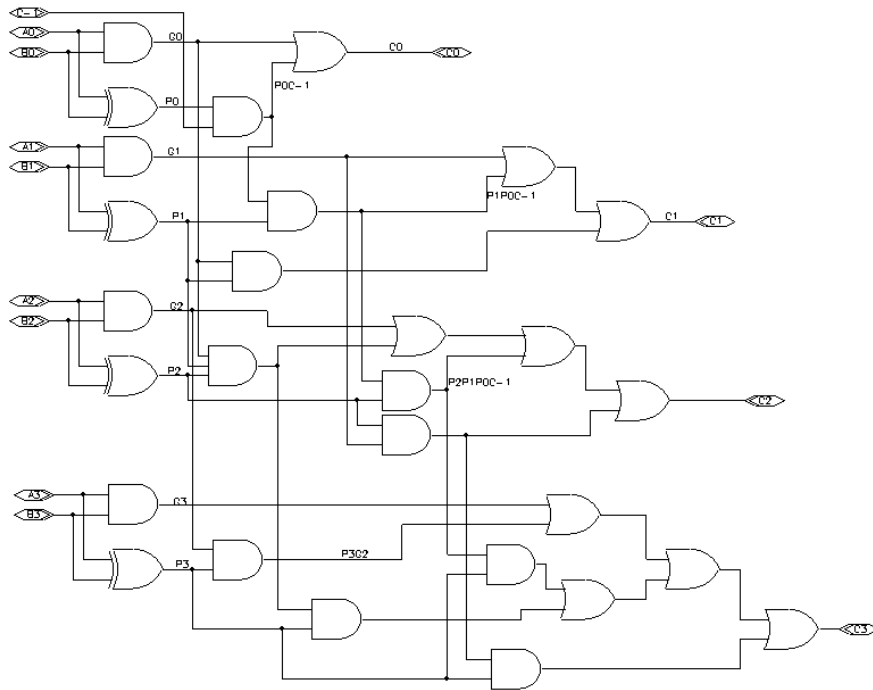
Calcular los retardos en este CLA suponiendo que los sumadores se construyen con semisumadores.

Comparar el resultado con el de un sumador CPA de 8 bits.



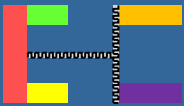
SUMADOR CON ANTICIPACIÓN DE ACARREO (CLA): EJEMPLO DE 4 BITS

Unidad
aritmética
entera.
Sumar y
restar



C3	C2	C1	C0		C3	C2	C1	C0
0	1	1	1		1	1	1	
	0	1	1	1		1	0	1
	0	1	1	1		1	1	1
						1	0	0

{A[3:0]}	7	8	A	6	4	F	E	C	0
{B[3:0]}	7	C	F	7	5	7	3		0
C-1									
C0									
C1									
C2									
C3									



LA MULTIPLICACIÓN

Unidad
aritmética
entera.
Multiplicar y
dividir

- ⊙ Algoritmo de sumas y desplazamientos
- ⊙ Si multiplicando de n bits y multiplicador de m bits, entonces el producto tendrá una longitud de $n+m$ bits.
- ⊙ Multiplicación binaria: sencilla ya que hay que multiplicar por 1 o por 0.

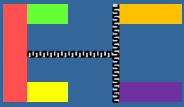
Multiplicando

Multiplicador

Producto

			5	3	2	
			4	3	1	
			<hr/>			
			5	3	2	
	1	5	9	6		
2	1	2	8			
<hr/>						
2	2	9	2	9	2	

			1	0	0	
				1	0	
			<hr/>			
			0	0	0	
1	0	0				
<hr/>						
1	0	0	0	0		



MULTIPLICACIÓN BINARIA SIN SIGNO

Unidad
aritmética
entera.
Multiplicar y
dividir

Repetir n veces

Si el bit 0 del multiplicador=1 entonces

Sumar el multiplicando a la mitad izquierda del producto y colocar el resultado en la mitad izquierda del producto.

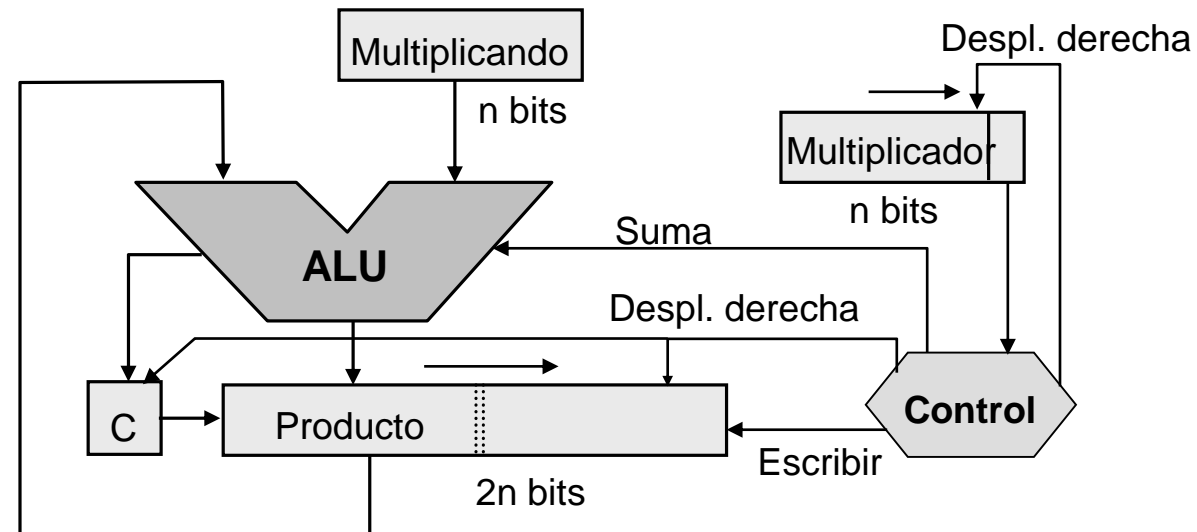
Fin entonces

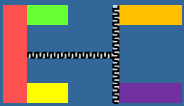
Desplazar 1 bit a la derecha el registro producto

Desplazar 1 bit a la derecha el registro multiplicador

Fin repetir

Versión preliminar



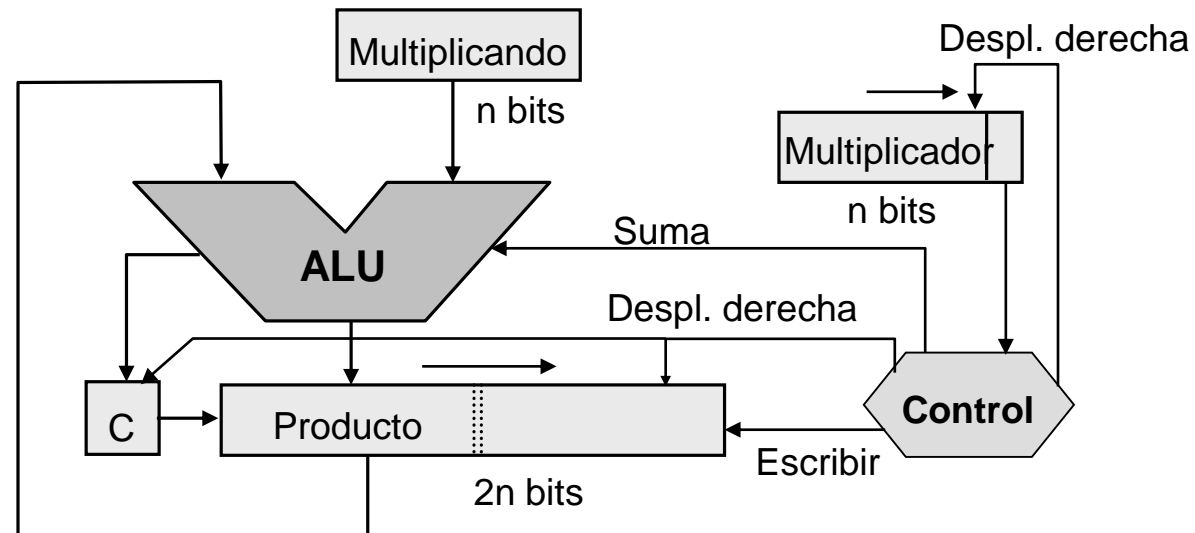


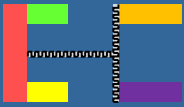
MULTIPLICACIÓN BINARIA SIN SIGNO

Unidad
aritmética
entera.
Multiplicar y
dividir

Multiplicando	1	0	1	1					11_d
Multiplicador					1	1	0	1	13_d
					1	0	1	1	
			0	0	0	0			
		1	0	1	1				
	1	0	1	1					
Producto	1	0	0	0	1	1	1	1	143_d

*Versión
preliminar*





MULTIPLICACIÓN BINARIA SIN SIGNO

Unidad
aritmética
entera.
Multiplicar y
dividir

Repetir n veces

Si el bit 0 del registro producto=1 entonces

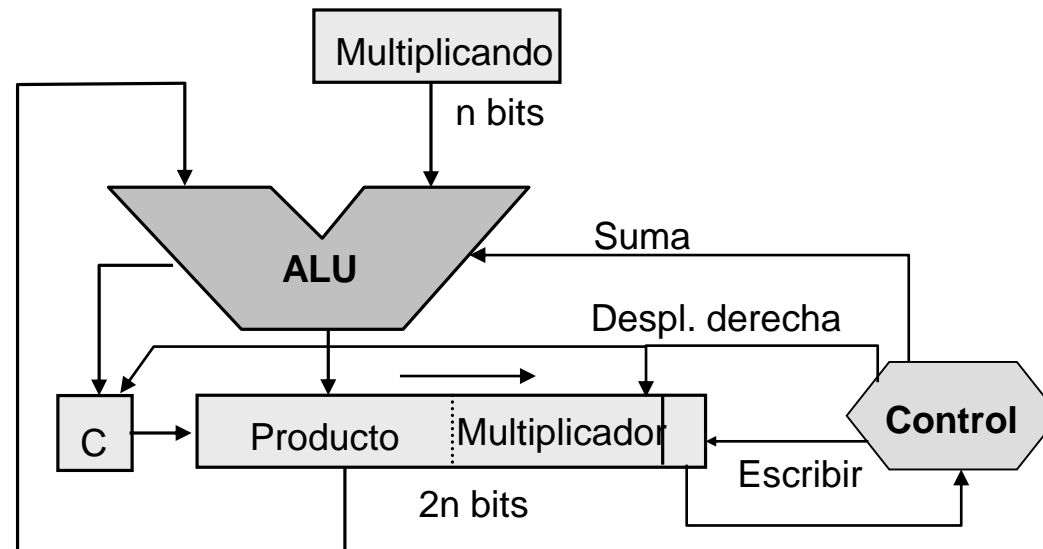
Sumar el multiplicando a la mitad izquierda del producto ($prod_H$) y colocar el resultado en la mitad izquierda del producto $\rightarrow prod_H = prod_H + \text{multiplicando}$

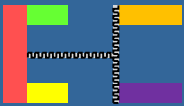
Fin entonces

Desplazar 1 bit a la derecha el registro producto

Fin repetir

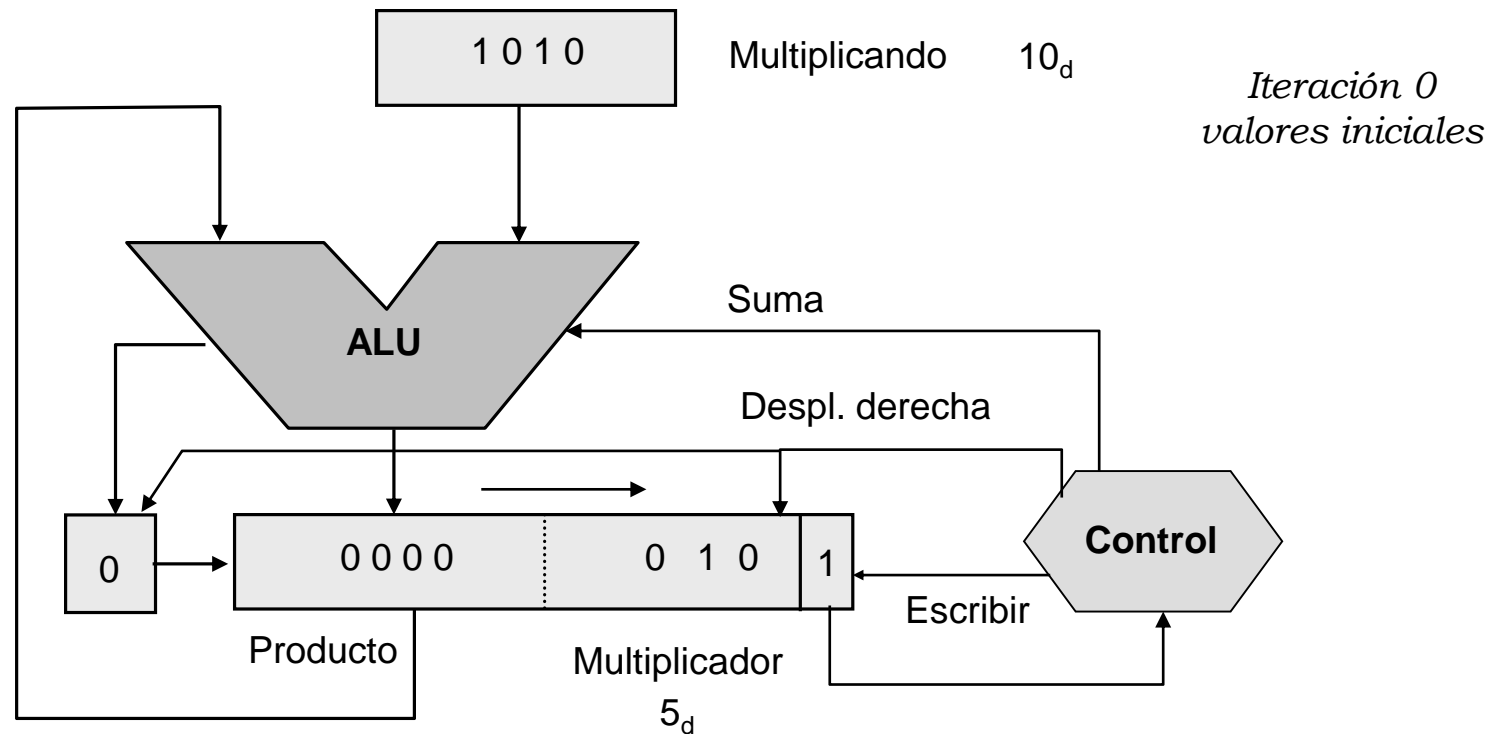
*Versión
final*

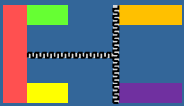




MULTIPLICACIÓN BINARIA SIN SIGNO

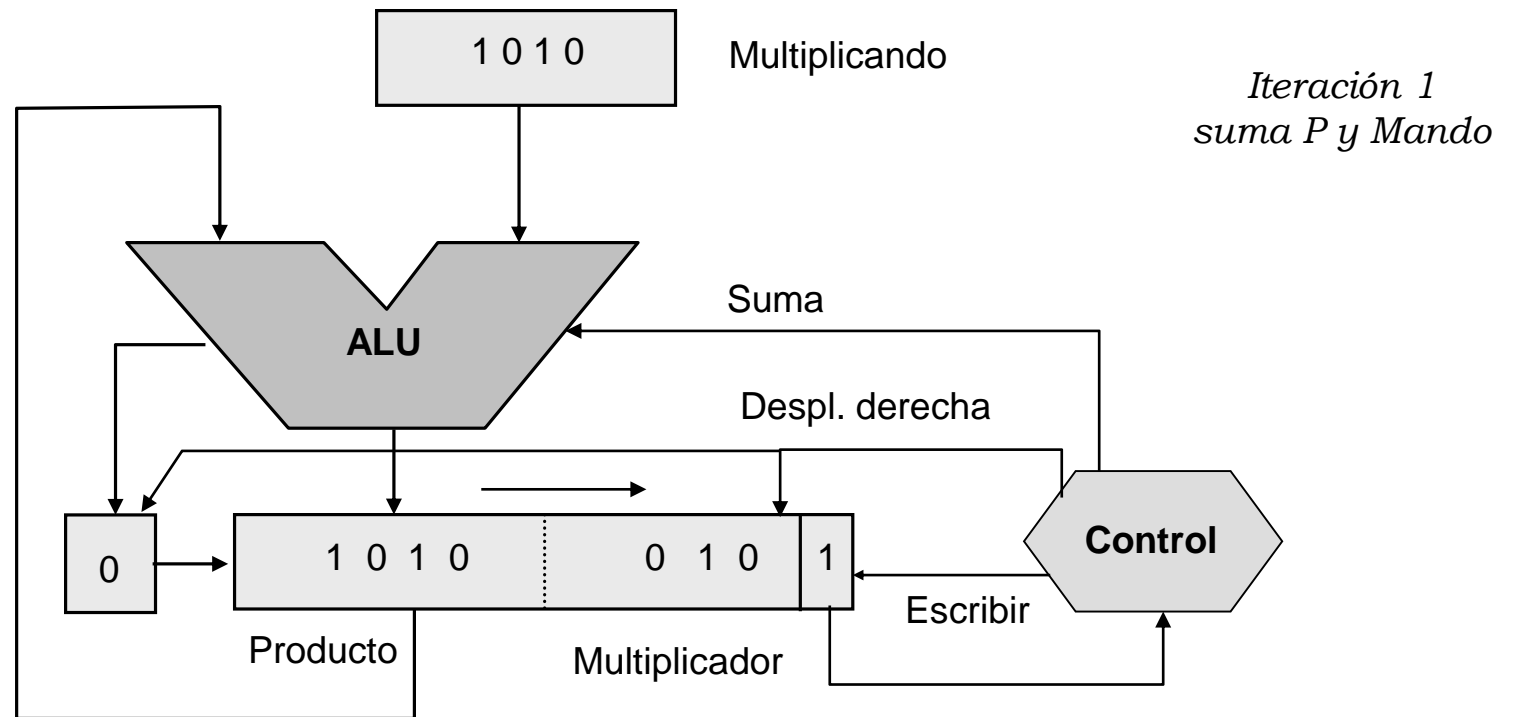
Unidad
aritmética
entera.
Multiplicar y
dividir

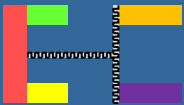




MULTIPLICACIÓN BINARIA SIN SIGNO

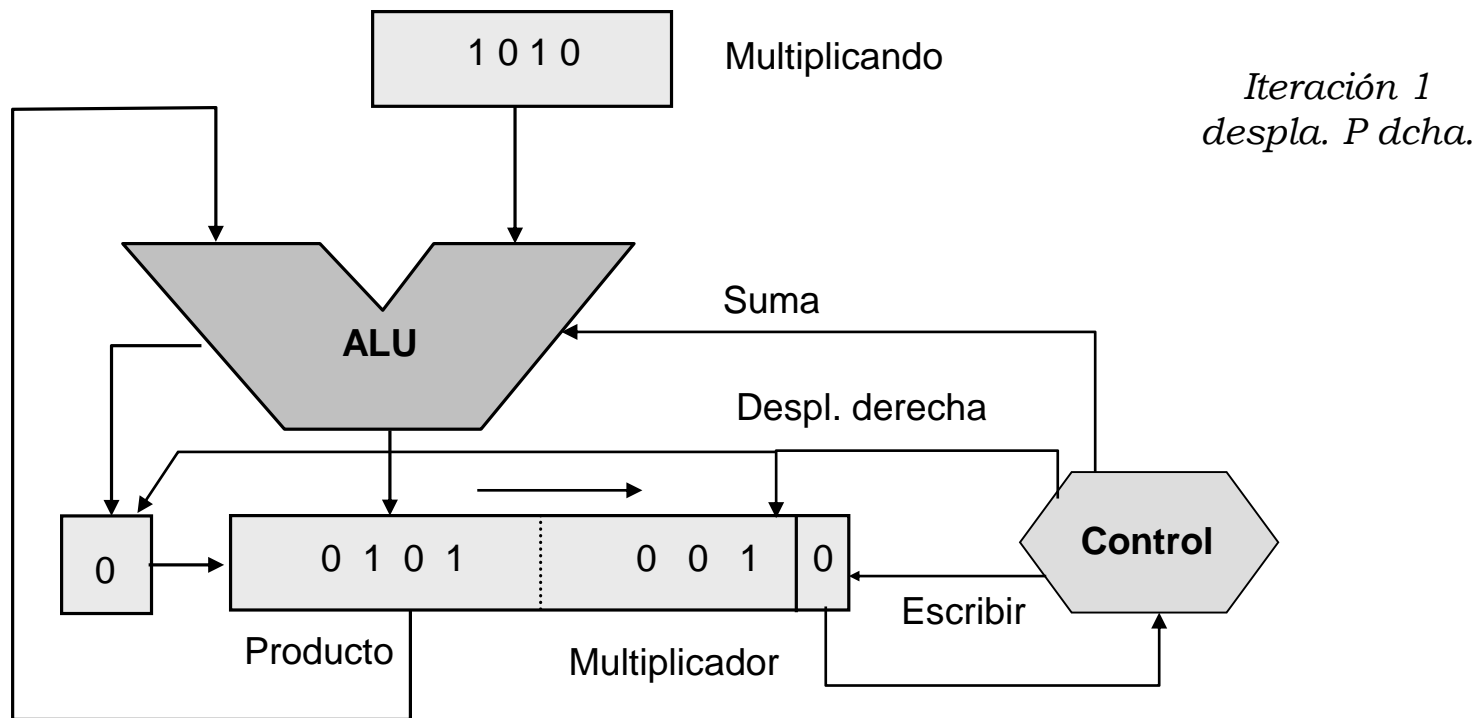
Unidad
aritmética
entera.
Multiplicar y
dividir

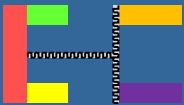




MULTIPLICACIÓN BINARIA SIN SIGNO

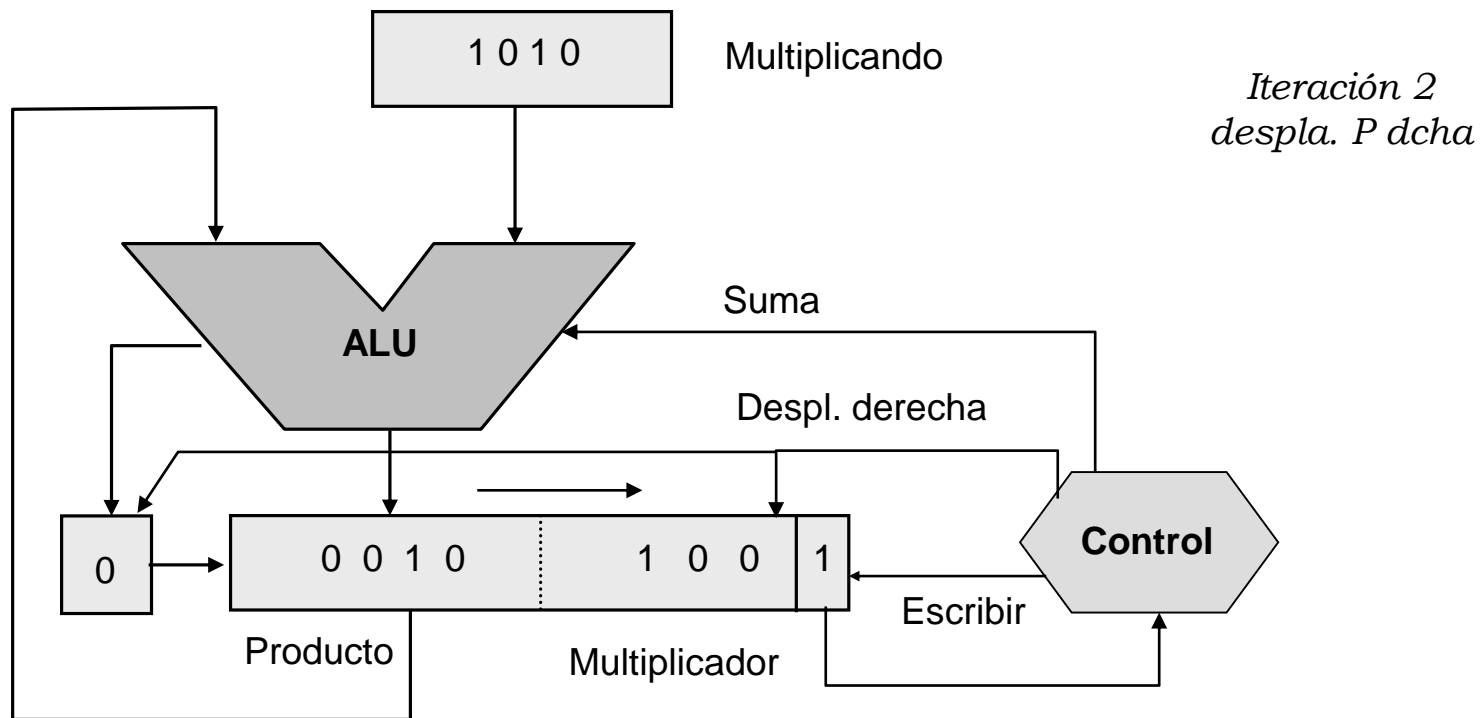
Unidad
aritmética
entera.
Multiplicar y
dividir

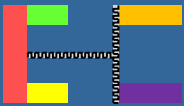




MULTIPLICACIÓN BINARIA SIN SIGNO

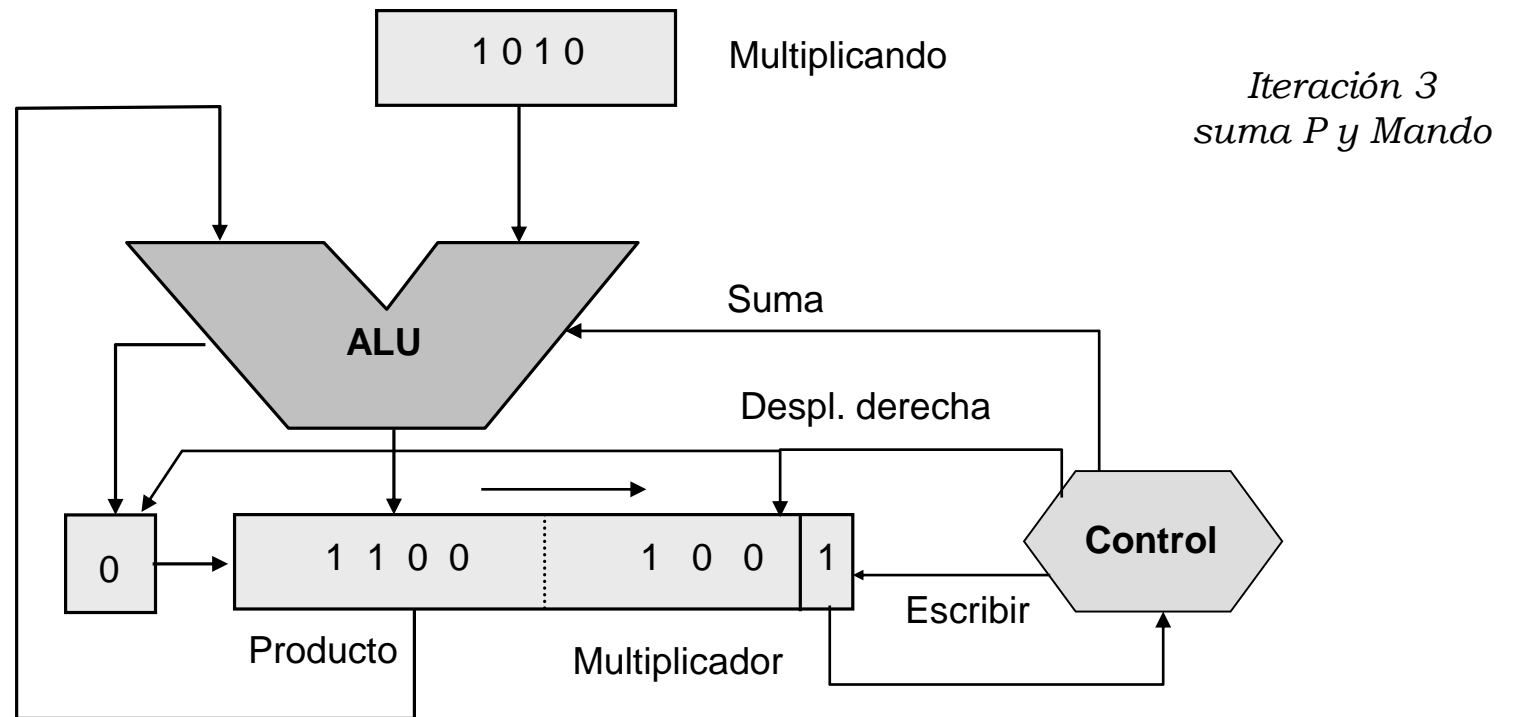
Unidad
aritmética
entera.
Multiplicar y
dividir

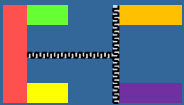




MULTIPLICACIÓN BINARIA SIN SIGNO

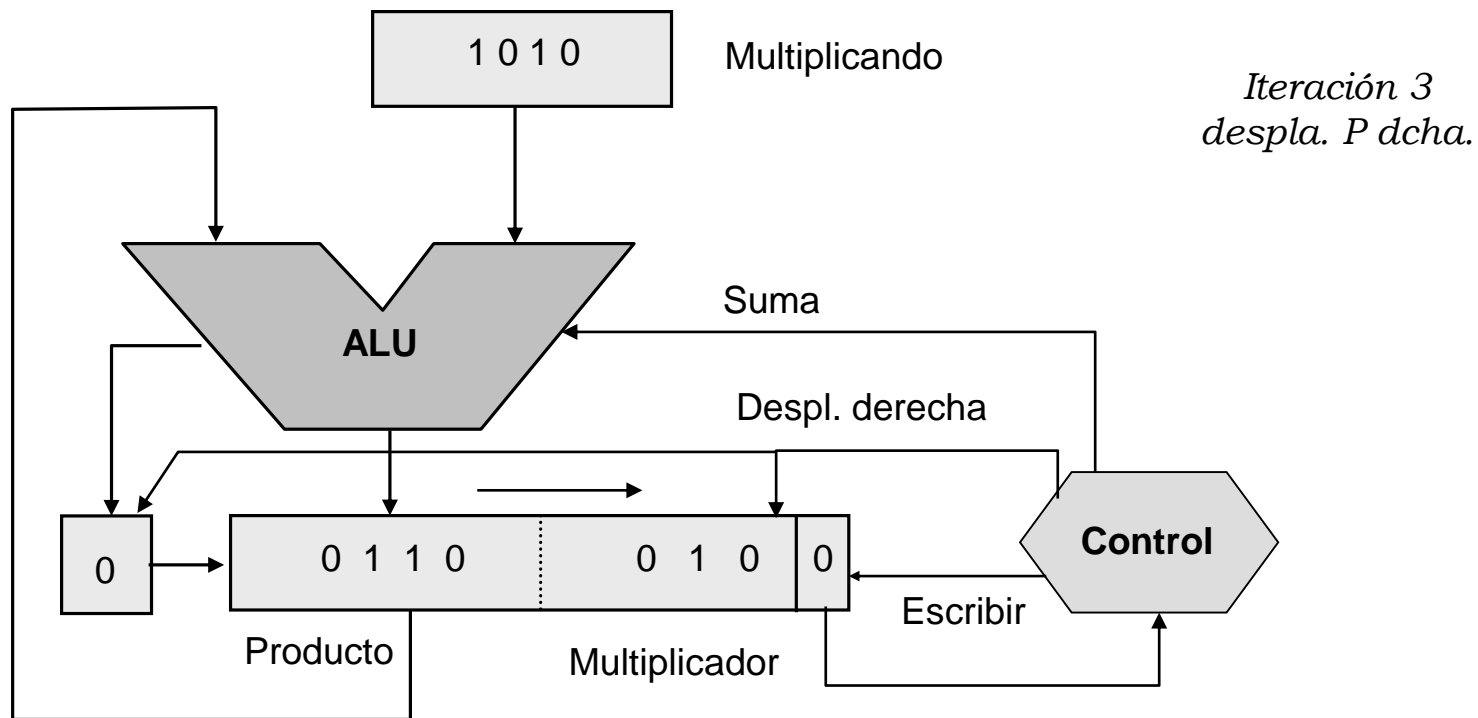
Unidad
aritmética
entera.
Multiplicar y
dividir

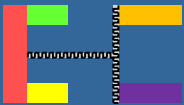




MULTIPLICACIÓN BINARIA SIN SIGNO

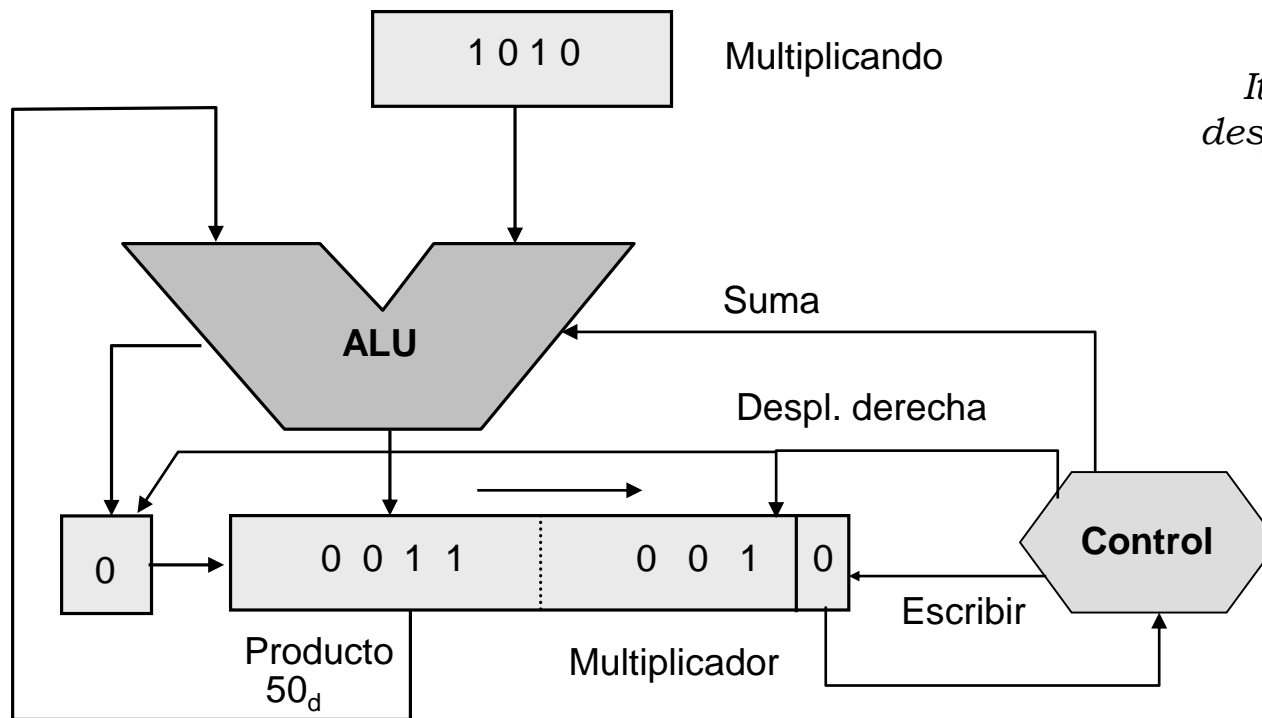
Unidad
aritmética
entera.
Multiplicar y
dividir

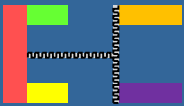




MULTIPLICACIÓN BINARIA SIN SIGNO

Unidad
aritmética
entera.
Multiplicar y
dividir





MULTIPLICACIÓN BINARIA SIN SIGNO

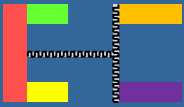
Unidad
aritmética
entera.
Multiplicar y
dividir

Multiplicando = 1010 10_d

Multiplicador = 0101 5_d

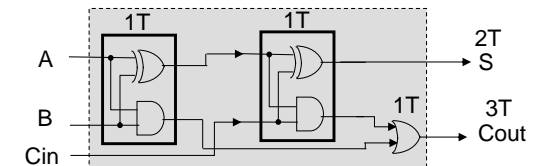
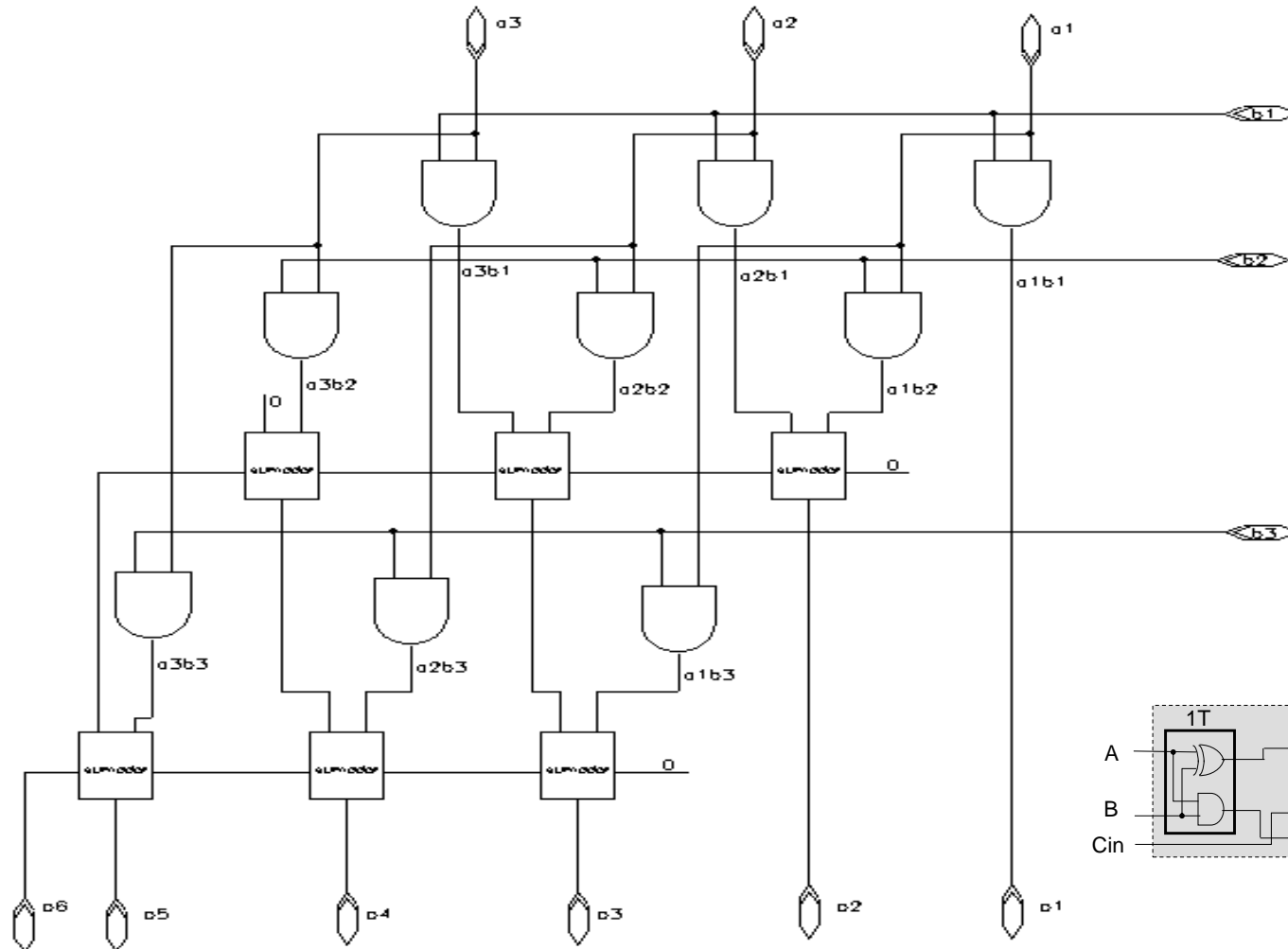
Producto	Multiplicando	Acción	Iteración
0000 0101	1010	Valores iniciales	0
1010 0101	1010	Sumar $prod_H$ + multiplicando	1
0101 0010	1010	Desplazar 1 bit a la derecha	1
0010 1001	1010	Despl. 1 bit a la derecha	2
1100 1001	1010	Sumar $prod_H$ + multiplicando	3
0110 0100	1010	Desplazar 1 bit a la derecha	3
0011 0010	1010	Desplazar 1 bit a la derecha	4

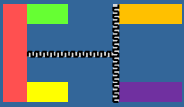
50_d



MULTIPLICACIÓN RÁPIDA (C. COMBINACIONAL)

Unidad
aritmética
entera.
Multiplicar y
dividir





MULTIPLICACIÓN BINARIA CON SIGNO (M. TRADICIONAL)

Unidad
aritmética
entera.
Multiplicar y
dividir

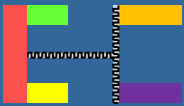
- Supongamos números expresados en C2
- $A = 1010 (-6_d)$ y $B = 0011 (+3_d)$
- Apliquemos algoritmo de sumas y desplazamientos

$$\begin{array}{r} 1010 \\ \times 0011 \\ \hline 1010 \\ 0000 \\ 0000 \\ \hline 0011110 \end{array}$$

Versión errónea

$$\begin{array}{r} 1010 \\ \times 0011 \\ \hline 11111010 \\ 1111010 \\ 000000 \\ 00000 \\ \hline 11101110 \end{array}$$

Versión correcta (ext. de signo)



MULTIPLICACIÓN BINARIA CON SIGNO (M. TRADICIONAL)

Unidad
aritmética
entera.
Multiplicar y
dividir

🎯 Ejemplos:

$$5 * 5 = 25$$

$$\begin{array}{r} 0101 \\ X 0101 \\ \hline 0101 \\ 0000 \\ 0101 \\ 0000 \\ \hline 00011001 \end{array}$$

$$(-5) * 5 = -25$$

$$\begin{array}{r} 1011 \\ X 0101 \\ \hline 1011 \\ 0000 \\ 1011 \\ 0000 \\ \hline 00110111 \neq -25 \end{array}$$

$$(-5) * 5 = -25$$

$$\begin{array}{r} 1011 \\ X 0101 \\ \hline 11111011 \\ 0000000 \\ 111011 \\ 00000 \\ \hline 11100111 = -25 \end{array}$$

Los productos parciales deben
representarse en C2

$$5 * (-5) = -25$$

$$\begin{array}{r} 0101 \\ X 1011 \\ \hline 0101 \\ 0101 \\ 0000 \\ 0101 \\ \hline 00110111 \neq -25 \end{array}$$

$$5 * (-5) = -25$$

$$\begin{array}{r} 0101 \\ X 1011 \\ \hline 00000101 \\ 0000101 \\ 000000 \\ \mathbf{11011} \\ \hline 11100111 = -25 \end{array}$$

En la última iteración hay que
restar el multiplicando



MULTIPLICACIÓN BINARIA CON SIGNO (ALGORITMO DE BOOTH)

Unidad
aritmética
entera.
Multiplicar y
dividir

- Supongamos Multiplicando = 2 y Multiplicador = 7 (en binario 0010 x 0111)
- Booth simplifica el multiplicador para realizar menos productos parciales buscando polinomios de menos potencias de 2: P. ej. Booth expresó 7 como $8 - 1$ y sustituyó el multiplicador por esta descomposición:

$$0111 = 1000 - 0001 = +100-1 = 2^3 + 0^2 + 0^1 - 1^0 = 7$$

				0	0	1	0
				x	+1	0	0
							-1
1	1	1	1	1	1	1	0
0	0	0	0	0	0		
0	0	0	1	0			
0	0	0	0	1	1	1	0

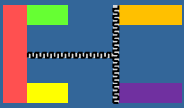
Multiplicando

Multiplicador según A. Booth

Restamos el multiplicando

2 despl. (2 ceros en el multiplicador)

Sumamos el multiplicando



MULTIPLICACIÓN BINARIA CON SIGNO (ALGORITMO DE BOOTH)

Unidad
aritmética
entera.
Multiplicar y
dividir

Bit n	Bit n-1	Sustitución
0	0	0 (no hay transición)
1	0	-1 (transición hacia negativo)
0	1	+1 (transición hacia positivo)
1	1	0 (no hay transición)

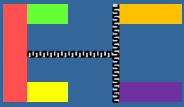
Se establece que $q_{-1} = 0$ para calcular Booth del bit menos significativo

Ejemplo: Multiplicando = 11101110 (-18) y Multiplicador = 01111010 (0) (122)

Recodificación del multiplicador según Booth = $+1000-1+1-10 = 2^7-2^3+2^2-2^1 = 122$

									1	1	1	0	1	1	1	0
								x	+1	0	0	0	-1	+1	-1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
1	1	1	1	1	1	1	1	1	1	0	1	1	1	0		
0	0	0	0	0	0	0	0	0	1	0	0	1	0			
1	1	1	1	0	1	1	1	0	0	0	0	0				
1	1	1	1	0	1	1	1	0	1	1	0	1	1	0	0	

(-2196)



MULTIPLICACIÓN BINARIA CON SIGNO (ALGORITMO DE BOOTH)

Unidad
aritmética
entera.
Multiplicar y
dividir

Inicialmente $q_{-1}=0$

Repetir n veces

Si $q_0 = 1$ y $q_{-1} = 0$ entonces

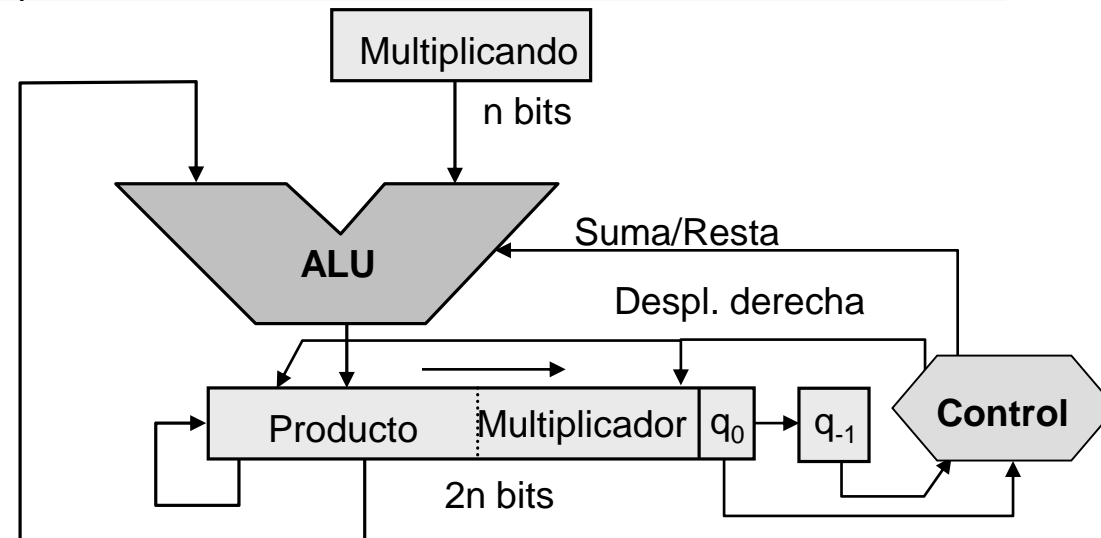
$\text{Producto}_h = \text{Producto}_h - \text{Multiplicando}$

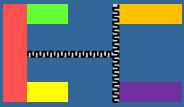
Si $q_0 = 0$ y $q_{-1} = 1$ entonces

$\text{Producto}_h = \text{Producto}_h + \text{Multiplicando}$

Desplazamiento aritmético a la derecha de Producto y q_{-1}

Fin repetir.





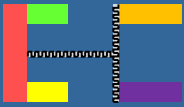
MULTIPLICACIÓN BINARIA CON SIGNO (ALGORITMO DE BOOTH)

Unidad
aritmética
entera.
Multiplicar y
dividir

Multiplicando = 1010 ⁽⁻⁶⁾

Multiplicador = 1110 ⁽⁻²⁾

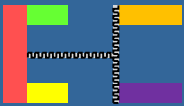
Multiplicando	Producto	q ₋₁	Acción	Iteración
1010	0000 1110	0	Valores iniciales	0
1010	0000 1110	0	00 → Ninguna operación	1
1010	0000 0111	0	Despl. arit. 1 bit dcha.	1
1010	0110 0111	0	10 → $\text{prod}_H = \text{prod}_H - \text{multiplicando}$	2
1010	0011 0011	1	Despl. arit. 1 bit dcha.	2
1010	0011 0011	1	11 → Ninguna operación	3
1010	0001 1001	1	Despl. arit. 1 bit dcha.	3
1010	0001 1001	1	11 → Ninguna operación	4
1010	0000 1100	1	Despl. arit. 1 bit dcha.	4



- ⊙ La división la podemos expresar como:
$$\text{Dividendo} = \text{Cociente} \times \text{Divisor} + \text{Resto}$$
- ⊙ El resto es más pequeño que el divisor. Hay que reservar el doble de espacio para el dividendo.
- ⊙ Supondremos operandos positivos.

Dividendo →

10010011 (147)	<u>1011</u> (11) ← Divisor
10010 (18)	01101 ₁₃ ← Cociente
1011	
001110 (14)	
1011	
001111 (15)	
1011	
0100 (4)	← Resto



DIVISIÓN: ALGORITMO SIN RESTAURACIÓN

Unidad
aritmética
entera.
Multiplicar y
dividir

$\text{Dividendo}_h = \text{Dividendo}_h - \text{Divisor}$

Repetir n veces

Si $\text{Dividendo}_h < 0$ entonces

Desplazar el Dividendo a la izquierda

$\text{Dividendo}_h = \text{Dividendo}_h + \text{Divisor}$

Sino

Desplazar el Dividendo a la izquierda

$\text{Dividendo}_h = \text{Dividendo}_h - \text{Divisor}$

Fin Si

Si $\text{Dividendo}_h < 0$ entonces

$q_0 = 0$

Sino

$q_0 = 1$

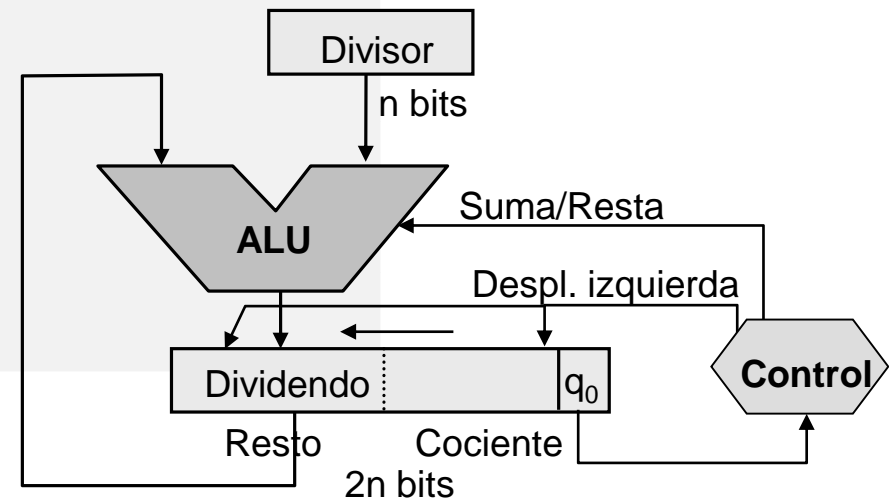
Fin Si

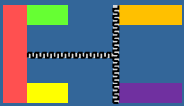
Fin Repetir

Si $\text{Dividendo}_h < 0$ entonces

$\text{Dividendo}_h = \text{Dividendo}_h + \text{Divisor}$

Fin Si





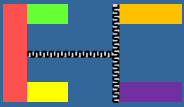
DIVISIÓN: ALGORITMO SIN RESTAURACIÓN

Unidad
aritmética
entera.
Multiplicar y
dividir

Dividendo	Divisor	Acción	Iteración
0000 0111 (7)	0010 (2)	Valores iniciales	0
1110 0111	0010	$\text{Dividendo}_h - \text{Divisor}$	0
1100 111_	0010	$\text{Dividendo}_h < 0 \Rightarrow$ Desplazar Izda	1
1110 111_	0010	$\text{Dividendo}_h + \text{Divisor}$	1
1110 1110	0010	$\text{Dividendo}_h < 0 \Rightarrow q_0 = 0$	1
1101 110_	0010	$\text{Dividendo}_h < 0 \Rightarrow$ Desplazar Izda	2
1111 110_	0010	$\text{Dividendo}_h + \text{Divisor}$	2
1111 1100	0010	$\text{Dividendo}_h < 0 \Rightarrow q_0 = 0$	2
1111 100_	0010	$\text{Dividendo}_h < 0 \Rightarrow$ Desplazar Izda	3
0001 100_	0010	$\text{Dividendo}_h + \text{Divisor}$	3
0001 1001	0010	$\text{Dividendo}_h \geq 0 \Rightarrow q_0 = 1$	3
0011 001_	0010	$\text{Dividendo}_h > 0 \Rightarrow$ Desplazar Izda	4
0001 001_	0010	$\text{Dividendo}_h - \text{Divisor}$	4
0001 0011	0010	$\text{Dividendo}_h > 0 \Rightarrow q_0 = 1$	4

↑ ↑

Resto(1) Cociente (3)



ARITMÉTICA EN COMA FLOTANTE

Unidad
aritmética
flotante.
IEEE 754

- ⊙ Representación para números fraccionarios
 - ⊙ Coma fija $1234,567$
 - ⊙ Logarítmica $\log 123,456 = 2,0915122$
 - ⊙ Coma flotante $1,234566 \times 10^3$
 - ⊙ Otras
- ⊙ Ventajas de estandarizar una representación determinada
 - ⊙ Posibilidad de disponer de bibliotecas de rutinas aritméticas
 - ⊙ Técnicas de implementación en hardware de alto rendimiento
 - ⊙ Construcción de aceleradores aritméticos estándar, etc.
- ⊙ El estándar más empleado es el 754-1985 del IEEE.



ESTÁNDAR IEEE 754 PARA COMA FLOTANTE

Unidad
aritmética
flotante.
IEEE 754

Formatos

Simple precisión (32 bits)

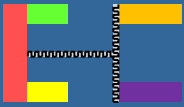
1 bit	8 bits	23 bits
signo	exponente	mantisa

Doble precisión (64 bits)

1 bit	11 bits	52 bits
signo	exponente	mantisa

Cuádruple precisión (128 bits)

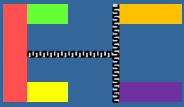
1 bit	15 bits	112 bits
signo	exponente	mantisa



ESTÁNDAR IEEE 754 PARA COMA FLOTANTE

Unidad
aritmética
flotante.
IEEE 754

- ⊙ Base del exponente 2
- ⊙ Exponente representado en exceso $2^{q-1}-1$
 - ⊙ Exceso a 127 en simple precisión
 - ⊙ Exceso a 1023 en doble precisión
- ⊙ Mantisa en valor absoluto; fraccionaria y normalizada con un uno implícito a la izquierda de la coma decimal.
 - ⊙ Mantisa de la forma 1,XXXXXX
 - ⊙ El primer uno nunca estará representado
 - ⊙ Valores posibles entre 1,00000..... y 1,11111....
- ⊙ S es el signo de la mantisa
- ⊙ Números
 - ⊙ $(-1)^S \times 1,M \times 2^{E-127}$ simple precisión
 - ⊙ $(-1)^S \times 1,M \times 2^{E-1023}$ doble precisión



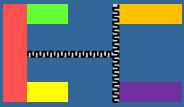
ESTÁNDAR IEEE 754 PARA COMA FLOTANTE

Unidad
aritmética
flotante.
IEEE 754

⊙ Casos especiales

E	M	Valores
2^{q-1}	$\neq 0$	NaN (no un Número)
2^{q-1}	0	$+\infty$ y $-\infty$ según el signo de S
0	0	Cero
0	$\neq 0$	Números desnormalizados

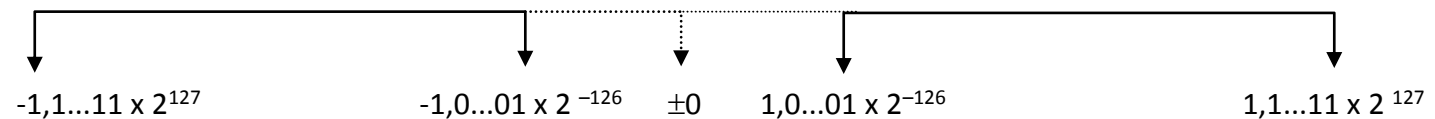
- ⊙ NaN resultado de operaciones tales como $0/0$, $\sqrt{-1}$
- ⊙ El valor cero tiene dos representaciones $+0$ y -0 .



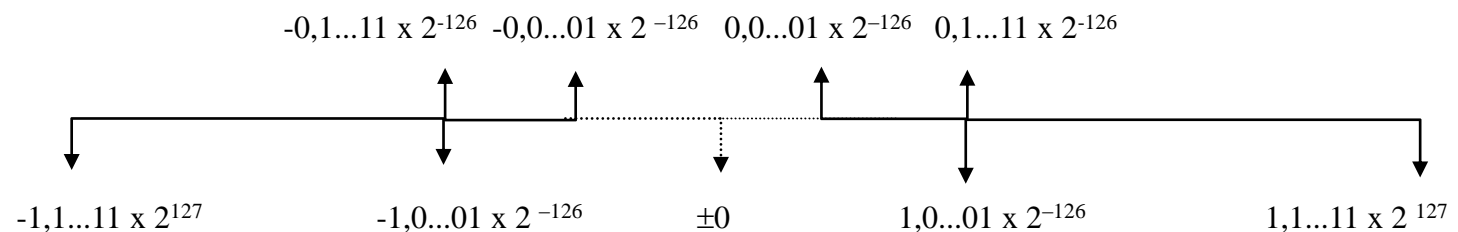
ESTÁNDAR IEEE 754 PARA COMA FLOTANTE

Unidad
aritmética
flotante.
IEEE 754

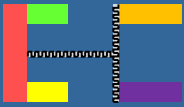
- Formato desnormalizado
 - $0, M \times 2^{-126}$ simple precisión
 - $0, M \times 2^{-1022}$ doble precisión



Sin números desnormalizados



Con números desnormalizados



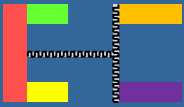
CONVERSIÓN DECIMAL-IEEE 754

Unidad
aritmética
flotante.
Sumar y
restar

🎯 Procedimiento

- 🕒 1. Representar en coma fija el número decimal.
- 🕒 2. Pasar número decimal a binario.
- 🕒 3. Normalizar mantisa.
- 🕒 4. Normalizar exponente.
- 🕒 5. Representar en formato IEEE 754



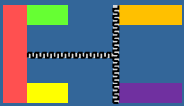


CONVERSIÓN DECIMAL-IEEE 754

Unidad
aritmética
flotante.
Sumar y
restar

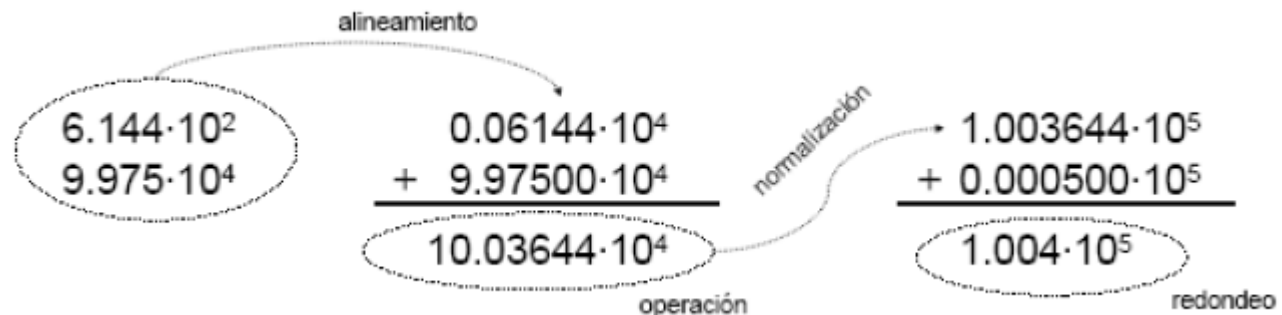
- ⊙ Ejemplo: -0.81375×10^2
 - ⊙ 1. Representar en coma fija el número decimal.
 - -81.375
 - ⊙ 2. Pasar número decimal a binario.
 - Parte entera: 81 $\rightarrow 1010001_2$
 - Parte decimal: 0.375 $\rightarrow 0.011_2$
 - 1010001.011_2
 - ⊙ 3. Normalizar mantisa.
 - 1.010001011×2^6
 - ⊙ 4. Normalizar exponente.
 - $E=6+127=133 \rightarrow 10000101_2$
 - ⊙ 5. Representar en formato IEEE 754

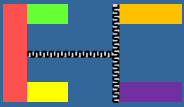
1 bit	8 bits	23 bits
1	10000101	010001011000.....00



Reglas de Suma/Resta

- 1. Seleccionar el número de menor exponente y desplazar su mantisa hacia la derecha tantas posiciones como la diferencia de los exponentes en valor absoluto.
- 2. Igualar el exponente del resultado al exponente mayor.
- 3. Operar las mantisas (según operación seleccionada y signos de ambos números) y obtener el resultado en signo y valor absoluto.
- 4. Normalizar el resultado y redondear la mantisa al número de bits apropiado.





LA SUMA Y LA RESTA

Unidad
aritmética
flotante.
Sumar y
restar

🎯 Selección de la operación real

SA	SB	Operación	Operación Real	Valores
0	0	Suma	$A+B$	0
0	1	Suma	$A-B$	Según resultado
1	0	Suma	$B-A$	Según resultado
1	1	Suma	$A+B$	1
0	0	Resta	$A-B$	Según resultado
0	1	Resta	$A+B$	0
1	0	Resta	$A+B$	1
1	1	Resta	$A-B$	Según resultado



```

graph TD
    RESTA([RESTA]) --> CambiarSigno[Cambiar signo de Y]
    SUMA([SUMA]) --> IsXZero{¿X = 0?}
    CambiarSigno --> IsXZero
    IsXZero -- Sí --> ZY[Z ← Y]
    IsXZero -- No --> IsYZero{¿Y = 0?}
    IsYZero -- Sí --> ZX[Z ← X]
    IsYZero -- No --> IsExponentsEqual{¿Exponentes iguales?}
    IsExponentsEqual -- Sí --> SumaSignificativas[Suma de partes significativas con signo]
    IsExponentsEqual -- No --> IncrementExponent[Incrementar el exponente más pequeño]
    SumaSignificativas --> IsPartSignificativaZero{¿Parte significativa = 0?}
    IsPartSignificativaZero -- Sí --> ZZero[Z ← 0]
    IsPartSignificativaZero -- No --> IsOverflowSignificativa{¿Desbordamiento en parte significativa?}
    IsOverflowSignificativa -- Sí --> DesplazarSignificativaDerecha[Desplazar parte significativa a la derecha]
    IsOverflowSignificativa -- No --> IsAgotamientoExponente{¿Agotamiento del exponente?}
    DesplazarSignificativaDerecha --> IncrementExponent2[Incrementar exponente]
    IncrementExponent2 --> IsOverflowExponente{¿Desbordamiento en exponente?}
    IsOverflowExponente -- Sí --> IndicarDesbordamiento[Indicar desbordamiento]
    IsOverflowExponente -- No --> IsResultosNormalizados{¿Resultados normalizados?}
    IndicarDesbordamiento --> RETORNAR1([RETORNAR])
    IsResultosNormalizados -- Sí --> Redondear[Redondear resultado]
    IsResultosNormalizados -- No --> DesplazarSignificativaIzquierda[Desplazar parte significativa a la izquierda]
    DesplazarSignificativaIzquierda --> DecrementExponent[Decrementar exponente]
    DecrementExponent --> IsAgotamientoExponente2{¿Agotamiento del exponente?}
    IsAgotamientoExponente2 -- Sí --> IndicarAgotamiento[Indicar agotamiento]
    IsAgotamientoExponente2 -- No --> IsPartSignificativaZero2{¿Parte significativa = 0?}
    IndicarAgotamiento --> RETORNAR2([RETORNAR])
    IsPartSignificativaZero2 -- Sí --> PonerOtroNumero[Poner otro número en Z]
    IsPartSignificativaZero2 -- No --> RETORNAR1
    PonerOtroNumero --> RETORNAR1
    ZZero --> RETORNAR1
    ZY --> RETORNAR1
    ZX --> RETORNAR1
    
```

El diagrama de flujo describe el algoritmo para la suma y resta de números en coma flotante. Comienza con la operación (SUMA o RESTA) y el cambio de signo de Y si es resta. Se verifica si X o Y son cero; si lo son, se copia el otro valor a Z y se retorna. Si ambos tienen exponentes iguales, se suman o restan las partes significativas. Si el resultado es cero, se retorna. Si hay un desbordamiento en la parte significativa, se desplaza a la derecha y se incrementa el exponente. Si hay un desbordamiento en el exponente, se indica desbordamiento. Si los resultados no están normalizados, se desplaza a la izquierda y se decrementa el exponente. Si hay un agotamiento del exponente, se indica agotamiento. Si la parte significativa es cero, se pone otro número en Z y se retorna. Si no, se retorna directamente.





CIRCUITO SUMADOR/RESTADOR

Unidad
aritmética
flotante.
Sumar y
restar

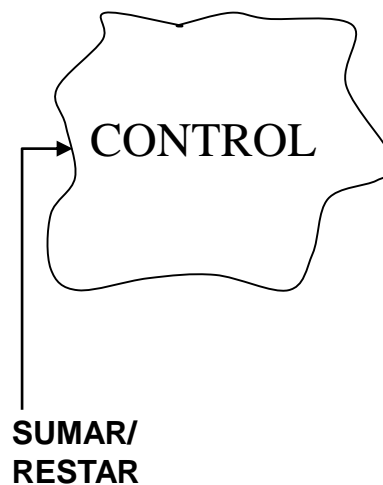
- Tenemos 2 números A y B
- Queremos sumar/restar
 - $R = A + B$
 - $R = A - B$

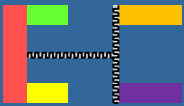
S_A	E_A	M_A
-------	-------	-------

S_B	E_B	M_B
-------	-------	-------

S_R	E_R	M_R
-------	-------	-------

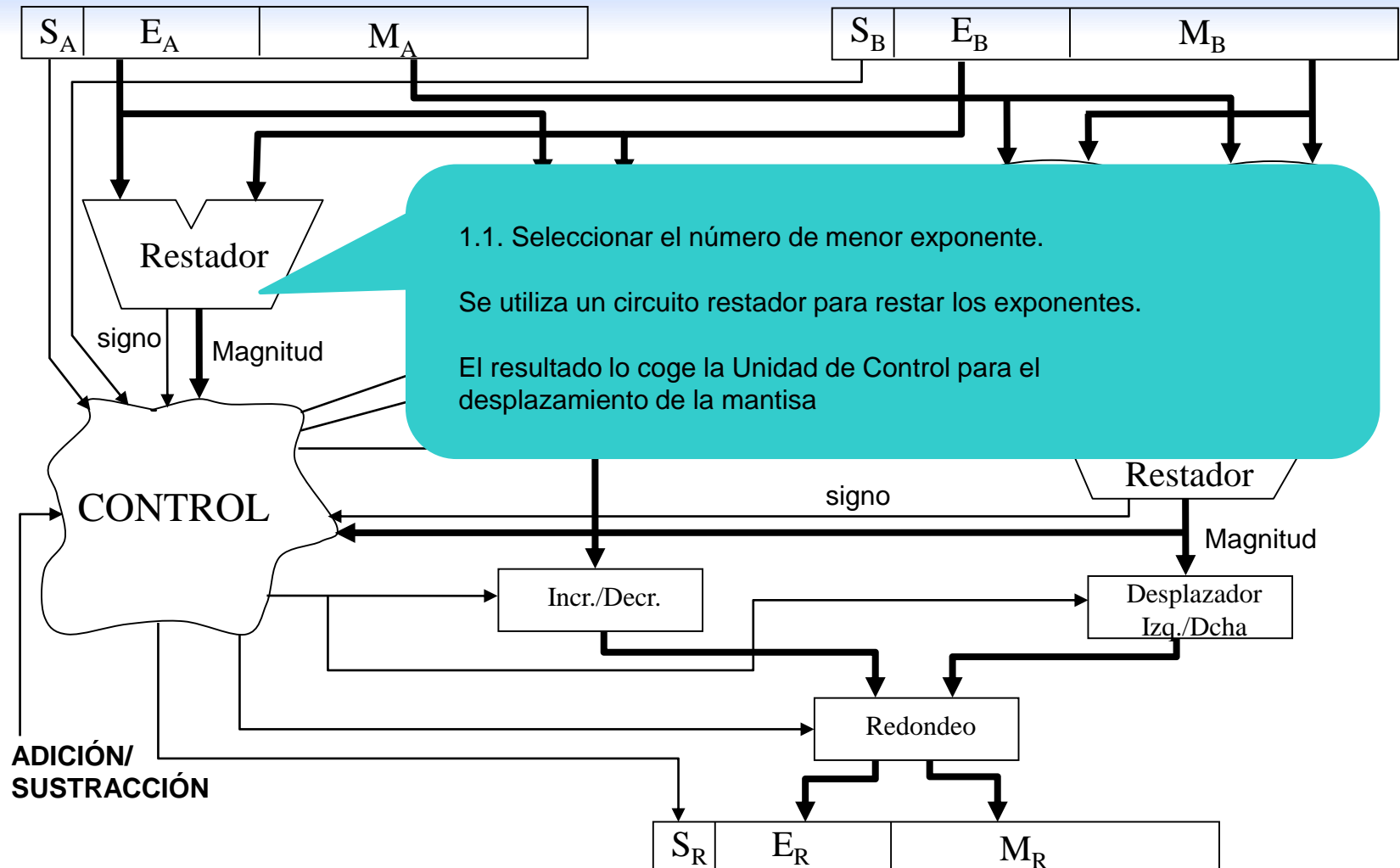
- La unidad de control se encarga de realizar la operación (SUMAR/RESTAR) Activando sus señales de salida

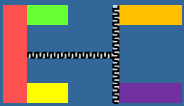




CIRCUITO SUMADOR/RESTADOR

Unidad
aritmética
flotante.
Sumar y
restar





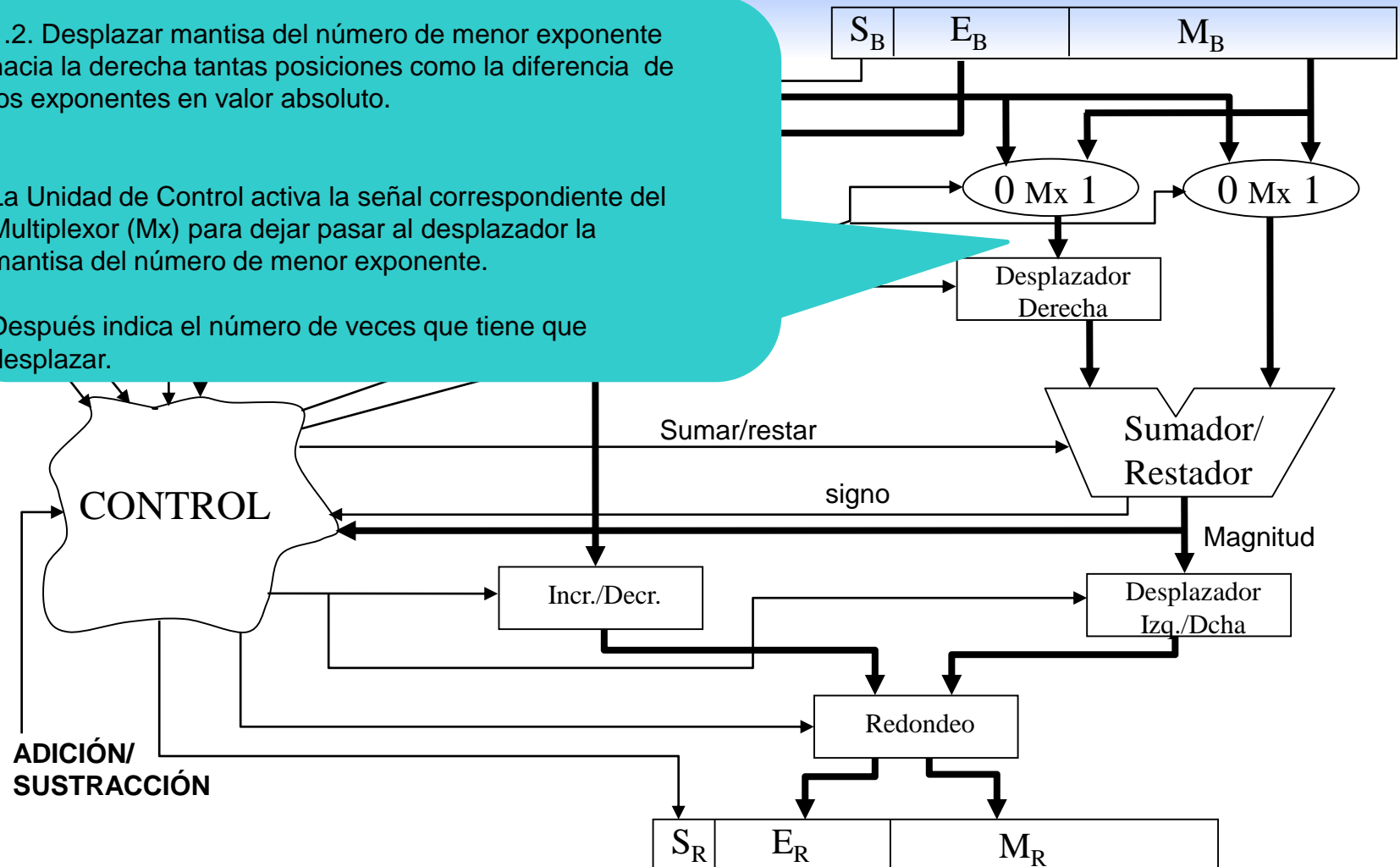
CIRCUITO SUMADOR/RESTADOR

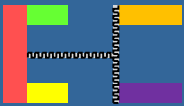
Unidad
aritmética
flotante.
Sumar y
restar

1.2. Desplazar mantisa del número de menor exponente hacia la derecha tantas posiciones como la diferencia de los exponentes en valor absoluto.

La Unidad de Control activa la señal correspondiente del Multiplexor (Mx) para dejar pasar al desplazador la mantisa del número de menor exponente.

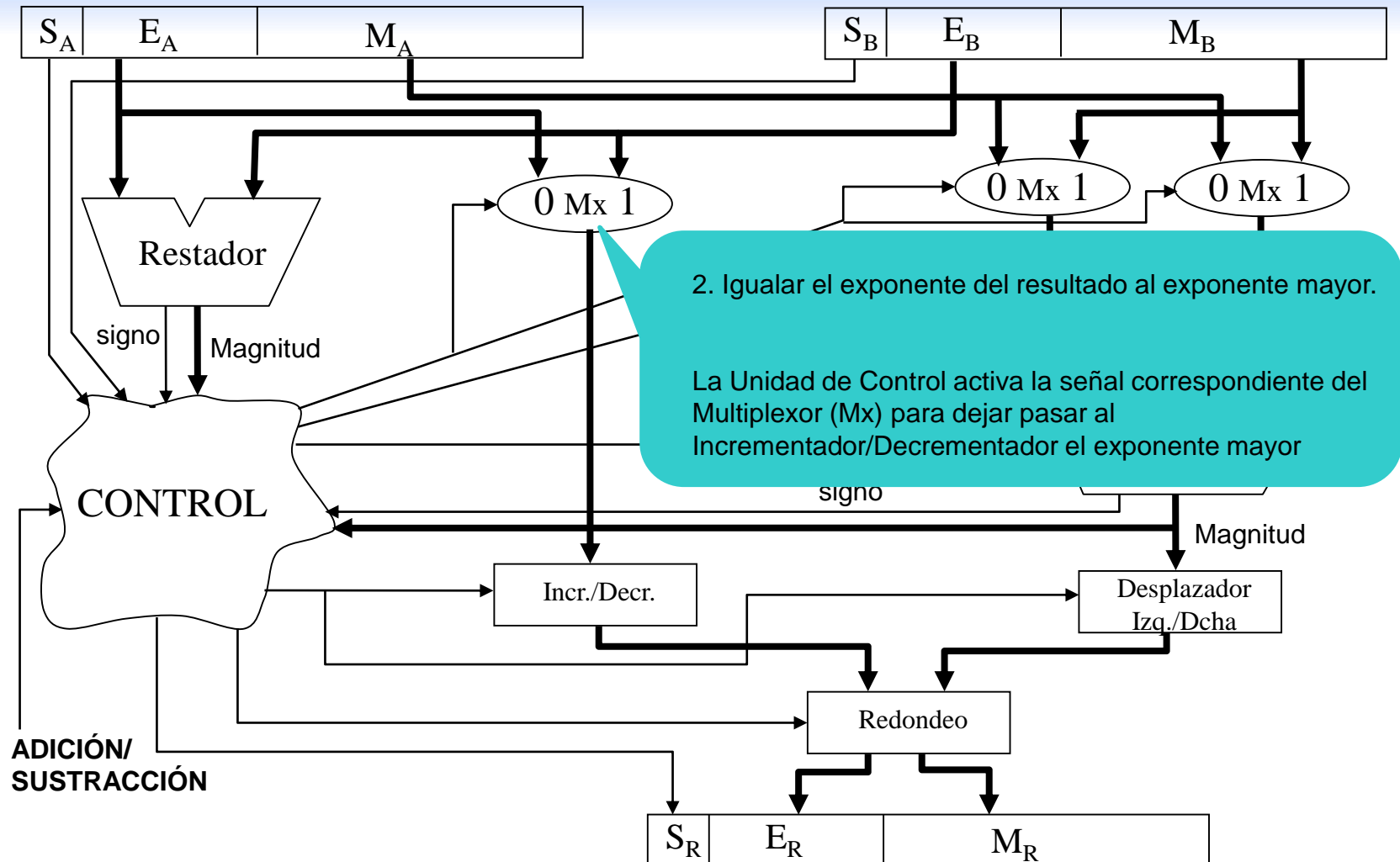
Después indica el número de veces que tiene que desplazar.

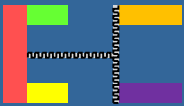




CIRCUITO SUMADOR/RESTADOR

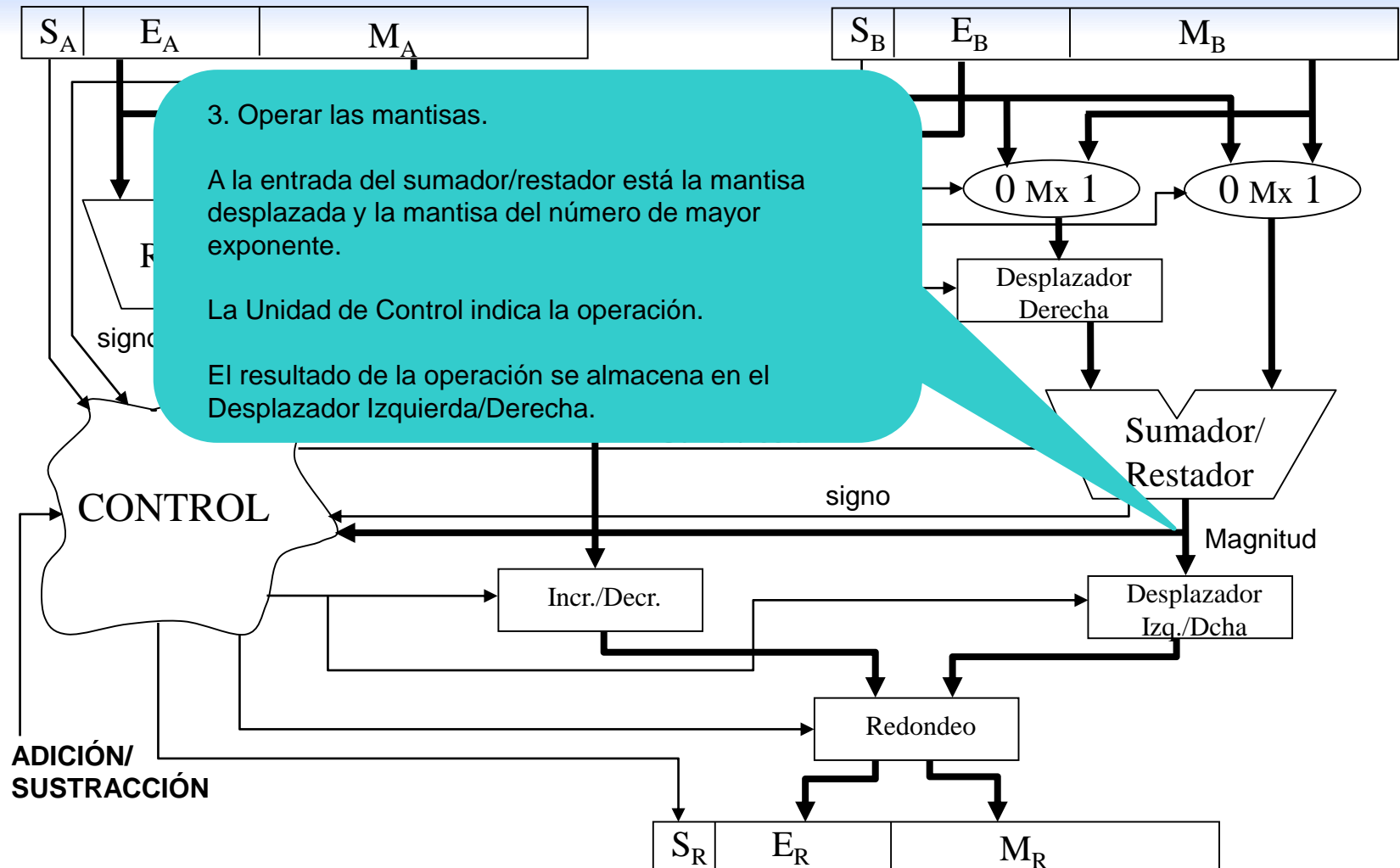
Unidad
aritmética
flotante.
Sumar y
restar

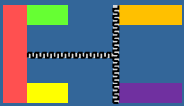




CIRCUITO SUMADOR/RESTADOR

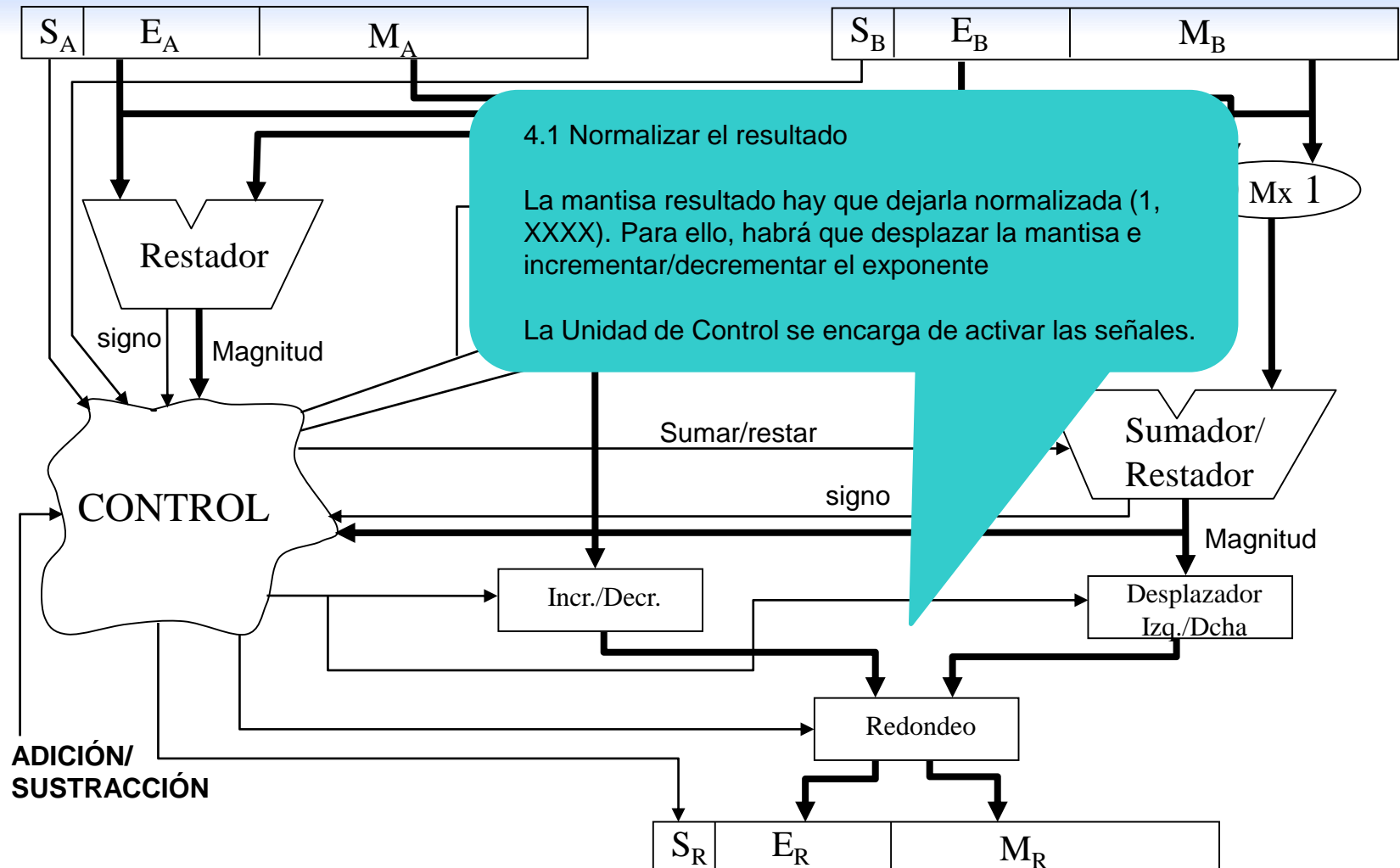
Unidad
aritmética
flotante.
Sumar y
restar

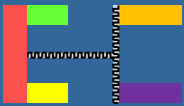




CIRCUITO SUMADOR/RESTADOR

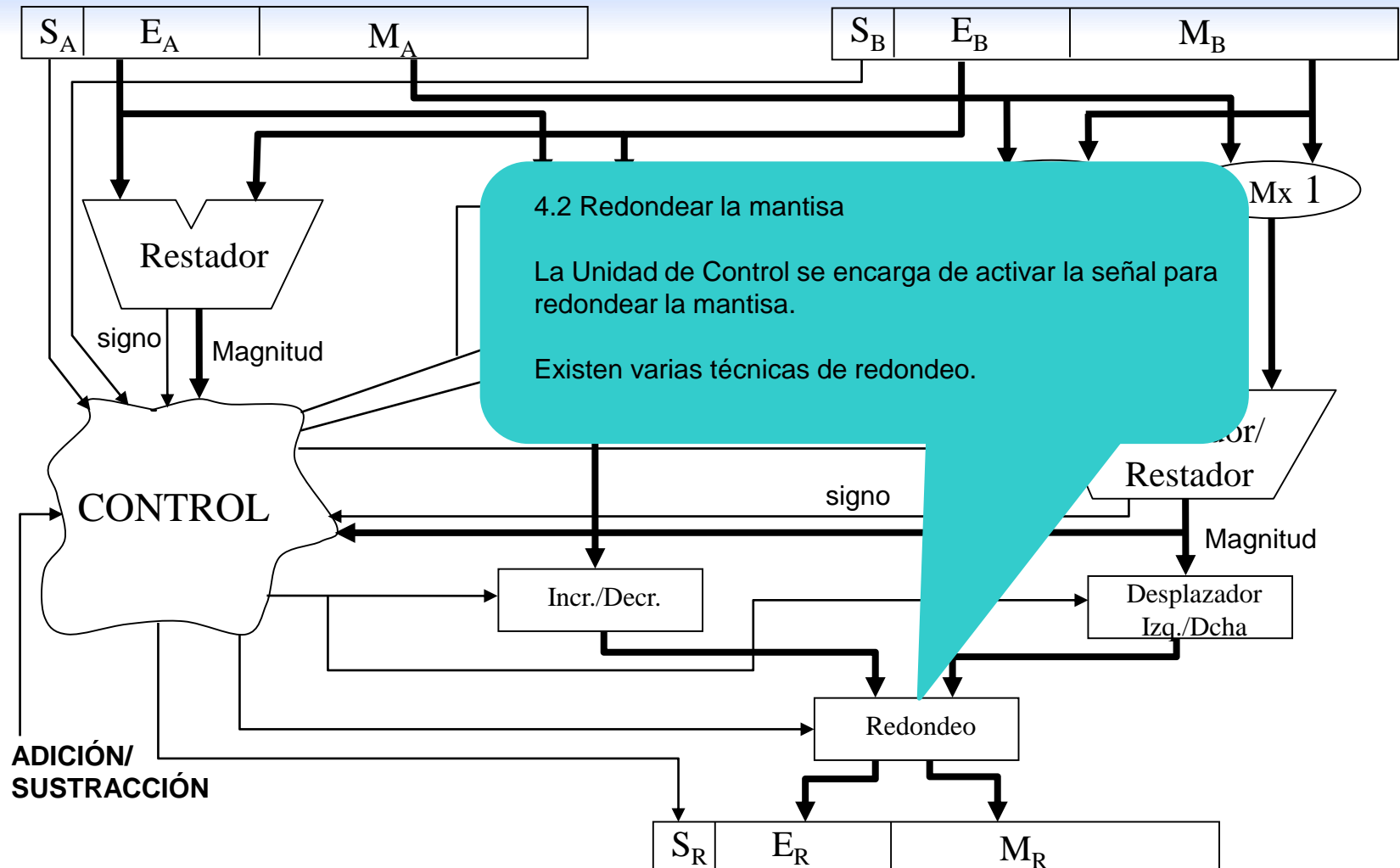
Unidad
aritmética
flotante.
Sumar y
restar





CIRCUITO SUMADOR/RESTADOR

Unidad
aritmética
flotante.
Sumar y
restar





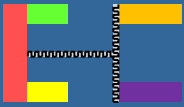
LA MULTIPLICACIÓN EN COMA FLOTANTE

Unidad
aritmética
flotante.
Multiplicar y
dividir

⊙ Reglas de Multiplicación

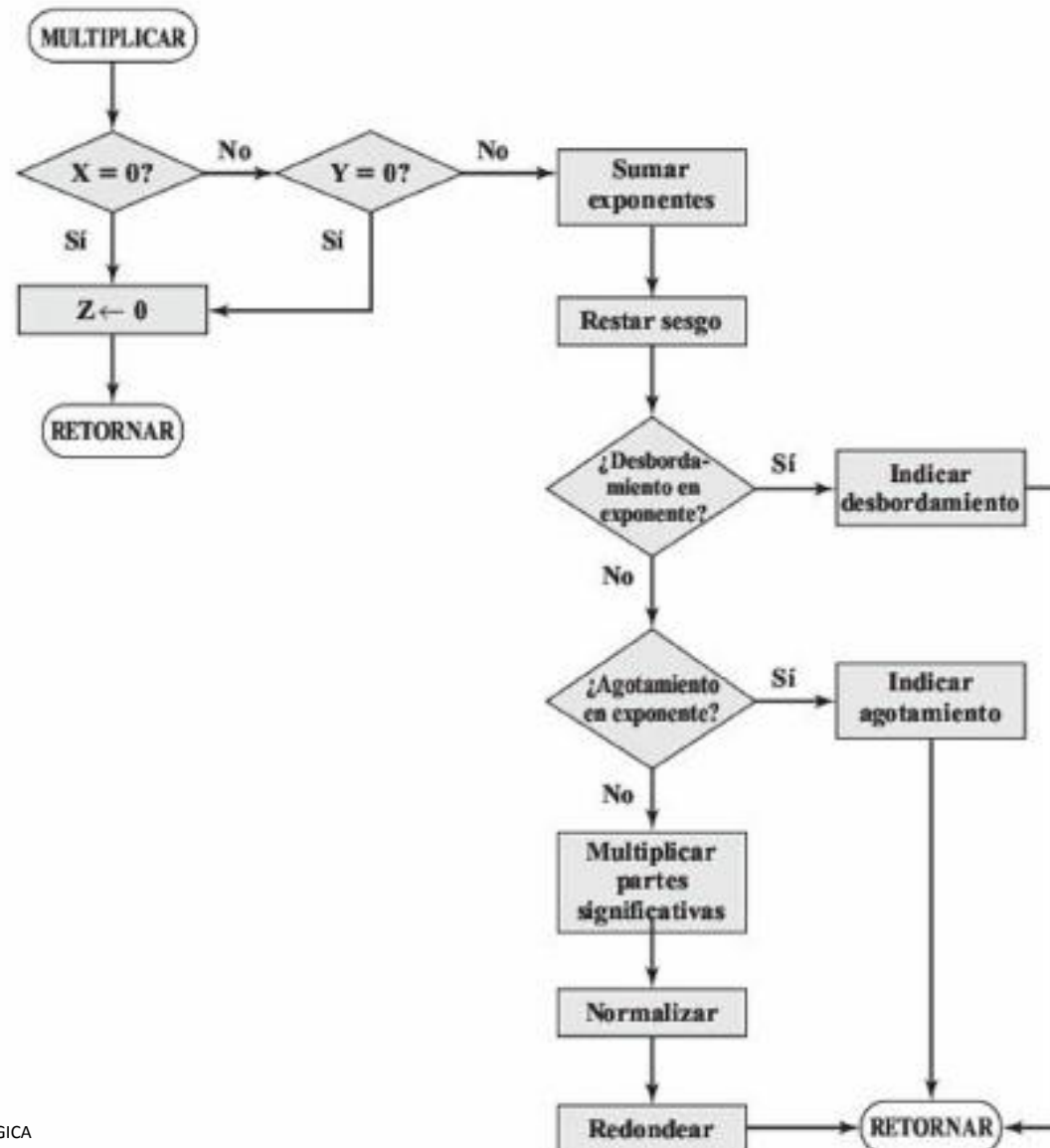
- ⊙ 1. Sumar los exponentes y restar el exceso para obtener el exponente del resultado
- ⊙ 2. Multiplicar las mantisas para determinar la mantisa del resultado
- ⊙ 3. Procesar los signos
- ⊙ 4. Normaliza y redondear si es necesario

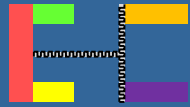
Resultado $\rightarrow (s1 \oplus s2) (1.MANT1 * 1.MANT2) \times 2^{(e1 + e2 - \text{sesgo})}$



LA MULTIPLICACIÓN EN COMA FLOTANTE

Unidad
aritmética
flotante.
Multiplicar y
dividir





LA DIVISIÓN EN COMA FLOTANTE

Unidad
aritmética
flotante.
Multiplicar y
dividir

- ⊙ Reglas de División
 - 1. Restar los exponentes y sumar el exceso para obtener el exponente resultado
 - 2. Dividir las mantisas para determinar la mantisa del resultado.
 - 3. Procesar los signos.
 - 4. Normalizar y redondear si es necesario.

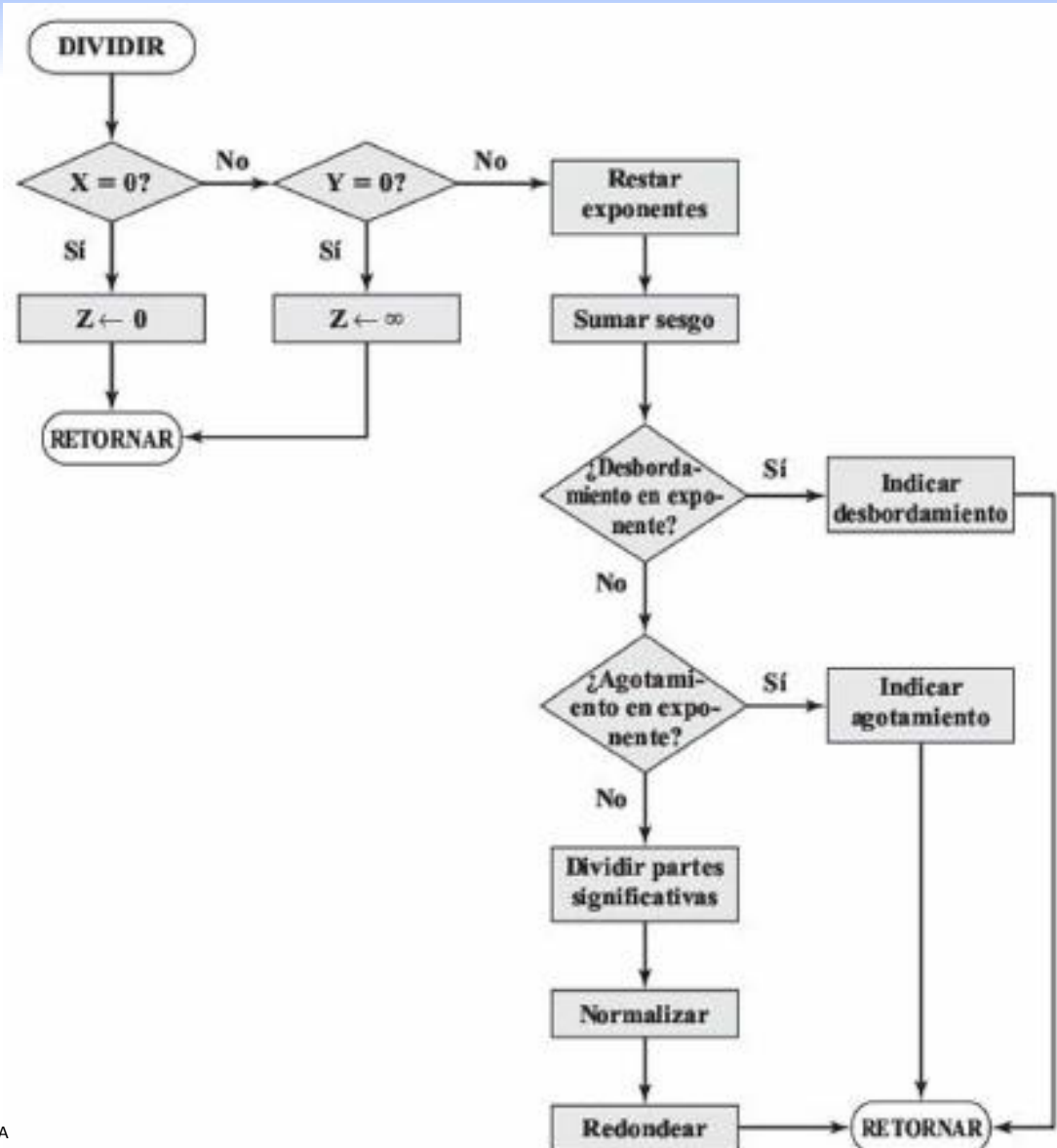
Resultado $\rightarrow (s_1 \oplus s_2) (1.MANT1 / 1.MANT2) \times 2^{(e_1 - e_2 + \text{sesgo})}$

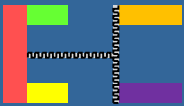
- ⊙ Procesamiento de los signos

S_A	S_B	S_R
0	0	0
0	1	1
1	0	1
1	1	0
$S_R = S_A \oplus S_B$		

LA DIVISIÓN EN COMA FLOTANTE

Unidad
aritmética
flotante.
Multiplicar y
dividir





CIRCUITO SUMADOR/RESTADOR

Unidad
aritmética
flotante.
Sumar y
restar

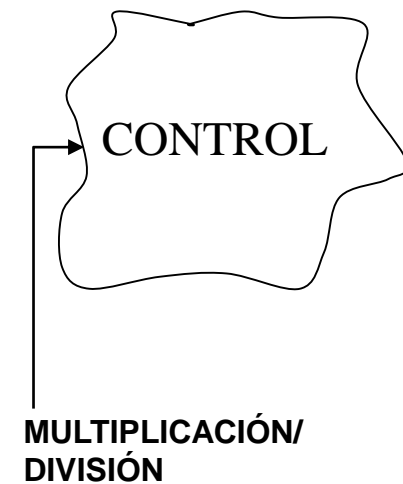
- Tenemos 2 números A y B
- Queremos multiplicar/dividir
 - $R = A * B$
 - $R = A / B$

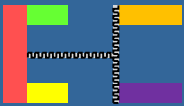
S_A	E_A	M_A
-------	-------	-------

S_B	E_B	M_B
-------	-------	-------

S_R	E_R	M_R
-------	-------	-------

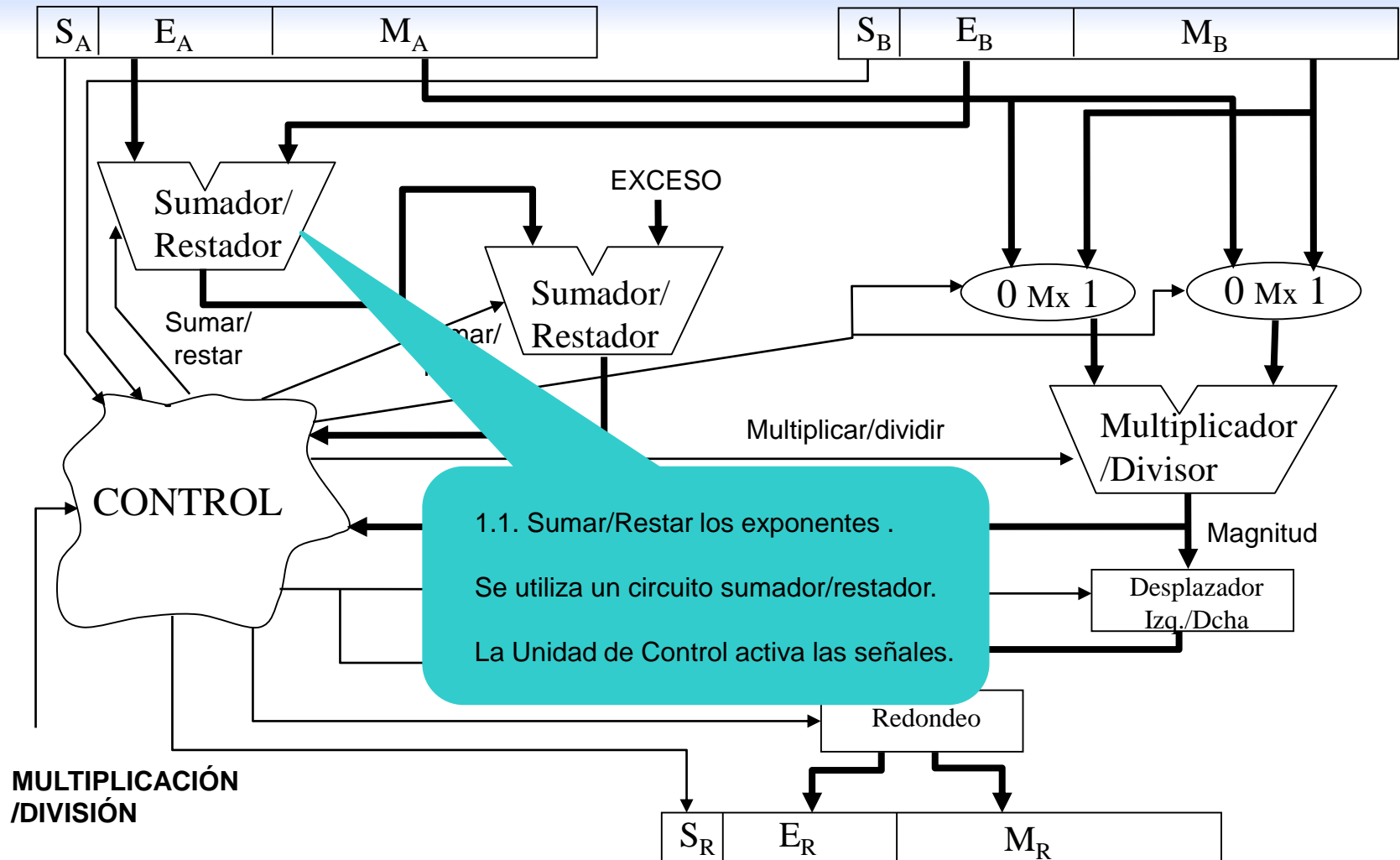
- La unidad de control se encarga de realizar la operación (MULTIPLICAR/DIVIDIR) Activando sus señales de salida





CIRCUITO MULTIPLICADOR/DIVISOR

Unidad
aritmética
flotante.
Multiplicar y
dividir



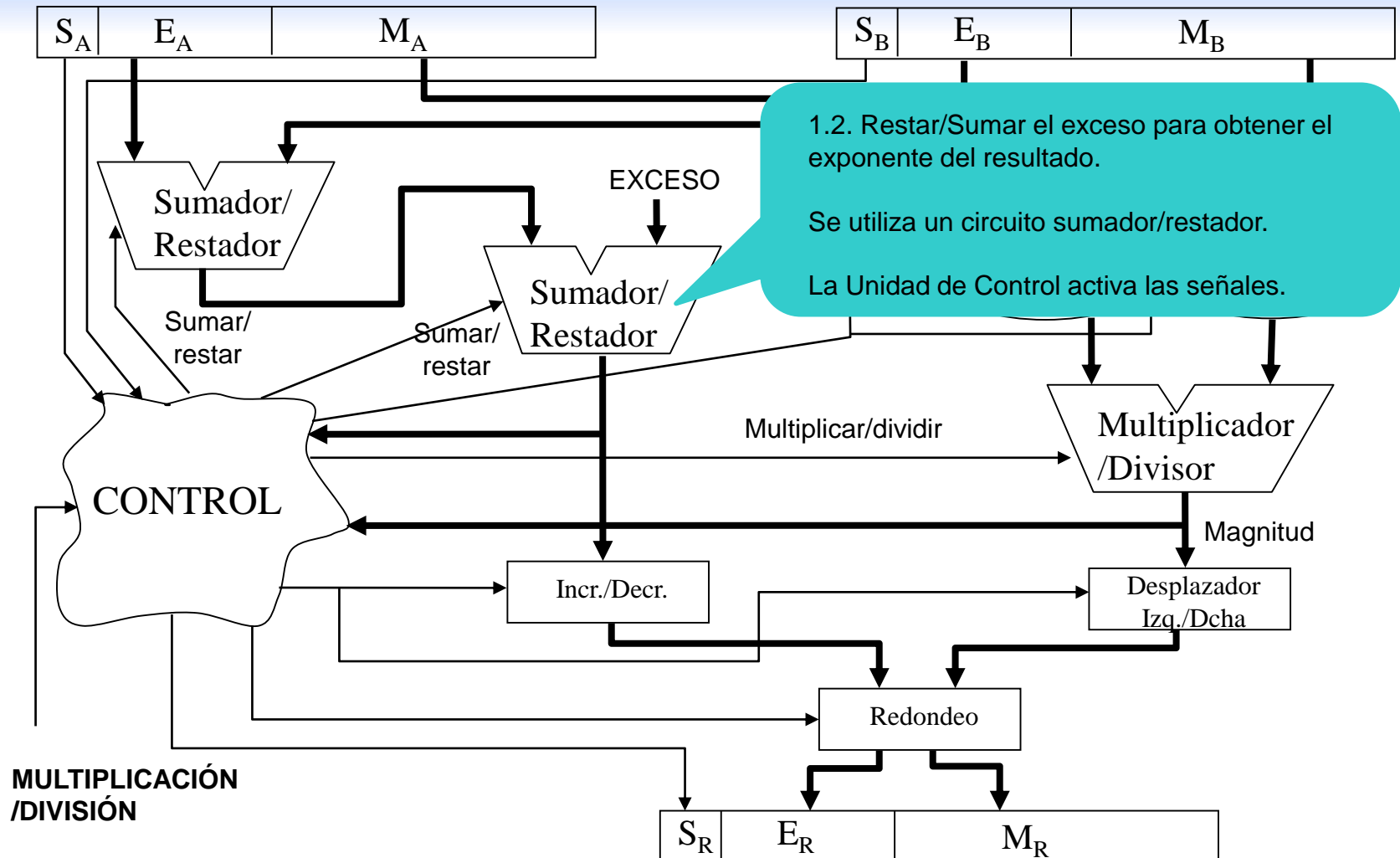
1.1. Sumar/Restar los exponentes .

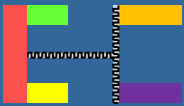
Se utiliza un circuito sumador/restador.

La Unidad de Control activa las señales.

CIRCUITO MULTIPLICADOR/DIVISOR

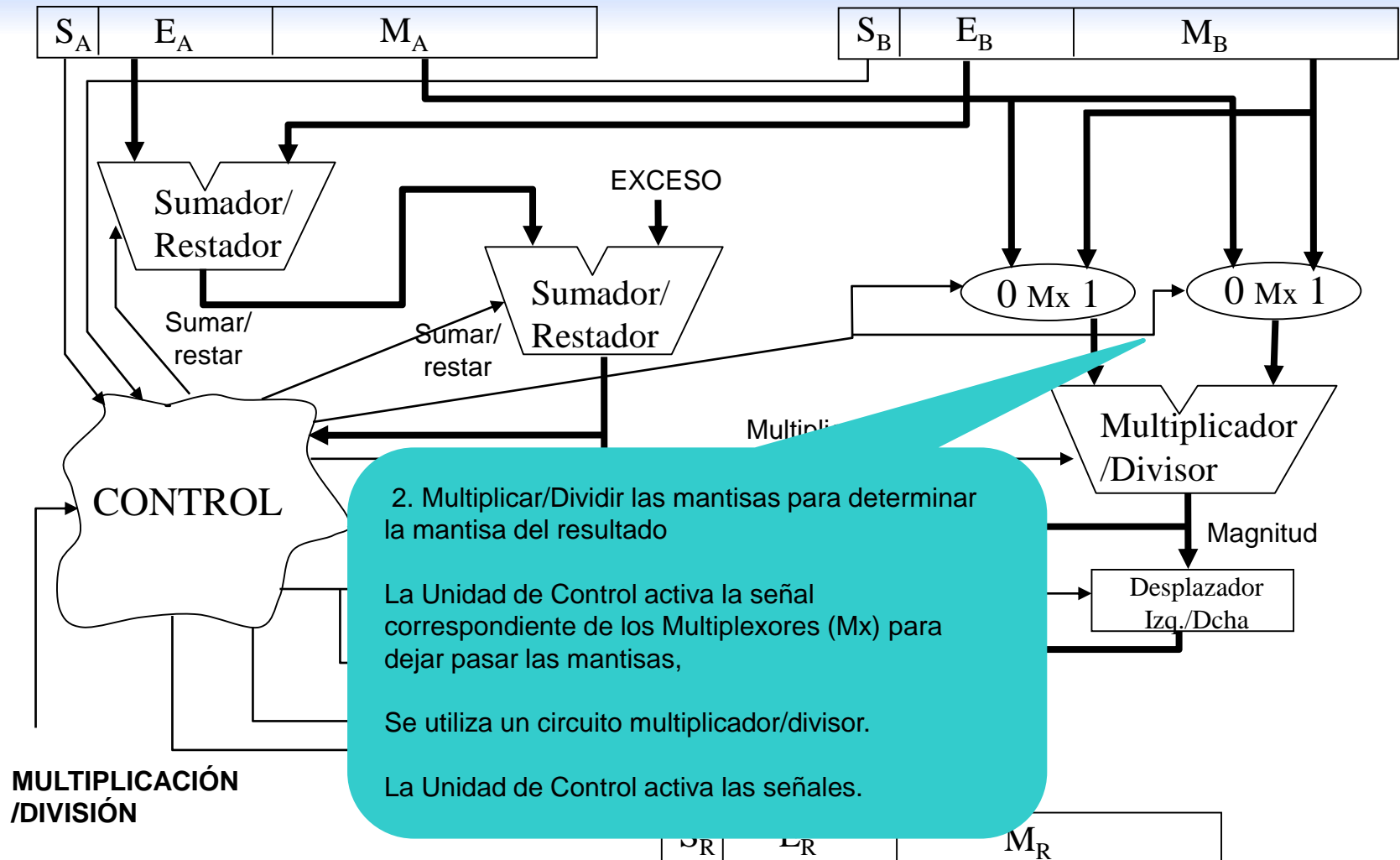
Unidad
aritmética
flotante.
Multiplicar y
dividir

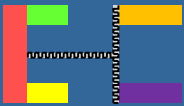




CIRCUITO MULTIPLICADOR/DIVISOR

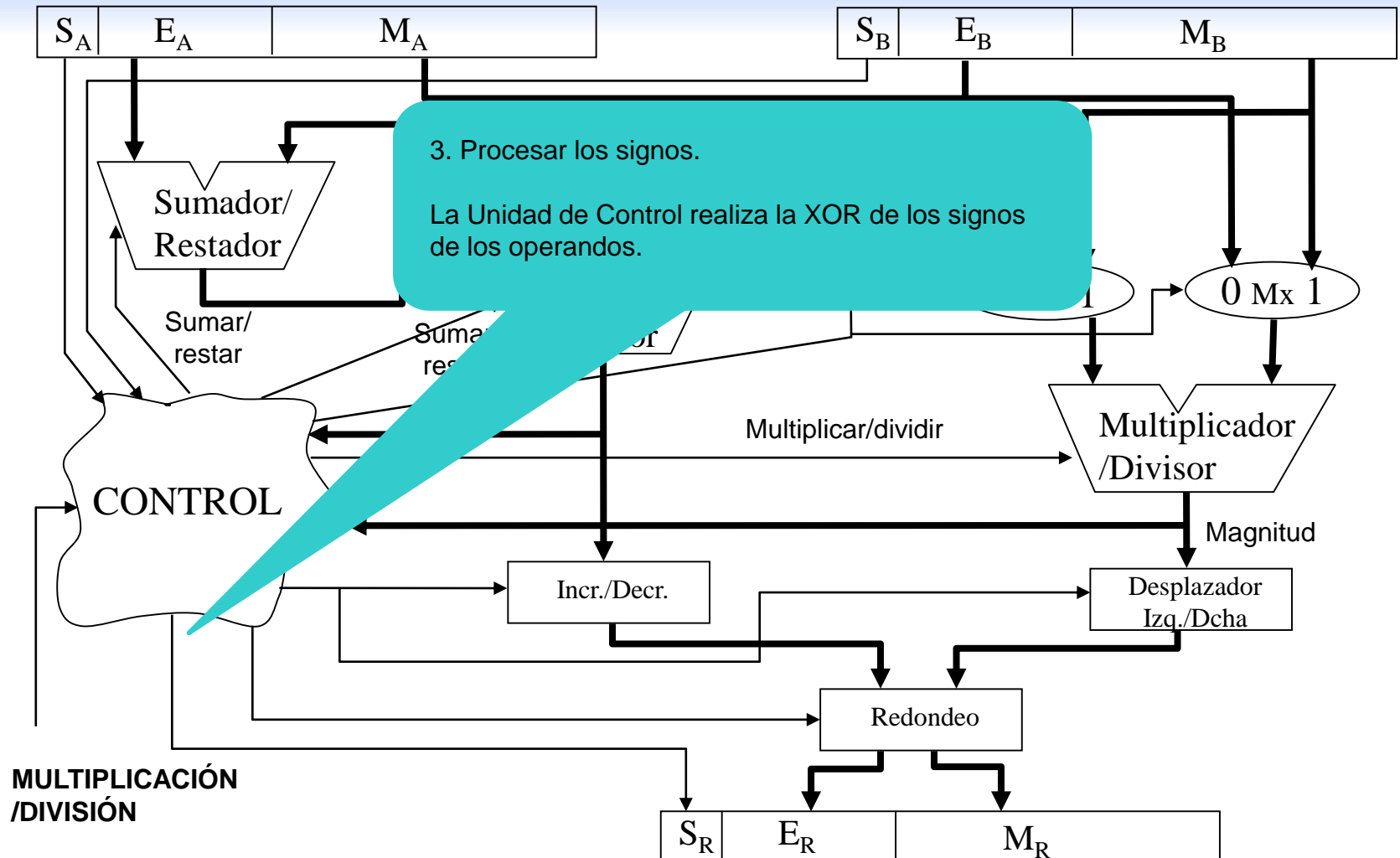
Unidad
aritmética
flotante.
Multiplicar y
dividir





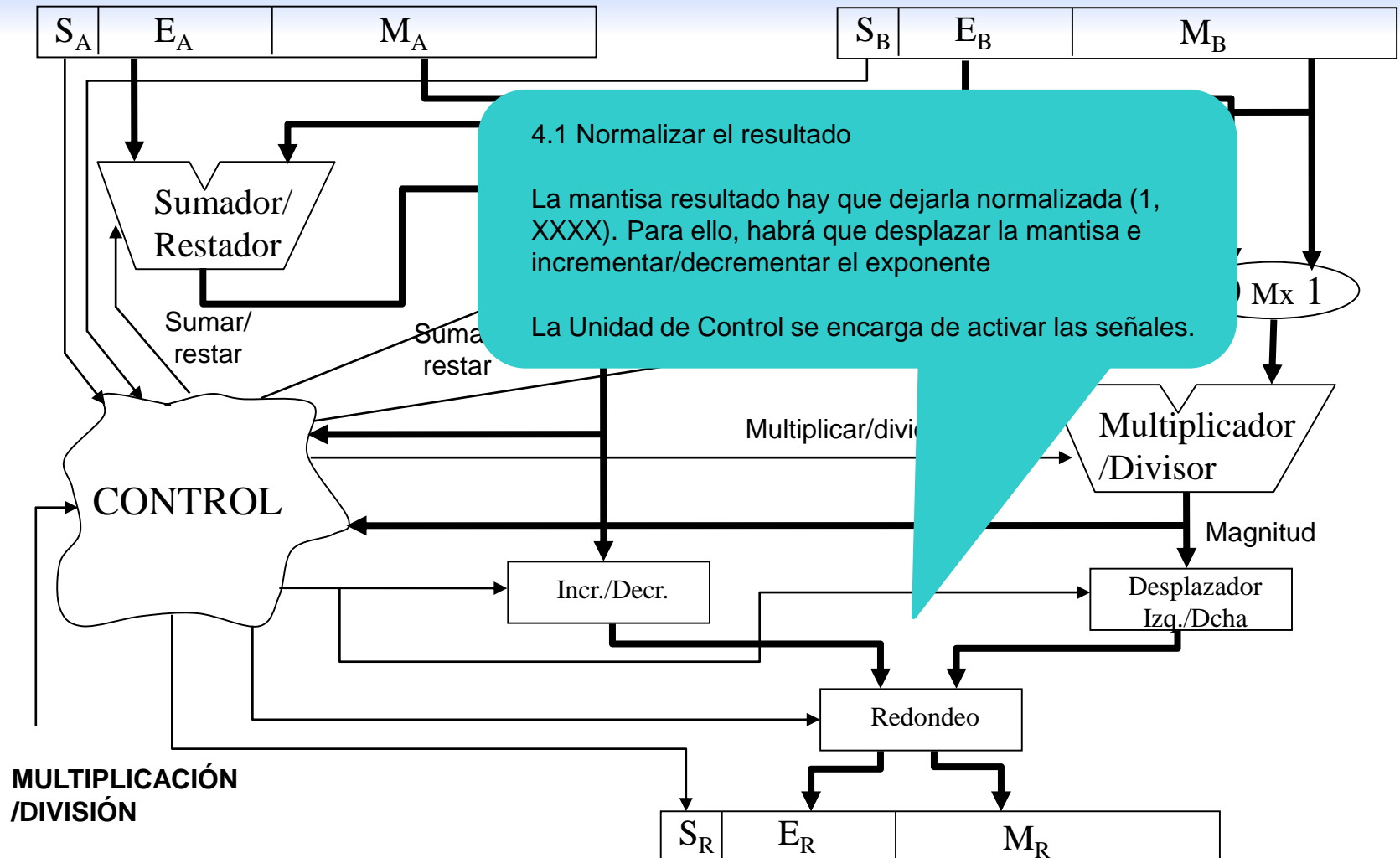
CIRCUITO MULTIPLICADOR/DIVISOR

Unidad
aritmética
flotante.
Multiplicar y
dividir



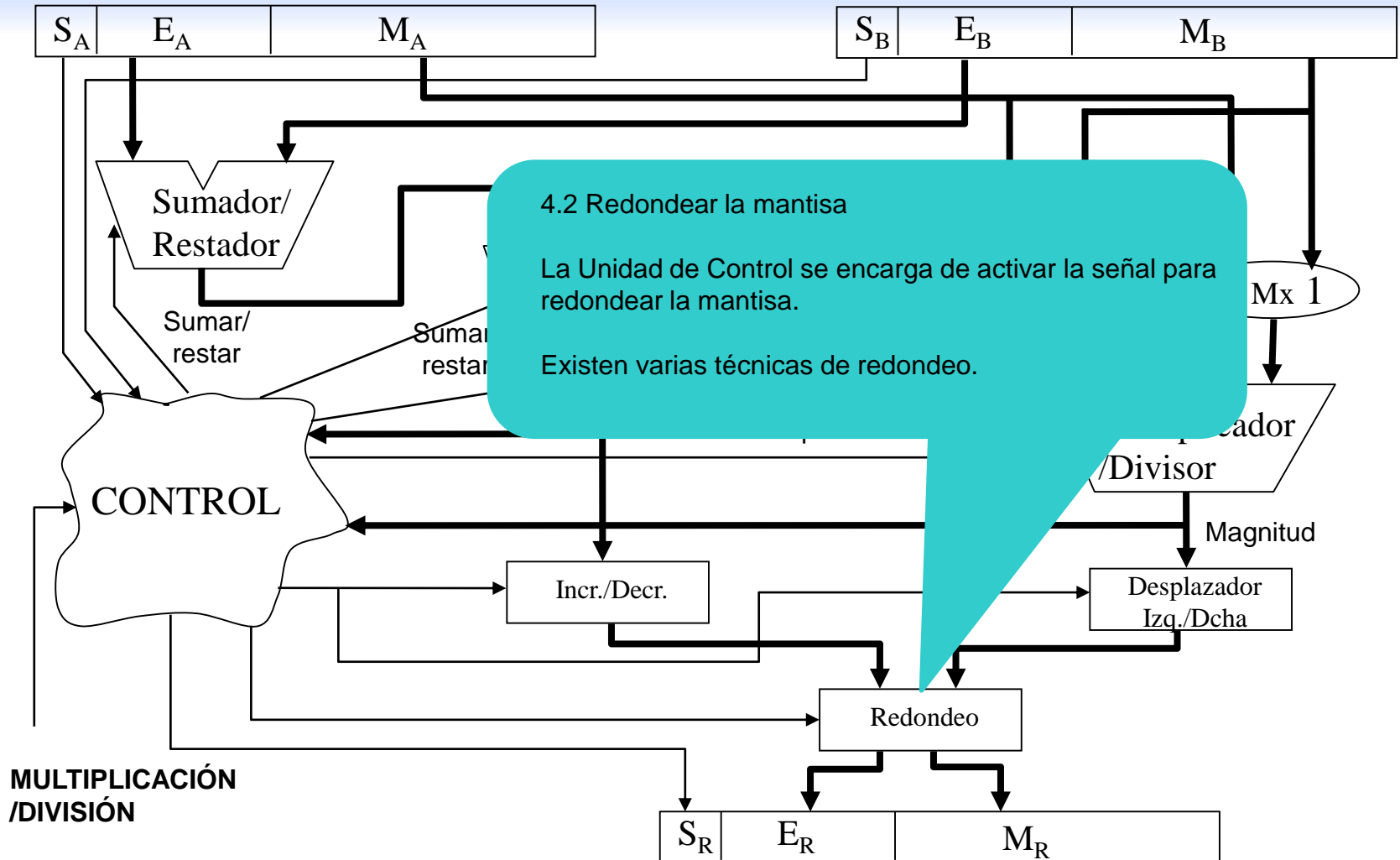
CIRCUITO MULTIPLICADOR/DIVISOR

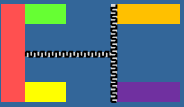
Unidad
aritmética
flotante.
Multiplicar y
dividir



CIRCUITO MULTIPLICADOR/DIVISOR

Unidad
aritmética
flotante.
Multiplicar y
dividir





TÉCNICAS DE REDONDEO

Unidad
aritmética
flotante.
Técnicas de
redondeo

- ⊙ Las técnicas de redondeo consisten en limitar el número de bits al disponible en el sistema de representación utilizado.
- ⊙ Normalmente las ALU utilizan **bits de guarda** que luego deben eliminar.
- ⊙ Dada una cantidad **C**, y un sistema de representación que permite representar los valores **$V_0, V_1, \dots V_r$** .
- ⊙ El redondeo consiste en asignar a **C** una representación **R** que se le aproxime. Si $V_{i-1} < C < V_i$ el redondeo consiste en asignar **V_{i-1}** o **V_i** como representación R de la cantidad C
- ⊙ El error absoluto se define como: **$\epsilon = |R - C|$**
- ⊙ Técnicas de redondeo
 - ⊙ Truncamiento
 - ⊙ Redondeo propiamente dicho
 - ⊙ Bit menos significativo forzado a “uno”

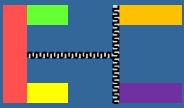


TÉCNICAS DE REDONDEO: TRUNCAMIENTO

Unidad
aritmética
flotante.
Técnicas de
redondeo

- ⊙ Elimina los bits a la derecha que no caben en la representación.
- ⊙ Es fácil de implementar.
- ⊙ El error del resultado es siempre por defecto.
- ⊙ El error puede crecer rápidamente (p. ej. en operaciones consecutivas)





TÉCNICAS DE REDONDEO: REDONDEO AL MÁS PRÓXIMO

Unidad
aritmética
flotante.

Técnicas de
redondeo

- El resultado de redondea al **número representable** más próximo:

$$\text{Si } |V_{i-1} - C| < |V_i - C| \rightarrow R \equiv V_{i-1} \text{ si no } V_i$$

Ejemplo: representación con 8 bits de punto implícito

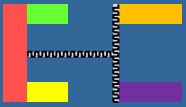
$$C = 0,01100000\ 01 \equiv 0,375976563$$

$$V_{i-1} = 0,01100000 \quad \equiv 0,375 \quad (\text{número representable anterior})$$

$$V_i = 0,01100001 \quad \equiv 0,37890625 \quad (\text{número representable posterior})$$

$$|V_{i-1} - C| = 0,000976563 \quad \longrightarrow \quad R = 0,01100000 \quad (\text{más próximo})$$

$$|V_i - C| = 0,00390625$$

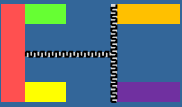


TÉCNICAS DE REDONDEO: BIT MENOS SIGNIFICATIVO FORZADO A 1

Unidad
aritmética
flotante.
Técnicas de
redondeo

- ⊙ Consiste en truncar y forzar el bit menos significativo a “uno”
- ⊙ Es muy rápido, tanto como el truncamiento
- ⊙ Sus errores son tanto por defecto como por exceso.





EJERCICIO 1

Ejercicios

- Sumar los números A y B según el estándar IEEE 754 en simple precisión teniendo en cuenta las tres técnicas de redondeo.

A

0	01111111	000000000000.....00
---	----------	---------------------

B

0	01100110	000000000000.....00
---	----------	---------------------



EJERCICIO 2

Ejercicios

- Obtener el resultado de la operación $A+B$ y $A \times B$ en el formato IEEE 754 de los siguientes números representados en este formato. Para obtener el resultado especificar los pasos seguidos utilizando el algoritmo de suma y multiplicación estudiado para números representados en el IEEE 754. Expresar el resultado en hexadecimal.

$A = C1340000$

$B = 3F980000$