TEMA 3. UNIDAD DE CÁLCULO DE UN COMPUTADOR

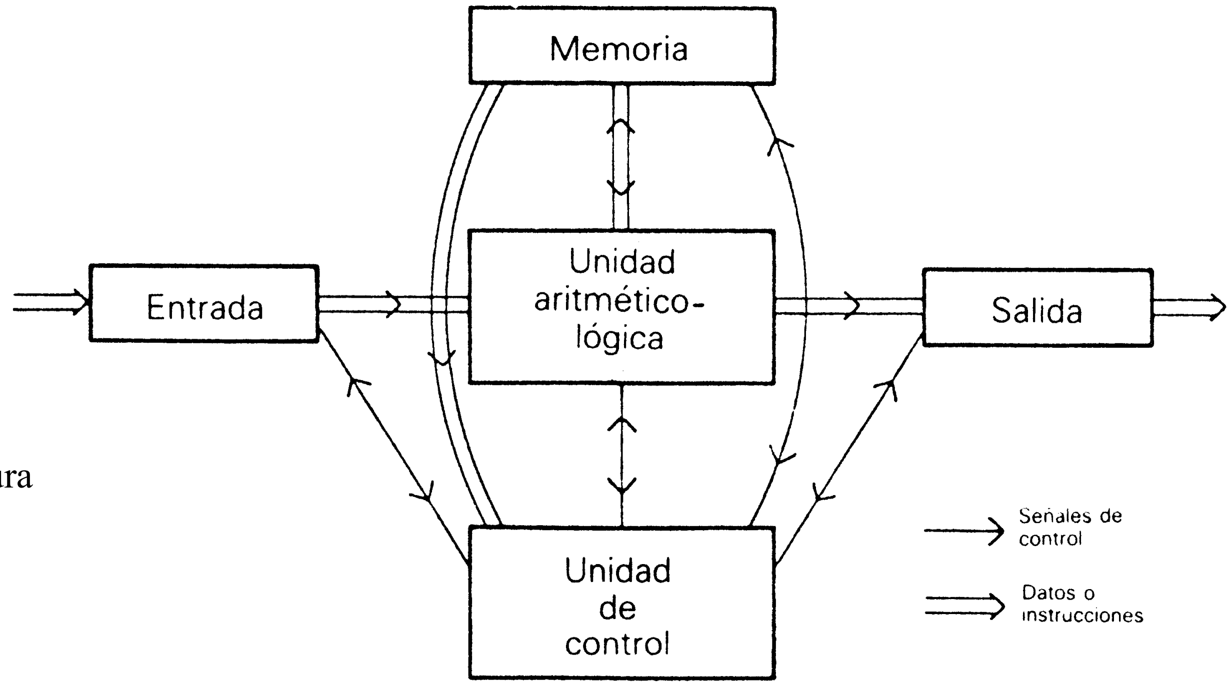
INTRODUCCION

* Objetivos

- Analizar el Hardware mínimo para esta Unidad
- Estudio de los Algoritmos Aritméticos

* Visión global

- Componentes básicos de la Estructura VON NEUMAN
  - Unidad de Cálculo
  - Unidad de Control
  - Unidad de Memoria
  - Unidades de E/S



Esquema de Von Neumann.

*** Operaciones a realizar**

- Comparación
- Suma
- Resta
- Multiplicación
- División

*** Tipos de Operandos**

- Enteros sin signo
- Enteros con signo
  - Signo Magnitud
  - Complemento a dos
  - Exceso M
- Reales
  - Punto flotante

*** Concepto de Algoritmo**

"Secuencia de operaciones cuya ejecución implica la realización de una operación determinada"
- Secuencialidad
- Existencia de bucles
- Saltos condicionales
- Condicionalidad de ejecución

*** Expresión de las operaciones**

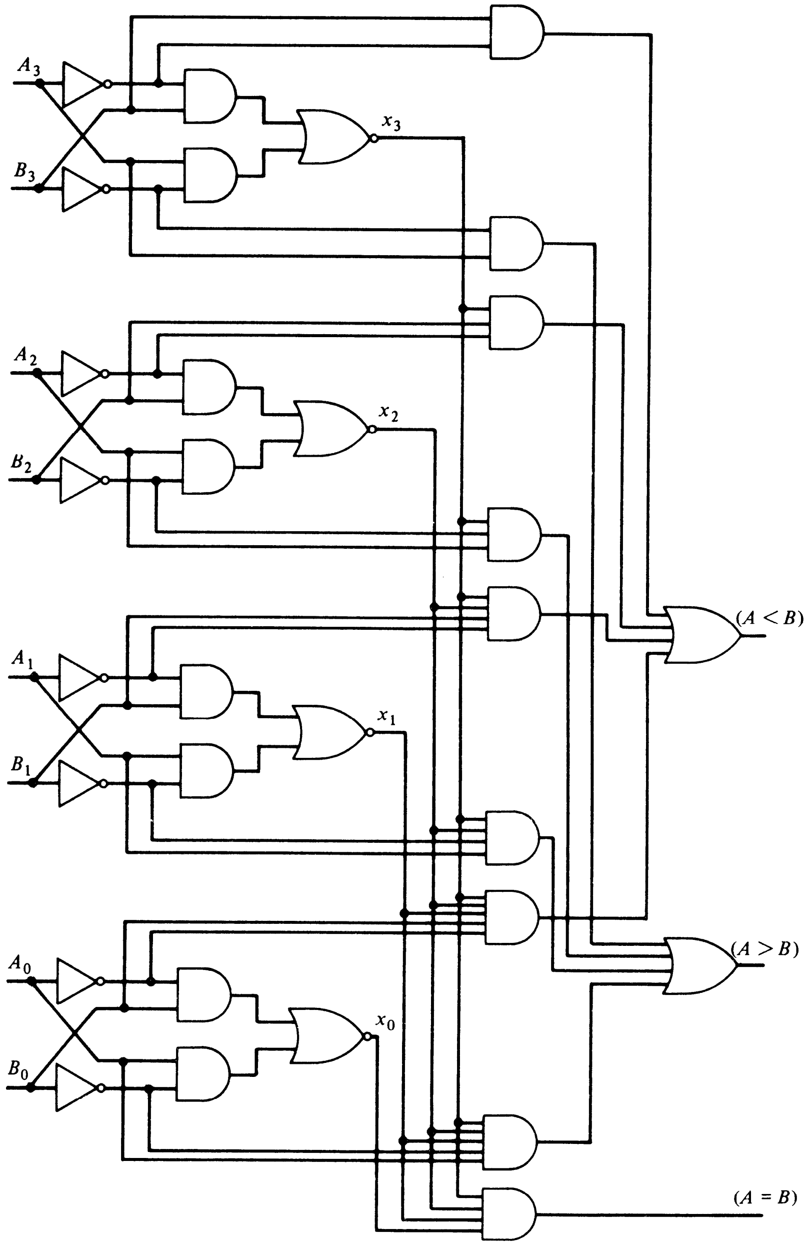
- Hardware necesario
  - Registros (almacenamiento, desplazamiento, etc.)
  - Circuitos Combinaciones (sumador, etc.)
  - Lógica adicional
- Algoritmo Aritmético (¿es necesario siempre?)
  - Secuencia de acciones que conllevan la realización de esa operación aritmética
  - Diagrama de flujo clásico
- Interdependencia entre el Hardware y el Algoritmo
  - Ejemplo: el producto

COMPARACIÓN Y RESTA DE NUMEROS BINARIOS SIN SIGNO

* Importancia de la Comparación

- como operación propia
- como toma de decisiones en otras operaciones

* Comparación de dos números binarios A y B (c. combinacional)



($X_i = A_iB_i + A_i'B_i'$)

- Igual A = B ----> si $X_i = 1$; i = 0,1,2,
- Mayor A > B -----> si $A_2B_2'+X_2A_1B_1'+X_2X_1A_0B_0' = 1$
- Menor A < B -----> si $A_2'B_2+X_2A_1'B_1+X_2X_1A_0'B_0 = 1$

*** Representación de los números en complemento a dos**

- Complemento a dos de A = $2_A = 2^n - A$

*** Resta de dos números A y B**

A - B
- Si A > B ---> prest = 0 resultado A - B
- Si A = B ---> prest = 0 " 0
- Si A < B ---> prest = 1 " $2_{(B-A)}$

$A < B \text{-->} 2^n + A - B = 2^n - (B - A) = 2_{(B-A)}$

*** Comparación mediante la resta**

- p = 0 implica A >= B p = 1 implica A < B
- Para distinguir A > B de A = B se usa el Bit Z
- Z = 1 implica que A = B

RESTA DE NUMEROS MEDIANTE COMPLEMENTO

*** Complemento a 1** ($1_B = B' = 2^n - 1 - B$)

$A - B = A + B' = A + 2^n - 1 - B$

- $A > B$ --> $c = 1$ resultado $A - B - 1$
- $A = B$ --> $c = 0$ " $2^n - 1$
- $A < B$ --> $c = 0$ " 1_{B-A}

*** Comparación mediante la resta con complemento a uno**

- $c = 0$ implica $A <= B$ $c = 1$ implica $A > B$
- Para distinguir $A < B$ de $A = B$ se usa el Bit Z
- $Z = 1$ implica que $A = B$

*** Complemento a 2** ($2_B = B' + 1 = 2^n - B$)

$A - B = A + B' + 1 = A + 2^n - B$

- $A > B$ --> $c = 1$ resultado $A - B$
- $A = B$ --> $c = 1$ " 0
- $A < B$ --> $c = 0$ " 2_{B-A}

*** Comparación mediante la resta con complemento a dos**

- $c = 1$ implica $A >= B$; $c = 0$ implica $A < B$
- Para distinguir $A < B$ de $A = B$ se usa el Bit Z
- $Z = 1$ implica que $A = B$

TABLA 9-1 Resta y comparación de números binarios sin signo

Operación	*si*	*E es*	*El resultado es*	*si*	*E es*	*El resultado es*	*Si $A = B$ el resultado contiene*
$A - B$	$A \geq B$	0	$A - B$	$A < B$	1	Complemento a 2 de $(B - A)$	Todos 0's
$A + \bar{B}$	$A \leq B$	0	Complemento a 1 de $(B - A)$	$A > B$	1	$A - B - 1$	Todos 1's
$A + \bar{B} + 1$	$A < B$	0	Complemento a 2 de $(B - A)$	$A \geq B$	1	$A - B$	Todos 0's

E es el acarreo final para la suma o el préstamo final para la resta.

ENTEROS CON SIGNO: REPRESENTACION SIGNO MAGNITUD

OPERACIONES DE SUMA Y RESTA

*** Definición de un Algoritmo común**

- Suma de dos números de diferente signo es equivalente a una resta
- Resta " " " " " " " " " " suma

*** Representación mediante signo magnitud**

- A, B representan la magnitud de los números
- Diferencia entre el signo del número (+ o -) y la operación
- Suma con igual signo o resta con distinto signo se realiza con una suma
- Resta con igual signo o suma con distinto signo se realiza con una resta
- Signo del resultado
  - * Si es suma el signo es el de A
  - * Si es resta :
    - ➢ El de A si A > B
    - ➢ El contrario de A si A < B
    - ➢ + si el resultado es cero A = B

TABLA 9-2 Suma y resta de números en magnitud con signo

Operación	*Sume las magnitudes*	*Reste las magnitudes*		
		Cuando A > B	*Cuando A < B*	*Cuando A = B*
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

IMPLEMENTACION HARDWARE

*** El Hardware necesario para la realización de estas operaciones**

- Registro AsA: Primer sumando o Minuendo
  * As para el signo y A para la magnitud
- Registro BsB: Segundo sumando o Substraendo
  * Bs para el signo y B para la magnitud
- Registro de Sobreflujo (1 bit) AVF
- Registro de Acarreo (1 bit) E
- Circuito Sumador/Restador (Complemento a dos)
  * Sumador, Acarreo inicial y Complementador
- El resultado se almacenará en AsA (Acumulador)

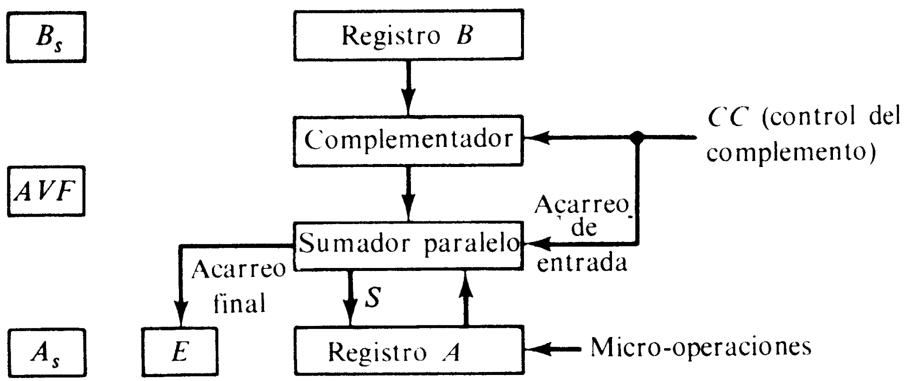


Figura 9-2 Diagrama de bloques del hardware para suma y resta.

ALGORITMO DEL HARDWARE

*** Secuencia de acciones para realizar dichas operaciones**

- Tener en cuenta las operaciones
- Ver el hardware que se dispone
- Necesidad de la función OR-EXCLUSIVE para comparar signos
- Solo se puede producir sobreflujo en la suma
- Si el resultado de resta es < 0 hay que ponerlo en signo magnitud
- Obtención de signo + si el resultado es cero

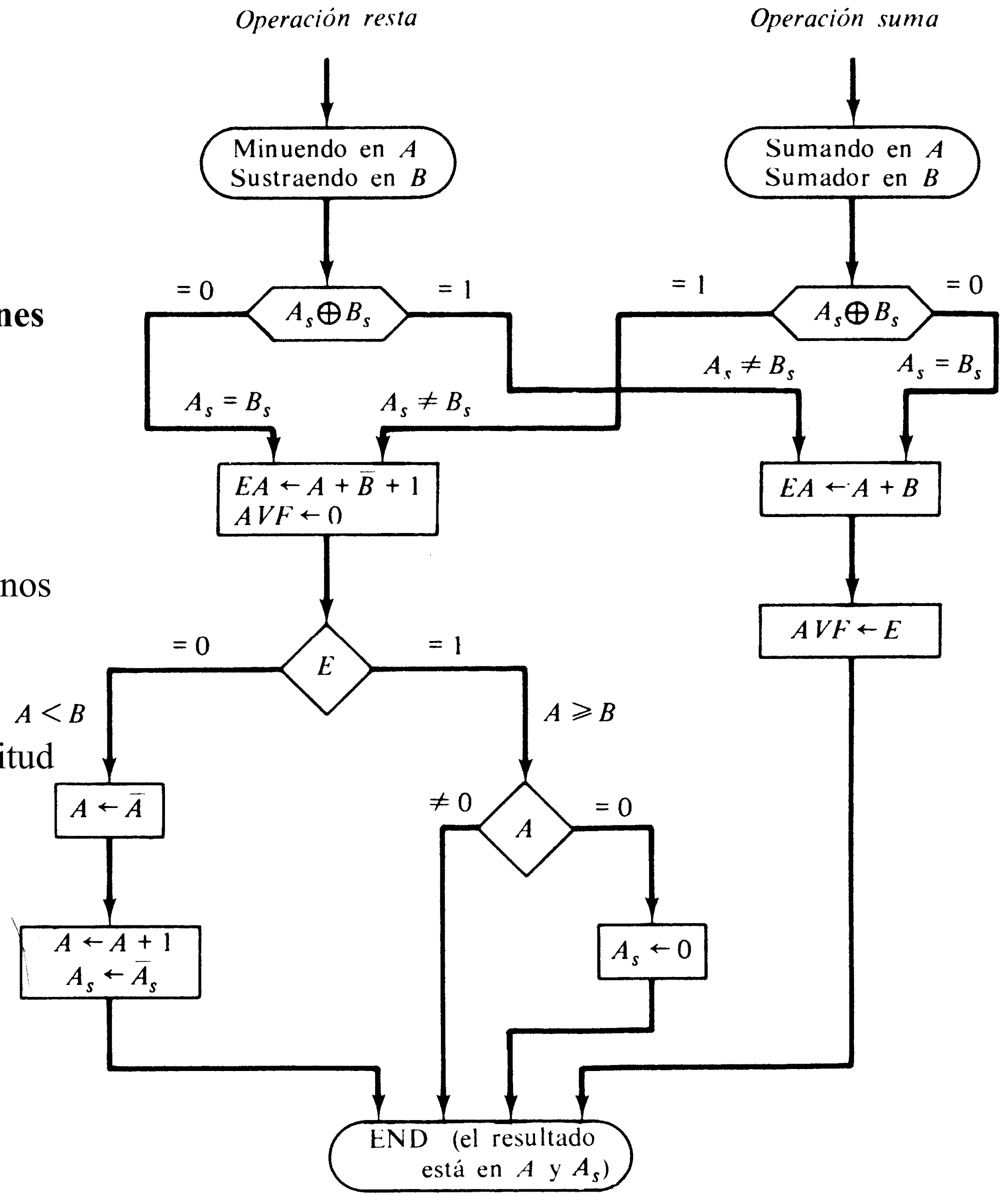


Figura 9-4 Diagrama de flujo para las operaciones sumar y restar.

OPERACION DE MULTIPLICACION

*** Multiplicación de dos número de N bits en representación SM**

- El resultado de 2N bits:
  - Signo: + si son de igual signo y - si son diferentes
  - Magnitud: producto de las dos magnitudes

*** Técnica para multiplicar: repetir N veces el bucle**

- Suma (2N bits) si procede del multiplicando al resultado parcial
- Desplazamiento a izquierda del multiplicando

*** Longitud de los Registros y sumador que asegure el resultado**

- 2N para multiplicando y resultado
- N para el multiplicador
- Longitud del sumador (2N)

```
 23          10111     Multiplicando
 19       x  10011     Multiplicador
 --          -----
             10111
            10111
           00000     +
          00000
         10111
         ---------
437      110110101     Producto
```

Figura 9-5 Ejemplo de multiplicación binaria.

IMPLEMENTACION HARDWARE

* **Modificaciones a la técnica** (ahorran longitud de registros y sumador)

  - Suma (N bits) si procede del multiplicando al resultado parcial
  - Desplazamiento a derecha del resultado parcial

* **Longitud de los Registros y sumador que asegure el resultado**

  - 2N para el Acumulador
  - N para multiplicando y multiplicador
  - Sumador de N bits

* **Registros**
  - BsB para el multiplicando
  - AsA y QsQ para el resultado (QsQ el multiplicador)
  - Registro de acarreo (1 bit) E y contador de secuencia SC

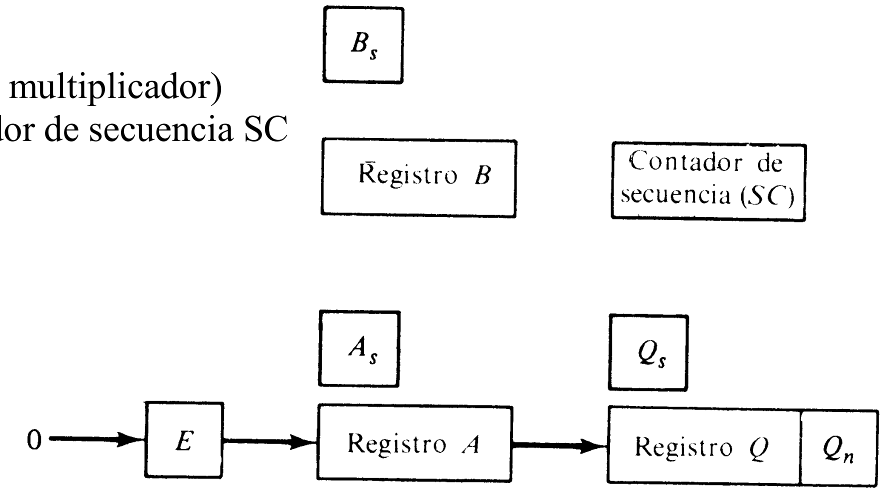


Figura 9-6 Registro EAQ y contador de secuencia necesario para multiplicación y división.

ALGORITMO DEL HARWARE

* Operación OR-EXCLUSIVE para determinar el signo del resultado
* Inclusión de E en los desplazamientos a derecha del resultado parcial
* Número de iteraciones del algoritmo N o N-1

Multiplicando: $B = 10111$	E	A	Q	SC
Multiplicador en Q:		00000	10011	5
$Q_n = 1$; sume B		10111		
Primer producto parcial	0	10111		
shr EAQ	0	01011	11001	4
$Q_n = 1$; sume B		10111		
Segundo producto parcial	1	00010		
shr EAQ	0	10001	01100	3
$Q_n = 0$; shr EAQ	0	01000	10110	2
$Q_n = 0$; shr EAQ	0	00100	01011	1
$Q_n = 1$; sume B		10111		
Quinto producto parcial	0	11011		
shr EAQ; producto final:		01101	10101	0

Figura 9-8 Ejemplo de multiplicación binario con hardware digital.

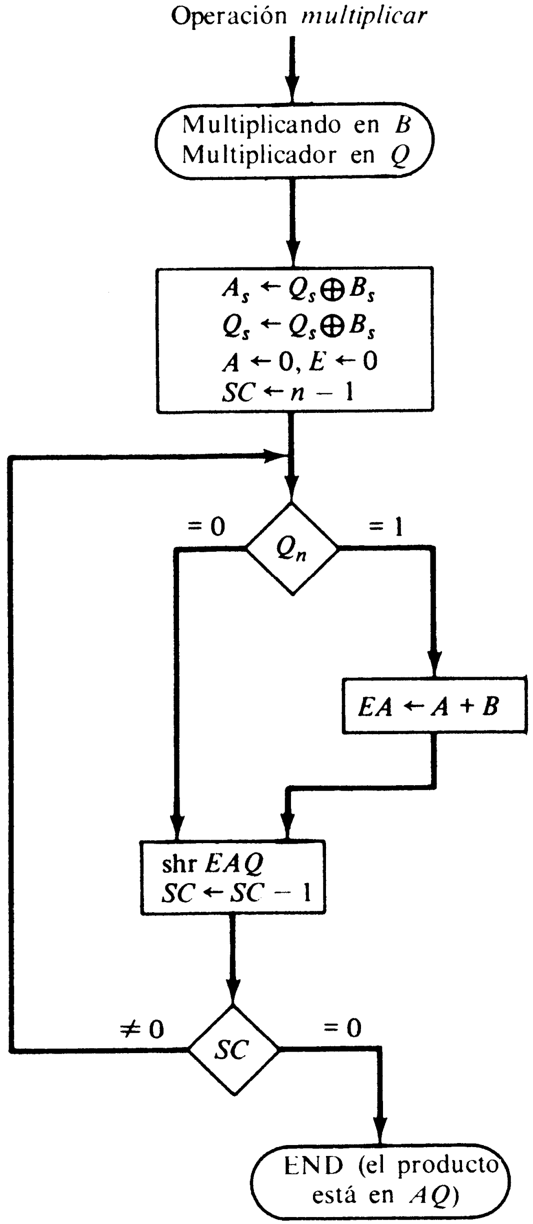


Figura 9-7 Diagrama de flujo para la operación de multiplicación.

OPERACION DE DIVISION

* División de un número de 2N bits por otro de N bits

- El resultado será N bits de cociente y N de resto:
  - Signo: + si son de igual signo y - si son diferentes
  - Magnitud: división de las dos magnitudes

Divisor:

B = 10001

11010	Cociente = Q
0111000000	Dividendo = A
01110	5 bits de $A < B$, cociente tiene 5 bits
011100	6 bits de $A \geqslant B$
-10001	Desplace B a la derecha y reste: entre 1 en Q
-010110	7 bits del residuo $\geqslant B$
--10001	Desplace B a la derecha y reste: entre 1 en Q
--001010	Residuo $< B$: entre 0 en Q: desplace B a la derecha
---010100	Residuo $\geqslant B$
----10001	Desplace B a la derecha y reste: entre 1 en Q
----000110	Residuo $< B$: entre 0 en Q
-----00110	Residuo final

Figura 9-9 Ejemplo de división binaria.

* Técnica para dividir: repetir N veces el bucle

- Resta (N bits) si cabe del divisor al residuo parcial
- Desplazamiento a la derecha del divisor

* Longitud de los Registros y restador que asegure el resultado

- 2N para dividendo y divisor
- N para el cociente y el resto
- Longitud del Restador (2N)

IMPLEMENTACION HARDWARE

* **Modificaciones a la técnica** que ahorran longitud de registros y restador
  - Resta (N bits) si cabe del divisor al residuo parcial
  - Desplazamiento a izquierda del residuo parcial

* **Longitud de los Registros y restador** que asegure el resultado
  - 2N para el dividendo, cociente y resto (un solo registro)
  - N para el divisor
  - Restador de N bits

* **Registros**
  - BsB para el divisor
  - AsA y QsQ para el dividendo, cociente y resto
  - Registro de Sobreflujo de división DVF
  - Registro de acarreo (1 bit) E
  - Registro contador de secuencia SC

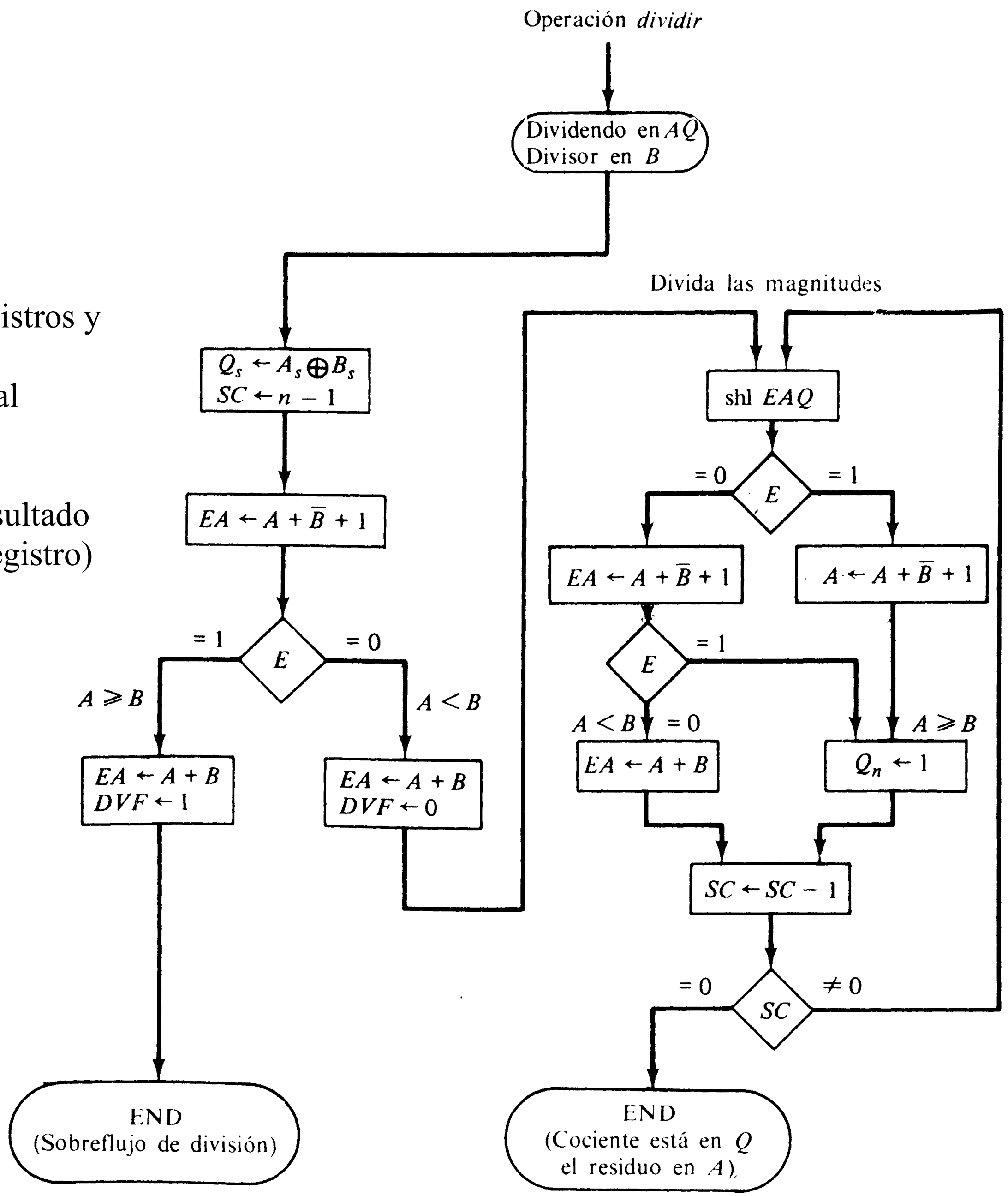


Figura 9-11 Diagrama de flujo para la operación de división.

ALGORITMO DEL HARWARE

* **Algoritmo de división con Restauración**

  - Comparación con resta (C. a dos)
  - Restauración si no cabe

* Problema del Sobreflujo

* Problema de la división por cero

* Operación OR-EXCLUSIVE para determinar el signo del resultado

* Número de iteraciones del algoritmo N o N-1

Divisor $B = 10001$, $\overline{B} + 1 = 01111$

	E	A	Q	SC
Dividendo:		01110	00000	5
shl EAQ	0	11100	00000	
Sume $\overline{B} + 1$		<u>01111</u>		
$E = 1$	1	01011		
Coloque $Q_n = 1$	1	01011	00001	4
shl EAQ	0	10110	00010	
Sume $\overline{B} + 1$		<u>01111</u>		
$E = 1$	1	00101		
Coloque $Q_n = 1$	1	00101	00011	3
shl EAQ	0	01010	00110	
Sume $\overline{B} + 1$		<u>01111</u>		
$E = 0$; deje $Q_n = 0$	0	11001	00110	
Sume B		<u>10001</u>		
Restaure el registro	1	01010		2
shl EAQ	0	10100	01100	
Sume $\overline{B} + 1$		<u>01111</u>		
$E = 1$	1	00011		
Colocar $Q_n = 1$	1	00011	01101	1
shl EAQ	0	00110	11010	
Sume $\overline{B} + 1$		<u>01111</u>		
$E = 0$; deje $Q_n = 0$	0	10101	11010	
Sume B		<u>10001</u>		
Restaurar el residuo	1	00110	11010	0
Desprecie E				
Residuo en A:		00110		
Cociente en Q:			11010	

Figura 9-10 Ejemplo de la división binaria con hardware digital.

OTROS ALGORITMOS

*** Método de Comparación**

- A y B se comparan antes de la resta
  - * si A >= B se realiza la resta y se desplaza
  - * si A < B solo se desplaza
- Requiere lógica adicional que decide si se resta o no

*** Método sin restauración**

- Se realiza la resta A -B como en el de restauración
  - ➢ si el resultado es positivo se sigue igual
  - ➢ si es negativo se desplaza y en la siguiente iteración se suma B

*** Equivalencia de los dos métodos**

- Con restauración

A - B --> (A - B)+ B = A --> 2A --> 2A - B

- Sin restauración

A - B --> 2(A - B) --> 2(A -B) + B = 2A - 2B + B = 2A -B

*** El método sin restauración**

- No necesita restaurar (más rápido)
- Requiere lógica adicional para recodar si se suma o resta

B = 1001 B'+1 = 0111

		E	A	Q		SC
		--	--------		--	
		-	0110	0101		4
shl		0	1100	1010		
+ B'+1			0111			
E=1 Qn=1;SC-1	1	0011	1011		3	
SHL		0	0111	0110		
+ B'+1			0111			
E=0 Qn=0;SC-1	0	1110	0110		2	
shl		1	1100	1100		
+B			1001			
E=1 Qn=1;SC-1	1	0101	1101		1	
shl		0	1011	1010		
+ B'+1			0111			
E=1 Qn=1;SC-1	1	0010	1011		0	

EJEMPLO DE DIVISION SIN RESTAURACION

ENTEROS CON SIGNO: REPRESENTACIÓN COMPLEMENTO A DOS MAS SIGNO OPERACIONES DE SUMA Y RESTA

- **Número en complemento a dos mas signo**
  - Bit de signo (MSB) :
    * 0 para positivos
    * 1 para negativos
  - Resto
    * magnitud para positivos
    * complemento a dos para negativos

- **Sobreflujo e invasión del bit de signo**
  - Suma de números del mismo signo
  - Resta de números de distinto signo
    * OR-EXCLUSIVE del Bs y Acarreo

IMPLEMENTACIÓN DEL HARDWARE

- **Registros**
  - AC; Registro Acumulador (AsA)
  - SC; Registro contador de secuencia
  - V; Registro de bit de sobreflujo
  - BR y QR para otros operandos (BsB y QsQ)
  - Separación del bit de signo de los registros
- **Utilización**
  - AC, BR y V en la suma y resta
  - AC, BR, QR, SC y V en las otras operaciones

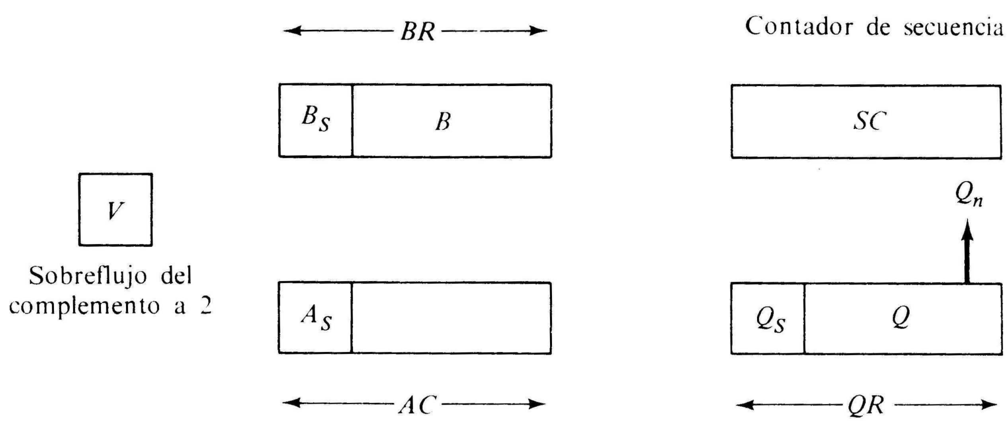


Figura 10-1 Registros para las operaciones aritméticas del complemento a 2 con signo.

SUMA Y RESTA

- La representación en complemento a dos facilita estas operaciones

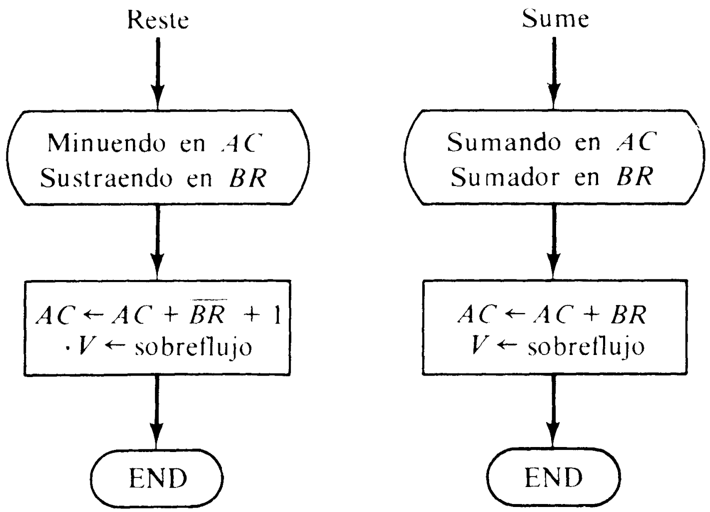


Figura 10-2 Algoritmo para sumar y restar números en la representación del complemento a 2 con signo.

DESPLAZAMIENTOS ARITMÉTICOS

- **Desplazamientos con conservación del signo**
  - Problema con los nº en complemento a dos
  - Desplazamiento a izquierda puede producir error
    * Comprobación posterior de overflow
  - Desplazamiento a derecha debe mantener el signo
    * Se puede realizar sin error

- **Aplicación al caso de la multiplicación**
  - Operación de desplazamiento después de una suma
    * Condición o no de overflow en la suma

TABLA 10-1 Valor del bit del signo para desplazamiento aritmético a la derecha

Sobreflujo, V	Bit del signo A_s antes del desplazamiento	Bit del signo A_s después del desplazamiento	Comentarios
0	0	0	No hay sobreflujo, el signo permanece positivo
0	1	1	No hay sobreflujo, el signo permanece negativo
1	0	1	Sobreflujo, necesita inversión de signo
1	1	0	Sobreflujo, necesita inversión de signo

- **La operación de desplazamiento aritmético** la expresaremos

  shr AC junto con As <--- As **OR-EX** V

MULTIPLICACIÓN Y DIVISIÓN

- **Soluciones con matrices multiplicadoras**
  - Costos aunque rápidos
- **Soluciones con ROMs o EPROMs**
  - Anchura no grande de palabra
- **Secuencia de operaciones de sumas y desplazamientos**
  - La más usada; económica pero lenta

MULTIPLICACIÓN CON NUMEROS EN COMPLEMENTO A DOS

- **Si los dos son positivos**
  - Algoritmo de los números S-M olvidando el signo
- **Si el Multiplicando es negativo y el Multiplicador positivo**
  - Algoritmo de S-M (la suma en complemento a dos va bien)
  - Realización de desplazamientos aritméticos
- **Si el multiplicador es negativo**
  - No se puede aplicar el Algoritmo de S-M
  - El multiplicador debe de ser positivo
- **Algoritmo de Multiplicación I**
  - Asegurar que ambos sean siempre positivos
    * se calcula primero el signo (Or-Ex)
    * se convierten a positivos
    * se hace el Algoritmo S-M
    * si el signo es negativo se hace el Complemento a dos

- **Algoritmo de Multiplicación II**
  - Asegurar que el multiplicador sea siempre positivo
    - * Se testea el signo del multiplicador
    - * Si es negativo se cambian los dos

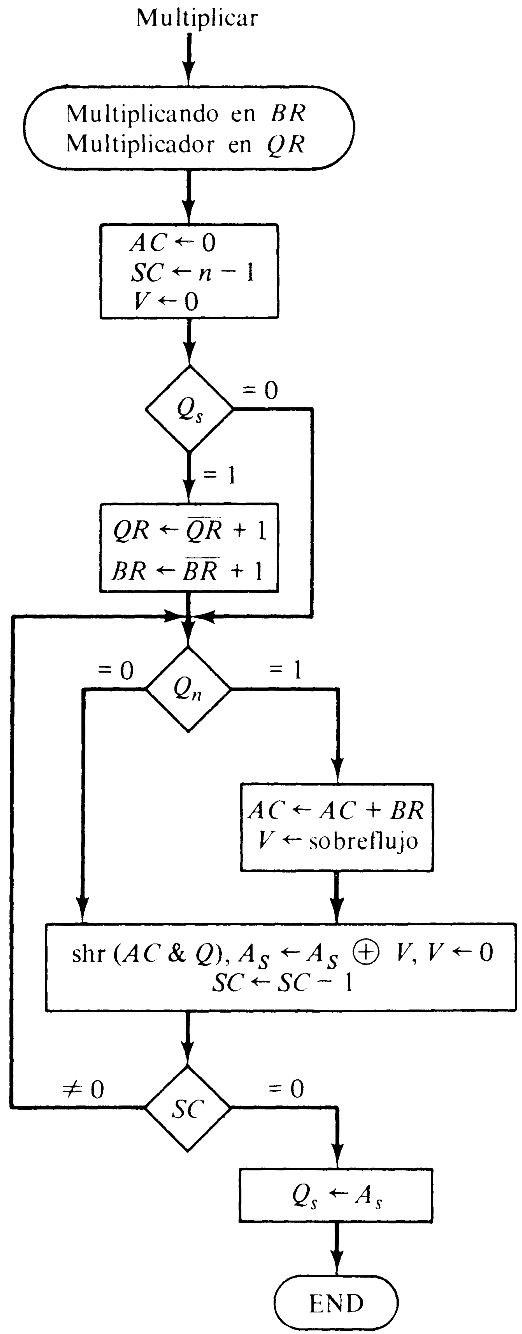


Figura 10-4 Algoritmo de multiplicación para números binarios en la representación del complemento a 2 con signo.

ALGORITMO DE MULTIPLICACIÓN DE BOOTH'S

- **Resuelve el problema del multiplicador negativo**

- **Fundamento**
  - Una hilera de 0s seguidos en el multiplicador no requiere sumas
  - Para una hilera de unos seguidos puede simplificarse

    ....01111...1110000 ---> aportación $(2^{u+1} - 2^{v})$* Multiplicando

    v y **u** posición donde empiezan (msb) y acaba la hilera de unos
  - ejemplo

    $0111100 = 2^6 - 2^2$
  - Si el multiplicador es negativo la última operación es una resta
  - Hardware adicional : Registro de 1 bit Q_{n+1}

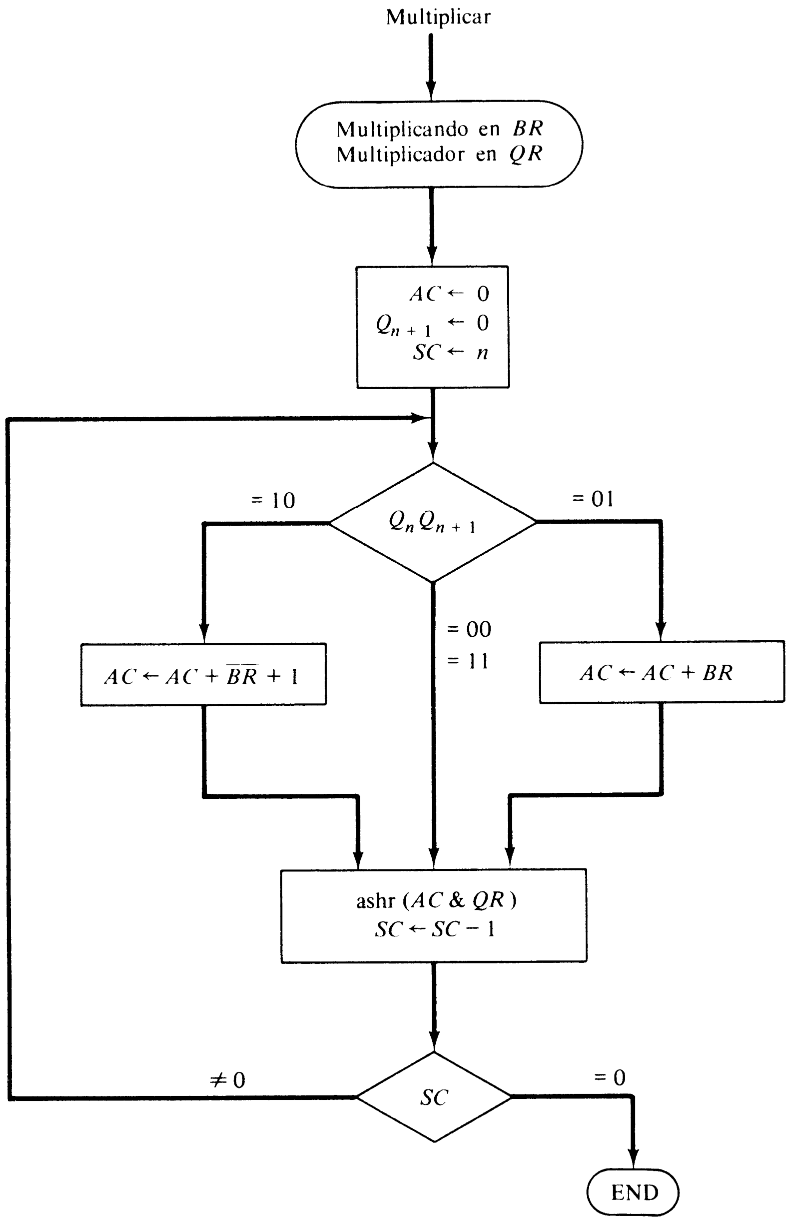


Figura 10-5 Algoritmo de Booth para la multiplicación de números en el complemento de 2 con signo.

TABLA 10-2 Ejemplo de la multiplicación con el algoritmo de Booth

$BR = 10111$

$\overline{BR} + 1 = 01001$

Q_nQ_{n+1}		AC	QR	Q_{n+1}	SC
	Iniciai	00000	10011	0	5
1 0	Reste BR	01001			
		01001			
	ashr	00100	11001	1	4
1 1	ashr	00010	01100	1	3
0 1	Sume BR	10111			
		11001			
	ashr	11100	10110	0	2
0 0	ashr	11110	01011	0	1
1 0	Reste BR	01001			
		00111			
	ashr	00011	10101	1	0

DIVISIÓN CON NUMEROS EN COMPLEMENTO A DOS

- **Algoritmo basado en el de S-M**

  - Se calcula signo del resultado
  - Se ponen los dos positivos
  - Se realiza la división
  - Se complementa en su caso

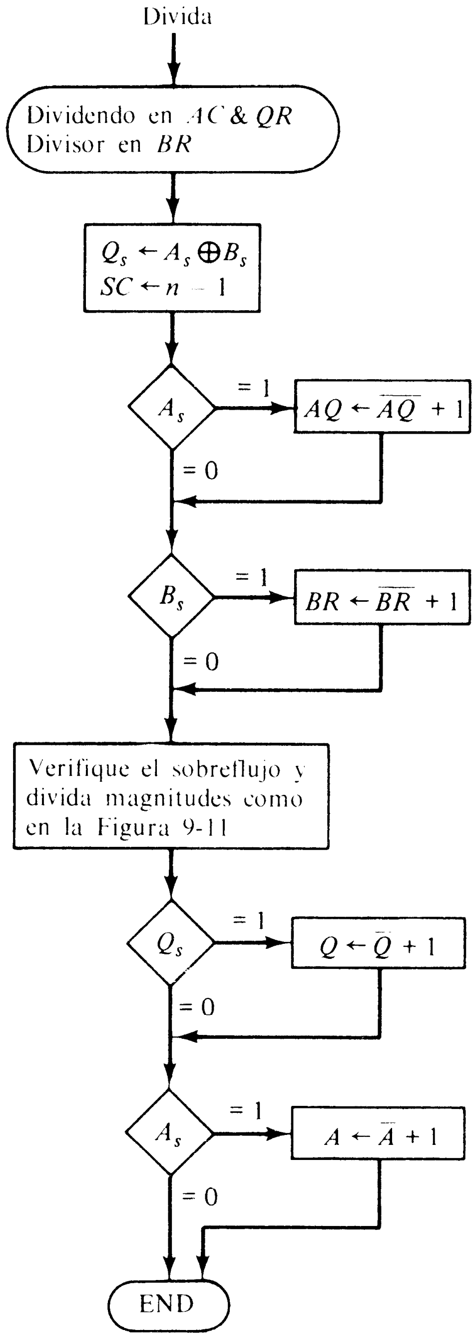


Figura 10-6 División de números binarios en la representación del complemento a 2 con signo.

OPERACIONES ARITMÉTICAS EN PUNTO FLOTANTE

- **Número real en notación científica** : $N = M * b^{e}$
  - **N** el número real; **M** la mantisa, **b** una base y **e** el exponente
  - La mantisa se puede expresar como
    > parte entera
    > parte decimal
    > mezcla de ambas
  - ejemplos
    > $J = - 4560 * 10^{32}$
    > $N = 5.659 * 10^{20}$
    > $H = 0.011101 * 2^{1011}$

- **Representación en Punto Flotante**
  - Poder variar la posición del punto
    > $5.659 * 10^{20} = 0.5659 * 10^{21} = 56.59 * 10^{19}$
    > $0.011101 * 2^{1011} = 0.111010 * 2^{1010}$

  - Expresión normalizada
    > cuando el primer dígito tras el punto es distinto de cero

  - Valores necesarios para almacenar un Nº en P.F.
    > la mantisa (M)
    > el exponente (e)
    > la base no es necesaria

- **Representación en P.F de números en binario**

  - La mantisa se da como parte no entera

  - Para incluir valores negativos se utiliza la representación S-M

  - El exponente se representa con sesgo para incluir valores negativos

- **Ejemplo para un número binario en P.F.**
  - para 32 bits

S/EEEEEEEE/MMMMMMMMMMMMMMMMMMMMMMM

- Distribución de los bits
  > 1 bit (msb) para signo de mantisa
  > 8 bits de exponente con sesgo = 128 en S.P.
  > 11 bits de exponente con sesgo =1024 en D.P.
  > 23 bits de mantisa en S.P.
  > 52 bits de mantisa en D.P.

- Valor del número
  > $(-!)^{msb} * 0.M * 2^{exp-128}$ en S.P.
  > $(-!)^{msb} * 0.M * 2^{exp-1024}$ en D. P.

- **Ejemplo**

0-10000111-11000000000000000000000

- Signo de mantisa: msb = 0 ---> positivo
- Mantisa 0.11000.....00
- Exponente 10000111-10000000= 00000111
- valor decimal

$$N = + 0.11 * 2^{111} = 0.75 * 2^{7} = 0.75 * 128 = 96.0$$

- **Normalización de un nº en P.F.**

- Se supone la mantisa como parte no entera
- Conseguir que el primer dígito significativo sea 1
- Desplazamientos a izquierda con decrementos del exponente

- **Standard IEEE 754, 1985**

- Simple (32 bits) y doble precisión (64 bits)
- Formato similar (signo-exponente-mantisa)
- Valor decimal (diferencia)

$$N = (-1)^{msb} * 1.M * 2^{exp-sesgo} \quad (sesgo = 127 \text{ ó } 1023)$$

- Se aumenta en un bit la capacidad de representación

- **Representación de números singulares**

  - CERO --> EXP = 0 y M = 0
  - + INFINITO ---> SIGNO = 0, EXP = 255 ó 2047 y M = 0
  - - INFINITO ---> SIGNO = 1, EXP = 255 ó 2047 y M = 0
  - Números desnormalizados ---> EXP = 0 y M distinto de 0

    > su valor $(-1)^{msb} * 0.M * 2^{-126\ (1022)}$

  - No un Número (NaNs) ---> EXP = 255 ó 2047 y M distinto de 0

- **Problemas con la representación en P.F.**

  - Truncamiento y perdida de precisión
  - Bits de guarda, etc.

- **Operaciones Aritméticas en Punto Flotante**

  - Suma [resta] (necesidad de tener los mismos exponentes)
    > Exponente el común
    > La mantisa será la suma [resta] de las mantisas

  - Multiplicación [división]
    > El exponente será la suma [resta] de los exponentes
    > La mantisa será el producto [división] de las mantisas

  - Necesidad de mantener siempre la representación normalizada

COMPONENTES HARDWARE PARA LAS OPERACIONES EN P.F.

- **Los registros en P.F. deben de almacenar**
  - La mantisa, incluido bit para signo
  - El exponente

- **Necesidad de operar independientemente mantisas y exponentes**
  - Mantisas (basados en los algoritmos de S-M)
    > Suma y Resta
    > Multiplicación y División

- **Los circuitos Aritméticos**
  - Sumadores/Restadores para operar las mantisas
  - Sumadores/Restadores y Comparadores para los exponentes

CONSIDERACIONES INICIALES Y FINALES

- **Condición de partida:** los operandos están normalizados (el 0 no)

- **Evitar operaciones con ceros**: Suma, Resta, Multiplicación y División

- **Condición de salida**: números normalizados

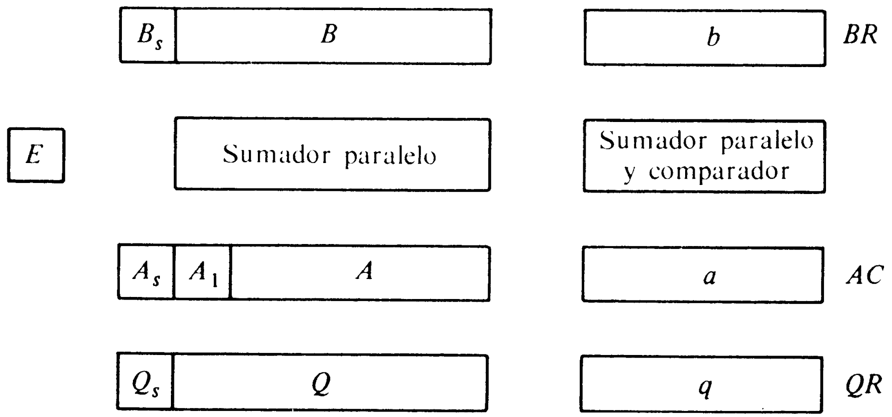


Figura 10-7 Registro para operaciones aritméticas en punto flotante.

SUMA Y RESTA DE NUMEROS EN P. F.

- **Pasos necesarios:**

  - Verificación de ceros
  - Alineación de las mantisa
  - Sumar o restar las mantisas
  - Normalizar el resultado

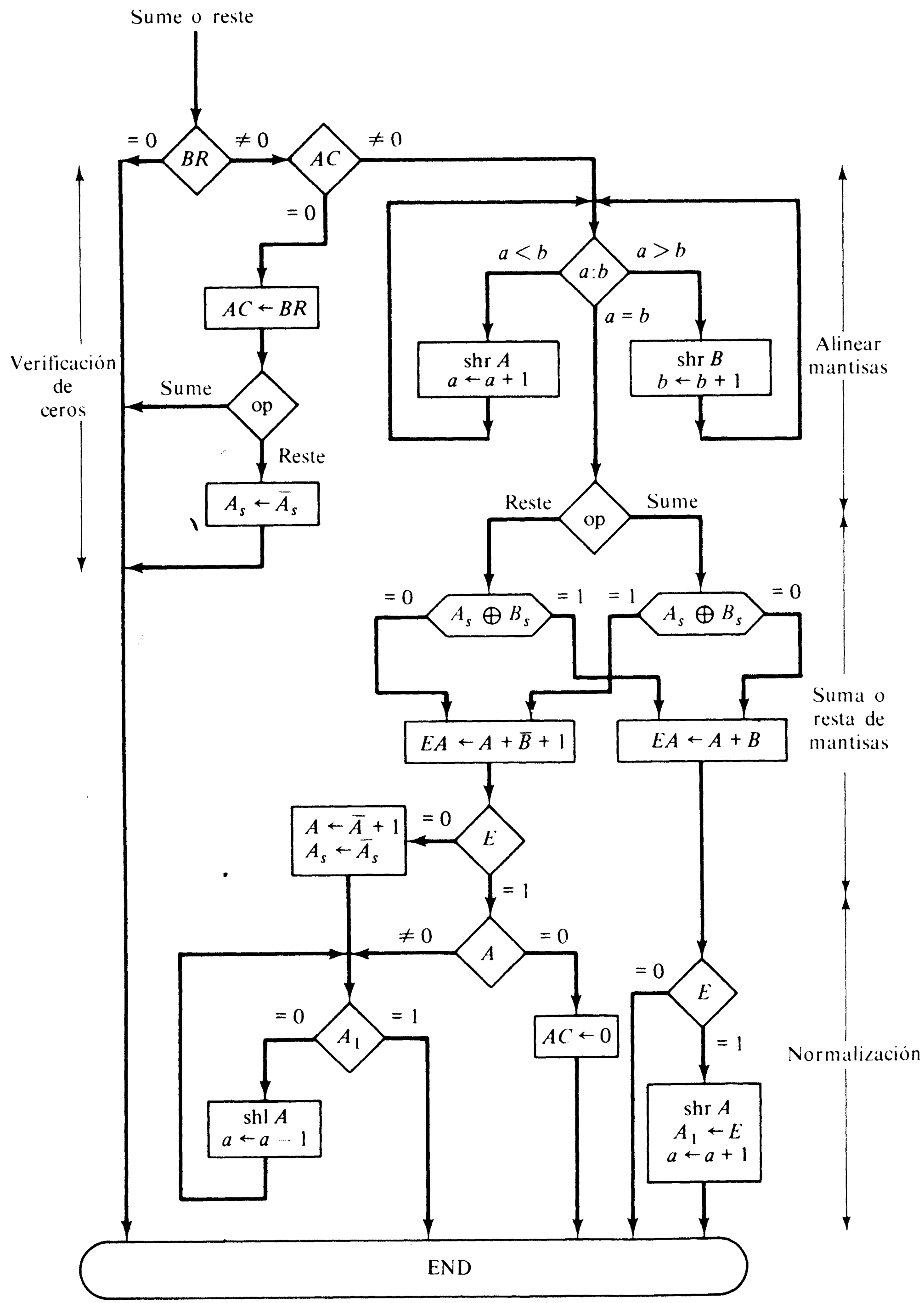


Figura 10-8 Suma y resta de número en punto flotante.

MULTIPLICACIÓN DE NUMEROS EN PUNTO FLOTANTE

- **Se multiplican las mantisas y se suman los exponentes**
  - Mantisas de N (o N-1) bits para los factores
  - Multiplicando en BR y Multiplicador en QR
  - Truncar resultado (AQ) de la mantisa a N (o N-1) bits
- **Pasos a seguir**
  - Verificar ceros
  - Sumar exponentes (nº en exceso m)
  - Multiplicar mantisas (alg. S-M)
  - Normalizar el resultado
- **Análisis del Algoritmo**
  - Resultado en AC con N (o N-1) bits de mantisa
  - Corrección de la suma de exponentes restándole el sesgo
  - La normalización final
    * Solo requiere, si procede, un solo paso
    * El desplazamiento incluye Q
  - Problemas de precisión al truncar el resultado

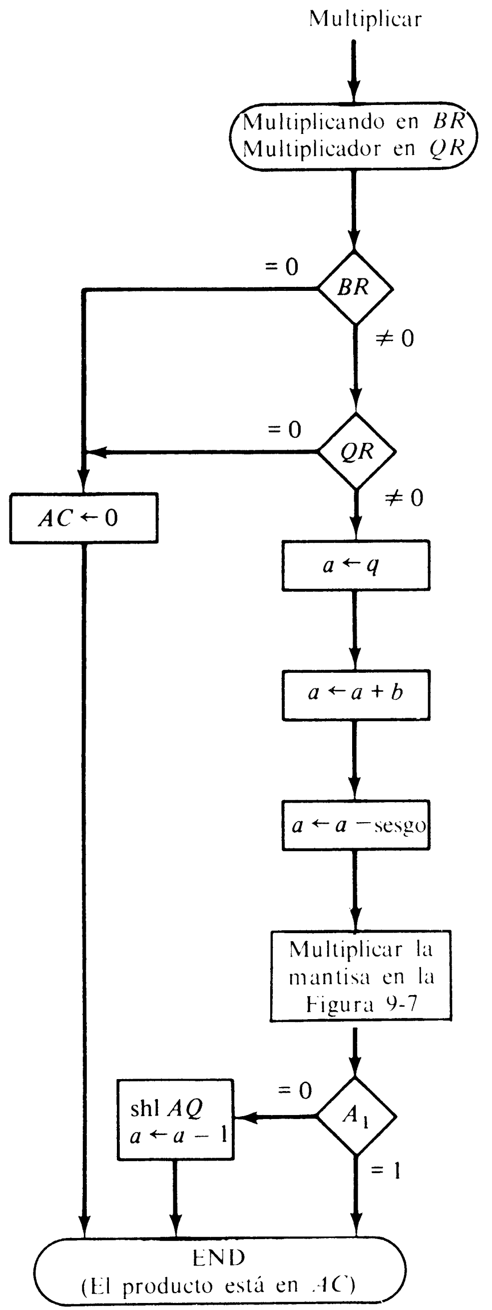


Figura 10-9 Multiplicación de números en punto flotante.

DIVISION DE NUMEROS EN PUNTO FLOTANTE

- **Se restan los exponentes y se dividen las mantisas**

  - Se parten de mantisas de N (o N-1) bits
  - Divisor en BR y Dividendo en AC
  - División S-M para las mantisas
    * Mantisa de dividendo en AQ (Q a 0)
    (se puede hacer así ya que la mantisa es fraccionaria)
    (no sería posible para mantisas enteras)

- **Pasos a seguir**

  - Verificar ceros
  - Inicializar registros y evaluar signo
  - Alinear el Dividendo (evita el sobreflujo)
  - Restar exponentes
  - Dividir mantisas (alg. S-M)

- **El resultado**

  - El cociente en QR
    * Recuperar el sesgo después de la resta de exponentes
  - El residuo en AC
    * Hay que ponerle el exponente adecuado
    (se le resta al exponente N-1 y luego se normaliza)

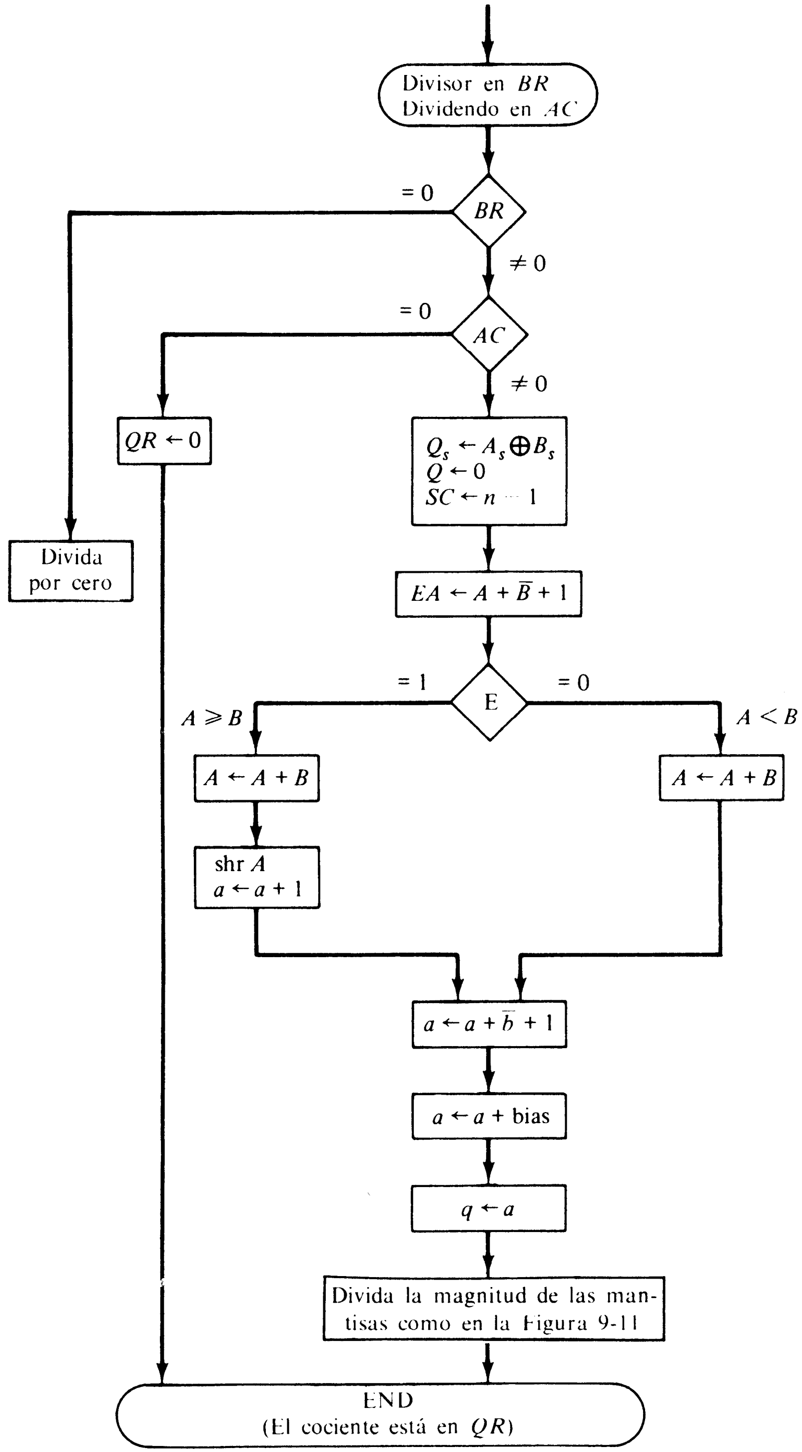


Figura 10-10 División de números en punto flotante.