

Arquitectura de Computadoras para Ingeniería

(Cód. 7526)
1° Cuatrimestre 2016

Dra. Dana K. Urribarri
DCIC - UNS

Operaciones Aritméticas

Implementación de las operaciones aritméticas básicas:

- 1) Suma
- 2) Resta
- 3) Multiplicación
- 4) División



Divisores

División

Es la más compleja y que consume tiempo de las operaciones básicas.

El resultado tiene dos partes:

- Dados un dividendo X y un divisor D , el resultado de X/D consta de un cociente Q y un resto R , tales que:

$$X = Q \cdot D + R \quad \text{con} \quad 0 \leq R < D.$$

División

- El producto de un número de n dígitos por un número de m dígitos da como máximo un resultado de $m + n$ dígitos.
- La división de un número de $n + m$ dígitos por un número de n dígitos da como máximo un cociente de $m + 1$ dígitos y un resto de n dígitos.

División $b = 2$

Si consideramos $n = m$

- Tenemos
 - Dividendo X de $2n$ bits.
 - Divisor D y resto R de n bits.
 - Cociente Q de $n + 1$ bits
- Si se trabaja con registros de n bits, que el cociente resulte de $n + 1$ bits genera overflow.

Posibles errores

- División por cero
 - Debe ser $D \neq 0$
- Overflow
 - D y Q enteros no signados positivos
 - El máximo valor posible en un registro de n bits es $2^n - 1$
 $\therefore Q, D, R < 2^n$ y $R < D$.
$$X = Q \cdot D + R < Q \cdot D + D < (2^n - 1) D + D = 2^n D$$
 - Por lo tanto, debe ser $X < 2^n \cdot D$
 - La parte alta de X debe ser menor estricto que D.
 - Esto también sirve para detectar división por cero.

División secuencial

- La multiplicación secuencial se resolvía con sumas sucesivas.
- La división secuencial se resuelve con restas sucesivas.

Pero requiere elegir o estimar el próximo dígito del cociente.

- La división es un proceso de prueba y error.

División secuencial X/D

- 1) Inicializar el resto parcial $R^{(0)}$ en X
- 2) Restar sucesivamente $q_{k-j} D$ desplazado.

									Divisor
Dividendo	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	d_3 d_2 d_1 d_0
$-q_3 D b^3$		●	●	●	●				q_3 q_2 q_1 q_0
$-q_2 D b^2$			●	●	●	●			Cociente
$-q_1 D b^1$				●	●	●	●		
$-q_0 D b^0$					●	●	●	●	
					●	●	●	●	Resto

División secuencial $b = 2$

- Asumiendo n bits para el dividendo X y k bits para el cociente Q .
- $R^{(0)} = X$
- Para j desde $k - 1$ hasta 0
 - $R^{(k-j)} = R^{(k-j-1)} - q_j 2^j D$
 - Donde $q_j \in \{0,1\}$ es tal que $0 \leq R^{(k-j)} < D$
- Finalmente, $R^{(k)} = R$

División secuencial $b = 2$

- Cada resta está desplazada un bit con respecto a la anterior.
- Alternativa:
 - Fijar el desplazamiento de $q_{k-j}D$ en 2^k
 - Desplazar sucesivamente el resto parcial un lugar a izquierda.

$$R^{(j)} = 2R^{(j-1)} - q_{k-j} 2^k D$$

$$R^{(0)} = X \quad y \quad R^{(k)} = 2^k R$$

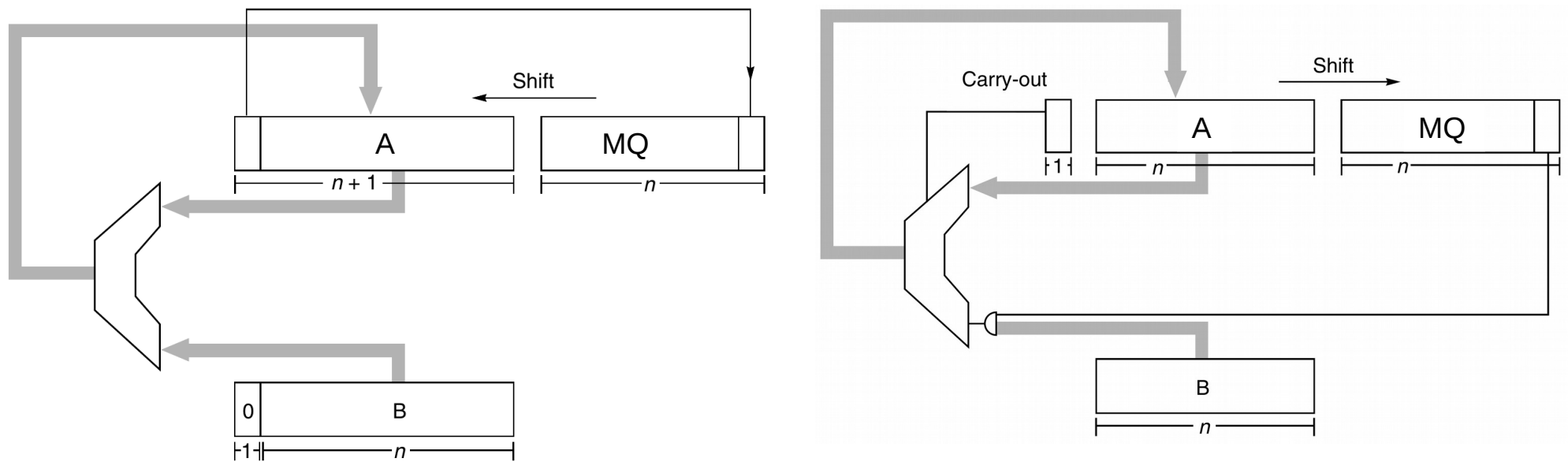
División con restauración

En base dos, los bits q_i pueden ser 0 o 1.

- La división con restauración (*restoring division*) encuentra los bits del cociente por prueba y error.
- Para cada bit q_i del cociente asume valor 1.
Si no era la elección correcta porque la resta dio un resultado negativo, restaura el valor anterior y q_i debía valer 0.
- La restauración del valor puede hacerse:
 - Manteniendo registros separados para el resto de prueba y el resto real.
 - Con un único registro y realizando la suma para restaurar.

Hardware de división

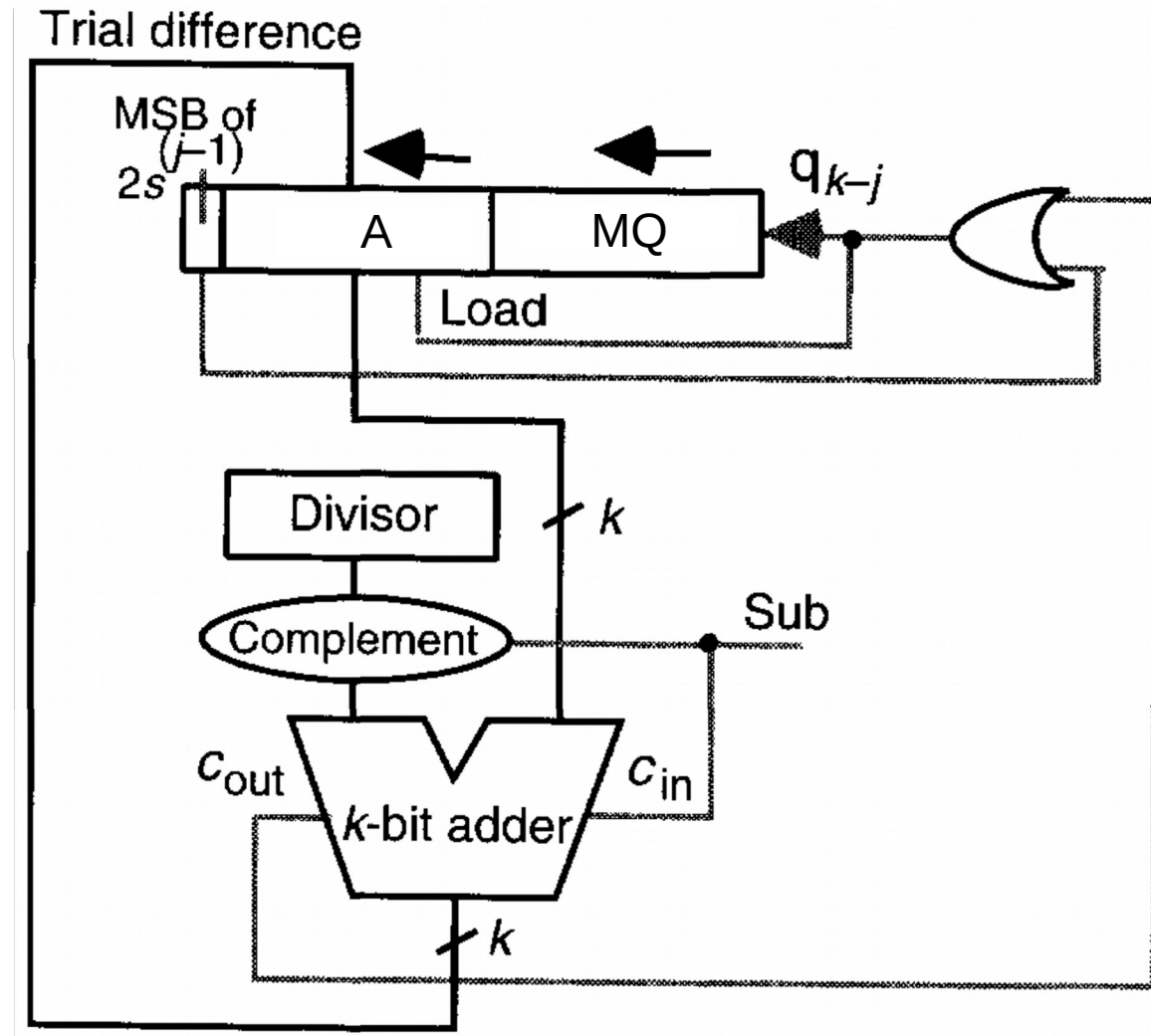
- La división y la multiplicación pueden compartir parte del hardware



- Dado que los bits del cociente ingresan a medida que se desplaza el resto parcial, ambos pueden compartir los mismos registros.

División con restauración

- Calcular X/D
- En el registro $A|MQ$ se carga con X y se completa con 0 a la izquierda.



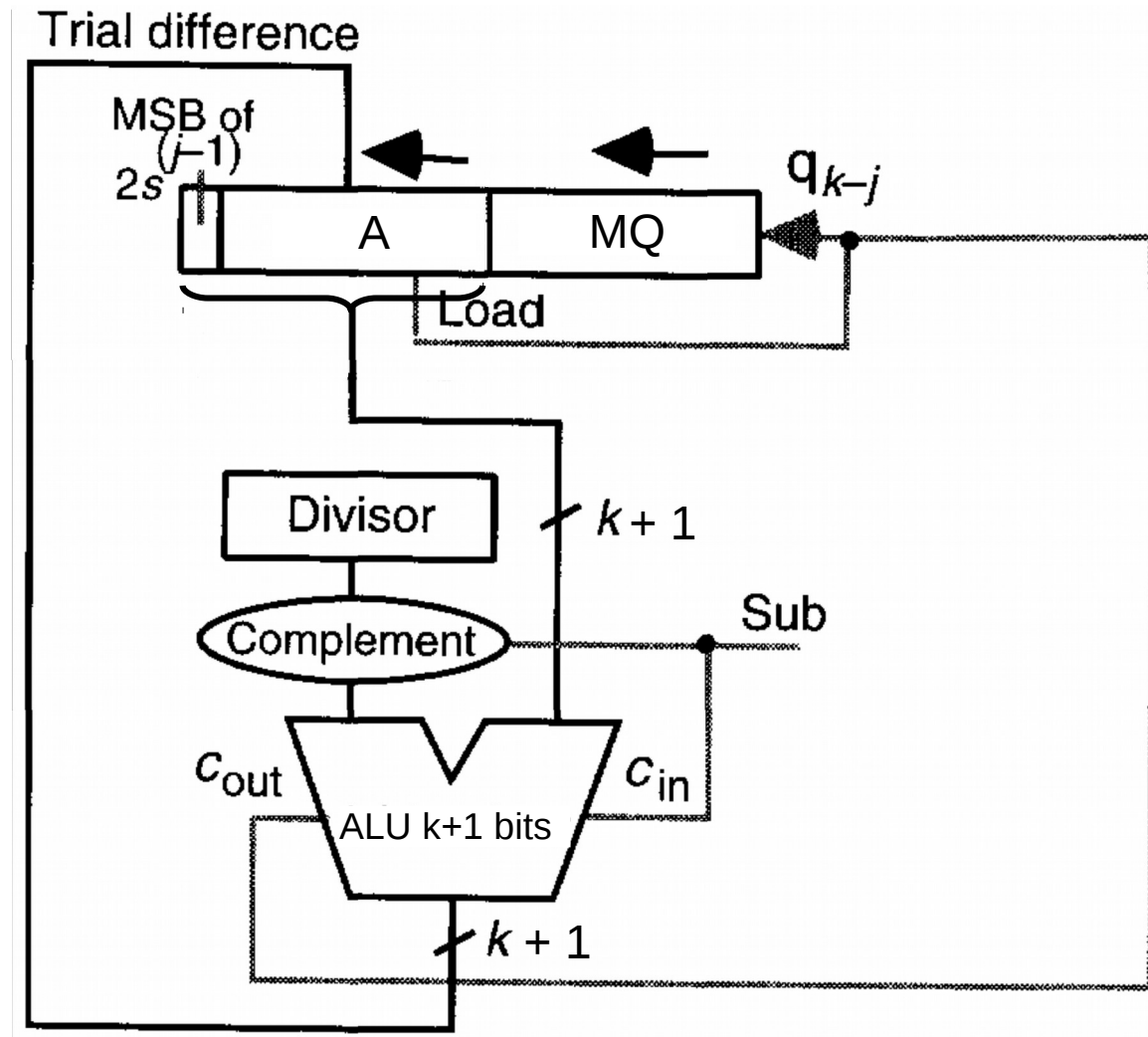
División con restauración

En cada ciclo j :

- 1) Desplazar A|MQ un lugar a izquierda. El MSB se almacena en un FF especial.
- 2) Calcular el resto de prueba $2 R^{(j-1)} - q_{k-j} 2^k D$
- 3) Si $FF = 1$ o la resta dio positivo ($C_{OUT} = 1$)
 - 1) El resto de prueba se carga en A
 - 2) $q_{k-j} = 1$
- 4) Cualquier otro caso, $q_{k-j} = 0$
- 5) Desplazar A|MQ a izquierda entrando q_{k-j} .
En el último ciclo desplazar solamente MQ.

División con restauración

- Se puede simplificar asumiendo
 - A de $n + 1$ bits y
 - ALU de $n + 1$ bits.



División con restauración

En cada ciclo j :

- 1) Desplazar $A|MQ$ un lugar a izquierda
- 2) Calcular el resto de prueba $2 R^{(j-1)} - q_{k-j} 2^k D$
- 3) Si el resto de prueba es positivo o cero ($C_{OUT} = 1$),
 - 1) El resto de prueba se carga en A
 - 2) $q_{k-j} = 1$
- 4) Si el resto de prueba es negativo, $q_{k-j} = 0$
- 5) Desplazar $A|MQ$ a izquierda entrando q_{k-j} .
En el último ciclo desplazar solamente MQ .

Ejemplo

$X = 26$

$D = 4$

$X = 11010$

$D = 000100$

$-D = 111100$

	A	MQ
	000000	11010
1 ^{er} desplazamiento.	000001	1010-
-4	111100	
< 0	111101	
Restauro. Shift, $q_i = 0$	000011	010-0
-4	111100	
< 0	111111	
Restauro. Shift, $q_i = 0$	000110	10-00
-4	111100	
≥ 0	000010	
Shift, $q_i = 1$	000101	0-001
-4	111100	
≥ 0	000001	
Shift, $q_i = 1$	000010	-0011
-4	111100	
< 0	111110	
Restauro. Shift MQ, $q_i = 0$	000010	00110
	R	Q

Ejemplo

$X = 117$

$D = 10$

$X = 1110101$

$D = 001010$

$-D = 110110$

	A	MQ
	000011	10101
1 ^{er} desplazamiento.	000111	0101-
-10	110110	
< 0	111101	
Restauró. Shift, $q_i = 0$	001110	101-0
-10	110110	
> 0	000100	
Shift, $q_i = 1$	001001	01-01
-10	110110	
< 0	111111	
Restauró. Shift, $q_i = 0$	010010	1-010
-10	110110	
≥ 0	001000	
Shift, $q_i = 1$	010001	-0101
-10	110110	
≥ 0	000111	
Shift MQ, $q_i = 1$	000111	01011
	R	Q


División sin restauración

- La implementación con restauración tienen problemas de temporizado.
- Cada ciclo de reloj debe ser lo suficientemente largo como para:
 - Desplazar registros
 - Propagar señales al sumador
 - Determinar y almacenar el próximo dígito del cociente
 - Eventualmente, almacenar el resto de prueba
- Además, los últimos eventos dependen de los primeros.

División sin restauración

- La división sin restauración (*nonrestoring division*) busca evitar esos problemas de temporizado.
- Siempre asume que $q_i = 1$, resta y almacena el resto parcial aunque sea incorrecto. Se corrige en el próximo ciclo.

División sin restauración

- Sea U el resto parcial ya desplazado.
- Si la resta $U - 2^k D < 0$ hay dos opciones:
 - 1) Restaurar el valor inicial de U , desplazar a izquierda nuevamente y volver a restar $2^k D$ para obtener: $2U - 2^k D$ *Div. C/ restauración*
 - 2) Mantener el valor negativo, desplazar izquierda y sumar $2^k D$: $2(U - 2^k D) + 2^k D = 2U - 2^k D$ 
- Si el resto final queda negativo, hay que corregir
 - $R \leftarrow R + D$
 - $Q \leftarrow Q - 1$

No hay próximo paso para corregir.

Ejemplo

$X = 117$

$D = 10$

$X = 1110101$

$D = 001010$

$-D = 110110$

1^{er} desplazamiento.

-10

< 0

Shift, $q_i = 0$

+10

≥ 0

Shift, $q_i = 1$

-10

< 0

Shift, $q_i = 0$

+10

≥ 0

Shift, $q_i = 1$

-10

≥ 0

Shift MQ, $q_i = 1$

	A	MQ
	000011	10101
	000111	0101-
	110110	
	<hr/>	
	111101	
	111010	101-0
	001010	
	<hr/>	
	000100	
	001001	01-01
	110110	
	<hr/>	
	111111	
	111110	1-010
	001010	
	<hr/>	
	001000	
	010001	-0101
	110110	
	<hr/>	
	000111	
	000111	01011
	R	Q

División rápida

- Hay dos formas diferentes de lograr acelerar la división:
 - Con sumas/restas y desplazamientos $O(n)$
 - Con multiplicación $O(\log(n))$

División por convergencia

- Sea X y D el dividendo y el divisor.
- Se puede escribir como una fracción donde X es el numerador y D el denominador.

$$Q = \frac{X}{D} = \frac{X \cdot R_0 \cdot R_1 \cdots R_{m-1}}{D \cdot R_0 \cdot R_1 \cdots R_{m-1}} \rightarrow \frac{Q}{1}$$

- Se calcula solamente el cociente.
- Adecuado para división en punto flotante.

División por convergencia

- Supongamos D una fracción normalizada $0.1\cdots$
- $\frac{1}{2} \leq D < 1$ y además $D = 1 - y$ donde $y \leq \frac{1}{2}$.
- Si se selecciona $R_0 = 1 + y$
 - $D_1 = D R_0 = (1 - y)(1 + y) = 1 - y^2$
 - Como $y^2 \leq \frac{1}{4}$, $D_1 = 0.11\cdots \geq \frac{3}{4}$
- Si se selecciona $R_1 = 1 + y^2$
 - $D_2 = D_1 R_1 = (1 - y^2)(1 + y^2) = 1 - y^4$
 - Como $y^4 \leq \frac{1}{16}$, $D_2 = 0.1111\cdots \geq \frac{15}{16}$

División por convergencia

Convergencia

- En general $D_i = 1 - y^{2^i}$
- Como $y \leq 2^{-1}$ entonces $y^{2^i} \leq 2^{-2^i}$

$$y^{2^i} = 0.\underbrace{0\dots0}_i\dots$$

$$D_i = 1 - y^{2^i} = 0.\underbrace{1\dots1}_i\dots$$

$$\lim_{i \rightarrow \infty} D_i = 1$$

División por convergencia

Cálculo del factor de multiplicación R_i

- Para obtener R_i a partir de D_i

$$D_i = 1 - y^{2^i} \quad R_i = 1 + y^{2^i}$$

$$R_i = 1 + 1 - D_i = 2 - D_i$$

- R_i es el complemento a 2 de D_i

Ejemplo

$$X = 0.1101 \quad D = 0.11$$

$$R_0 = 2 - D = 1.01$$

$$- D_1 = D \cdot R_0 = 0.1111$$

$$- X_1 = X \cdot R_0 = 0.1000001$$

$$R_1 = 2 - D_1 = 1.0001$$

$$- D_2 = D_1 \cdot R_1 = \\ 0.11111111$$

$$- X_2 = X_1 \cdot R_1 = 0.1000101$$

$$R_2 = 2 - D_2 = 1.00000001$$

$$- D_3 = D_2 \cdot R_2 = \\ 0.1111111111111111$$

$$- X_3 = X_2 \cdot R_2 = \\ 0.100010101010101$$

División SRT

- Acelera la división sin restauración.
- Para el cociente usa los dígitos $\{-1, 0, 1\}$
 - -1 : sumar
 - 0 : no operar
 - 1 : restar
- Busca evitar sumar/restar cuando el resto parcial está entre $[-D, D)$.
 - $q_i = 0$ y se desplaza el resto parcial a izquierda $[-2D, D)$.

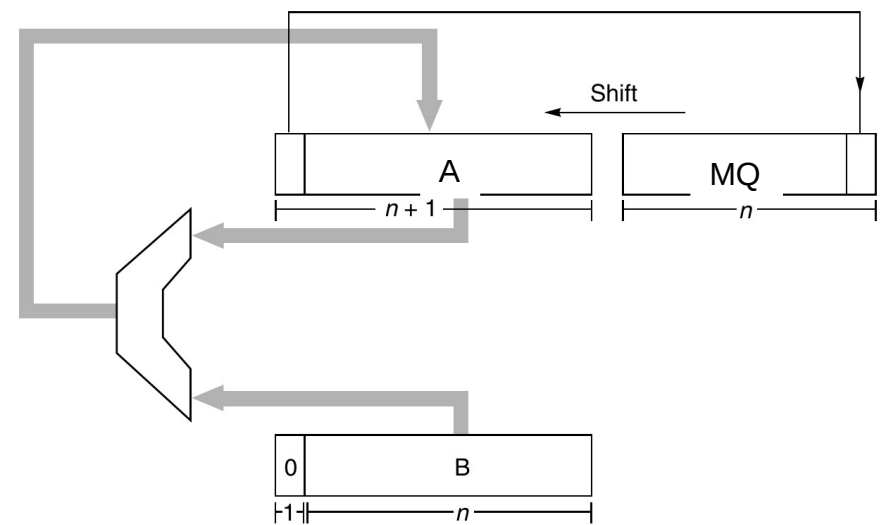
División SRT

- ¿Cómo se sabe que el resto parcial está entre $[-D, D)$?

Restando

- La solución se conoce como algoritmo SRT.
- Sweeney, Robertson y Tocher llegaron independientemente al mismo algoritmo.

Algoritmo



- X/D
- $MQ \leftarrow X \quad B \leftarrow D$
- Si B tiene k ceros adelante, desplazar todos los registros k lugares a izquierda.
- Para $i = 0 \dots n-1$
 - 1) Si los tres MSB de A son iguales: $q_i \leftarrow 0$ y desplazar A|MQ 1 bit a izquierda.
 - 2) Si los tres MSB de A no son iguales y $A < 0$: $q_i \leftarrow -1$, desplazar A|MQ 1 bit a izquierda y sumar B.
 - 3) Si no: $q_i \leftarrow 1$, desplazar A|MQ 1 bit a izquierda y restar B.
- Si el resto final es negativo, sumar B y restar 1 a Q.
- Desplazar el resto final k lugares a derecha.

División SRT

$$X = 8$$

$$D = 3$$

$$X = 1000$$

$$D = 0011$$

D tiene 2 ceros adelante

$$D_{5\text{bits}} \leftarrow 01100$$

$$-D_{5\text{bits}} \leftarrow 10100$$

	A	MQ
	00000	1000
Shift A MQ 2 lugares a izq.	00010	0000
[000] Shift, $q_i = 0$	00100	0000
[001 y $A \geq 0$] Shift, $q_i = 1$	01000	0001
-D	10100	
	11100	0001
[111] Shift, $q_i = 0$	11000	0010
[110 y $A < 0$] Shift, $q_i = -1$	10000	010 $\bar{1}$
+D	01100	
	11100	010 $\bar{1}$
Resto < 0. +D y Q--	01100	
	01000	01 $\bar{1}0$
Shift A 2 lugares a derecha	00010	Q_{DS}
	R	

$$R = 0010$$

$$Q = 0100 - 0010 = 0010$$

División SRT: Explicación

- $X/D = Q$ implica $X = D \cdot Q + R$ ($0 \leq R < D$)
- Suposiciones:
 - Mantisa fraccionaria: punto a la izquierda del MSB.
 - Para registros de n bits:

$$\begin{array}{l} a = X/2^n \\ b = D/2^n \end{array} \quad \frac{X}{D} = \frac{a}{b} = \frac{X/2^n}{D/2^n} = Q \quad \begin{array}{l} r = R/2^n \\ a = b \cdot Q + r \end{array}$$

- b es una fracción normalizada ($b \geq 1/2$)
 - Si b no está normalizado, desplazar a y b k lugares a izquierda para normalizar b .

División SRT: Explicación

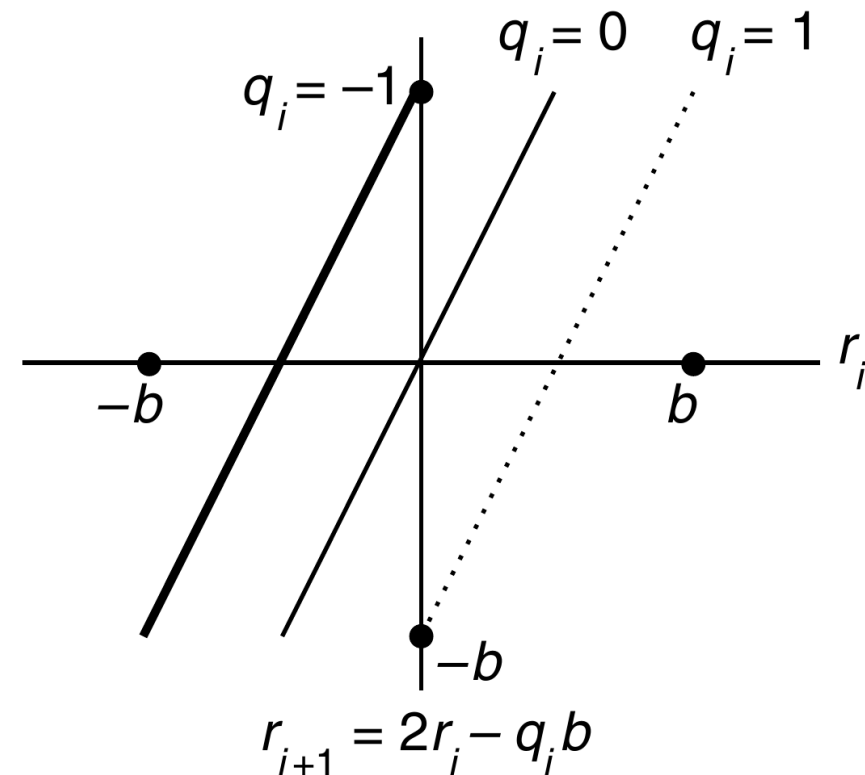
- El resto parcial i

$$r^{(i)} = 2 r^{(i-1)} - q_i b$$

- La selección de los bits del cociente es:

$$q_i = \begin{cases} 1 & \text{si } 2r_{i-1} \geq b \\ 0 & \text{si } -b \leq 2r_{i-1} < b \\ -1 & \text{si } 2r_{i-1} < -b \end{cases}$$

- La dificultad está en comparar b con $2r_{i-1}$



División SRT: Explicación

- Si asumimos b normalizada $\Rightarrow \frac{1}{2} \leq b < 1$
- Eso permite reducir la zona en la que $q_i = 0$ a:

$$-b \leq -\frac{1}{2} \leq 2 r^{(i-1)} < \frac{1}{2} \leq b$$

$$-\frac{1}{4} \leq r^{(i-1)} < \frac{1}{4}$$

$$1.110_2 \leq r^{(i-1)} < 0.01_2$$

$$1.110_2 \leq r^{(i-1)} \leq 0.001_2$$

División SRT: Explicación

$$-1/2 \leq r^{(i)} < 1/2$$

$$1.10_2 \leq r^{(i)} < 0.1_2 \leq 0.01_2$$

$$-1/4 \leq r^{(i-1)} < 1/4$$

$$1.110_2 \leq r^{(i-1)} < 0.01_2$$

$$1.110_2 \leq r^{(i-1)} \leq 0.001_2$$

- En vez de analizar los primeros 3 bits alcanza con los primeros 2 excluyendo el signo.
- Esto permite usar la ALU de n bits.

División SRT

$$X = 8$$

$$D = 3$$

$$X = 1000$$

$$D = 0011$$

D tiene 2 ceros adelante

$$D \leftarrow 1100$$

$$-D \leftarrow 0100$$

	A	MQ
	0000	1000
Shift A MQ 2 lugares a izq.	0010	0000
[00] Shift, $q_i = 0$	0100	0000
[01 y $A \geq 0$] Shift, $q_i = 1$	1000	0001
$-D$	0100	
	1100	0001
[11] Shift, $q_i = 0$	1000	0010
[10 y $A < 0$] Shift, $q_i = -1$	0000	010 $\bar{1}$
$+D$	1100	
	1100	010 $\bar{1}$
	1100	
Resto < 0. $+D$ y $Q--$	1100	
	1000	01 $\bar{1}0$
Shift A 2 lugares a derecha	0010	Q_{DS}

$$R = 0010$$

$$Q = 0100 - 0010 = 0010$$

Acelerar la división

- Para acelerar la multiplicación
 - Carry-save adder
 - Bases mayores a 2 (higher radix)
- En la división
 - Se necesita el signo de A para elegir el próximo dígito y la operación de la ALU.
El CSA no alcanza.
 - Al procesar más de un bit cada dígito del cociente puede tomar varios valores (no solo 0 o 1).
- La solución a ambos es el uso de dígito signado.

CSA

- En SRT el cociente está en dígito signado.
- Se elige el dígito 0 siempre que sea posible para evitar operar con la ALU.
- En la división con CSA el resto parcial no se conoce exactamente
 - No se puede elegir el dígito del cociente con exactitud.

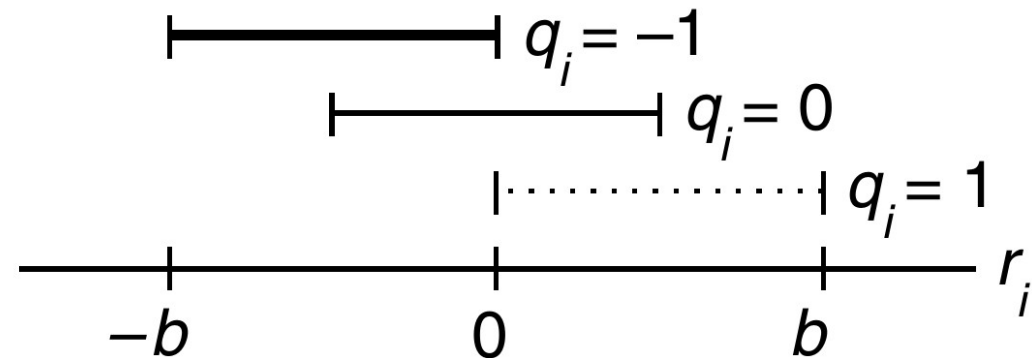
CSA

- Mientras que $|r^{(i+1)}| \leq b$ en

$$r^{(i+1)} = 2r^{(i)} - q_i b$$

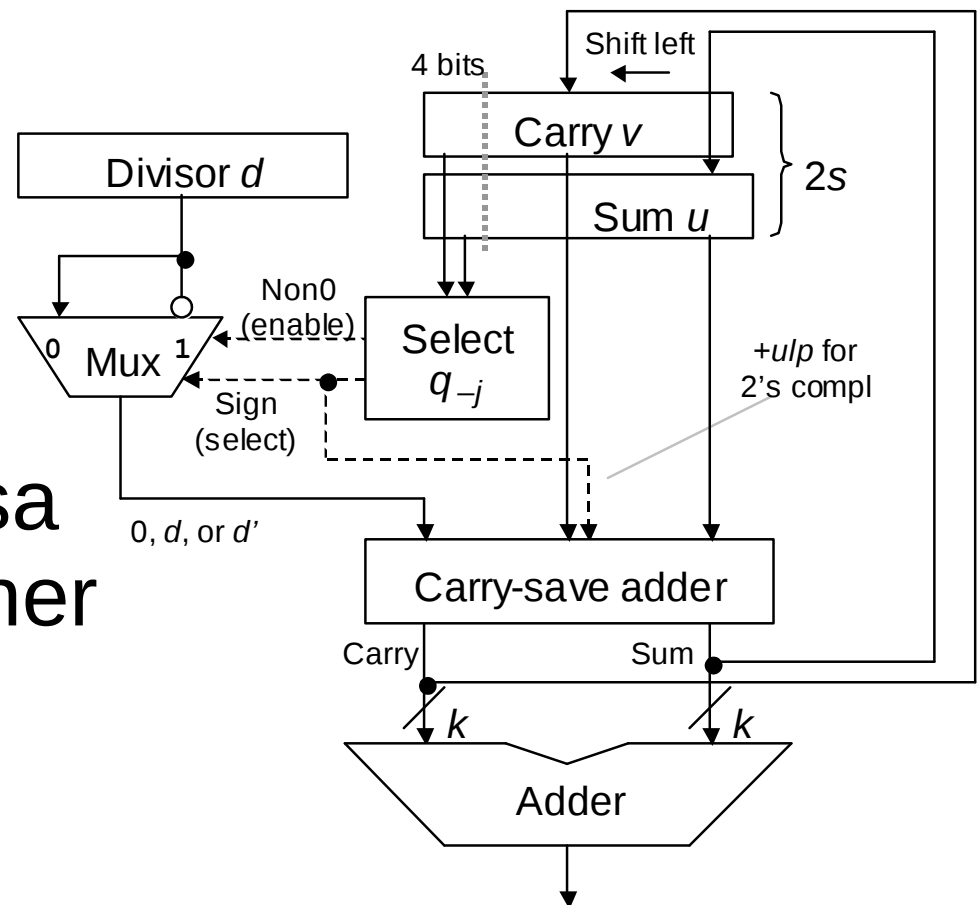
se puede elegir cualquier valor de q_i .

- No hace falta conocer el valor exacto de $r^{(i)}$ para elegir un dígito del cociente.
- Solo hace falta que $r^{(i)}$ esté en un intervalo lo suficientemente pequeño para que esté contenido dentro del rango de un bit del cociente.



CSA

- Alcanza con sumar propagando carry los primeros bits del carry y la semisuma.
- Se usa un sumador de 4 bits para sumar los 4 primeros bits del carry y de la semisuma.
- Los primeros 3 bits de esa suma se usan para obtener el próximo dígitos del cociente.



SRT Base 4

- Los dígitos posibles para el cocientes son -3, -2, -1, 0, 1, 2, 3
- Como el cociente está en dígito signado, los dígitos se pueden restringir a -2, -1, 0, 1, 2
- En cada paso se calcula: $r^{(i+1)} = 4 r^{(i)} - q_i b$
 - Si r_i está cerca de b , entonces $4r_i$ estará cerca de $4b$.
 - El dígito más grande (2) no alcanza para que $|r^{(i+1)}| \leq b$.
 - Si $|r^{(i)}| \leq \frac{2}{3} b$ entonces

$$-4 \frac{2}{3} b - q_j b \leq 4 r^{(i)} - q_i b \leq 4 \frac{2}{3} b - q_i b$$

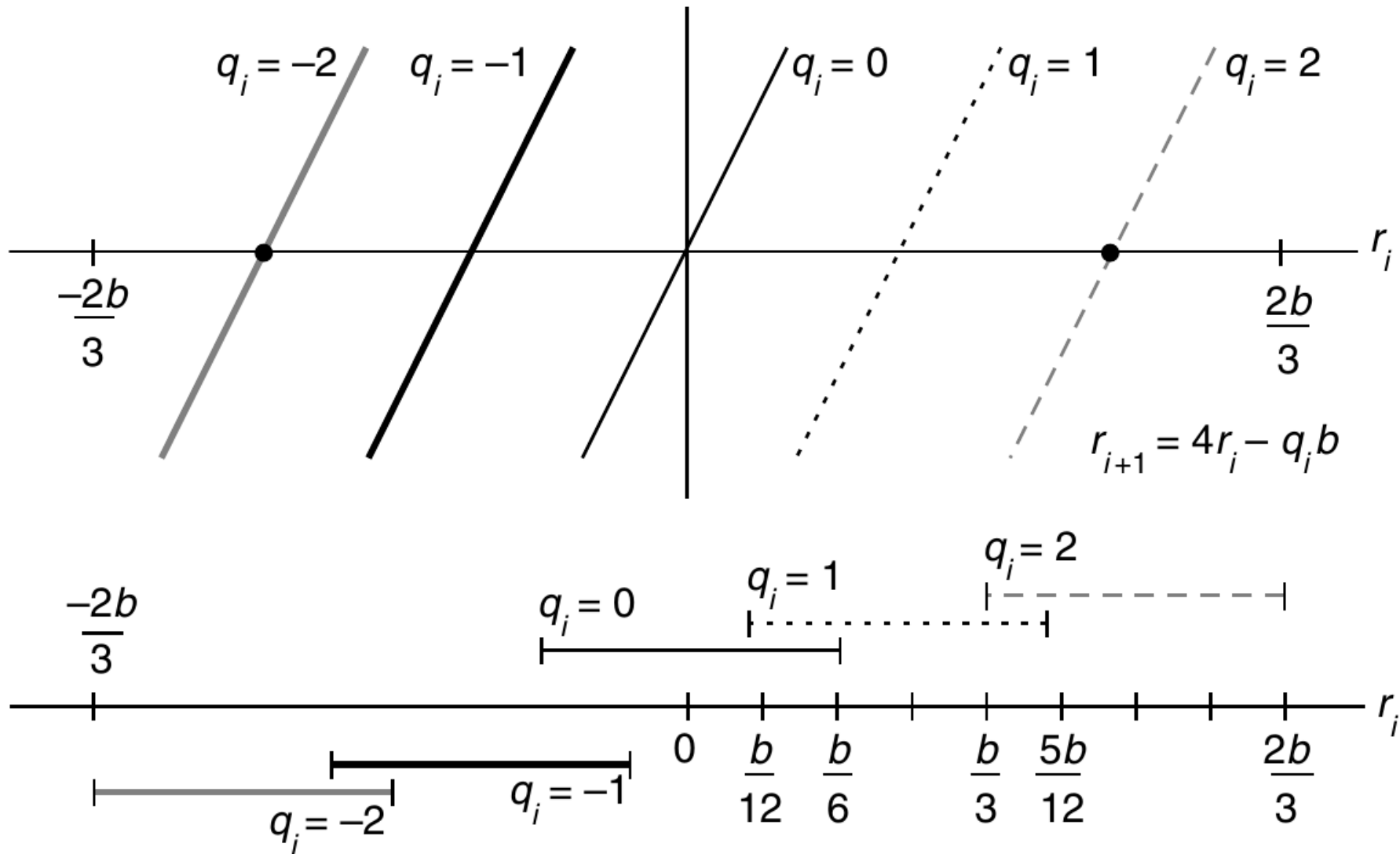
Dígito que maximiza esta cota: -2

$$-4 \frac{2}{3} b - \underbrace{q_j}_b b \leq r^{(i+1)} \leq 4 \frac{2}{3} b - \underbrace{q_i}_b b$$

Dígito que minimiza esta cota: 2

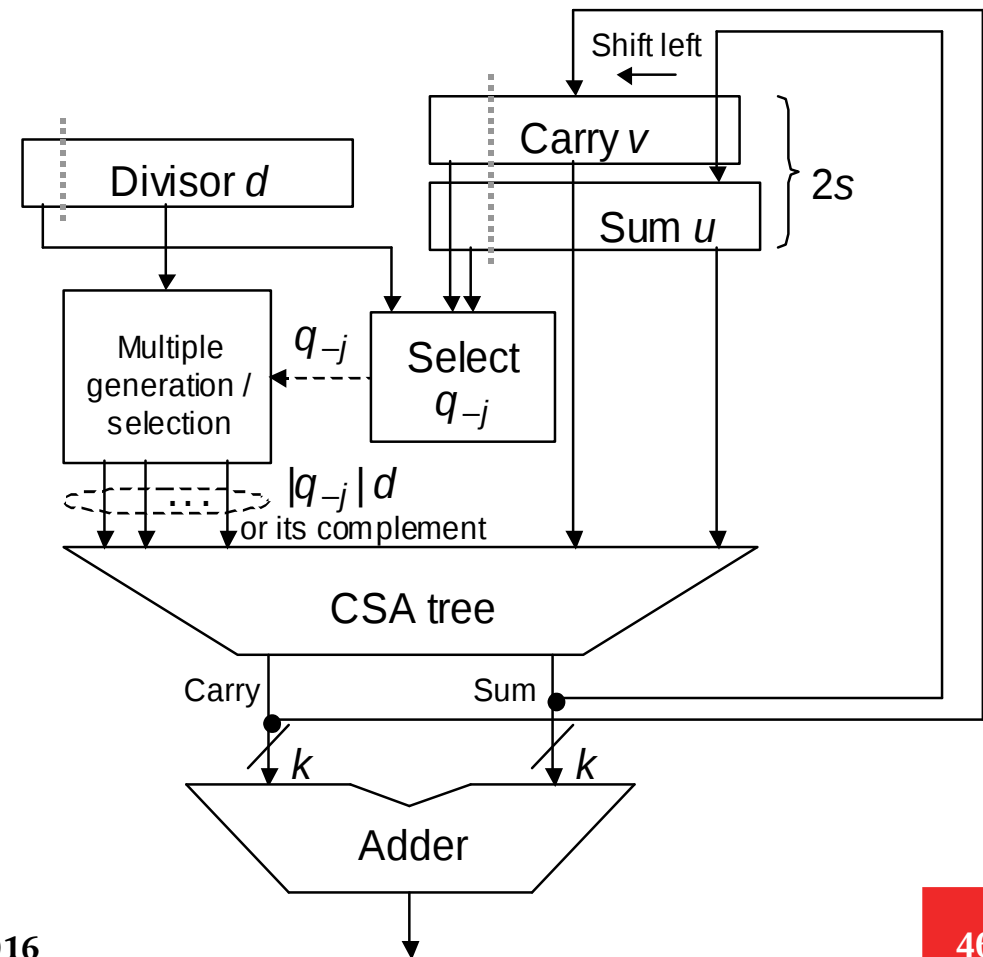
$$-\frac{2}{3} b \leq r^{(i+1)} \leq \frac{2}{3} b$$

SRT Base 4



SRT Base 4

- Para elegir qué dígito corresponde en cada caso, se deben mirar los primeros 6 dígitos de $A|MQ$ y los primeros 4 de b .
- Todas las posibles combinaciones se programan en una *lookup table*.



SRT Base 4

Primeros
4 bits de b

Primeros
6 bits de $A|MQ$

Próximo
dígitos de Q

b	Range of P			q	b	Range of P			q
8	-12	-7	-2	-2	12	-18	-10	-2	
8	-6	-3	-1	-1	12	-10	-4	-1	
8	-2	1	0	0	12	-4	3	0	
8	2	5	1	1	12	3	9	1	
8	6	11	2	2	12	9	17	2	
9	-14	-8	-2	-2	13	-19	-11	-2	
9	-7	-3	-1	-1	13	-10	-4	-1	
9	-3	2	0	0	13	-4	3	0	
9	2	6	1	1	13	3	9	1	
9	7	13	2	2	13	10	18	2	
10	-15	-9	-2	-2	14	-20	-11	-2	
10	-8	-3	-1	-1	14	-11	-4	-1	
10	-3	2	0	0	14	-4	3	0	
10	2	7	1	1	14	3	10	1	
10	8	14	2	2	14	10	19	2	
11	-16	-9	-2	-2	15	-22	-12	-2	
11	-9	-3	-1	-1	15	-12	-4	-1	
11	-3	2	0	0	15	-5	4	0	
11	2	8	1	1	15	3	11	1	
11	8	15	2	2	15	11	21	2	

División SRT

$$X = 149$$

$$D = 5$$

$$X = 10010101$$

$$D = 00000101$$

D tiene 5 ceros adelante

$$D \leftarrow 010100000$$

$$-D \leftarrow 101100000$$

$$2D \leftarrow 101000000$$

$$-2D \leftarrow 011000000$$

i	r_i	q_i	r_{i+1}
10	-15	-9	-2
10	-8	-3	-1
10	-3	2	0
10	2	7	1
10	8	14	2

Shift **A|MQ** 5 lugares a izq.

$$b = 1010$$

[000010] Shift x2, $q = 0$

[001001] Shift x2, $q = 2$

- 2D

[111101 = -3] Shift x2, $q_i = 0$

[110101 = -11] Shift x2, $q = -2$

+2D

Resto negativo.

+D. $Q \leftarrow Q - 1$

Resto desplazado 5 lugares.

A	MQ
000000000	10010101
000010010	10100000
001001010	1000000
100101010	000002
011000000	
111101010	000002
110101000	00020
010100000	020 $\bar{2}$
101000000	
111100000	020 $\bar{2}$
010100000	0203 $\bar{3}$
010000000	Q_{DS}

R

$$R = 0100 = 4$$

$$Q = 00100000 - 11 = 11101 = 29$$

Bibliografía

- Capítulo 13 y 14. *Computer Arithmetic: Algorithms and Hardware Designs.* Behrooz Parhami, Oxford University Press, New York, 2002.
- Capítulo 3 y 8. *Computer Arithmetic Algorithms.* Israel Koren, 2da Edición, A K Peters, Natick, MA, 2002.

Adapted from Koren, UMass. Copyright 2008 Koren, UMass and A.K. Peters.

- Apéndice J. J. Hennessy & D. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers INC. 2011, 5ta Ed.

Suplementaria

- Apéndice B. David A. Patterson & John L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface.* Elsevier. (5ta Ed. 2014)
- Capítulo 43. Editor Wai-Kai Chen. *The VLSI Handbook.* CRC Press. (2da Ed. 2007)