UNIVERSIDAD DE LA RIOJA

Estructura de Computadores

Tema 3:

ARITMÉTICA DEL COMPUTADOR

Dr. Iván Luis Pérez Barrón
Grupo de Computación Científica (GRUCACI)

Contenidos:

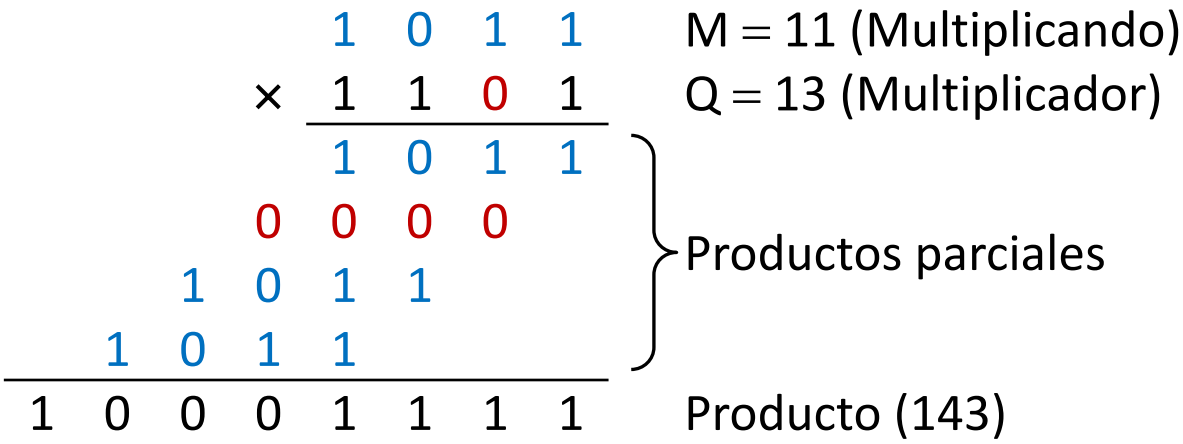
3.3. Aritmética en coma fija

- 3.3.1. Opuesto
- 3.3.2. Suma y resta
- **3.3.3. Multiplicación**
- 3.3.4. División

Contenidos:

3.3.3.1. MUL – Enteros sin signo

- Recordemos el **algoritmo manual** para la multiplicación:



- Se <u>recorren los bits de Q</u> (Q_i) desde el LSB hasta el MSB ($\leftarrow$):
  - Si $Q_i = 0 \rightarrow M \times 0 = 0 \rightarrow$ Producto parcial: 0
  - Si $Q_i = 1 \rightarrow M \times 1 = M \rightarrow$ Producto parcial: M
- Cada producto parcial se <u>desplaza</u> una posición hacia la izquierda.
- Se <u>suman</u> los productos parciales.
- Resultado: su <u>longitud</u> puede ser hasta el doble de la longitud de los factores.

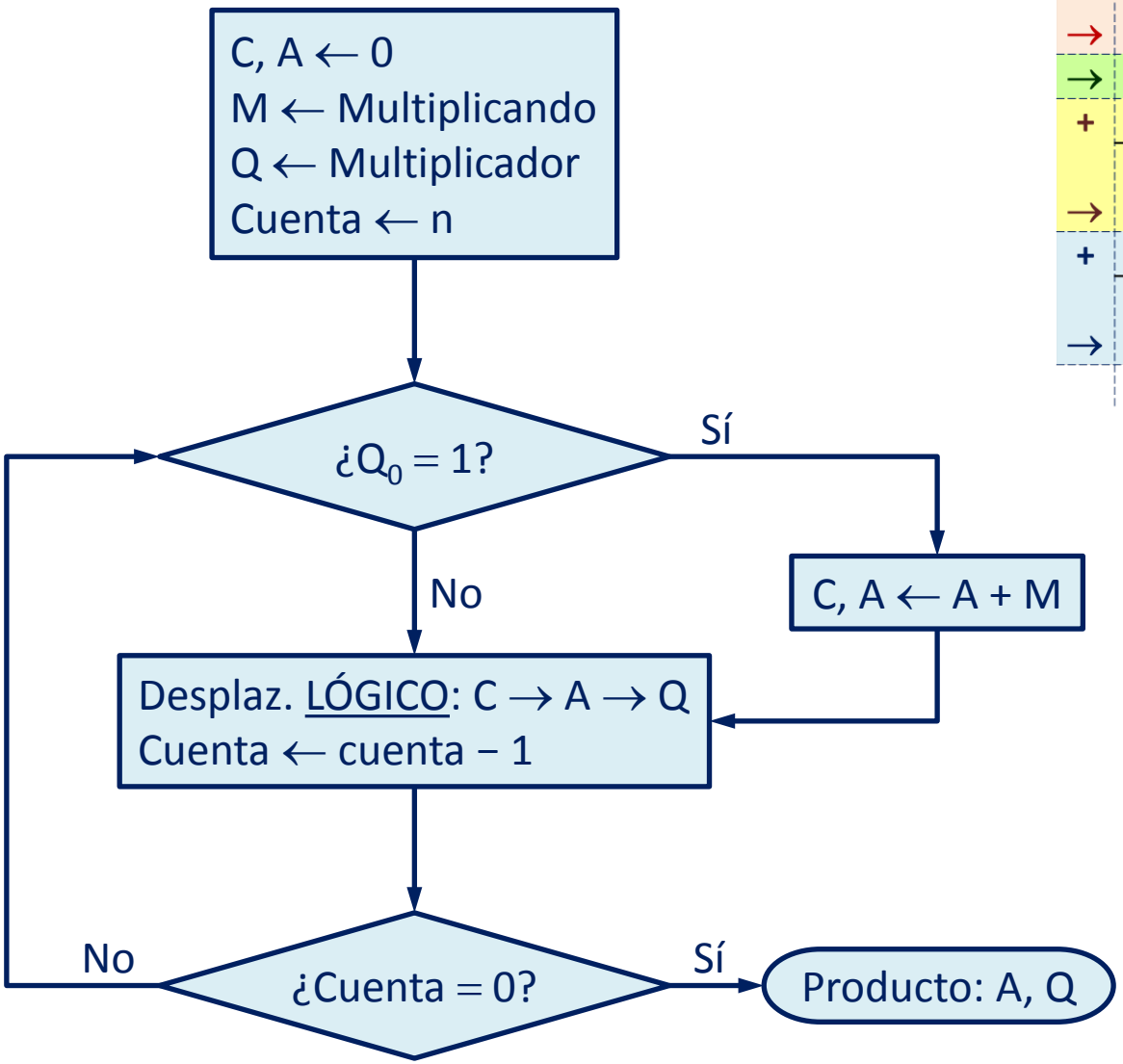
3.3.3.1. MUL – Enteros sin signo

- Conversión en **algoritmo máquina** → Cambios:
  - Los productos parciales se van sumando a medida que se generan.
  - En vez de desplazar cada nuevo producto parcial (←), se desplaza la suma acumulada de los productos parciales previos (→).
  - Si $Q_i = 1$: sumar M y desplazar (→). Si $Q_i = 0$: sólo desplazar (→).

		1	0	1	1				M
×		1	1	0	1				Q (v. inicial)
		0	0	0	0				A (v. inicial)
+		1	0	1	1				
		1	0	1	1				
→		0	1	0	1	1			
→		0	0	1	0	1	1		
+		1	0	1	1				
		1	1	0	1	1	1		
→		0	1	1	0	1	1	1	
+		1	0	1	1				
	1	0	0	0	1	1	1	1	
→		1	0	0	0	1	1	1	1
	C	A (v. final)				Q (v. final)			

3.3.3.1. MUL – Enteros sin signo

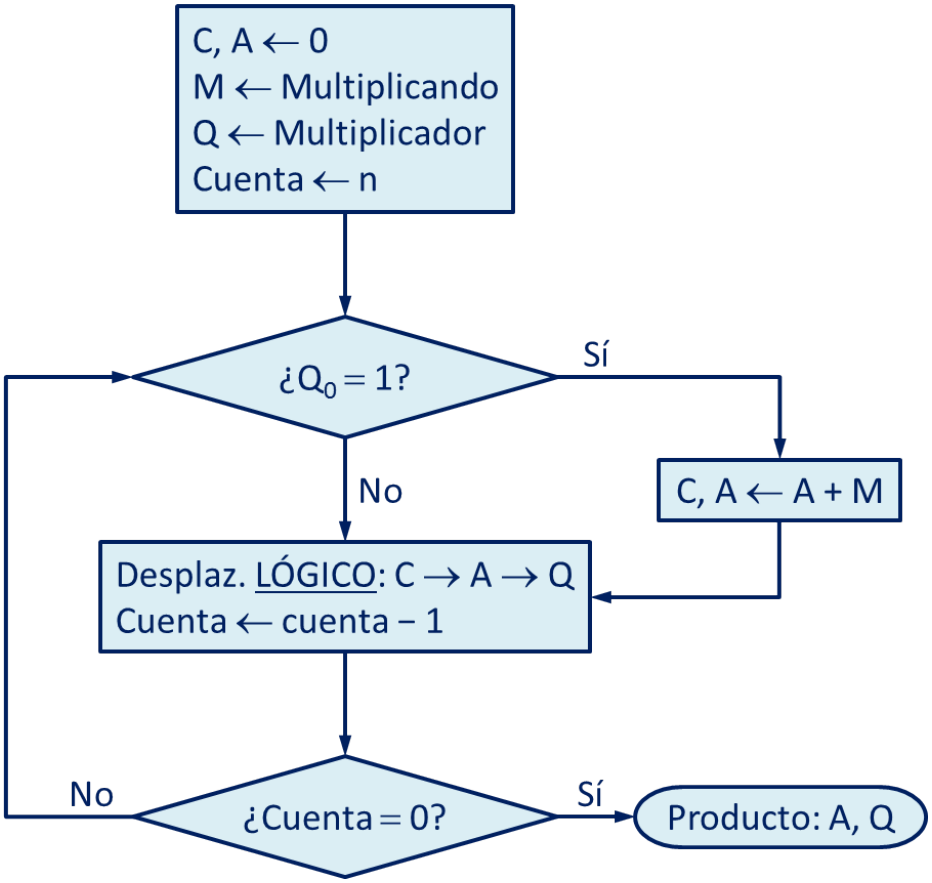
- **Diagrama de flujo**:



		1	0	1	1				M
×		1	1	0	1				Q (v. inicial)
		0	0	0	0				A (v. inicial)
+		1	0	1	1				
		1	0	1	1				
→		0	1	0	1	1			
→		0	0	1	0	1	1		
+		1	0	1	1				
		1	1	0	1	1	1		
→		0	1	1	0	1	1	1	
+		1	0	1	1				
	1	0	0	0	1	1	1	1	
→		1	0	0	0	1	1	1	1
	C	A (v. final)				Q (v. final)			

3.3.3.1. MUL – Enteros sin signo

- **Diagrama de flujo**:

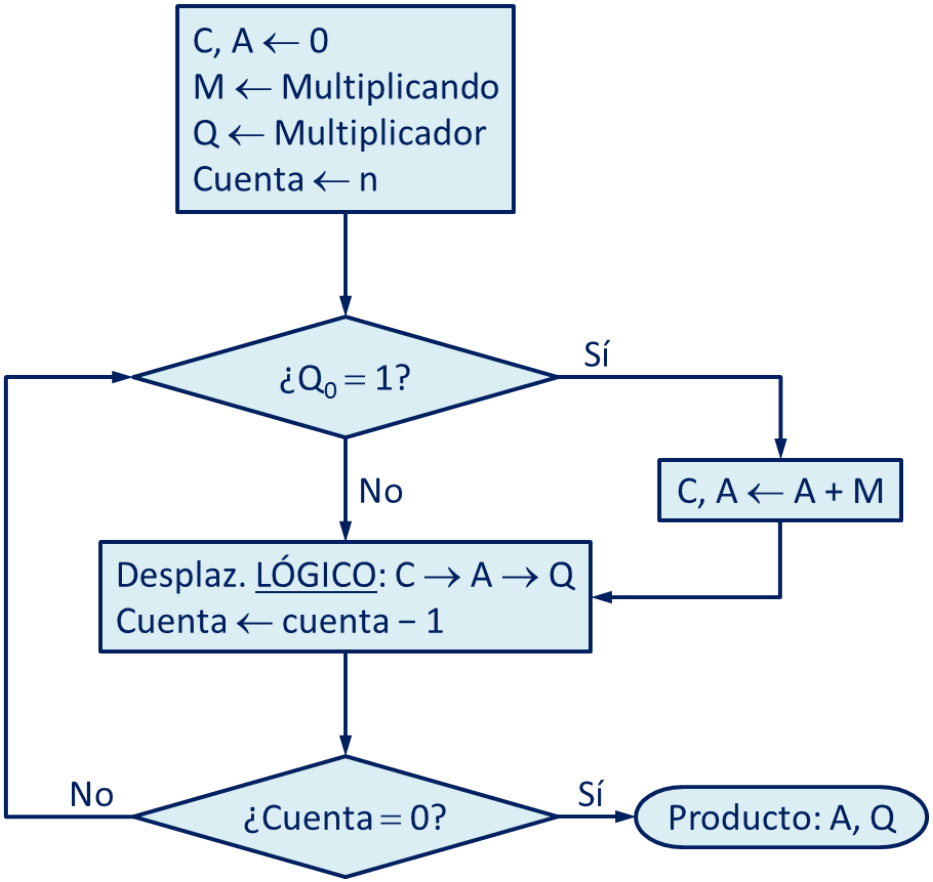


- **C**: acarreos de las sumas.
- **A**: registro auxiliar:
  - Va guardando la suma acumulada.
  - Al final contendrá la parte alta del resultado.
- **n**: longitud de los factores.
- **Q**:
  - Inicialmente: multiplicador.
  - En cada paso: desplazamiento (→). Se expulsa Q_0 y el siguiente bit de Q ocupa su posición.
  - Todos los bits de Q van pasando por Q_0
    - Q_0=1: (+) y (→)
    - Q_0=0: sólo (→)
  - Tras n desplazamientos: Q ha perdido todos los bits originales y contiene la parte baja del resultado.
- **Desplaz. LÓGICO (→)**: el hueco generado se rellena con un cero.

3.3.3.1. MUL – Enteros sin signo

- **Ejemplo**:

	1011	(11)
×	1101	× (13)
	1000 1111	(143)



C	A	Q (Q_0)	M: 1 0 1 1
0	0 0 0 0	1 1 0 **1**	Valores iniciales
0	1 0 1 1	1 1 0 1	C, A ← A + M
0	0 1 0 1	1 1 1 **0**	C → A → Q
0	0 0 1 0	1 1 1 **1**	C → A → Q
0	1 1 0 1	1 1 1 1	C, A ← A + M
0	0 1 1 0	1 1 1 **1**	C → A → Q
1	0 0 0 1	1 1 1 1	C, A ← A + M
0	1 0 0 0	1 1 1 1	C → A → Q

Producto

Contenidos:

3.3.3.2. MUL – Complemento a dos

- A diferencia de la suma, el algoritmo de multiplicación de **enteros sin signo** no resulta válido en C-2:

$$
\begin{array}{rrrrrrrrl}
 & & & 1 & 0 & 1 & 1 & & (M = -5) \\
 & & \times & 1 & 1 & 0 & 1 & & (Q = -3) \\
\hline
 & & & 1 & 0 & 1 & 1 & & \\
 & & 0 & 0 & 0 & 0 & & & \\
 & 1 & 0 & 1 & 1 & & & & \\
1 & 0 & 1 & 1 & & & & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \textbf{(Producto = −113)}
\end{array}
$$

- Para entender la **causa**, hay que tener en cuenta dos cosas:
  - El valor del multiplicador Q = 1101 como entero sin signo viene determinado por una suma de potencias de 2:

$$1101 \rightarrow 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

  - El producto del multiplicando M = 1011 por 2^n se obtiene desplazándolo n posiciones hacia la izquierda (←):

$$1011 \cdot 2^3 \rightarrow 1011000$$

3.3.3.2. MUL – Complemento a dos

- Haciendo explícitos ambos detalles, podemos reescribir la anterior multiplicación de **enteros sin signo** como:

				1	0	1	1	
			×	1	1	0	1	$1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0$
0	0	0	0	1	0	1	1	$1011 \cdot 1 \cdot 2^0$
0	0	0	0	0	0	0	0	$1011 \cdot 0 \cdot 2^1$
0	0	1	0	1	1	0	0	$1011 \cdot 1 \cdot 2^2$
0	1	0	1	1	0	0	0	$1011 \cdot 1 \cdot 2^3$
1	0	0	0	1	1	1	1	

- Razones por las que este procedimiento **no es válido en C-2**:
  - La extensión de la longitud de los productos parciales se ha hecho incluyendo ceros, pero en C-2 hay que replicar el bit de signo.
  - En C-2, el peso del bit de signo es el opuesto del que corresponde a su misma posición en enteros sin signo, por lo que habría que hacer el opuesto del último producto parcial. En este caso:

$$\mathbf{1}101 \rightarrow 1\cdot\mathbf{(-2^3)} + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0$$

3.3.3.2. MUL – Complemento a dos

- Tomando ambas precauciones, podríamos **adaptar el algoritmo** de multiplicación de enteros sin signo a C-2:

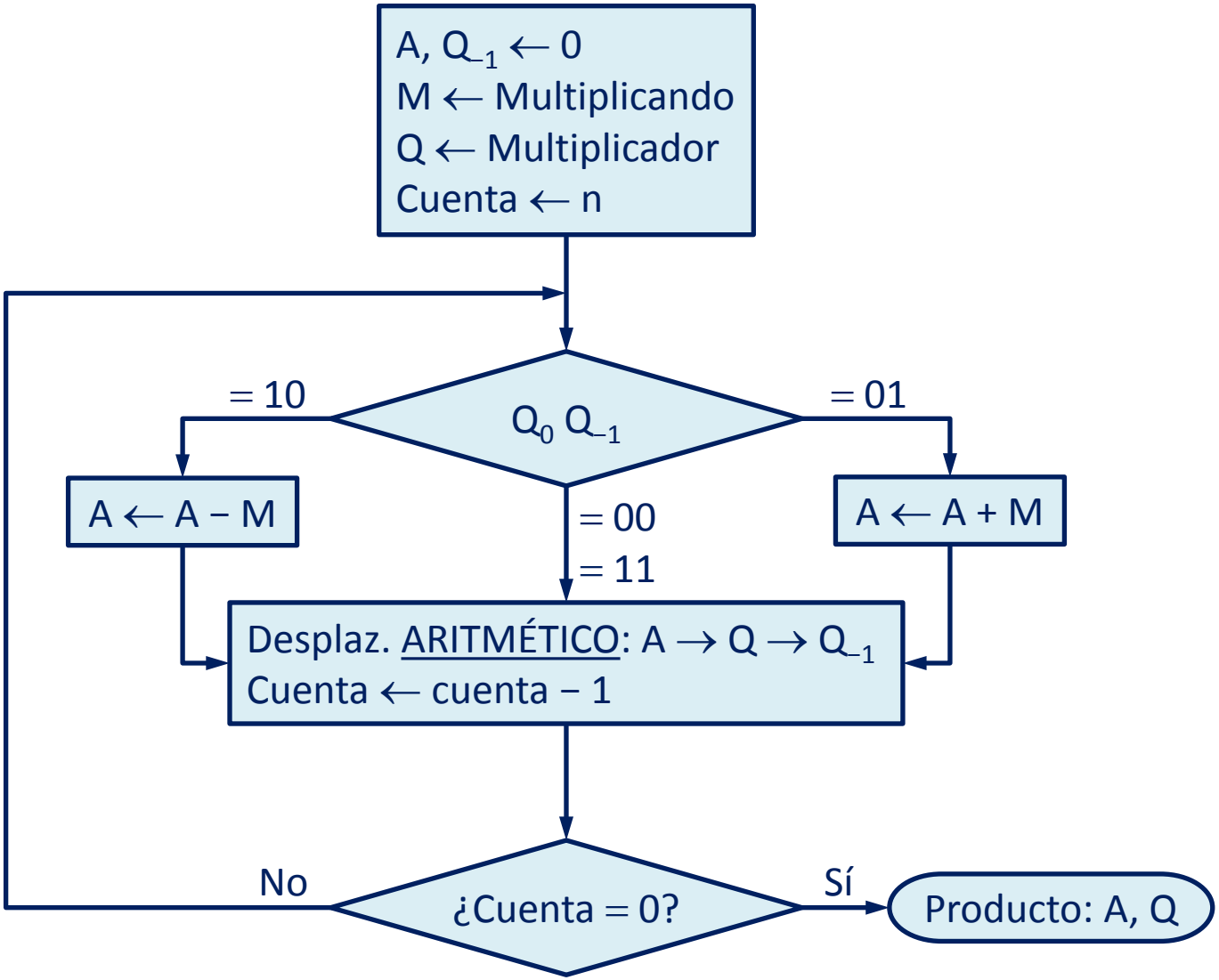
(−5)					1	0	1	1	
(−3)				×	1	1	0	1	$1\cdot(-2^3) + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0$
	1	1	1	1	1	0	1	1	$1011 \cdot 1 \cdot 2^0$
	0	0	0	0	0	0	0	0	$1011 \cdot 0 \cdot 2^1$
	1	1	1	0	1	1	0	0	$1011 \cdot 1 \cdot 2^2$
	0	0	1	0	1	0	0	0	$1011 \cdot 1 \cdot (-2^3)$
(+15)	~~1~~ (C)	0	0	0	0	1	1	1	1	

- **Otra alternativa** posible:
  - Convertir multiplicando M y multiplicador Q en positivos.
  - Aplicar el algoritmo de multiplicación para enteros sin signo.
  - Hacer el opuesto del resultado si M y Q tenían distinto signo.

- Sin embargo, hay una opción aun mejor, que además acelera el proceso: el **algoritmo de Booth** para la multiplicación en C-2.

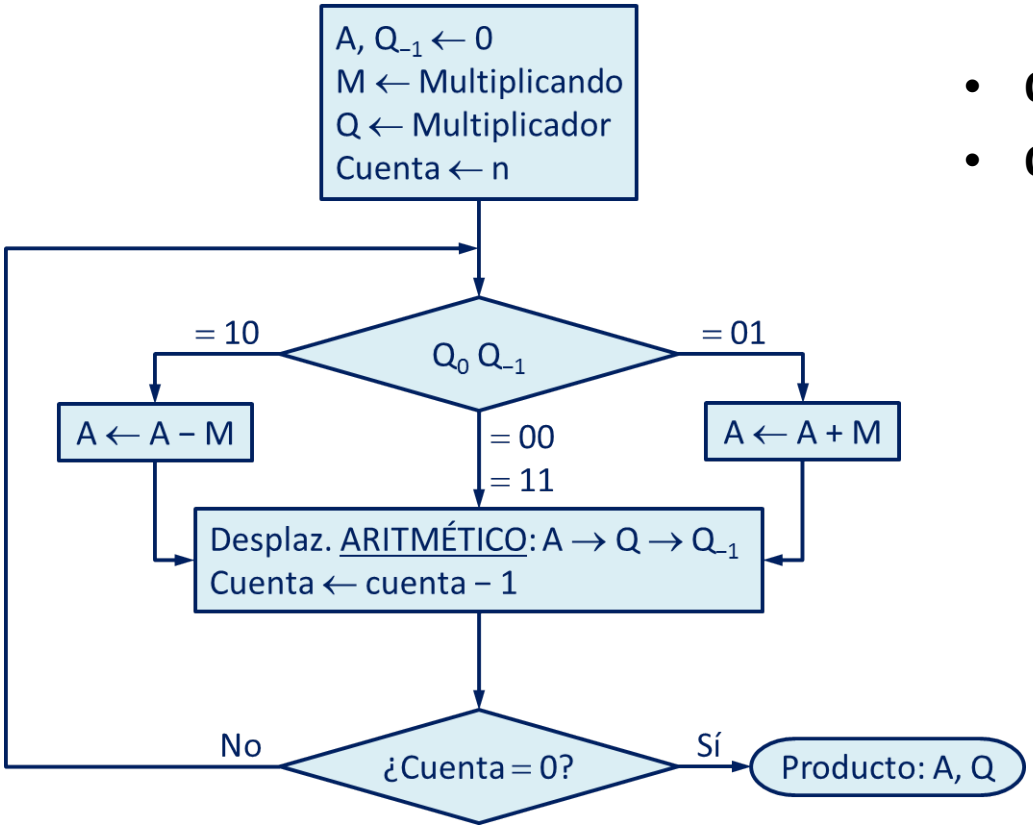
3.3.3.2. MUL – Complemento a dos

- **Algoritmo de Booth**:



3.3.3.2. MUL – Complemento a dos

- **Algoritmo de Booth**:

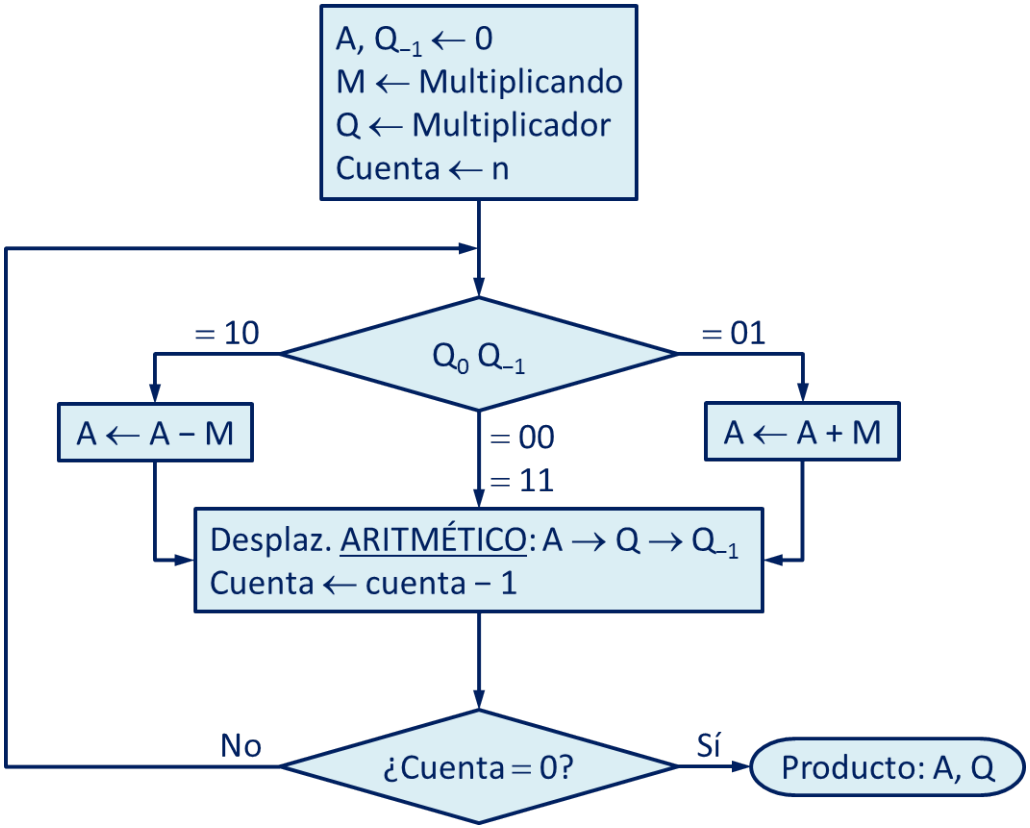


- **C-2**:
  - Los acarreos se ignoran.
  - Resta: suma del opuesto.
- **A**: registro auxiliar:
  - Va guardando la suma acumulada.
  - Al final: parte alta del resultado.
- **Q_{-1}**: bit auxiliar a la derecha de Q_0.
- **Q**:
  - Inicialmente: multiplicador.
  - En cada paso: desplazamiento (→). Se expulsa Q_0, que pasa a Q_{-1}, y el siguiente bit de Q ocupa su posición.
  - Todos los bits de Q van pasando por Q_0 y Q_{-1}:
    - $Q_0\ Q_{-1} = 01$: (+) y (→)
    - $Q_0\ Q_{-1} = 10$: (−) y (→)
    - $Q_0\ Q_{-1} = 00, 11$: sólo (→)
  - Tras n desplazamientos: Q ha perdido todos los bits originales y contiene la parte baja del resultado.
- **Desplaz. ARITMÉTICO (→)**: para conservar el signo, el hueco se rellena con BS.

3.3.3.2. MUL – Complemento a dos

- **Algoritmo de Booth**: ejemplo.

	1001	(−7)
×	0011	× (+3)
	1110 1011	(−21)



M: 1 0 0 1

A	Q	(Q_0) Q_{-1}	
0 0 0 0	0 0 1 **1**	**0**	Valores iniciales
0 1 1 1	0 0 1 1	0	$A \leftarrow A - M$
0 0 1 1	1 0 0 **1**	**1**	$A \rightarrow Q \rightarrow Q_{-1}$
0 0 0 1	1 1 0 **0**	**1**	$A \rightarrow Q \rightarrow Q_{-1}$
1 0 1 0	1 1 0 0	1	$A \leftarrow A + M$
1 1 0 1	0 1 1 **0**	**0**	$A \rightarrow Q \rightarrow Q_{-1}$
1 1 1 0	1 0 1 1	0	$A \rightarrow Q \rightarrow Q_{-1}$

Producto

3.3.3.2. MUL – Complemento a dos

- **Algoritmo de Booth**: justificación.

  - Se basa en una **propiedad** ya conocida:
    La suma de una serie de potencias de 2 consecutivas es igual a la potencia siguiente a la mayor menos la potencia menor:

$$2^{n} + 2^{n-1} + \ldots + 2^{n-k} = 2^{n+1} - 2^{n-k}$$

  - Aplicado a la **multiplicación**, por ejemplo, M × (00011110):
    - Algoritmo de enteros sin signo: $M \times (2^4 + 2^3 + 2^2 + 2^1) = M \times 30$
    - Algoritmo de Booth: $M \times (2^5 - 2^1) = M \times 30$

  - ¿Cómo detectar el inicio y el final del **bloque de unos**?
    - Inicio: 00**01**1110 → **+**2^5 (**01** → suma)
    - Final: 0001111**10** → **–**2^1 (**10** → resta)

3.3.3.2. MUL – Complemento a dos

- **Algoritmo de Booth**: justificación.

		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
M	×	0	0	**0**	**1**	1	1	**1**	**0**	
Alg. enteros sin signo:					$+2^4$	$+2^3$	$+2^2$	$+2^1$		$= 30$
				↓				↓		
Alg. Booth:				$+2^5$				-2^1		$= 30$

- **Aceleración del proceso**:
  - Algoritmo de enteros sin signo:
    - Una suma por cada 1 en el multiplicador.
  - Algoritmo de Booth:
    - Una suma y una resta por cada bloque de unos en el multiplicador.
    - Los bloques de unos o de ceros se saltan.

Contenidos:

3.3. Aritmética en coma fija

- 3.3.1. Opuesto
- 3.3.2. Suma y resta
- 3.3.3. Multiplicación
- **3.3.4. División**

Contenidos:

3.3.4.1. DIV – Enteros sin signo

- Durante el proceso de la división se hacen restas.
- **¿Cómo se restan los enteros sin signo?**
  - M − S = R → Buscar el número R que, sumado a S, dé lugar a M:

$$\begin{array}{r} 0101 \\ -\ 0011 \end{array} \quad \rightarrow \quad \begin{array}{r} 0101 \\ -\ 0011 \\ \hline 0010 \\ \leftarrow \end{array} \qquad \begin{array}{r} (5) \\ -\ (3) \\ \hline (2) \end{array}$$

    - ¿Qué sumar a 1 para que dé 1? **0**
    - ¿Qué sumar a 1 para que dé 0? **1**, y nos llevamos 1 (1+1 = 10)
    - ¿Qué sumar a 0 y 1 que nos llevábamos para que dé 1? **0**
    - ¿Qué sumar a 0 para que dé 0? **0**
  - Los acarreos que se generan en la resta se denominan adeudos o acarreos negativos.

3.3.4.1. DIV – Enteros sin signo

- (…) **¿Cómo se restan los enteros sin signo?**
  - M − S = R → ¿Qué sucede cuando M < S?

$$\begin{array}{r} 0101 \\ -\ 1001 \end{array} \quad \rightarrow \quad \begin{array}{r} 1\ 0101 \\ -\ 1001 \\ \hline 1100 \\ \leftarrow \end{array} \qquad \begin{array}{r} (5) \\ -\ (9) \\ \hline (12) \end{array}$$

    - ¿Qué sumar a 1 para que dé 1? **0**
    - ¿Qué sumar a 0 para que dé 0? **0**
    - ¿Qué sumar a 0 para que dé 1? **1**
    - ¿Qué sumar a 1 para que dé 0? **1**, y nos llevamos **1** (1+1 = 10)
  - Si se genera adeudo al final de la resta, el resultado es negativo.
  - Aunque el resultado es correcto si se interpreta en C-2 (1100 = −4), no lo es como entero sin signo (1100 = 12).
  - Interpretación del adeudo obtenido como entero sin signo:

$$\begin{array}{r} 1\ 0101 \\ -\ 1001 \\ \hline 1100 \end{array} \qquad \begin{array}{r} (21) \\ -\ (\ 9) \\ \hline (12) \end{array}$$

3.3.4.1. DIV – Enteros sin signo

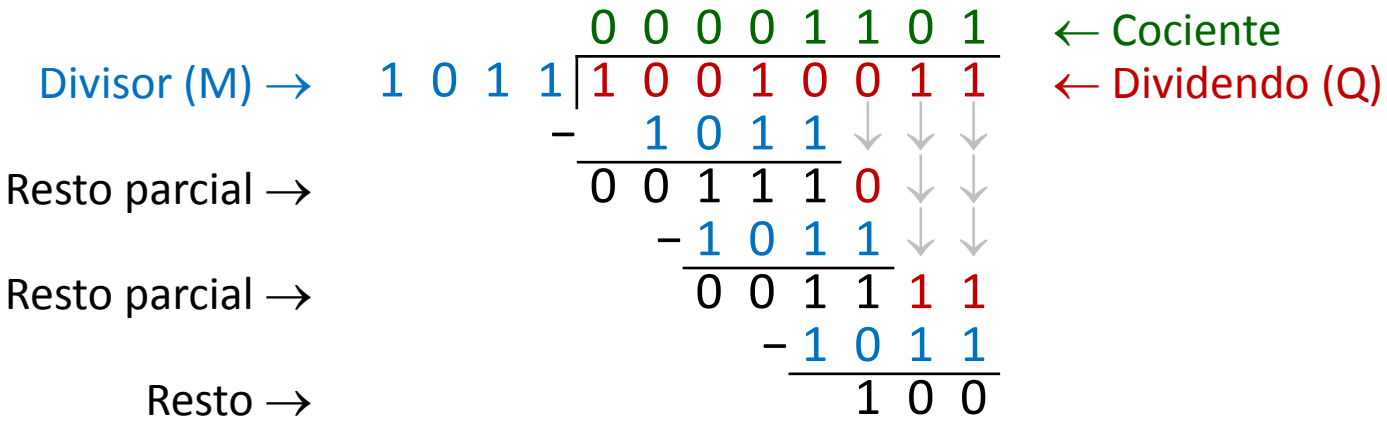
- Recordemos el **algoritmo manual** para la división:

```
  1 4 7 | 1 1
- 1 1   | 1 3
    3 7
  - 3 3
      4
```

```
                        0 0 0 0 1 1 0 1   ← Cociente
Divisor (M) →   1 0 1 1 | 1 0 0 1 0 0 1 1 ← Dividendo (Q)
                      - 1 0 1 1
Resto parcial →       0 0 1 1 1 0
                        - 1 0 1 1
Resto parcial →         0 0 1 1 1 1
                            - 1 0 1 1
Resto →                     1 0 0
```

3.3.4.1. DIV – Enteros sin signo

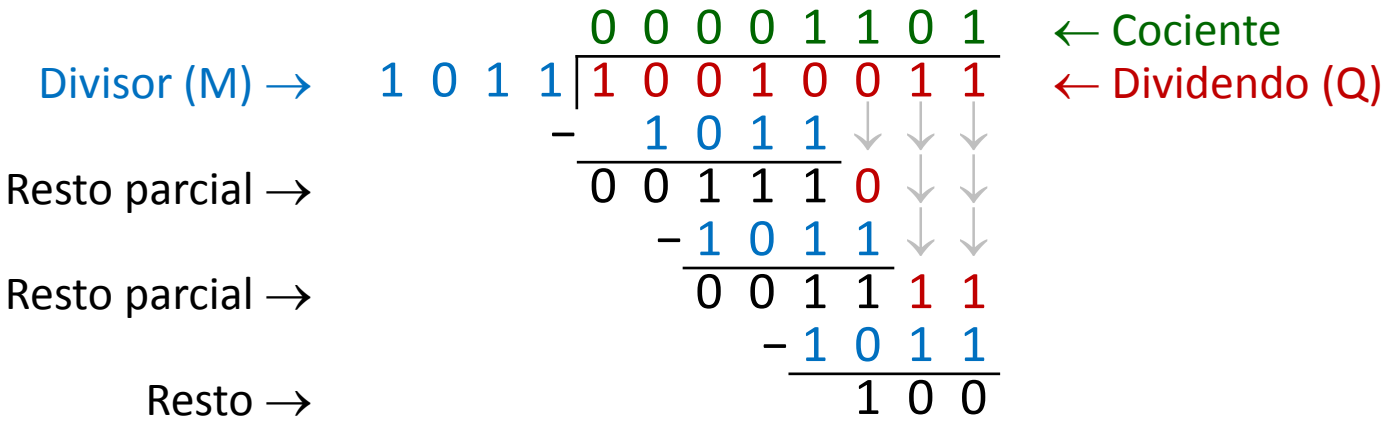
- Recordemos el **algoritmo manual** para la división:



- Se van **tomando bits de Q** empezando por la izquierda (→).
  - Si el <u>número < M</u>: cociente = 0.
    - No se resta nada (M·0 = 0).
  - Si el <u>número > M</u>: cociente = 1.
    - Se resta M·1 = M.
- La operación **finaliza** cuando se agotan los bits de Q.
- **Resto**: resultado de la última resta.

3.3.4.1. DIV – Enteros sin signo

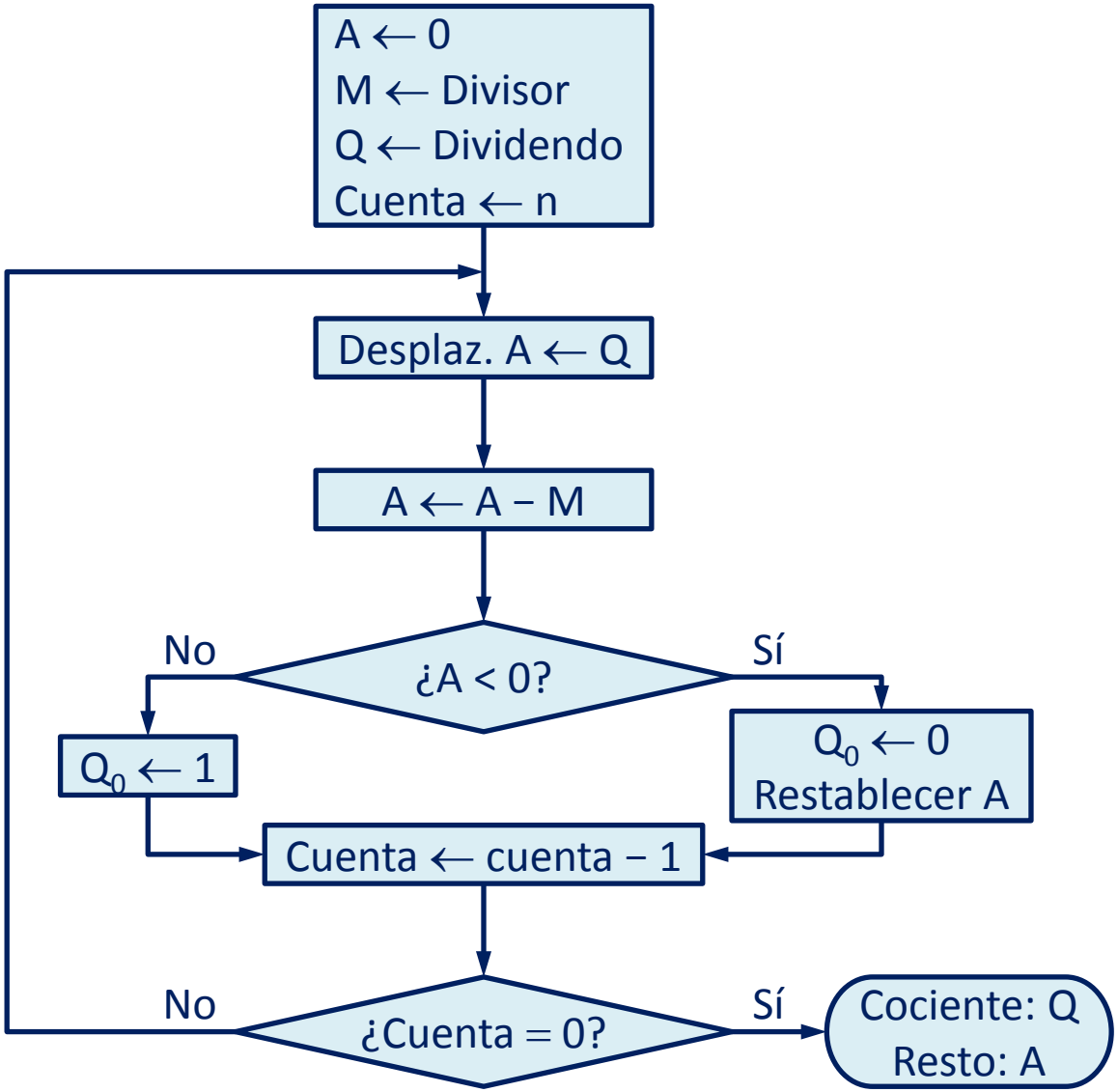
- Recordemos el **algoritmo manual** para la división:



- **Simplificaciones** respecto de la división en sistema decimal:
  - El proceso de tanteo resulta innecesario, pues el cociente sólo admite ceros o unos.
  - No hace falta efectuar multiplicaciones para determinar los números a restar, que sólo pueden ser $M \cdot 1 = M$ ó $M \cdot 0 = 0$ (resta innecesaria).

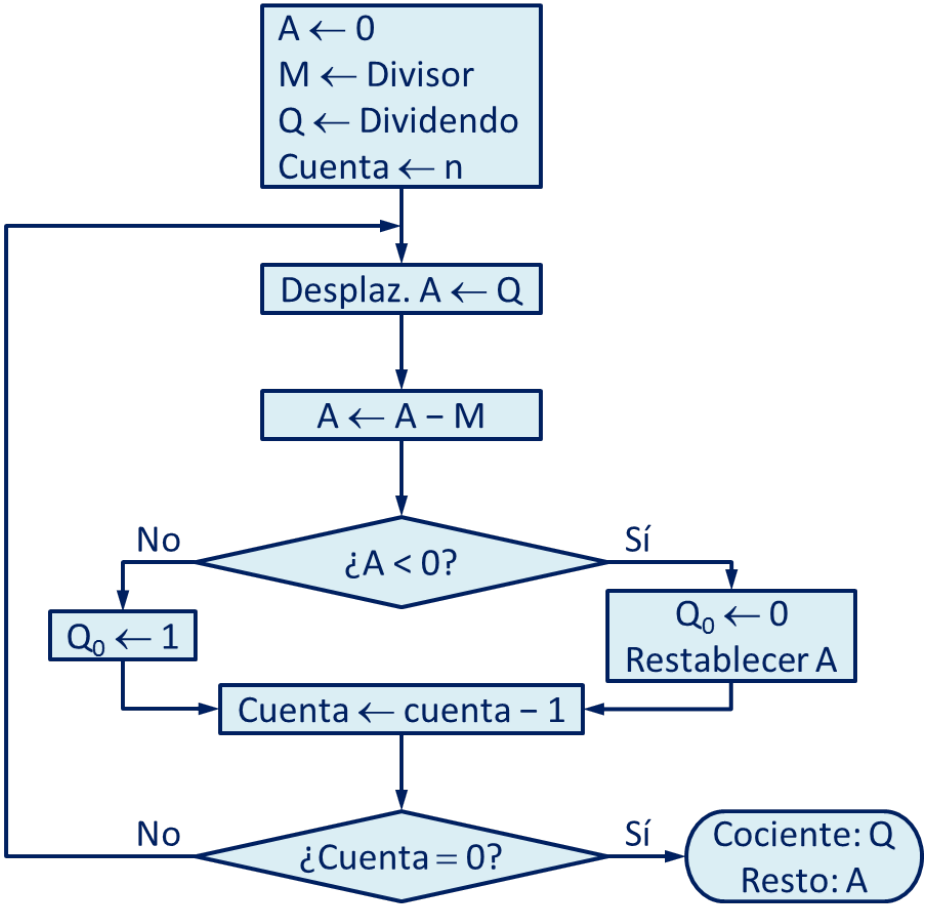
3.3.4.1. DIV – Enteros sin signo

- Conversión en **algoritmo máquina**:



3.3.4.1. DIV – Enteros sin signo

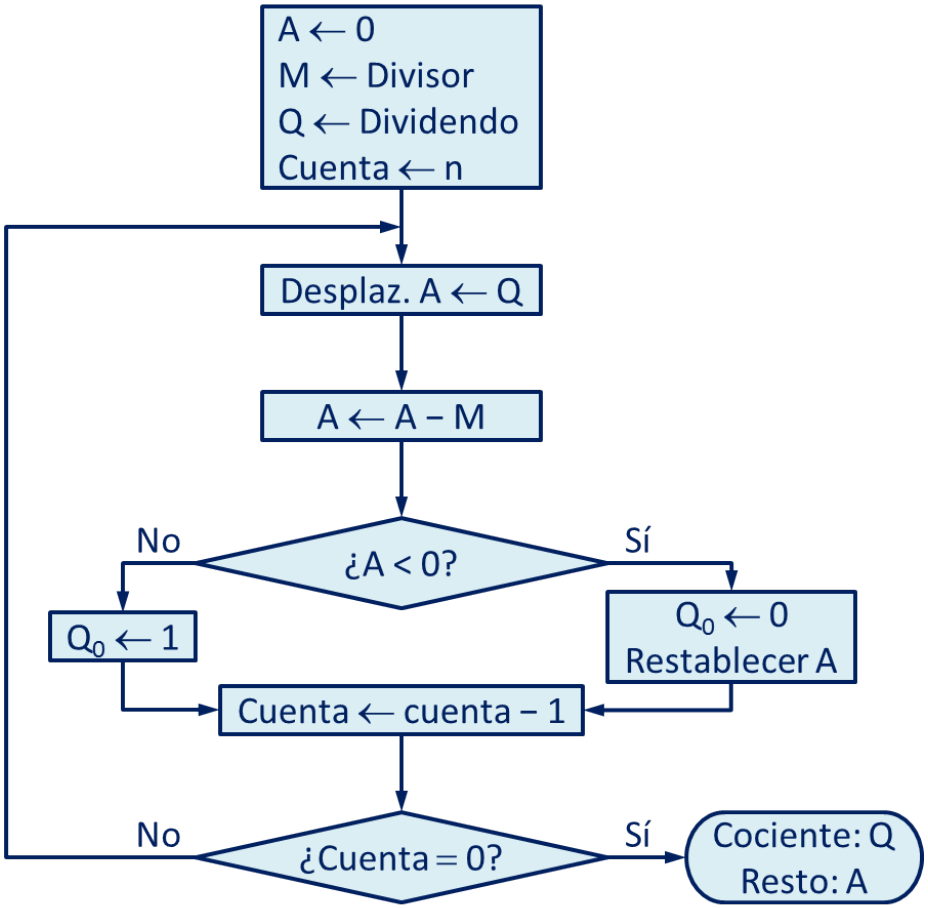
- **Algoritmo máquina**:



- **Q**:
  - Inicialmente: dividendo.
  - En cada paso: desplazamiento (←).
    - Los bits del dividendo van pasando a A, comenzando por los más significativos.
    - Los huecos generados por la derecha se aprovechan para ir guardando los bits del cociente.
  - Tras n desplazamientos: Q ha perdido todos los bits originales y contiene el cociente.
- **A**: registro auxiliar:
  - Va recibiendo bits de Q.
  - En cada paso: A se compara con el divisor M, haciendo la resta A−M y comprobando si se produce adeudo:
    - Si A > M: cociente (Q_0) = 1.
      - La resta ya ha sido hecha.
    - Si A < M: cociente (Q_0) = 0.
      - Se deshace la resta.
  - Al final: resto (resultado última resta).

3.3.4.1. DIV – Enteros sin signo

- **Algoritmo máquina**:

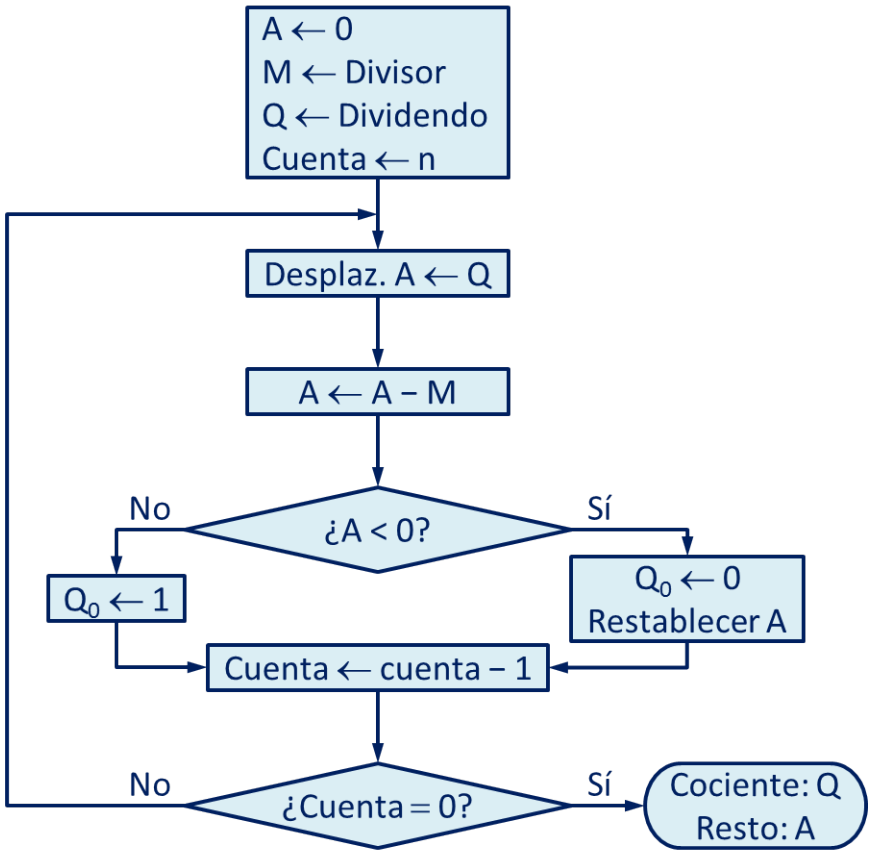


- **Diferencias MUL – DIV**:
  - Desplazamientos:
    - **MUL**: (→)
      - Se desplaza la suma acumulada de los productos parciales anteriores.
    - **DIV**: (←)
      - Se van tomando bits del dividendo comenzando por los más significativos.
  - Orden desp. – operación (suma/resta):
    - **MUL**:
      1. Operación.
      2. Desplazamiento, para dejar desplazada la suma acumulada.
    - **DIV**:
      1. Desplazamiento, para tomar el siguiente bit del dividendo.
      2. Operación, para comprobar si el número es mayor que el divisor.

3.3.4.1. DIV – Enteros sin signo

- **Ejemplo**:

```
1101 |0101        13 | 5
0011  0010          3   2
```



C	A	Q (Q_0)	M: 0 1 0 1
	0 0 0 0	1 1 0 1	Valores iniciales
1	0 0 0 1	1 0 1 0	A ← Q
↑	– 0 1 0 1		A ← A – M
↑	← 1 1 0 0		
	0 0 0 1	1 0 1 **0**	Restablecer A
1	0 0 1 1	0 1 0 0	A ← Q
↑	– 0 1 0 1		A ← A – M
↑	← 1 1 1 0		
	0 0 1 1	0 1 0 **0**	Restablecer A
	0 1 1 0	1 0 0 0	A ← Q
	– 0 1 0 1		A ← A – M
	0 0 0 1	1 0 0 **1**	Q_0 ← 1
1	0 0 1 1	0 0 1 0	A ← Q
↑	– 0 1 0 1		A ← A – M
↑	← 1 1 1 0		
	0 0 1 1	0 0 1 **0**	Restablecer A
	↓ Resto	↓ Cociente	

Contenidos:

3.3.4.2. DIV – Complemento a dos

- **Adaptaciones** al algoritmo de división de enteros sin signo para C-2:
  - **A** debe ser inicialmente la extensión del dividendo Q, por lo que no se rellena con ceros, sino con el bit de signo de Q.
  - La **operación** para comparar, en cada paso, el dividendo parcial con el divisor M ya no es siempre una resta:
    - Si A y M tienen igual signo: $A \leftarrow A-M$ (C-2: sumar opuesto).
    - Si A y M tienen distinto signo: $A \leftarrow A+M$.

    De este modo, el resultado de la operación siempre va a ser más cercano a 0 que A, que es lo que debe suceder con los restos.

    Recordar en ambos casos que en C-2 se ignoran los acarreos.
  - Se introduce un **caso especial** en el que el bit Q_0 a incluir en el cociente puede ser 1 (véase el ejemplo 2 más adelante):
    - Si A no cambia de signo en la operación ($A \leftarrow A-M$ o $A \leftarrow A+M$) o si se cumple [$A = 0$ y $Q = 0$]: hacer $Q_0 \leftarrow 1$.
    - En caso contrario: hacer $Q_0 \leftarrow 0$ y restablecer el valor de A previo a la operación.

3.3.4.2. DIV – Complemento a dos

- (…) **Adaptaciones** al algoritmo de división de enteros sin signo para C-2:
  - **Resultado**:
    - Q: valor absoluto del cociente.
      - Si dividendo y divisor tenían distinto signo → hacer el opuesto de Q.
    - A: resto con signo.
      - El signo del resto es aquel que verifica:

$$\begin{array}{c|c} D & d \\ \hline r & q \end{array} \quad \rightarrow \quad D = d \cdot q + r$$

$$\begin{array}{r|l} +7 & +3 \\ \hline +1 & +2 \end{array} \qquad \begin{array}{r|l} -7 & +3 \\ \hline -1 & -2 \end{array} \qquad \begin{array}{r|l} +7 & -3 \\ \hline +1 & -2 \end{array} \qquad \begin{array}{r|l} -7 & -3 \\ \hline -1 & +2 \end{array}$$

      - Por lo tanto, el signo del resto es siempre igual al signo del dividendo.

3.3.4.2. DIV – Complemento a dos

- **Ejemplo 1**:

1001	0011	−7	+3
1111	1110	−1	−2

C	A	Q (Q_0)	M: 0 0 1 1
	1 1 1 1	1 0 0 1	Valores iniciales
	1 1 1 1	0 0 1 0	A ← Q
	+ 0 0 1 1		A ← A + M
1	0 0 1 0		
	1 1 1 1	0 0 1 **0**	Restablecer A
	1 1 1 0	0 1 0 0	A ← Q
	+ 0 0 1 1		A ← A + M
1	0 0 0 1		
	1 1 1 0	0 1 0 **0**	Restablecer A
	1 1 0 0	1 0 0 0	A ← Q
	+ 0 0 1 1		A ← A + M
	1 1 1 1	1 0 0 **1**	Q_0 ← 1
	1 1 1 1	0 0 1 0	A ← Q
	+ 0 0 1 1		A ← A + M
1	0 0 1 0		
	1 1 1 1	0 0 1 **0**	Restablecer A
	Resto	|Cociente|	

$\frac{(-)}{(+)} = (-) \rightarrow$ Hacer el opuesto de Q

↓

Cociente:

0 0 1 0 → **1 1 1 0**

3.3.4.2. DIV – Complemento a dos

- **Ejemplo 2**:

1100	0001	−4	+1
0000	1100	0	−4

C	A	Q (Q_0)	M: 0 0 0 1
	1 1 1 1	1 1 0 0	Valores iniciales
	1 1 1 1	1 0 0 0	A ← Q
	+ 0 0 0 1		A ← A + M
~~1~~	0 0 0 0		
	1 1 1 1	1 0 0 **0**	Restablecer A
	1 1 1 1	0 0 0 0	A ← Q
	+ 0 0 0 1		A ← A + M
~~1~~	0 0 0 0	0 0 0 **1**	Q_0 ← 1 (**A=Q=0**)
	0 0 0 0	0 0 1 0	A ← Q
	− 0 0 0 1		A ← A − M
	(+ 1 1 1 1)		
	1 1 1 1		
	0 0 0 0	0 0 1 **0**	Restablecer A
	0 0 0 0	0 1 0 0	A ← Q
	− 0 0 0 1		A ← A − M
	(+ 1 1 1 1)		
	1 1 1 1		
	0 0 0 0	0 1 0 **0**	Restablecer A
	Resto	|Cociente|	

$\frac{(-)}{(+)} = (-) \rightarrow$ Hacer el opuesto de Q

↓

Cociente:

0 1 0 0 → **1 1 0 0**