EMI

- MATERIA : ANALISIS DISENO DE SISTEMAS BASADOS EN MICROPROCESADORES
- CODIGO : SIS – 03 2 12
- DOCENTE : CESAR MARTIN SUAREZ SUAREZ

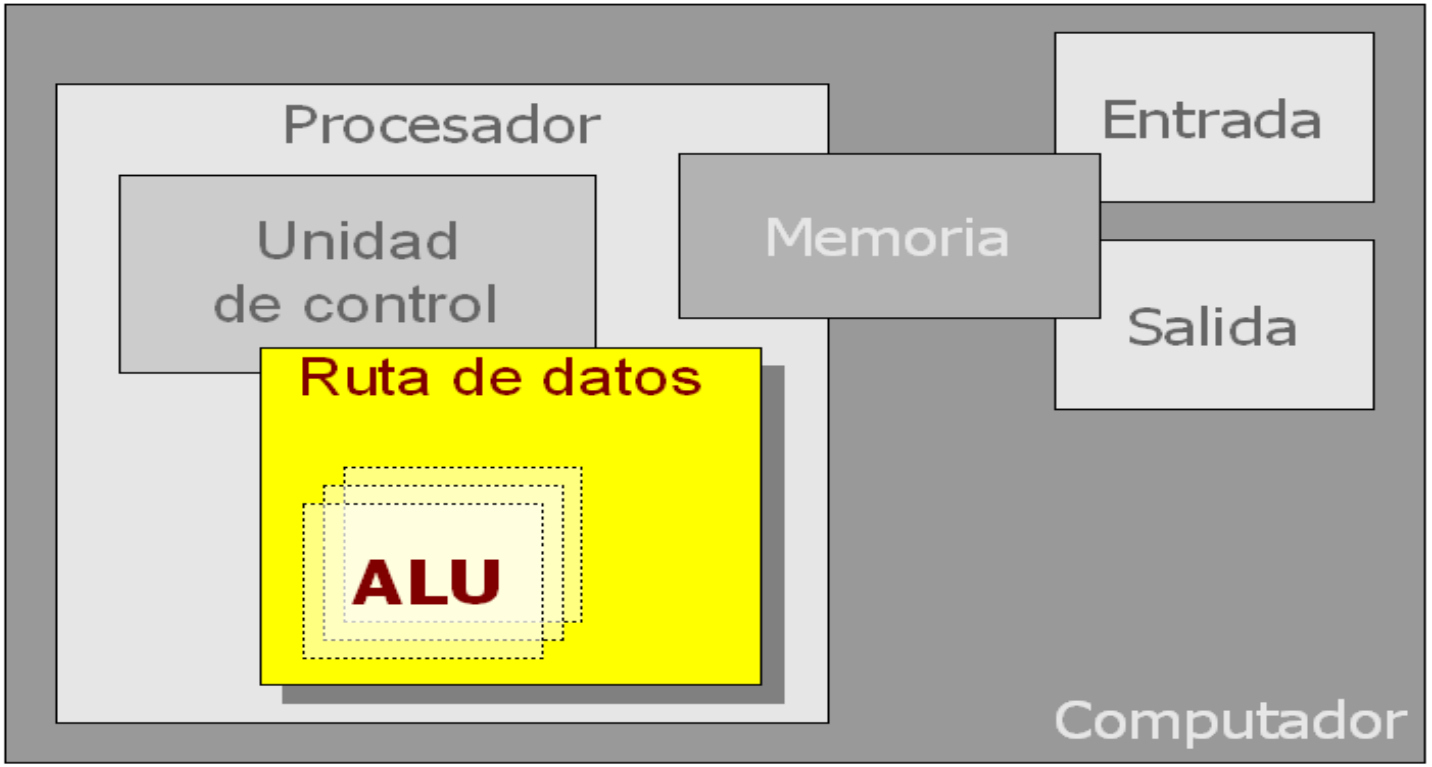
Unidad Aritmético Lógica

Contenido

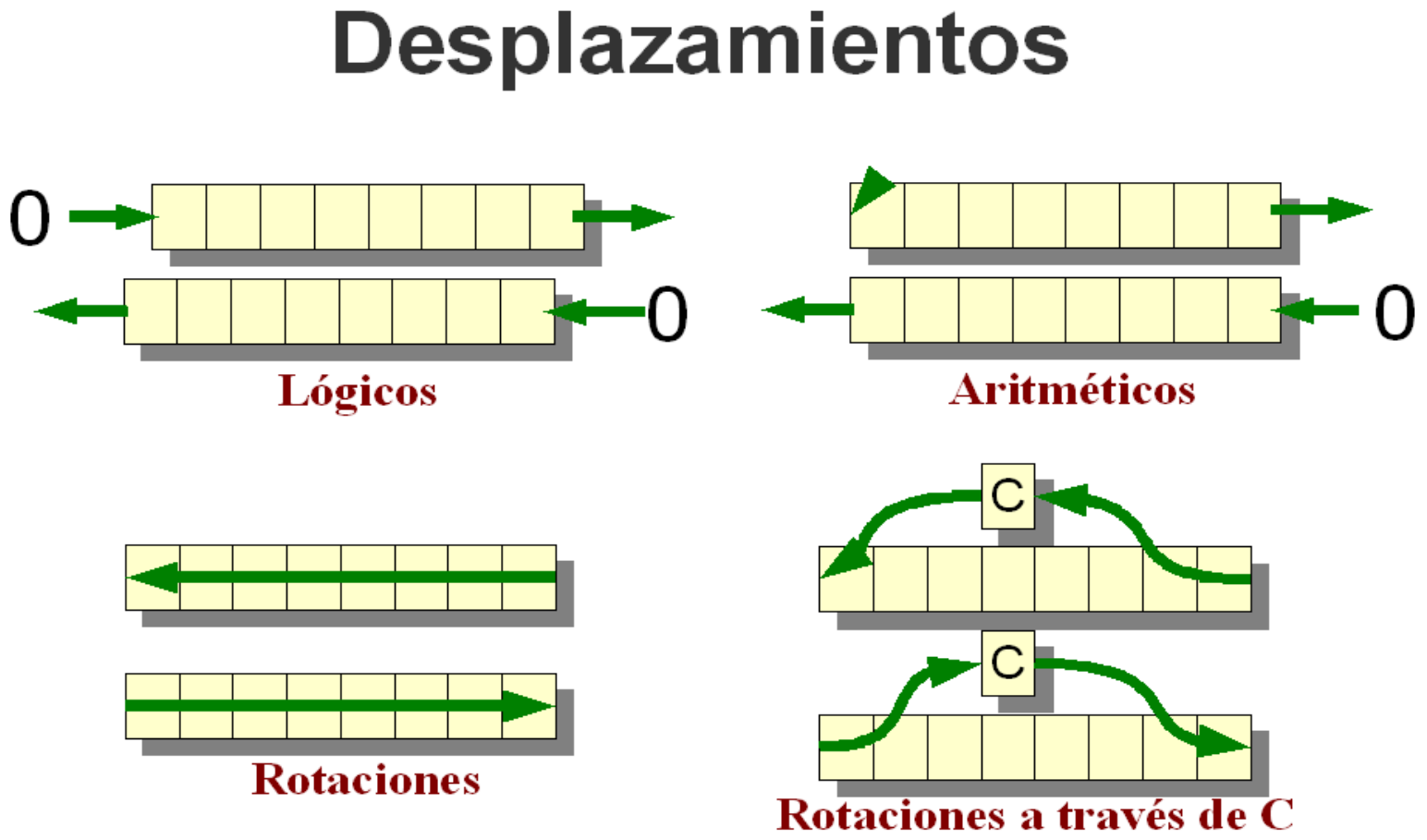
- Introducción
  - Operaciones lógicas: desplazamientos lógicos
  - Operaciones aritméticas: multiplicación y división
- Multiplicación
  - Sin signo: suma desplazamiento
  - Sin signo: sumas y restas
  - Con signo: Algoritmo de Booth
- División
  - Con restauración
  - Sin restauración
- Coma flotante

La ALU dentro del computador

ALU: Componente de la Ruta de Datos



Operaciones lógicas: desplazamientos



Operaciones aritméticas: multiplicación y división

- La operación básica es la suma, que se supone conocida en este curso

- También se suponen conocidos los multiplicadores paralelo de tipo combinacional

- En este tema se estudian los algoritmos para realizar las operaciones desde el punto de vista de la arquitectura, centrados en la multiplicación y la división

MULTIPLICACIÓN BINARIA SIN SIGNO: **SUMA DESPLAZAMIENTO**

Fundamentos

Se multiplica cada bit del multiplicador, empezando por el menos significativo, por todos los bits del multiplicando

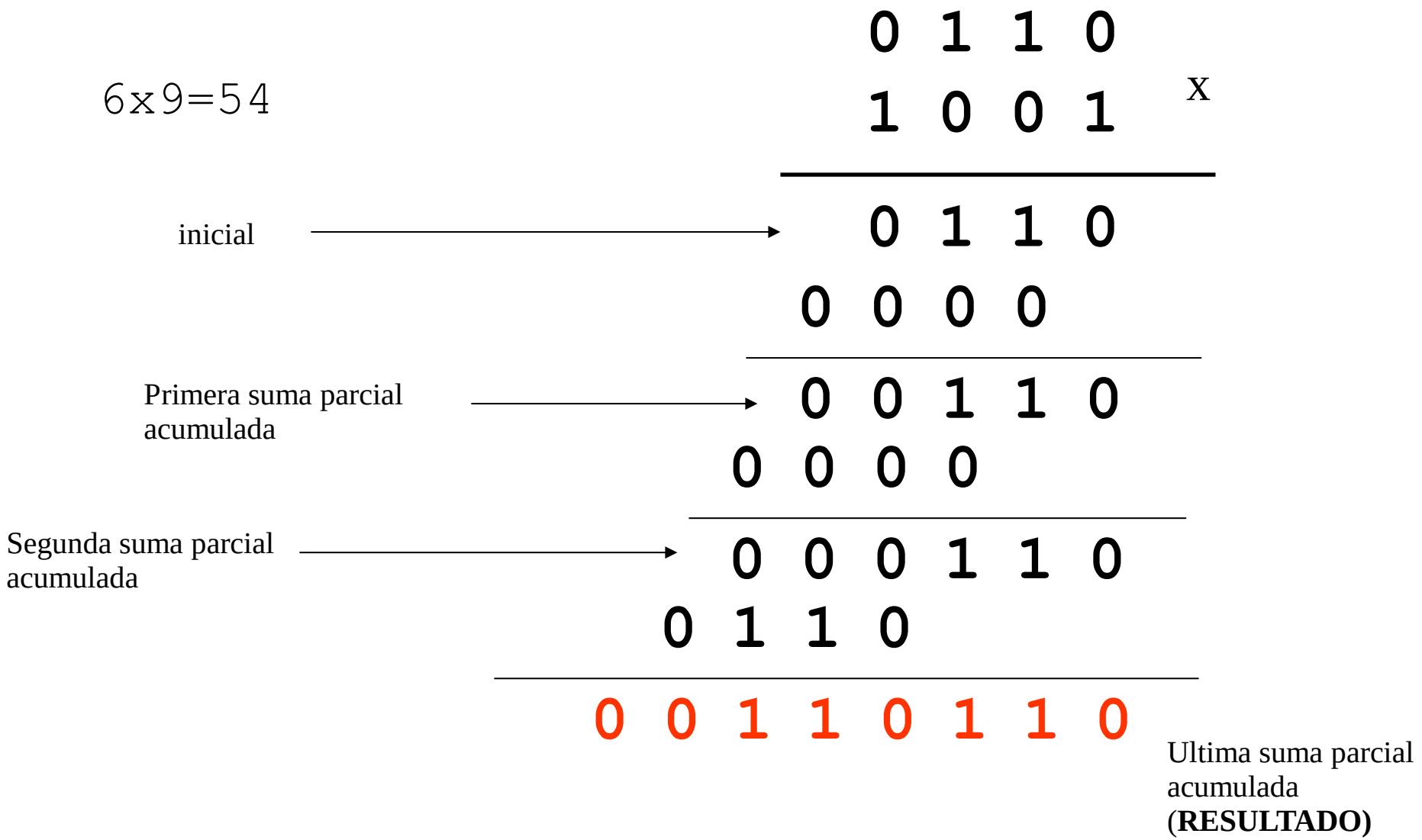
Si es 0, el resultado es 0

Si es 1, el resultado es el multiplicando

Cada vez que se considera un nuevo bit del multiplicador, se desplaza hacia la izquierda una posición el nuevo producto y se añade a la suma parcial acumulada

La última suma parcial será el resultado

Ejemplo


6x9=54
0 1 1 0
1 0 0 1
x
inicial
0 1 1 0
0 0 0 0
Primera suma parcial acumulada
0 0 1 1 0
0 0 0 0
Segunda suma parcial acumulada
0 0 0 1 1 0
0 1 1 0
0 0 1 1 0 1 1 0
Ultima suma parcial acumulada (RESULTADO)

IDEA BÁSICA

Hay que sumar las sumas acumuladas con el multiplicando DESPLAZADO A LA IZQUIERDA cuando el LSB del multiplicador sea 1.

EN LUGAR DE DESPLAZAR A LA IZQUIERDA EL MULTIPLICANDO, SE DESPLAZAN A LA DERECHA LAS SUMAS PARCIALES

Necesidades

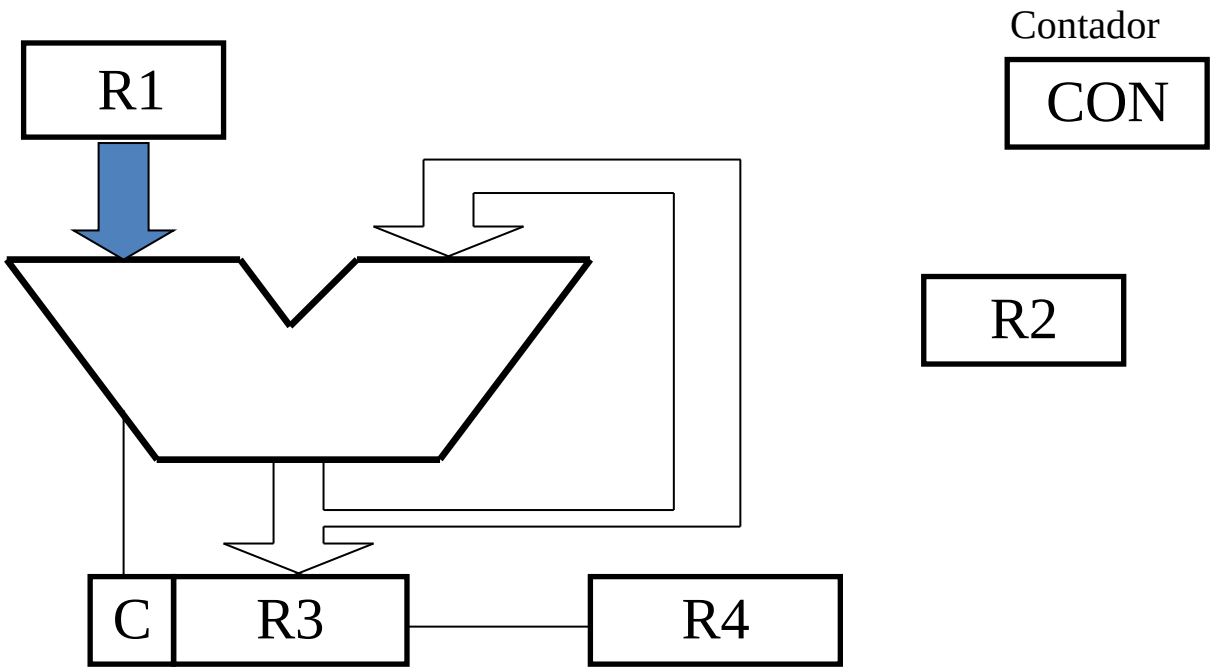
El producto de un número binario de n bits con otro de m bits necesita m+n bits

Sea A el multiplicador y B el multiplicando

En el caso de que m=n, se necesitan dos registros iguales para almacenar el resultado

Sean R3 y R4 los registros donde se va almacenando el resultado

Arquitectura



Algoritmo Suma Desplazamiento

Se inicializan a 0 el contador de bits del multiplicador y el registro R3

Se carga R1 con el multiplicando A y R2 con el multiplicador B

Se analiza el LSB de R2

Si es 1, se suma R1+ R3 y el resultado se deja en R3

Si es 0, se suma R1 + 0 (es decir, no se hace nada) y el resultado se deja en R3

Se incrementa el contador

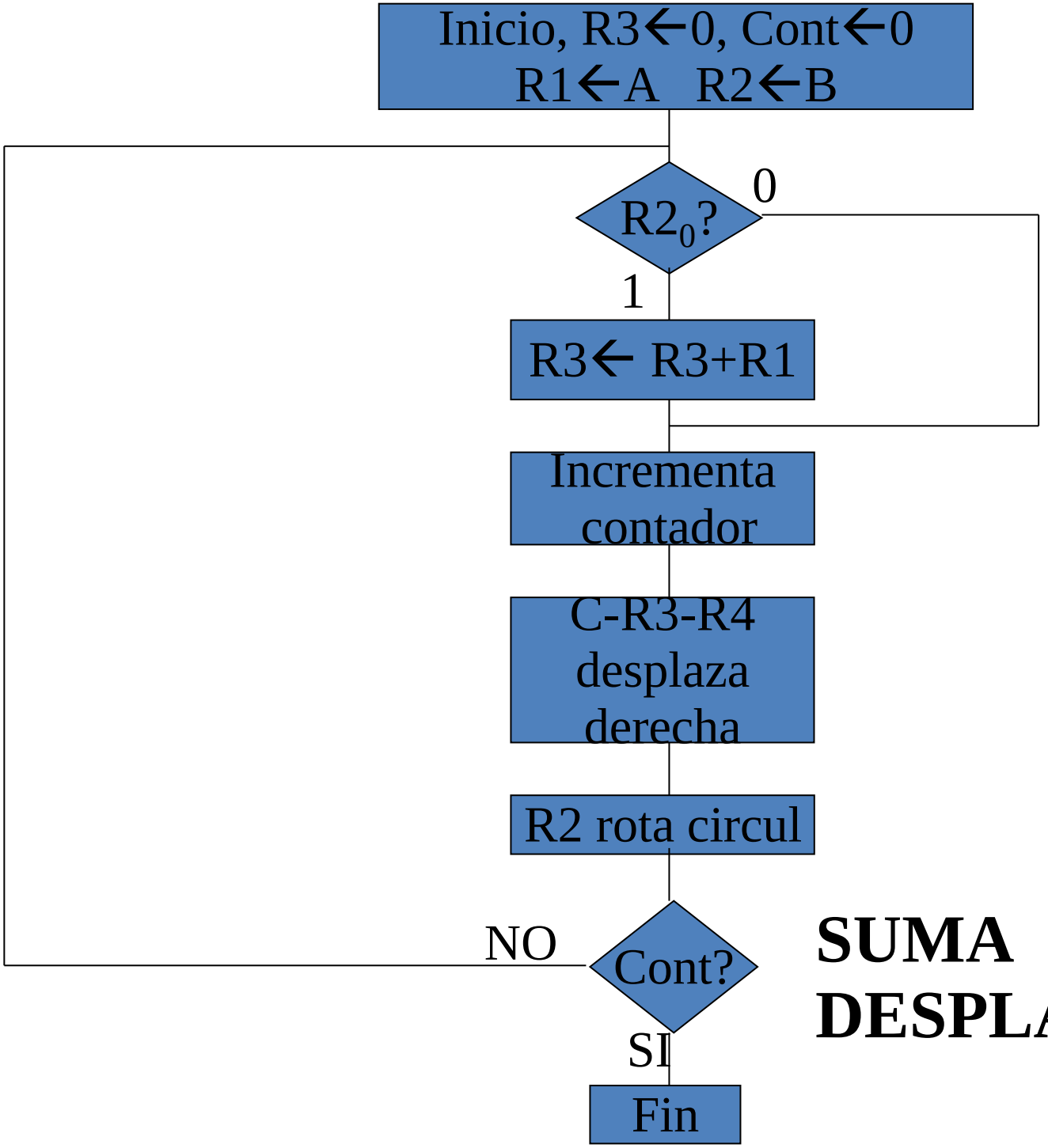
Se produce un desplazamiento aritmético (con extensión de signo) de una posición hacia la derecha del conjunto C-R3-R4

Se realiza una rotación circular a derecha de R2

Se mira si el contador ha llegado hasta el final

En caso negativo se repite el proceso, volviendo a preguntar por el LSB de R2

En caso afirmativo se finaliza el proceso y el resultado está en C-R3-R4

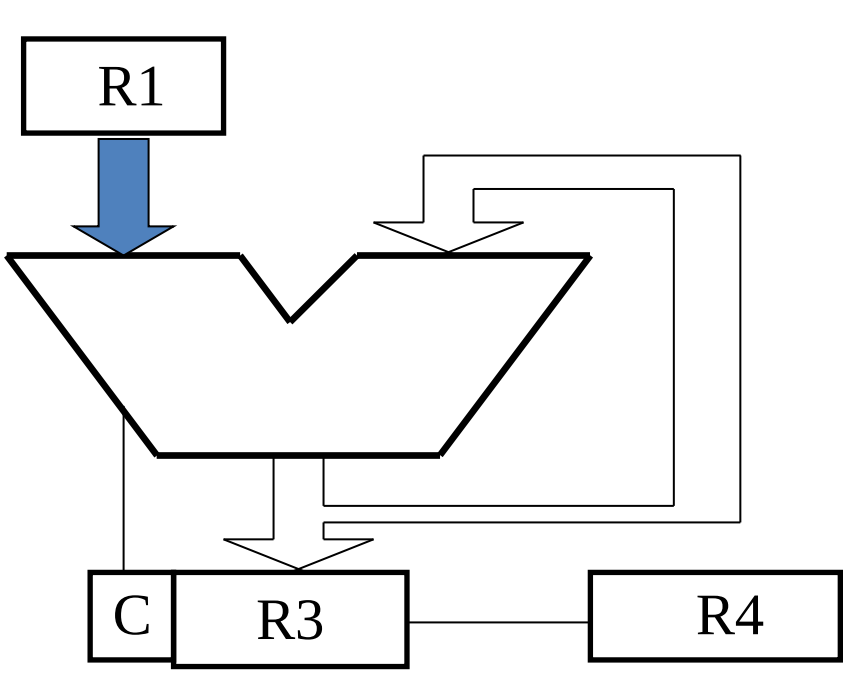


SUMA DESPLAZAMIENTO

Ejemplo

Multiplicar A=0110 por B=1001

El resultado debe ser 6x9=54 (en binario 0011 0110)



Contador

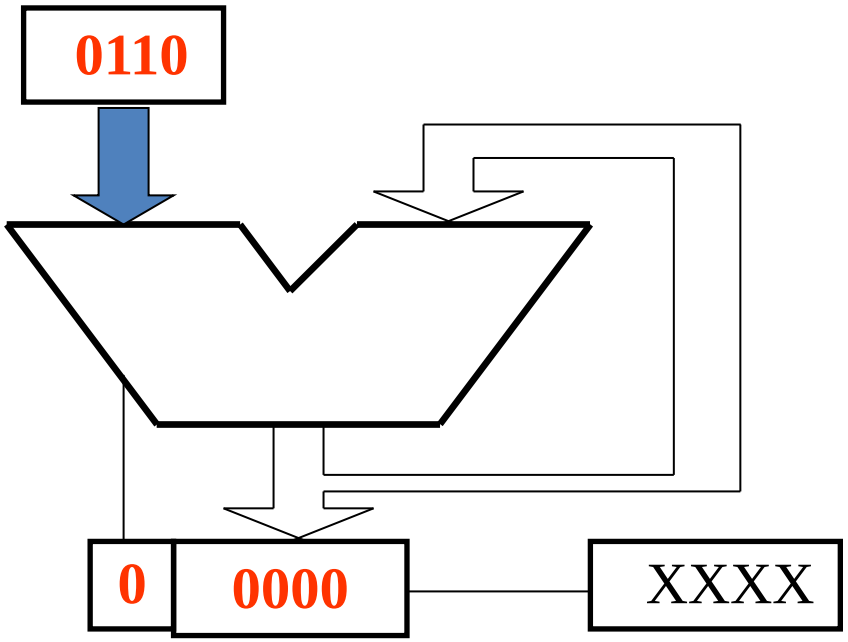
R2

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Inicialización

Se carga A en R1 y B en R2.

Se pone a 0 el registro R3 y el contador



Contador

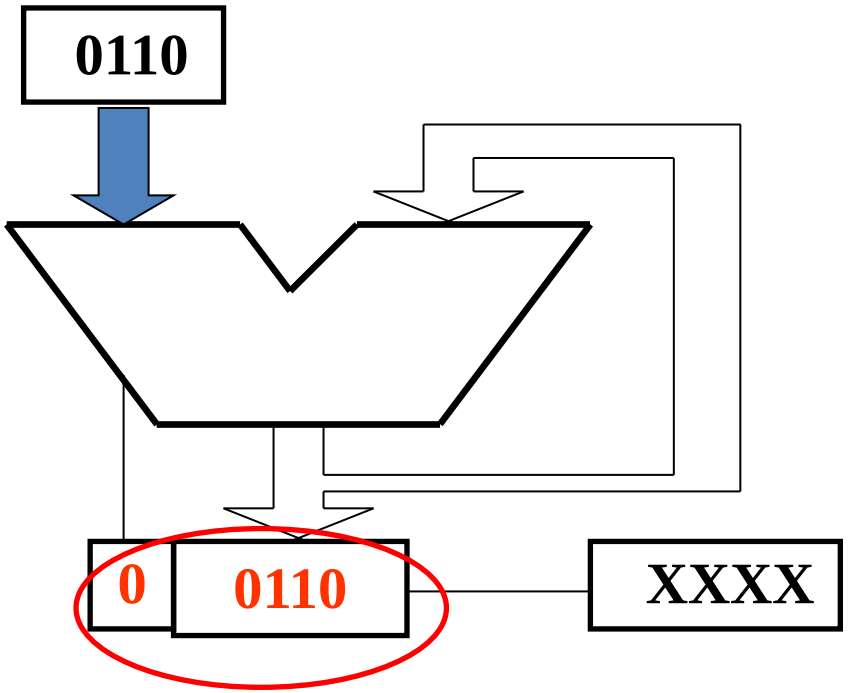
0

1001

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Comprobación de $R2_0$ y acción

Como el LSB de R2 es 1 se suman R3 + R1 y se deja el resultado en R3



Contador

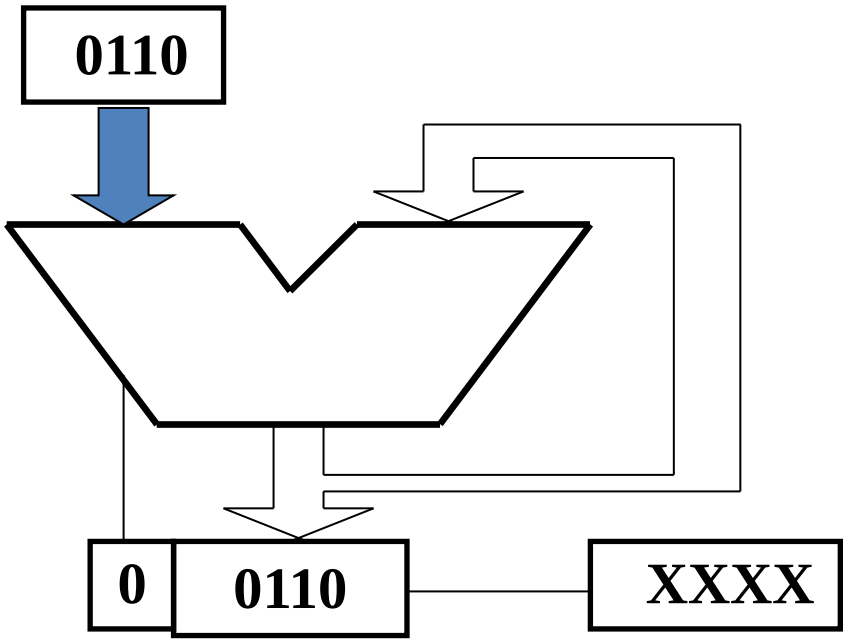
0

1001

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Incremento contador

Se incrementa el contador en una unidad



Contador

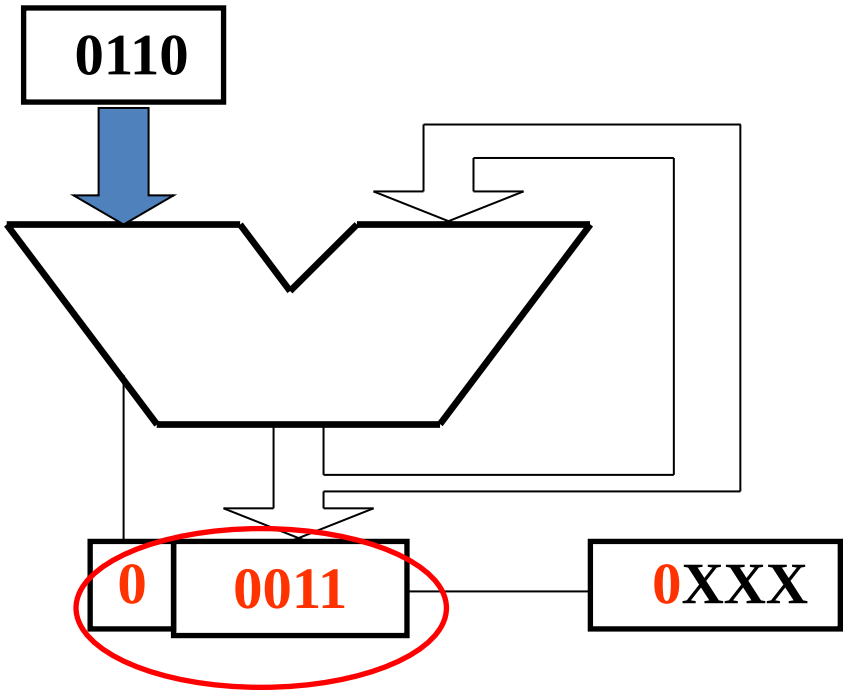
1

1001

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Contador

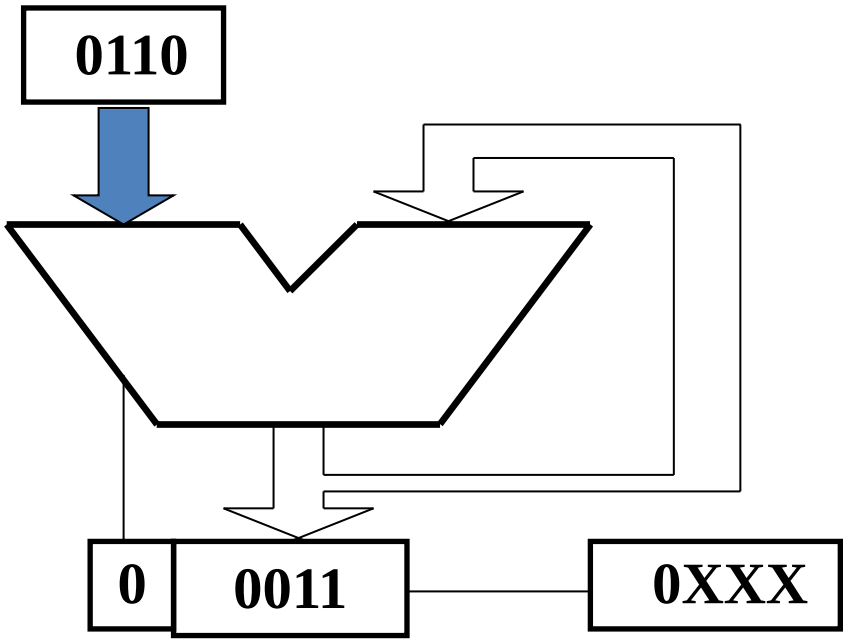
1

1001

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Rotación circular de R2

El registro R2 rota a la derecha de forma circular



Contador

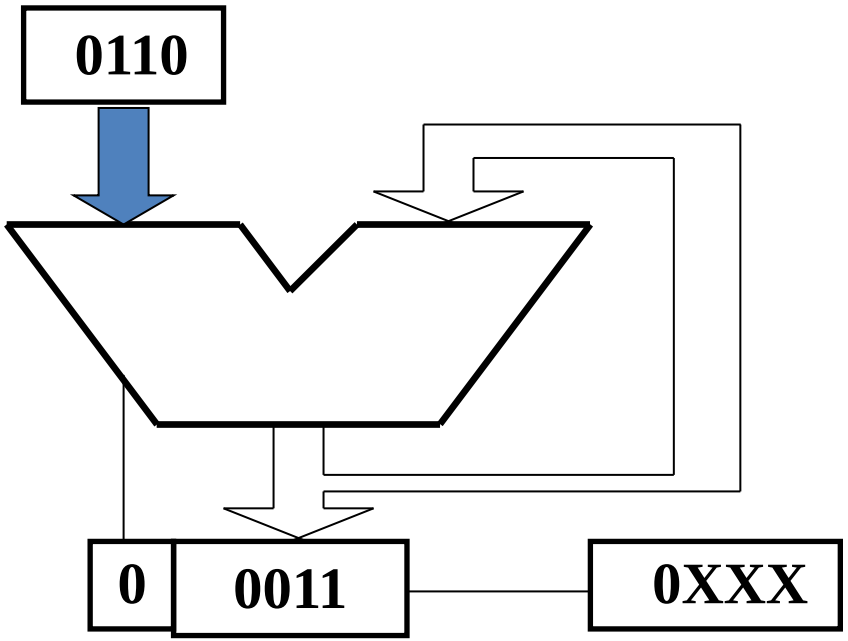
1

1100

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



Contador

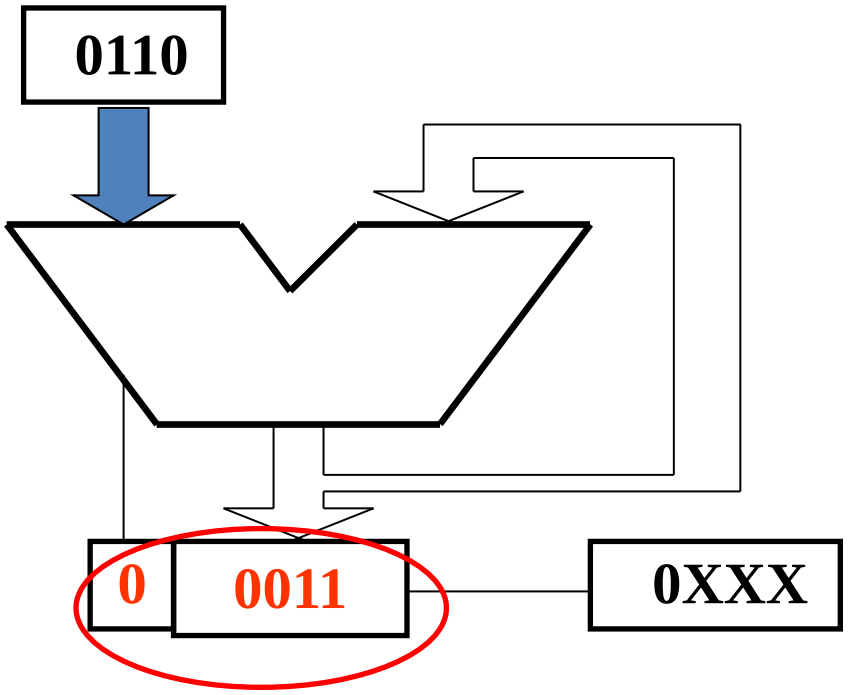
1

1100

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Comprobación de $R2_0$ y acción

Como el LSB de R2 es 0 se suman R3 + 0, es decir, NO hace nada, y se deja el resultado que había en R3



Contador

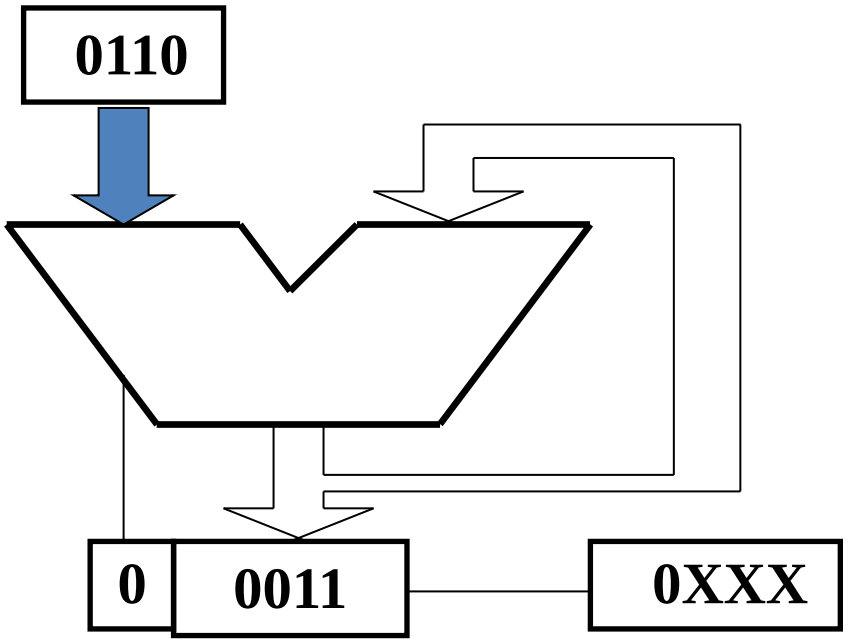
1

1100

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Incremento contador

Se incrementa el contador en una unidad



Contador

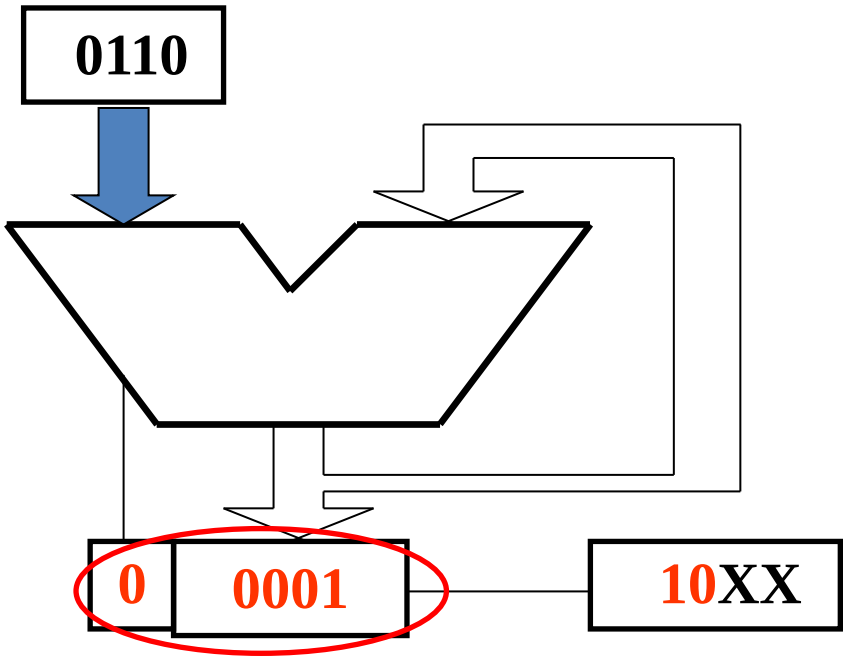
2

1100

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Contador

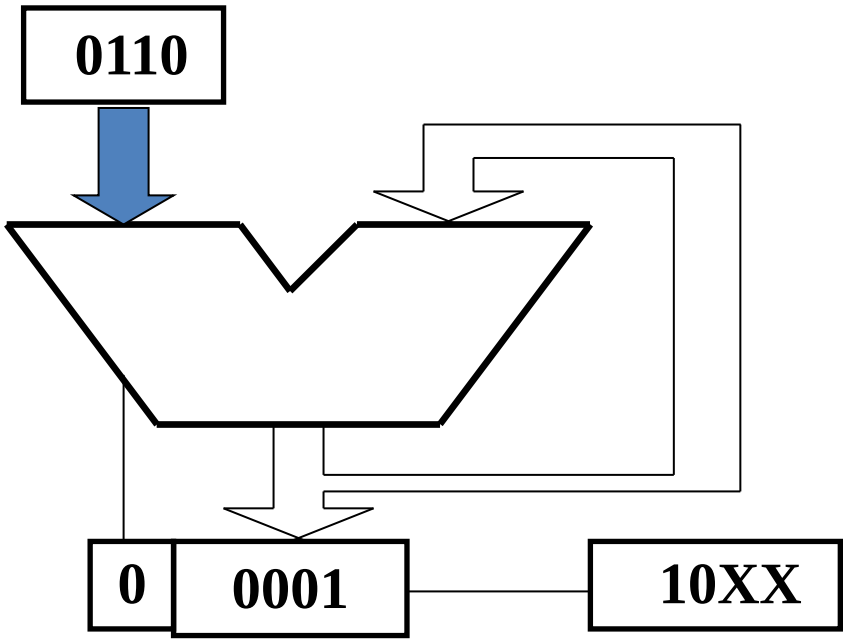
2

1100

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Rotación circular de R2

El registro R2 rota a la derecha de forma circular



Contador

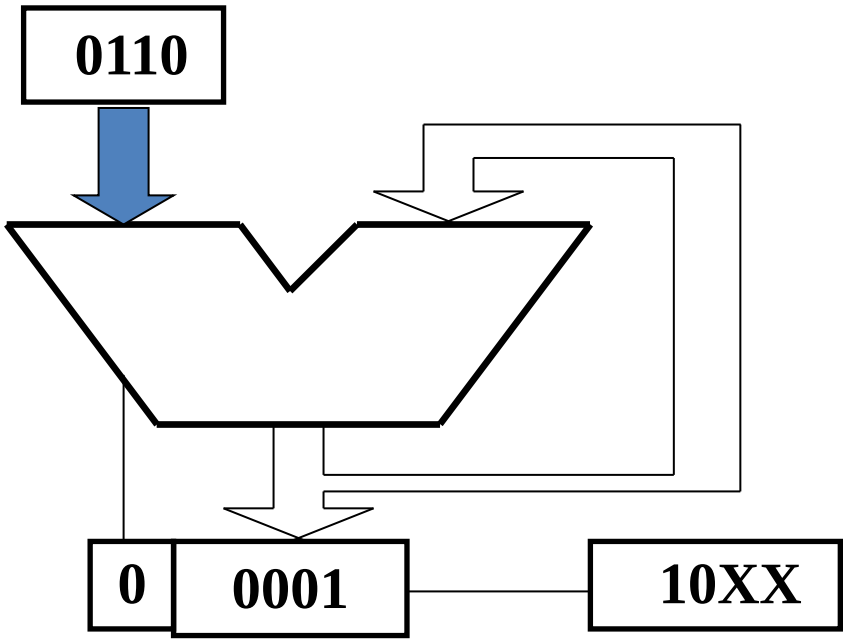
2

0110

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



Contador

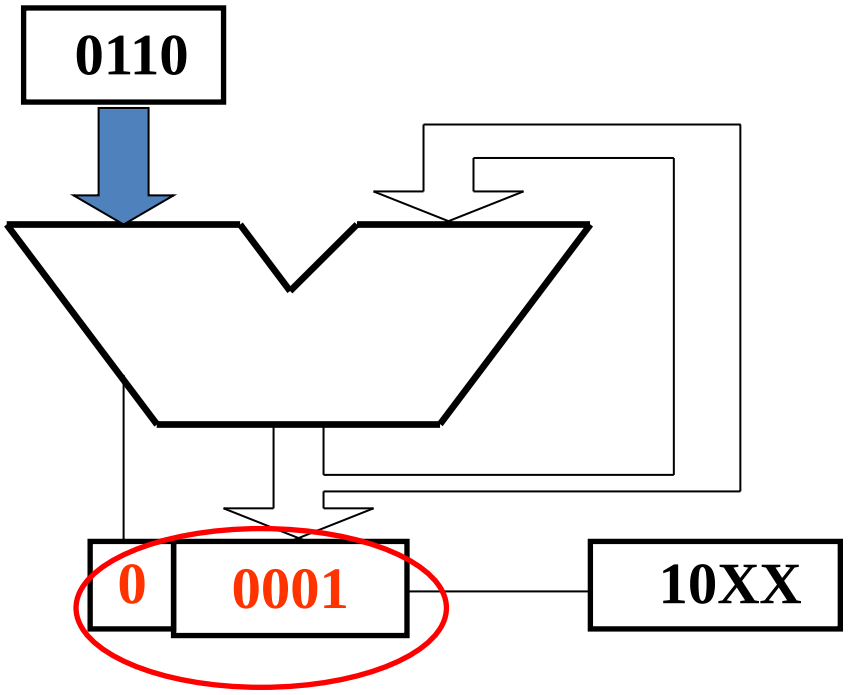
2

0110

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Comprobación de $R2_0$ y acción

Como el LSB de R2 es 0 se suman R3 + 0, es decir, NO hace nada, y se deja el resultado que había en R3



Contador

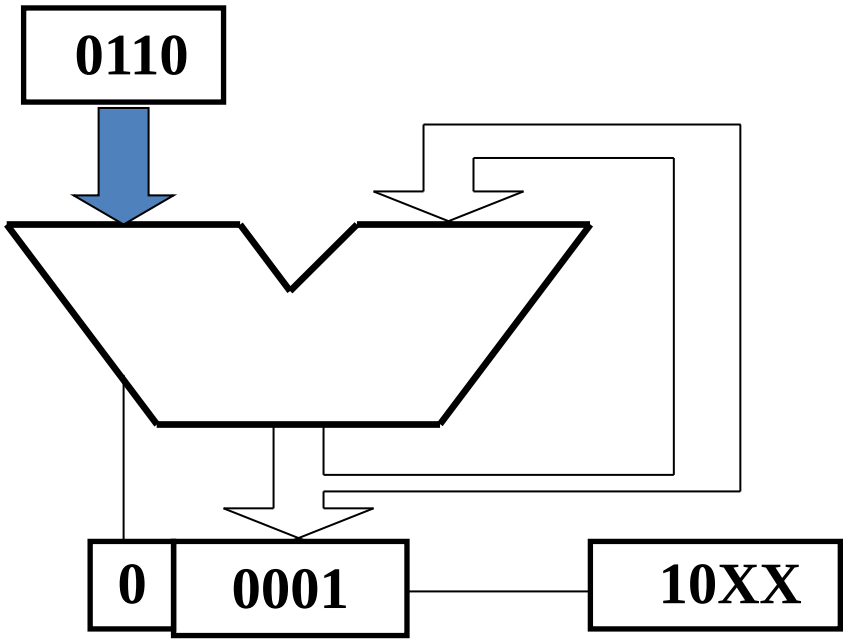
2

0110

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Incremento contador

Se incrementa el contador en una unidad



Contador

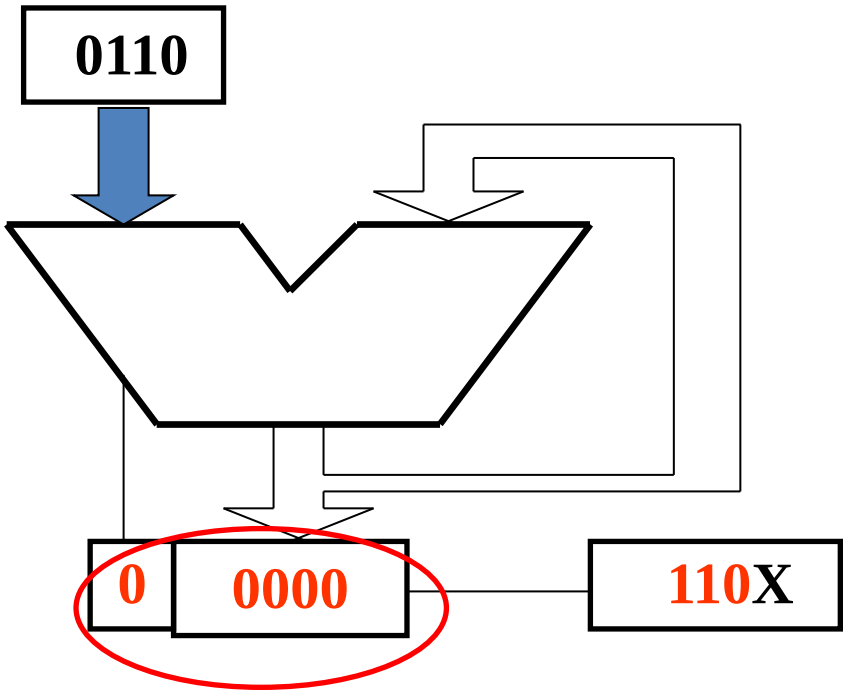
3

0110

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Contador

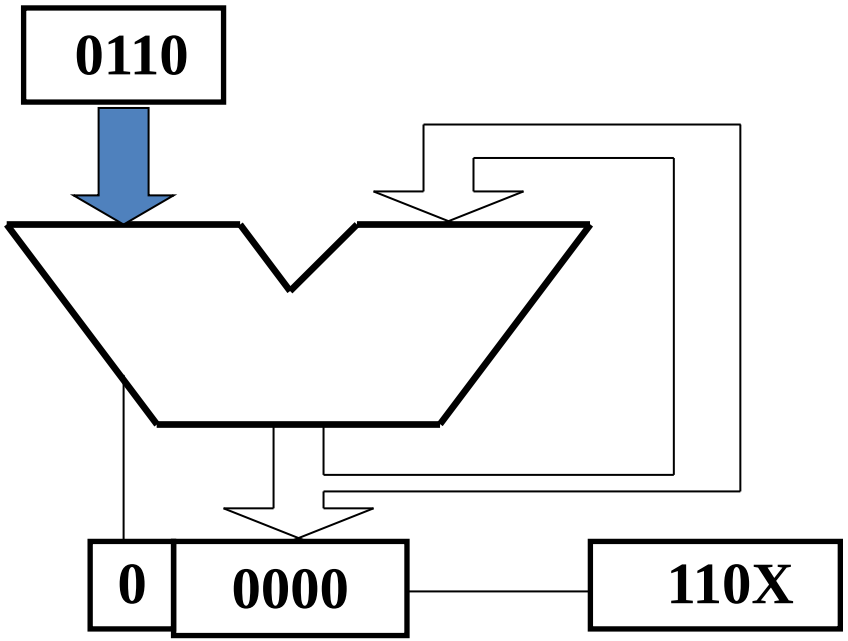
3

0110

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Rotación circular de R2

El registro R2 rota a la derecha de forma circular



Contador

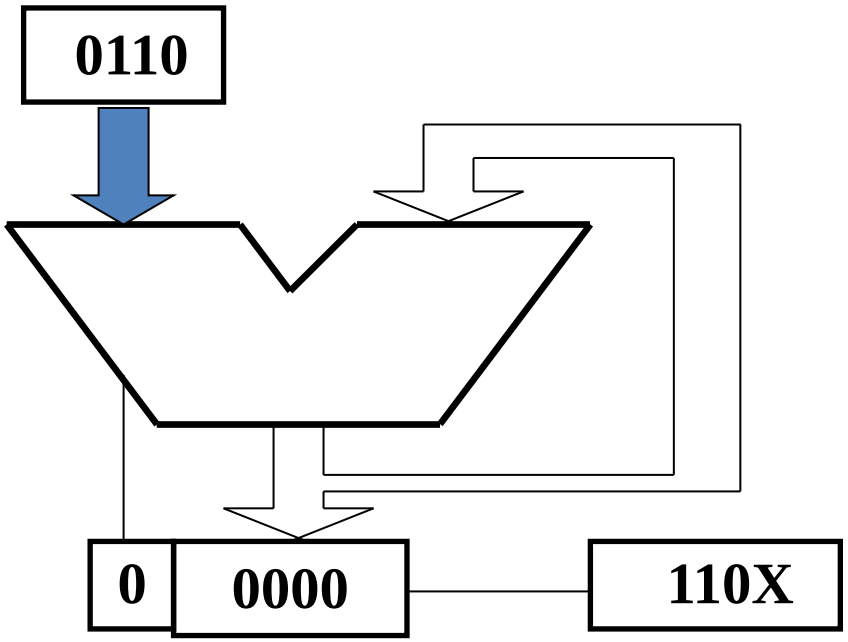
3

0011

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



Contador

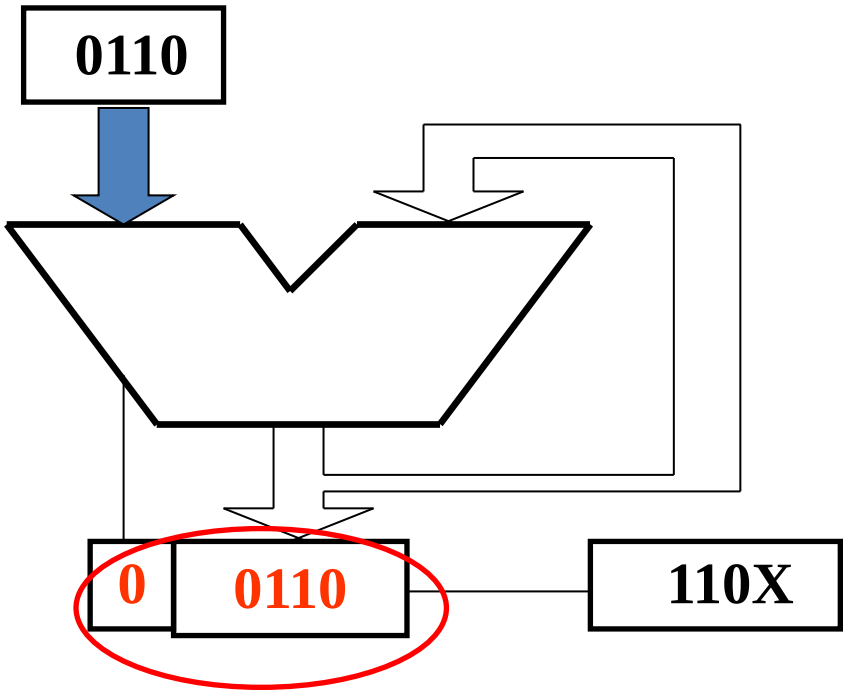
3

0011

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Comprobación de $R2_0$ y acción

Como el LSB de R2 es 1 se suman R3 + R1 y se deja el resultado en R3



Contador

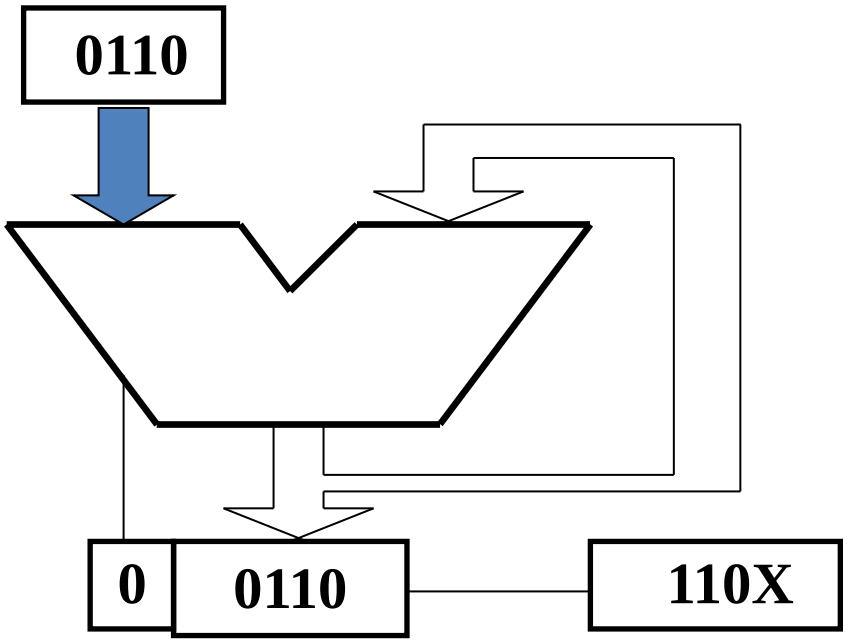
3

0011

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Incremento contador

Se incrementa el contador en una unidad



Contador

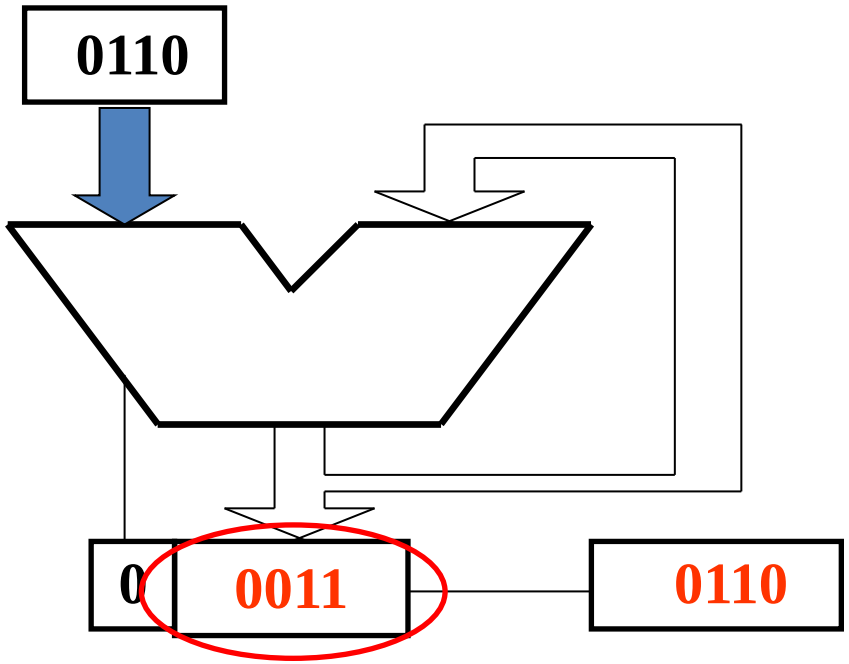
4

0011

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Contador

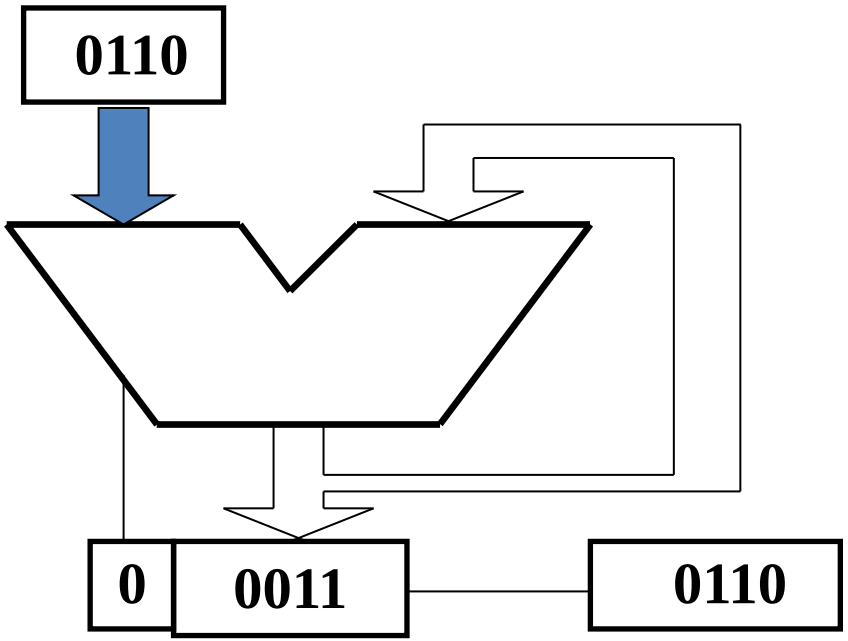
4

0011

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Rotación circular de R2

El registro R2 rota a la derecha de forma circular



Contador

4

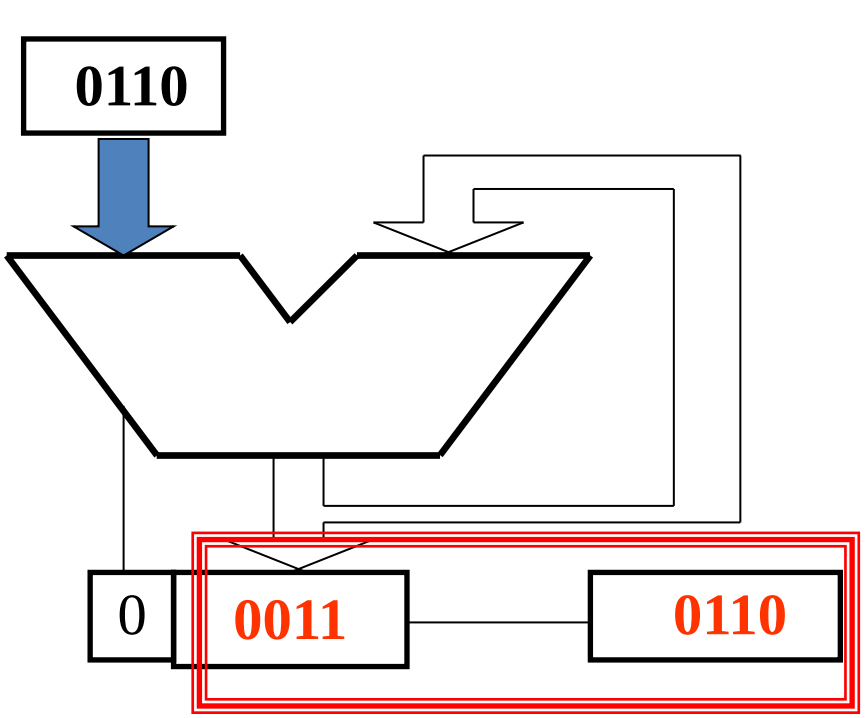
1001

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

¿Ha llegado al final?

Comprueba que el contador es 4 y acaba.

El resultado está en el registro R3 seguido de R4



Contador

4

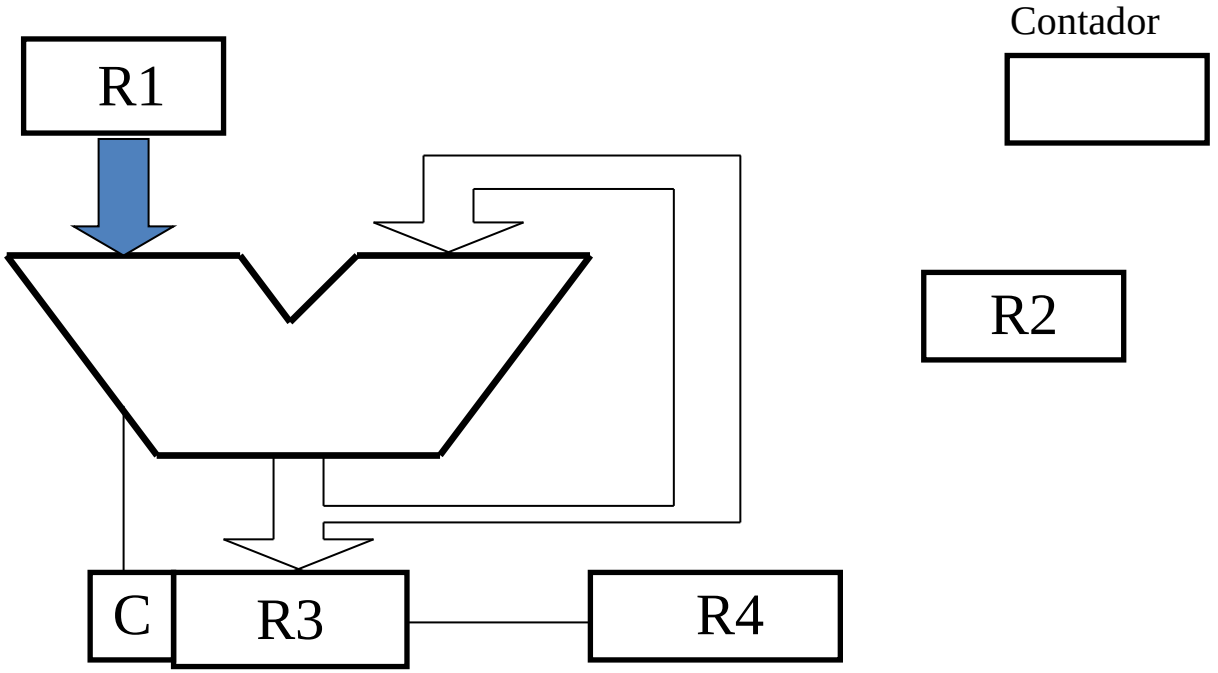
0011

				0	1	1	0
			x	1	0	0	1
				0	1	1	0
			0	0	0	0	
		0	0	0	1	1	0
		0	0	0	0		
	0	0	0	0	1	1	0
	0	1	1	0			
0	0	1	1	0	1	1	0

Otro ejemplo

Multiplicar A=1001 por B=0110

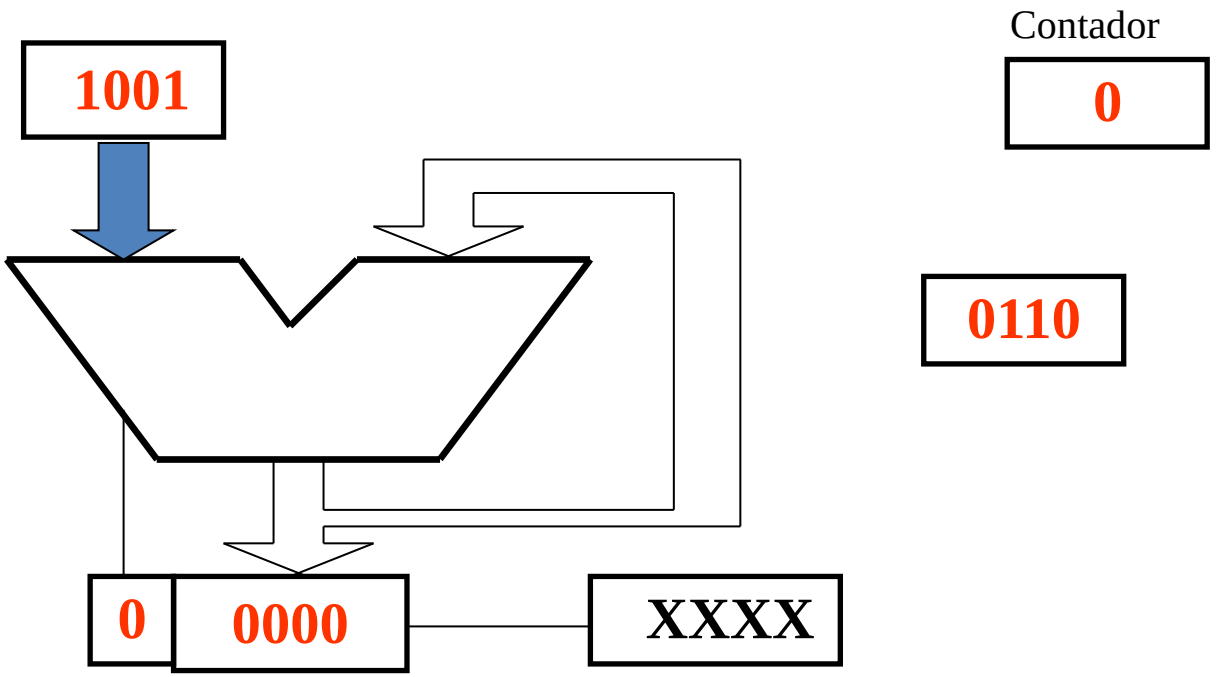
El resultado debe ser 9x6=54 (en binario 0011 0110)



Inicialización

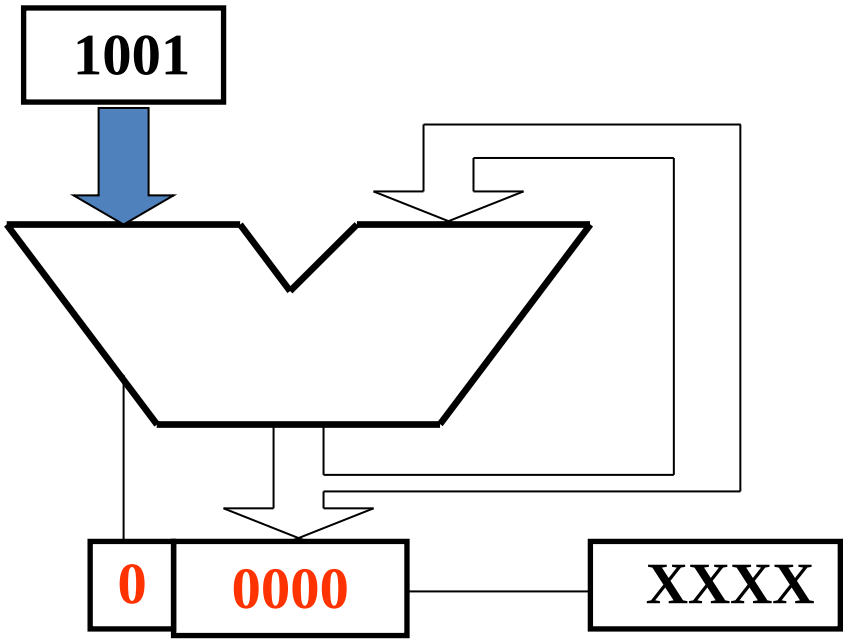
Se carga A en R1 y B en R2.

Se pone a 0 el registro R3 y el contador



Comprobación de $R2_0$ y acción

Como el LSB de R2 es 0 se suman R3 + 0, es decir, NO hace nada, y se deja el resultado que había en R3



Contador

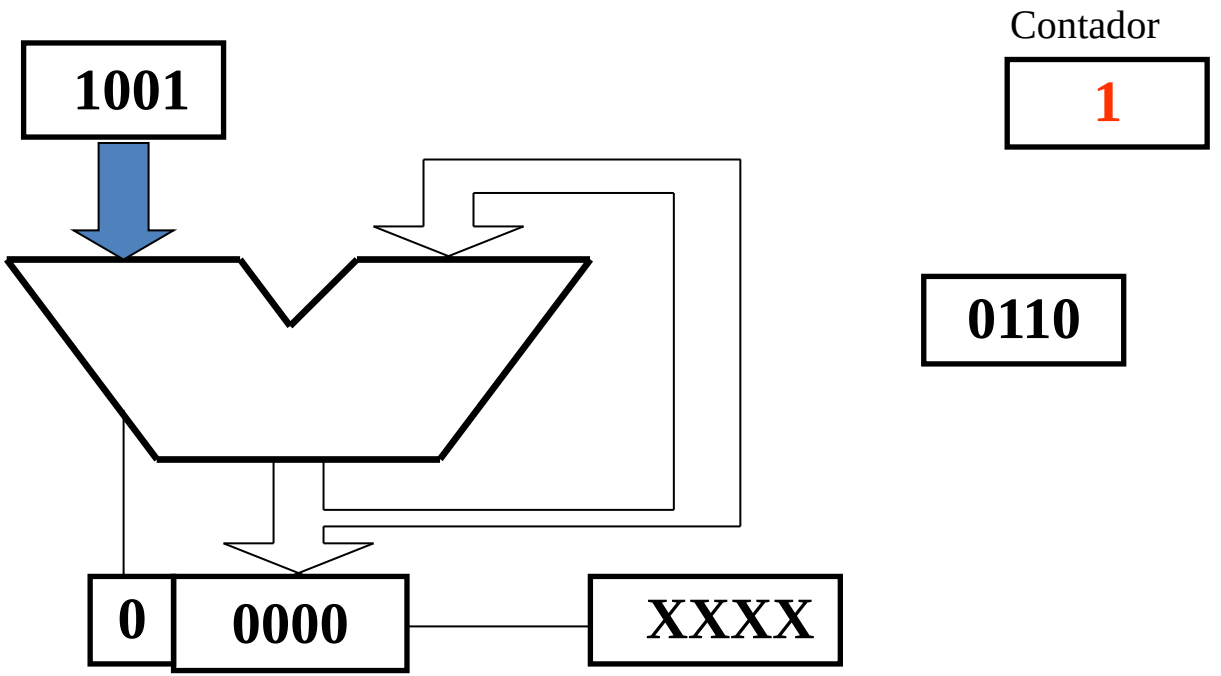
0

0110

```
  1001
  0110
 -----
  00000
```

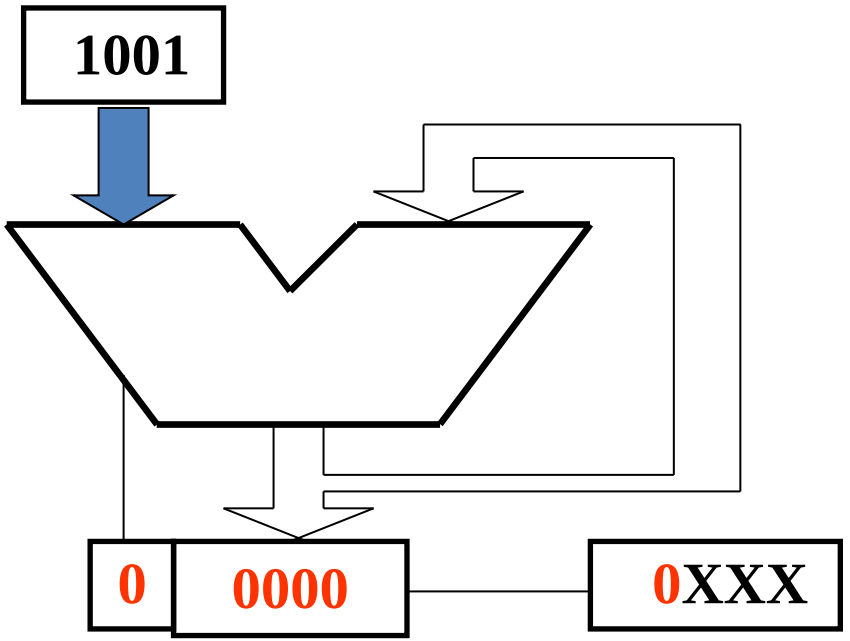
Incremento contador

Se incrementa el contador en una unidad



Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Contador

1

0110

1001

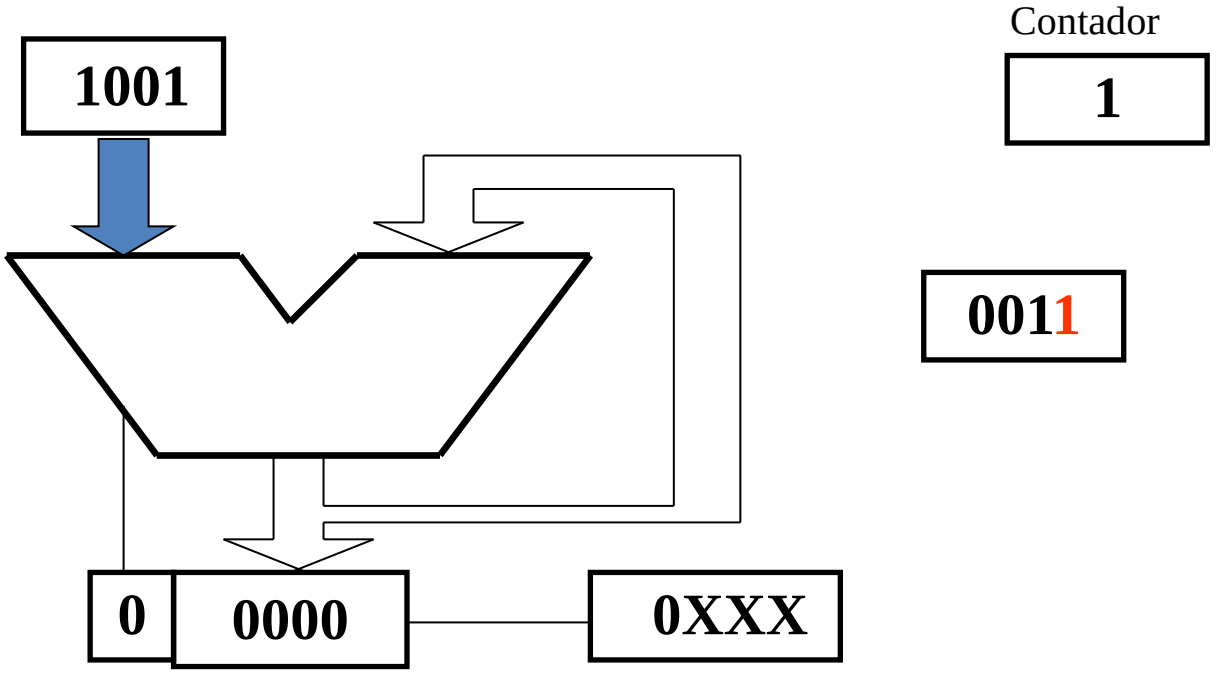
0110

00000

00000 0

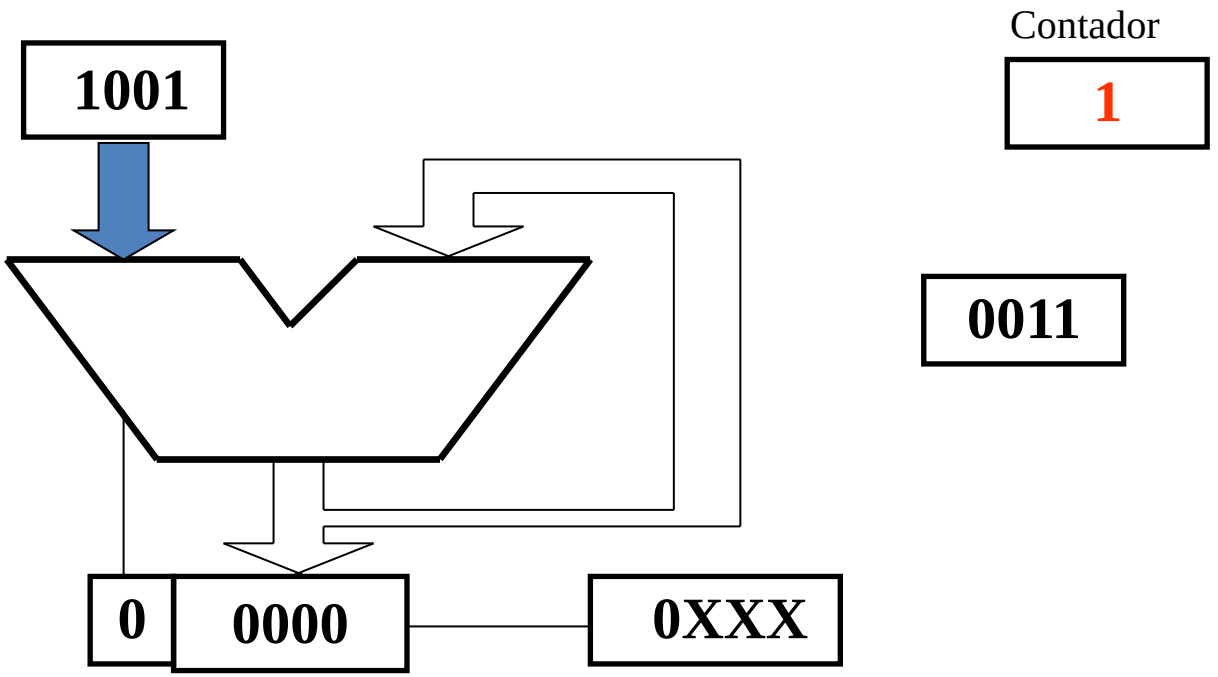
Rotación circular de R2

El registro R2 rota a la derecha de forma circular



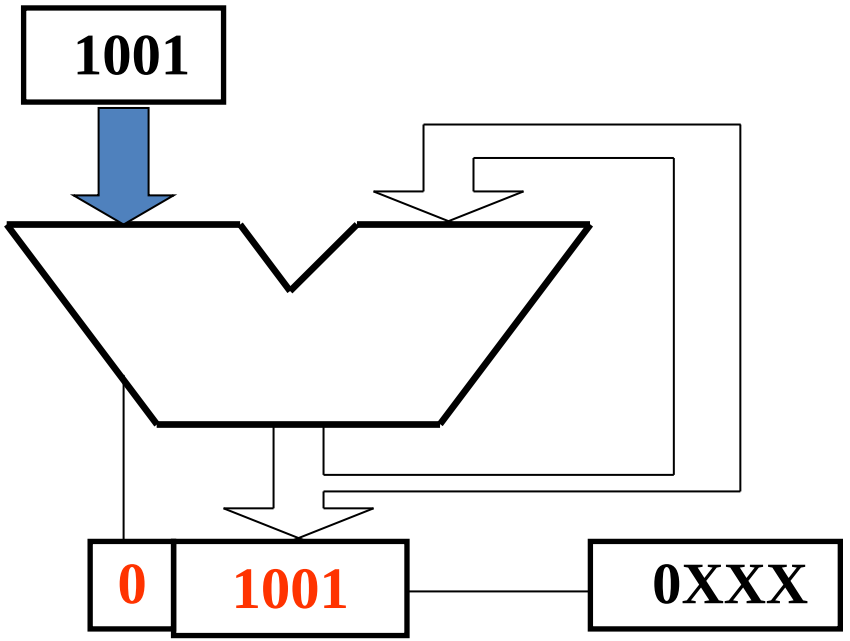
¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



Comprobación de $R2_0$ y acción

Como el LSB de R2 es 1 se suman R3 + R1 y se deja el resultado en R3



Contador

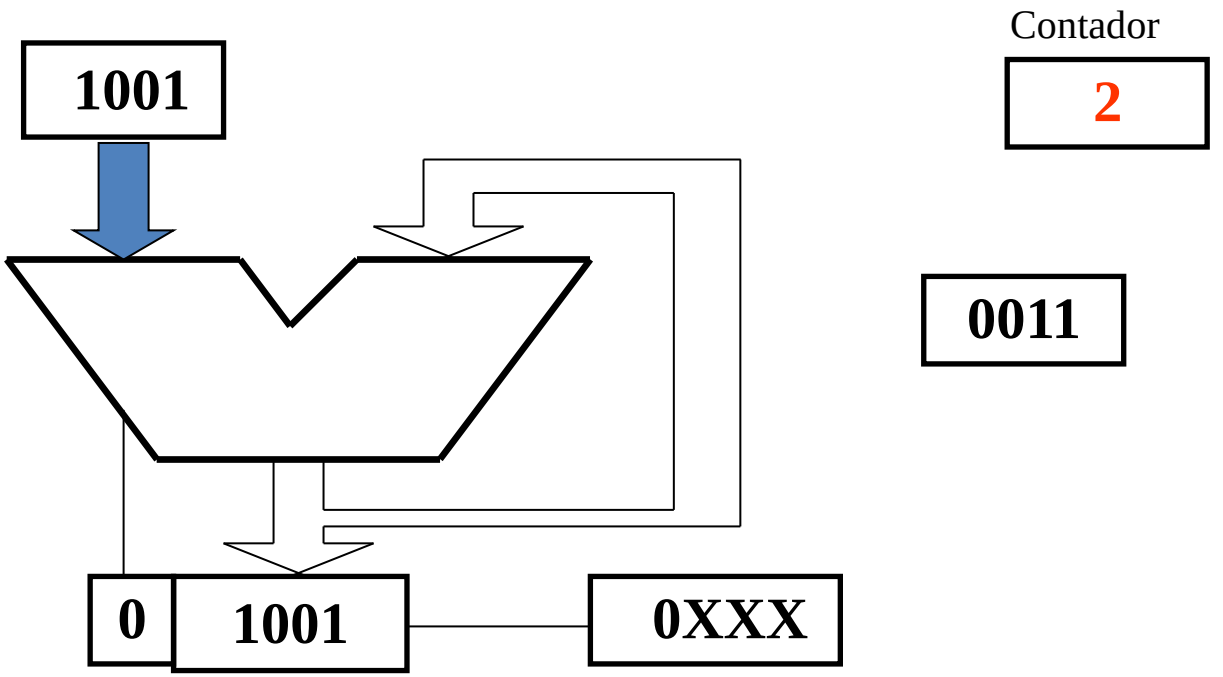
1

0011

```
 1001
 0110
-----
 00000
  ↘
 0000 0
 1001+
-----
01001 0
```

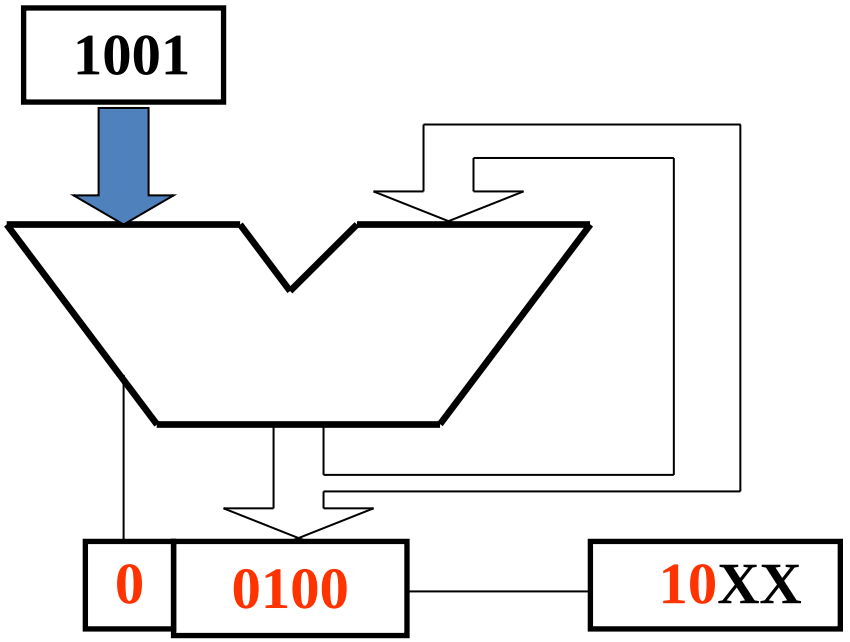
Incremento contador

Se incrementa el contador en una unidad



Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Contador

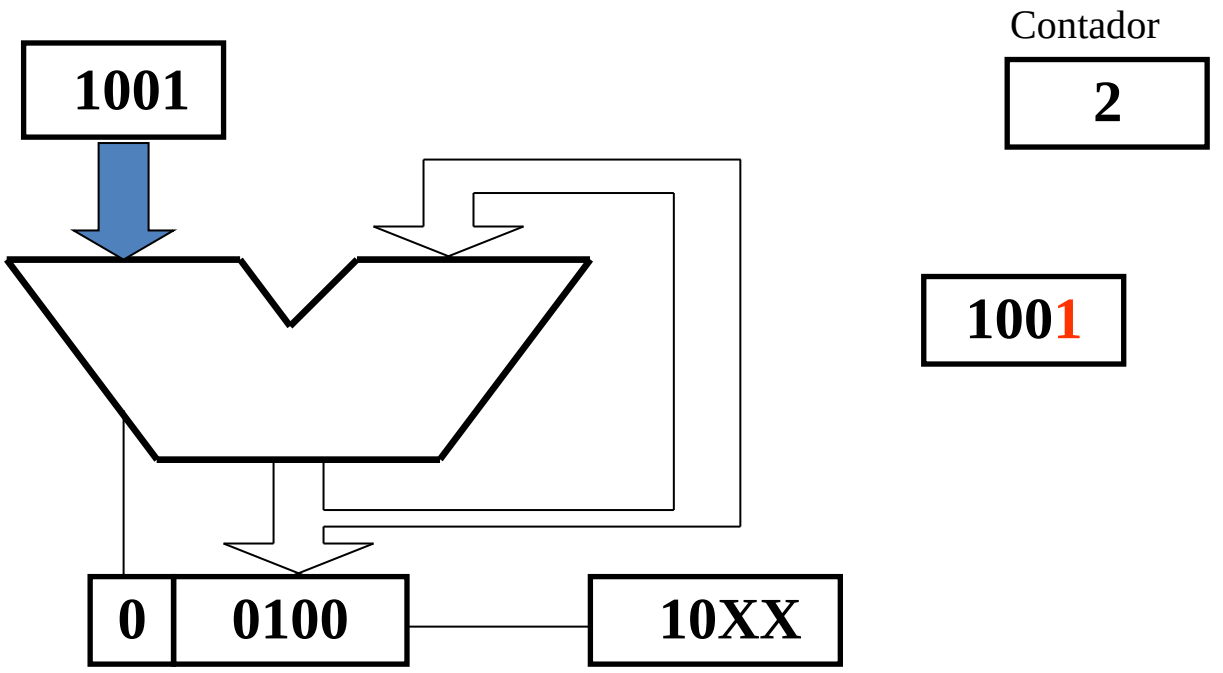
2

0011

```
1001
 0110
-----
00000
 ↘
 0000  0
 1001+
-----
01001  0
 ↘
00100 10
```

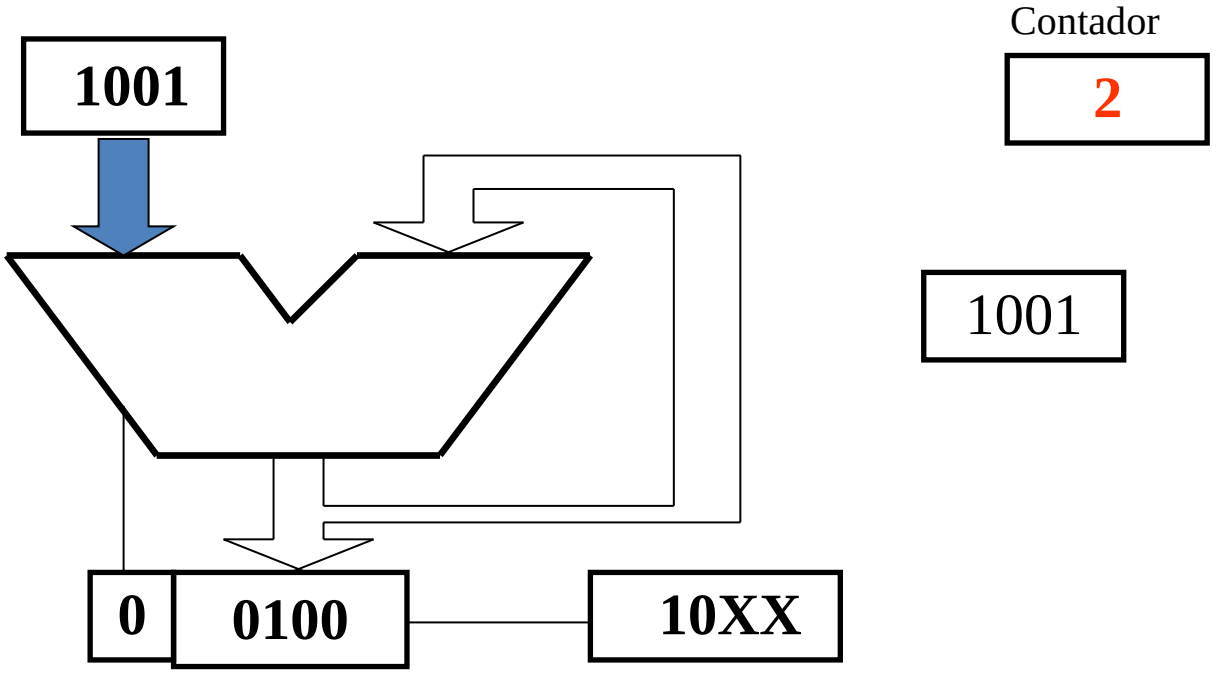
Rotación circular de R2

El registro R2 rota a la derecha de forma circular



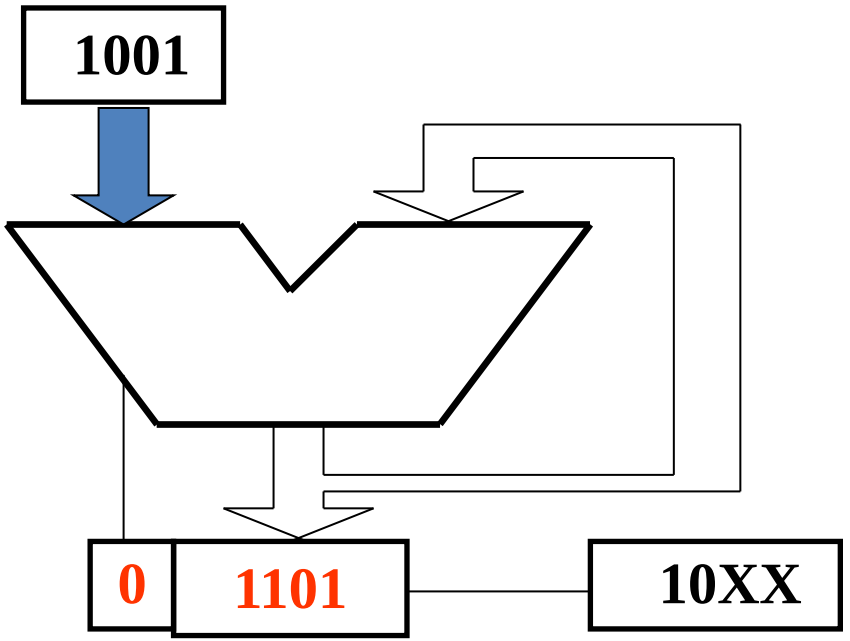
¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



Comprobación de $R2_0$ y acción

Como el LSB de R2 es 1 se suman R3 + R1 y se deja el resultado en R3



Contador

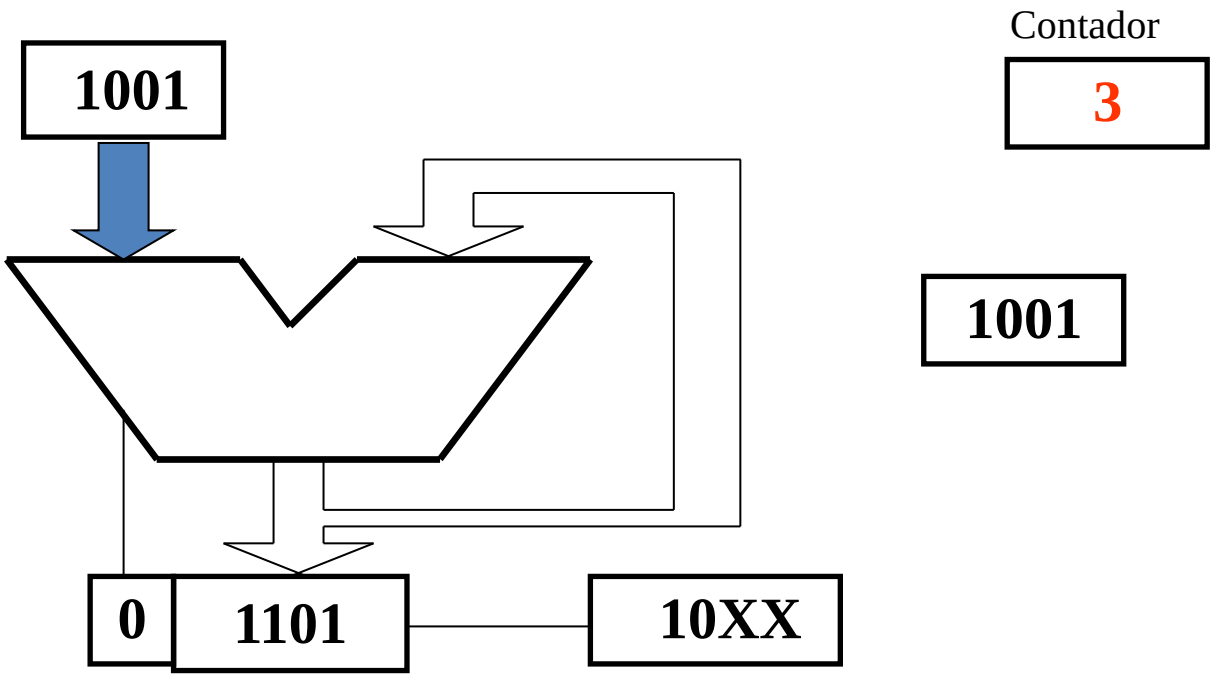
2

1001

```
  1001
  0110
 -----
 00000
   ↘
  0000 0
  1001+
 -----
 01001 0
   ↘
  0100 10
  1001+
 -----
 01101 10
```

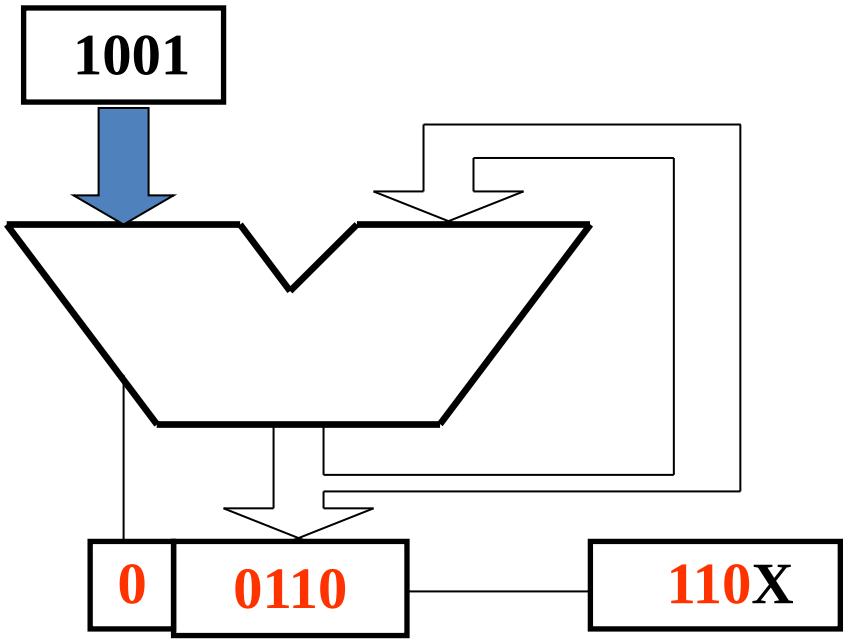
Incremento contador

Se incrementa el contador en una unidad



Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Contador

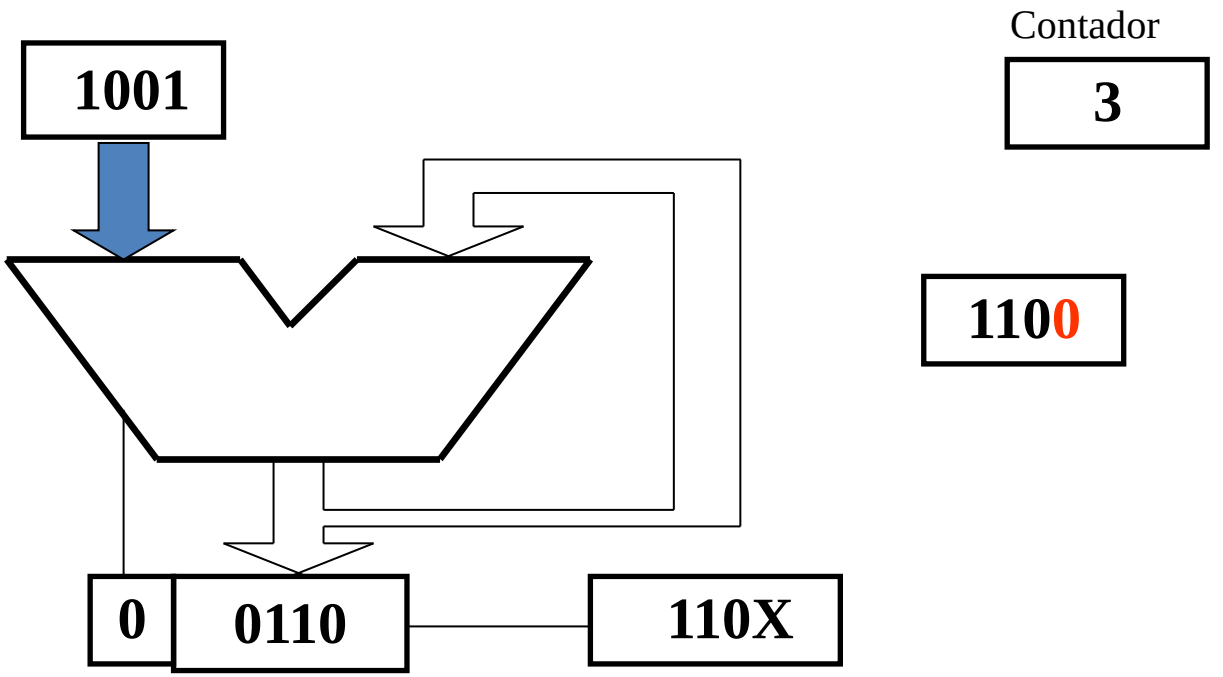
3

1001

```
  1001
  0110
 -----
 00000
      ↘
  0000  0
  1001+
 -----
 01001  0
      ↘
  0100  10
  1001+
 -----
 01101  10
      ↘
 00110  110
```

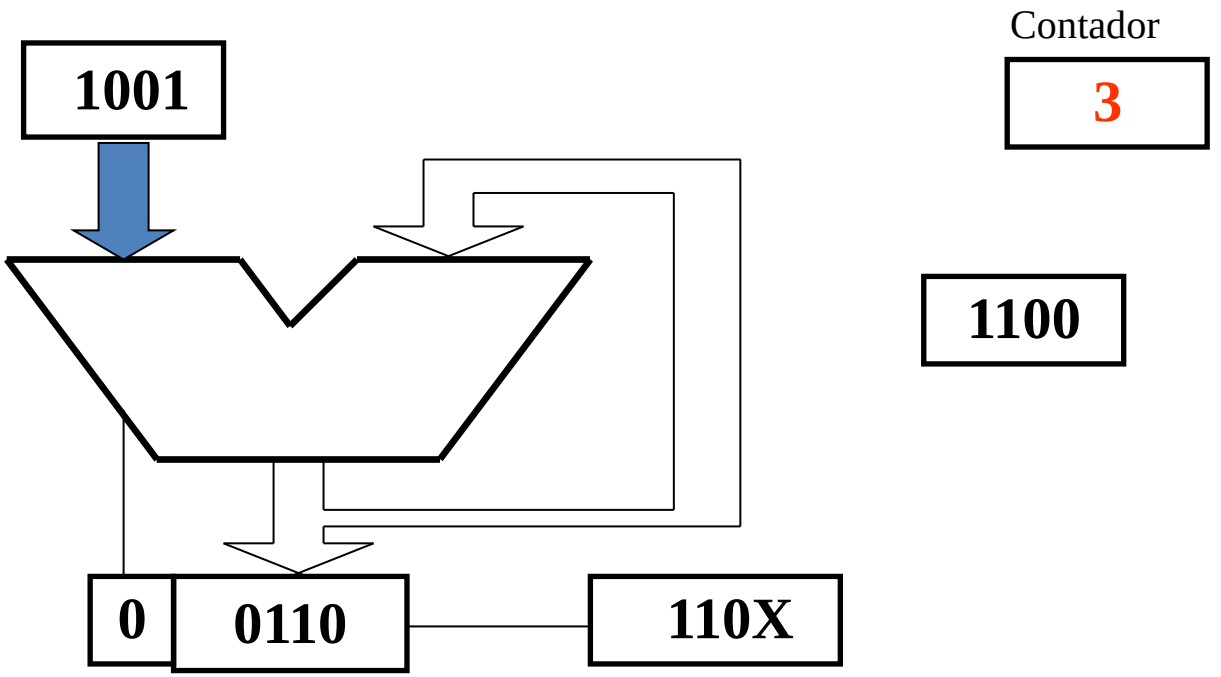
Rotación circular de R2

El registro R2 rota a la derecha de forma circular



¿Ha llegado al final?

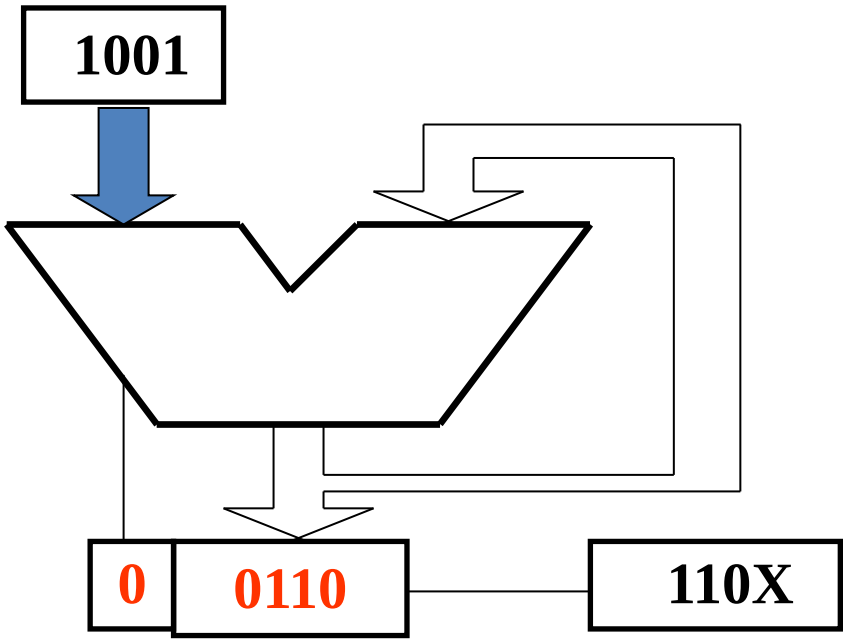
Comprueba que el contador no es 4 y repite el bucle



Comprobación de $R2_0$ y acción

Como el LSB de R2 es 0 se suman R3 + 0, es decir,

NO hace nada, y se deja el resultado que había en R3



Contador

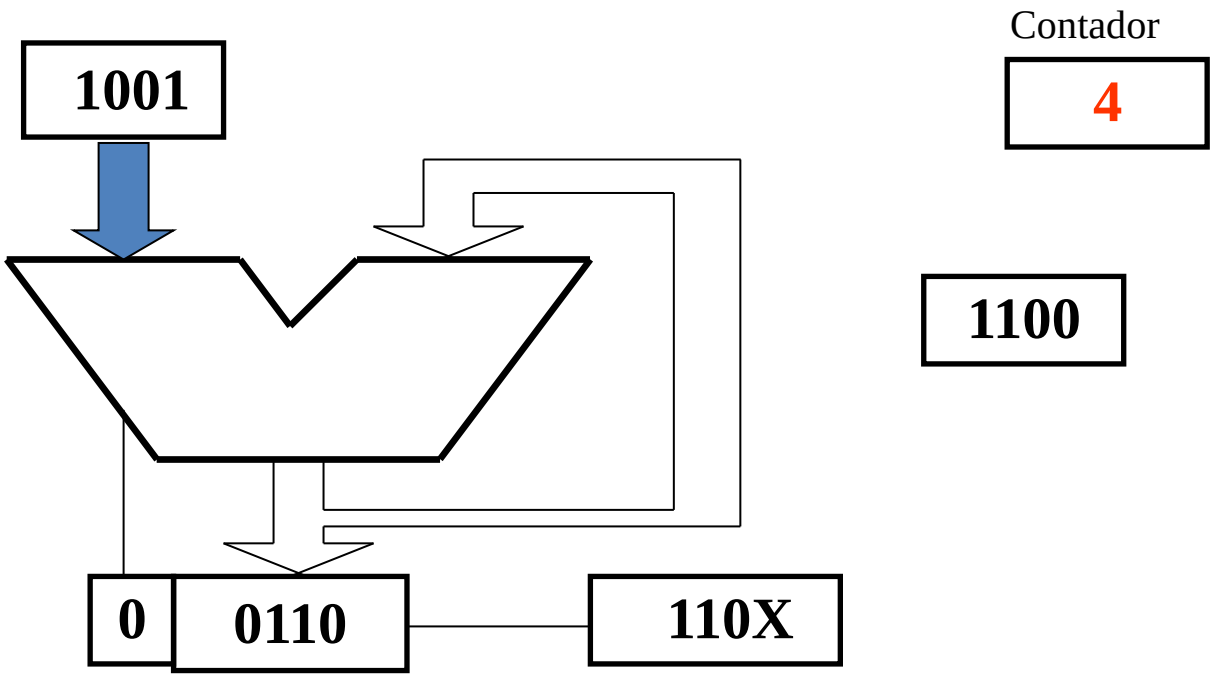
3

1100

```
  1001
  0110
 -----
 00000
   ↘
  0000 0
  1001+
 -----
 01001 0
   ↘
  0100 10
  1001+
 -----
 01101 10
   ↘
  0110 110
  0000+
 -----
 00110 110
```

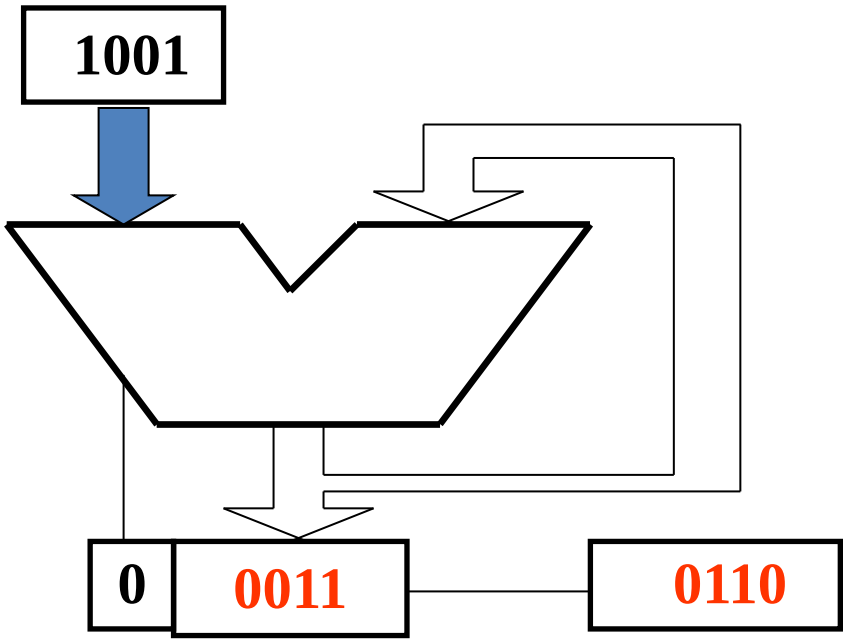
Incremento contador

Se incrementa el contador en una unidad



Desplazamiento a derecha

Se desplaza en conjunto C-R3-R4 una posición a la derecha



Contador

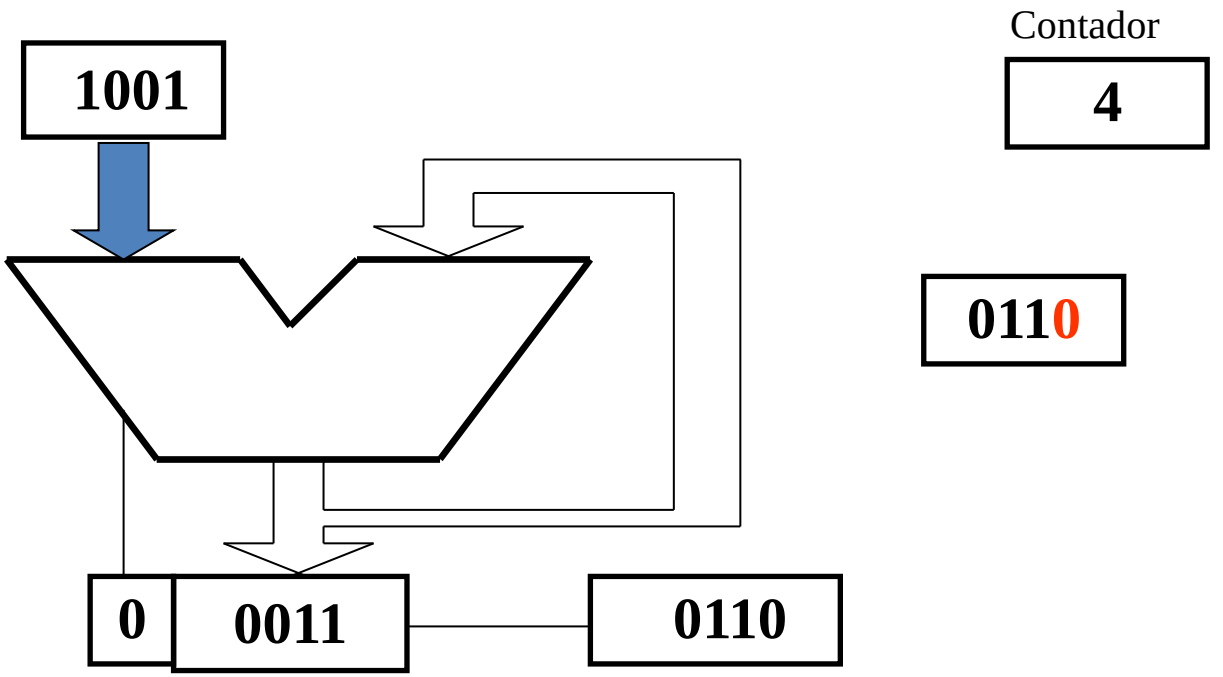
4

1100

```
 1001
 0110
-----
00000
 ↘
 0000 0
 1001+
-----
01001 0
 ↘
 0100 10
 1001+
-----
01101 10
 ↘
 0110 110
 0000+
-----
00110 110
 ↘
 0011 0110
```

Rotación circular de R2

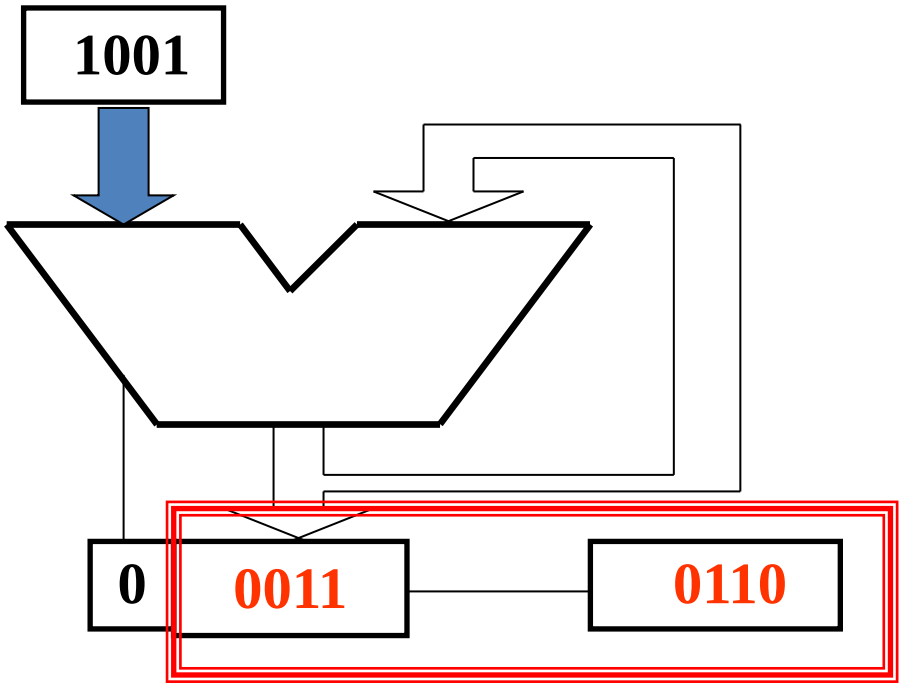
El registro R2 rota a la derecha de forma circular



¿Ha llegado al final?

Comprueba que el contador es 4 y acaba.

El resultado está en el registro R3 seguido de R4



Contador

4

0110

```
  1001
  0110
 -----
 00000
   ↘
  0000  0
  1001+
 -----
 01001  0
   ↘
  0100  10
  1001+
 -----
 01101  10
   ↘
  0110  110
  0000+
 -----
 00110  110
   ↘
  0011  0110
```

MULTIPLICACIÓN BINARIA SIN SIGNO: **SUMAS Y RESTAS**

Fundamentos (I)

Progresión geométrica de razón *r* y primer término a_1

$$S = a_1 + a_1 r + \ldots + a_1 r^{n-1}$$

Restando miembro a miembro y despejando S, resulta que la suma de los n términos de una progresión geométrica cuyo primer término es a_1 y cuya razón es r, viene dada por

$$rS = a_1 r + \ldots + a_1 r^{n-1} + a_1 r^n$$

$$S = \frac{a_1 r^n - a_1}{r - 1}$$

- También se puede ver de la forma

$$S = \frac{a_1 r^{n-1} r - a_1}{r - 1} = \frac{ultimoxrazon - primero}{r - 1}$$

- **Si r =2, el denominador se hace 1 y queda que la suma es el doble del último término, menos el primero**

Fundamentos (II)

El valor de un número expresado en binario se puede suponer la suma de una progresión geométrica de razón 2. Supongamos que tiene varios unos consecutivos

Ejemplo: el binario 0011100 equivalen a $2^4+2^3+2^2$

Según lo visto anteriormente,

$2^4+2^3+2^2 = 2\cdot 2^4-2^2 = 2^5-2^2$

Fundamentos (III)

Si consideramos la suma de las sucesivas potencias de 2 cuyos exponentes van desde m hasta m+n-1, tendremos la suma de una progresión geométrica de razón 2

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

- Si llamamos k = m + n, se puede poner

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

Fundamentos (IV)

Ejemplo $2^6 - 2^3 = 2^5 + 2^4 + 2^3$

Por tanto, multiplicar un número por 111000, es equivalente a multiplicarlo por 2^6

Posteriormente se multiplica por 2^3

Finalmente se restan

$A \cdot (0111000) = A \cdot (2^5 + 2^4 + 2^3) = A \cdot (2^6 - 2^3)$

Fundamentos (V)

En general, un número que tiene varias cadenas de 1's y 0's se puede descomponer en varios números con sólo una cadena de 1's:

$$0011001110 = 0011000000 + 0000001110$$

Así pues, multiplicar A por 0011001110 será equivalente a:

$$A \cdot 0011000000 + A \cdot 0000001110$$

Que equivale a:

$$A \cdot (2^7 + 2^6) + A \cdot (2^3 + 2^2 + 2^1)$$

O lo que es lo mismo a:

$$A \cdot (2^8 - 2^6) + A \cdot (2^4 - 2^1) = A \cdot (2^8 - 2^6 + 2^4 - 2^1)$$

Fundamentos (VI)

El método de Sumas y Restas se basa, pues, en multiplicar el multiplicando por -2 elevado a la posición en que aparezca un 1 en el multiplicador, moviéndonos de derecha a izquierda

Multiplicar el multiplicando por +2 elevado a la posición en que aparezca un 0, después de una cadena de 1's

Seguir desplazándonos hacia la izquierda y hacer lo mismo cuando encontremos un nuevo 1

De nuevo, se repite con la aparición de un 0

Y así sucesivamente hasta que ya no aparezca ningún 1

Se para cuando empieza la última cadena de ceros, haciendo, por tanto, la última suma: para asegurarse, es suficiente con añadir un 0 a la izquierda del 1 más significativo del multiplicador

Finalmente se suman TODOS los términos (de golpe o por partes)

Ejemplo

A=1100 (12) B=1010 (10)

Se completa B con un 0 a la izquierda y se calcula también –A (A en ca2) que necesariamente siempre empezará por 1

A=1100 B=01010 -A=10100

Se puede rellenar A con todos los **CEROS** a la izquierda que sean necesarios y –A con todos los **UNOS** a la izquierda que sean necesarios

A=0…01100 B=01010 -A=1…10100

Cada vez que aparece en el multiplicador B una cadena de unos (aunque sólo tenga un 1) se escribe A multiplicado por 2 elevado a la potencia donde está dicho 1 y cambiado de signo ($-A \cdot 2^s$)

Cada vez que aparece en el multiplicador B una cadena de ceros (aunque sólo tenga un 0) se escribe A multiplicado por 2 elevado a la potencia donde está dicho 0 y sin cambiar el signo ($+A \cdot 2^t$)

El proceso empieza con la primera cadena de unos que aparezca y termina con la última cadena de ceros (último 0), **SIEMPRE MOVIÉNDONOS DE DERECHA A IZQUIERDA EN EL MULTIPLICADOR**

Ejemplo (continuación)

12 x 10 = 120
(base 10)

A=1100 B=01010 -A=1…10100

Valor		7	6	5	4	3	2	1	0	Suma
-24	$-A\cdot 2^1$	1	1	1	0	1	0	0	0	→ -24
+48	$+A\cdot 2^2$	0	0	1	1	0	0	0	0	→ +24
-96	$-A\cdot 2^3$	1	0	1	0	0	0	0	0	→ -72
+192	$+A\cdot 2^4$	1	1	0	0	0	0	0	0	→ **+120**
		0	**1**	**1**	**1**	**1**	**0**	**0**	**0**	

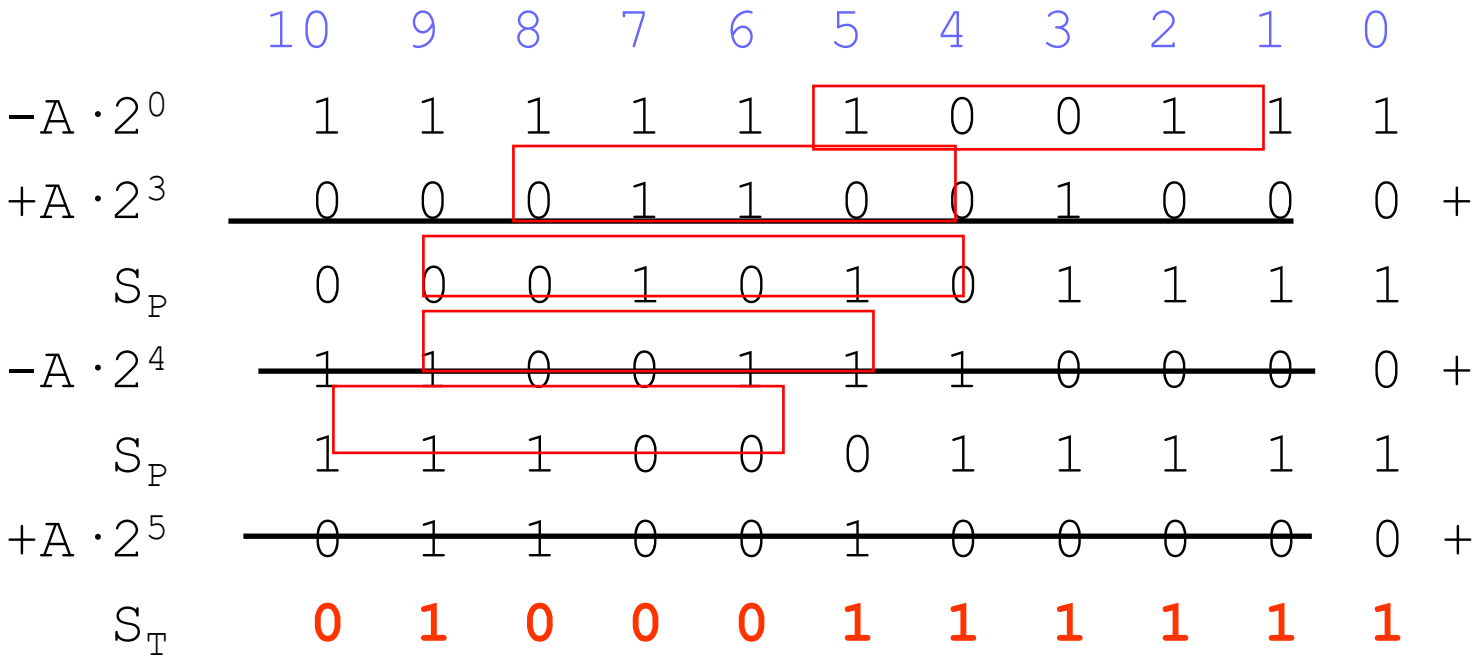
(-2 + 4 - 8+ 16) x12 = 120

Otro ejemplo

25 x 23 = 575

A= 0…011001 -A = 1…100111

B= 010111



$$575 = 2^9 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$

Necesidades

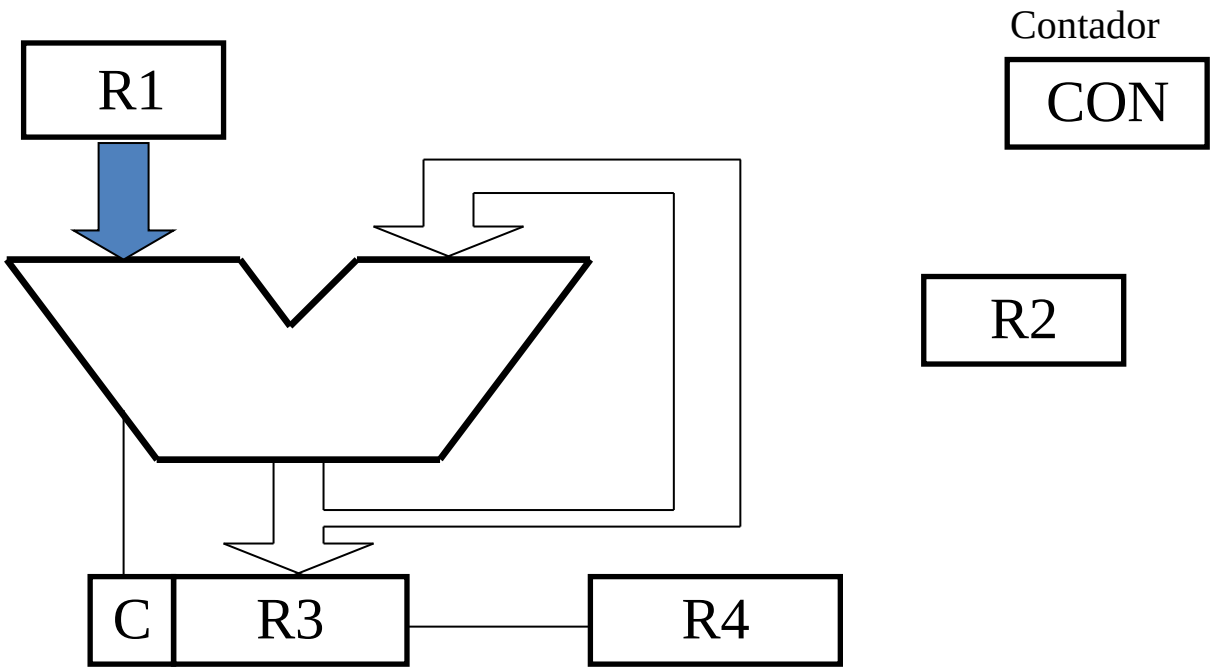
El producto de un número binario de n bits con otro de m bits necesita m+n bits

Sea A el multiplicador y B el multiplicando

En el caso de que m=n, se necesitan dos registros iguales para almacenar el resultado

Sean R3 y R4 los registros donde se va almacenando el resultado

Arquitectura



Algoritmo de Sumas y Restas

Se inicializan a 0 el contador de bits del multiplicador y el registro R3

Se carga R1 con el multiplicando A y R2 con el multiplicador B

Se analiza $R2_0$, el LSB de R2, y se llama $R2_{-1}$ al bit donde irá a parar $R2_0$ cuando R2 rote a derechas. $R2_{-1}$ se inicializa a 0

Se leen los bits del multiplicador de derecha a izquierda

El proceso comienza cuando se lee el **PRIMER UNO**, haciendo R3←R3-R1

En los sucesivos bucles se aplica el siguiente criterio:

- Si **es el principio de una cadena de unos**, se hace la resta R3←R3 – R1
- Si **es el principio de una cadena de ceros**, se hace la suma R3← R3 + R1

Se incrementa el contador

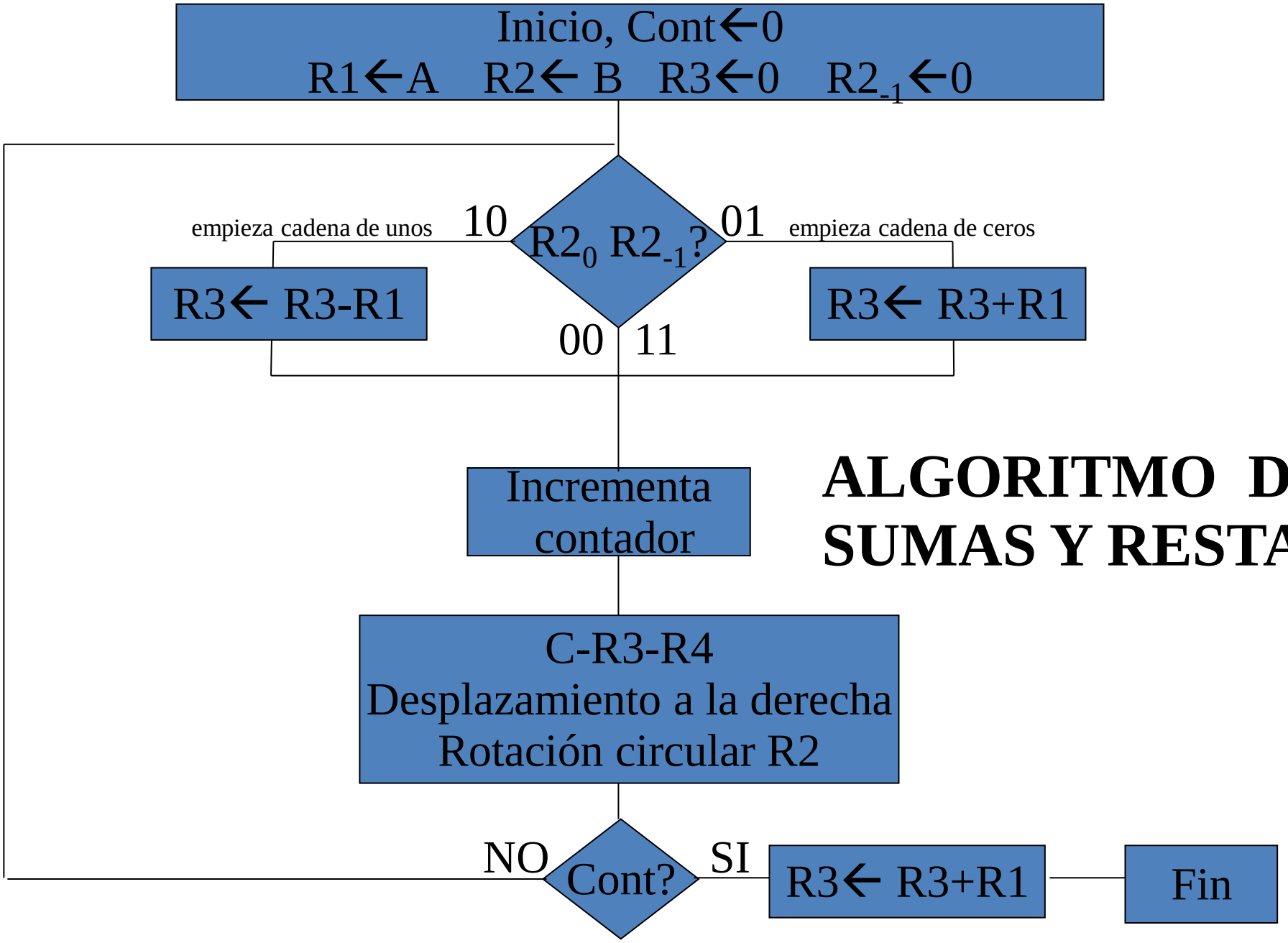
Se rota hacia la derecha una posición el conjunto C-R3-R4, con extensión de signo

Se realiza una rotación circular a derecha de R2

Se mira si el contador ha llegado hasta el final

En caso negativo se repite el proceso, volviendo a preguntar por el LSB de R2

En caso afirmativo se suma R3+R1 dejando el resultado en R3, con lo que se finaliza el proceso y el resultado queda en R3-R4. **ESTO ES EQUIVALENTE A EJECUTAR EL ÚLTIMO CERO QUE SIEMPRE SE SUPONE QUE TIENE UN NÚMERO POSITIVO**

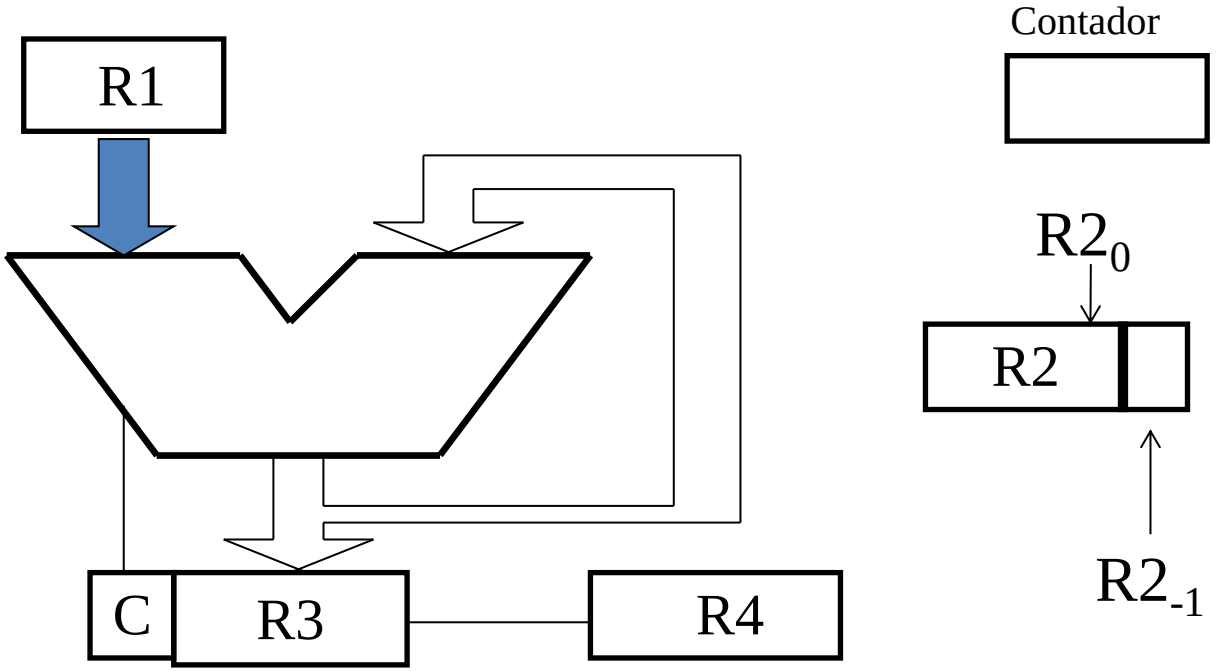


ALGORITMO DE SUMAS Y RESTAS

Ejemplo

Multiplicar A=1100 por B=1010

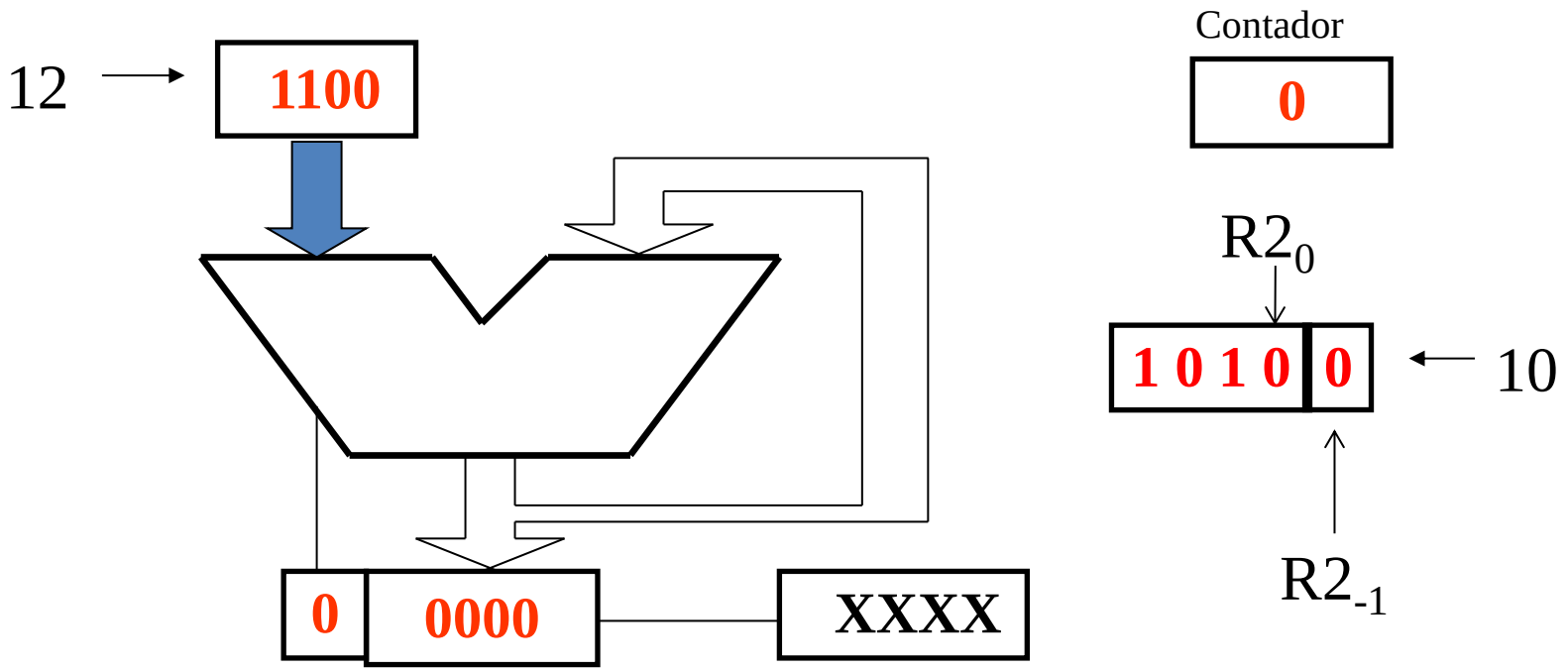
El resultado debe ser 12x10=120 (en binario 0111 1000)



Inicialización

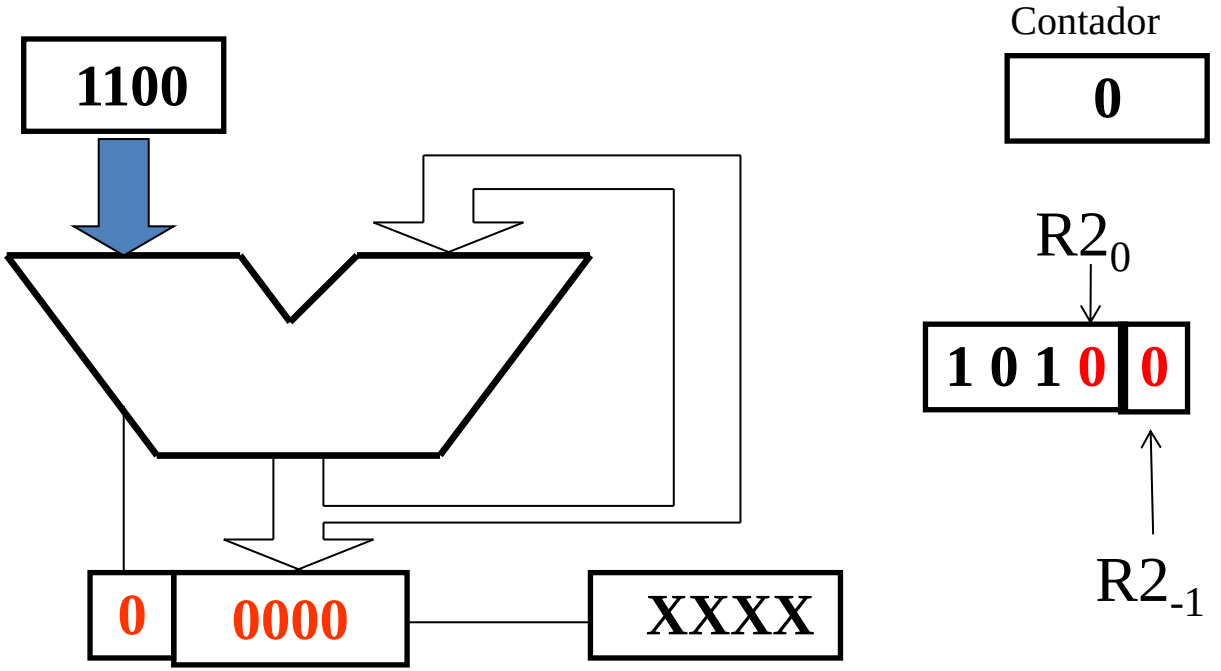
Se carga A en R1 y B en R2.

Se pone a 0 C, el registro R3, el Contador y $R2_{-1}$



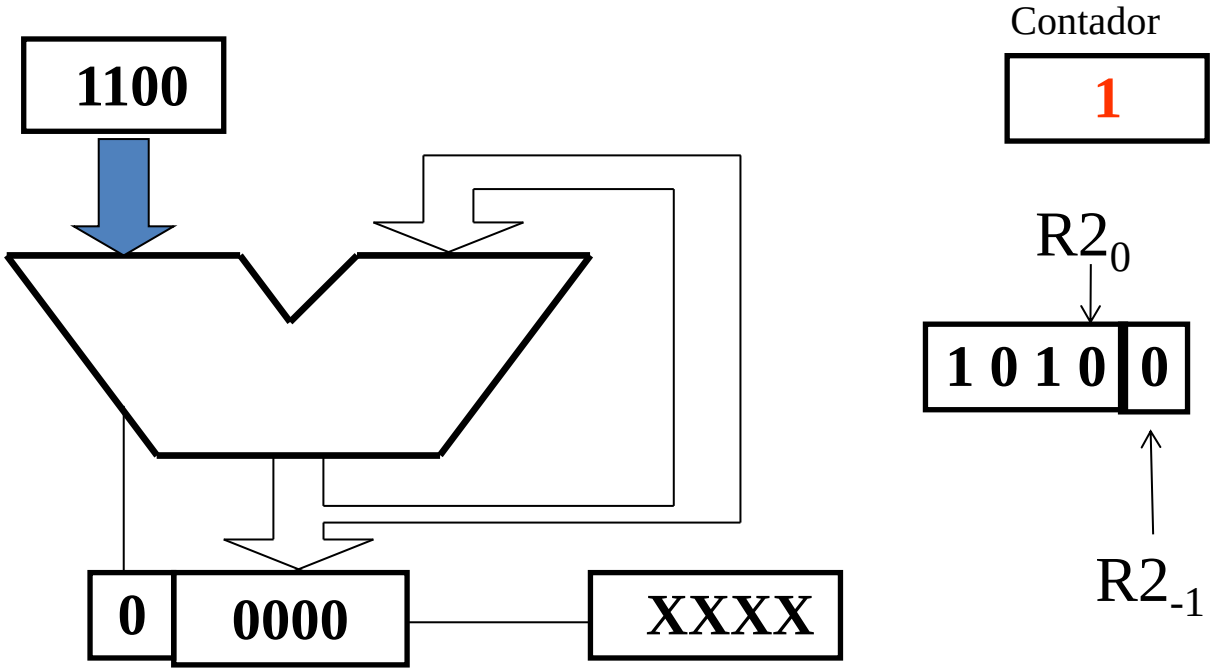
Comprobación de $R2_0$ y acción

Como el LSB de R2 es el primer 0 no se hace nada (todavía no ha llegado el primer 1) y se deja el resultado en C-R3 como estaba



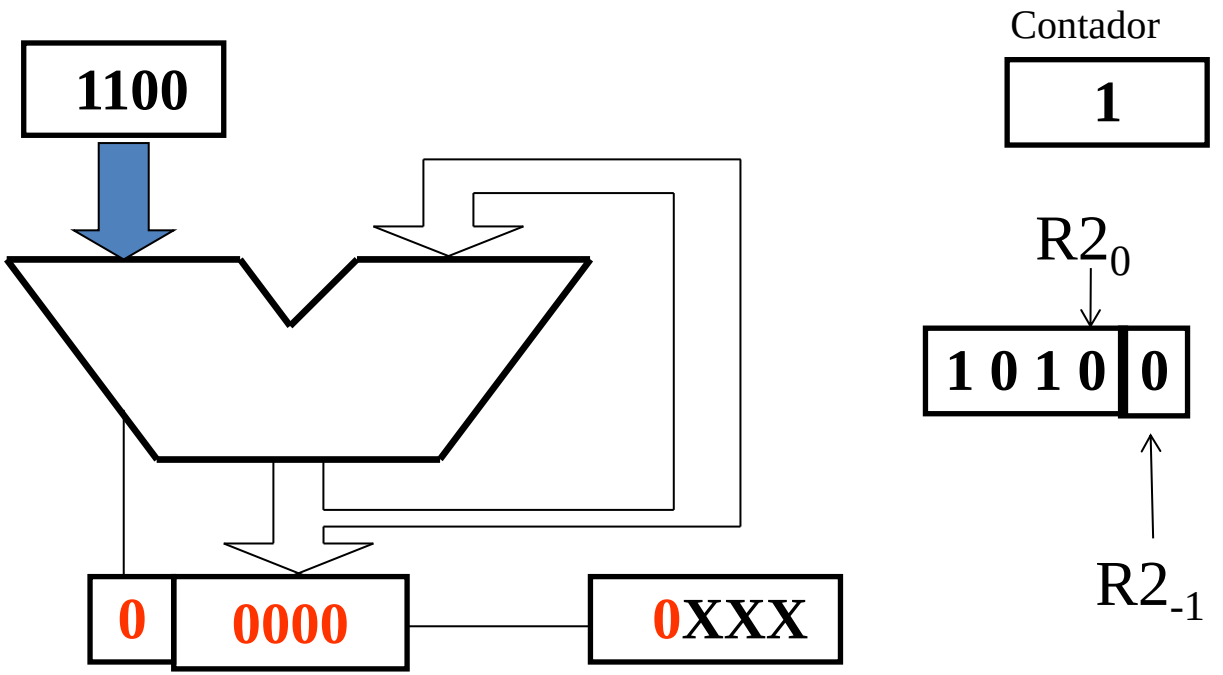
Incremento contador

Se incrementa el contador en una unidad



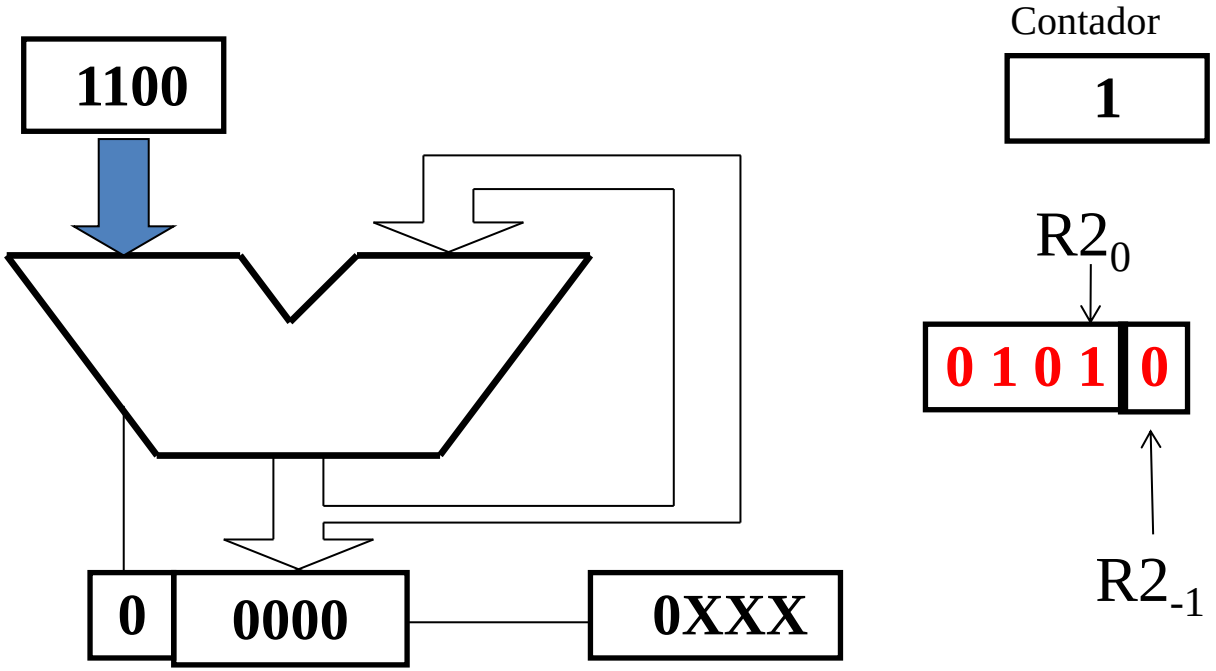
Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



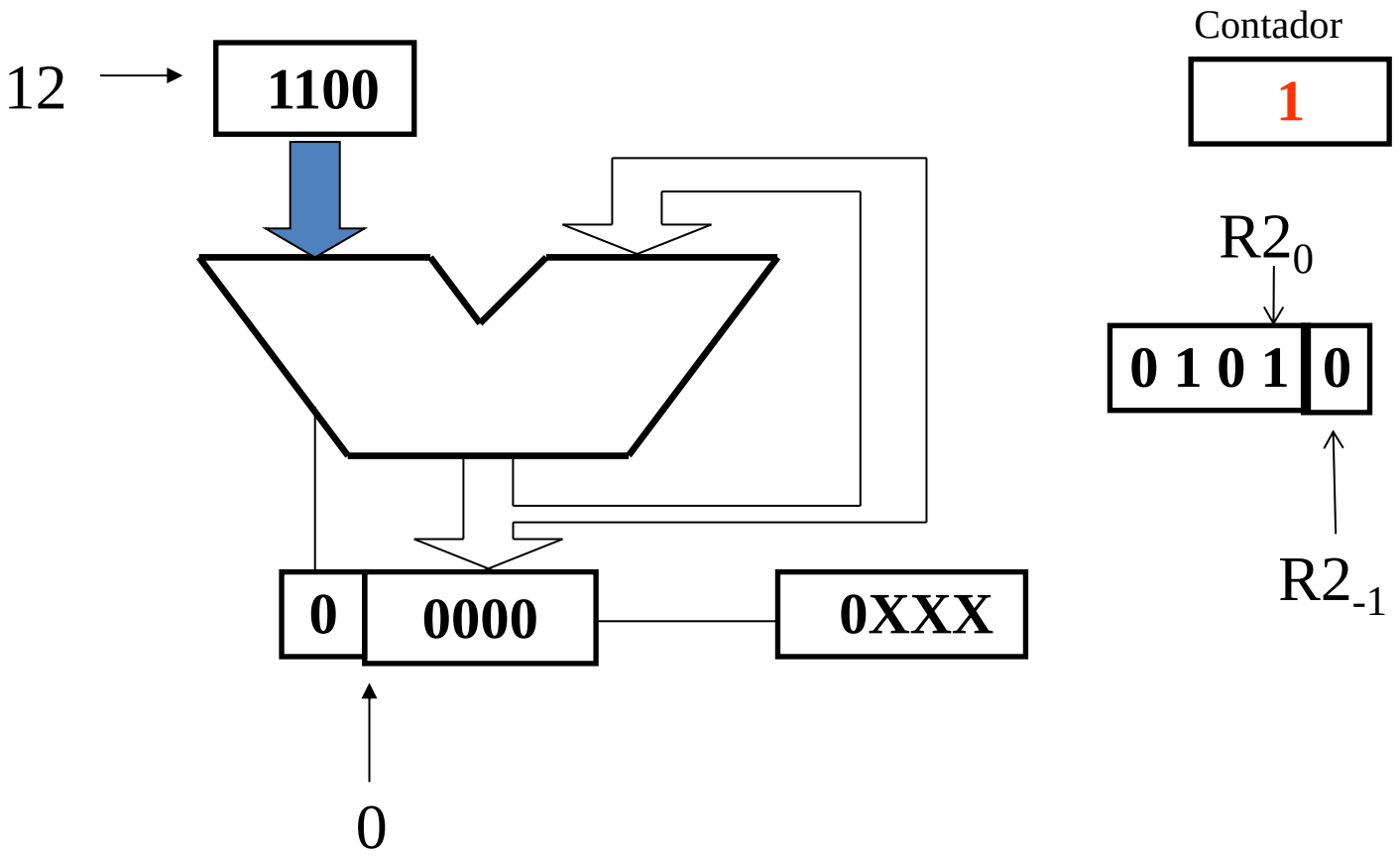
Rotación circular de R2

El registro R2 rota a la derecha de forma circular



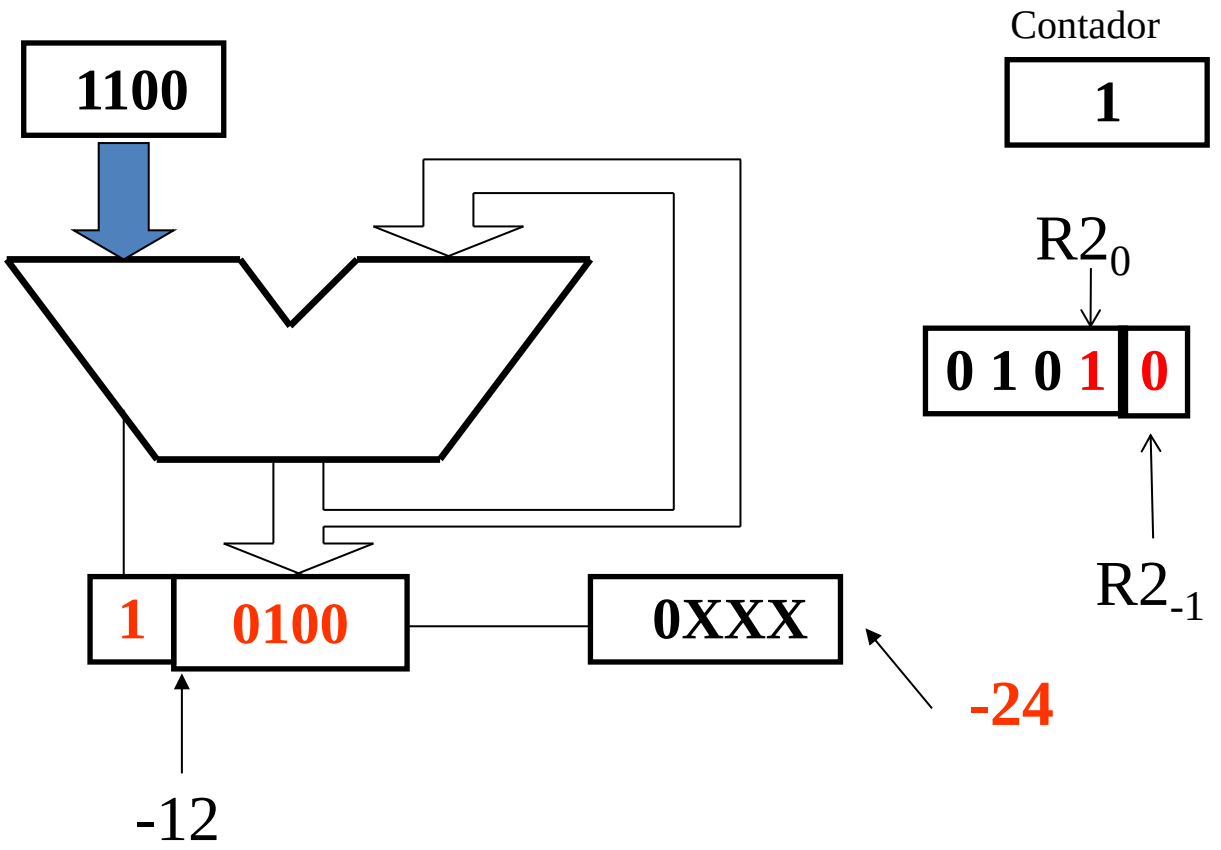
¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



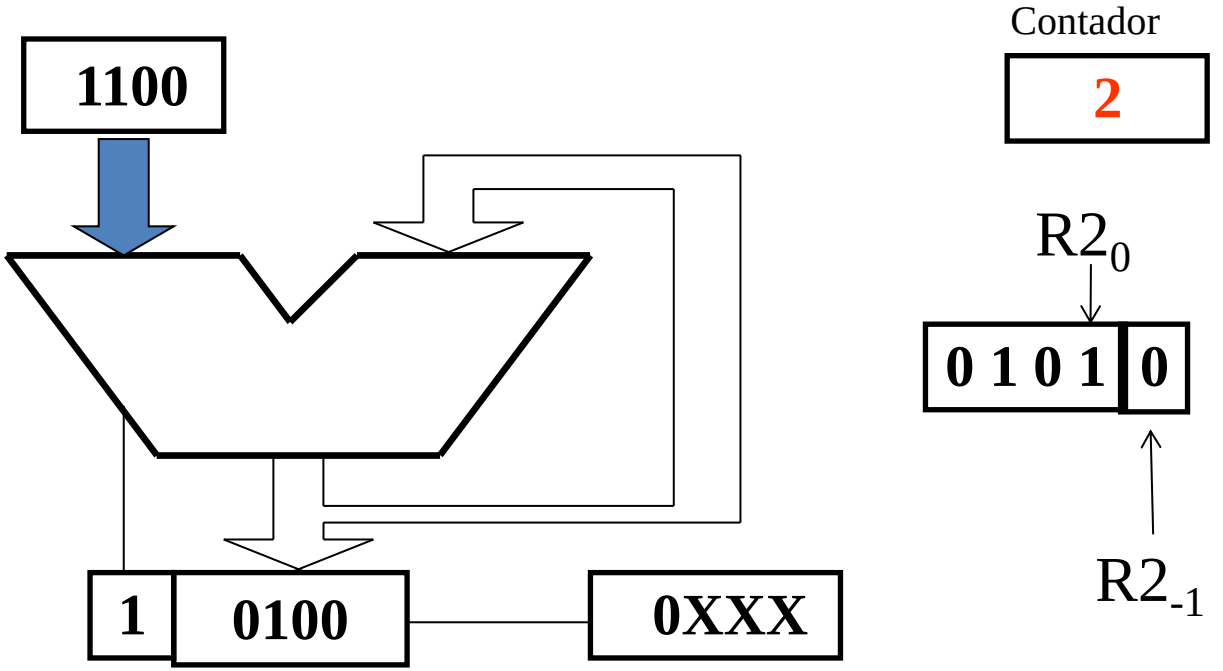
Comprobación de $R2_0$ y acción

Como el LSB de R2 es 1 (EMPIEZA UNA CADENA DE UNOS) se hace R3 - R1 y se deja el resultado en C-R3, quedando 0 – 12 = -12 en complemento a 2



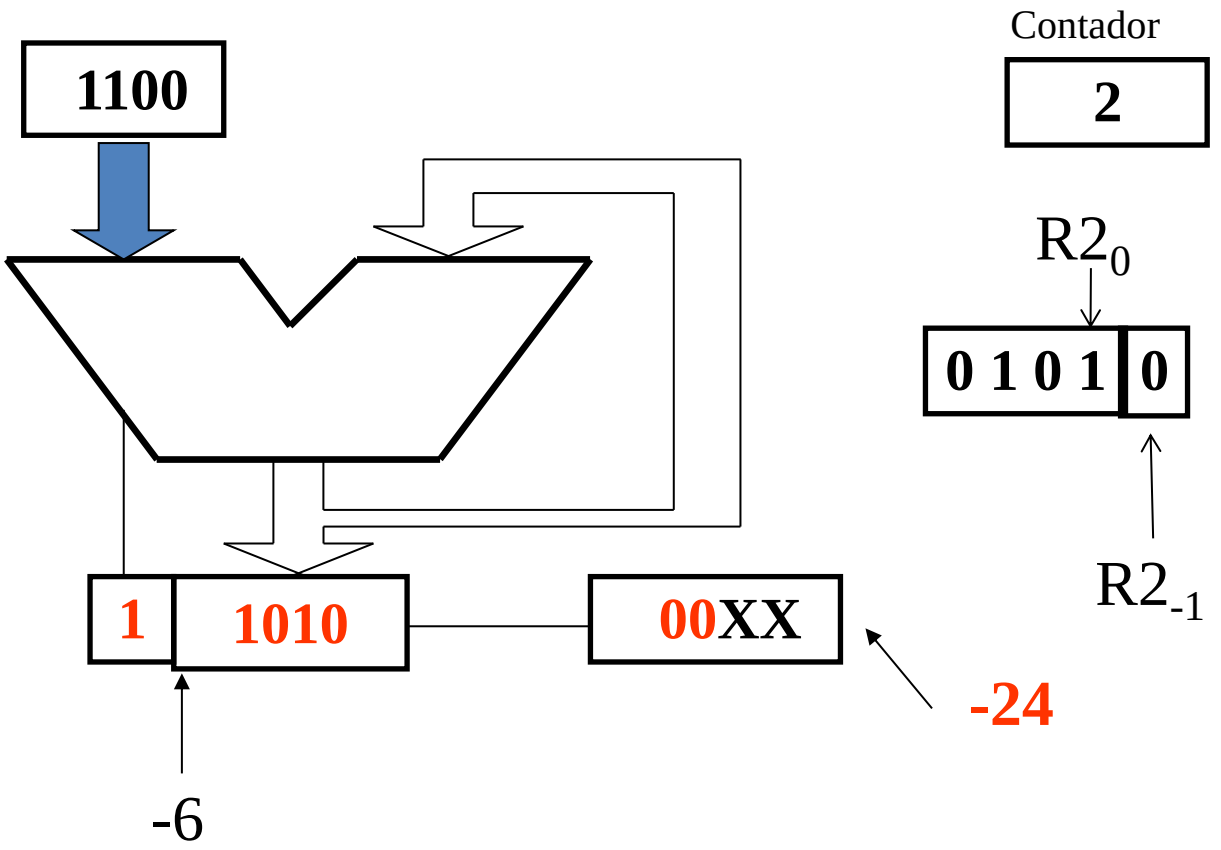
Incremento contador

Se incrementa el contador en una unidad



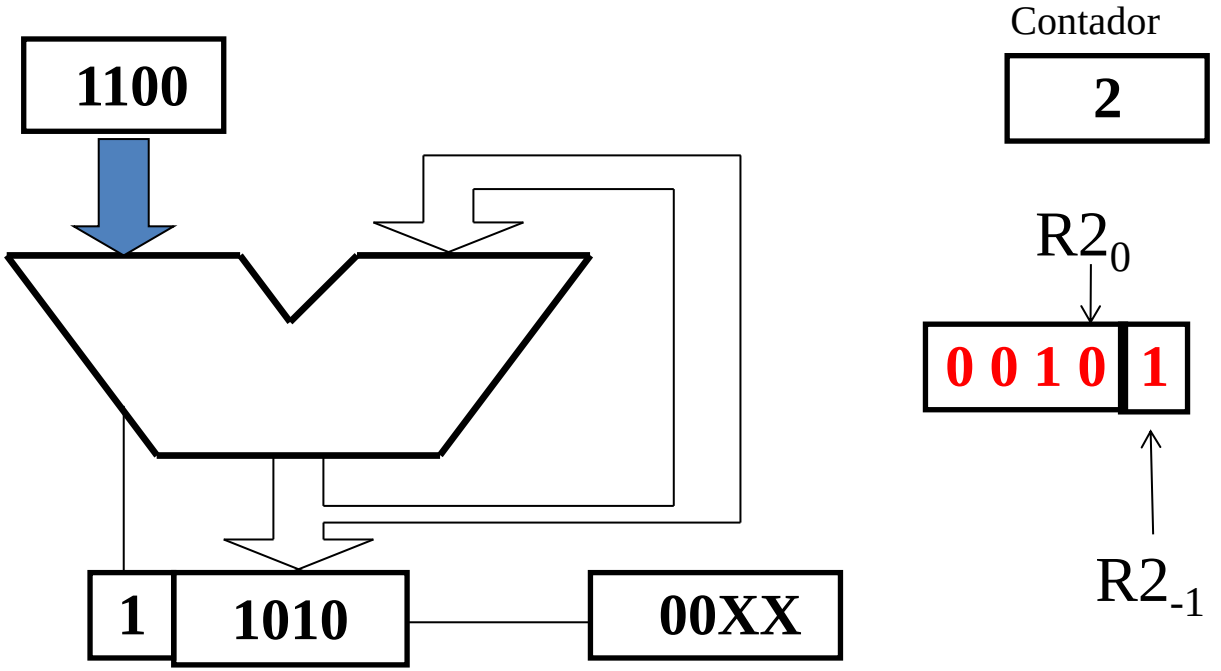
Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



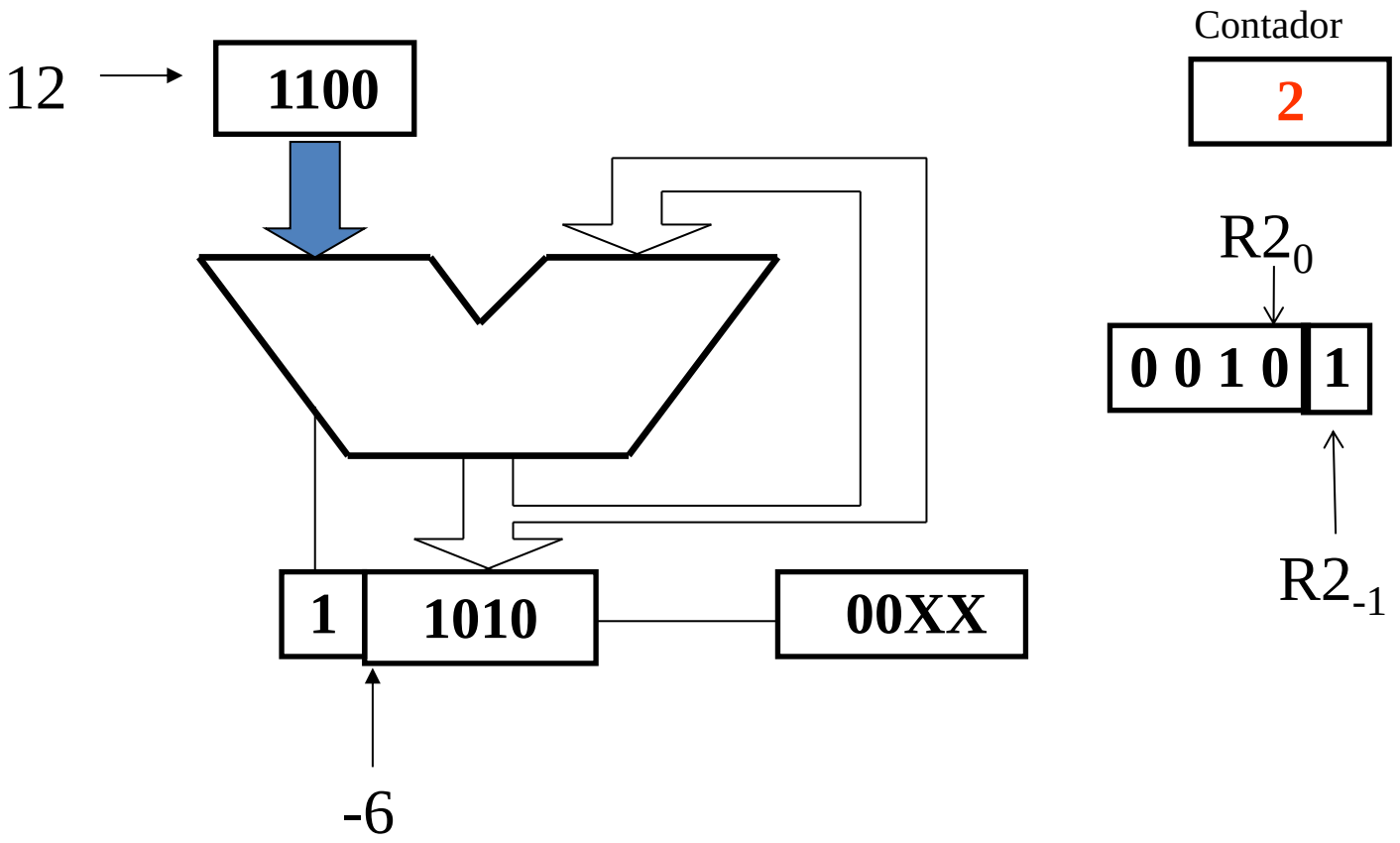
Rotación circular de R2

El registro R2 rota a la derecha de forma circular



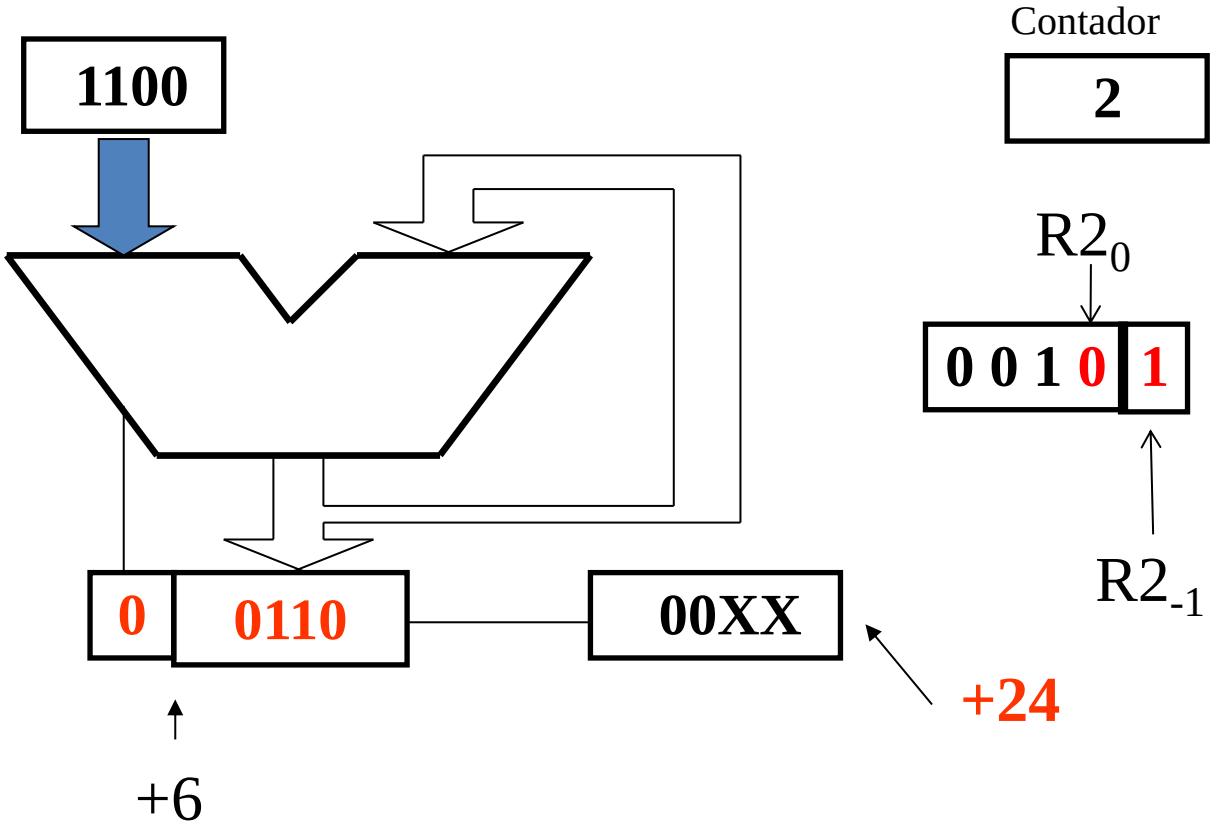
¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



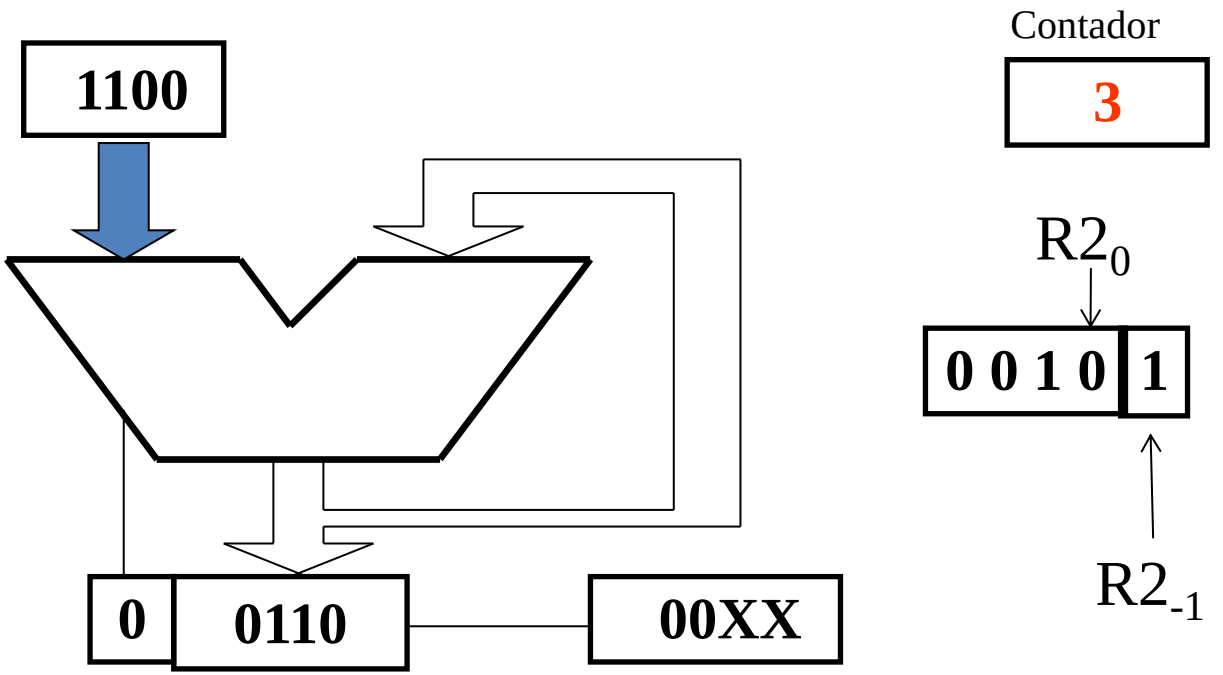
Comprobación de $R2_0$ y acción

Como el LSB de R2 es 0 (EMPIEZA UNA CADENA DE CEROS) y ya llegó un 1 anteriormente, se suman R3 + R1 y se deja el resultado en C-R3, quedando -6 + 12 = +6



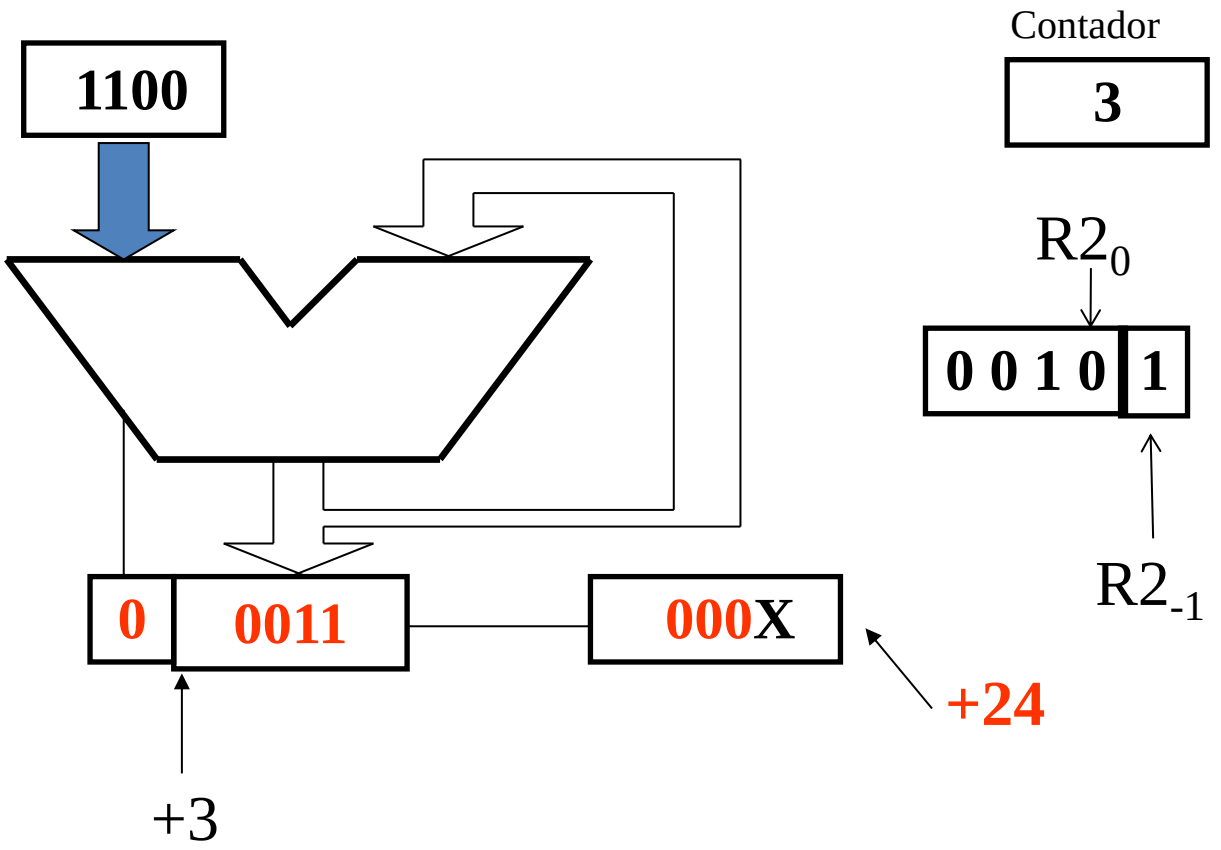
Incremento contador

Se incrementa el contador en una unidad



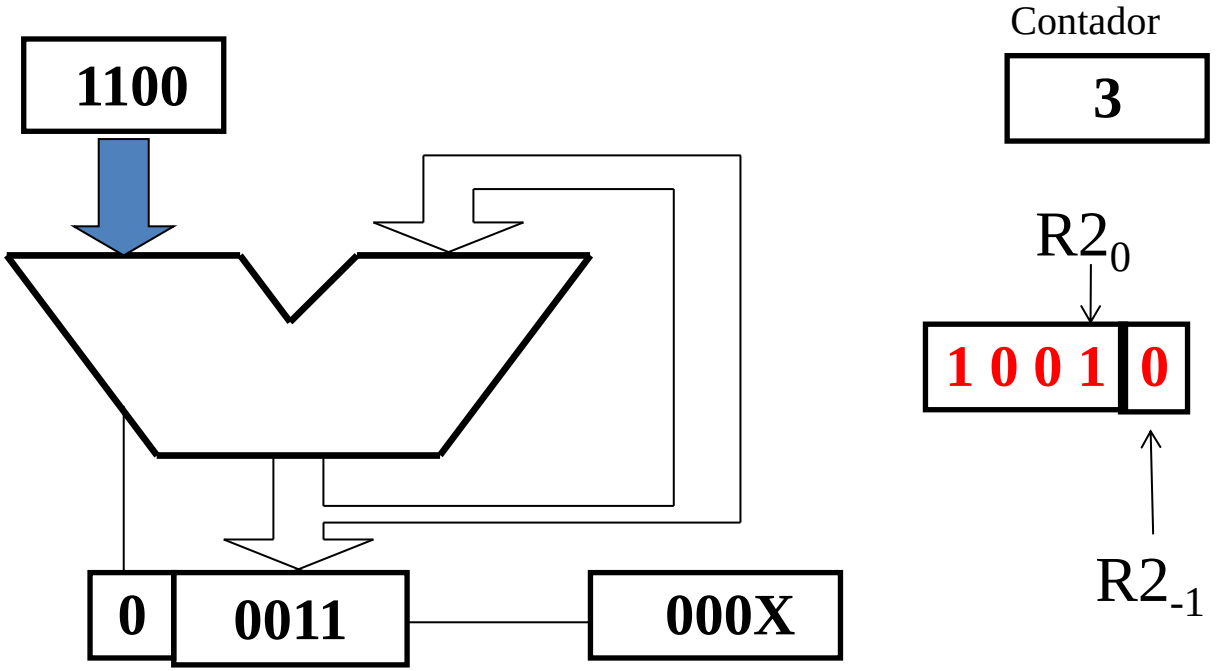
Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



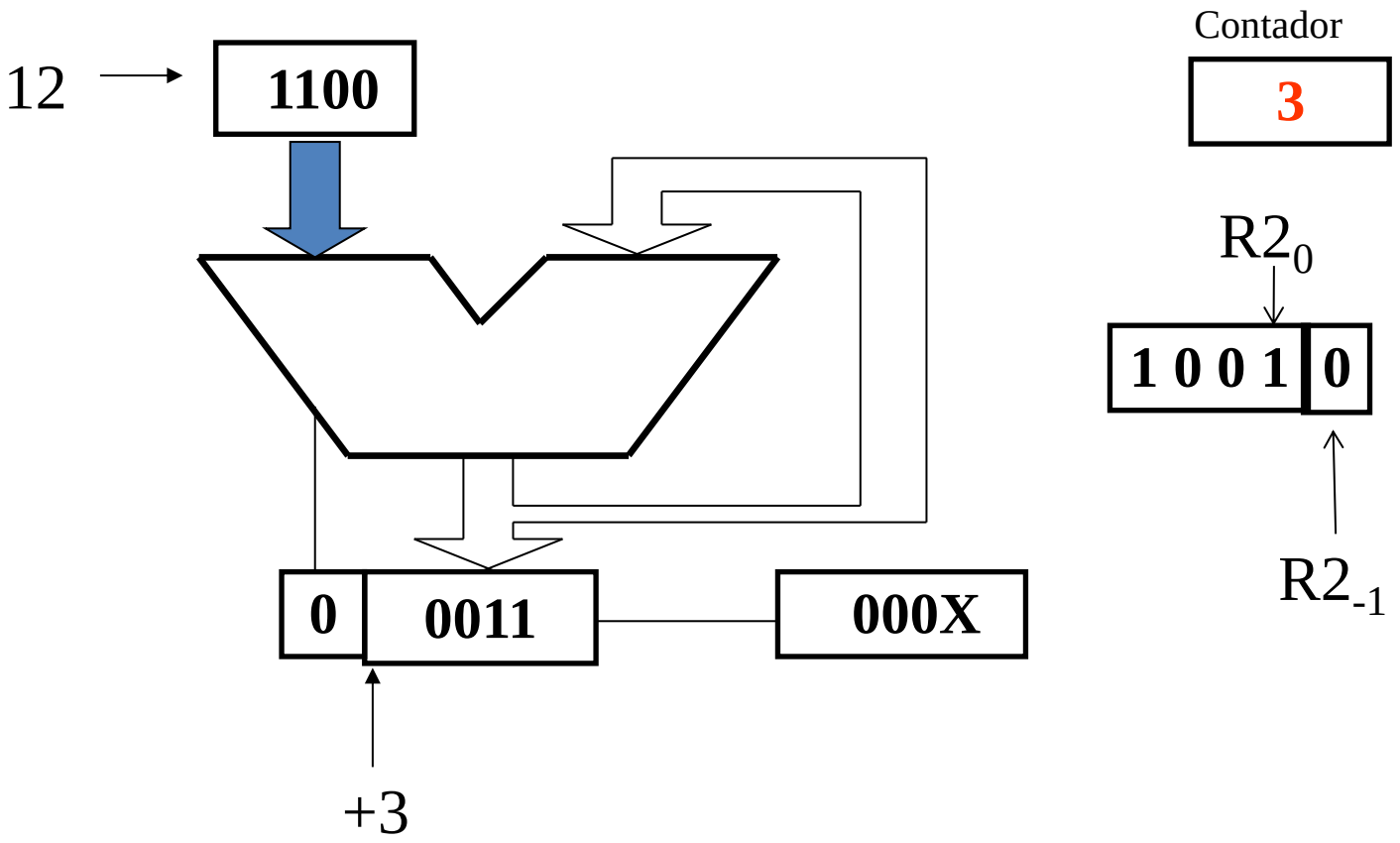
Rotación circular de R2

El registro R2 rota a la derecha de forma circular



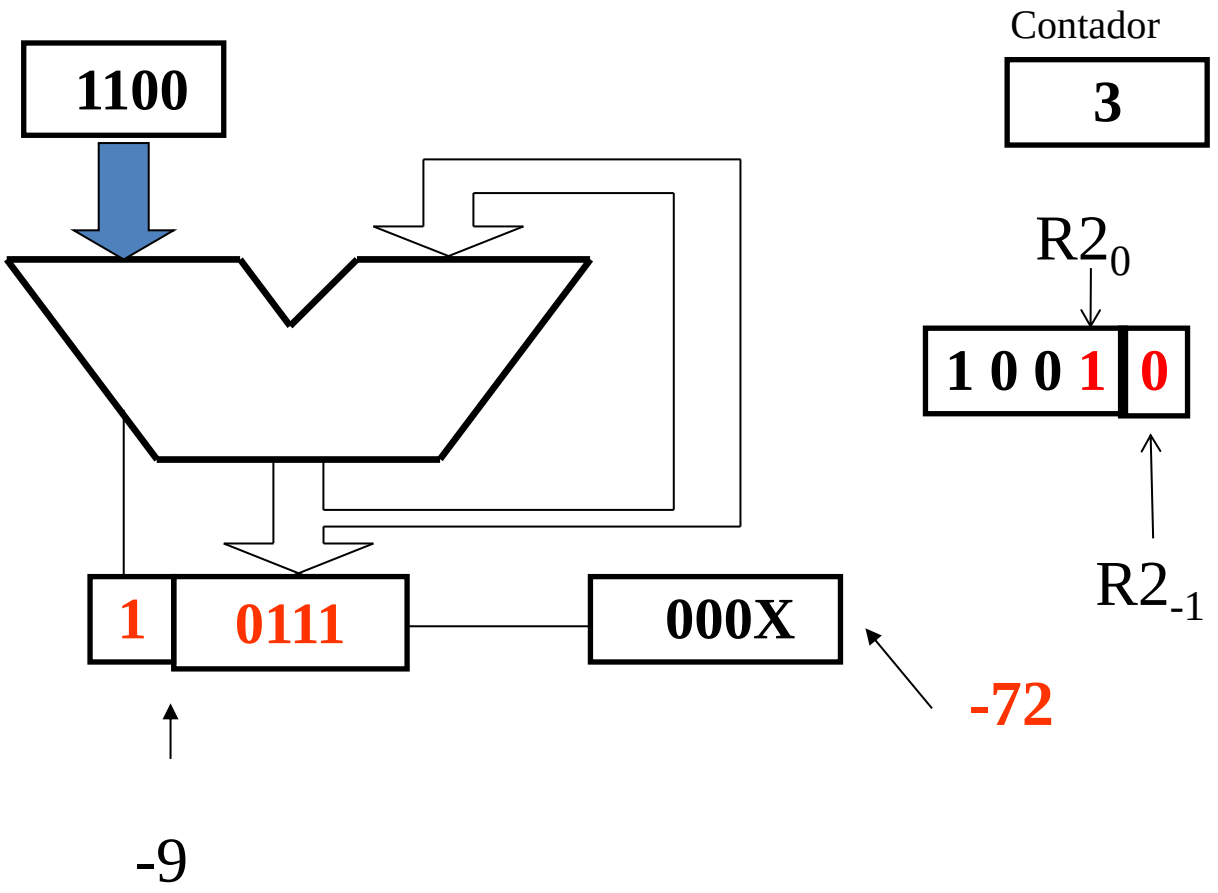
¿Ha llegado al final?

Comprueba que el contador no es 4 y repite el bucle



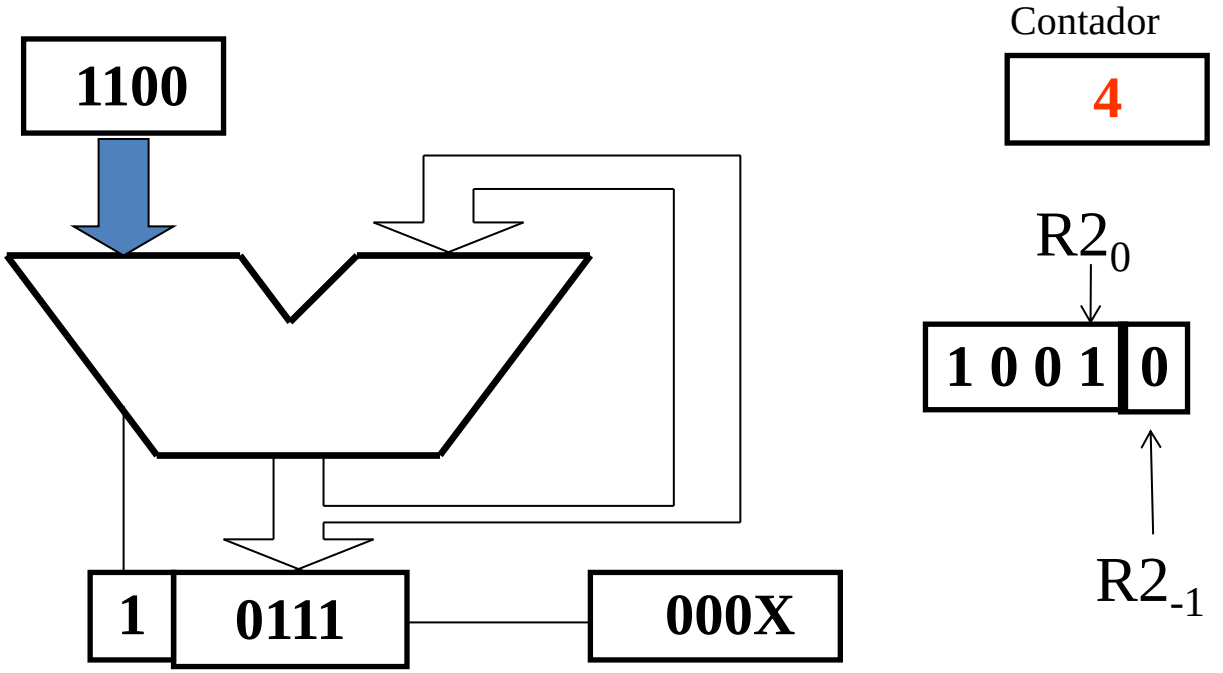
Comprobación de $R2_0$ y acción

Como el LSB de R2 es 1 (EMPIEZA UNA CADENA DE UNOS) se hace R3 - R1 y se deja el resultado en C-R3, quedando 3 – 12 = -9



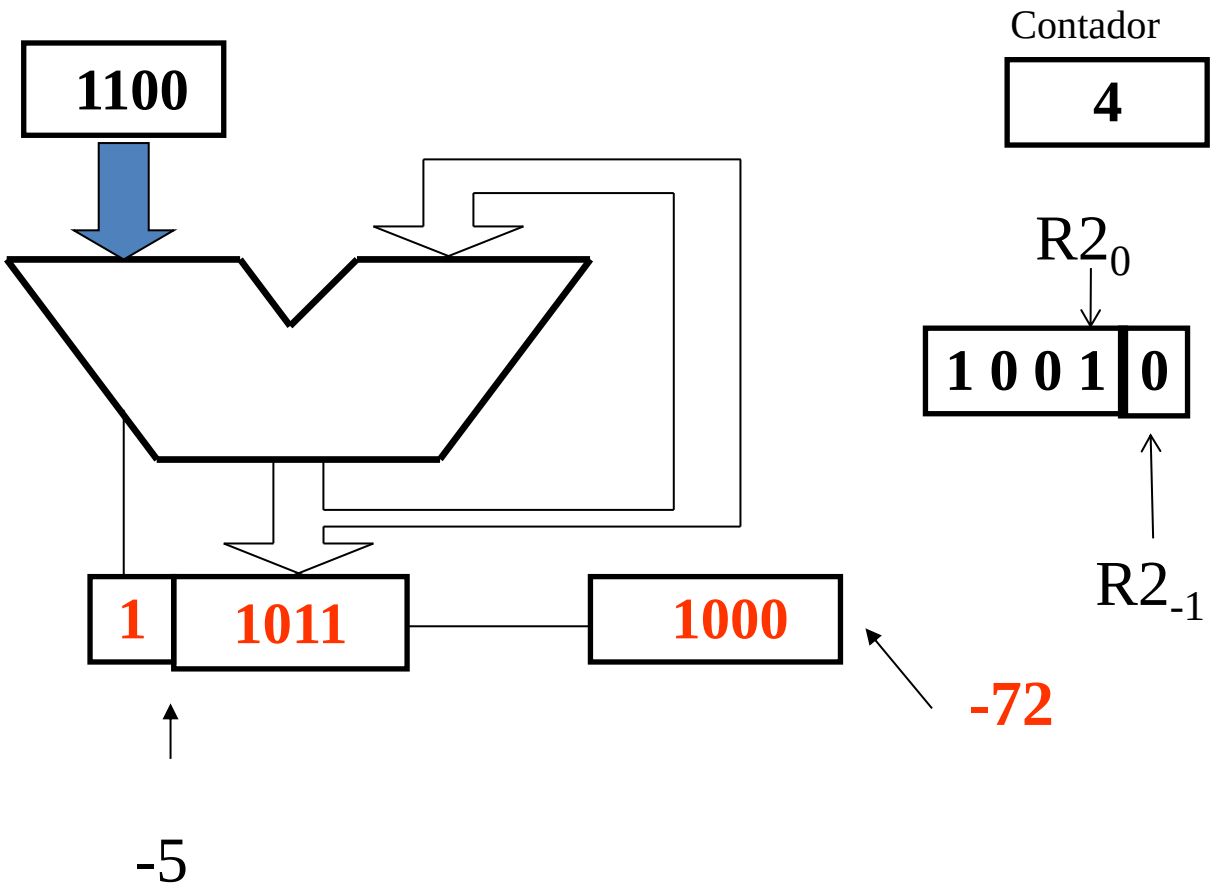
Incremento contador

Se incrementa el contador en una unidad



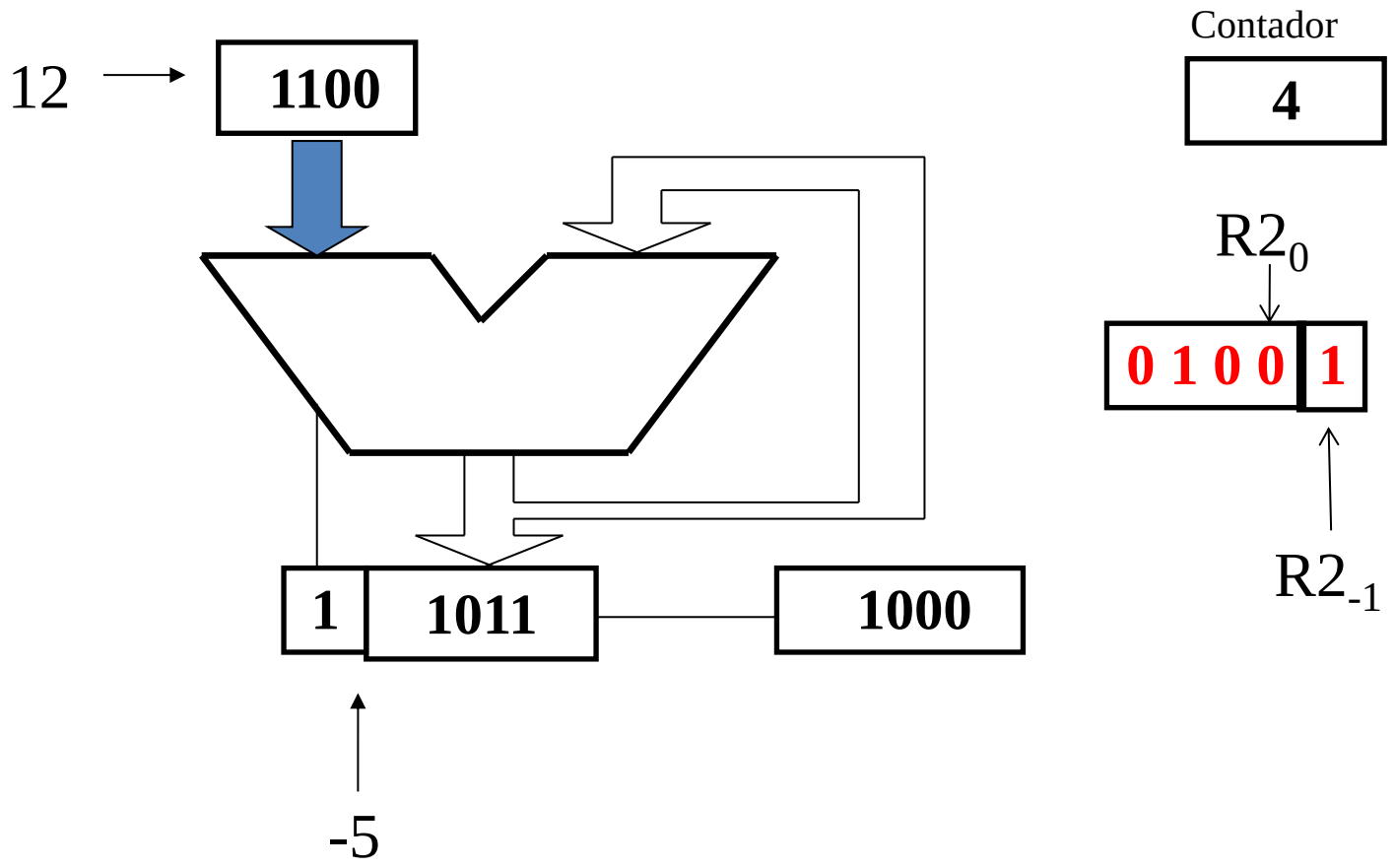
Desplazamiento a derecha

Se desplaza el conjunto C-R3-R4 una posición a la derecha



Rotación circular de R2

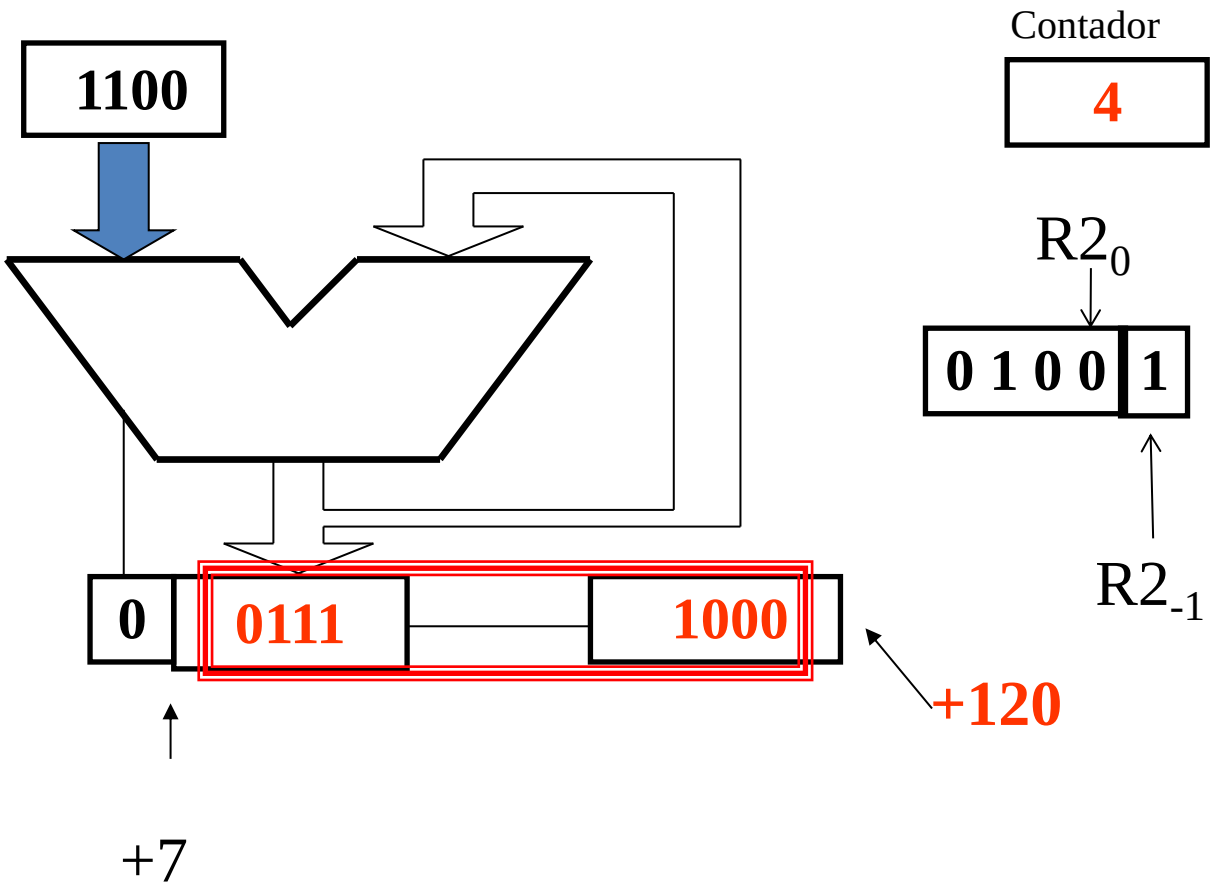
El registro R2 rota a la derecha de forma circular



¿Ha llegado al final?

Comprueba que el contador es 4 y acaba, sumando previamente R3 + R1 y dejándolo en R3, sumando -5 + 12 = +7

El resultado está en el registro R3 seguido de R4. C no es necesario para el resultado



MULTIPLICADOR BINARIO CON SIGNO:

ALGORITMO DE BOOTH

Multiplicación de números de distinto signo

Una forma de hacerlo es multiplicar los valores absolutos con los algoritmos vistos hasta ahora y asignarle el signo que corresponda, conocido a priori

Si los dos números son del mismo signo, el resultado será positivo. En caso contrario será negativo.

Cuando el resultado sea negativo, se realizará además el complemento a 2

Idea

Sirve para números enteros

Analiza los bits del multiplicador

Toma decisiones en función de que comience una cadena de ceros, de unos o ninguno de los casos

Desplaza a la derecha los resultados provisionales

Fundamentos (I)

Supongamos que vamos a multiplicar un multiplicando A positivo (expresado en binario sin signo con n bits) por un multiplicador B negativo, expresado en complemento a 2 también con n bits

El producto A·B es igual que si se multiplicara sin signo $A\cdot(2^n-|B|)$

Ejemplo, con 4 bits, 4x(-3) sería equivalente a 4x13, ya que la representación de -3 en ca2 con 4 bits coincide con la de 13 en binario natural con 4 bits, en ambos casos es 1101

Sea $R^*=A\cdot(2^n-|B|)$ lo que sale aplicando el algoritmo de sumas y restas, en nuestro ejemplo 4x13=52 (binario 00110100)

Sea $R=2^{2n} - A\cdot|B|$ lo que debería salir (el resultado buscado). Téngase en cuenta que la potencia 2^{2n} queda fuera del campo de representación del producto. Las representaciones de R y del resultado $- A\cdot|B|$ coinciden en sus 2n bits menos significativos

En nuestro ejemplo, tanto 244 en binario natural como -12 en complemento a 2 se representan como 11110100 con 8 bits

Fundamentos (II)

La relación entre R* y R es la siguiente

$R^* = A\cdot(2^n-|B|) = 2^{2n} - 2^{2n} + A\cdot 2^n - A\cdot|B|$

Como $R=2^{2n}-A\cdot|B|$

$R^* = R + A\cdot 2^n - 2^{2n}$, donde el valor de 2^{2n} en los 2n bits menos significativos (donde cabe el producto) es irrelevante

Por tanto, $R = R^* - A\cdot 2^n$

Es decir, el resultado buscado es el resultado que se obtiene con el algoritmos de sumas y restas, al que finalmente se le resta $A\cdot 2^n$. Simplemente se aplica dicho algoritmo, ACABANDO con un 1, lo cual siempre ocurre puesto que el bit de signo de un número negativo en complemento a 2 es siempre 1

Mecánica

Se inicializan a:

- Cero: R3, el contador, el bit $R2_{-1}$
- Al valor del multiplicando: R1
- Al valor del multiplicador R2 y R4

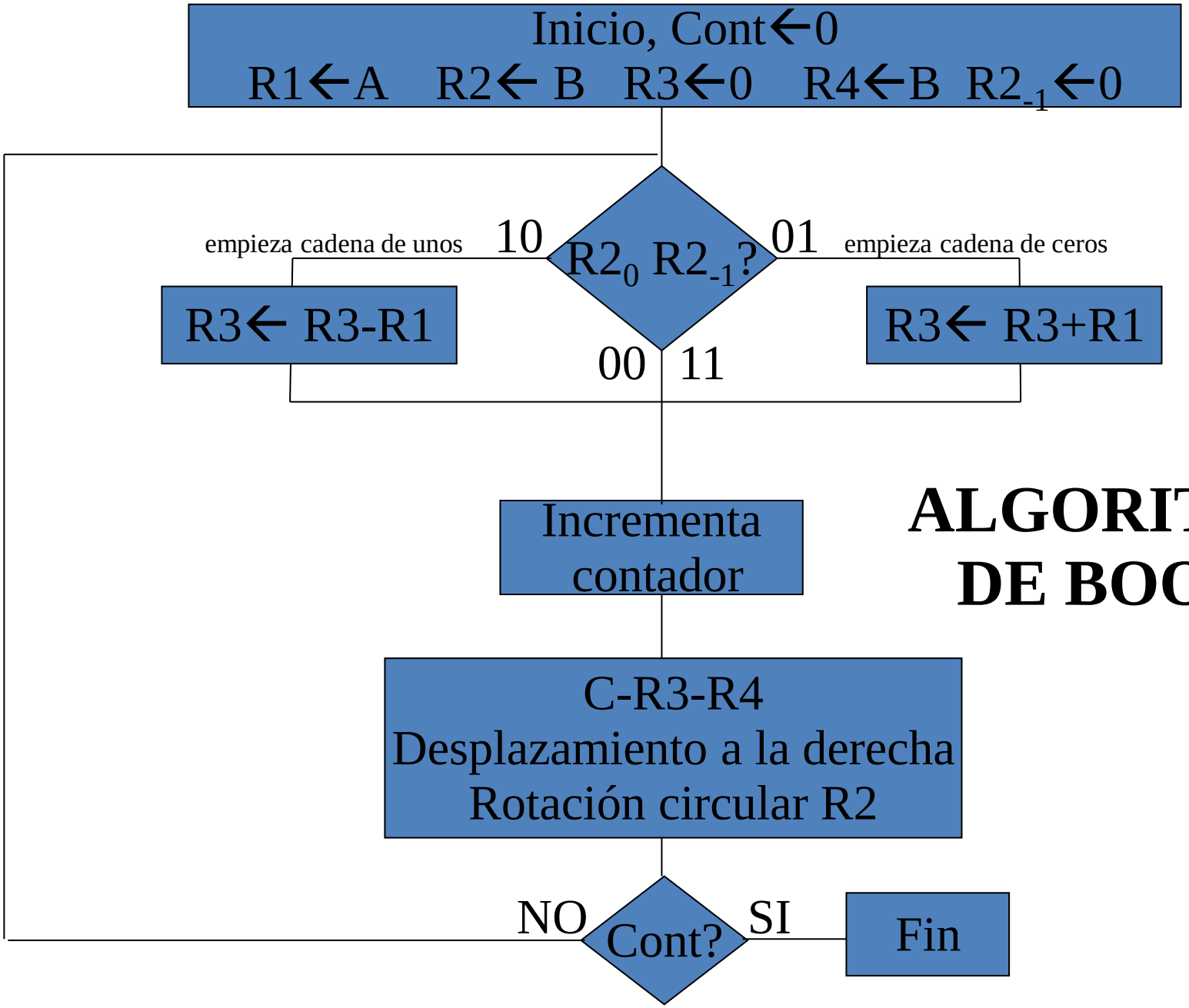
Se leen los bits $R2_0$ y $R2_{-1}$

Según sea un valor 10, 01 o el resto (00 o bien 11) se toma la decisión de restar, sumar o no hacer nada, respectivamente

Se incrementa el contador

Se desplaza todo a la derecha, incluido R2

Se pregunta si ha acabado. En caso contrario se repite el bucle



ALGORITMO DE BOOTH

Ejemplo

A=11001 (+25)

Se trabaja con n=5 bits

B=10111 (-9)

A x B = -225 (1 1100011111)

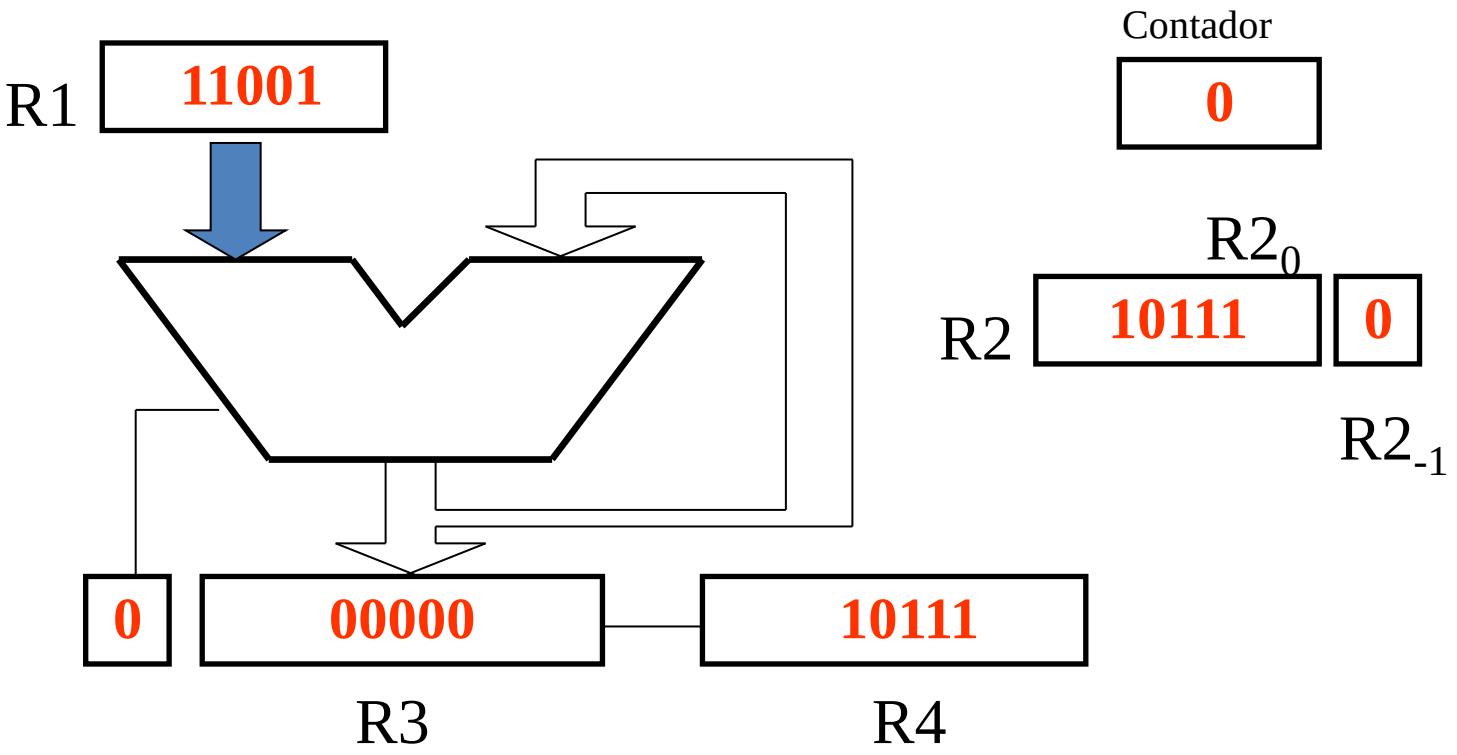
ca2 (1 1100011111) = 0 0011100001

$0\ 0011100001_{2)} = 1+32+64+128 = 225$

Inicialización

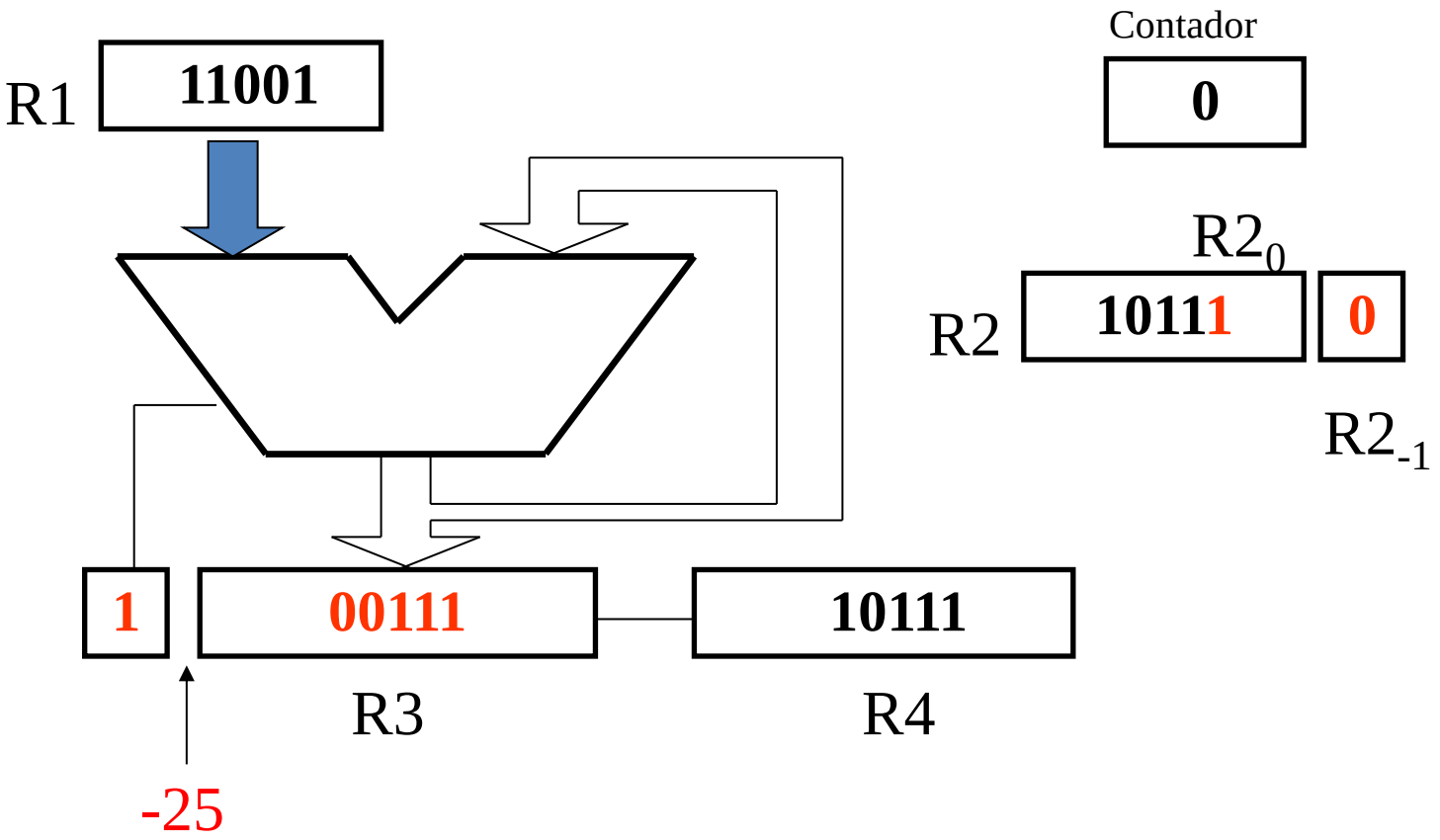
Se carga A=11001 en R1 y B=10111 en R2 y R4

Se ponen a 0 el contador, el bit $R2_{-1}$ y R3



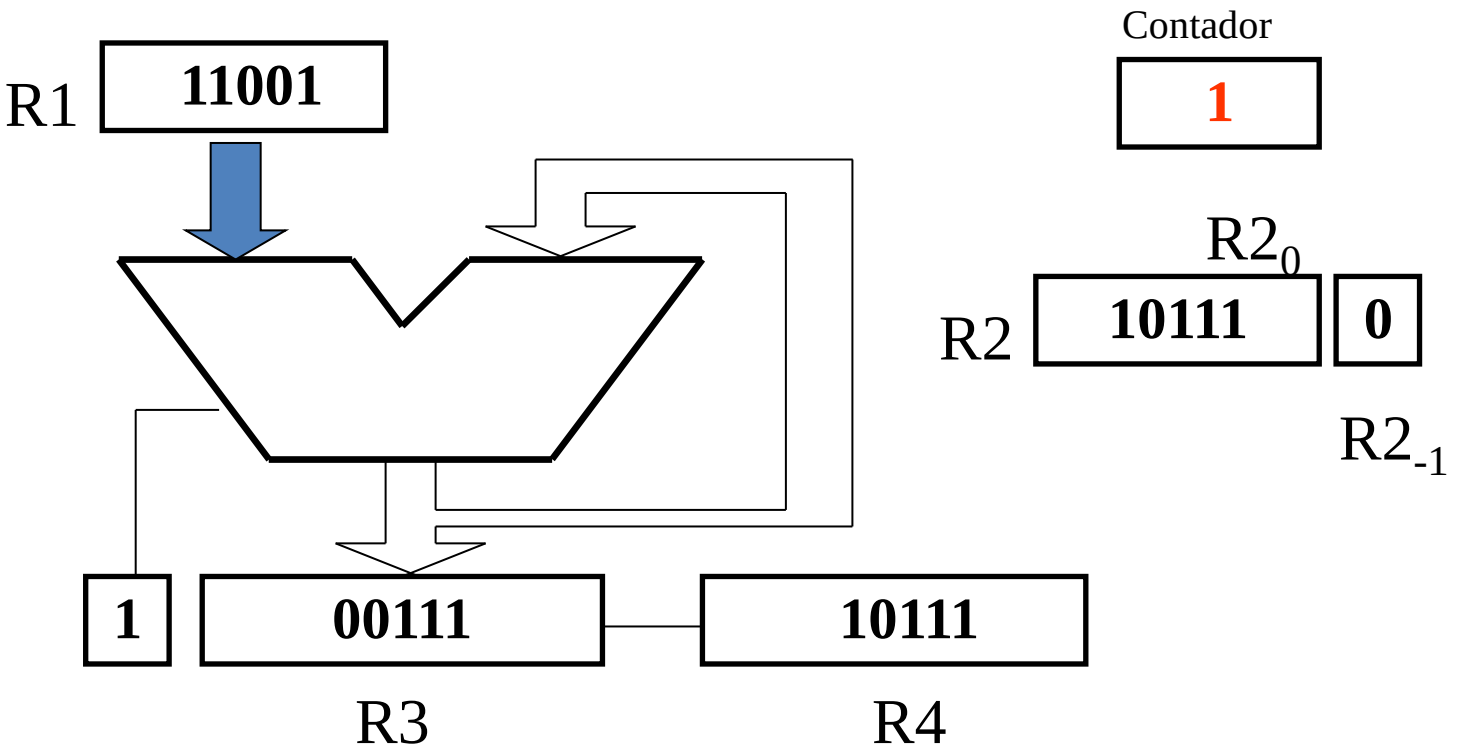
Comprobación multiplicador y decisión

Comienza cadenas de unos, R3←R3-R1, es decir, se hace 0 00000 + 1 00111 = 1 00111



Incremento contador

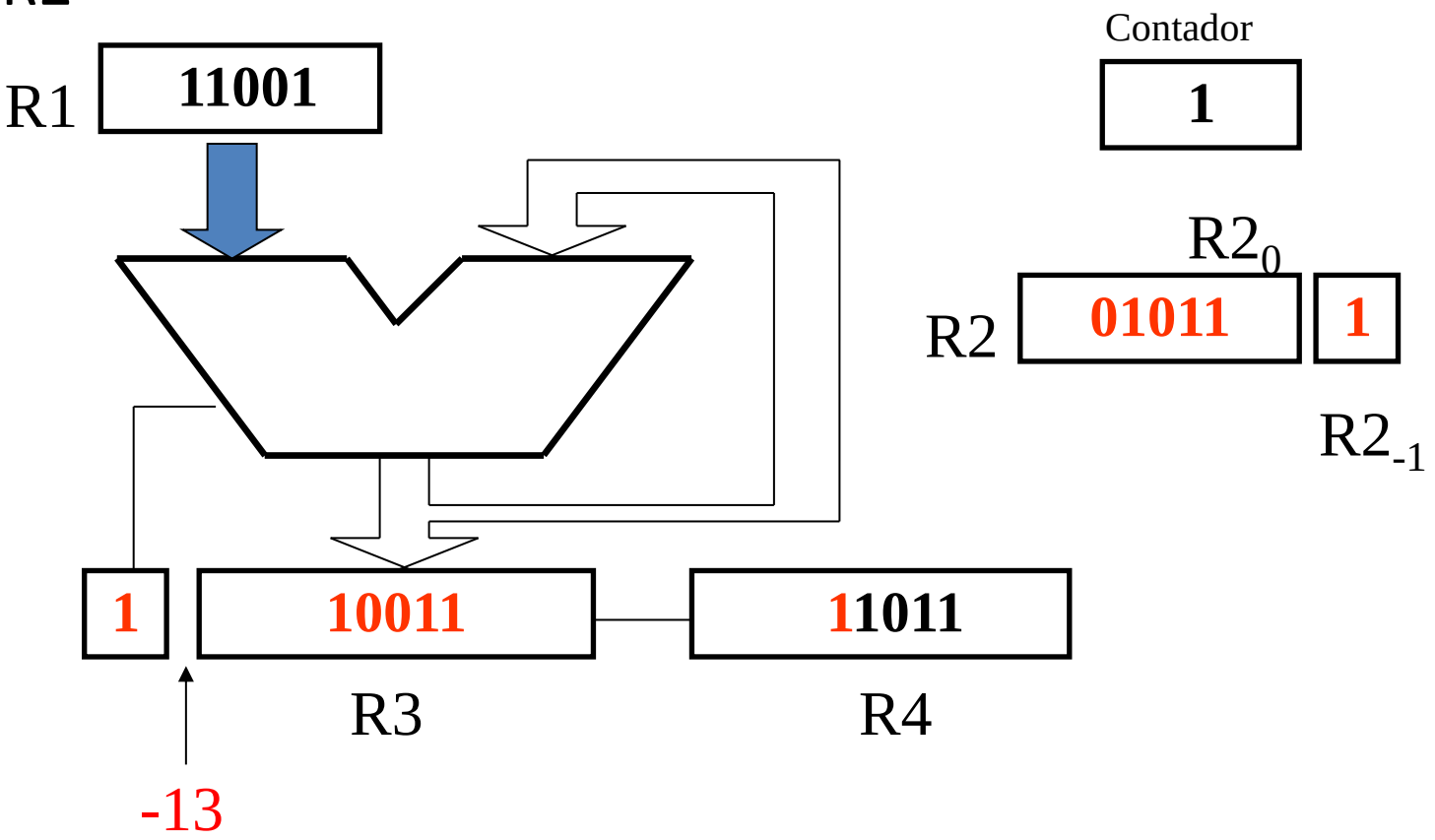
Se incrementa el contador



Desplazamiento a derecha

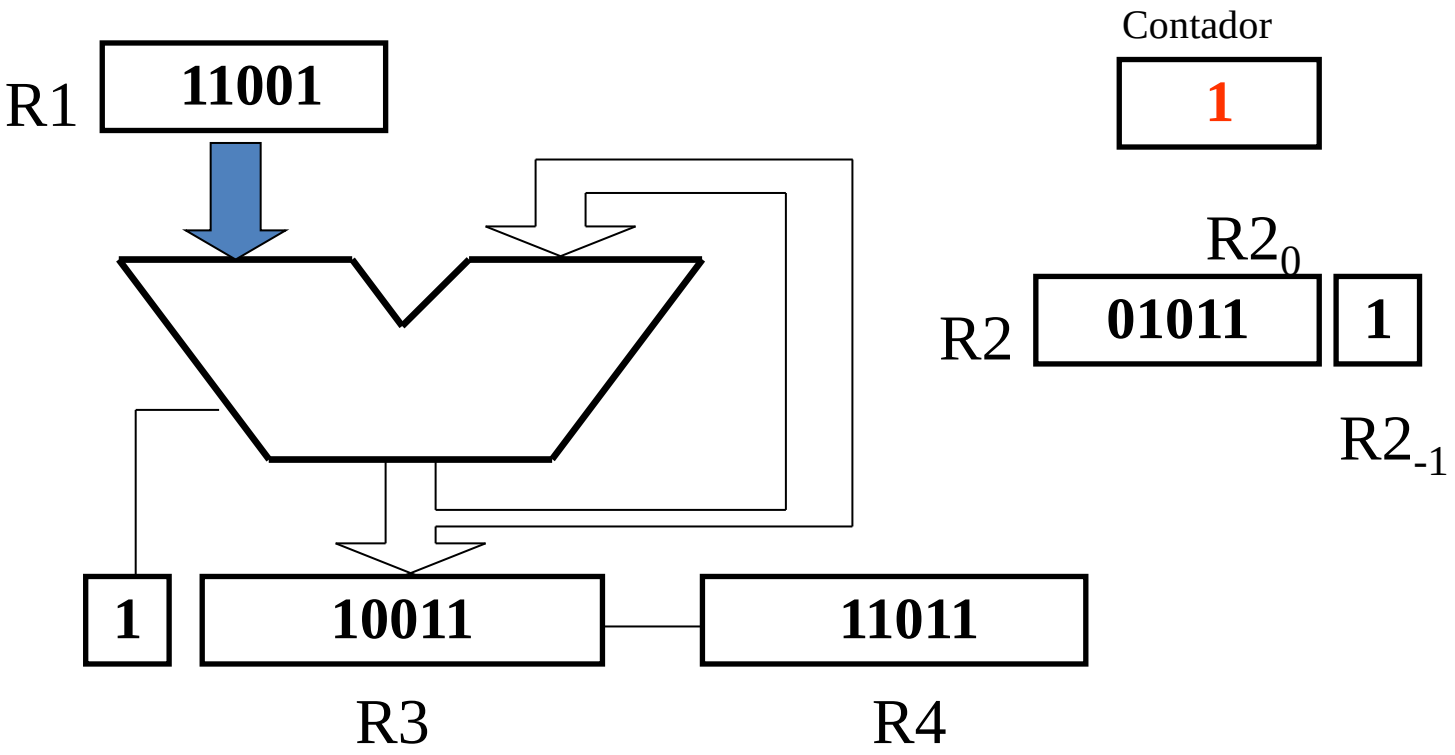
R3 y R4 se desplazan a la derecha, perdiéndose el bit de la derecha de R4

Rota R2



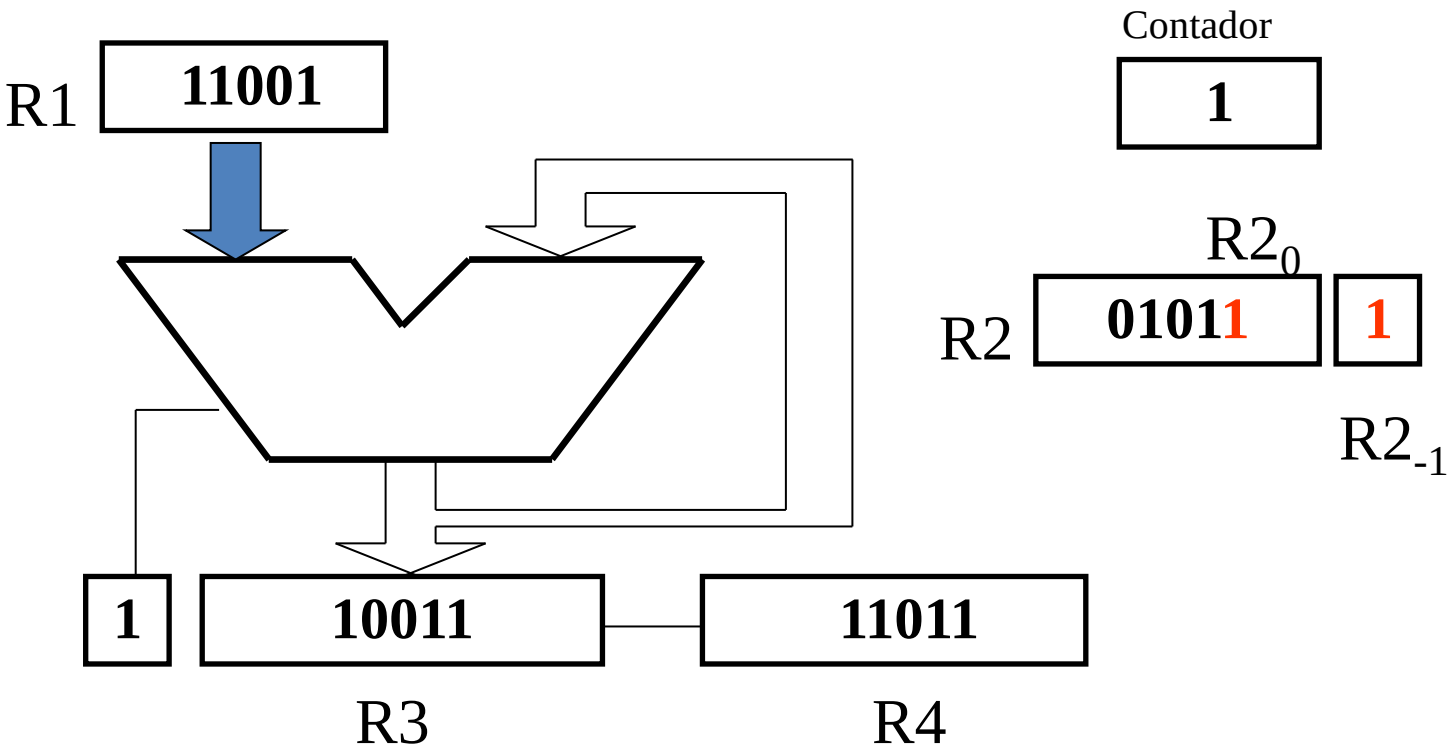
¿Ha llegado al final?

Comprueba que el contador no es 5 y repite el bucle



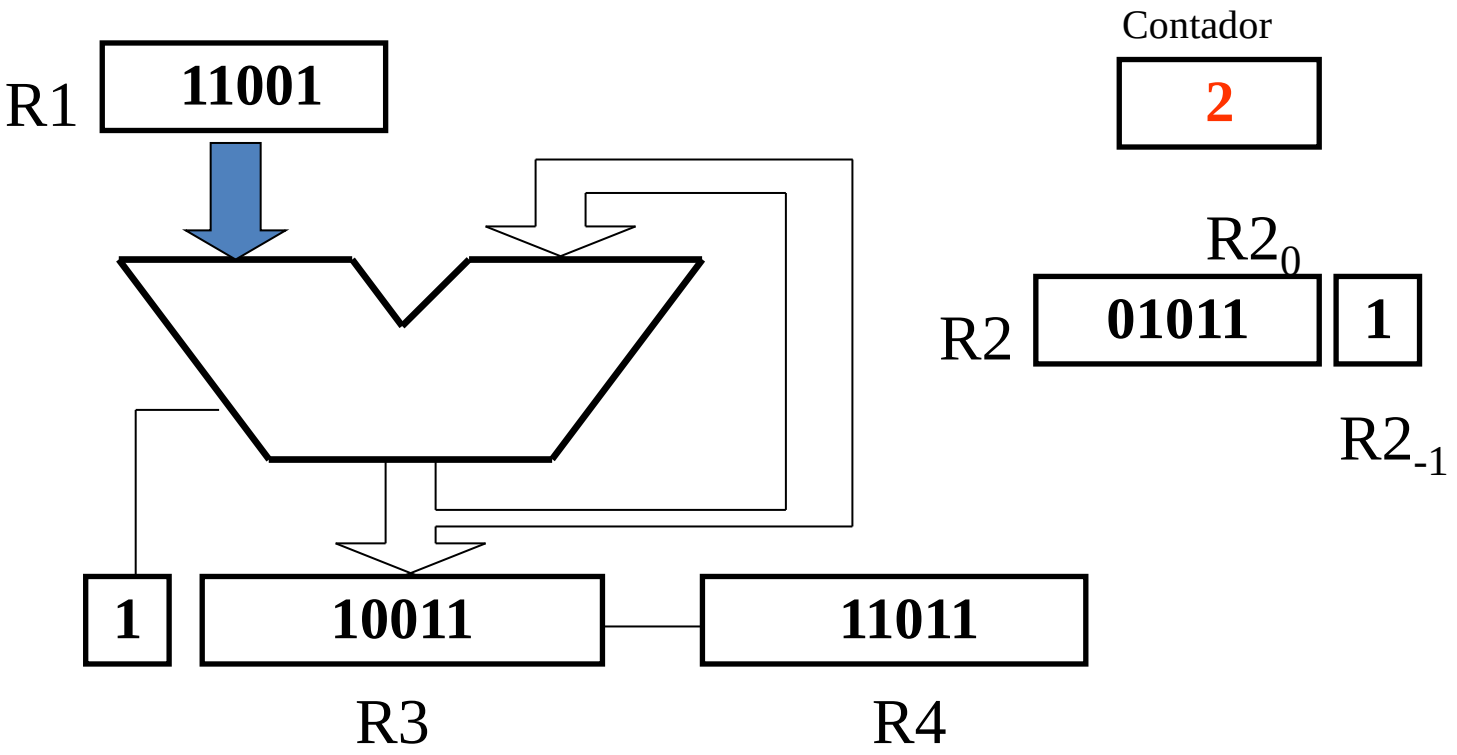
Comprobación multiplicador y decisión

No hay cambio de cadena. No se hace nada, salvo comprobar que $R2_0$ y $R2_{-1}$ son iguales.



Incremento contador

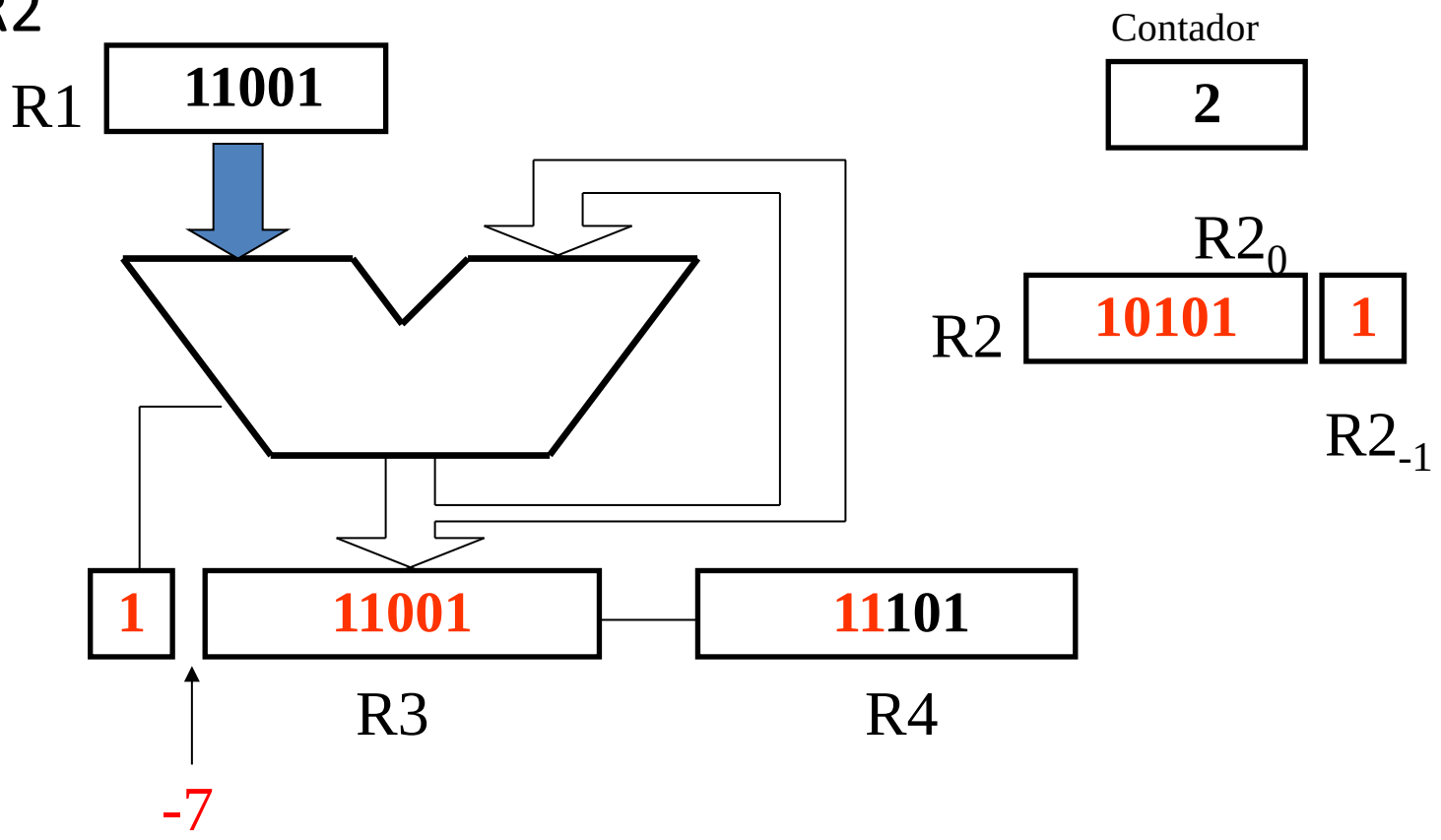
Se incrementa el contador



Desplazamiento a derecha

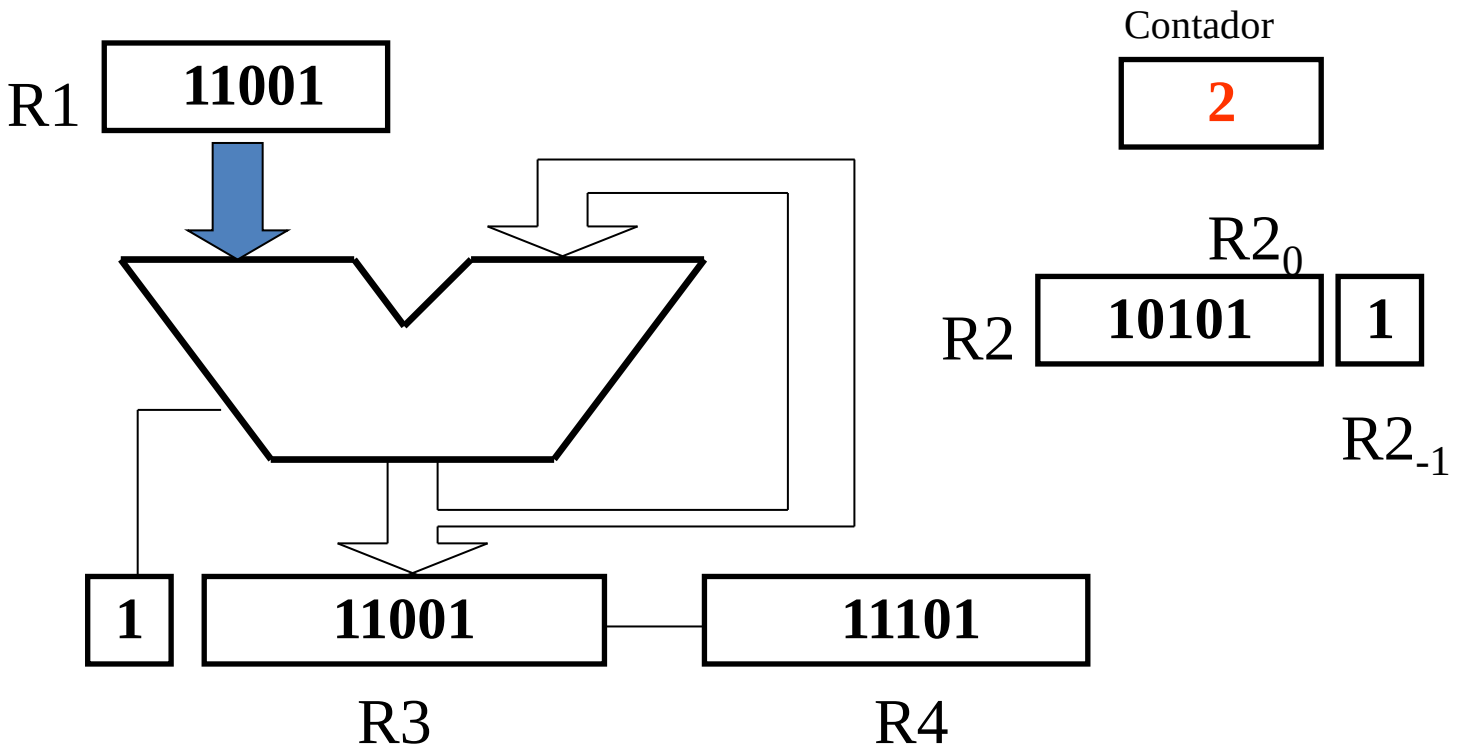
R3 y R4 se desplazan a la derecha, perdiéndose el bit de la derecha de R4

Rota R2



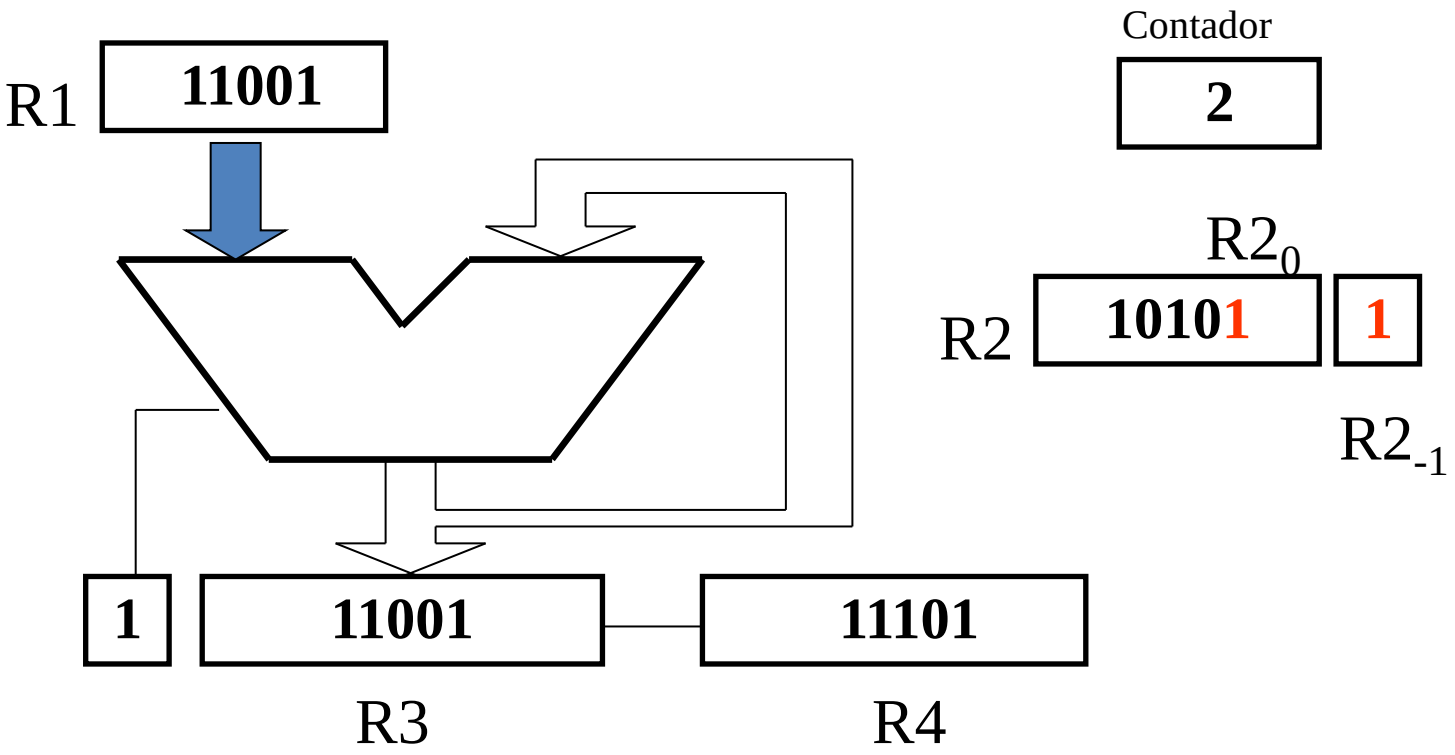
¿Ha llegado al final?

Comprueba que el contador no es 5 y repite el bucle



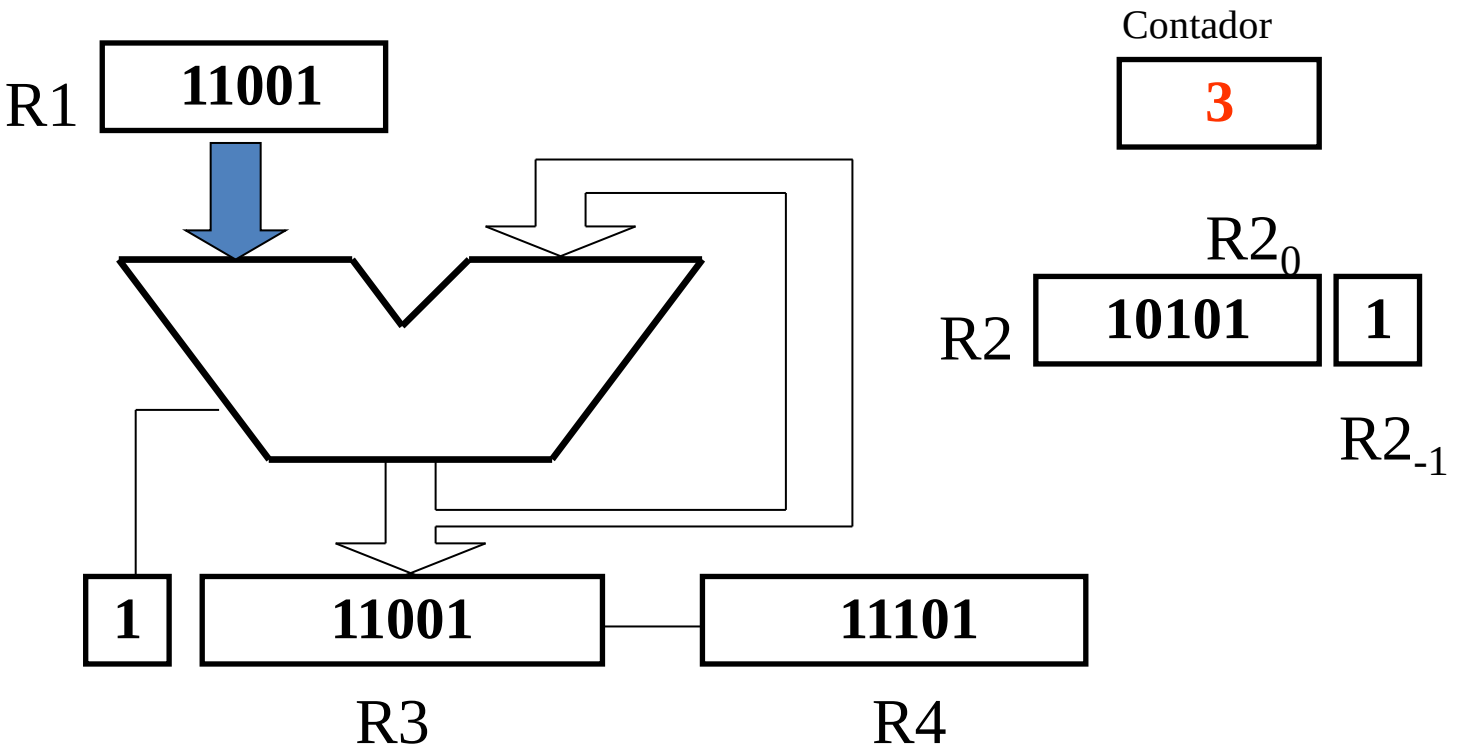
Comprobación multiplicador y decisión

No hay cambio de cadena. No se hace nada, salvo comprobar que $R2_0$ y $R2_{-1}$ son iguales.



Incremento contador

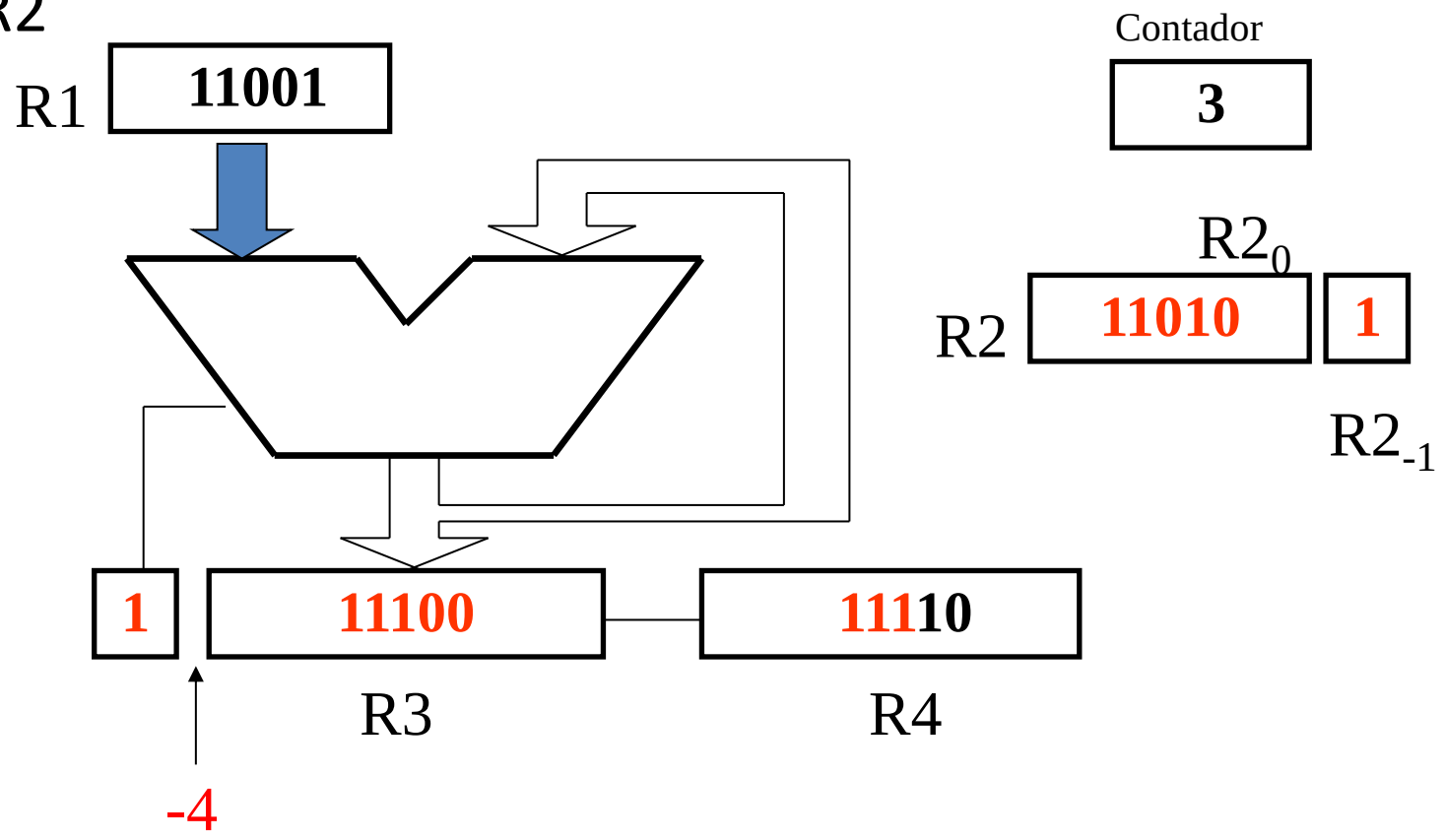
Se incrementa el contador



Desplazamiento a derecha

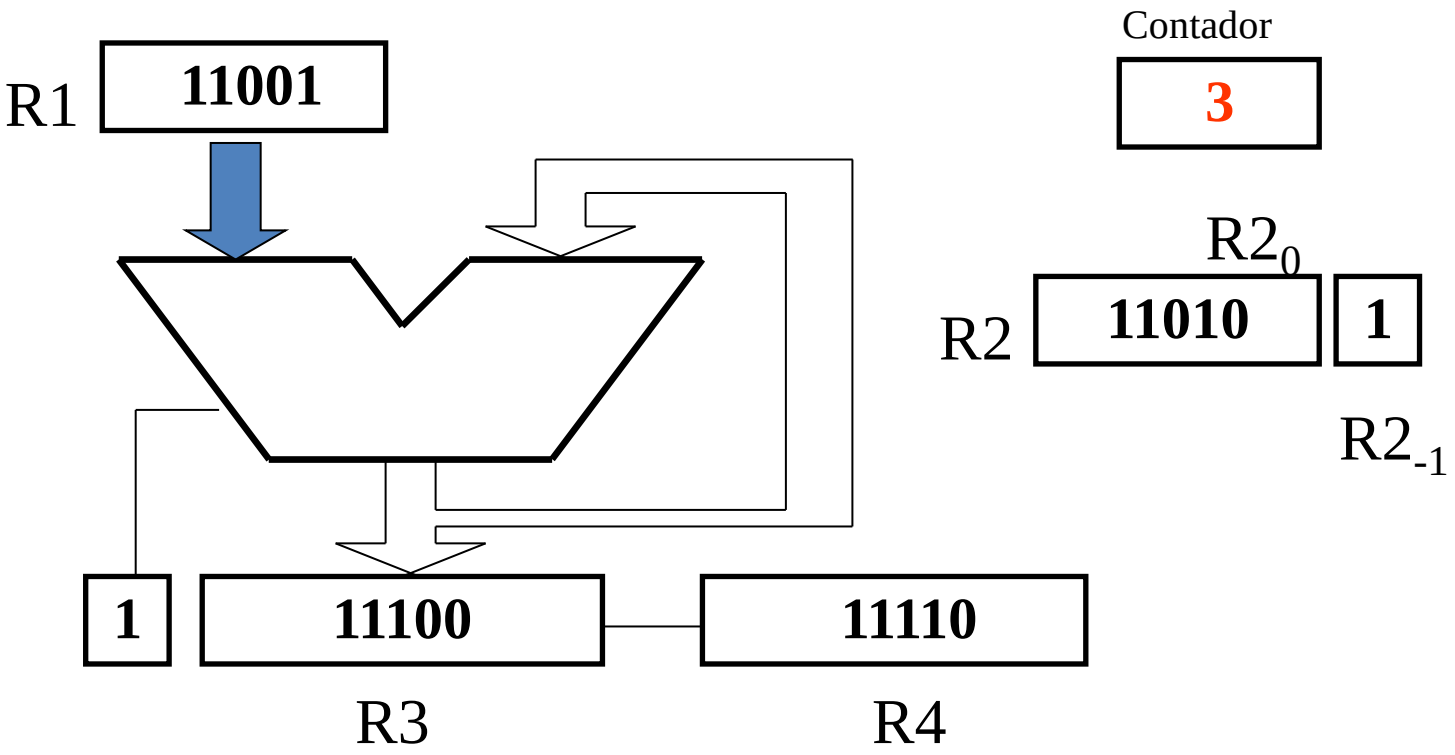
R3 y R4 se desplazan a la derecha, perdiéndose el bit de la derecha de R4

Rota R2



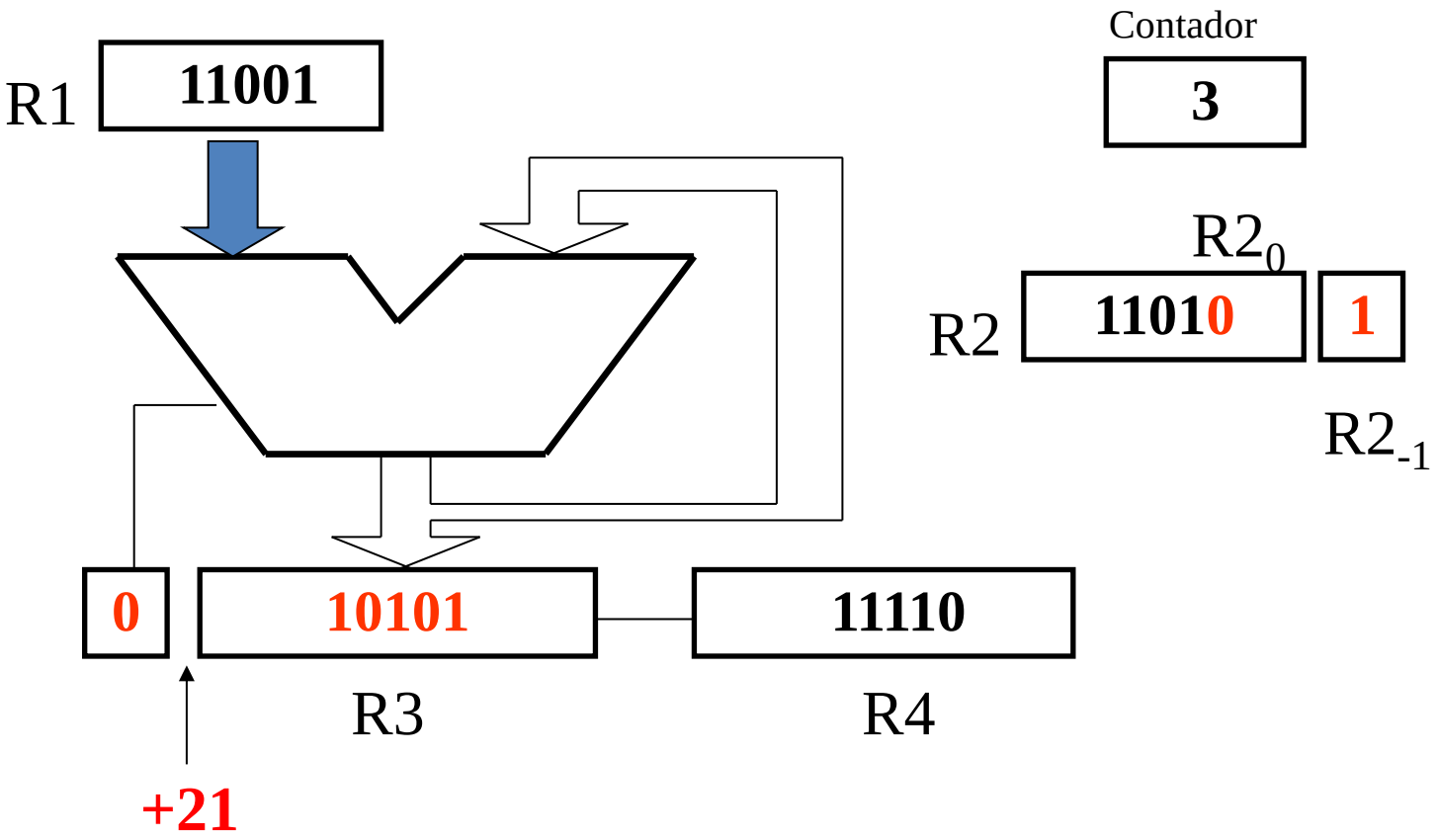
¿Ha llegado al final?

Comprueba que el contador no es 5 y repite el bucle



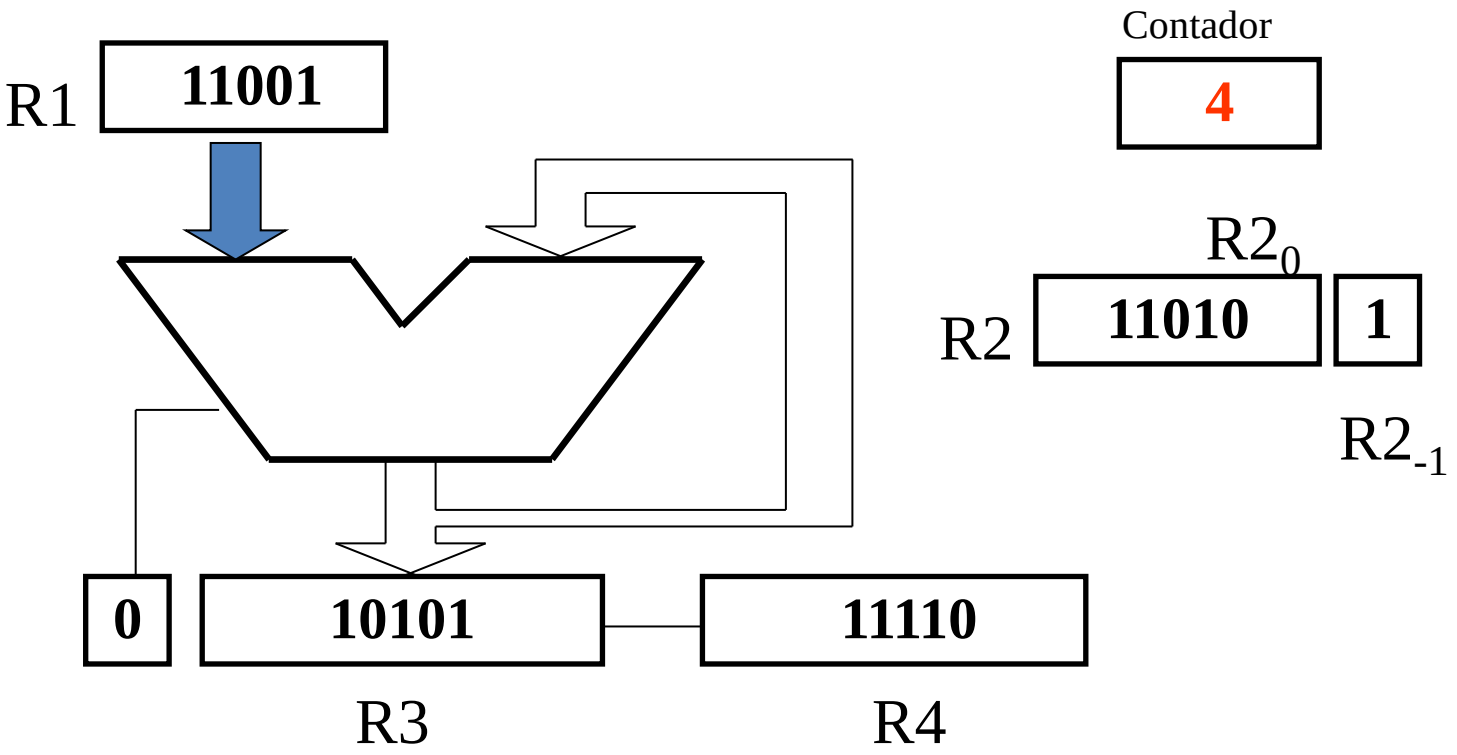
Comprobación multiplicador y decisión

Comienza cadenas de ceros, R3←R3+R1, es decir, se hace 1 11100 + 0 11001 = 0 10101 (-4+25 = +21)



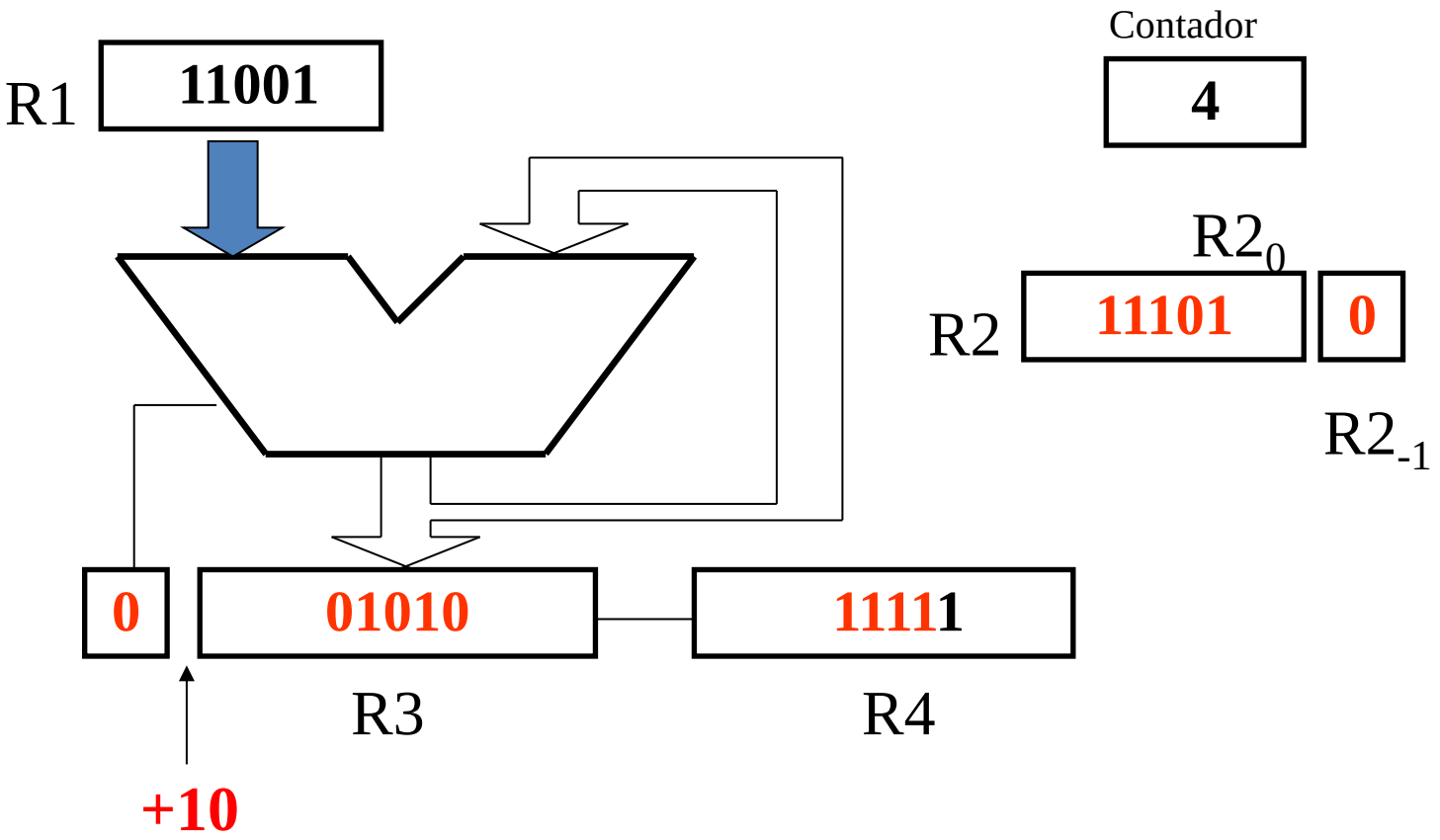
Incremento contador

Se incrementa el contador



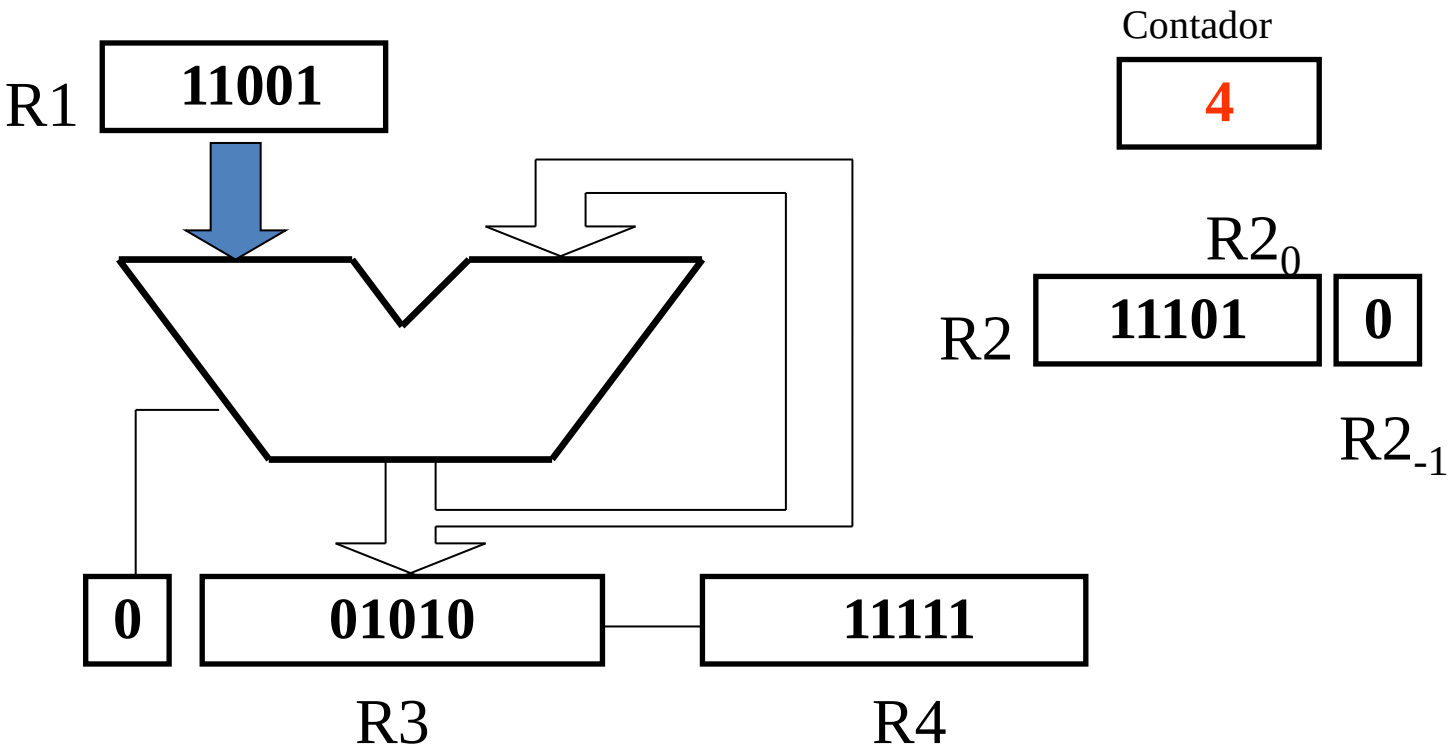
Desplazamiento a derecha

R3, R4 y $R2_{-1}$ se desplazan a la derecha, perdiéndose el bit de la derecha de R4



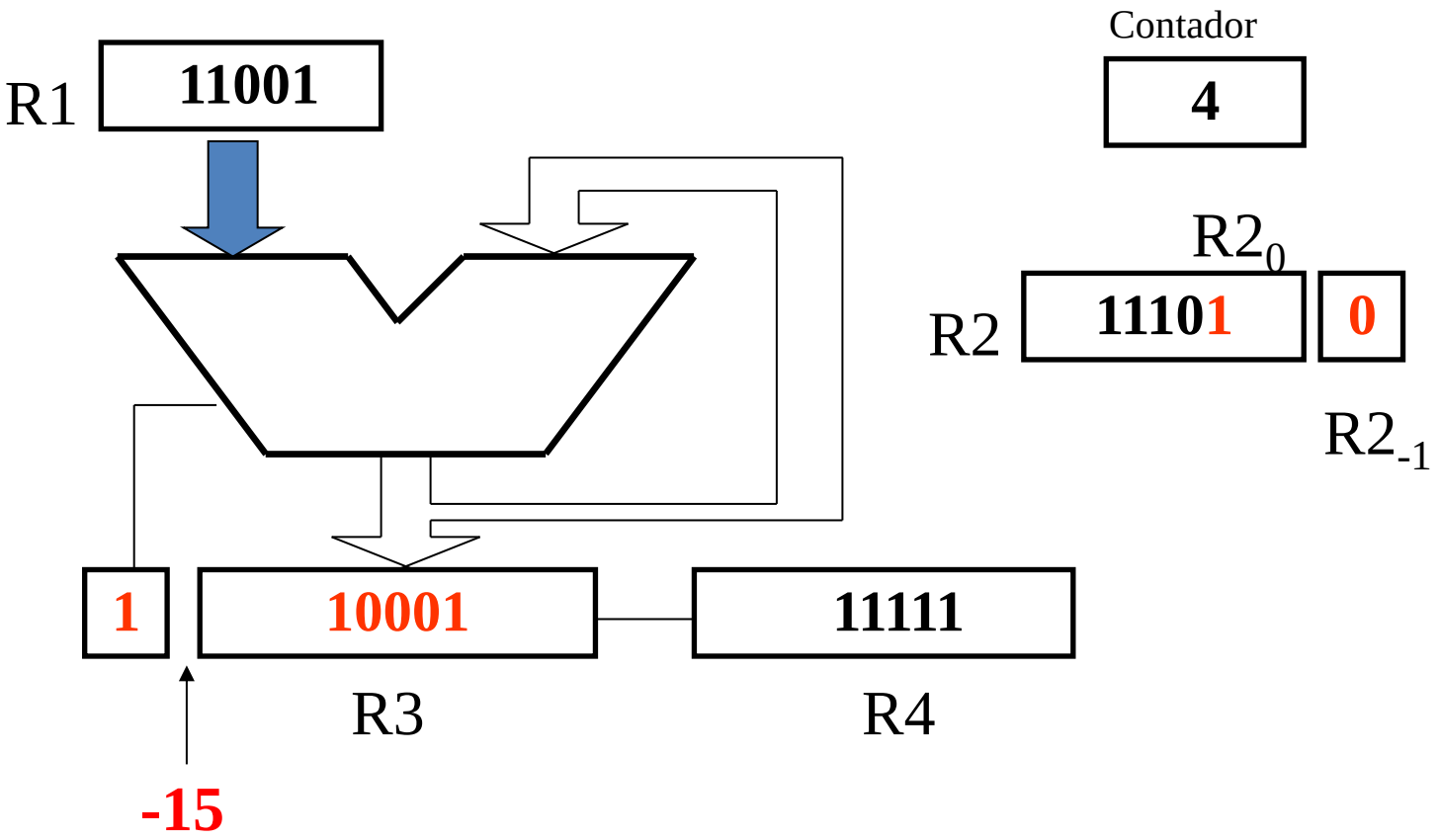
¿Ha llegado al final?

Comprueba que el contador no es 5 y repite el bucle



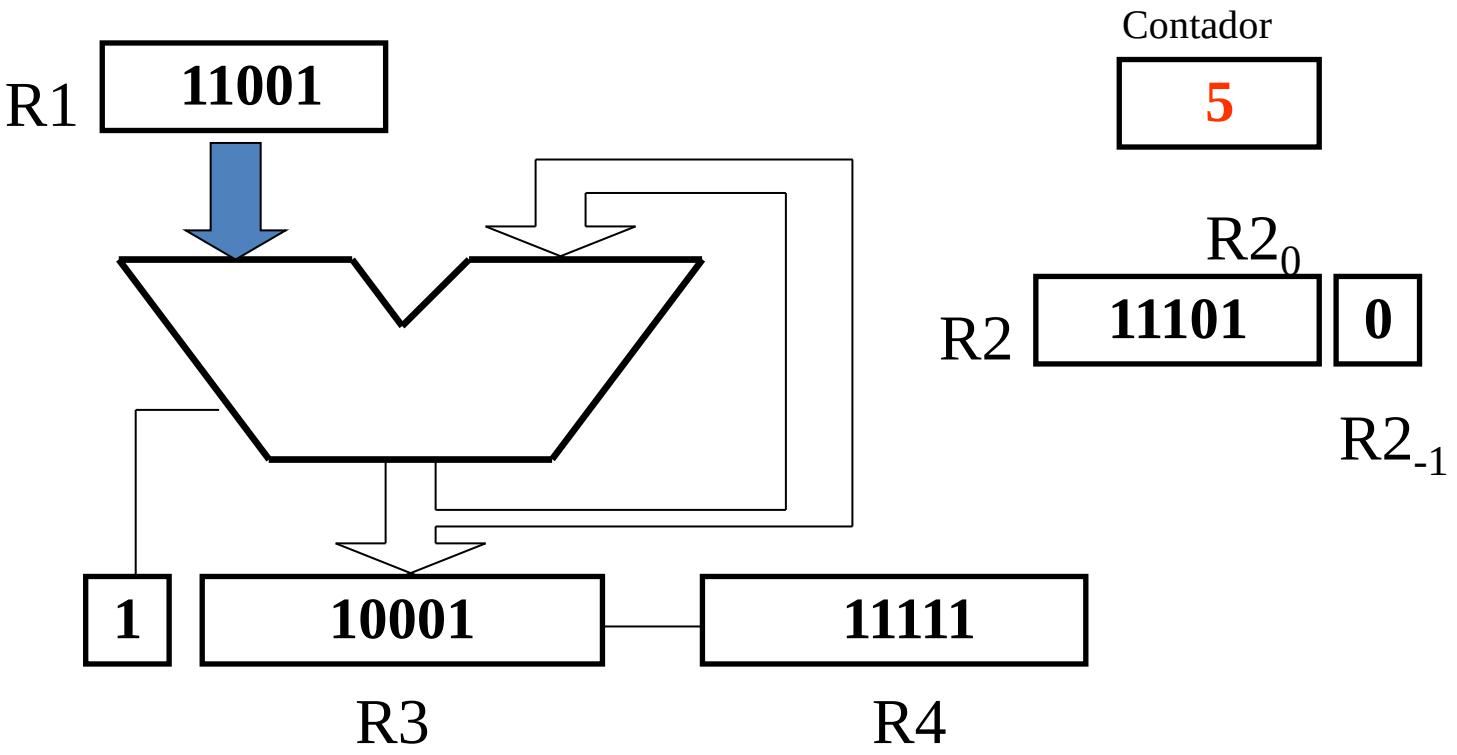
Comprobación multiplicador y decisión

Comienza cadenas de unos, R3←R3-R1, es decir, se hace 0 01010 +1 00111 = 1 10001 (+10-25 = -15)



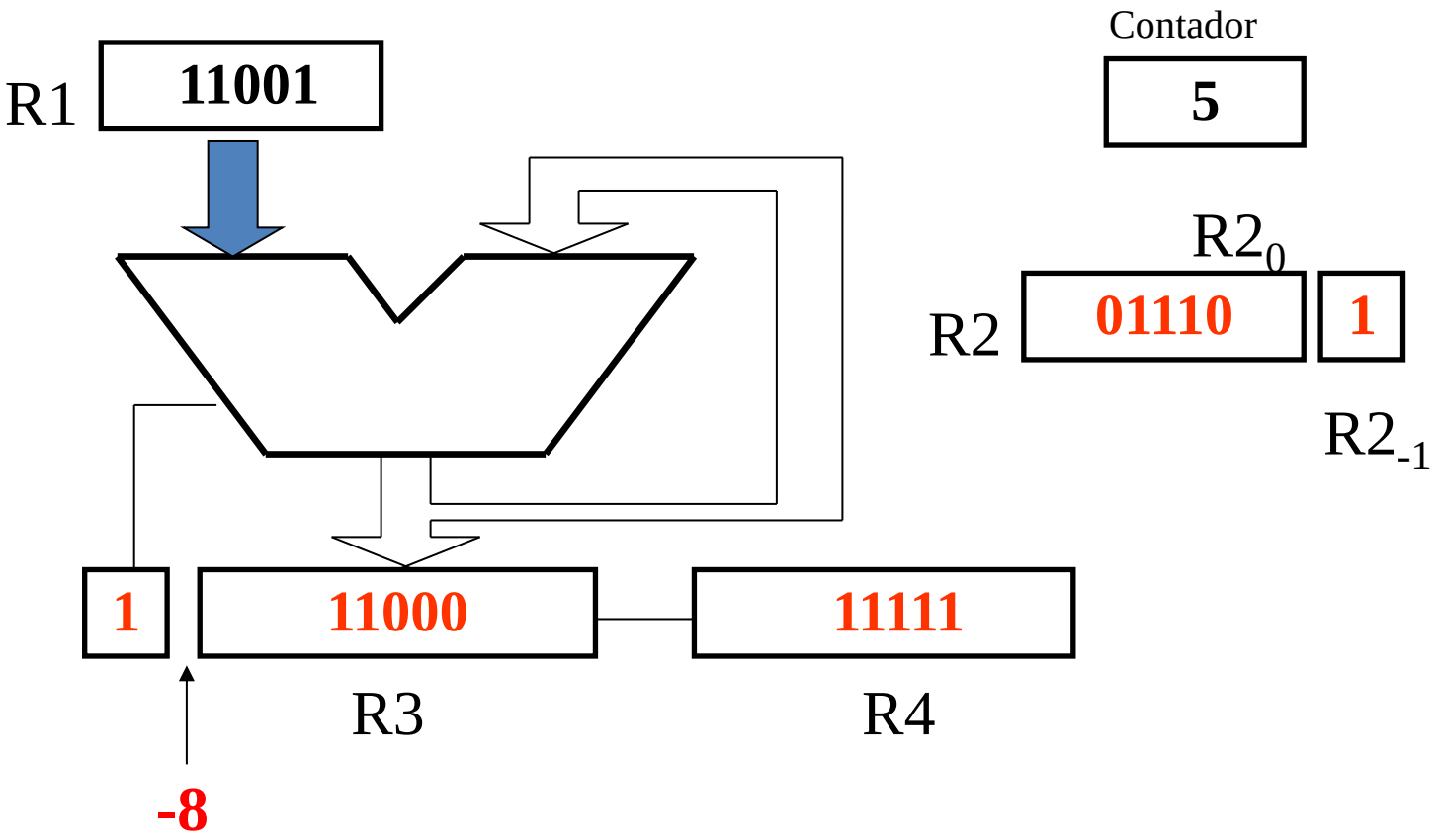
Incremento contador

Se incrementa el contador



Desplazamiento a derecha

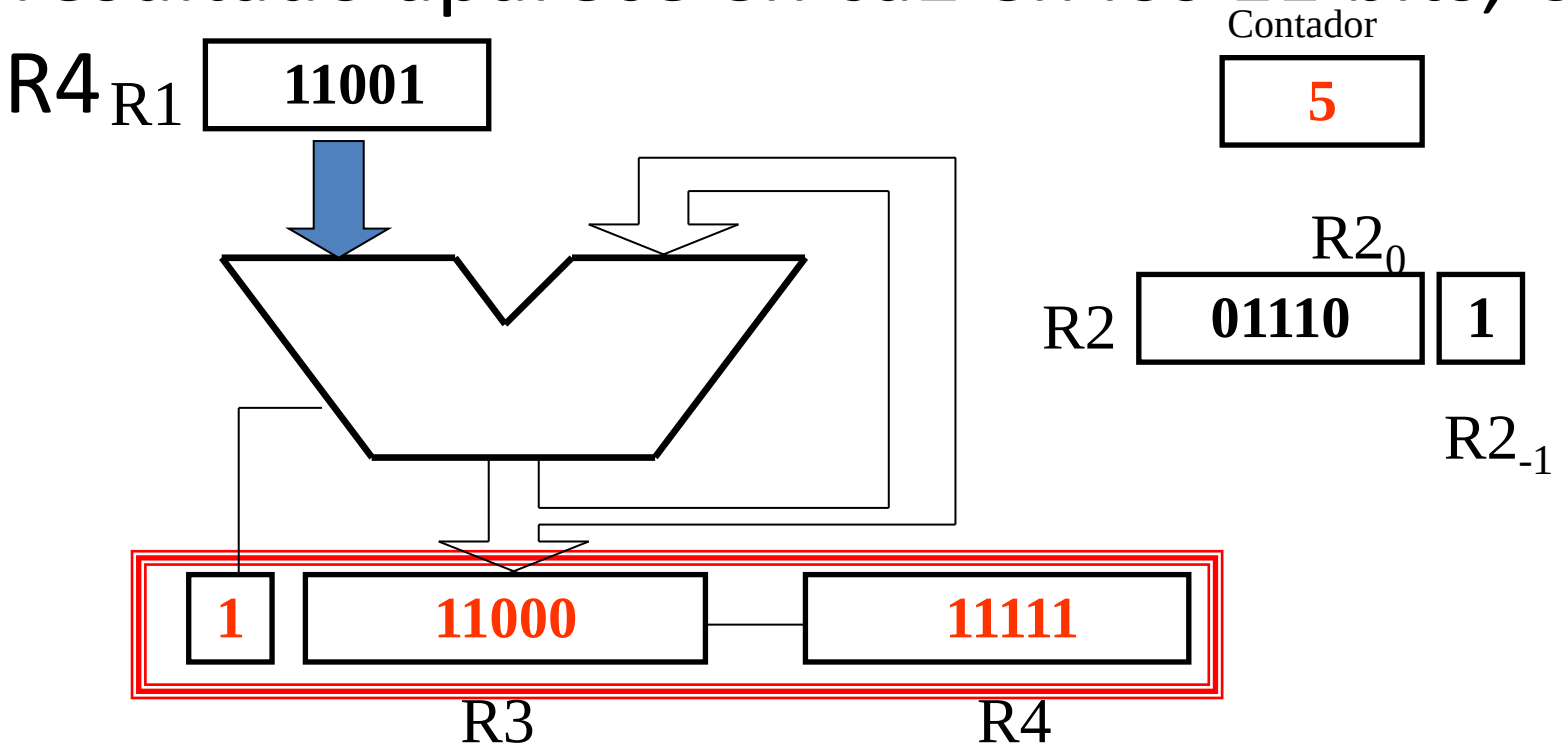
R3, R4 y $R2_{-1}$ se desplazan a la derecha, perdiéndose el bit de la derecha de R4



¿Ha llegado al final?

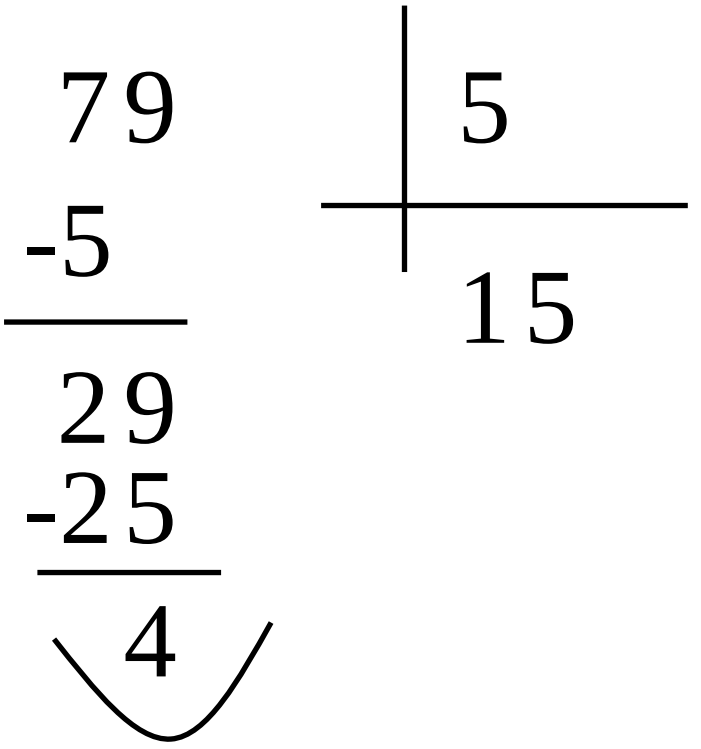
Comprueba que el contador es 5 y acaba

El resultado aparece en ca2 en los 11 bits, C-R3-R4

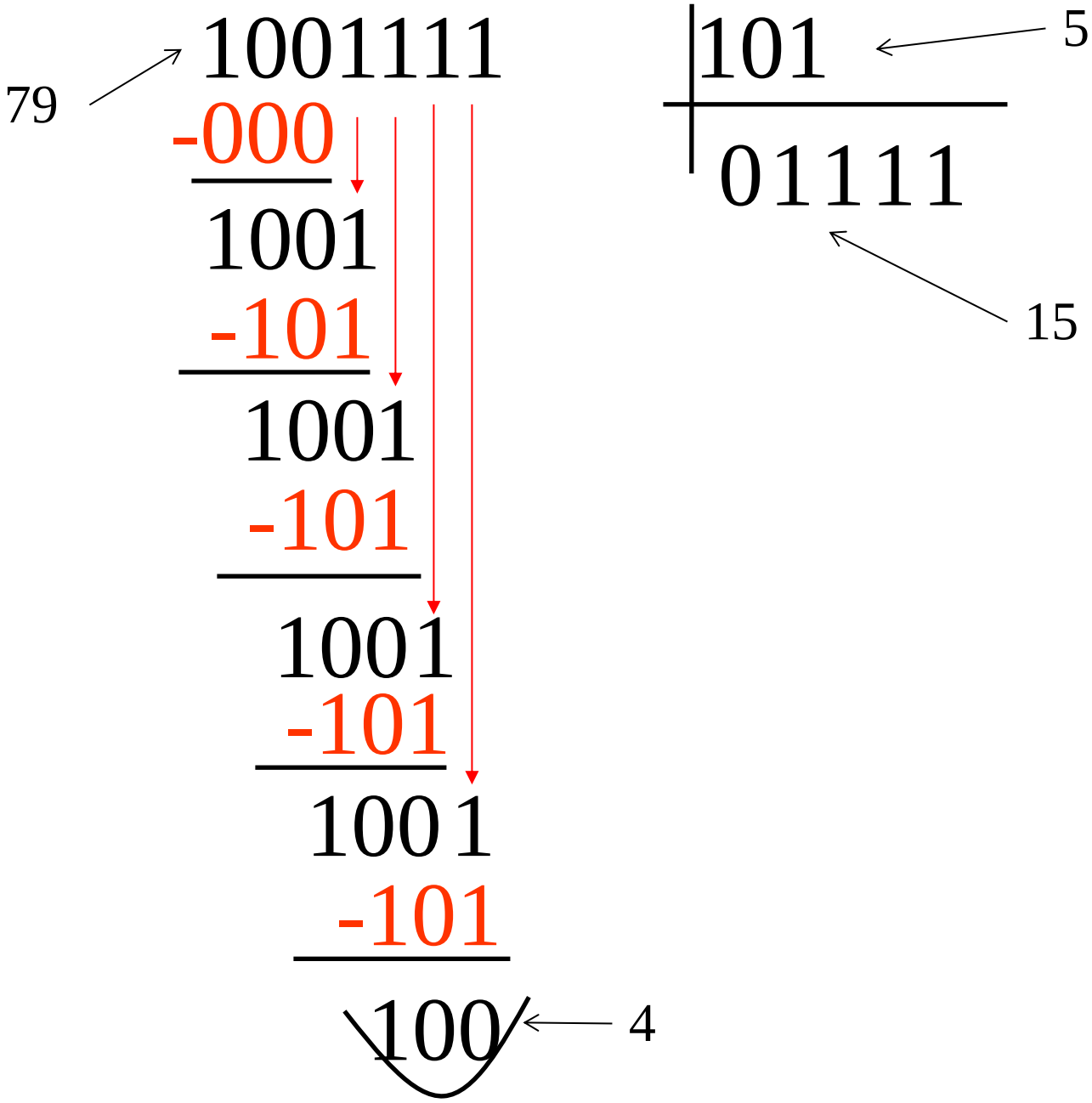


DIVISIÓN BINARIA SIN SIGNO:
CON RESTAURACIÓN

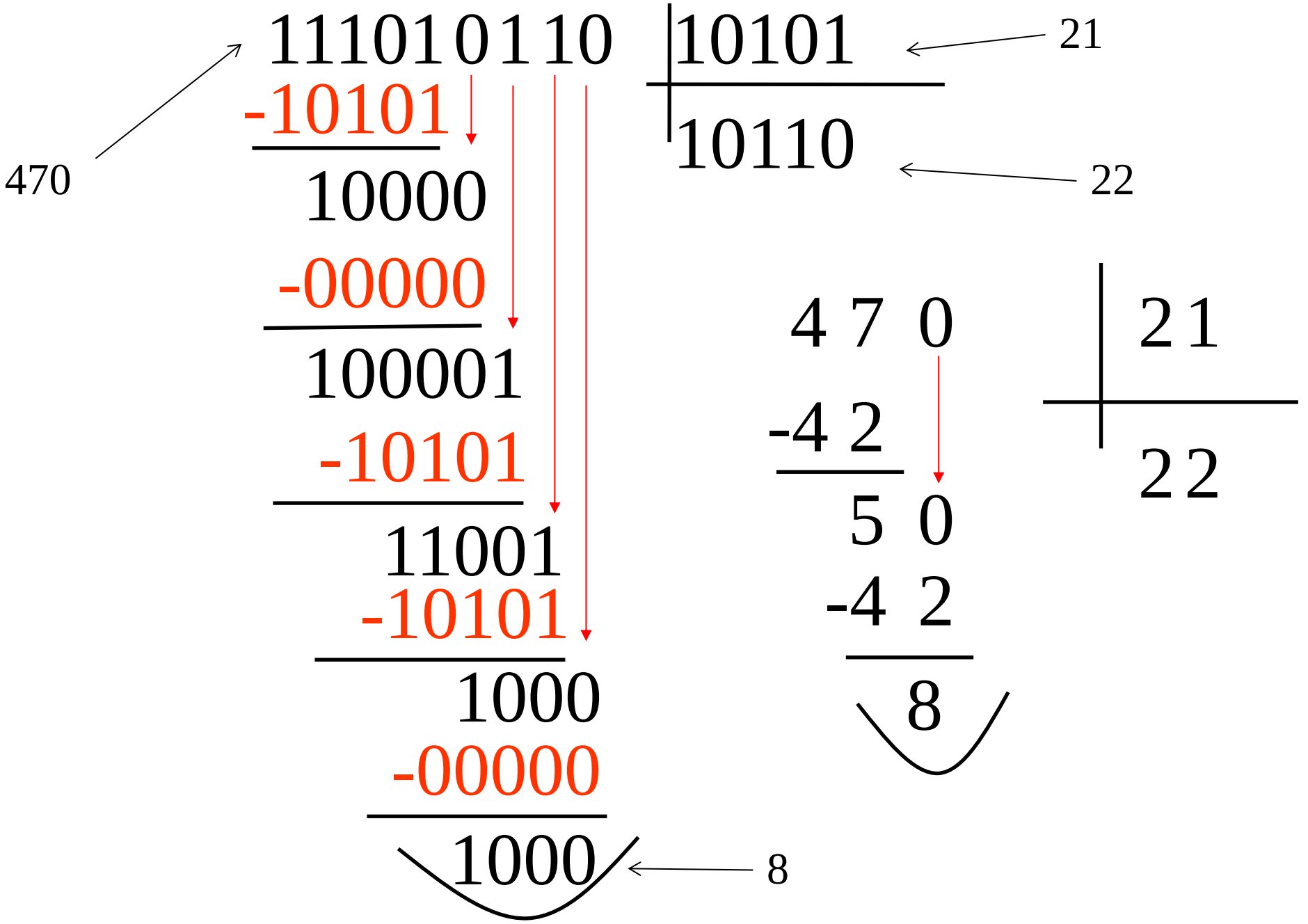
Ejemplo en base 10



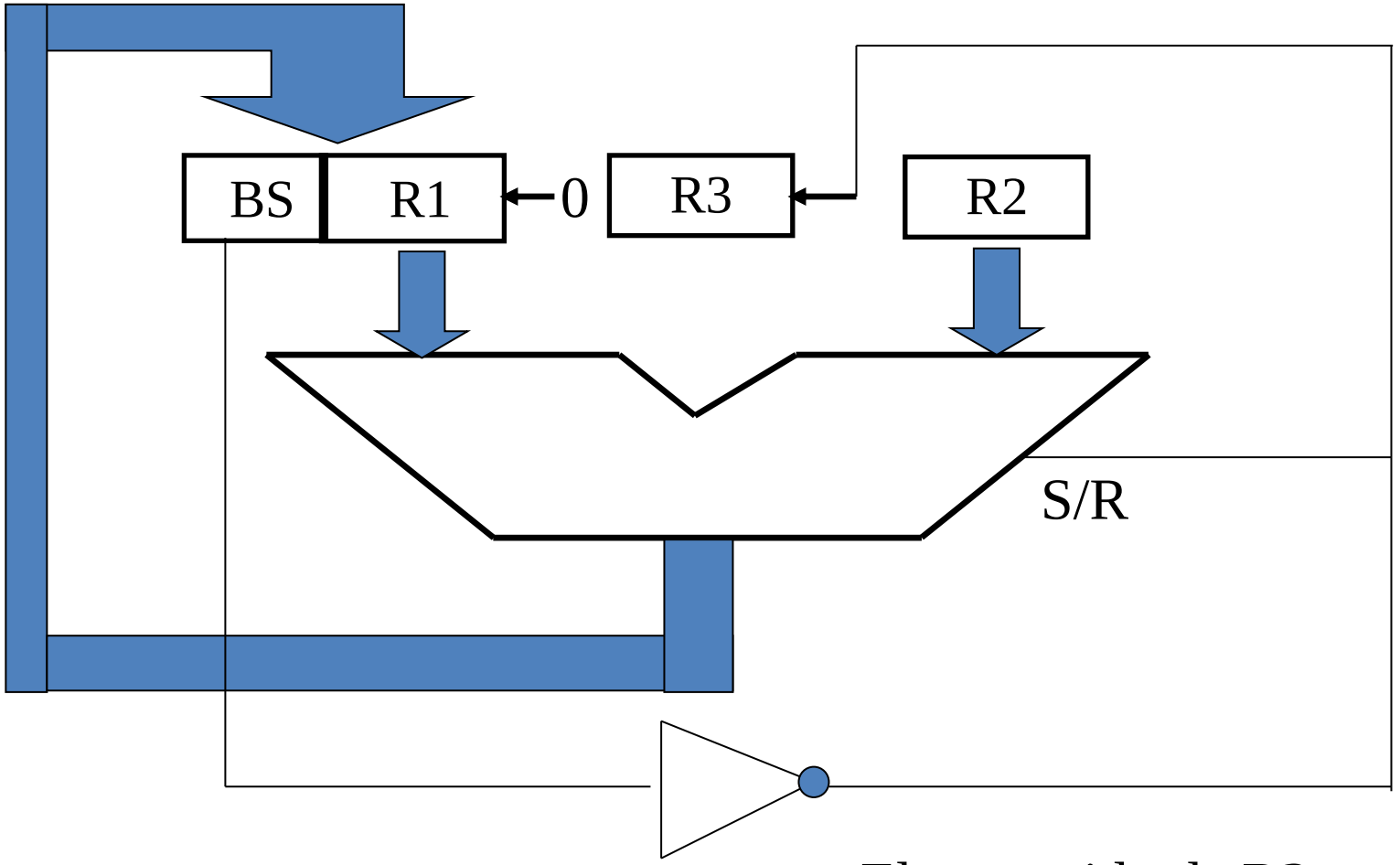
Ejemplo en base 2



Otro ejemplo en ambas bases



Arquitectura



El contenido de R2 no varía

Algoritmo de División con Restauración

Se carga R1 con el dividendo A y R2 con el divisor B. El registro R3 es irrelevante inicialmente y almacenará el cociente

Se hace la resta R1 – R2, para ver si “cabe o no cabe”

Si el resultado es POSITIVO (sí “cabe”), se introduce un 1 en R3 desde la derecha, desplazando los demás bits de R3 a la izquierda

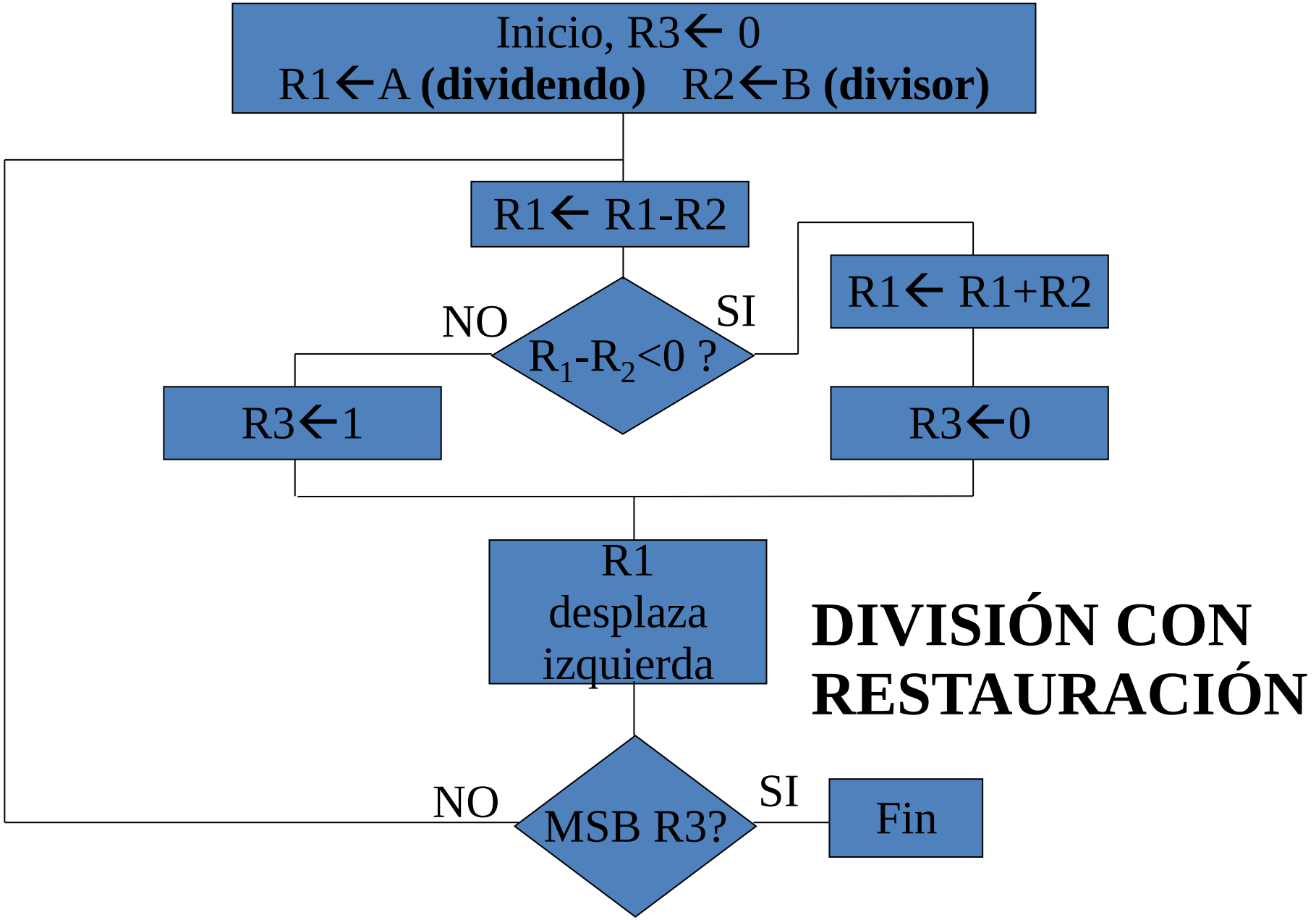
Si el resultado es NEGATIVO (no “cabe”), se hace la suma R1 + R2 (para RESTAURAR el dividendo) y el resultado se deja en R1 y se introduce un 0 en R3 por la derecha, desplazando los demás bits de R3 a la izquierda

Se desplaza R1 a la izquierda rellenando los huecos con CEROS. Es como si nos moviéramos a la derecha en el dividendo

Se repite el bucle hasta que el bit más significativo de R3 entre completamente en el registro R3

El contenido de R2 NUNCA se altera

Finalmente, el cociente queda en R3



DIVISIÓN CON RESTAURACIÓN

Ejemplo

Dividir A=1001111 (79 en decimal) entre B=1010000 (80 en decimal)

Supongamos que trabajamos con registros de 8 bits, en complemento a 2,

	R1	R3	R2
Inicio	01001111	00000000	01010000

Resta

R1- R2

R1	R3	R2
NEGATIVO	00000000	01010000

Suma

R1+R2

R3← 0

R1	R3	R2
01001111	00000000	01010000

Desplazamiento

Izquierda

R1	R3	R2
1001111 0	0000000 **0**	01010000

	R1	R3	R2
Resta	01001110	00000000	01010000
R1- R2	**POSITIVO**		

	R1	R3	R2
Desplazamiento Izquierda R3← 1	1001110 0	0000000**1**	01010000

Resta

R1- R2

R1	R3	R2
01001100	00000001	01010000

POSITIVO

Desplazamiento

Izqda.

R3← 1

R1	R3	R2
1001100 0	0000001 **1**	01010000

	R1	R3	R2
Resta	01001000	00000011	01010000
R1- R2	**POSITIVO**		

	R1	R3	R2
Desplazamiento Izqda.	1001000 0	0000011 **1**	01010000
R3← 1			

	R1	R3	R2
Resta	01000000	00000111	01010000
R1- R2	**POSITIVO**		

	R1	R3	R2
Desplazamiento Izqda.	1000000 0	0000111 **1**	01010000
R3← 1			

	R1	R3	R2
Resta	00110000	00001111	01010000
R1- R2	**POSITIVO**		

	R1	R3	R2
Desplazamiento Izqda.	0110000 0	0001111 **1**	01010000
R3← 1			

	R1	R3	R2
Resta	00010000	00011111	01010000
R1- R2	**POSITIVO**		

	R1	R3	R2
Desplazamiento Izqda.	0010000 0	0011111 **1**	01010000
R3← 1			

Resta

R1- R2

R1	R3	R2
NEGATIVO	00111111	01010000

Suma

R1+R2

R1	R3	R2
00100000	00111111	01010000

Desplazamiento

Izquierda

R3← 0

R1	R3	R2
01000000	**01111110**	01010000

Cifras decimales (I)

Según el ejemplo visto:

$$\underline{01}001111$$
$$1010000$$

es igual a 01111110

¿Qué significa esto?

Cifras decimales (II)

Como el numerador es menor que el denominador, el resultado **DEBE SER MENOR QUE 1**. Por tanto, el valor de la división es:

$$\frac{1001111}{1010000}$$

es igual a 0,1111110

En base 10 sería:

$2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} = 0{,}984375$ (exacto: 79/80=0,9875)

Cifras decimales (III)

BASE 2	BASE 10 (aprox.)	BASE 10 (exacto)
1001111 / 101000 = 1,111110	1,96875	79/40=1,975
1001111 / 10100 = 11,11110	3,9375	79/20=3,95
1001111 / 1010 = 111,1110	7,875	79/10=7,9
1001111 / 101 = 1111,110	15,75	79/5=15,8

Cifras decimales (IV)

Por tanto, para dividir dos números binarios con distintos números de bits, **SE AJUSTA** el menor al mayor, se divide con el algoritmo y finalmente se aplica lo visto en las transparencias anteriores

Ejemplo con obtención del resto **(CON RESTAURACIÓN)**

Dividir A=1101011 (107 en decimal) entre B=101 (5 en decimal), cuyo cociente es 21 y el resto 2

Vamos a trabajar con registros de 8 bits, en complemento a 2.

En este ejemplo, los operandos no tienen el mismo número de bits

	R1	R3	R2
Inicio	0110 1011		0101

	R1	R3	R2
Resta	0001 1011		0101
R1- R2	**POSITIVO**		

	R1	R3	R2
Desplazamiento Izquierda	0011 011	**1**	0101
R3← 1			

Resta

R1- R2

R1	R3	R2
NEGATIVO	1	0101

Suma

R1+R2

R1	R3	R2
0011 011	1	0101

Desplazamiento

Izqda.

R3← 0

R1	R3	R2
0110 11	**10**	0101

Resta

R1- R2

R1	R3	R2
0001 11	10	0101

POSITIVO

Desplazamiento

Izqda.

R3← 1

R1	R3	R2
0011 1	**101**	0101

Resta

R1- R2

R1	R3	R2
NEGATIVO	101	0101

Suma

R1+R2

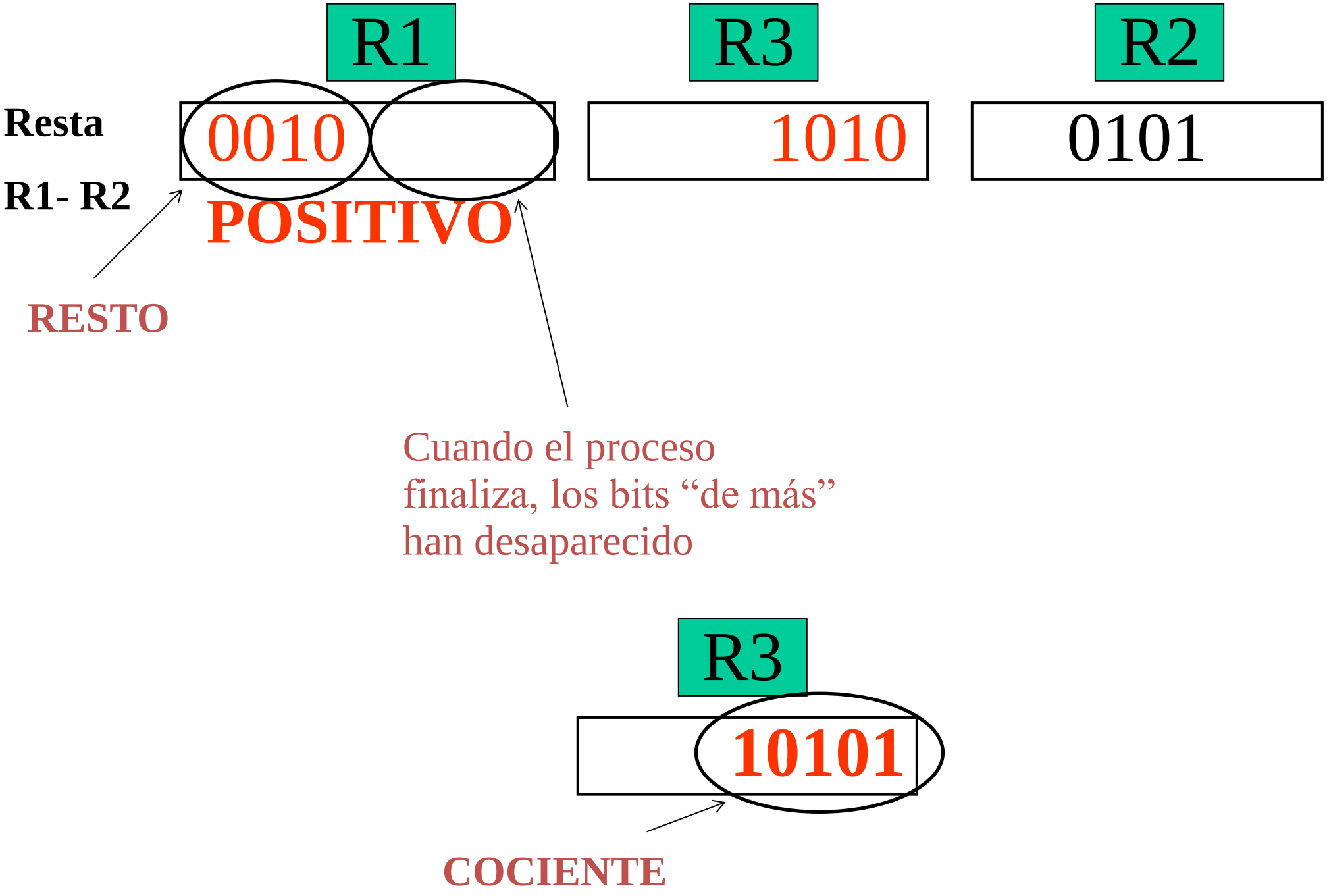
R1	R3	R2
0011 1	101	0101

Desplazamiento

Izqda.

R3← 0

R1	R3	R2
0111	**1010**	0101


R1
R3
R2
Resta
R1- R2
0010
1010
0101
POSITIVO
RESTO
Cuando el proceso
finaliza, los bits “de más”
han desaparecido
R3
10101
COCIENTE

DIVISIÓN BINARIA SIN SIGNO:
SIN RESTAURACIÓN

Fundamentos (I)

En un determinado momento el dividendo residual DR_i está en R1 y B, el divisor, fijo en R2

Se resta R2, obteniéndose $DR_i - B$

Si dicho resultado fue negativo se vuelve a sumar R2, obteniéndose $DR_i - B + B$, para RESTAURAR el dividendo

Se produce un desplazamiento a la izquierda, almacenando en R1 el valor $2 \cdot (DR_i - B + B)$

En el paso siguiente, se vuelve a restar B, quedando

$2 \cdot (DR_i - B + B) - B$, que es igual a **$2 \cdot (DR_i - B) + B$**

Dicha cantidad se podría haber obtenido mucho más rápidamente haciendo la resta de B, el desplazamiento **SIN SUMAR** (sin restaurar) y finalmente la suma de B

Fundamentos (II)

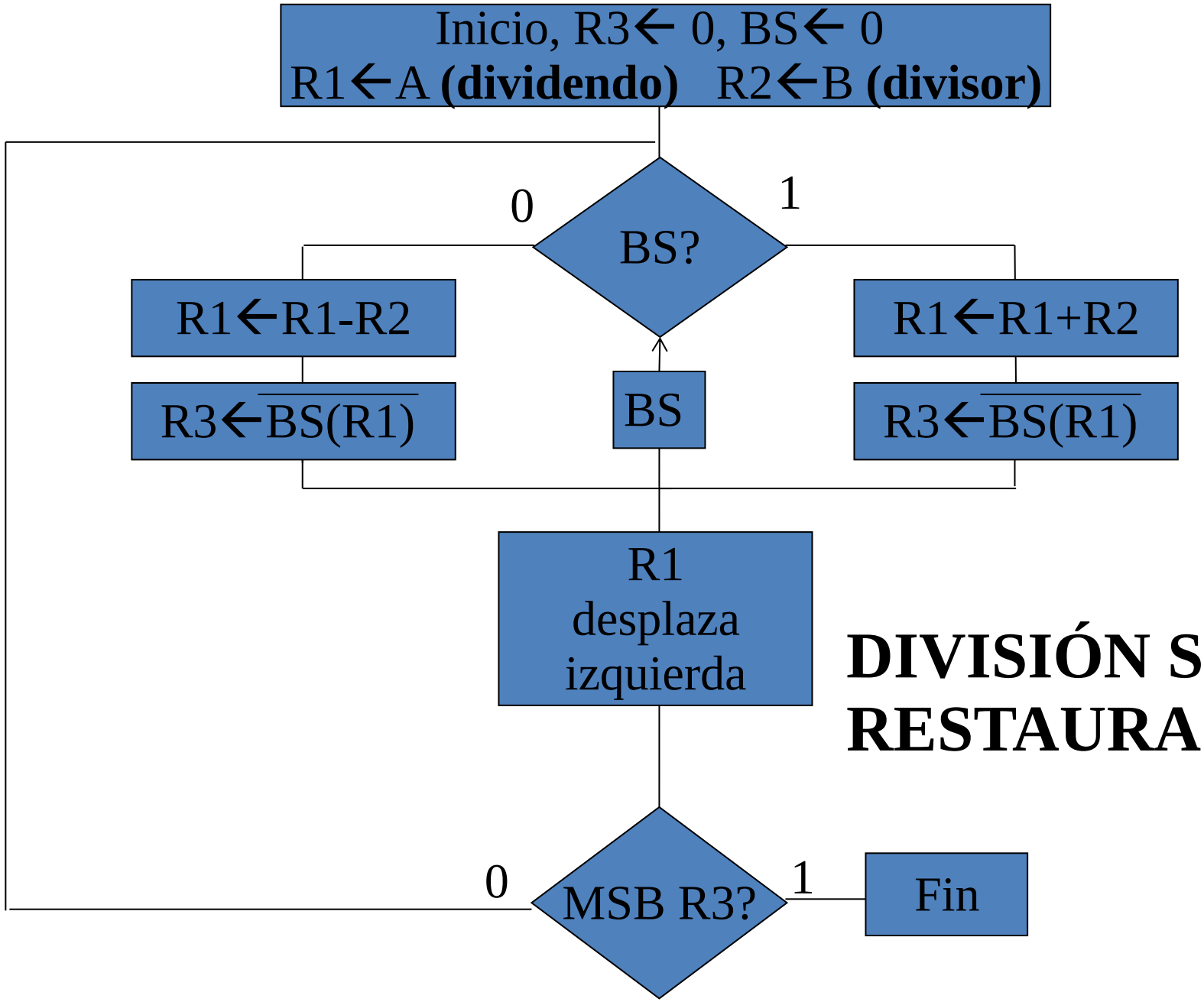
Siempre hay que mirar el bit de signo (BS) de R1. Inicialmente se inicializa a 0

Cuando BS=0 (R1 positivo) se restan R1 y R2 dejando el resultado en R1, se introduce un 1 en R3 por la derecha y se desplaza R1 a la izquierda

Cuando BS=1 (R1 negativo) se suman R1 y R2 dejando el resultado en R1, se introduce un 0 en R3 por la derecha y se desplaza a la izquierda R1

El proceso terminará cuando R3 se “llene” de bits, siendo MSB (R3)=1

NOTA: Las introducciones de bits por la derecha en R3 suponen un desplazamiento hacia la izquierda de dicho registro



DIVISIÓN SIN RESTAURACIÓN

COMPARACIÓN
CON Y SIN RESTAURACIÓN

CON RESTAURACIÓN

Siempre se hace la resta

Se pregunta por el signo del dividendo residual:

Si fue positivo, bien hecha estaba la resta y se introduce un 1 en R3

Si fue negativo, era innecesaria y hay que deshacerla (RESTAURAR) el valor haciendo una suma y se introduce un 0 por R3

SIN RESTAURACIÓN

Se pregunta por el signo del dividendo residual:

Si fue positivo, se hace la resta y se introduce un 1 en R3

Si fue negativo, se hace la suma y se introduce un 0 por R3

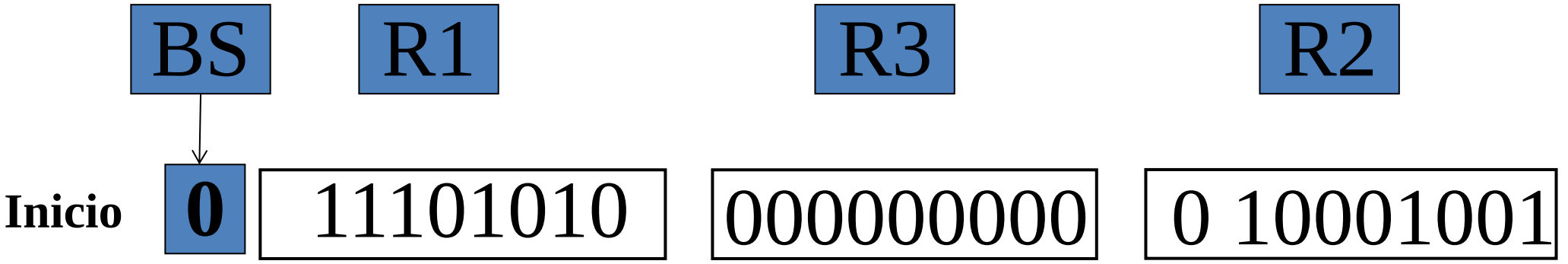
No se hace una operación previa para después deshacerla

Ejemplo

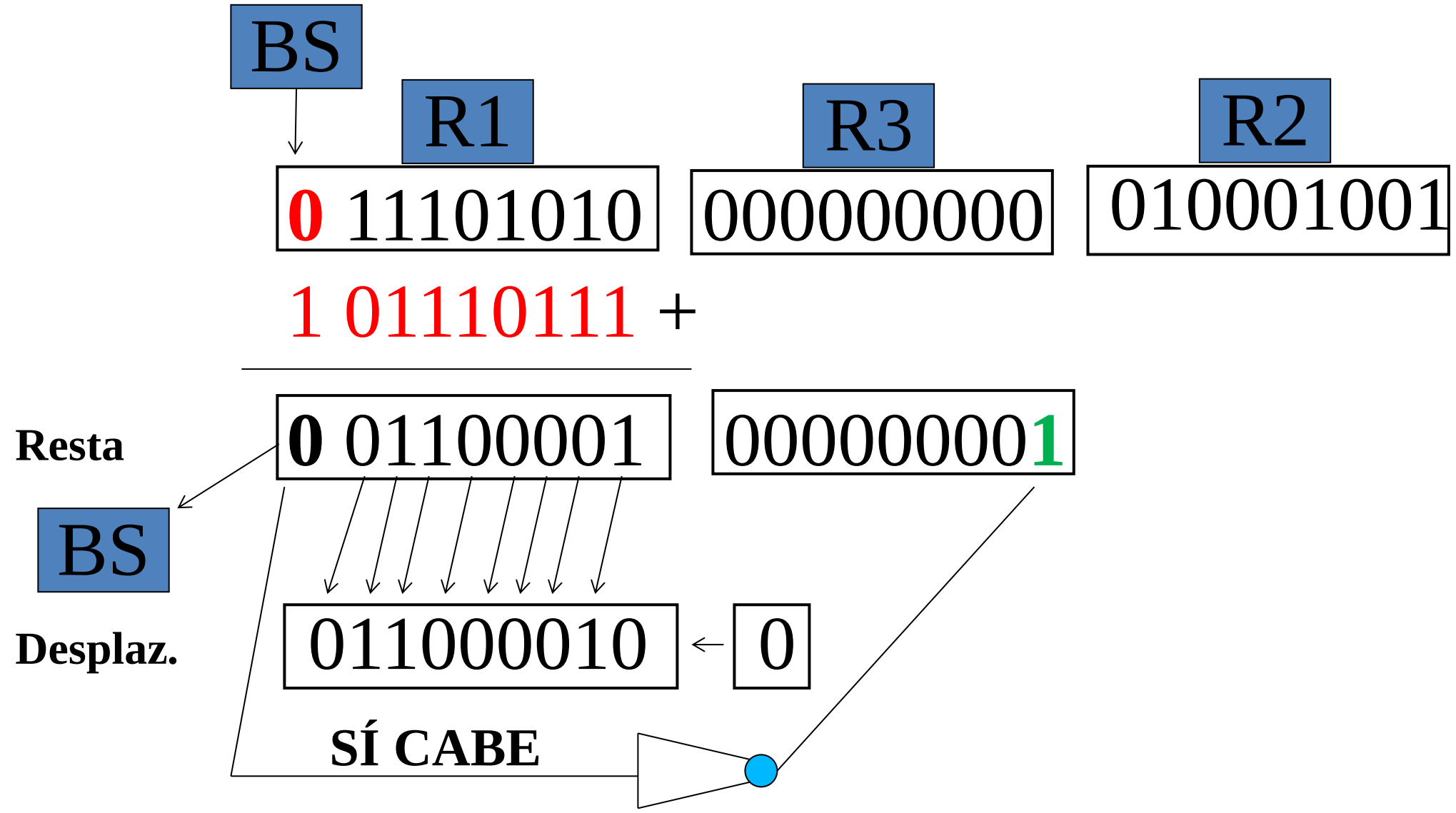
Dividir A=11101010 (234 en decimal) entre B=10001001 (137 en decimal)

Supongamos que trabajamos con registros de 9 bits, en complemento a 2,

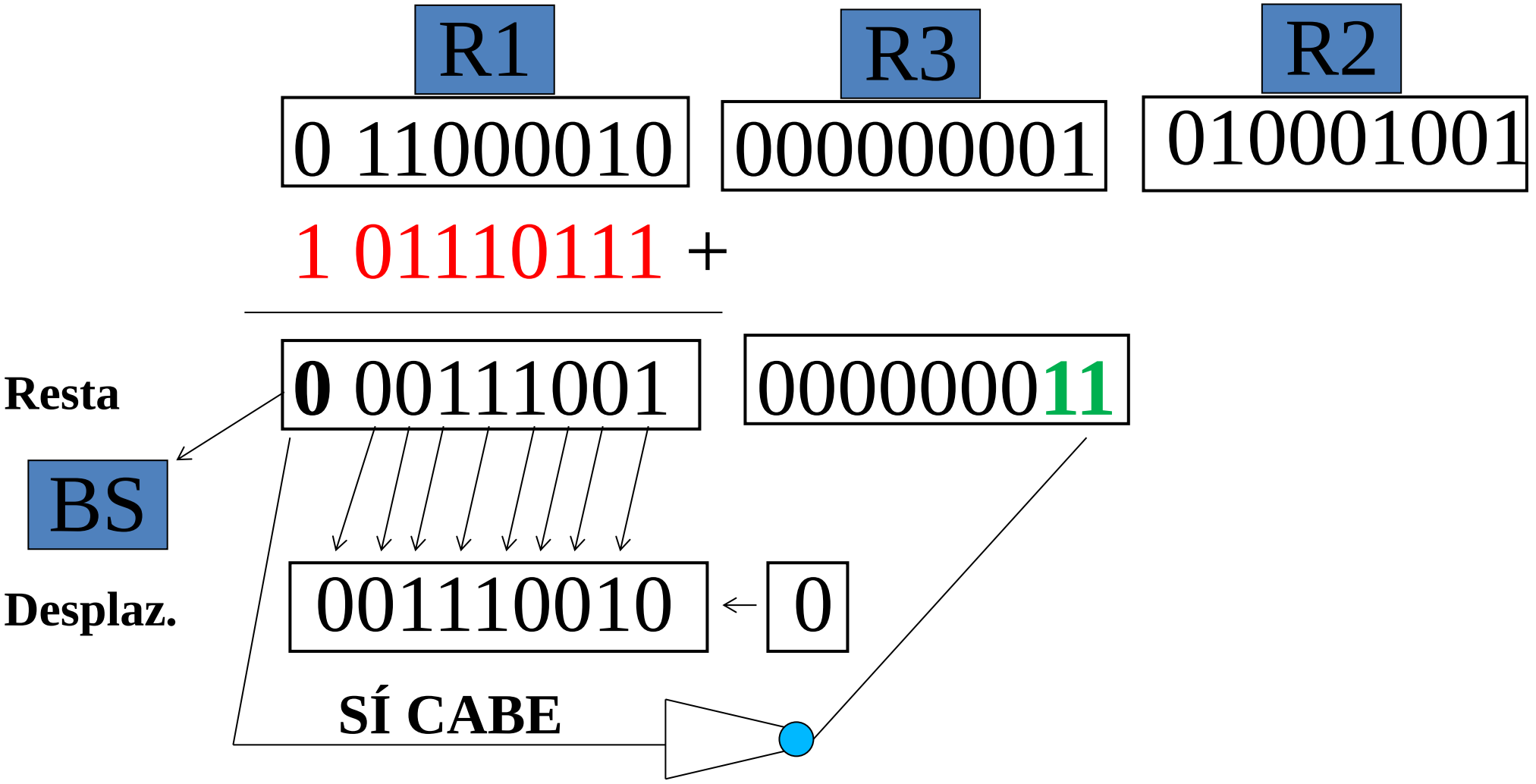
-B=1 01110111 (en complemento a 2)



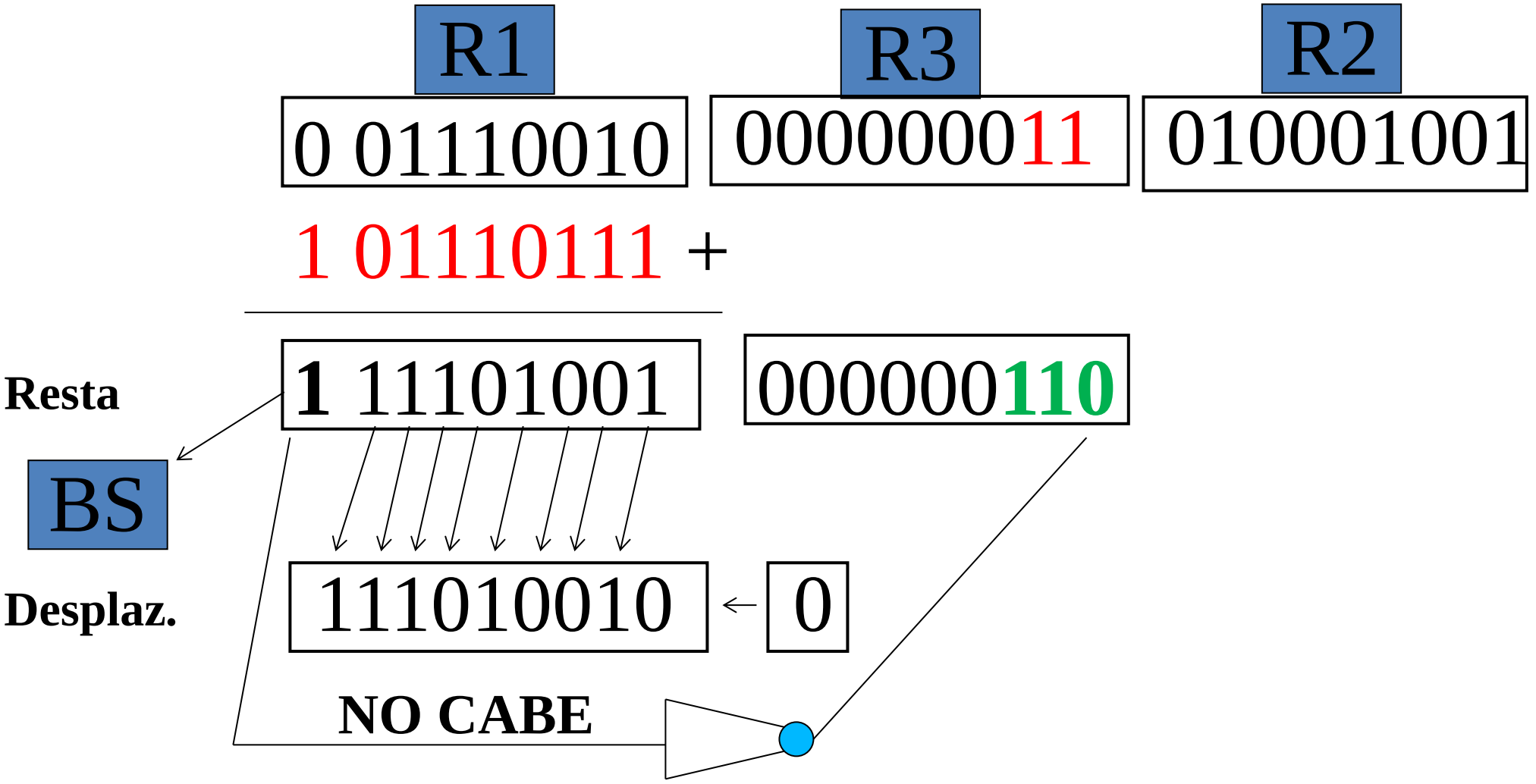
La primera vez BS=0, por tanto, R1←R1-R2



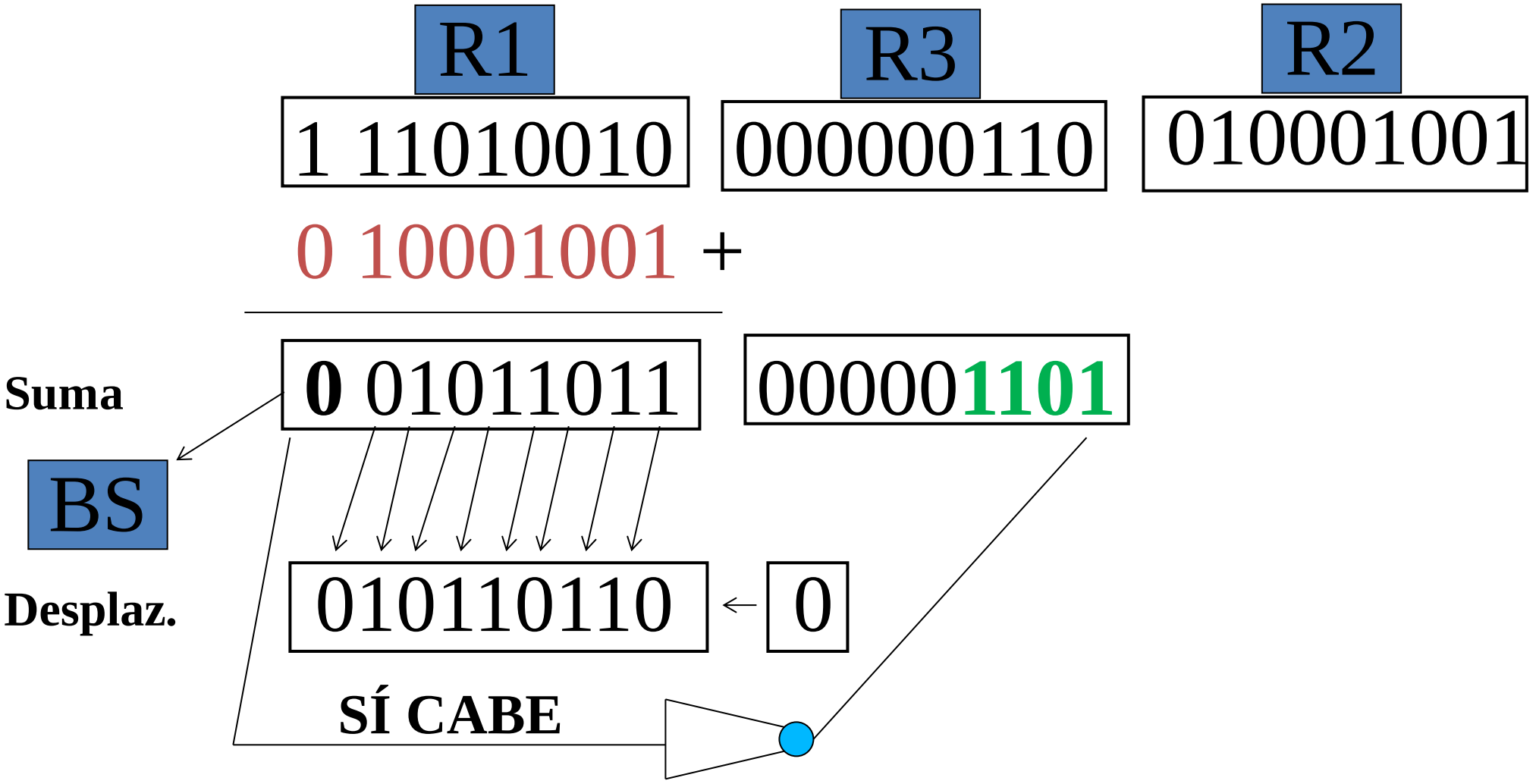
Como BS=0, R1←R1-R2



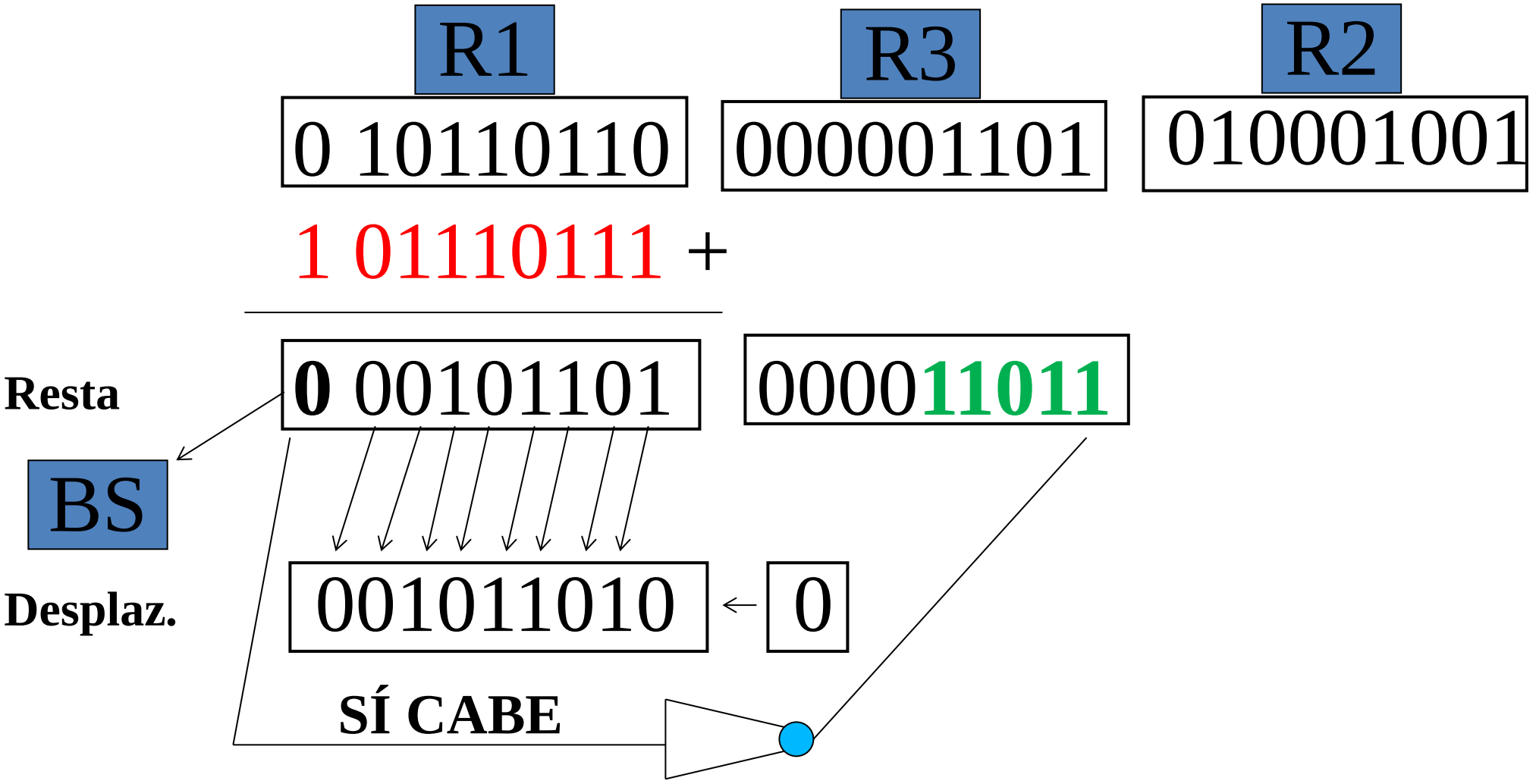
Como BS=0, R1←R1-R2



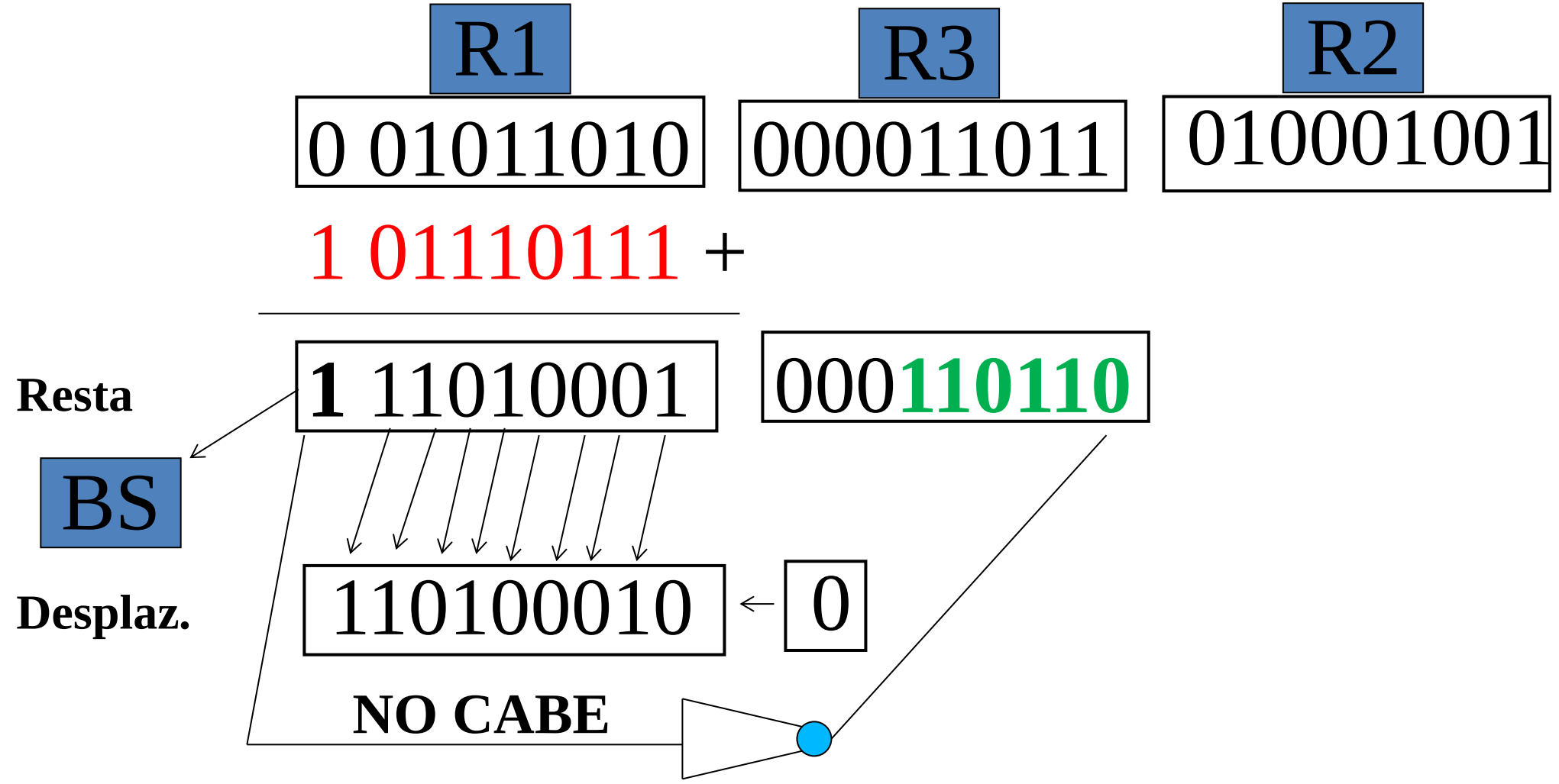
Como BS=1, R1←R1+R2



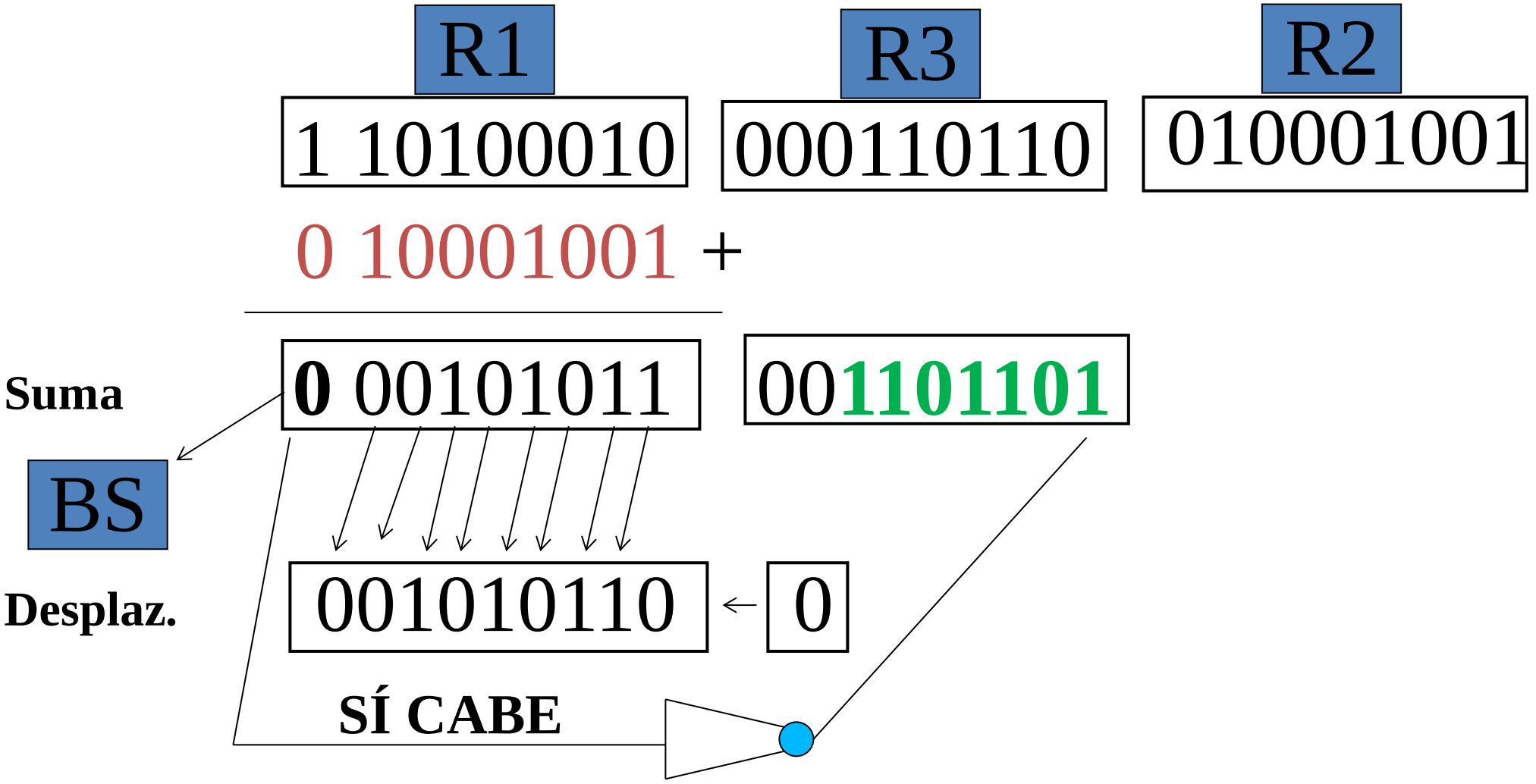
Como BS=0, R1←R1-R2



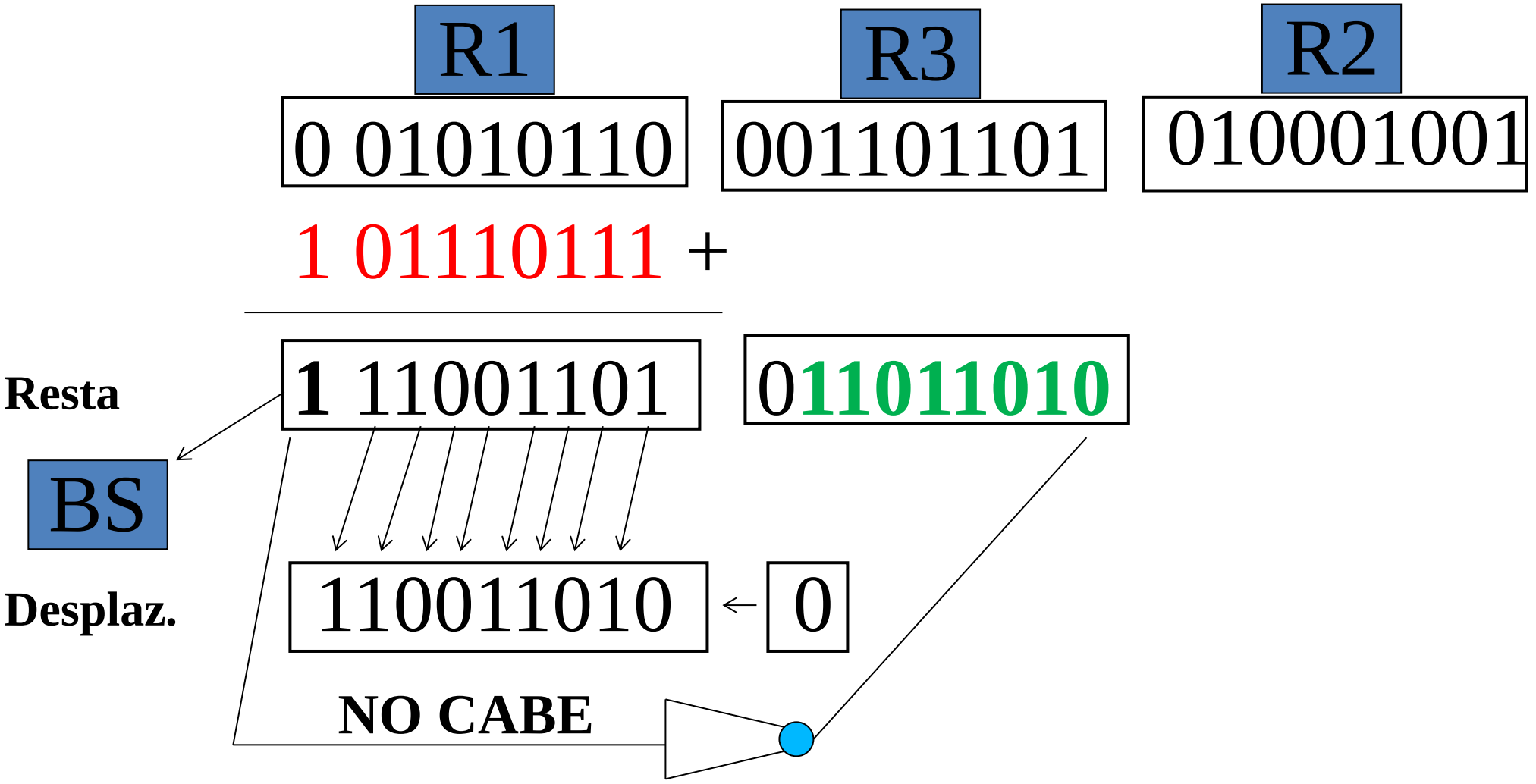
Como BS=0, R1←R1-R2



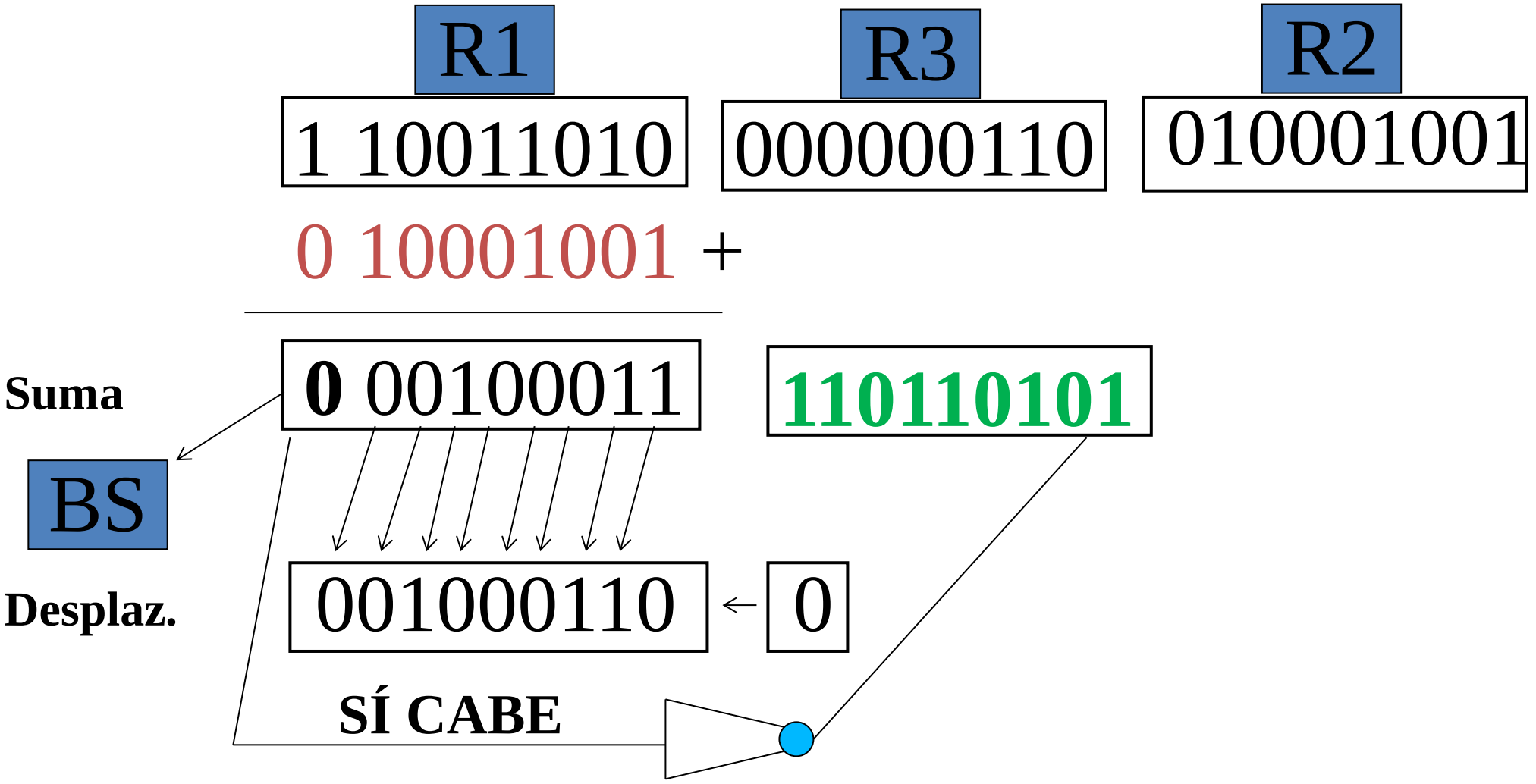
Como BS=1, R1←R1+R2



Como BS=0, R1←R1-R2



Como BS=1, R1←R1+R2



Cifras decimales (I)

Según el ejemplo visto:

$$\begin{array}{r} 11101010 \\ \hline 10001001 \end{array}$$

es igual a 110110101

¿Qué significa esto? Como el numerador es mayor que el denominador, el resultado será mayor que 1, pero no mayor que 2, ya que tienen el mismo número de bits. Por tanto, el resultado es:

1,10110101

Cifras decimales (II)

El valor exacto sería

234 / 137 = 1,708029197…

mientras que el valor que resulta en binario es 1,70703125

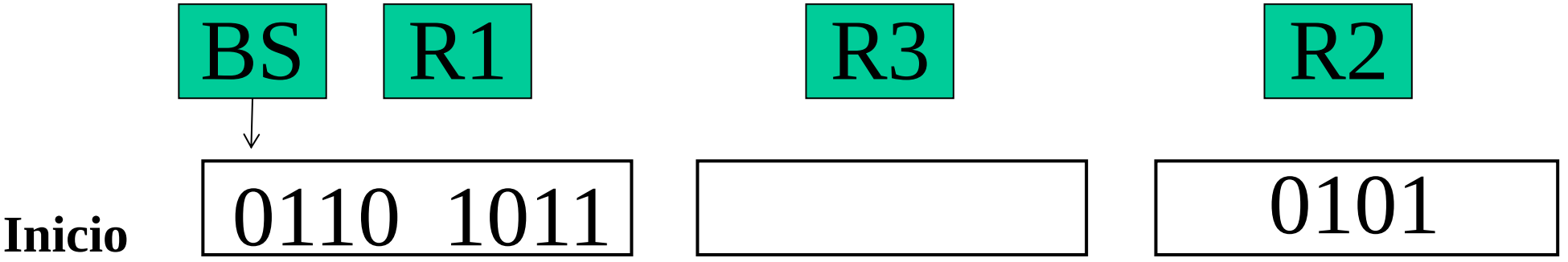
Ejemplo con obtención del resto **(SIN RESTAURACIÓN)**

Dividir A=1101011 (107 en decimal) entre B=101 (5 en decimal), cuyo cociente es 21 y el resto 2

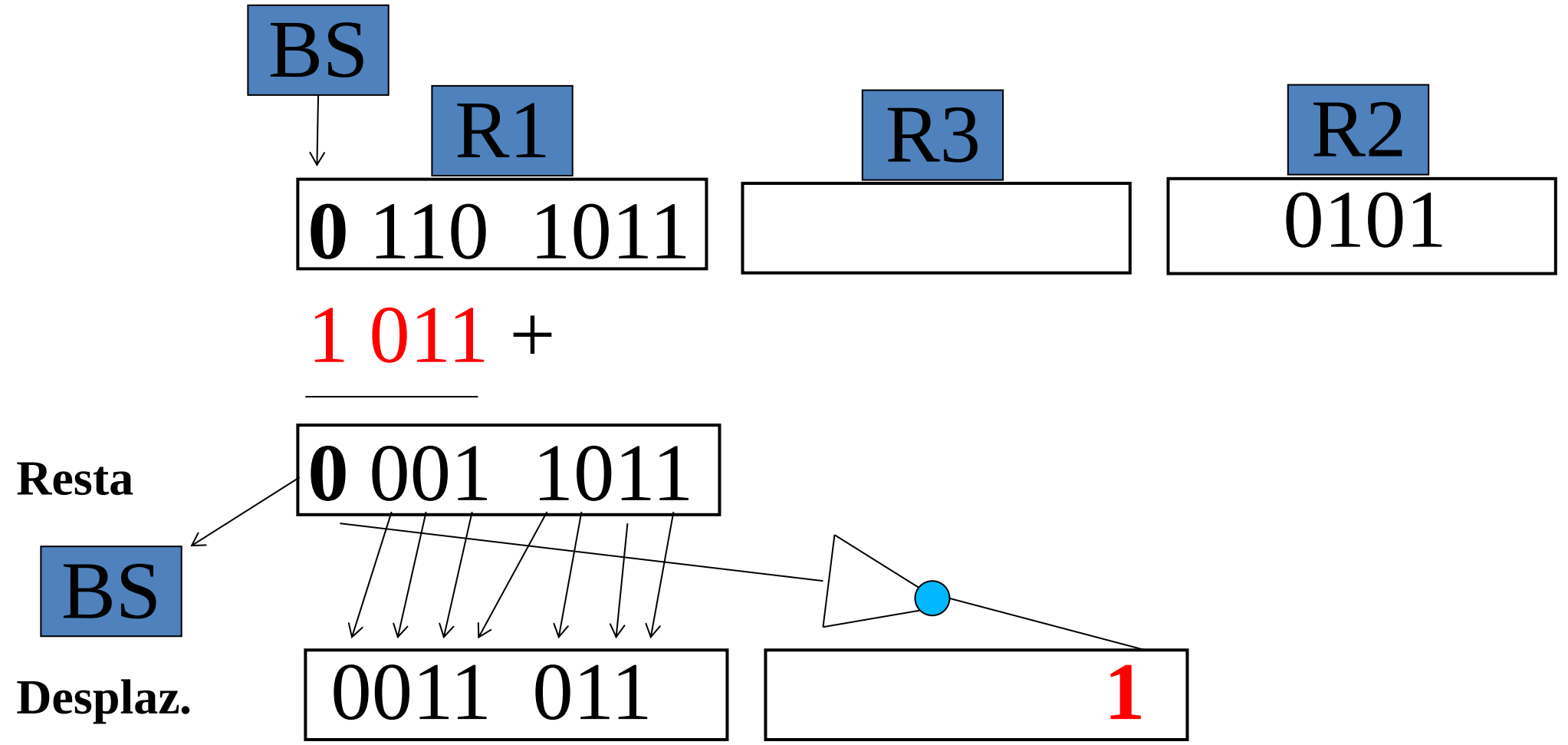
Vamos a trabajar con registros de 8 bits, en complemento a 2.

En este ejemplo, los operandos no tienen el mismo número de bits

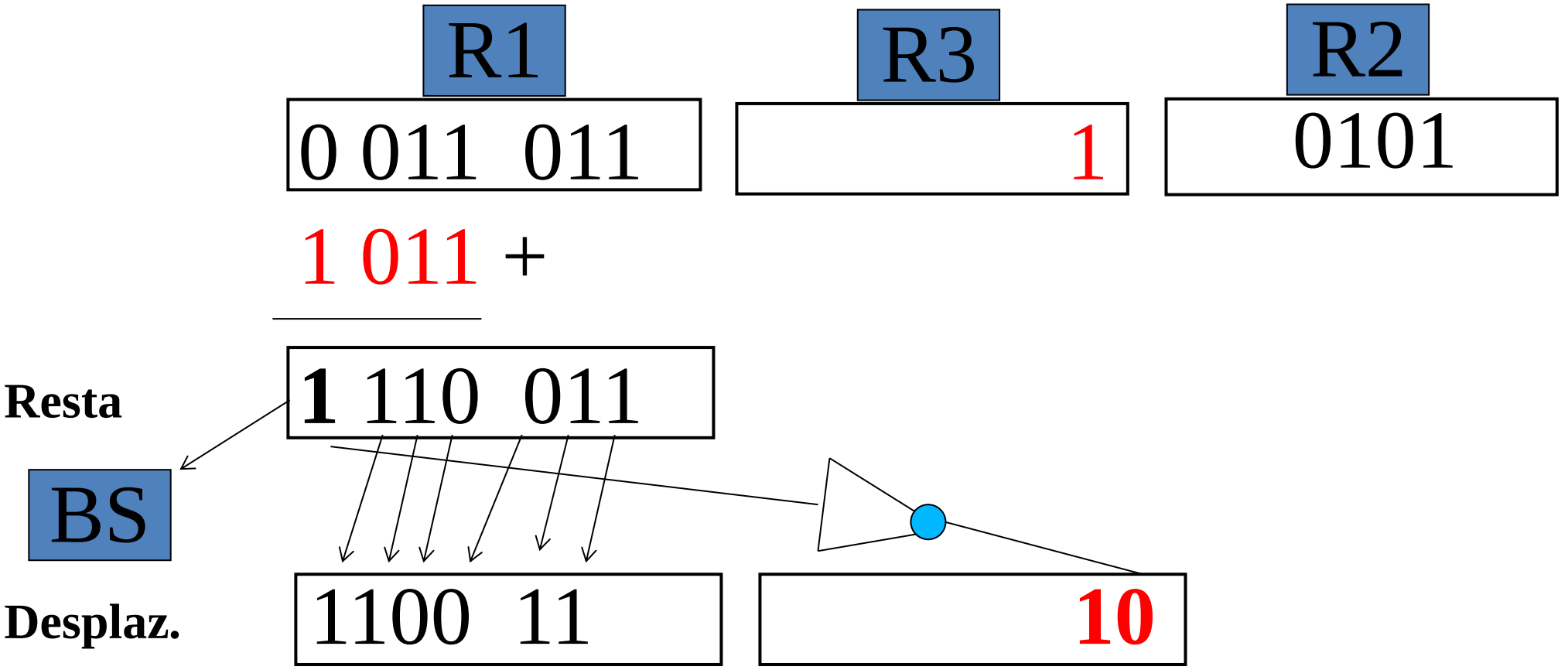
Para sumar R1+R2 se suma **0 101** (+5 en complemento a 2)
Para restar R1-R2 se suma **1 011** (-5 en complemento a 2)



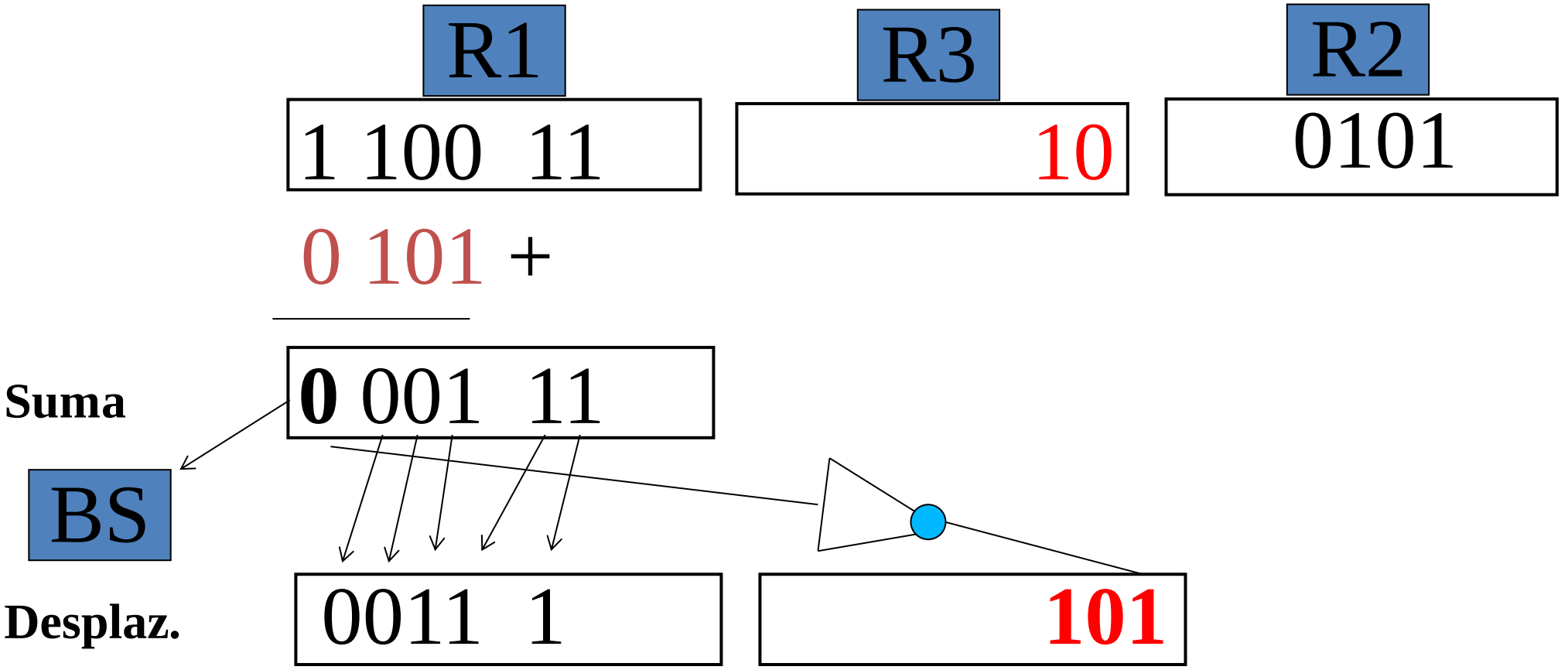
La primera vez BS=0, por tanto, R1←R1-R2



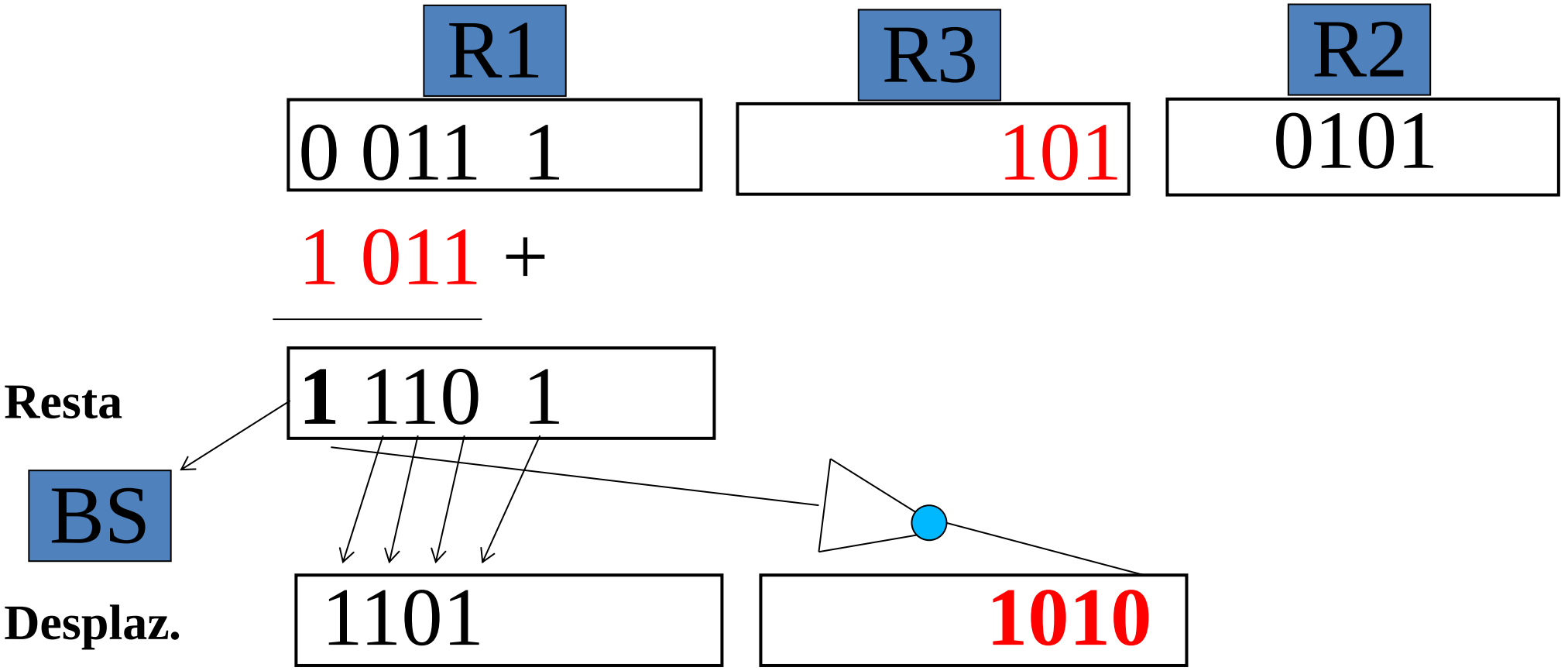
Como BS=0, R1←R1-R2



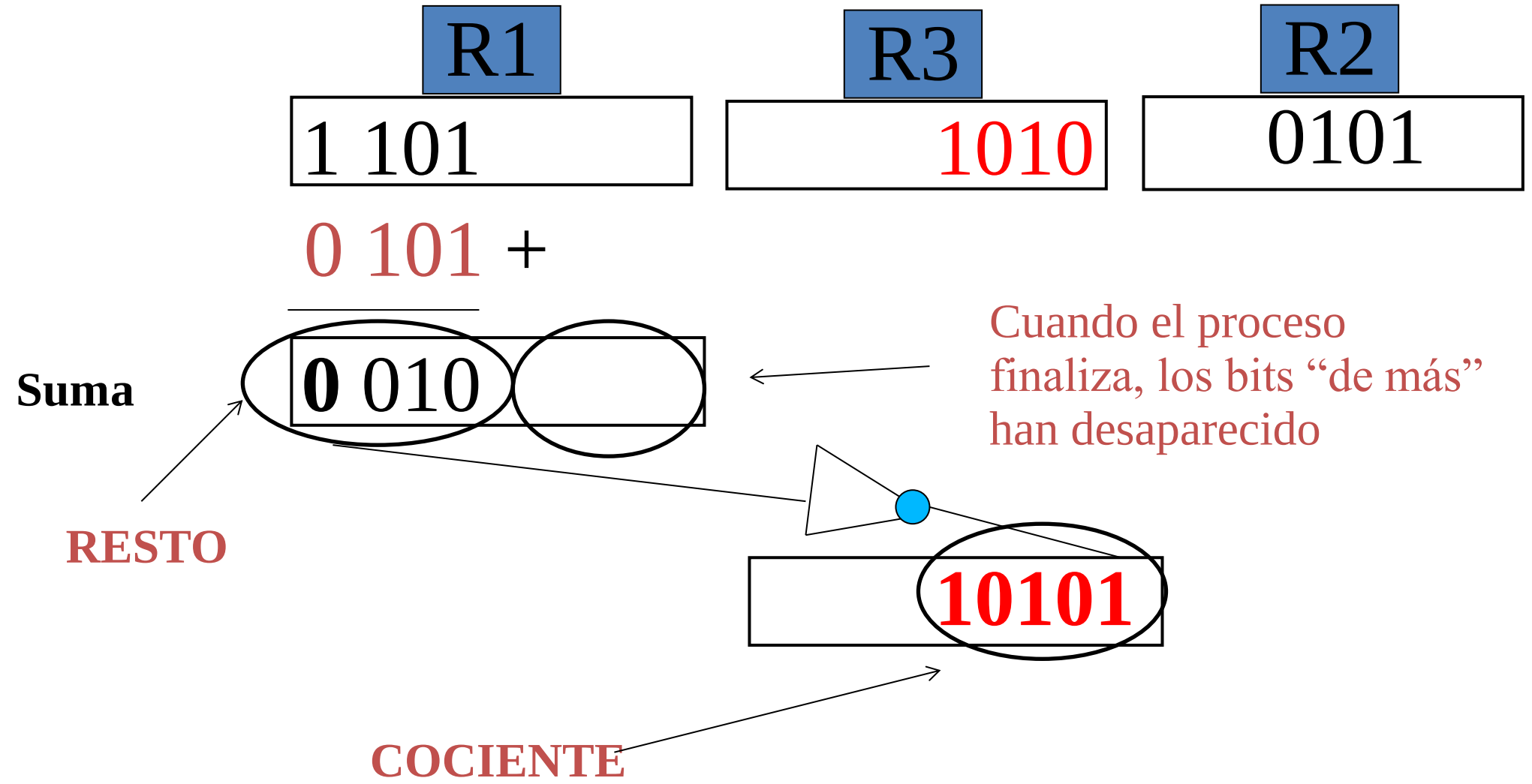
Como BS=1, R1←R1+R2



Como BS=0, R1←R1-R2



Como BS=1, R1←R1+R2



COMA FLOTANTE (I)

Inicialmente cada fabricante tenía su propia representación de números en coma flotante.

- Muchos problemas a la hora de transportar programas y datos entre computadores

El IEEE (Institute of Electrical and Electronics Engineers) definió un estándar en 1985 para la representación de números en coma flotante en binario IEEE 754

En 1987 se extendió el estándar para representar números en coma flotante independientemente de la base de numeración, IEEE 854

- Este nuevo estándar no supone cambios a IEEE 754

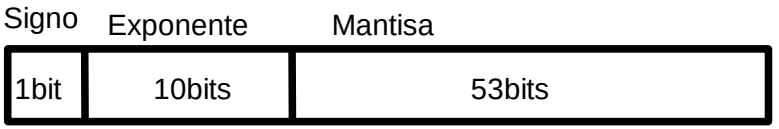
COMA FLOTANTE (II)

Todo número se puede representar como

$M * B^{exp}$

En IEEE 754 se pueden representar los números en simple precisión (32 bits) y en doble precisión (64 bits)

Simple Precisión. Exceso 127



Doble Precisión. Exceso 1023

$$N = (-1)^S * 1.M * 2^{(exp-exc)}$$

Ejemplo (I)

Representar el número real +2,71 en el formato IEEE 754, truncando los bits que no quepan en la representación

2,71 → 10,10110101111000010100011 x 2^0

2,71 → 1,01011010111000010100011 x 2^1

Mantisa→ 01011010111000010100011

Exponente → +1

Exponente + 127 → 128

Signo	Exponente + 127	Mantisa
0	1 0 0 0 0 0 0 0	01011010111000010100011

En Hexadecimal se podría expresar como 4 0 2 D 7 0 A 3

Ejemplo (II)

Determinar qué número real representa el hexadecimal

C8860800

El valor binario es

1 100 1000 1 000 0110 0000 1000 0000

Mantisa→ 00001100000010000000

Signo→ Negativo

Exponente +127 → 128+16+1 Exponente → 18

Valor absoluto → $1{,}000011000001 \times 2^{18}$

Valor absoluto → $1000011000001 \times 2^{6}$

Valor real → -4.289 x 64 = -274.496

Suma/Resta en coma flotante

Dados dos números

$A = m_1 * B^{e1}$

$B = m_2 * B^{e2}$

Se ajustan las mantisas y los exponentes hasta que e1=e2=e.

Entonces tendremos

$A = m_1 * B^{e}$ $B = m_2 * B^{e}$

La suma y la resta entonces se realizan de esta forma

$A + B = (m_1 + m_2) * B^{e}$ $A – B = (m_1 - m_2) * B^{e}$

Se normalizan los resultados a IEEE 754

Multiplicación / División en coma flotante

Dados dos números

$A = m_1 * B^{e1}$

$B = m_2 * B^{e2}$

La multiplicación y división se realiza:

$A * B = (m_1 * m_2) * B^{e1+e2}$

$A / B = (m_1 / m_2) * B^{e1-e2}$

Se normalizan los resultados a IEEE 754

Técnicas de Redondeo

El redondeo es la aproximación necesaria para representar en el computador una cantidad que no puede expresarse de forma exacta

El estándar IEEE 754 exige que el resultado de una operación sea el que se hubiese obtenido teniendo una precisión infinita y al final se redondee el resultado a la precisión exigida (simple o doble)

Tipos de redondeo:

- Truncación
- Redondeo propiamente dicho
- Bit menos significativo forzado a 1

Distintos casos

- **Truncación:** eliminación de los bits de la derecha (menos significativos) que no caben en la representación. Es muy sencillo de implementar. Puede producir errores apreciables, siempre del mismo signo, lo que propicia la acumulación de errores
- **Redondeo:** se representa haciendo una aproximación al más cercano. Similar al redondeo de los céntimos de €. Los errores pueden ser positivos o negativos, pero menores que en el caso de la truncación. Es más difícil de implementar
- **Forzado a 1:** se trunca, pero se obliga a que el bit menos significativo sea 1. Es fácil de implementar y produce errores pequeños y de signo variable, evitando su acumulación

Ejemplos de redondeo a 10 bits

Valor exacto	Truncación	Redondeo	Forzado a 1
0,1100110111 111	0,1100110111	0,1100111000	0,1100110111
0,1000100010 111	0,1000100010	0,1000100011	0,1000100011
0,1000000000 100	0,1000000000	0,1000000001	0,1000000001
0,1111111111 000	0,1111111111	0,1111111111	0,1111111111