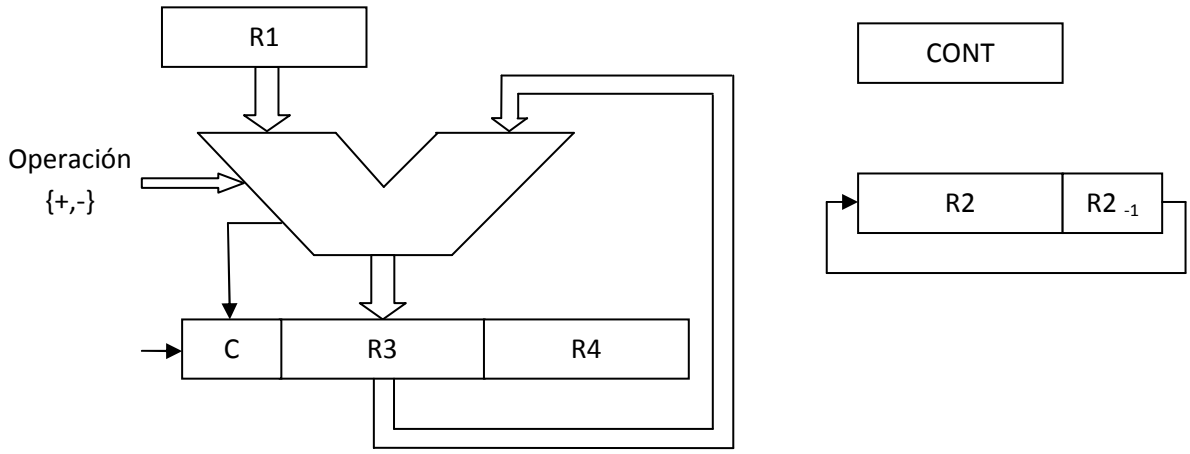
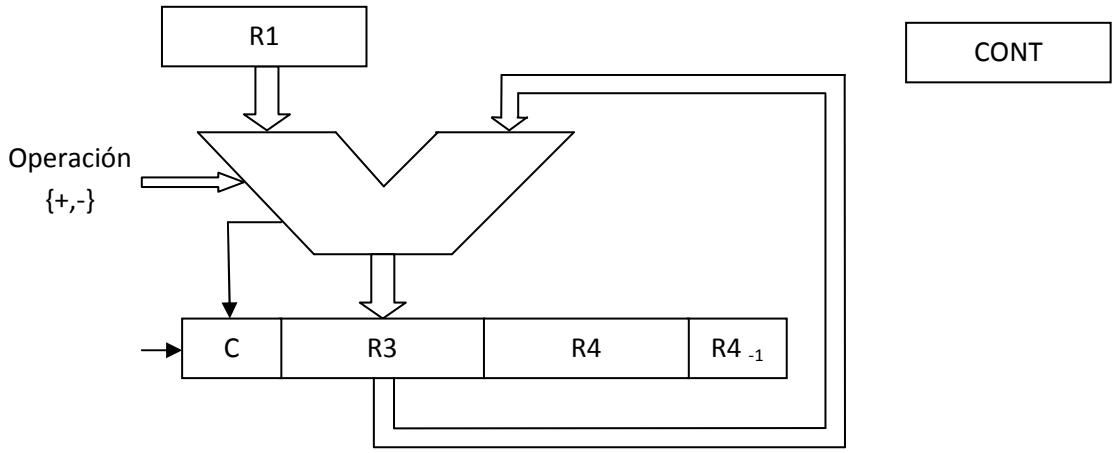
Multiplicación: algoritmo de Booth

El algoritmo de Booth permite multiplicar números positivos y negativos representados en complemento a 2. Existen 2 posibles arquitecturas:

Empleando el registro R2 para almacenar el multiplicador y mantenerlo la final de la operación. R1 contiene el multiplicando y C-R3-R4 contiene el resultado.



Empleando el registro R4 para almacenar el multiplicador. El valor del multiplicador se pierde al final de la operación. R1 contiene el multiplicando y C-R3-R4 contiene el resultado.



Principio de funcionamiento (empleando la primera arquitectura):

Se analizan el bit menos significativo de R2 ($R2_0$) y $R2_{-1}$ y se establecen los siguientes casos:

$R2_0$	$R2_{-1}$	Operación
0	0	No se realiza ninguna operación
0	1	R3=R3+R1 (Inicio cadena 0's)
1	0	R3=R3-R1 (Inicio cadena 1's)
1	1	No se realiza ninguna operación

En cada iteración se incrementa el contador (CONT), se desplaza C-R3-R4 a la derecha y se rota circularmente a la derecha R2.

Ejemplos:

a) $A \cdot B = 0110_2 \cdot 0010_2 = 000001100_2$ $(6_{10} \cdot 2_{10} = 12_{10})$

CONT	C	R3	R4	$R2_{321}$	$R2_0$	$R2_{-1}$	Operación realizada
0	0	0000	xxxx	001	**0**	**0**	Inicializar
1	0	0000	0xxx	000	1	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
1	0 1 1	0000 1010 1010	0xxx	000	1	0	R3=R3-A (R3=R3+ca2(A))
2	1	1101	00xx	000	0	1	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
2	1 0 0	1101 0110 0011	00xx	000	0	1	R3=R3+A
3	0	0001	100x	100	0	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
4	0	0000	1100	010	0	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$

Nota: Rotando R2 una vez más se restaura el valor del multiplicador.

b) $A \cdot B = 0110_2 \cdot 1110_2 = 111110100_2$ $(6_{10} \cdot (-2_{10}) = -12_{10})$

CONT	C	R3	R4	$R2_{321}$	$R2_0$	$R2_{-1}$	Operación realizada
0	0	0000	xxxx	111	**0**	**0**	Inicializar
1	0	0000	0xxx	011	1	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
1	0 1 1	0000 1010 1010	0xxx	011	1	0	R3=R3-A
2	1	1101	00xx	001	1	1	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
3	1	1110	100x	100	1	1	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
4	1	1111	0100	110	0	1	Desplazar C-R3-R4, rotar R2-$R2_{-1}$

c) $A \cdot B = 1010_2 \cdot 0010_2 = 111110100_2$ $(-6_{10} \cdot 2_{10} = -12_{10})$

CONT	C	R3	R4	$R2_{321}$	$R2_0$	$R2_{-1}$	Operación realizada
0	0	0000	xxxx	001	**0**	**0**	Inicializar
1	0	0000	0xxx	000	1	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
1	0 0 0	0000 0110 0110	0xxx	000	1	0	R3=R3-A
2	0 1 1	0011 1010 1101	00xx	000	0	1	R3=R3+A
3	1	1110	100x	100	0	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
4	1	1111	0100	010	0	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$

d) $A \cdot B = 1010_2 \cdot 1110_2 = 000001100_2$ $((-6_{10}) \cdot (-2_{10})) = 12_{10}$

CONT	C	R3	R4	$R2_{321}$	$R2_0$	$R2_{-1}$	Operación realizada
0	0	0000	xxxx	111	**0**	**0**	Inicializar
1	0	0000	0xxx	011	1	0	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
1	0 0 0	0000 0110 0110	0xxx	011	1	0	R3=R3-A
2	0	0011	00xx	001	1	1	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
3	0	0001	100x	100	1	1	Desplazar C-R3-R4, rotar R2-$R2_{-1}$
4	0	0000	1100	110	0	1	Desplazar C-R3-R4, rotar R2-$R2_{-1}$

Comentarios finales:

- **El algoritmo de Booth es igual al de multiplicación con sumas y restas, simplemente demuestra su validez para multiplicación con signo representando los operandos en complemento a 2.**
- **Este algoritmo (y el de multiplicación con sumas y restas) permiten ahorrar operaciones si existen cadenas de 1s o 0s en el multiplicador.**
- **Una posible mejora para aumentar la rapidez en la ejecución del algoritmo sería, aprovechando la propiedad conmutativa de la multiplicación, seleccionar como multiplicador entre ambos operandos aquel que tenga menos transiciones entre 0 y 1 y viceversa.**