Universidad del País Vasco
Facultad de Informática

Departamento de Arquitectura y Tecnología de Computadores

Arquitectura de Computadores I

Aritméticos 4 (solución)
Multiplicaciones: algoritmo suma o salta

Explica paso a paso cómo se consigue el resultado de la multiplicación entre los números A = –7 eta B = –11 mediante el algoritmo *suma o salta*. Hazlo de dos maneras: sin recodificar los operandos y recodificando los operandos según la representación de Booth en base 2. Los operandos están representados en 6 bits utilizando complemento a la base, con base 2.

Solución

Primero tenemos que representar los valores de los operandos. Están en complemento a la base y la base es 2, por lo tanto, la representación es complemento a 2. En principio, para representar el número –7, son suficientes 3 bits, pero para representar –11 son necesarios 4 bits. De todas maneras, en el enunciado se nos pide que los representemos con 6 bits cada uno.

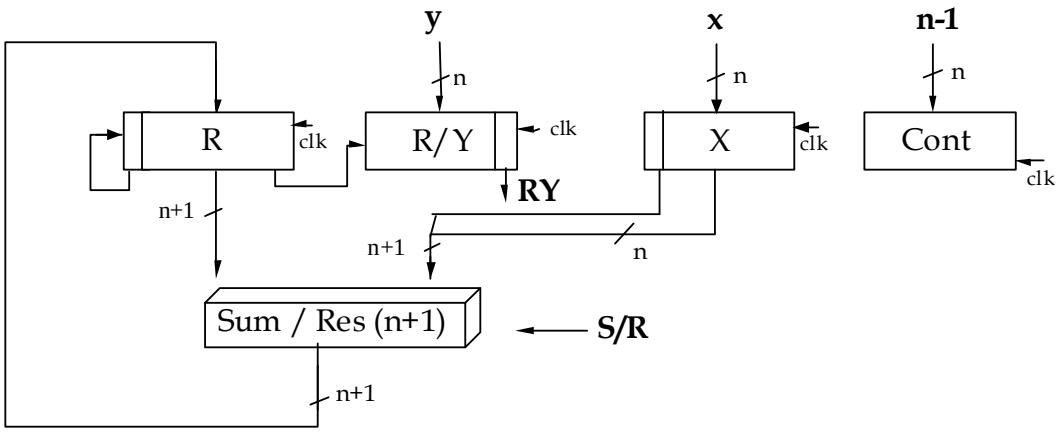
Para representar el valor –7, con 6 bits y en complemento a 2 se utiliza la siguiente secuencia de bits: `111001`; para representar el valor –11 se utiliza la siguiente secuencia de bits: `110101`.

En cambio, si utilizáramos Booth en base dos para representar esos números, –7 se representaría con la secuencia `00-101-1` y –11 con la secuencia `0-11-11-1` utilizando en ambos casos 6 bits. En esta solución utilizaremos –11 recodificado.

Veamos cómo se hace la multiplicación mediante el algoritmo de *"Suma o salta"*, si los números están representados en complemento a 2. Hay que recordar que para que no haya desbordamiento en las sumas intermedias, hay que hacerlas con (n + 1) bits (en este caso 7 bits). En lo que respecta al hardware, los registros son los siguientes:

X	registro de n bits, multiplicando
R	un registro de desplazamiento de n+1 bits, para almacenar los bits de mayor peso del resultado (ampliando el signo)
R/Y	un registro de desplazamiento de n bits, para almacenar los bits de menor peso del resultado y/o el multiplicador.

Además de estos registros, en el hardware utilizado también aparece un contador, para controlar las n multiplicaciones/sumas parciales, y un sumador/restador de (n + 1) bits en complemento a 2 para poder hacer tanto las sumas parciales como las restas (en el caso de la restauración).



Utilizando este circuito, el algoritmo de multiplicación de *n* bits es el siguiente:

```
1:   X := x; Cont := n-1; RY := y; R := 0;
2:   Cont := Cont-1;  if RY(0) = 0 then goto 4;
3:   R := ADD (R, X);
4:   D_DE (R-RY); if Cont ≠ 0 then goto 2;
5:   if RY(0) = 1 then R := SUB (R, X);
6:   D_DE (R-RY);  FIN := 1;
```

Aplicando ese algoritmo a nuestros datos, nos quedan los siguientes cálculos:

```
              R         R|Y         X       Cont
           0000000   |110101   111001    5
(1)  -7    1111001                       4
           -------
           1111001
     ->    1111100   1|11010             3
(0)
     ->    1111110   01|1101             2
(1)  -7    1111001
           -------
           1110111
     ->    1111011   101|110             1
(0)
     ->    1111101   1101|11             0
(1)  -7    1111001
           -------
           1110110
     ->    1111011   01101|1
(-1) +7    0000111
           -------
           0000010
     ->    0000001   001101|
```

Por lo tanto, el resultado es `000001001101`, es decir, el número 77 (el valor que componen los doce bits de menor peso del resultado obtenido).

A la hora de explicar los pasos, hay que recordar que en el paso 2 del algoritmo Cont se decrementa en el mismo momento en que se analiza el bit de menor peso del registro R|Y. Después Cont pasará a tener el valor 0, entonces hay que analizar el valor del bit de menor peso del registro R|Y, y, si es 1, entonces hay que restarle X en vez de sumarle. En este ejercicio nos ha ocurrido eso.

En el registro R|Y en principio se almacena el multiplicando Y y luego se va desplazando hacia la derecha, teniendo en cuenta los bits provenientes del registro R. Al final, el resultado de la multiplicación se compone de los bits de menor peso de entre los bits obtenidos al concatenar los registros R y R|Y. Al principio en el registro X se almacena el otro multiplicando y no cambia durante la ejecución. Por otro lado, el contador Cont en todo momento indica cuántos pasos quedan para acabar el proceso.

Ahora aplicaremos el mismo algoritmo, pero representando el valor –11 según la recodificación de Booth en base 2. En este caso no es necesaria la variable Cont.

```
             R         R|Y            X
           0000000   |0-11-11-1     111001
(-1) +7    0000111
           -------
           0000111
     ->    0000011   1|0-11-11
(1)  -7    1111001
           -------
           1111100
     ->    1111110   01|0-11-1
(-1) +7    0000111
           -------
           0000101
     ->    0000010   101|0-11
(1)  -7    1111001
           -------
           1111011
     ->    1111101   1101|0-1
(-1) +7    0000111
           -------
           0000100
     ->    0000010   01101|0
(0)  ->    0000001   001101|
```

Aquí también nos sale el mismo resultado. El algoritmo irá analizando los bits de menor peso del registro R|Y. Cuando sea 0, habrá que desplazar como en el caso anterior. Cuando sea -1 se suma -X y después se desplaza. Por otro lado, cuando es 1, se suma +X. Al final el resultado será el obtenido al concatenar los contenidos de los registros R y R|Y.

En cualquier caso, el resultado se representa con 12 bits y no habido desbordamiento en ninguna operación intermedia.