Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Arquitectura de Computadoras

Trabajo Práctico Nº 5

Implementación de las Operaciones Básicas: Multiplicación y División

Primer Cuatrimestre de 2010

Ejercicios

1. Considerando la siguiente técnica de multiplicación para *dos-complemento*:

   > *Si el primer multiplicando es negativo, trabajar controlando la entrada serie (esto es, 0 o x_{n-1} luego de la primer suma) y cuando el segundo multiplicando es negativo, realizar la corrección del último paso.*

   determinar el resultado obtenido en los siguientes casos:

   a) 3×5

   b) -3×5

   c) 3×-5

   d) -3×-5

   Obs.: A manera de simplificación, trabajar con operandos de sólo *cinco bits*.

2. Llevar adelante los siguientes multiplicaciones trabajando en *dos-complemento* y utilizando la técnica de *recodificación de Booth*. Hacer explícitos los productos parciales, asumiendo en este caso operandos de *seis bits*.

   a) 5×-19

   b) -7×13

3. Sean los enteros en *dos-complemento* $X = 10010010$ e $Y = 11011101$:

   a) Calcular el producto $X \times Y$ trabajando con el algoritmo de multiplicación básico para enteros signados, haciendo las correcciones que correspondan.

   b) Determinar los distintos múltiplos de X requeridos previo a la realización de la multiplicación al hacer uso de la recodificación de Booth sobre Y, agrupando de a *dos bits* (en otras palabras, *radix 4*), siguiendo el esquema *look-ahead*. Tener presente que bajo este esquema se deben analizar los dos bits a recodificar (b_i y b_{i+1}) junto a el último bit de la parte aún no codificada (b_{i+2}).

   c) Rehacer el inciso anterior, en esta oportunidad siguiendo el esquema *look-behind*. Tener presente que bajo este esquema se deben analizar los dos bits a recodificar (b_i y b_{i+1}) junto con el último bit de la parte ya codificada (b_{i-1}). ¿Cambian los múltiplos de X que se deben calcular previo a la multiplicación? ¿Cuáles resultan más fáciles de obtener?

4. Se desean acelerar el cálculo de las siguientes multiplicaciones haciendo uso de la técnica *recodificación de Booth* para *radix 4*, haciendo explícito los productos parciales que resultan al adoptar tanto el esquema *look-ahead* como el *look-behind*:

   a) 45×-39

   b) -12×65

5. Construir la tabla que permite realizar la *recodificación de Booth*, pero esta vez de a tres bits (esto es, *radix 8*). ¿Qué se puede afirmar en cuanto a los operandos que se necesitan para llevar adelante esta recodificación y sobre su generación en comparación con la recodificación estándar de tres bits?

6. Esquematizar un *Wallace Tree* con **CSA**s para resolver el producto de dos operandos de 16 bits. ¿Cuántos **CSA**s se requieren? ¿Respeta la cota presentada en la teoría? ¿Cuántos niveles de **CSA**s hacen falta?

7. Esquematizar un *Wallace Tree* con **CSA**s para resolver el producto de dos operandos de *ocho bits*, indicando como quedan los operandos de entrada al árbol al suponer $X = 89$ e $Y = 55$. Determinar los operandos intermedios y finales a la salida de los **CSA** y efectuar también la suma paralela a manera de verificación.

8. Esquematizar un *Wallace Tree* con **CSA**s para resolver el producto de dos operandos $X = 67$ e $Y = -12$ de *ocho bits*, recodificando de a *dos bits por vez* (esto es, tomando como entrada a los distintos **CSA**s al producto $Y_{i+1}Y_i \times X$), indicando claramente la entrada a ser recibida por los distintos **CSA**s del primer nivel. Bajo esta nueva configuración, ¿cuántos niveles de **CSA**s hacen falta? ¿Qué ganancia se obtiene al recodificar de a varios bits? ¿Qué costo adicional aparece?

9. Bosquejar un *árbol binario de multiplicación* que permita calcular el producto de dos operandos $X = 16$ e $Y = 24$ de *ocho bits* al utilizar *dígito signado*, indicando a su vez como quedan los operandos de entrada al árbol y a los nodos interiores. A manera de verificación, resolver la suma paralela que convierte a binario el resultado.

10. Esquematizar el proceso de *división con restoring* $X \div Y$ para los enteros positivos $X = 01100$ (dividendo) e $Y = 00101$ (divisor) y obtener Q (cociente) y R (resto). Trabajar con *seis bits*, teniendo en cuenta que tanto el acumulador como la **ALU** deben ser de $n + 1$ bits, dado que se debe tener un bit adicional para representar el signo.

11. Esquematizar el proceso de *división sin restoring* $X \div Y$ para los enteros positivos de $n = 7$ bits $X = 1101011$ (dividendo) e $Y = 0010110$ (divisor) y obtener Q (cociente) y R (resto).

12. Sea $X = 0,11011$ e $Y = 0,11$ dos mantisas normalizadas de punto flotante. Realizar el cálculo del cociente $X \div Y$, empleando el método de división rápida basado en la recíproca. Hacer el cómputo de cada uno de los r_i recordando que si genéricamente $Y \cdot r_1.r_2.r_3. \cdot \cdot \cdot \cdot \cdot r_{k-1} = 1 - Z_k$ el r_k resultante es $r_k = 1 + Z_k$ con lo cual los r_k se pueden calcular como el dos-complemento de $Y \cdot r_1.r_2.r_3. \cdot \cdot \cdot \cdot \cdot r_{k-1}$. Obtener solo los dos primeros coeficientes r_1 y r_2, calcular $X \cdot r_1 \cdot r_2$ y verificar el error respecto al resultado exacto.

13. Calcular en base al *algoritmo de division SRT* el cociente entre 49/6 y 36/5 con hardware de *6 bits*.

    Obs.: La determinación de cuándo el R_i está acotado por $-b/2 \leq Ri < b/2$ (lo que asegura que $\mid R_i \mid \leq b$), dado que $b \geq 1/2$ se satisface con el intervalo $[-1/4, +1/4)$ adoptado para el r_i, y esto se puede resolver analizando dos bits en la posición más significativa de $A.MQ$ y no tres bits, el bit de signo podemos no tomarlo en cuenta dado que es redundante, puesto que el bit de signo replica al bit de la posición más significativa. Verificar esta aseveración por el absurdo.

14. Calcular en base al *algoritmo de division sin restoring SRT higher-radix 4* el cociente $X \div Y$, consultando la *lookup-table*, con $X = 135$ e $Y = 4$.

    Obs.: Trabajar con *nueve bits* (signo más ocho bits para la fracción).

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

Referencias

[Bae80] Baer, J. L. *Computer Systems Architecture*. Computer Science Press, 1980.

[HP96] Hennessy, J., and Patterson, D. *Computer Architecture*, second ed. Morgan Kaufmann, 1996.