2. ARITMÉTICA EN COMPUTADORES

[Entre corchetes figura el apartado del tema más relacionado con el problema.]

2.1. [2.1] A) Reescribe la expresión de c_n que aparece en la transparencia *Suma entera-2* utilizando sumatorios (Σ) y "productorios" (Π). B) Teniendo en cuenta que $c_{j+1} = G_{ij} + P_{ij}\ c_i$, (j>i) y el resultado del apartado anterior, obtén una expresión para G_{ij} y otra para P_{ij}. C) Demuestra las expresiones de G_{ik} y P_{ik} que aparecen en la transparencia *Suma entera-3*.

2.2. [2.2 y transparencia *Mult/div enteras-2*] En la codificación de Booth ordinaria el múltiplo de *b* utilizado en el paso i-ésimo de la multiplicación es $(a_{i-1} - a_i) \cdot b$. Averigua una fórmula de este tipo para la codificación de Booth de números en base 4.

2.3. [2.2.2] A) Utiliza la tabla de suma de dígitos con signo que tienes en la transparencia *Mult/div enteras-6* para sumar los números $1\underline{1}0_2$ y 001_2. El dígito –1 lo representamos como $\underline{1}$.

B) Dibuja el sumador de tres dígitos que corresponde a la implementación hardware del método de suma aplicado en el apartado A. El diseño ha de realizarse de acuerdo con las siguientes directrices:

Utilizar una codificación de los dígitos que permita saber si un dígito es negativo consultando un único bit. Por ejemplo, la codificación que asocia cada par de bits con cada dígito del siguiente modo:

c(0,0) = 0; c(0,1) = 1; c(1,0) = c(1,1) = $\underline{1}$.

Usar dos tipos de módulos, uno para cada paso del algoritmo. Para el paso primero, los módulos calculan los dígitos de acarreo y suma a partir de los dígitos de entrada, a y b, y del bit de signo de los dígitos a su derecha. Para el paso segundo, los módulos obtienen el dígito de suma de peso i, d_i, a partir de los dígitos s_i y c_i obtenidos en el paso primero.

Solo se exige la arquitectura externa de cada tipo de módulo (sus entradas y salidas) y un dibujo de la forma en que se conectan los módulos necesarios entre sí para formar un sumador de tres dígitos.

C) Demuestra que la tabla garantiza que al dar el segundo paso de suma no se genera acarreo.

2.4. Pruébese que al finalizar el algoritmo de la trasparencia Mult/div enteras-7, el cociente por el divisor más el resto es igual al dividendo.

2.5. [2.4] Tomando mantisas binarias con 4 bits significativos (p=4) aplica el algoritmo de multiplicación en punto flotante para calcular –5 x 10 en cada uno de los cuatro modos de redondeo.

2.6. [2.4] Escribe el algoritmo para detectar desbordamiento tras la multiplicación en punto flotante de dos números en simple precisión. Denomina ce1 y ce2 a los campos de exponente de los números y supón que el registro P contiene el producto de las mantisas antes del redondeo. ¿Por qué se utiliza un sumador de 9 bits?

2.7. [2.5] Aplica el algoritmo de suma en punto flotante al cálculo de: a) $-1.001_2 \times 2^{-2} + -1.111_2 \times 2^0$; b) $-1.010_2 + 1.100_2$.

2.8. [2.6] Raíz cuadrada iterativa:

A) Utiliza el método de Newton para obtener un algoritmo iterativo que permita calcular la raíz cuadrada de un número a. En cada iteración debe aparecer una división (sin contar una división por 2).

B) ¿Cómo dividirías un número en punto flotante por 2?

C) Si la división es una operación lenta, el cálculo de la raíz cuadrada planteado en el apartado A también lo será. Pero aplicando la iteración de Newton a la función $f(x) = 1/x^2 - a$ ¡se puede encontrar un método que no precisa ninguna división! ¿Cuál es ese método?

D) Supongamos que la relación entre tiempos necesarios para las operaciones *división por 2 : suma en punto flotante : multiplicación en punto flotante* es 1 : 2 : 4. ¿Qué relación debe existir entre el tiempo de una división en punto flotante y el de una multiplicación en punto flotante para que una iteración del método utilizado en el apartado A sea tan rápida como una del método utilizado en el apartado C.

E) Cuando se utiliza el método del apartado A, ¿cuántos bits correctos debe tener como mínimo la primera aproximación de $a^{1/2}$ (x_0) para garantizar en tres iteraciones un valor correcto en doble precisión? Supón que el número de bits correctos de una aproximación, x, a un valor exacto, v, viene dado por $\lfloor \log_2 |v / (x - v)| \rfloor$.