# Ejercicios repaso 1° parcial

1) Utilizar el algoritmo de Booth para multiplicar 010011 (multiplicando) por 011011 (multiplicador).

   El algoritmo de Booth permite hacer multiplicaciones en Complemento a 2.

   Ej. 19= 010011 x 27= 011011

   LEYES:

| | |
|---|---|
| **0 a 1** | Reg. A + C2 (M) (multiplicando) y desplazo a la derecha |
| **1 a 1** | desplazo a la derecha |
| **0 a 0** | desplazo a la derecha |
| **1 a 0** | Reg. A + M (multiplicando) desplazo a la derecha |
| | si el resultado es negativo (1 adelante) antes de desplazar introduzco un 1 en el bit de mayor peso y luego desplazo (así mantengo el signo), si no, introduzco un 0 |

2) Realizar las siguientes divisiones (en magnitud y signo)
   a) 100101101 ÷ 01010 = (-45) ÷ 10
   b) 001100100 ÷ 10111 = 100 ÷ (-7)

*Contador*: Se carga con el número de bits del divisor y lleva la cuenta de los desplazamientos realizados.
*Acumulador*: Se carga con la parte más significativa del dividendo. Va almacenando los resultados.
*Q*: Se carga con la parte menos significativa del dividendo. Al finalizar el algoritmo queda almacenado el cociente.
*D*: Se carga con el divisor. (es el sustraendo de la resta).

Ej: 100 /7 01100100 / 0111

| ACCIÓN | REG. ACUMUL. | REG Q. | REG D. | CONTADOR |
|---|---|---|---|---|
| | 0110 | 0100 | 0111 | 4 |
| ← desplazo hasta que aparezca un 1 y se pueda hacer la resta | 1100 | 1000 | | 3 |
| resto y | 0111<br>0101 | 1000<br>1001 | Cuando se pueda hacer la resta entra un uno en $Q_0$ | |
| ← desplazo | 1011 | 0010 | | |
| resto y | 0111<br>0100 | 0010<br>0011 | Cuando se pueda hacer la resta entra un uno en $Q_0$ | 2 |
| ← desplazo | 1000 | 0110 | | |

| | | | | |
|---|---|---|---|---|
| resto y | 0111<br>0001 | 0110<br>0111 | | 1 |
| ←<br>desplazo | 0010 | 1110 | | |

Cociente 1110: 14
Resto 0010: 2

3) Realizar las siguientes operaciones con números en punto flotante en el formato IEEE 754 (en las que las partes significativas se truncan a cuatro dígitos decimales). Indique los resultados en forma normalizada). Calcule el error cometido.
   a) C30C0000 + C1500000
   b) 3B370000 + 39F68000

**Representación de números en punto flotante**
Según esta normalización, los números pueden ser representados de acuerdo a los siguientes formatos según correspondan a simple o doble precisión.

SIMPLE PRECISIÓN

| S | Exponente | Mantisa |
|---|---|---|
| 1 bit | 8 bits | 23 bits |

DOBLE PRECISIÓN

| S | Exponente | Mantisa |
|---|---|---|
| 1 bit | 11 bits | 52 bits |

Donde:
**S** es el bit de signo del número y es **0** para **+** y es **1** para **-**

**EXP** es el exponente de 2 que está representado en exceso 127 para simple precisión y exceso 1023 para doble precisión.

**MANT** es la mantisa y cumple con la condición de ser la parte del número representado que queda a la derecha de la coma cuando a la izquierda de ella hay un sólo 1.

Por lo tanto el número representado se obtiene así:

$(-1)^S \cdot 2^{(EXP-127)} \cdot 1,MANT$ simple precisión

$(-1)^S \cdot 2^{(EXP-1023)} \cdot 1,MANT$ doble precisión

Para representar los números cercanos a cero se utiliza un formato desnormalizado, haciendo EXP=0 y MANT ¹ 0.
El número que se representa cuando EXP=0 es:

| | |
|---|---|
| $(-1)^S . 2^{-126} . 0,MANT$ | simple precisión |
| $(-1)^S . 2^{-1022} . 0,MANT$ | doble precisión |

En esta normalización el cero se representa con EXP=0 y MANT=0.

a) `C30C0000 + C1500000`

C30C0000 → 1 | 100 0011 0 | 00011000000000000000000 = $-1.00011 \times 2^7$
C1500000 → 1 | 100 0001 0 | 10100000000000000000000 = $-1.101 \times 2^3$

Igualamos los exponentes al mayor de los dos y sumamos. El resultado obtenido se vuelve a normalizar. No olvidar el bit implícito que no está en la mantisa.

$-1.000\ 1100 \times 2^7$
$\underline{-0.000\ 1101 \times 2^7}$
$-1.001\ 1001 \times 2^7$

$-1.001\ 1001 \times 2^7$ → C3190000

4) **Escribir** un programa que defina **4 variables**: *a*,*b*,*c* y *d* inicializadas a *0*. Debe cargar en los registros *x5*,*x6*,*x7* y *x8* las **direcciones** de cada una de ellas usando 4 pseudoinstrucciones **la.** Ensamblar el programa. Abre la tabla de símbolos y comprueba que en los registros x5-x8 se han cargado las direcciones de las variables a-d respectivamente. ¿Cuántos bytes ocupa el programa?

5) **Escribir** un programa que recorra una TABLA de diez números enteros y determine cuántos elementos son mayores que X. El resultado debe almacenarse en una dirección etiquetada CANT. El programa debe generar además otro arreglo de palabras llamado RES cuyos elementos sean ceros y unos. Un '1' indicará que el entero correspondiente en el arreglo TABLA es mayor que X, mientras que un '0' indicará que es menor o igual.

6) Hacer un diagrama de tiempos usando adelantamiento de datos, analizar los riesgos (hazards) en la ejecución de esta porción de código y buscar alguna alternativa para optimizar el tiempo

```
lw t3, 0(t5)
add t8, t6, t3
addi t2, t5, 0x4
sw t5, 0(t2)
```

7) Analiza las dependencias de datos de la siguiente secuencia de instrucciones en una máquina segmentada Mips usando un diagrama de tiempos

```
lw t3 , 0 ( t5 )
add t8 , t6 , t3
add t9 , t8 , t5
add t7 , t2 ,t1
sw t5 , 0 ( t2 )
```

   a. Analiza la mejora a través de un diagrama de tiempo usando adelantamiento de datos.
   b. Analiza la mejora a través de un diagrama de tiempo usando reordenamiento.
   c. Analiza la mejora a través de un diagrama de tiempo usando reordenamiento y adelantamiento de datos.

8) Explica a nivel transferencia entre registros el proceso de búsqueda, decodificación y ejecución de la instrucción: addi t2, t8,0x5a

9) Describe a nivel transferencias entre registros el proceso de búsqueda, decodificación y ejecución de la instrucción: beq t3, t4, salto