Examen Parcial de
FCO – Temas 5 y 6

16 de Enero 2019

1.- **(1,5 puntos)** Complete las tres primeras filas de la tabla siguiente representando los 3 valores numéricos que aparecen, en las representaciones que faltan. Si no puede representar alguno de los valores, escriba FdR (fuera de rango). Indique también (en decimal) en la última fila el rango de representación de cada convenio:

Decimal	Signo y Magnitud 9 bits	Complemento a 2 9 bits	Exceso 255 9 bits
+128	010000000	010000000	101111111
-20	100010100	111101100	011101011
-3	100000011	111111101	011111100
Rango	[-255 , +255]	[-256 , +255]	[-255 , +256]

2.- **(1 punto)** Dados los números enteros A = 10110111_{Ca2} y B = 01101111_{Ca2} representados en complemento a 2 con 8 bits, realice las operaciones que se indican a continuación sin cambiar de representación. Indique claramente y justifique si hay o no hay desbordamiento. Muestre el detalle de su solución.

a) A + B **(0,4 puntos)**

Dado que se trata de una suma, los números se suman tal como están:

```
11111111
 10110111
+01101111
100100110      (se descarta el acarreo final)
```

Los dos últimos bits de acarreo, indicados por el recuadro, son iguales, luego NO hay desbordamiento.

El resultado de A + B es: 00100110_{Ca2}

b) A-B (0,6 puntos)

Como se trata de una resta, se cambia el signo al sustraendo haciendo el complemento a dos, y se convierte la resta en una suma:

Ca2(B) = Ca2(01101111) = 10010001

Ahora realizamos la suma A + (-B)

```
10110111
 10110111
+10010001
101001000
```

Los dos últimos bits de acarreo, indicados por el recuadro, son DISTINTOS, luego SÍ hay desbordamiento.

No hay resultado válido porque ha habido desbordamiento

3.- **(1 punto)** Dado el número real cuya representación en IEEE 754 de simple precisión es la secuencia de bits 0xC3018000 obtenga su valor en decimal detallando los pasos seguidos.

1100 0011 0000 0001 1000 0000 0000 0000

1 |10000110| 00000011000000000000000

Signo=1 (Negativo)

Exp: $10000110_{(exc\ 127)} - 01111111 = 00000111_{(2} = 7_{(10}$

Mant:0.00000011000000000000000

Bit implícito: **1**.00000011000000000000000 x 2^7=

10000001.1000000000000000 x 2^0=

Parte entera: **1**0000001$_{(2}$ = $2^7 + 2^0 = 129_{(10}$

Parte fraccionaria: 0.1000000000000000

0.1=1x2^{-1}= 0.5 Resultado = **-129,5**

4.- A partir del siguiente código escrito en lenguaje ensamblador del MIPS R2000:

```
.data 0x10001000
vector: .half  -12, 20, 8,-1, 15,-8
n: .word 6
calc1: .space 4
calc2: .space 4
.text 0x00400000
.globl __start

__start:
        la $2, vector
        la $3,n
        lw $3,0($3)
        add $4,$0,$0
        add $5,$0,$0
        add $6,$0,$0
loop:   lh $8,0($2)
        addi $6,$6,1
        slt $9,$8,$0
        beq $9,$0,salto1
        addi $4,$4,1
        j salto2
salto1: addi $5,$5,1
salto2: addi $2,$2,2
        slt $9,$6,$3
        bne $9,$0,loop
        la $2, calc1
        sw $4,0($2)
        sw $5,4($2)
.end.
```

a. **(0,5 puntos).** Indique el contenido del segmento de datos antes de ejecutarse el programa, teniendo en cuenta que los datos se almacenan en formato "little endian". El contenido debe especificarse por cada byte en hexadecimal. Indique claramente las zonas de memoria de contenido desconocido mediante un interrogante o guión.

31 … 24	23 … 16	15 … 8	7 … 0	Dirección
0x00	0x14	0xFF	0xF4	0x10001000
0xFF	0xFF	0x00	0x08	0x10001004
0xFF	0xF8	0x00	0x0F	0x10001008
0x00	0x00	0x00	0x06	0x1000100C
0x00	0x00	0x00	0x00	0x100010010
0x00	0x00	0x00	0x00	0x100010014
–	–	–	–	

b. **(1 punto).** Indique el contenido de los siguientes registros al finalizar la ejecución del código:

Registro	Contenido
$2	0x10001010
$3	0x00000006
$4	0x00000003
$5	0x00000003
$6	0x00000006
$8	0XFFFFFFF8
$9	0x00000000

c. **(0,5 puntos).** Indique la secuencia de instrucciones en que se traduce la pseudoinstrucción `la $2,vector`.

lui $1, 4096

ori $2, $1, 4096

d. (1,5 puntos). Indique el contenido del segmento de datos al finalizar completamente la ejecución del programa, teniendo en cuenta que los datos se almacenan en formato “little endian”. El contenido debe especificarse por cada byte en hexadecimal. Indique claramente las zonas de memoria de contenido desconocido mediante un interrogante o guión.

31 … 24	23 … 16	15 … 8	7 … 0	Dirección
				0x10001000
				0x10001004
				0x10001008
				0x1000100C
0x00	0x00	0x00	0x03	0x10001010
0x00	0x00	0x00	0x03	0x10001014

e. (1 punto). Se extrae del segmento de código la instrucción con código máquina 0x20c60001. Indique cuál de las instrucciones del programa anterior tiene esa codificación. Justifique la respuesta.

addi $6,$6,1

f. (1 punto). Codifique la instrucción `j salto2.` Indique el resultado en binario y hexadecimal y detalle los pasos realizados. Indique, además, la dirección en hexadecimal asociada a la etiqueta `salto2.`

Dirección etiqueta salto2: 0x0040003C

Codificación 0x0810000F

5.- **(1 punto)** Dada la asignación en memoria siguiente:

```
        .data 0x10000004
tira:.asciiz “Fundamentos”
long:.byte 0
```

escriba las líneas de código de un programa en lenguaje ensamblador del MIPS R2000 que calcule el número de caracteres almacenados en la posición **tira** y guarde el resultado en la posición de memoria etiquetada como **long**. El programa deberá ser capaz de realizar ese cálculo para cualquier tira de caracteres ubicada en la posición etiquetada como **tira**.

Utilice el registro $2 para almacenar la dirección de la cadena de caracteres etiquetada como **tira** y el $9 inicializado a 0 para calcular la longitud de la cadena de caracteres.

Solución posible:

```
        .globl __start
        .text 0x00400000

__start: la $2, tira  #carga dirección tira en registro
          li $9, 0 # inicializamos reg 9 para almacenar la longitud
  bucle: lb $10,0($2)
          beq $10,$0, fin # si carácter NULL, final de la tira
          addi $2, $2, 1 #inc. en 1 para leer sig. car
          addi $9, $9, 1 # inc. longitud
          j bucle
    fin: la $3, long    #dirección de long al reg 3
         sb $9,0($3)  #almacenamos long calculada en posición de memoria
        .end
```