Examen Segundo Parcial de FCO – Temas 5 y 6

23 de Enero de 2020

APELLIDOS: _____ NOMBRE: _____

DNI: _____ GRUPO:_____ FIRMA:

Normativa:

- La duración del examen es de 2:00h.
- **Por favor, escriba su nombre y apellidos en letras MAYÚSCULAS.**
- DEBE responder en el espacio asignado.
- No se permiten calculadoras ni apuntes.
- Debe permanecer en silencio durante la realización del examen.
- No se puede abandonar el examen hasta que el profesor lo indique.
- Debe tener una identificación en la mesa a la vista del profesor (DNI, carnet UPV, tarjeta residente, etc.)

1.- (2 puntos) Dados los números decimales A = -35, B = 81.

A) (**0,5 puntos**) Represéntelos mediante 8 bits en el convenio de representación Ca2, detallando todas las operaciones necesarias para llegar al resultado.

-35 = 11011101 y 81 = 01010001

B) (**0,75 puntos**) Realice la operación A+B en el convenio de representación Ca2 utilizando 8 bits, e indique claramente si hay desbordamiento, justificándolo.

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

11010001 (Acarreos)
11011101 (A)
+01010001 (B)
00101110 (Resultado. Se descarta el acarreo final)

Los dos últimos bits de acarreo, indicados por el recuadro, son iguales (11), luego NO hay desbordamiento (V=0).

C) **(0,75 puntos)** Realice la operación A-B en el convenio de representación Ca2 utilizando 8 bits, e indique claramente si hay desbordamiento, justificándolo.

A–B= A+Ca2(B)
Ca2(B)= Ca2(01010001) = 10101111

11111111 (Acarreos)
11011101 (A)
+10101111 (Ca2(B))
10001100 (Resultado. Se descarta el acarreo final)

Los dos últimos bits de acarreo, indicados por el recuadro, son iguales (11), luego NO hay desbordamiento (V=0).

2.- (1,5 puntos) Represente el número -1025,875 en el formato de simple precisión de IEEE754. Detalle todos los pasos realizados y exprese el resultado final en hexadecimal.

1025 = 10000000001
0,875 = 0,111

Normalizamos mantisa:

10000000001,111 = 1,0000000001111 x 2^{10}
Exponente en exceso-127 = 127 + 10 = 137 = 10001001

Codificación:
1 10001001 00000000011110000000000

En hexadecimal:
0xc4803c00

3.- (5,5 puntos) A partir del siguiente código, escrito en ensamblador del MIPS R2000:

```
              .globl __start
              .data 0x10000000
  cad_valor: .ascii "255"
    cad_fin: .asciiz "000"
       base: .half 10
        res: .word 0x12345678

              .text 0x00400000
    __start: la $4, cad_valor
             la $5, cad_fin
             la $6, res
             la $10, base
             lh $10, 0($10)
             addi $9, $0, 0
             addi $11, $0, 1
    repetir: addi $5, $5, -1
             lb $7, 0($5)
             addi $7, $7, -0x30
             mult $7, $11
             mflo $8
             add $9, $9, $8
             beq $5, $4, acabar
             mult $11, $10
             mflo $11
             j repetir
     acabar: sw $9, 0($6)
             .end
```

a) **(1 punto**) 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**. Necesitará saber que el código ascii del carácter ‘0’ es 0x30, el de ‘1’ es 0x31, el del ‘2’ 0x32 y así sucesivamente para las 10 cifras decimales. **Indique claramente los bytes de contenido desconocido mediante un interrogante o guión**.

31 … 24	23 … 16	15 … 8	7 … 0	Dirección
0x30	0x35	0x35	0x32	0x10000000
-	0x00	0x30	0x30	0x10000004
-	-	0x00	0x0A	0x10000008
0x12	0x34	0x56	0x78	0x1000000C

b) **(1 punto)** Indique el contenido en decimal o hexadecimal de los siguientes registros antes de ejecutar por primera vez **la instrucción beq $5, $4,** acabar

Registro	**Contenido (Decimal o Hexadecimal)**
$4	0x10000000
$5	0x10000002
$6	0x1000000C
$7	0x00000005 / 5
$8	0x00000005 / 5
$9	0x00000005 / 5
$10	0x0000000A / 10
$11	0x00000001 / 1

c) **(1 punto)** Indique el contenido (en decimal o hexadecimal) de los siguientes registros **al finalizar la ejecución del programa**

Registro	Contenido (Decimal o Hexadecimal)
$5	0x10000000
$7	0x00000002 / 2
$8	0x000000c8 / 200
$9	0x000000FF / 255
$11	0x00000064 / 100

d) **(0,5 puntos)** Indique solo las zonas de la memoria que se hayan modificado tras la ejecución del programa. Exprese el valor de cada byte en decimal o hexadecimal.

31 … 24	23 … 16	15 … 8	7 … 0	Dirección
0x00	0x00	0x00	0xFF / 255	0x1000000C

e) **(1 punto**) Indique la secuencia de instrucciones por las que el ensamblador del MIPS R2000 traduciría las pseudoinstrucciones:

```
la $4, cad_valor
la $5, cad_fin
```

```
la $4, cad_valor
        lui $4, 0x1000
la $5, cad_fin
        lui $1, 0x1000
        ori $5, $1, 3
```

NOTA: De ser necesario, debe utilizar el registro $1 para los cálculos intermedios.

f) **(1 punto)** Codifique la instrucción `beq $5, $4, acabar.` Indique el resultado en binario y hexadecimal y detalle los pasos realizados.

Instrucción tipo I: CO(6b) rs(5b) rt(5b) desp(16b)

CO= 0x04 (000100) rs= 5 (00101) rt= 4 (00100) desp= +4 (0000000000000100)

Binario: 000100 00101 00100 0000000000000100

Hexadecimal: 0x10a40004

4.- (1 punto) Realice un programa en ensamblador del MIPS R2000 que:

- A partir de la dirección 0x1000000 declare unas variables `fil` y `col`, enteras de 8 bits, inicializadas a los valores decimales:
  - `fil` = 14
  - `col` = 10
- Declare a continuación una variable `car` para contener un carácter, que se inicializará con el carácter "A".
- Declare a partir de la dirección 0x10001000 espacio para albergar 2000 caracteres (bytes), inicializado a ceros.

El programa deberá entonces:

- Leer el contenido de las variables `fil`, `col` y `car` y guardarlo en los registros $2, $3 y $4 respectivamente.
- Calcular y guardar en el registro $10 el resultado de esta expresión:

  $10 = fil * 80 + col + 0x10001000

- Escribir el carácter `car` en la posición de memoria indicada por el registro $10.

Posible solución:

```
          .globl __start
          .data 0x10000000
    fil:  .byte 14
    col:  .byte 10
    car:  .ascii "A"
          .data 0x10001000
  video:  .space 2000

          .text
__start:  la $5, fil
          lb $2, 0($5)
          lb $3, 1($5)
          lb $4, 2($5)
          li $8, 80
          mult $2, $8
          mflo $10
          add $10, $10, $3
          li $9, 0x10001000
          add $10, $10, $9
          sb $4, 0($10)
          .end
```