Fundamentos de Computadores. Examen oportunidad ordinaria. 03/06/2024.

Nome: [redacted]

1. **(1,25 ptos)** Consideremos el formato de representación IEEE 754 para punto flotante de 32 bits. Considerando esta representación responde de manera razonada breve:

   a) **(0,5 ptos)** Indicar la división en campos y el valor decimal (en base 10) del número que se representa como 0xC1400000.

   b) **(0,75 ptos)** ¿Qué es la precisión de un formato de representación? ¿Aumentaría o disminuiría si, partiendo de la representación IEEE754 de 32 bits, aumento el número de bits dedicados al exponente manteniendo los 32 bits totales para la representación?

2. **(1,5 ptos)** Tras ejecutar un programa en dos computadores en arquitecturas diferentes se obtienen las siguientes medidas:

	computador A	computador B
N. Instrucciones	1000 millones	800 millones
Frecuencia de reloj	4 GHz	4 GHz
CPI	1,0	1,2

   Indicar de manera razonada breve:

   a) **(0,5 ptos)** El valor de la medida de rendimiento MIPS en cada computador.

   b) **(0,5 ptos)** El tiempo de ejecución en cada computador.

   c) **(0,5 ptos)** A la vista de los resultados de los apartados anteriores, ¿Cuál de los dos computadores presenta mejores prestaciones y por qué?

3. **(2,25 ptos)** En el MIPS la instrucción 0x8D90FFEC está almacenada en la dirección [0x0040000C] de la memoria de instrucciones y se ejecuta, como las vistas en clase, en un solo ciclo de reloj. En el ciclo anterior a que se ejecute dicha instrucción el estado del banco de registros y de la memoria son los que se muestran a continuación:

BANCO DE REGISTROS

```
$0=00000000 $8=10010000 $16=00000000 $24=00000000
$1=00000000 $9=10010010 $17=00000006 $25=00000000
$2=00000000 $10=00000000 $18=00000000 $26=00000000
$3=00000000 $11=00000000 $19=00000000 $27=00000000
$4=00000003 $12=10010024 $20=00000002 $28=00000000
$5=00000000 $13=00000000 $21=00000000 $29=70000000
$6=00000000 $14=00000008 $22=00000000 $30=00000000
$7=00000000 $15=00000000 $23=00000000 $31=00000000
```

MEMORIA DE DATOS

```
[0x100010000] 0x00000001 0x00000002 0x00000003 0x00000004
[0x100010010] 0x00000005 0x00000006 0x00000007 0x00000008
[0x100010020] 0x00000009 0x00000000
```

Se dispone de la información sobre códigos de operación siguiente:

Instrucción	Código Operación (en decimal)
add	0
sub	0
j	2
beq	4
bne	5
lw	35
sw	43

Indicar, **razonando la respuesta**:

a) **(0,25 ptos)** Formato de la instrucción en binario detallando longitud y significado de cada campo en que se divide la instrucción.

b) **(0,25 ptos)** Indica cómo se escribiría la instrucción en ensamblador.

0.1 c) **(0,25 ptos)** Especifica la ruta de datos sobre la figura.

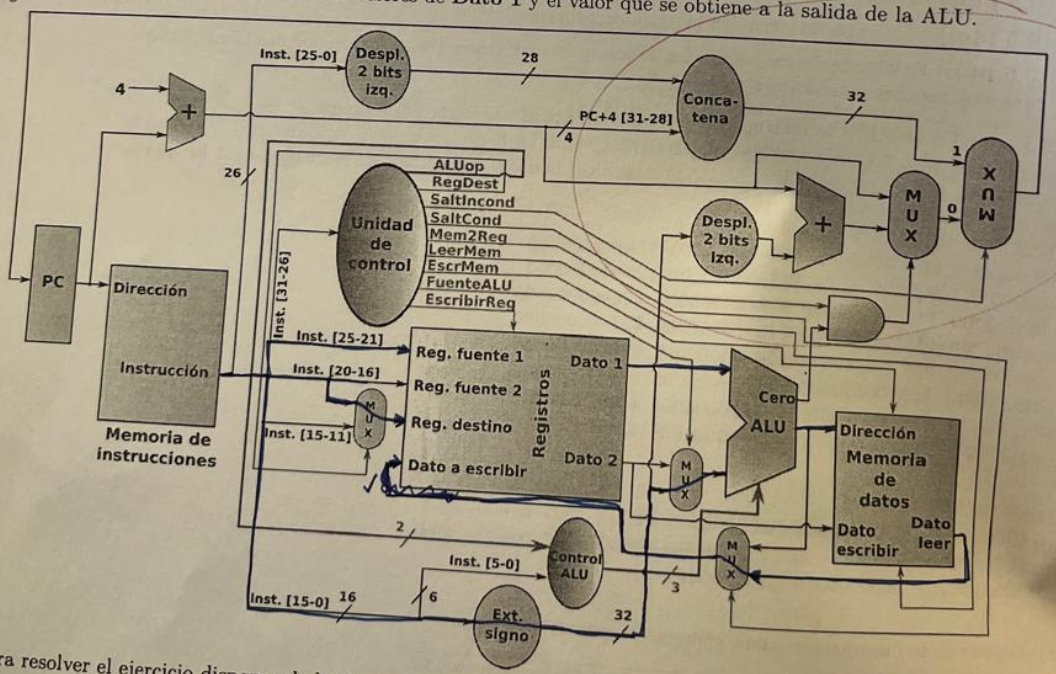
+ d) **(0,5 ptos)** Indica en la tabla que se muestra a continuación los valores que deben tomar todas las señales de control para que se pueda ejecutar la instrucción.

Señal de control	Valor
ALUop	00
RegDest	1 — 0
SaltIcond	0
SaltCond	0
Mem2Reg	1
LeerMem	1
EscrMem	1 — 0
FuenteALU	1
EscribirReg	1

f e) **(0,5 ptos)** Indica en hexadecimal los valores de salidas de los multiplexores del camino de datos cuyas entradas de control se muestran en la siguiente tabla:

Señal de control multiplexor	Salida del multiplexor (hexadecimal)
regDest	
SaltIncond	
Mem2Reg	
FuenteALU	

0.25 f) **(0,5 ptos)** En hexadecimal los valores de **Dato 1** y el valor que se obtiene a la salida de la ALU.



Para resolver el ejercicio dispones de la siguiente información adicional:

ALUop	funct	Acción de la ALU	Control la ALU
00	XXXXXX	suma	010
01	XXXXXX	resta	110
10	100000	suma	010
10	100010	resta	110
10	100100	and	000
10	100101	or	001

4. **(3 ptos)** Considerando el repertorio de instrucciones del MIPS de 32 bits estudiado en clase, y dado el siguiente código:

0/6

```
.data
num: .word 5
datos: .word 5, 4, 3, 2, 1

.text
.globl main
main:
la $a0, datos   $a0 = [Mem] → 5,4,3,2,1 (datos)
la $t0, num     $t0 = [Mem] → 5 (num)
lw $a1, 0($t0)  $a1 = 5

jal etiqueta1

addi $v0, $0, 10   $v0 = $0 + 10
syscall → sale del programa

etiqueta1:
addi $t0, $a0, 0  $t0 = $a0
addi $t1, $0, 0   $t1 = $0

etiqueta2:
lw $t2, 0($t0)  $t2 = 4
addi $a0, $t2, 0  $a0 = $t2 *

addi $v0, $0, 1  $v0 = $0 + 1
syscall → "10 salir del programa, 1 imprimir int"

addi $t0, $t0, 4  $t0 = ($t0 + 4 = 8)  Mem → 4 (datos)
addi $t1, $t1, 1  $t1 = 1

bne $t1, $a1, etiqueta2  $t1 ≠ $a1

jr $ra
```

a) **(0,75 ptos)** Indica para qué se utilizan los registros $a0 y $a1. Indicar cuantos bucles hay en el código y cuantas iteraciones se ejecutan de cada bucle. Si el programa imprime en pantalla, indicar qué se imprime en pantalla.

b) **(0,5 ptos)** Sabiendo que la codificación en hexadecimal de la instrucción jr es 0x03E00008, escribe su codificación en binario especificando el significado de cada campo.

c) **(0,75 ptos)** Las instrucciones del código anterior se almacenan en posiciones contiguas de memoria. Calcula la codificación en binario de la instrucción bne sabiendo que: se encuentra almacenada en la posición de memoria [0x00400040], que su código de operación es el 5, el registro $t1 es el $9 y el $a1 es el $5. Especifica el significado de cada campo.

d) **(1 pto)** ¿Se cumple en el programa anterior con el convenio de uso de registros del MIPS vista en clase? Indica brevemente porqué. Si no se cumple, modifica el código para que así sea. Debes usar la pila.

5. **(2 ptos)** Consideremos una caché de un sistema BigEndian, con ISA MIPS de 32 bits, es decir, con direcciones y palabras de 32 bits, y direccionamiento de memoria a nivel de byte. La caché es de tamaño 64 bytes y líneas de 4 palabras, asociativa por conjuntos de 2 vías con algoritmo de reemplazo LRU. Consideramos que la caché está vacía y a continuación la CPU emite la siguiente secuencia de direcciones de acceso a memoria:
0xABCD0003, 0xABCD0000, 0x0AA00000, 0xABCD0FFD, 0xABC00FC5, 0xABCE0FC5, 0x12340CC7, 0xABCD0FF9.

0'75

- **(1,25 ptos)** Para cada una de las direcciones anteriores indicar la división en campos de la dirección y el significado de cada campo justificando si se trata de un acierto o de un fallo y en qué lugar de la caché se almacenará cada bloque de datos. Indicar cuándo se produce reemplazo. Indicar el contenido final de la caché.
- **(0,25 ptos)** Indicar cuantos fallos de cada tipo se producen en el apartado anterior y cuantos reemplazos justificando la respuesta.
- **(0,5 ptos)** Para el apartado anterior indicar el rango de direcciones de memoria en hexadecimal cuyo contenido se mueve a la caché cuando la CPU realiza el acceso a la dirección 0xABCD0003.