FUNDAMENTOS DE COMPUTADORES (20 de mayo de 2015)

1ª (2p)	2ª (2p)	3ª (2.5p)	4ª (2.5p)	5ª (1p)	*Nota (10p)*

NOMBRE: _____

1. Si tenemos un sistema que utiliza IEEE 754 de precisión simple y redondeo al más próximo, indica lo que imprimirá el siguiente código C (`uint8_t` indica un entero sin signo de 8 bits):

```
#include<stdio.h>
#include<stdint.h>
void main() {
      float f = -1023.4;
      uint8_t * u = (uint8_t *) &f;
      printf("%x-%x", *u, *(u+1));
}
```

   Suponiendo:

   a) Un sistema big-endian.

   b) Un sistema little-endian.

Solución: pasamos a binario:

1023 = `1111111111`

0.4 * 2 = **0**.8; 0.8*2=**1**.6; 0.6*2=**1**.2; 0.2*2=**0**.4; 0.4 * 2 = **0**.8; 0.8*2=**1**.6; 0.6*2=**1**.2; 0.2*2=**0**.4;

Es decir:

0.4 = `0.0110011001100110011001100110011…`

Por lo tanto:

1023.4 = `1111111111.0110011001100110011…` =>

`1023.4 = 1.11111111101100110011001`*`100110011`*`...*2`^{9}

Redondeando para dejar 23 cifras decimales (redondeo al más próximo):

`1023.4 = 1.11111111101100110011010 * 2`^{9}

El exponente en exceso = 127+9 = 136 = `10001000`

Por lo tanto, el número IEEE 754 SP será:

`1 10001000 11111111101100110011010`

En hexadecimal:

`1100 0100 0111 1111 1101 1001 1001 1010 = C4 7F D9 9A`

Por lo tanto, en big-endian escribirá C4-7F y en little endian 9A-D9

2. Conseguimos cambiar el código de un programa de forma que paralelizamos el 85% de las instrucciones, consiguiendo que se ejecuten (las instrucciones paralelizadas) N veces más rápido en un sistema multicore con N cores. Indica cuántos cores necesitaremos para conseguir que el código completo sea 4 veces más rápido.

Solución: aplicar la ley de Amdahl. Si el tiempo original es T, el tiempo mejorado es T'=T/4 =>

T'/T = ¼ = F/M + (1-F)

F es la fracción mejorada = 0,85, y M es lo que la mejoramos, que en este caso es igual al número de cores:

0,25 = 0,85/M + (1-0,85) => 0,1 = 0,85/M => M = 8,5 = 9 cores

3. Dado el siguiente código MIPS:

Dirección	Código instrucción	Instrucción
⋮	⋮	⋮
[0x00400010]	0x00094020	add ...
[0x00400014]	0x1128ffff	beq ...
⋮	⋮	⋮

Responde a las siguientes preguntas:

a) ¿Es posible saber si el salto se toma o no?

b) En el caso de que sea posible, ¿a qué dirección está saltando? ¿qué se ejecuta a continuación?

Solución: La instrucción add es de tipo R y tiene el siguiente formato:

000000 00000 01001 01000 00000 100000

por lo tanto rs=$0, rt=$9 y rd=$8, es decir, es la instrucción add $8, $0, $9. Como $0==0, después de la suma $8==$9

La instrucción beq es de tipo I y tiene el siguiente formato:

000100 01001 01000 1111111111111111

es decir, es beq $9, $8, destino, por lo que se salta.

La dirección destino del salto se obtiene a partir del dato, con el signo extendido y multiplicado * 4:

11111111111111111111111111111100 = -4

Por lo tanto, la dirección destino de salto es [PC+4] – 4 = PC, con lo que se ejecuta de nuevo el beq => entramos en un bucle infinito.

4. Supón un computador de 32 bits con una memoria cache de 64 KiB, asociativa por conjuntos de 8 vías. El tamaño de la línea es de 16 palabras y usa direccionamiento a nivel de byte. Indica:

a) El formato de una dirección desde el punto de vista de la caché. ¿Cuántos conjuntos tiene la cache?

b) El tamaño en bits del directorio caché, incluyendo el bit de validez. ¿Cuántos comparadores necesitaremos para determinar si hay acierto o fallo?

c) Considerando la siguiente secuencia de direcciones 0x0007B042, 0x0007D042, 0x0007D074, indica cuales ocasionan un fallo caché y por qué, suponiendo la cache inicialmente vacía. Indica también a qué conjunto va cada uno de los datos y a qué línea en el conjunto.

Solución:

a) N.º conjuntos = Tamaño cache/tamaño conjuntos

1 conjunto = 8 vías = 8 líneas. 1 línea = 16 palabras * 4 bytes =64 bytes => 1 conjunto = 8 * 64 = 512 bytes

N.º conjuntos = 64*1024/512 = 128 conjuntos = 2^7 conjuntos

Por lo tanto la dirección será

19	7	4	2
Etiqueta	Conjunto	Palabra/ línea	Byte/ palabra

b) Para calcular el tamaño del directorio necesitamos el número total de líneas:

N Líneas = N Ctos * lineas_cto = 128 * 8 = 1024 líneas

Por cada línea se guardan 20 bits (19 etiqueta + 1 validez) =>

Tamaño directorio = 1024 líneas * 20 bits = 20480 bits

Se necesita 1 comparador por vía => 8 comparadores

c)

0x0007B042 = `00000000000000111101 1000001 000010` => fallo, va al conjunto 65, a cualquier línea

0x0007D042 = `00000000000000111110 1000001 000010` => fallo, va al conjunto 65, a cualquier línea libre

0x0007D074 = `00000000000000111110 1000001 110100` => acierto, está en el conjunto 65

5. Responde brevemente a las siguientes preguntas

a) Describe los componentes del tiempo de acceso a un disco duro magnético.

b) Ventajas e inconvenientes de la entrada/salida por interrupciones y por DMA.

1. (2 puntos) Indica que valor representan cada una de las siguientes secuencias de bits según el estándar IEEE754 para la representación de números en punto flotante de simple precisión.

1	00000000	10000000000000000000000
1	11111111	00000000000000000000000
0	00011110	01000000000000000000000
1	10000001	10000000000000000000000

   - Explica por qué en una operación de suma la normalización (en caso de ser necesaria) se realiza desplazando el resultado a la izquierda y en una resta (en caso de ser necesaria) se realiza desplazando el resultado a la derecha. Ilustra ambos casos con un ejemplo.
   - Indica cuantos bits habría que añadir y donde si queremos modificar este formato para que el menor número normalizado positivo representable sea el 2^{-510}.

2. (2 puntos) Dada la secuencia de instrucciones del procesador MIPS de 32 bits:
   lw $4, -8 ($3)
   addi $2, $2
   bne $2, $0, salto
   j salto 2
   Teniendo en cuenta que las instrucciones del MIPS tienen 6 bits para el campo de operación y 32 registros, y suponiendo que inicialmente el contenido del registro $2 es 5, el del registro $3 es 0x10011000 y que la primera instrucción se encuentra en la dirección [0x00400010].
   a) Calcula a qué dirección de memoria está accediendo la instrucción lw y cuál es su codificación (haz todas las cuentas y muestra el resultado en hexadecimal, supón que el código de operación en esta instrucción es 100011.
   b) Si la codificación de la instrucción bne es 0x1440FFFD. Calcula cual es la dirección a la que se realiza el salto (haz todas las cuentas y muestra el resultado en hexadecimal) ¿Es un salto hacia adelante o hacia atrás?
   c) Indica las direcciones mínima y máxima a las que se puede acceder con la instrucción j.

3. (2 puntos) La instrucción jal realiza un salto incondicional, para ejecutar una subrutina y guarda la dirección de retorno en $ra=31. El único argumento de la instrucción es el nº de palabras que se saltarán, por ejemplo jal 2000.
   Considerando los caminos de datos y control vistos en clase para el MIPS (figura) responde a los siguientes apartados razonando.
   a) Añade el hardware necesario para implementar la instrucción.
   b) Especifica el formato que se usará para codificar la instrucción, los campos en que se dividirá la instrucción, así como el tamaño y el significado de cada campo.
   c) Especifica sobre la figura la ruta de datos que estará activa cuando se ejecute la instrucción.

4. (2 puntos) Dado un ordenador con direcciones de K bits, una memoria direccionable por B bytes, una caché de S bytes y asociativa por conjuntos de A vías, asumiendo que K, S, B y A son potencias de 2. Determina en función de K, S, B, A.
    a) El nº de conjuntos de la caché.
       Nº de líneas = tam caché / tam línea = S/B
       Nº conjuntos = Nº líneas / vías (líneas por conjunto)= (S/B)/A
    b) El nº de bits de la etiqueta de la dirección.
       Etiqueta = K – log2(A)-log2 (B)
    c) El nº de bits de la caché.
       S*8
    d) Considerando S = 256 KB, B = 16, y A = 128, determina los valores que deben tener X, Y y Z para que las dos direcciones F00101A7 y 0B3XYZAW vayan al mismo conjunto.
       Log2 (128) = 10 bits para el conjunto
       Tienen k coincidir los bits del conjunto en ambas direcciones para que pertenezcan al mismo conjunto.
5. (2 puntos) Dado un computador con una CPU que ejecuta 2000 millones de instrucciones por segundo y un bus PCI con una velocidad de transferencia de 1Gb/s.
    a) Calcula cual es la velocidad de transferencia máxima posible con una E/S con interrupciones considerando que el procesamiento de la interrupción necesita 200 instrucciones para la transferencia de 32 bits de datos.
    b) Calcula cual es la velocidad de transferencia máxima posible con una E/S con DMA, considerando que se necesitan 2000 instrucciones para el inicio de la transferencia y otras 1000 para la finalización y se realizan transferencias de bloques de 2KB de datos.
    c) Si en el próximo año se consiguiese duplicar la frecuencia de la CPU pero no la del BUS, calcula cual sería el aumento del rendimiento en los casos de los apartados a y b.

1- Usando el formato IEEE754 de simple precisión (2puntos)
   a) Indica como se representan los números -0.765625 y 0.75
   b) Realiza en binario la suma de ambos números indicando los pasos que se utilizan.
   c) Realiza en binario la multiplicación de ambos números indicando los pasos.

2- Se ha modificado el repertorio de instrucciones de MIPS de 32 bits de acuerdo a las siguientes características: 128 operaciones, de 32 bits. Se dispone de un sistema de 32 registros y se utilizan los modos de direccionamiento del MIPS vistos en clase.

   Dado el siguiente fragmento de código:

```
Addi $4,$5,4     #$4+$5+4
Lw $6,4($7)      #$6=memoria [4+$7]
Add $4,$4,$6     #$4=$4+$6
Beq $4,$3,salto#if($4==$3) PC=salto
J 125
```

   Y teniendo en cuenta que la dirección de la instrucción addi es 0x070ac310, calcula las direcciones de memoria menor y mayor a las que se puede referenciar las instrucciones: lw, beq y j de dicho código. Especifica los resultados en Hex. Razona respuestas. (2.5 puntos)

3- A partir de la implementación conocida vista en clase y retocando el repertorio: (2 puntos)

   swri -100,$s1,$s2

   a) Añadir el hardware necesario para implementar la instrucción “swri” que almacena una constante en la dirección de memoria obtenida a partir de la suma de dos registros.
   b) Especificar ruta de datos en el dibujo.
   c) Modificar el control para implementar la nueva función.

4- (2 puntos) Considera un procesador con direcciones físicas de 32 bits y caché de 32 bytes organizada asociativamente en 1024 conjuntos, con una palabra de 1byte por línea y direccionable por byte. Indica razonando:

   a) ¿Cuántas líneas puede almacenar cada conjunto? ¿Nº de vías de caché?
   b) ¿A qué conjunto se asignarán las direcciones (hex) f002167a y 17081000?
   c) ¿Cuál es la etiqueta asociada a cada una de los dos datos en la direcciones del apartado B?

5- (1.5 puntos)
   a) ¿Por qué E/S con interrupciones es más lenta que con DMA?
   b)¿Para qué se usan los niveles de prioridad en interrupciones?

c) Diferencias entre buses síncronos y asíncronos
d) ¿Cual es la necesidad de que existan los árbitros de buses?

Examen final Fundamentos de Computadores 1ª convocatoria

Grado en Ingeniería Informática – Curso 2020/21

Ejercicio 1. (2 puntos) Consideremos un formato similar a IEEE 754 para punto flotante de 32 bits y para el cual el primer bit es de signo, a continuación, tenemos 7 bits de exponente y 24 bits de mantisa normalizada. Se mantiene el esquema del IEEE 754 para representar el $\pm\infty$, los NaN y los números no normalizados. Indica para esta representación:

a) (0.5 puntos) El valor en base 10 del número positivo más pequeño mayor que 0 que se puede representar con esta representación incluyendo a los números no normalizados. Indica también su representación en 32 bits indicando la división en campos.

b) (0.5 puntos) Valor mínimo positivo (mayor que 0) normalizado en decimal y en representación de 32 bits.

c) (0.5 puntos) Indicar el valor decimal del número que se representa como 0xFC200000.

d) (0.5 puntos) La representación en 32 bits del número decimal $-2500{,}4$ e indicar su valor en hexadecimal.

Ejercicio 2. (2 puntos) Responde brevemente a las siguientes cuestiones, razonando la respuesta:

a) (0.5 puntos) Calcula el AMAT (tiempo medio de acceso por instrucción) para un procesador con un tiempo de ciclo de reloj de 1 ns, una penalización por fallo de lectura o de escritura de 20 ciclos de reloj, una tasa de fallos de $0{,}05$ fallos por instrucción y un tiempo de acceso a la caché (sea el acceso un acierto o un fallo) de 1 ciclo de reloj.

b) (0.5 puntos) ¿Es cierto que la potencia consumida por un procesador es directamente proporcional al voltaje de alimentación e inversamente proporcional a la frecuencia de trabajo del procesador?

c) (0.5 puntos) Cuando un programa se compila para el computador A requiere 1000 millones de instrucciones y el CPI en dicho computador es de 1. Calcular el valor de MIPS para este computador considerando que tiene una frecuencia de reloj de 4 GHz.

d) (0.5 puntos) Indica muy brevemente las principales ventajas que puede tener una gestión del sistema de entrada/salida por interrupciones frente a otras opciones.

Ejercicio 3. (2 puntos) Considerando el repertorio de instrucciones del MIPS de 32 bits y dado el siguiente fragmento de código:

```
lui $8, 0x1001
ori $8, $8, 0x0040
instrucción_b
jal resta_vector
add $16, $17, $18
```

a) (0.25 puntos) ¿Qué funciones realizan las 2 primeras instrucciones?

b) (0.25 puntos) ¿Por qué pseudo-instrucción podemos sustituir las 2 primeras instrucciones? Indica su codificación equivalente en lenguaje ensamblador suponiendo que la dirección referenciada equivale a la etiqueta vector.

c) (0.5 puntos) Queremos cargar en el registro $16 la palabra almacenada en la dirección de memoria 0x10010034. Sustituye *instrucción_b* por la instrucción que nos permite realizar esa operación. Indica también su codificación en binario indicando cada uno de sus campos sabiendo que su código de operación es 100011.

d) (0.5 puntos) ¿Qué acciones realiza la instrucción *jal resta_vector*?

e) (0.5 puntos) La dirección de memoria en la que está almacenada la instrucción jal resta_vector es 0x00400090 y su codificación en hexadecimal es 0x0c10003e. ¿A qué dirección de memoria realiza un salto? ¿De qué tipo de direccionamiento se trata?

Ejercicio 4. (2 puntos) Consideremos una caché de asignación directa 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. Tenemos una caché de asignación directa de 32 bytes y líneas de una palabra. Consideramos que la caché está vacía cuando la CPU emite la siguiente secuencia de direcciones de acceso a memoria:

$$0x00000003, 0x00000002, 0x000000E2, 0x000000A2, 0x000000BE, 0x000000BA$$

a) (1 punto) 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. Indicar el número de reemplazos que se producen. Indicar cual es el contenido final de la caché tras producirse la lista de accesos anterior.

b) (0.5 puntos) Si la caché fuese asociativa por conjuntos de 2 vías, ¿cómo se dividiría cada dirección en campos indicando su significado? Mostrar como ejemplo la división en campos de la dirección 0x000000BA.

c) (0.5 puntos) Indicar el número y tamaño de los comparadores que harían falta para cada una de las dos cachés consideradas en el ejercicio justificando la respuesta.

Ejercicio 5. (2 puntos) Se desea añadir la instrucción de tipo de acceso a memoria addsw al procesador tipo MIPS de la figura (implementación vista en clase) que realiza la siguiente función que se ejecuta en un solo ciclo de reloj:

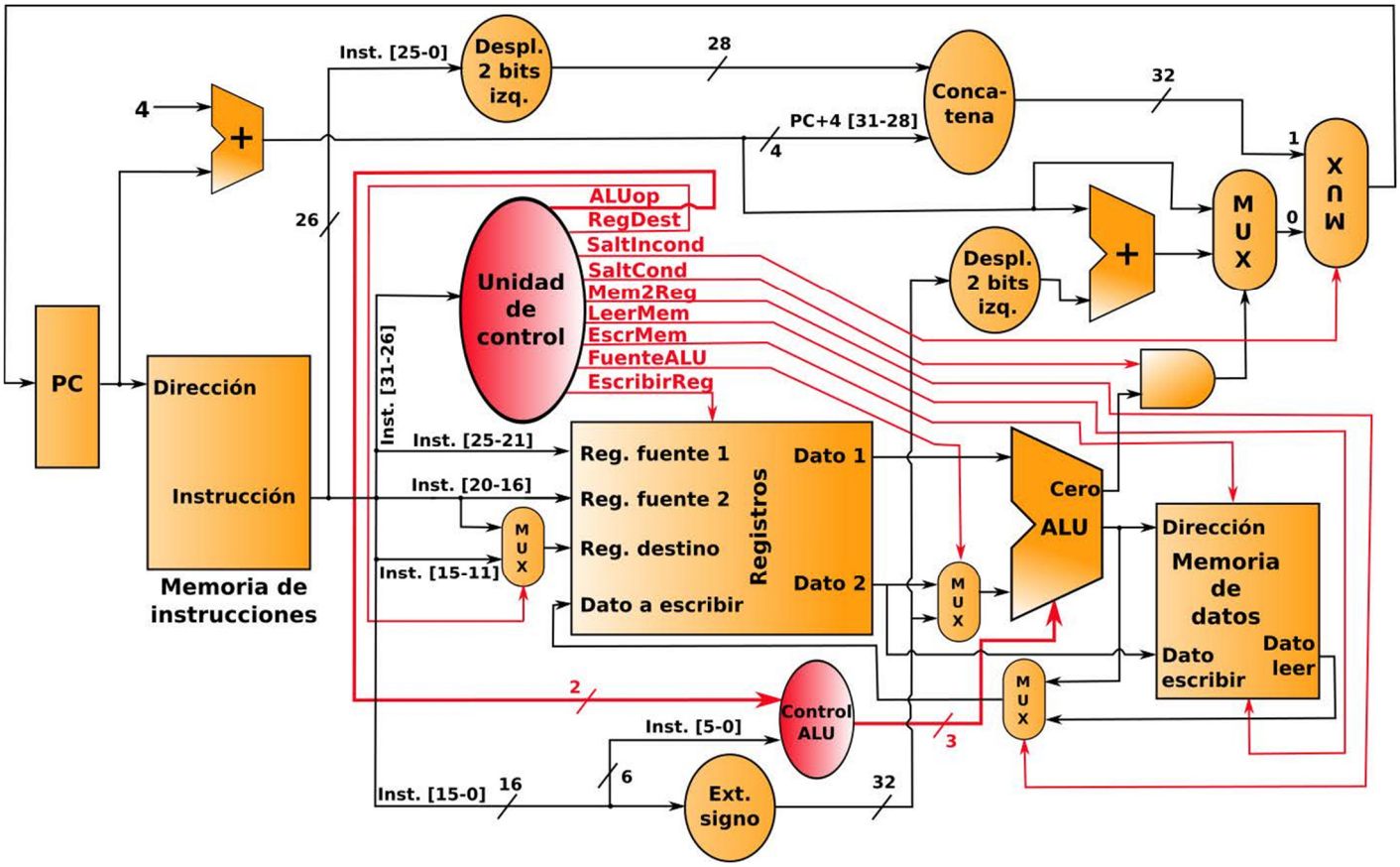
```
addsw $t1, cte($t2);   $t1 → Mem[($t2+$t1) + ext.signo.cte]
```

Esta instrucción almacena el contenido de un registro ($t1 en el ejemplo) en memoria, en la posición obtenida como la suma del contenido de dos registros ($t2 y $t1 en el ejemplo) y una constante a la que se le extiende el signo.
Indica, **razonando la respuesta**:

a) (0.5 puntos) Formato de la nueva instrucción detallando la longitud y el significado de cada campo en que se divide. Indicar de qué tipo es la instrucción atendiendo a su división en campos.

b) (0.75 puntos) Especifica la ruta de datos sobre el dibujo indicando, en caso de ser necesario, qué hardware adicional se requeriría, dónde sería necesario colocarlo y cómo se conectaría en la figura y para qué.

c) (0.75 puntos) 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 nueva instrucción, añadiendo nuevas filas si es necesario añadir nuevas señales.



ALUop	Funct	Acción de la ALU	Control de 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

Señal de control	Valor
ALUop	
RegDest	
SaltIncond	
SaltCond	
Mem2Reg	
LeerMem	
EscrMem	
FuenteALU	
EscribirReg	