Arquitectura e Ingeniería de Computadores - 3er Curso. Sol. del Examen recup. (1er parcial) 1/2/2023

Responde cada pregunta en una hoja distinta. Tiempo disponible: 1h30m

1. **(2.5 puntos)** En una fábrica de computadores el proceso de fabricación de un supercomputador vectorial está distribuido en 5 etapas: **Mecanizado de la placa base** (2 horas), **Soldadura de componentes** (no hay información), **Cableado e instalación en la carcasa** (9 horas), **Serigrafiado de la carcasa** (4 horas) e **Instalación del sistema operativo** (resto del tiempo). El tiempo necesario para la fabricación del supercomputador a través de las 5 etapas es de 20h.

   Con objeto de acelerar el proceso de fabricación, se han propuesto diferentes mejoras:

   A. Sustituir el técnico de soldadura de la etapa **Soldadura de componentes** por un soldador robot. Se sabe que éste suelda 4 veces más rápido que un técnico humano y se ha observado que el tiempo de fabricación se reduce a 17h.
   B. Emplear una pintura de secado rápido en el serigrafiado, esto reduciría a la mitad el tiempo de permanencia en la etapa de **Serigrafiado de la carcasa**.
   C. Emplear a una persona más en la etapa de **Cableado e instalación en la carcasa** lo que permitiría reducir el tiempo de esta etapa a 5 horas.

   Responder a las siguientes cuestiones:

   a) ¿Qué mejora global ofrece el proceso de fabricación del supercomputador al sustituir el técnico de soldadura por un robot (mejora A)? ¿Qué fracción de tiempo del proceso de fabricación empleaba el técnico de soldadura?

   b) Si finalmente se descarta la opción del soldador robot por temas sindicales, ¿cuál de las otras dos opciones propuestas, B o C, ofrecerá una mayor mejora global?

   c) Si implementamos las mejoras B y C simultáneamente, y éstas representan un incremento del coste original de un 33 %, ¿sería interesante la propuesta desde el punto de vista coste/prestaciones? Justifica la respuesta.

   Solución:

   a) S'=20/17=1.1765 $\rightarrow$ 17.65 % ;; S=4 ;; F=(S(1-S'))/(S'(1-S))=0.2 $\rightarrow$ 20 %

   b) E1: 2/20=0.1 ; E2: 4/20=0.20 ; E3: 9/20=0.45 ; E4: 4/20=0.2 ; E5: 1/20=0.05
   Opción B: S=2 ; S'=1/(0.8+0.2/2) = 1.1111
   Opción C: S=9/5=1.8; S'=1/(0.55+0.45/1.8) = 1.25 ;; Ésta ofrecerá una mayor mejora, un 25 %

   c) Mejoras B y C juntas: S'=1/(0.2/2+0.45/1.8+0.35) = 1.4286
   Desde el punto de vista de costes/prestaciones sería interesante ya que con un incremento del coste del 33 % obtendríamos una mejora del 42.86 %

□

2. **(2.5 puntos)** Disponemos de un sistema con un procesador MIPS segmentado que funciona a 1 GHz. Los riesgos de datos se resuelven mediante cortocircuitos, insertando ciclos de parada en la fase ID en caso necesario, mientras que los de control se resuelven mediante *predict-not-taken*, calculando la condición, la dirección y escribiendo el PC en la tercera etapa del ciclo de instrucción.

   Las estadísticas de los tipos de instrucciones al ejecutar un programa P son las siguientes:

Operación	%
mul	10 %
add	20 %
otras ALU	20 %
Carga	25 %
Almacenamiento	15 %
Saltos	10 %

El programa P ejecuta 100×10^6 instrucciones. Durante su ejecución, un $30\,\%$ de las instrucciones de Carga provoca que una instrucción posterior inserte 2 ciclos de parada para resolver los riesgos de datos, mientras que un $20\,\%$ de las instrucciones de Almacenamiento necesitan que se inserte 1 ciclo de parada para resolver los riesgos de datos con una instrucción previa. Las estadísticas obtenidas también indican que el $70\,\%$ de los saltos son efectivos.

Se pretende evaluar la conveniencia de añadir unas nuevas instrucciones `madd` que combinan en una única instrucción una multiplicación y una suma. Las nuevas instrucciones permitirían al compilador realizar la siguiente sustitución:

Secuencia

```
mul r4,r1,r2
add r4,r4,r3     -->     madd r4,r1,r2,r3
```

El cambio se puede hacer en el $80\,\%$ de las instrucciones `mul`. Sin embargo, dado que hay que añadir nuevos códigos de operación, la frecuencia de reloj se reduce a 0.9 GHz.

Se solicita, justificando la respuesta con el máximo detalle:

a) Calcular el CPI y el tiempo de ejecución (en segundos) en el MIPS original para el programa P.

b) Calcular el nuevo número de instrucciones que ejecutará el programa P si se utilizan las nuevas instrucciones `madd`.

c) Calcular el nuevo CPI del programa P.

d) Calcular el tiempo de ejecución (en segundos) del programa P al utilizar las nuevas instrucciones `madd`. ¿Es interesante la modificación propuesta?

Solución:

a) Calcular el CPI y el tiempo de ejecución (en segundos) en el MIPS original para el programa P.

$$CPI = 1 + c.p.datos + c.p.control$$

$$CPI = 1 + \underbrace{0{,}25 \times 0{,}3 \times 2}_{\text{c.p. datos Load}} + \underbrace{0{,}15 \times 0{,}2 \times 1}_{\text{c.p. datos Store}} + \underbrace{0{,}1 \times 0{,}7 \times 2}_{\text{c.p. control pnt}}$$

$$CPI = 1 + 0{,}15 + 0{,}03 + 0{,}14 = 1{,}32 \text{ ciclos/instr}$$

Siendo $I = 100 \times 10^6$ instr y $f_{cpu} = 1$ GHz $\rightarrow T = 1$ ns/ciclo:

$$T_{ej} = I \times CPI \times T = 100 \times 10^6 \times 1{,}32 \times 1 \text{ ns} = 132 \text{ ms}$$

b) Calcular el nuevo número de instrucciones que ejecutará el programa P si se utilizan las nuevas instrucciones `madd`.

$$I' = I \times (1 - \underbrace{0{,}1 \times 0{,}8 \times 2}_{\text{mul, add}} + \underbrace{0{,}1 \times 0{,}8 \times 1}_{\text{madd}} = I \times (1 - 0{,}16 + 0{,}08) = 0{,}92 \times I$$

c) Calcular el nuevo CPI del programa P.

$$CPI = 1 + c.p.datos + c.p.control$$

$$CPI' = 1 + \frac{\overbrace{0{,}25 \times 0{,}3 \times 2}^{\text{c.p. datos Load}} + \overbrace{0{,}15 \times 0{,}2 \times 1}^{\text{c.p. datos Store}} + \overbrace{0{,}1 \times 0{,}7 \times 2}^{\text{c.p. control pnt}}}{0{,}92}$$

$$CPI' = 1 + \frac{\overbrace{0{,}15}^{\text{c.p. datos Load}} + \overbrace{0{,}03}^{\text{c.p. datos Store}} + \overbrace{0{,}14}^{\text{c.p. control pnt}}}{0{,}92} = 1 + \frac{0{,}32}{0{,}92} = 1 + 0{,}3478 = 1{,}3478$$

d) Calcular el tiempo de ejecución (en segundos) del programa P al utilizar las nuevas instrucciones de control de bucles. ¿Es interesante la modificación propuesta?

$$T'_{ej} = I' \times CPI' \times T = 0{,}92 \times 100 \times 10^6 \times 1{,}3478 \times 1{,}11 \text{ ns} = 137{,}78 \text{ ms}$$

No es interesante la modificación propuesta, pues aumenta el tiempo de ejecución. El sistema original es mejor en un factor $\frac{137{,}78}{132{,}00}$ =1.04 veces.

□

3. **(2.5 puntos)** Un procesador compatible binario con el MIPS64 ejecuta el siguiente bucle que calcula $\vec{z} = A\vec{x}/B\vec{y}$:

```
 1 loop: l.d f0,x(r10)
 2       l.d f1,y(r11)
 3       mul.d f4,f2,f0   ;F2 contiene A.
 4       mul.d f5,f3,f1   ;F3 contiene B.
 5       div.d f6,f4,f5
 6       s.d f6,z(r12)
 7       daddi r10,r10,8
 8       daddi r11,r11,8
 9       daddi r12,r12,8
10       daddi r14,r14,-1
11       bnez r14,loop
12       <sgte1>
13       <sgte2>
14       <sgte3>
```

El procesador cuenta con dos bancos de registros para almacenar datos enteros y de coma flotante. Además, para la ejecución de las operaciones en coma flotante, dispone de los siguientes operadores multiciclo:

- Un sumador segmentado con $T_{ev} = 2$ e $IR = 1$, con etapas denominadas A1, A2.
- Un multiplicador segmentado con $T_{ev} = 3$ e $IR = 1$, con etapas denominadas M1, M2, etc.
- Un divisor no segmentado con $T_{ev} = 3$ e $IR = 1/3$, con etapas denominadas D1, D2, etc.

Las restantes instrucciones se ejecutan utilizando el pipeline clásico de 5 etapas (IF,ID,EX,ME,WB). Los riesgos de datos se resuelven con cortocircuitos e insertando ciclos de parada cuando es necesario y los riesgos de control se resuelven mediante *predict-not-taken*. En la figura 1 se puede apreciar el detalle de como se resuelven los riesgos de control.

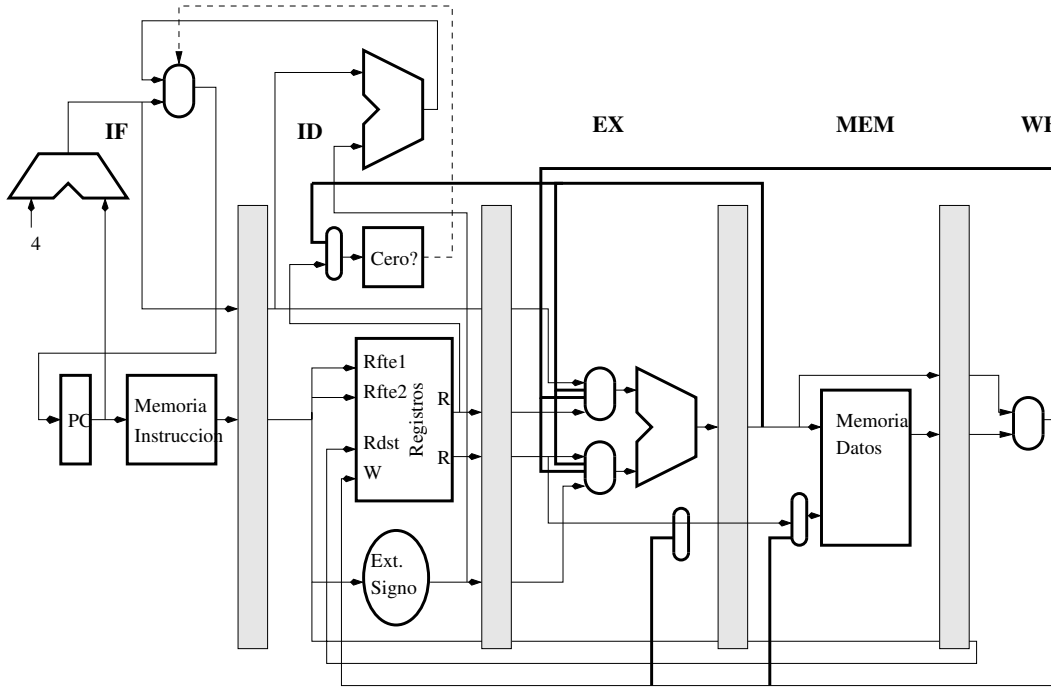

Figura 1: Pipeline de 5 etapas

Al ejecutar dicho código se obtiene el siguiente diagrama instrucciones-tiempo perteneciente a la primera iteración del bucle:

```
Num.      Instrucción          1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18
 1  loop: l.d f0,x(r10)        IF ID EX ME WB
 2        l.d f1,y(r11)           IF ID EX ME WB
 3        mul.d f4,f2,f0;            IF ID M1 M2 M3 WB
 4        mul.d f5,f3,f1;               IF ID M1 M2 M3 WB
 5        div.d f6,f4,f5                   IF id id ID D1 D2 D3 WB
 6        s.d f6,z(r12)                       if if IF id ID EX ME WB
 7        daddi r10,r10,8                              if IF ID EX ME WB
 8        daddi r11,r11,8                                 IF ID EX ME WB
 9        daddi r12,r12,8                                    IF ID EX ME WB
10        daddi r14,r14,-1                                      IF ID EX ME WB
11        bnez r14,loop                                            ...
```

Se pide **(responda en la hoja de respuestas adjunta)**:

a) Para cada riesgo de datos existente en el diagrama proporcionado, rellene los campos en la tabla proporcionada en la hoja de respuestas.

b) Respecto al riesgo de control, y basándose en la figura 1, responda a las siguientes preguntas:

1) Indique en qué etapa se calcula la dirección destino, en qué etapa se evalua la condición, en qué etapa se escribe el PC, cuál es la latencia del salto y cuál es la penalización en caso de fallo de predicción.
2) Complete el diagrama hasta ejecutar la busqueda de la instrucción `loop: l.d f0,x(r10)` de la siguiente iteración.
3) Respecto al riesgo de datos existente en el registro `r14`, indique con detalle como se resuelve dicho riesgo. En caso de ser necesario aplicar un cortocircuito, remárquelo en la figura de la hoja de respuestas. **Justifique la respuesta**.

c) ¿Como se podrían reordenar las instrucciones para reducir los ciclos de parada? ¿Es posible eliminarlos todos? **Justifique la respuesta**.

Solución:

a) Para cada riesgo de datos existente en el diagrama proporcionado, rellene la siguiente tabla:

Registro	Instr. Productora	Instr. Consumidora	Ciclos Parada	Cortocircuito Empleado
f0	`l.d` **`f0`**`,x(r10)`	`mul.d f4,f2,`**`f0`**	0	WBaM1
f1	`l.d` **`f1`**`,y(r11)`	`mul.d f5,f3,`**`f1`**	0	WBaM1
f4	`mul.d` **`f4`**`,f2,f0`	`div.d f6,`**`f4`**`,f5`	2	Sin Cortoc.
f5	`mul.d` **`f5`**`,f3,f1`	`div.d f6,f4,`**`f5`**	2	WBaD1
f6	`div.d` **`f6`**`,f4,f5`	`s.d` **`f6`**`,z(r12)`	1	WBaME

b) Respecto al riesgo de control, y basandose en la figura 1, responda a las siguientes preguntas:

1) Indique en que etapa

se calcula la dirección destino : `ID`

se evalua la condición : `ID`

se escribe el PC : `ID`

latencia del salto : 1

penalización en caso de fallo de predicción : 1 ciclo

2) Complete el diagrama hasta ejecutar la busqueda de la instrucción `loop: l.d f0,x(r10)` de la siguiente iteración.

```
Instrucción               12 13 14 15 16 17 18 19 20 21 22
      daddi r12,r12,8     IF ID EX ME WB
      daddi r14,r14,-1       IF ID EX ME WB
      bnez r14,loop             IF id ID EX ME WB
      <sgte1>                      if IF X
loop: l.d f0,x(r10)                      IF ID EX ME WB
```

3) Respecto al riesgo de datos existente en el registro `r14`, indique con detalle como se resuelve dicho riesgo. **Justifique la respuesta**.

Cortocircuito 5, MEaID porque la instrucción daddi tendrá el resultado en el registro de segmentación EX-MEM y por tanto la información se pasara del registro EX-MEM al comparador que esta en ID en la etapa ME.

c) ¿Como se podría reordenar el código para reducir los ciclos de detención? ¿Es posible eliminarlos todos?

```
Instrucción             1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22
loop: l.d f0,x(r10)     IF ID EX ME WB
      l.d f1,y(r11)        IF ID EX ME WB
      mul.d f4,f2,f0;         IF ID M1 M2 M3 WB
      mul.d f5,f3,f1;            IF ID M1 M2 M3 WB
      daddi r10,r10,8               IF ID EX ME WB
      daddi r11,r11,8                  IF ID EX ME WB
      div.d f6,f4,f5                      IF ID D1 D2 D3 WB
      daddi r14,r14,-1                       IF ID EX ME WB
      s.d f6,z(r12)                             IF ID EX ME WB
      daddi r12,r12,8                              IF ID EX ME WB
      bnez r14,loop                                   IF ID EX ME WB
      <sgte1>                                            IF X
loop: l.d f0,x(r10)                                        IF ID EX ME WB
```

□

4. **(2.5 puntos)** El siguiente fragmento de código es parte de un bucle que procesa un vector $\vec{v}$ formado por componentes de tipo *dword* con valores *booleanos* (es decir, cada componente sólo puede tener valor '0' o '1'). En cada iteración, se accede a una de las componentes del vector con la instrucción `ld r10,v(r19)`, de manera que si en una iteración se accede a la componente v[i], en la siguiente se accede a la componente v[i+1].

```
v: .dword ... ; v es un vector de componentes 0 o 1

loop:
      ...
      ld r10,v(r19)     ; lee v[i]
      ...
      ...
      bnez r10, fi      ; salta si v[i] != 0
      ...
fi:
      ...
      daddi r20,r20,-1  ;
      bnez r20,loop     ; salta si r20 != 0
```

Sabiendo que el bucle ejecuta 5000 iteraciones, contesta a las siguientes preguntas:

a) Si se utiliza un predictor simple tipo *predict-not-taken*, indica el número de aciertos del predictor en el bucle sabiendo que el vector contiene 3000 componentes '0'. ¿Cuántos aciertos son atribuibles a la instrucción `bnez r10,fi`? ¿Cuántos a la instrucción `bnez r20,loop`? Razona la respuesta.

b) Para la implementación de un predictor de tipo *predict-not-taken* en un procesador MIPS de 5 etapas (IF, ID, EX, MEM y WB) que actualiza el PC en la etapa MEM, indica la latencia de salto.

c) Asumiendo que el fragmento en ensamblador se ejecuta en el procesador del apartado anterior, calcula el CPI medio del salto `bnez r10,fi`.

d) Considera dos iteraciones consecutivas del bucle, i e i+1. Existen 4 combinaciones posibles de componentes $v[i]$ y $v[i+1]$:

Combinación	$v[i]$	$v[i+1]$	%
C0	0	0	40
C1	0	1	20
C2	1	0	10
C3	1	1	30

Indica, en qué combinaciones un predictor de 1 bit acierta para la instrucción `bnez r10,fi` en la iteración i+1 y en qué casos falla. ¿Qué relación deben tener los valores de las componentes $v[i]$ y $v[i+1]$ para que el predictor acierte? ¿Por qué?

e) Para la implementación de una BTB que predice la condición y el destino de los saltos en la etapa IF; que calcula la condición y la dirección de destino, y escribe el PC en la etapa ID, indica la penalización de salto para los siguientes casos:

- Se predice que no salta y finalmente no salta.
- Se predice que no salta y finalmente sí salta.
- Se predice que sí salta y finalmente no salta.
- Se predice que sí salta y finalmente sí salta.

f) Se han estudiado los componentes del vector $\vec{v}$, y se ha comprobado que las combinaciones se producen en los porcentajes mostrados en tabla anterior. Considerando las penalizaciones calculadas en el apartado anterior, calcula el CPI medio del salto `bnez r10,fi`. Nota: asume que para cualquier combinación C0,C1,C2 o C3, el salto siempre tiene entrada en la tabla.

Solución:

a) Si se utiliza un predictor simple tipo *predict-not-taken*, indica el número de aciertos del predictor en el bucle sabiendo que el vector contiene 3000 componentes '0'. ¿Cuántos aciertos son atribuibles a la instrucción `bnez r10,fi`? ¿Cuántos a la instrucción `bnez r20,loop`? Razona la respuesta.
El predictor *predict-not-taken* acierta 3000 veces para la instrucción `bnez r10,fi`, ya que este último no salta si la componente es igual a '0'. Con respecto a la instrucción `bnez r20,loop` sólo acierta en la última iteración (1 acierto), ya que es la única en la que el salto no es efectivo. En total acierta 3001 veces.

b) Para la implementación de un predictor de tipo *predict-not-taken* en un procesador MIPS de 5 etapas (IF, ID, EX, MEM y WB) que actualiza el PC en la etapa MEM, indica la latencia de salto.
La latencia de salto para ese diseño es 3.

c) Asumiendo que el fragmento en ensamblador se ejecuta en el procesador del apartado anterior, calcula el CPI medio del salto `bnez r10,fi`.

$$CPI = 1 \times \frac{3000}{5000} + (1+3) \times \frac{2000}{5000} = 2{,}2$$

d) Indica, en qué combinaciones un predictor de 1 bit acierta para la instrucción `bnez r10,fi` en la iteración i+1 y en qué casos falla. ¿Qué relación deben tener los valores de las componentes $v[i]$ y $v[i+1]$ para que el predictor acierte? ¿Por qué?
Acierta en los casos C0 y C3 ya que el predictor de 1 bit predice en la iteración i+1 lo mismo que ocurrió en la iteración i. Por tanto, para que el predictor acierte para i+1, debe cumplirse que $v[i]$ y $v[i+1]$ sean iguales.

e) Para la implementación de una BTB que predice la condición y el destino de los saltos en la etapa IF; que calcula la condición y la dirección de destino, y escribe el PC en la etapa ID, indica la penalización de salto para los siguientes casos:

- Se predice que no salta y finalmente no salta: 0.
- Se predice que no salta y finalmente sí salta: 1.
- Se predice que sí salta y finalmente no salta: 1.
- Se predice que sí salta y finalmente sí salta: 0.

f) Considerando las penalizaciones calculadas en el apartado anterior, calcula el CPI medio del salto `bnez r10,fi`.

$$CPI = 1 \times \frac{40+30}{100} + (1+1) \times \frac{20+10}{100} = 1{,}3$$

□