Arquitectura e Ingeniería de Computadores - 3er Curso. Sol. del Examen final (1er parcial) 24/1/2022

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

1. **(3 puntos)** Un diseñador está valorando el nuevo procesador M1 Pro para usarlo con el software de procesamiento de video PV. El software PV utiliza tanto la CPU y la GPU del procesador como la E/S. Los procesadores M1 tienen núcleos de CPU *Firestorm* de alto rendimiento a una frecuencia de $3{,}2$ GHz. El tiempo de ejecución total del software PV sobre el procesador Apple M1 original es de 100 min. De dicho tiempo, el programa PV consume 20 min ejecutando código no paralelizable sobre uno de los núcleos *Firestorm* y 40 min ejecutando código paralelizable sobre todos los núcleos *Firestorm* del procesador.

   A continuación se muestran las características de los procesadores Apple M1 y M1 Pro.

Características	Apple M1	M1 Pro
Núcleos CPU *Firestorm*	4	8
Núcleos GPU	8	16

   Ambos modelos utilizan la misma frecuencia reloj para los núcleos de la CPU y la GPU.

   Se solicita, justificando la respuesta:

   a) Determinar el tiempo en minutos que se utilizará la CPU en el programa PV ejecutándose sobre el M1 Pro, suponiendo la misma distribución del código que sobre el Apple M1 original.

   b) ¿Cuál será la aceleración local de la CPU, teniendo en cuenta la parte secuencial y paralela, obtenida por el programa PV en el modelo M1 Pro?

   c) ¿Cuál es el porcentaje de tiempo que el programa PV utilizaba la CPU sobre el procesador Apple M1?

   d) Si el software PV tarda 70 min sobre el procesador M1 Pro ¿Qué porcentaje de tiempo se dedicaba a la GPU sobre el Apple M1 original?

Solución:

a) Determinar el tiempo en minutos que se utilizará la CPU en el programa PV ejecutándose sobre el M1 Pro, suponiendo la misma distribución del código que sobre el Apple M1 original.

La aceleración que aplicamos a T_{seq} no cambia.
La aceleración que aplicamos a $T_{par-fire}$ depende del número de núcleos de CPU *Firestorm* en el modelo M1 Pro. En este caso, pasamos de 4 núcleos a 8, por lo tanto, $S_{par-fire} = 4/2 = 2$.
Así pues, el nuevo tiempo del software PV en la CPU será:

$$T_{cpu-M1-Pro} = T_{seq} + \frac{T_{par-fire}}{S_{par-fire}} = 20 + \frac{40}{2} = 20 + 20 = 40 \text{ min}$$

b) ¿Cuál será la aceleración local de la CPU, teniendo en cuenta la parte secuencial y paralela, obtenida por el programa PV en el modelo M1 Pro?

En el ejercicio anterior hemos visto que el tiempo de ejecución en la CPU pasaba de $T_{cpu-M1} = 20 + 40 = 60$ min a $T_{cpu-M1-pro} = 40$ min.
Así pues, tenemos:

$$S_{cpu} = \frac{T_{cpu-M1}}{T_{cpu-M1-pro}} = \frac{20+40}{40} = \frac{60}{40} = 1{,}50$$

c) ¿Cuál es el porcentaje de tiempo que el programa PV utilizaba la CPU sobre el procesador Apple M1?

$$F_{cpu} = \frac{T_{cpu}}{T_{ej}} = \frac{20+40}{100} = \frac{60}{100} = 0{,}6 \rightarrow 60\,\%$$

d) Si el software PV tarda 70 min sobre el procesador M1 Pro ¿Qué porcentaje de tiempo se dedicaba a la GPU sobre el Apple M1 original?

De los apartados anteriores sabemos que la aceleración local de la CPU es $S_{cpu} = 1{,}50$. Sabemos que $T_{ej-M1-pro} = 70$ min y que el número de núcleos de la GPU se duplica, $S_{gpu} = \frac{16}{8} = 2$
Así pues usando la ley de Amdahl:

$$S_{M1-pro} = \frac{T_{ej-M1}}{T_{ej-M1-pro}} = \frac{1}{1 - F_{cpu} - F_{gpu} + \frac{F_{cpu}}{S_{cpu}} + \frac{F_{gpu}}{S_{gpu}}}$$

$$\frac{100}{70} = \frac{1}{1 - 0{,}6 - F_{gpu} + \frac{0{,}6}{1{,}50} + \frac{F_{gpu}}{2}}$$

Tras operar nos queda:

$$F_{gpu} = 0{,}2 \rightarrow 20\,\%$$

□

2. **(3.5 puntos)**

Disponemos de un sistema con un procesador MIPS que funciona a 2 GHz, segmentado en 5 etapas (IF,ID,EX,MEM,WB). 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 fase MEM.

Las estadísticas de ejecución de los programas son las siguientes:

Operación	%
ALU	55 %
Saltos	15 %
Carga	20 %
Almacenamiento	10 %

Por otra parte, un 20 % de las instrucciones ALU necesitan que se inserte un ciclo de parada para resolver los riesgos de datos cuando les precede una instrucción de Carga, mientras que un 75 % de los saltos son efectivos.

Con el objeto de mejorar las prestaciones, se plantean dos opciones:

Modificación 1: Modificar la resolución de riesgos de control para utilizar un predictor BTB que obtiene su predicción en la fase IF y que acierta el 90 % de los saltos. Un 80 % de las instrucciones de salto están en la tabla.

Modificación 2: Manteniendo *predict-not-taken*, modificar la ruta de datos para que los saltos calculen la condición, la dirección y escriban el PC en la etapa ID. Como consecuencia del cambio:

- Algunas instrucciones de salto insertan un ciclo de parada para resolver riesgos de datos relacionados con la evaluación de la condición.
- Algunas instrucciones de salto insertan dos ciclos de parada para resolver riesgos de datos relacionados con la evaluación de la condición.
- El período de reloj aumenta.

Se pide:

a) Calcular el tiempo de ejecución (en segundos) en el MIPS original de un programa que ejecuta 100M de instrucciones.

b) Calcular el tiempo de ejecución (en segundos) en el MIPS con la **Modificación 1** de un programa que ejecuta 100M de instrucciones.

c) Explicar, utilizando ejemplos de diagramas instrucciones–tiempo, las razones de la aparición de ciclos de parada por riesgos de datos en los saltos al incorporar la **Modificación 2**. Indicar también una posible causa del incremento del periodo de reloj en ese caso.

Solución:

a) Calcular el tiempo de ejecución (en segundos) en el MIPS original de un programa que ejecuta 100000000 de instrucciones.

$$T_{ejec} = I \times CPI \times T$$

donde:

- $I = 100000000$

- CPI
  El $20\,\%$ de las instrucciones ALU ($55\,\%$ del total) insertan un ciclo de parada. Y los saltos efectivos ($75\,\%$) insertan tres ciclos de parada para resolver los riesgos de control con *predict-not-taken*. Nótese que la latencia de salto es igual a 3, dado que se escribe el PC en la 4ª etapa del ciclo de instrucción:
  $CPI = 1 + stalls = 0{,}2 \cdot 0{,}55 \cdot 1 + 0{,}75 \cdot 0{,}15 \cdot 3 = 1 + 0{,}11 + 0{,}3375 = 1{,}45$
- $T = \frac{1}{2GHz} = 0{,}50$ ns

$$T_{ejec} = I \times CPI \times T = 100000000 \times 1{,}45 \times 0{,}5 = 0{,}0725\,\text{s}$$

b) Calcular el tiempo de ejecución (en segundos) en el MIPS con la **Modificación 1** de un programa que ejecuta 100000000 de instrucciones.

$$T_{ejec} = I \times CPI \times T$$

donde:

- $I = 100000000$
- CPI
  De igual manera que en el procesador original, el $20\,\%$ de las instrucciones ALU ($55\,\%$ del total) insertan un ciclo de parada. En el caso de los saltos, como hay un predictor dinámico BTB, solo se insertarán ciclos de parada cuando no hay entrada en la tabla ($20\,\%$) y el salto sea efectivo ($75\,\%$) o bien estando la instrucción de salto en la tabla BTB ($80\,\%$), cuando se falla en la predicción ($10\,\%$). La latencia de salto es 3, igual que en el procesador original:
  $CPI = 1 + stalls = 1 + 0{,}2 \cdot 0{,}55 \cdot 1 + 0{,}2 \cdot 0{,}75 \cdot 0{,}15 \cdot 3 + 0{,}8 \cdot 0{,}1 \cdot 0{,}15 \cdot 3 = 1 + 0{,}11 + 0{,}0675 + 0{,}036 = 1{,}21$
- $T = \frac{1}{2GHz} = 0{,}5$ ns

$$T_{ejec} = I \times CPI \times T = 100000000 \times 1{,}21 \times 0{,}5 = 0{,}0605\,\text{s}$$

c) Explicar, utilizando cronogramas de ejemplo, las razones de la aparición de ciclos de parada por riesgos de datos en los Saltos al incorporar la **Modificación 2**. Indica también una posible causa del incremento de la frecuencia de reloj en ese caso.
El problema aparece al adelantar el cálculo de la condición a la etapa ID. No es posible eliminar ciclos de parada mediante cortocircuitos. Las siguientes secuencias de código ilustran el problema:

- Salto precedido de instrucción aritmética. 1 ciclo de parada.

```
seq r3,r2,r1    IF ID EX M  WB
bnez r3, loop      IF id ID EX M  WB
```

- Salto precedido de instrucción de carga. 2 ciclos de parada.

```
ld r3, A(r1)    IF ID EX M  WB
bnez r3, loop      IF id id ID EX M  WB
```

En lo relativo al aumento del periodo de reloj, la causa puede ser que la etapa ID puede aumentar su retardo y convertirse en la etapa más lenta, la cual fijará el periodo de reloj. En particular, en la etapa ID habrá que:

- Leer registros
- Evaluar la condición
- Configurar el multiplexor que escribe en el PC
- Escribir el PC

□

3. **(3.5 puntos)** Se dispone de un procesador MIPS con los siguientes operadores multiciclo:

   - Sumador/Restador. Lat= 2, IR= $\frac{1}{2}$.
   - Multiplicador. Lat= 3, IR= $\frac{1}{3}$.
   - Divisor. Lat= 4, IR= 1.

   Los riesgos estructurales y de datos se detectan en la fase ID, insertando tantos ciclos de parada como sean necesarios y utilizando cortocircuitos siempre que sea posible. Los riesgos de control se resuelven mediante detención, teniendo en cuenta que la condición y la dirección destino del salto se calculan durante la etapa EX del salto y el PC se actualiza al final de la etapa EX.

   El siguiente bucle accede a un vector de n elementos sobre el que se realizan distintas operaciones aritméticas:

```
loop:   l.d f1, 0(r1)
        add.d f2, f1, f0
        mult.d f3, f1, f0
        mult.d f4, f2, f3
        div.d f5, f4, f6
        s.d f5, 0(r1)
        dadd r1, r1, 8
        dsub r5, r4, r1
        bnez r5, loop
        trap 0
```

   Responda a las siguiente cuestiones utilizando la notación: `IF` fase de búsqueda, `ID` fase de decodificación, `EX` fase de ejecución monociclo, `A1,A2` fases de ejecución del sumador/restador, `M1,M2,M3` fases de ejecución del multiplicador, `D1,D2,D3,D4` fases de ejecución del divisor, `ME` fase de acceso a memoria y `WB` fase de escritura en registros. Las instrucciones multiciclo no realizan la fase `ME`.

   Conteste a las siguientes preguntas, **justificando todas las respuestas**.

   a) Complete el diagrama instrucciones-tiempo perteneciente a la primera iteración del bucle incluyendo hasta la primera instrucción de la segunda iteración.

   b) ¿Cuántos ciclos de penalización añaden las instrucciones de salto?

   c) Para cada uno de los operadores, indique si se trata de un operador segmentado o no segmentado.

   d) Indique cuántos ciclos se emplean en cada iteración y calcule el CPI medio de una iteración del bucle.

   e) Si el multiplicador fuera reemplazado por otro con la misma latencia e IR = 1, ¿cómo afectaría este cambio al diagrama instrucciones-tiempo y al tiempo de ejecución del bucle?

   f) Con el fin de reducir los ciclos de parada y suponiendo que todos los operadores son segmentados, un compilador avanzado aplicaría la técnica de loop-unrolling. ¿Cuántas veces como mínimo se debería "desenrollar" el bucle para eliminar los ciclos de parada por dependencias de datos?

   Solución:

   a) Complete el diagrama instrucciones-tiempo perteneciente a la primera iteración del bucle incluyendo hasta la primera instrucción de la segunda iteración.

```
                        1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23
loop:   l.d f1,0(r1)    IF ID EX ME WB
        add.d f2,f1,f0  |  IF id ID A1 A2 WB
        mult.d f3,f1,f0 |     if IF ID M1 M2 M3 WB
        mult.d f4,f2,f3 |           IF id id ID M1 M2 M3 WB
        div.d f5,f4,f6  |              if if IF id id ID D1 D2 D3 D4 WB
        s.d f5,0(r1)    |                        if if IF id id ID EX ME WB
        dadd r1,r1,8    |                              if if IF ID EX ME
        dsub r5,r4,r1   |                                    IF ID EX ME WB
        bnez r5,loop    |                                       IF ID EX ME WB
loop:   l.d f1,0(r1)    |                                          if if IF ID EX ME WB
                        |----------------------18 ciclos -------------------|
```

b) ¿Cuántos ciclos de penalización añaden las instrucciones de salto? Cada instrucción de salto añade dos ciclos de parada debido a que la escritura del PC se realiza al final de la etapa EX, retrasando la busqueda de la siguiente instrucción 2 ciclos.

c) Para cada uno de los operadores, indique si se trata de un operador segmentado o no segmentado. Si los operadores tienen $IR < 1$ se trata de operadores no segmentados ya que puede empezar una nueva operación cada x ciclos, siendo $IR = \frac{1}{x}$, mientras que si tienen $IR = 1$ el operador es segmentado. Así pues, tanto el operador sumador/restador como el de multiplicación son no-segmentados, mientras que el de división es un operador segmentado.

d) Indique cuántos ciclos se emplean en cada iteración y calcule el CPI medio de una iteración del bucle. Se emplean 18 ciclos en cada iteración ya que la primera instrucción de la segunda iteración empieza en el ciclo 19. $CPI = \frac{18\ ciclos}{9\ instrucciones} = 2$

e) Si el multiplicador fuera reemplazado por otro con la misma latencia e IR = 1, ¿cómo afectaría este cambio al diagrama instrucciones-tiempo y al tiempo de ejecución del bucle?

No mejoraría. Las dos multiplicaciones del bucle proporcionado presentan una riesgo de datos entre ellas y por tanto no se podría aprovechar el hecho de tener un operador segmentado. Para poder aprovechar un multiplicador segmentado ambas multiplicaciones deberían ser independientes.

f) Con el fin de reducir los ciclos de parada y suponiendo que todos los operadores son segmentados, un compilador avanzado aplicaría la técnica de loop-unrolling. ¿Cuántas veces como mínimo se debería "desenrollar" el bucle para eliminar los ciclos de parada por dependencias de datos?

Se debería "desenrollar" 3 veces ya que el máximo número de ciclos de parada consecutivos por dependencias de datos son 2.

□