Tecnología de Computadores

Departamento de Arquitectura de Computadores

Relación de Problemas del Tema 3.

1. (H.P. 4.12) Supongamos que las etapas del camino de datos del procesador visto en clase tienen las siguientes latencias:

IF	ID	EX	MEM	WB
200ps	150ps	120ps	190ps	140ps

Se pide:

a) ¿Cuál es el ciclo de reloj si el procesador se decide implementar en una versión monociclo? ¿Y si fuera segmentado?

- **Monociclo:** $TC_{mon} = 200ps + 150ps + 120ps + 190ps + 140ps = 800ps$
- **Monociclo:** $TC_{seg} = MAX(200ps, 150ps, 120ps, 190ps, 140ps) = 200ps$

b) ¿Cuál es la latencia de una instrucción `lw` en un procesador segmentado y en uno monociclo?

- **Monociclo:** Latencia LW $= TC_{mon} = 800ps$
- **Monociclo:** Latencia LW $= 5ciclos \times TC_{seg} = 5 \times 200ps = 1000ps$

c) Si se divide una etapa del camino de datos segmentado en dos nuevas etapas, cada una con una latencia mitad de la etapa original, ¿qué etapa se debería dividir y cuál sería el nuevo ciclo de reloj del procesador?

- Se debería dividir la etapa más lenta ya que esta es la que fija el tiempo de ciclo (TC_{seg}). En este caso sería la etapa IF que dura 200ps. Al dividirla tendríamos dos etapas (IF_1, IF_2) de 100ps cada una. Ahora el nuevo tiempo de ciclo vendría marcado por la etapa más lenta de las 6, que en este caso sería la de MEM ($TC_{seg} = 190ps$).

2. (H.P. 4.13) Sean las siguientes secuencias de instrucciones:

```
a.   lw  $1, 40($6)          b.   lw  $5, -16($5)
     add $6, $2, $2               sw  $5, -16($5)
     sw  $6, 50($1)               add $5, $5, $5
```

a) Indica las dependencias y su tipo.

- Código a:
  - Antidependencia entre lw y add ($6)
  - Verdadera entre add y sw ($6)
  - Verdadera entre lw y sw ($6)
- Código b:
  - Verdadera entre lw y sw ($5)
  - Verdadera entre lw y add ($5)

- Antidependencia entre sw y add ($5)
- De salida entre lw y add ($5)
- Antidependencia entre lw y add ($5)

b) Indica los riesgos y añade instrucciones `nop` para resoverlos.

- Código a:
  - Riesgo de datos entre **add** y sw
  - Riesgo de datos entre lw y **sw**
  - Código modificado (suponemos anticipación en banco de registros)

```
a.  lw  $1, 40($6)
    add $6, $2, $2
    nop
    nop
    sw  $6, 50($1)
```

- Código b:
  - Riesgo de datos entre lw y sw
  - Riesgo de datos entre **lw** y sw
  - Código modificado (suponemos anticipación en banco de registros)

```
b.  lw  $5, -16($5)
    nop
    nop
    sw  $5, -16($5)
    add $5, $5, $5
```

3. (H.P. 4.15) Sean las nuevas instrucciones:

`bezi (rt), desp`	opec $\|r0\|rt\|desp$	**If Mem[rt]=0 then PC ← PC+4+desp**
`swi rd, rs(rt)`	opec $\|rs\|rt\|rd\|xxxx$	**Mem[rs+rt] ← rd**

a) ¿Qué cambios hay que hacer en el camino de datos para añadir estas instrucciones a la ISA del MIPS?

- Ver diseño al final del documento

b) ¿Qué nuevas señales de control deben añadirse al diseño?

- Ver diseño al final del documento

c) ¿Qué tipo de riesgos se pueden producir con estas instrucciones?

- Riesgos de datos (porque leen el contenido de registros) y de control (bezi escribe en PC)

4. (H.P. 4.16) Para cada una de las siguientes instrucciones:

```
a.  lw  $1, 40($6)
b.  add $5, $5, $5
```

a) Identifica qué se almacena en cada uno de los registros situados entre dos etapas del pipeline.

- a. Sólo aparece la información relacionada con la instrucción lw
  - IF/ID: Código instrucción lw.
  - ID/EX: Contenido $6. Constante 40 (16 bits). Código de reg. destino (IB[20:16]).
  - EX/MEM: Dirección de memoria a leer. Código de reg. destino (IB[20:16]).
  - MEM/WB: Dato de memoria leído. Código de reg. destino (IB[20:16]).
- b. Sólo aparece la información relacionada con la instrucción add
  - IF/ID: Código instrucción add.
  - ID/EX: Contenido $5 por duplicado. Código de reg. destino (IB[15:11]).
  - EX/MEM: Resultado de la suma. Código de reg. destino (IB[15:11]).
  - MEM/WB: Resultado de la suma. Código de reg. destino (IB[15:11]).

b) ¿Qué registros necesitan leerse y cuáles se leen realmente?

- a. Se leen $1 y $6, sólo se necesita $6.
- b. Se leen $5 por los dos puertos de lectura y son necesarias ambas lecturas.

c) ¿Qué hace la instrucción en las etapas EX y MEM?

- a. En EX calcula la dir. de memoria a leer sumando al contenido del reg $6 la extensión de signo de la constante 40 (ambos almacenados en ID/EX). En MEM lee la posición de memoria calculada en EX (almacenada en EX/MEM).
- b. En EX suma el contenido de los dos registros (en este caso los dos son $5 y están almacenados en ID/EX). En MEM no se hace nada, solo se pasa el resultado de la suma al siguiente registro de segmentación (de EX/MEM pasa a MEM/WB).

5. Sea el programa `suma1` dado por el siguiente código MIPS:

```
      sub  $5, $0, $0
suma: lw   $10, 1000($20)
      add  $5, $5, $10
      addi $20, $20, -4
      bne  $20, $0, suma
```

Se pide:

a) Describir brevemente la tarea realizada por `suma1`.

- El código suma los datos (tamaño word) contenidos en un vector almacenado a partir de la posición de memoria 1000 y con un tamaño de vector igual al valor del registro $20 entre 4.

b) Detectar las dependencias que afectan a `suma1` en el MIPS segmentado en 5 etapas y clasificarlas en dependencias de datos y dependencias de control.

- Dependencias de datos: sub-add, lw-add, addi-bne, addi-lw, add-add
- Dependencias de control: bne-lw

c) Gestión de dependencias por parte del compilador:

c.1 Suponer un MIPS segmentado sin ningún tipo de soporte hardware para solventar los problemas derivados de los riesgos de datos que conlleva la segmentación. Reescribir el código reordenándolo e insertando el el mínimo número de códigos de no-operación (nop) para producir una nueva versión, `suma2`, que pueda ejecutarse correctamente en este MIPS.

```
        sub  $5, $0, $0
suma2:  lw   $10, 1000($20)
        addi $20, $20, -4
        nop
        nop
        add  $5, $5, $10
        bne  $20, $0, suma
        nop
        nop
        nop
```

c.2 Evaluar la mejora obtenida hasta ahora con respecto al punto de partida. Para ello, comparar el tiempo de ejecución que ofrece `suma2` al ejecutarse sobre el MIPS anterior con respecto al de `suma1` sobre el MIPS sin segmentar de la implementación monociclo. Suponer un número `N` de iteraciones en ambos programas y que las latencias de las etapas del camino de datos del procesador son las vistas en clase:

IF	ID	EX	MEM	WB
40ps	20ps	40ps	40ps	20ps

- Tiempo monociclo: $(1\ instr. + N \times 4\ instr.) \times 160\ ps/instr. = (160 + 640N)\ ps$
- Tiempo sementado: $200\ ps + (N \times 9\ instr.) \times 40\ ps/instr. = (200 + 360N)\ ps$
- Si suponemos N suficientemente grande, podemos despreciar el tiempo de la primera instrucción (fuera del bucle), con lo que el cociente entre el tiempo monociclo y sementado sería: $640N \div 360N = 16 \div 9 = 1{,}777$ (casi el doble de rápido el sementado)

6. Sea una arquitectura RISC con un conjunto de instrucciones similar al del procesador MIPS visto en clase. Su camino de datos presenta una segmentación en 3 etapas y una única memoria, común para instrucciones y datos. Esto hace que las instrucciones se ejecuten según se muestra a continuación:

INST	1	2	3	4	5
Inst1	`IFD`	`REX`	`MEW`		
Inst2		`IFD`	`REX`	`MEW`	
Inst3			`IFD`	`REX`	`MEW`

donde:

- `IFD` es la etapa de búsqueda y decodificación de instrucción.
- `REX` es la etapa de búsqueda de operandos (lectura del banco de registros), ejecución de operación ó resolución de condiciones de salto y cálculo de dirección para operando destino ó salto.

- `MEW` es la etapa de acceso a memoria para instrucciones de carga/almacenamiento, escritura de resultados en registros para instrucciones aritmético/lógicas y de carga y actualización del PC para instrucciones de salto.

Además no podrá leerse en un mismo ciclo un registro que va a ser escrito en dicho ciclo (es decir, no hay anticipación en el banco de registros).

Bajo estas condiciones, responder a las siguientes cuestiones:

a) ¿Qué tipo de riesgos pueden darse? Poner ejemplos.
   - Riesgos estructurales: Acceso simultáneo a memoria de un solo puerto en IFD (Inst3) y MEW (Inst1)
   - Riesgos de datos: Acceso simultáneo de lectura y escritura a un mismo registro en MEW (Inst1) y REX (Inst2)
   - Riesgos de control: Se carga PC en MEW, por lo que entran 2 instrucciones antes de cargarlo.

b) ¿Qué ocurriría con las instrucciones de salto?
   - Habría que poner dos NOPs para evitar los riesgos de control.

c) Representar en un diagrama de ciclos la evolución del cauce para el siguiente trozo de código e identifica los riesgos que puedan darse.

Siempre hay riesgo estructural entre MEW de una instrucción e IFD de la instrucción 2 instrucciones más adelante.

`sw $5, 3000($7)`	IFD	R(5,7)	M( )					
`or $5, $4, $5`		IFD	R(4,5)	M(5)				
`sub $8, $4, $7`			IFD	R(4,7)	**M(8)**			
`lw $4, 3000($8)`				IFD	**R(8)**	M(4)		
`bne $7, $0, etiq`					IFD	R(7,0)	**M( )**	
`add $8, $8, $5`						**IFD**	R(8,5)	**M(8)**
`etiq: sw $5, 3000($8)`							IFD	**R(8,5)**

7. Sea el siguiente código MIPS, que a partir de ahora referenciaremos como `mi_prog`:

```
      ori  $2, $0, 1000h
loop: lw   $1, 2800h($2)
      sub  $4, $1, $0          rotar: add  $10, $4, $4
      jal  rotar                      muli $7, $10, 2
      sw   $7, 7800h($2)              jr $31
      sw   $1, C800h($2)
      subi $2, $2, 4
      bne  $2, $0, loop
```

Señalar las instrucciones que **pueden producir riesgos** en `mi_prog` para un MIPS segmentado con anticipación en el banco de registros. Responder en las siguientes tablas:

Riesgos de datos

Instrucción(es) (dar mnemotécnicos)	Registro(s) involucrado(s)
ori - lw	$2
lw - sub	$1
sub - add (jal)	$4
add - muli	$10
muli - sw (jr)	$7
subi - bne	$2
subi - lw	$2

Riesgos de control

Instrucción(es) (dar mnemotécnicos)
jal
jr
bne

Señales de control

- Etapa EX: IB[5:0], OPALU, FB, EX, FDC, FPC1
- Etapa MEM: RM, WM, FPC0
- Etapa WB: FC, WR