Soluciones a ejercicios de Paralelismo a Nivel de instrucción

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Arquitectura de Computadores
Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

1. Ejercicios de examen

Ejercicio 1 *Examen de junio de 2015.*

Considere un procesador tipo MIPS con arquitectura segmentada (pipeline) que tiene dos bancos de registros separados uno para números enteros y otro para números en coma flotante. El banco de registros enteros dispone de 32 registros. El banco de registros de coma flotante dispone de 16 registros de doble precisión (**$f0**, **$f2**, . . . , **$f30**).

Se supone que se dispone de suficiente ancho de banda de captación y decodificación como para que no se generen detenciones por esta causa y que se puede iniciar la ejecución de una instrucción en cada ciclo, excepto en los casos de detenciones debidas a dependencias de datos.

La siguiente tabla muestra las latencias adicionales de algunas categorías de instrucciones, en caso de que haya dependencia de datos. Si no hay dependencia de datos la latencia no es aplicable.

Cuadro 1: Latencias adicionales por instrucción

Instrucción	Latencia adicional	Operación
ldc1	+2	Carga un valor de 64 bits en un registro de coma flotante.
sdc1	+2	Almacena un valor de 64 bits en memoria principal.
add.d	+4	Suma registros de coma flotante de doble precisión.
mul.d	+6	Multiplica registros de coma flotante de doble precisión.
addi	+0	Suma un valor a un registro entero.
subi	+0	Resta un valor a un registro entero.
bnez	+1	Salta si el valor de un registro no es cero.

La instrucción bnez usa bifurcación retrasada con una ranura de retraso (*delay slot*).

En esta máquina se desea ejecutar el siguiente código:

```
bucle:  ldc1 $f0, ($t0)
        ldc1 $f2, ($t1)
        add.d $f4, $f0, $f2
        mul.d $f4, $f4, $f6
        sdc1 $f4, ($t2)
        addi $t0, $t0, 8
        addi $t1, $t1, 8
        subi $t3, $t3, 1
        bnez $t3, bucle
        addi $t2, $t2, 8
```

Inicialmente los valores de los registros son:

- **$t0**: **0x00100000**.
- **$t1**: **0x00140000**.
- **$t2**: **0x00180000**.
- **$t3**: **0x00000100**.

Se pide:

1. Enumere las dependencias de datos RAW que hay en el código anterior.
2. Indique todas las detenciones que se producen al ejecutar una iteración del código anterior, e indique el número total de ciclos por iteración.
3. Intente planificar el bucle para reducir el número de detenciones.
4. Desenrolle el bucle de manera que en cada iteración se procesen cuatro posiciones de los arrays y determine el speedup conseguido. Utilice nombres de registros reales (**$f0**, **$f2**, ..., **$f30**).

IMPORTANTE: Se considerarán incorrectas soluciones que no usen registros realmente existentes (p. ej. **$f2'** o **$f2"** son nombres no válidos).

Solución 1

Depenencias Si se numeran las instrucciones consecutivamente comenzando en **I1** (hasta **I10**), se pueden identificar las siguientes dependencias RAW:

1. **$f0**: **I1** → **I3**.
2. **$f2**: **I2** → **I3**.
3. **$f4**: **I3** → **I4**.
4. **$f4**: **I4** → **I5**.
5. **$t3**: **I8** → **I9**.

Detenciones A continuación se indican las detenciones que se producen al ejecutar el código:

```
ldc1 $f0, ($t0)          #I1
ldc1 $f2, ($t1)          #I2
<stall> x 2
add.d $f4, $f0, $f2      #I3
<stall> x 4
mul.d $f4, $f4, $f6      #I4
<stall> x 6
sdc1 $f4, ($t2)          #I5
addi $t0, $t0, 8         #I6
addi $t1, $t1, 8         #I7
subi $t3, $t3, 1         #I8
bnez $t3, bucle #I9
addi $t2, $t2, 8         #I10
```

En total se requieren 22 ciclos.

Planificación de bucle Si se reordenan las instrucciones se puede reducir el número de detenciones:

```
ldc1 $f0, ($t0)          #I1
ldc1 $f2, ($t1)          #I2
addi $t0, $t0, 8         #I6
addi $t1, $t1, 8         #I7
add.d $f4, $f0, $f2      #I3
subi $t3, $t3, 1         #I8
<stall> x 3
mul.d $f4, $f4, $f6      #I4
<stall> x 6
sdc1 $f4, ($t2)          #I5
bnez $t3, bucle #I9
addi $t2, $t2, 8         #I10
```

En total se requieren 19 ciclos.

Bucle desenrollado Al desenrollar el bucle con un factor de cuatro se obtiene.

```
ldc1 $f0, ($t0)
ldc1 $f2, ($t1)
ldc1 $f8, 8($t0)
ldc1 $f10, 8($t1)
ldc1 $f14, 16($t0)
ldc1 $f16, 16($t1)
ldc1 $f20, 24($t0)
ldc1 $f22, 24($t1)
add.d $f4, $f0, $f2
add.d $f12, $f8, $f10
add.d $f18, $f14, $f16
add.d $f24, $f20,$f22
<stall>
mul.d $f4, $f4, $f6
mul.d $f12, $f12, $f6
mul.d $f18, $f18, $f6
mul.d $f24, $f24, $f6
addi $t0, 32
addi $t1, 32
<stall>
sdc1 $f4, ($t2)
sdc1 $f12, 8($t2)
sdc1 $f18, 16($t2)
sdc1 $f24, 24($t2)
subi $t3, 4
bnez $t3, bucle
addi $t2, 32
```

En total se requieren 27 ciclos cada 4 iteraciones. Esto es 6,75 ciclos por iteración

Ejercicio 2 *Examen de enero de 2015.*

Sea el siguiente fragmento de código se encuentra almacenado a partir de la dirección de memoria **0x1000100C** en una máquina en la que todas las instrucciones ocupan 4 bytes:

```
bucle:  lw $r2, 0($r0)
        addi $r3, $r2, 20
        sw $r3, 0($r1)
        addi $r0, $r0, 4
        addi $r1, $r1, 4
        bnez $r2, bucle
```

Dicho código se ejecuta en una máquina que dispone de una caché L1 para datos asociativa por conjuntos de 2 vías y con un tamaño de 32 KB y una caché L1 para instrucciones de iguales características. También dispone de una caché L2 unificada asociativa por conjuntos de 8 vías con un tamaño de 1 MB. En ambos casos el tamaño de línea es de 32 bytes. Se asume que un acierto en la caché L1 requiere 4 ciclos, y un acierto en la caché L2 requiere 14 ciclos adicionales y la penalización por traer un bloque de memoria principal a la caché de nivel 2 es de 80 ciclos. Todas las cachés tienen una política de escritura write-back.

Inicialmente el valor de los registros es:

- **$r0**: **0x00010000**.
- **$r1**: **0x00080000**.

A partir de la posición **0x00010000** todos los valores en memoria son distintos de cero hasta la posición **0x000100FC**. En la posición de memoria **0x000100FC** hay un valor de cero.

1. Determine cuál debería ser el tiempo medio de acceso asumiendo que un programa (distinto del facilitado) realiza en promedio 2 accesos a datos por instrucción y tiene las siguientes tasas de fallo:
   - L1 instrucciones: 10 %
   - L1 datos: 5 %
   - L2: 2 %
2. Determine el número de fallos que se producen durante la ejecución del fragmento de código facilitado en el enunciado para las cachés L1 de datos, L1 de instrucciones y L2.
3. 1.3: Elabore un diagrama de tiempos para una arquitectura MIPS con un pipeline de 5 etapas para la primera iteración del bucle asumiendo que inicialmente no hay datos ni instrucciones en las cachés y con las siguientes consideraciones:
   - No hay hardware de forwarding.
   - La arquitectura permite que una instrucción escriba en un registro y otra instrucción lea ese mismo registro sin problemas.
   - Las bifurcaciones se tratan vaciando el pipeline.
   - La dirección efectiva de salto se calcula en la etapa de ejecución.

   NOTA: Tenga en cuenta en el cronograma las detenciones debidas a los fallos en toda la jerarquía de caché tanto para instrucciones (etapa IF) como para lecturas y escrituras de datos (etapa M).
4. Repita el diagrama de tiempos para la segunda iteración.

Solución 2

Tiempo medio de acceso En cuanto a los accesos a la caché de nivel 1, se tiene que se realiza 2 accesos a datos por cada acceso a instrucciones. Por tanto la tasa de fallos se obtiene mediante media ponderanda:

$$m_{L1} = \frac{m_{L1I} + 2 \cdot m_{L1D}}{3}$$

$$m_{L1} = \frac{0{,}1 + 2 \cdot 0{,}05}{3} = \frac{0{,}2}{3} = 0{,}0667$$

Por tanto el tiempo medio de acceso sería:

$$T = 4 + m_{L1} \cdot (14 + m_{L2} \cdot 80) = 4 + 0{,}0667 \cdot (14 + 0{,}02 \cdot 80) = 5{,}04 \text{ ciclos}$$

Número de fallos El bucle se ejecuta un total de $\frac{2^8}{4} = 2^6 = 64$ iteraciones.

Analizaremos por separado los accesos de instrucciones y datos.

La primera instrucción se almacena en la dirección **0x1000100C**. La última instrucción se almacena en la dirección en la dirección **0x1000100C** $+(6-1)*4 =$ **0x10001020**.

En la primera iteración, la primera instrucción genera un fallo de caché y trae el bloque de direcciones **0x10001000** – **0x1000101F**, que contiene una instrucción de interés. Las siguientes instrucciones (I2, I3, I4, e I5) generan aciertos. Por último, la instrucción I6 vuelve a generar un fallo. Por tanto, la primera iteración genera 2 fallos y 4 aciertos. El resto de iteraciones generan aciertos en todos los casos.

Como el bucle se ejecuta 64 veces, el acceso a las instrucciones genera los siguientes accesos:

- Fallos L1I: 2
- Aciertos L1I: $4 + 63 \cdot 6$
- Fallos L2: 1
- Aciertos L2: 0

En cada iteración del bucle se lee una dirección de memoria en el rango **0x00010000** – **0x000100FC**. Esto se corresponde con $\frac{2^8}{2^5} = 8$ líneas de caché.

Igualmente se escriben valores en las direcciones **0x00080000** – **0x000800FC**, que se escriben en 8 líneas de la caché. Como las cachés son asociativas por conjuntos no se dan conflictos entre los datos leídos y escritos. Como no se expulsa ninguna línea de la caché L1 no hay escrituras en la caché L2.

1. Fallos L1D: 8 lecturas + 8 escrituras
2. Aciertos L1D: 56 lecturas + 56 escrituras
3. Fallos L2: 8 lecturas
4. Aciertos L2: 0

Diagrama de tiempos para primera iteración Si se numeran las instrucciones de la siguiente forma:

```
bucle:  lw $r2, 0($r0)          #1
        addi $r3, $r2, 20       #2
        sw $r3, 0($r1)          #3
        addi $r0, $r0, 4        #4
        addi $r1, $r1, 4        #5
        bnez $r2, bucle #6
```

Se tienen las siguientes dependencias RAW:

1. **$r2**: **I2** → **I1**
2. **$r3**: **I3** → **I2**
3. **$r0**: **I4** → **I1**
4. **$r1**: **I3** → **I5**

Al no haber *forwarding*, cuando hay una dependencia RAW debe esperarse al ciclo WB de la instrucción fuente para inicial el ciclo de la instrucción destino. La Tabla 2 muestra el correspondiente diagrama de tiempos.

- La primera instrucción se detiene 98 ciclos (80+14+4) en el fetch por ser un fallo en toda la jerarquía de memoria.
- La lectura de datos de la primera instrucción es un fallo de lectura y requiere 98 ciclos también.
- La segunda instrucción es un acierto y requiere cuatro ciclos para realizar la captación de la caché L1I.
- La instrucción I3 es un acierto y requiere cuatro ciclos para realizar la captación de la caché L1I.
- La escritura de datos de la instrucción I3 es un fallo de escritura y requiere 98 ciclos.
- La instrucción I4 es un acierto y requiere cuatro ciclos para realizar la captación de la caché L1I.
- La instrucción I4 no puede iniciar su ciclo de memoria hasta que no termine el acceso a memoria de I3.
- La instrucción I5 es un acierto y requiere cuatro ciclos para realizar la captación de L1I.
- La instrucción I5 no puede iniciar su ciclo de ejecución hasta que no termine la ejecución de I4.
- La instrucción I6 es un fallo y requiere 98 ciclos para realizar la captación de L1I.
- La instrucción I7 (la siguiente a la bnez) no puede comenzar la captación hasta que no se libera la unidad de fetch.
- Si bien se conoce la dirección de salto al final de la etapa de decodificación de I6, el sentido de salto (tomar o no tomar) no se conoce hasta el final de la etapa de ejecución.

En total se requieren 310 ciclos.

Instrucción	1–98	99	100	101	102	103–198	199	200	201	202	203	204	205	206	207	208	209	210
I1: lw $r2, 0($r0)	IF	ID	EX	M	M	M	WB											
I2: addi $r3, $r2, 20		IF1	IF2	IF3	IF4	–	ID	EX	M	WB								
I3: sw $r3, 0($r1)							IF1	IF2	IF3	IF4	ID	EX	M	M	M	M	M	M
I4: addi $r0, $r0, 4											IF1	IF2	IF3	IF4	ID	EX	–	–
I5: addi $r1, $r1, 4															IF1	IF2	IF3	IF4

Instrucción	211	212	213	214	215–302	303	304	305	306	307	308	309	310	311	312
I1: lw $r2, 0($r0)															
I2: addi $r3, $r2, 20															
I3: sw $r3, 0($r1)	M	M	M	M	M	WB									
I4: addi $r0, $r0, 4	–	–	–	–	–	M	WB								
I5: addi $r1, $r1, 4	ID	–	–	–	–	EX	M	WB							
I6: bnez $r2, bucle	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	ID	EX	M	
I7: ?												IF	flush		
I1: lw $r2, 0($r0)														IF	

Cuadro 2: Diagrama de tiempos de la pimera iteración del ejercicio 2.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: lw $r2, 0($r0)	IF	IF	IF	IF	ID	EX	M	M	M	M	WB										
I2: addi $r3, $r2, 20					IF	IF	IF	IF	–	–	ID	EX	M	WB							
I3: sw $r3, 0($r1)											IF	IF	IF	IF	ID	EX	M	M	M	M	WB
I4: addi $r0, $r0, 4															IF	IF	IF	IF	ID	EX	M
I5: addi $r1, $r1, 4																			IF	IF	IF

Instrucción	22	23	24	25	26	27	28	29	30
I1: lw $r2, 0($r0)									
I2: addi $r3, $r2, 20									
I3: sw $r3, 0($r1)									
I4: addi $r0, $r0, 4	WB								
I5: addi $r1, $r1, 4	IF	ID	EX	M	WB				
I6: bnez $r2, bucle		IF	IF	IF	IF	ID	EX	M	WB
I7: ?						IF	flush		
I1: lw $r2, 0($r0)								IF	

Cuadro 3: Diagrama de tiempos de la segunda iteración del ejercicio 2.

Diagrama de tiempos para segunda iteración La Tabla 3 muestra el correspondiente diagrama de tiempos. En total se requieren 28 ciclos de reloj.

Ejercicio 3 *Examen de octubre de 2014.*

Sea el siguiente fragmento de código:

```
bucle:  lw $f0, 0($r1)
        lw $f2, 0($r2)
        mul.f $f4, $f0, $f2
        add.d $f6, $f6, $f4
        addi $r1, $r1, 4
        addi $r2, $r2, 4
        sub $r3, $r3, 1
        bnez $r3, bucle
```

1. Haga una lista con todas las posibles dependencia de datos, sin considerar una estructura específica de la arquitectura segmentada. Para cada dependencia debe indicar, registro, instrucción de origen, instrucción de destino y tipo de dependencia.
2. Elabore un diagrama de tiempos para una arquitectura MIPS con un pipeline en 5 etapas, con las siguientes consideraciones:
   - No hay hardware de forwarding.
   - La arquitectura permite que una instrucción escriba en un registro y otra instrucción lea ese mismo registro sin problemas.
   - Las bifurcaciones se tratan vaciando el pipeline.
   - Las referencias a memoria requieren un ciclo de reloj.
   - La dirección efectiva de salto se calcula en la etapa de ejecución.
3. Determine cuantos ciclos se necesitan para ejecutar N iteraciones del bucle.
4. Elabore un diagrama de tiempos para una arquitectura MIPS con un pipeline en 5 etapas con las siguientes consideraciones:
   - Hay hardware completo de forwarding.
   - Asuma que las bifurcaciones se tratan prediciendo todos los saltos como tomados.
5. Determine cuantos ciclos se necesitan para ejecutar N iteraciones del bucle en las condiciones del apartado 4.

Solución 3

Dependencias de datos Si se numeran las instrucciones desde **I1** (primera instrucción) hasta **I8** (última instrucción) se tienen las siguientes dependencias:

- **$f0**: **I1** → **I3** (RAW)
- **$f2**: **I2** → **I3** (RAW)
- **$f4**: **I3** → **I4** (RAW)
- **$r3**: **I7** → **I8** (RAW)

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: lw $f0, 0($r1)	ID	ID	EX	M	WB																
I2: lw $f2, 0($r2)		IF	ID	EX	M	WB															
I3: mul.f $f4, $f0, $f2			IF	–	–	ID	EX	M	WB												
I4: add.d $f6, $f6, $f4						IF	–	–	ID	EX	M	WB									
I5: addi $r1, $r1, 4									IF	ID	EX	M	WB								
I6: addi $r2, $r2, 4										IF	ID	EX	M	WB							
I7: sub $r3, $r3, 1											IF	ID	EX	M	WB						
I8: bnez $r3, bucle												IF	–	–	ID	EX	M	WB			
I9: (sig a bnez)															IF						
I1: lw $f0, 0($r1)																	IF	ID	EX	M	WB

Cuadro 4: Primer diagrama de tiempos del ejercicio 3.

Primer diagrama de tiempos Al no haber forwarding, cuando hay una dependencia RAW debe esperarse al ciclo WB de la instrucción fuente para inicial el ciclo de la instrucción destino. La Tabla 4 muestra el correspondiente diagrama de tiempos.

- La instrucción I3, tiene una detención hasta que I2 haya escrito el valor leído en **$f2**. Se puede realizar la lectura de banco de registros en el mismo ciclo en que I2 escribe en el banco de registros (ciclo 6).
- La instrucción I4 no puede comenzar la captación hasta que no se libera la unidad de fetch (ciclo 6).
- La instrucción I4, tiene una detención hasta que I3 haya escrito el valor calculado en **$f4**. Se puede realizar la lectura de banco de registros en el mismo ciclo en que I3 escribe en el banco de registros (ciclo 9).
- La instrucción I5 no puede comenzar la captación hasta que no se libere la unidad de fetch (ciclo 9).
- La instrucción I8 tiene una detención hasta que I7 haya escrito el valor **$r3**. Se puede realizar la lectura de banco de registros en el mismo ciclo en que I7 escribe en el banco de registros (ciclo 15).
- La instrucción I9 (la siguiente a la bnez) no puede comenzar la captación hasta que no se libera la unidad de fetch (ciclo 15).
- Si bien se conoce la dirección de salto al final de la etapa de decodificación de I8, el sentido de salto (tomar o no tomar) no se conoce hasta el final de la etapa de ejecución Por tanto la captación se repite en el ciclo 17.

Primera estimación de ciclos Para determinar el número de ciclos se debe determinar, cuántos ciclos se requieren para cualquier iteración y cuántos para la última iteración.

El coste de una iteración distinta de la última se obtiene determinando el número de ciclos desde que se inicia la ejecución de I1 hasta que esta se vuelve a iniciar. Esto son 16 ciclos.

El coste de la última iteración se obtiene determinando el número de ciclos hasta que se finaliza la ejecución de la instrucción I8. Estos son 18 ciclos.

$$Coste = 16 \cdot n + 2$$

.

Segundo diagrama de tiempos Ahora se permite forwarding siempre que sea posible y no hay que esperar a la etapa de WB. La Tabla 5 muestra el correspondiente diagrama de tiempos.

- La instrucción I3 puede iniciar ejecución tras la etapa de memoria de I2 (ciclo 6) gracias al forwarding.
- La instrucción I4 no puede iniciar la decodificación hasta que no se ha liberado la unidad de decodificación (ciclo 6).
- La instrucción I4 puede iniciar ejecución tras la etapa de ejecución de I3 (ciclo 7) gracias al forwarding.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: lw $f0, 0($r1)	ID	ID	EX	M	WB																
I2: lw $f2, 0($r2)		IF	ID	EX	M	WB															
I3: mul.f $f4, $f0, $f2			IF	ID	–	EX	M	WB													
I4: add.d $f6, $f6, $f4				IF	–	ID	EX	M	WB												
I5: addi $r1, $r1, 4						IF	ID	EX	M	WB											
I6: addi $r2, $r2, 4							IF	ID	EX	M	WB										
I7: sub $r3, $r3, 1								IF	ID	EX	M	WB									
I8: bnez $r3, bucle									IF	ID	EX	M	WB								
I1: lw $f0, 0($r1)										IF	ID	EX	M	WB							

Cuadro 5: Segundo diagrama de tiempos del ejercicio 3.

- La instrucción I5 no puede iniciar la captación hasta que no se libera la unidad de fetch (ciclo 6).
- La instrucción I8 no puede iniciar la decodificación hasta que no se ha calculado el valor de **$r3** (ciclo 10) y se le pasa mediante forwarding (ciclo 11).

Segunda estimación de ciclos El coste de una iteración distinta de la última es de 10 ciclos. La última iteración requiere 14 ciclos.

$$Coste = 10 \cdot n + 4$$

Ejercicio 4 *Octubre de 2014.*

Sea el siguiente fragmento de código:

```
bucle:  lw $f0, 0($r1)
        lw $f2, 0($r2)
        sub.f $f4, $f0, $f2
        mul.d $f4, $f4, $f4
        add.d $f6, $f6, $f4
        addi $r1, $r1, 4
        addi $r2, $r2, 4
        sub $r3, $r3, 1
        bnez $r3, bucle
```

1. Haga una lista con toda la posible dependencia de datos, sin considerar una estructura específica de la arquitectura segmentada. Para cada dependencia debe indicar, registro, instrucción de origen, instrucción de destino y tipo de dependencia.
2. Elabore un diagrama de tiempos para una arquitectura MIPS con un pipeline en 5 etapas, con las siguientes consideraciones:
   - No hay hardware de forwarding.
   - La arquitectura permite que una instrucción escriba en un registro y otra instrucción lea ese mismo registro sin problemas en el mismo ciclo de reloj.
   - Las bifurcaciones se tratan vaciando el pipeline.
   - Las referencias a memoria requieren un ciclo de reloj.
   - La dirección efectiva de salto se calcula en la etapa de ejecución.
3. Determine cuantos ciclos se necesitan para ejecutar N iteraciones del bucle.
4. Elabore un diagrama de tiempos para una arquitectura MIPS con un pipeline en 5 etapas con las siguientes consideraciones:
   - Hay hardware completo de forwarding.
   - Asuma que las bifurcaciones se tratan prediciendo todos los saltos como tomados.
5. Determine cuantos ciclos se necesitan para ejecutar N iteraciones del bucle en las condiciones del apartado 4.

Solución 4

Depdencias de datos Si se numeran las instrucciones desde **I1** (primera instrucción) hasta **I9** (última instrucción) se tienen las siguientes dependencias:

- **$f0**: **I1** → **I3** (RAW)
- **$f2**: **I2** → **I3** (RAW)
- **$f4**: **I3** → **I4** (RAW, WAW)
- **$f4**: **I4** → **I5** (RAW)
- **$r3**: **I8** → **I9** (RAW)

Primer diagrama de tiempos Al no haber forwarding, cuando hay una dependencia RAW debe esperarse al ciclo WB de la instrucción fuente para inicial el ciclo de la instrucción destino. La Tabla 6 muestra el correspondiente diagrama de tiempos.

- La instrucción I3, tiene una detención hasta que I2 haya escrito el valor leído en **$f2**. Se puede realizar la lectura de banco de registros en el mismo ciclo en que I2 escribe en el banco de registros (ciclo 6).
- La instrucción I4 no puede comenzar la captación hasta que no se libera la unidad de fetch (ciclo 6).
- La instrucción I4, tiene una detención hasta que I3 haya escrito el valor calculado en **$f4**. Se puede realizar la lectura de banco de registros en el mismo ciclo en que I3 escribe en el banco de registros (ciclo 9).
- La instrucción I5 no puede comenzar la captación hasta que no se libere la unidad de fetch (ciclo 9).
- La instrucción I5, tiene una detención hasta que I4 haya escrito el valor calculado de **$f4**. Se puede realizar la lectura de banco de registros en el mismo ciclo en que I4 escribe en el banco de registros (ciclo 12).
- La instrucción I6 no puede comenzar la captación hasta que no se libere la unidad de fetch (ciclo 12).
- La instrucción I9 tiene una detención hasta que I8 haya escrito el valor **$r3**. Se puede realizar la lectura de banco de registros en el mismo ciclo en que I7 escribe en el banco de registros (ciclo 18).
- La instrucción I10 (la siguiente a la bnez) no puede comenzar la captación hasta que no se libera la unidad de fetch (ciclo 18).
- Si bien se conoce la dirección de salto al final de la etapa de decodificación de I9, el sentido de salto (tomar o no tomar) no se conoce hasta el final de la etapa de ejecución Por tanto la captación se repite en el ciclo 20.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: lw **$f0**, 0(**$r1**)	IF	ID	EX	M	WB																
I2: lw **$f2**, **0**(**$r2**)		IF	ID	EX	M	WB															
I3: sub.f **$f4**, **$f0**, **$f2**			IF	–	–	ID	EX	M	WB												
I4: mul.d **$f4**, **$f4**, **$f4**						IF	–	–	ID	EX	M	WB									
I5: add.d **$f6**, **$f6**, **$f4**									IF	–	–	ID	EX	M	WB						
I6: addi **$r1**, **$r1**, **4**												IF	ID	EX	M	WB					
I7: addi **$r2**, **$r2**, **4**													IF	ID	EX	M	WB				
I8: sub **$r3**, **$r3**, **4**														IF	ID	EX	M	WB			
I9: bnez **$r3**, **bucle**															IF	–	–	ID	EX	M	WB
I10: (siguiente a **bnez**)																		IF	–		
I1: lw **$f0**, 0(**$r1**)																				IF	ID

Cuadro 6: Primer diagrama de tiempos del ejercicio 4.

Primera estimación de ciclos Para determinar el número de ciclos se debe determinar, cuántos ciclos se requieren para cualquier iteración y cuántos para la última iteración.

El coste de una iteración distinta de la última se obtiene determinando el número de ciclos desde que se inicia la ejecución de I1 hasta que esta se vuelve a iniciar. Esto son 19 ciclos.

El coste de la última iteración se obtiene determinando el número de ciclos hasta que se finaliza la ejecución de la instrucción I9. Estos son 21 ciclos.

$$Coste = 19 \cdot n + 2.$$

Segundo diagrama de tiempos Ahora se permite forwarding siempre que sea posible y no hay que esperar a la etapa de WB. La Tabla 7 muestra el correspondiente diagrama de tiempos.

- La instrucción I3 puede iniciar ejecución tras la etapa de memoria de I2 (ciclo 6) gracias al forwarding.
- La instrucción I4 no puede iniciar la decodificación hasta que no se ha liberado la unidad de decodificación (ciclo 6).
- La instrucción I4 puede iniciar ejecución tras la etapa de ejecución de I3 (ciclo 7) gracias al forwarding.
- La instrucción I5 no puede iniciar la captación hasta que no se libera la unidad de fetch (ciclo 6).
- La instrucción I5 puede iniciar ejecución tras la etapa de ejecución de I4 (ciclo 8) gracias al forwarding.
- La instrucción I9 no puede iniciar la decodificación hasta que no se ha calculado el valor de **$r3** (ciclo 11) y se le pasa via forwarding (ciclo 12).

Segunda estimación de ciclos El coste de una iteración distinta de la última es de 11 ciclos. La última iteración requiere 15 ciclos.

$$Coste = 11 \cdot n + 4$$

Ejercicio 5 *Examen de junio de 2014.*

En un determinado procesador se dispone de la tabla 8 de latencia entre instrucciones:
En esta máquina se desea ejecutar el siguiente trozo de código:

```
BUCLE:  L.D   F0, 0(R1)
        L.D   F2, 0(R2)
        ADD.D F4, F0, F2
        S.D   F4, 0(R3)
        DADDUI R1, R1, #-8
        BNE   R1, R4, BUCLE
```

Inicialmente se tienen los siguientes valores:

- **R1**: Último elemento de primer array origen.
- **R2**: Último elemento de segundo array origen.
- **R3**: Último elemento de array destino.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I1: lw $f0, 0($r1)	IF	ID	EX	M	WB																
I2: lw $f2, 0($r2)		IF	ID	EX	M	WB															
I3: sub.f $f4, $f0, $f2			IF	–	ID	EX	M	WB													
I4: mul.d $f4, $f4, $f4				IF	–	ID	EX	M	WB												
I5: add.d $f6, $f6, $f4						IF	ID	EX	M	WB											
I6: addi $r1, $r1, 4							IF	ID	EX	M	WB										
I7: addi $r2, $r2, 4								IF	ID	EX	M	WB									
I8: sub $r3, $r3, 4									IF	ID	EX	M	WB								
I9: bnez $r3, bucle										IF	–	ID	EX	M	WB						
I1: lw $f0, 0($r1)												IF	ID	EX	M	WB					

Cuadro 7: Segundo diagrama de tiempos del ejercicio 4.

Cuadro 8: Latencia entre instrucciones

Instrucción que produce el resultado	Instrucción que usa el resultado	Latencia
Operación ALU FP	Otra operación ALU FP	6
Operación ALU FP	Almacenar doble	3
Cargar doble	Operación ALU FP	2
Cargar doble	Almacenar doble	0

- **R4**: Precalculado. Tal que **8(R4)** sea el primer elemento del primer array origen.

Todos los arrays tienen un tamaño de 4000 elementos.
Se pide:

1. Determine cuántos ciclos se requiere para ejecutar todas las iteraciones del bucle si no se realiza ninguna modificación.

2. Determine cuántos ciclos se requiere para ejecutar todas las iteraciones si se realiza planificación del bucle.

3. Determine cuántos ciclos se requiere para ejecutar todas las iteraciones si se desenrolla el bucle para cada dos iteraciones.

4. Determine cuántos ciclos se requiere para ejecutar todas las iteraciones si se desenrolla el bucle para cada cuatro iteraciones.

Solución 5

Apartado 1 La ejecución de una iteración del bucle sería:

```
L.D F0, 0(R1)
L.D F2, 0(R2)
<stall> x 2
ADD.D F4, F0, F2
<stall> x 3
S.D F4, 0(R3)
DADDUI R1, R1, #-8
BNE R1, R4, BUCLE
```

En total cada iteración requiere 11 ciclos para haber iniciado todas las instrucciones. Lo que da lugar a un total de 44,000 ciclos.

Apartado 2 La instrucción DADDUI podría adelantarse:

```
L.D F0, 0(R1)
L.D F2, 0(R2)
DADDUI R1, R1, #-8
<stall> x 1
ADD.D F4, F0, F2
<stall> x 3
S.D F4, 0(R3)
BNE R1, R4, BUCLE
```

En total cada iteración requiere ahora 10 ciclos. Lo que da lugar a un total de 40,000 ciclos

```
L.D F0, 0(R1)
L.D F2, 0(R2)
L.D F6, -8(R1)
L.D F8, -8(R2)
DADDUI R1, R1, #-16
ADD.D F4, F0, F2
ADD.D F10, F6, F8
<stall> x 2
S.D F4, 0(R3)
S.D F10, -8(R3)
BNE R1, R4, BUCLE
```

Se requieren un total de 12 ciclos por iteración. Lo que da lugar a un total de 24,000 ciclos

```
L.D F0, 0(R1)
L.D F2, 0(R2)
L.D F6, -8(R1)
L.D F8, -8(R2)
L.D F12, -16(R1)
L.D F14, -16(R2)
L.D F18, -24(R1)
L.D F20, -24(R2)
DADDUI R1, R1, #-32
ADD.D F4, F0, F2
ADD.D F10, F6, F8
ADD.D F16, F10, F12
ADD.D F22, F18, F20
S.D F4, 0(R3)
S.D F10, -8(R3)
S.D F16, -16(R3)
S.D F22, -24(R3)
BNE R1, R4, BUCLE
```

Se requieren un total de 18 ciclos por iteración. Lo que da lugar a un total de 18,000 ciclos

Ejercicio 6 *Examen de enero de 2014.*

En un determinado procesador se pretende ejecutar el siguiente segmento de código:

```
i0: lw $r4, 0($r1)
i1: lw $r5, 0($r2)
i2: add $r4, $r4, $r5
i3: sw $r4, 0($r3)
i4: addi $r1, $r1, 4
i5: addi $r2, $r2, 4
i6: addi $r3, $r3, 4
i7: bne $r3, $r0, i0
```

Asuma que el procesador tiene una arquitectura segmentada de 5 etapas (captación, decodificación, ejecución, memoria y post-escritura) sin envío adelantado (forwarding). Todas las operaciones se ejecutan en un ciclo por etapa, excepto:

- Las operaciones de carga y almacenamiento, que requieren un total de dos ciclos para la etapa de memoria (un ciclo adicional).
- Las instrucciones de salto, que requieren un ciclo adicional en la etapa de ejecución. Considere que estas instrucciones no cuentan con ningún tipo de predicción de saltos.

1. Determine los riesgos de datos RAW que presenta el código que tienen impacto en la ejecución del código.

2. Muestre un diagrama de tiempos con las fases de ejecución de cada instrucción para una iteración.
3. Determine cuantos ciclos requiere la ejecución de una iteración del bucle si no hay ningún tipo de predicción de saltos.
4. Proponga un desenrollamiento (*loop unrolling*) del bucle asumiendo que bucle se ejecuta 1000 iteraciones. Desenrolle con un factor de cuatro iteraciones.
5. Determine la aceleración o speedup obtenido mediante el desenrollamiento del apartado anterior.

Solución 6

Apartado 1

- **$r4**: **i0** → **i2**
- **$r5**: **i1** → **i2**
- **$r4**: **i2** → **i3**
- **$r3**: **i6** → **i7**

Apartado 2 La Tabla 9 muestra el correspondiente diagrama de tiempos.

- I0: Requiere dos ciclos de memoria.
- I1: No puede empezar etapa de memoria hasta que I0 no termina etapa de memoria. Requiere dos ciclos de memoria.
- I2: No puede empezar a decodificar hasta que I1 no hace WB.
- I3: No puede empezar captación hasta que se libera unidad por I2.
- I4: No puede empezar etapa de memoria hasta que I3 libera unidad de memoria.
- I5: No puede empezar etapa de ejecución hasta que I4 libera unidad de ejecución.
- I6: No puede empezar etapa de decodificación hasta que I5 libera unidad de decodificación.
- I7: No puede empezar decodificación hasta que I6 hace WB.

Apartado 3 Si se considera que la etapa de decodificación incluye un comparador sobre el banco de registros la siguiente instrucción a I7 puede comenzar después de que finalice la etapa de decodificación de I7 (esto es en el ciclo 19) por lo que cada iteración requiere 18 ciclos de reloj.

Si no se considera que la etapa de decodificación incluye un comparador, la comparación de dos registros debe hacerse con la ALU general y por tanto, no se podrá tomar la decisión hasta que haya finalizado la etapa de ejecución de I7 (en el ciclo 21). En este caso cada iteración requiere 20 ciclos de reloj.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
I0	IF	ID	EX	M1	M2	WB																
I1		IF	ID	EX	–	M1	M2	WB														
I2			IF	–	–	–	–	ID	EX	M	WB											
I3								IF	–	–	ID	EX	M1	M2	WB							
I4											IF	ID	EX	–	M	WB						
I5												IF	ID	–	EX	M	WB					
I6													IF	–	ID	EX	M	WB				
I7															IF	–	–	ID	X1	X2	M	WB

Cuadro 9: Primer diagrama de tiempos del ejercicio 6.

Apartado 4 A continuación se presenta una posible solución. Obsérvese, sin embargo, que son posibles soluciones más agresivas que podrían dar lugar a mejores *speedups*.

```
I0: lw $r4, 0($r1)
i1: lw $r5, 0($r2)
i2: lw $r6, 4($r1)
i3: lw $r7, 4($r2)
i4: lw $r8, 8($r1)
i5: lw $r9, 8($r2)
i6: lw $r10, 12($r1)
i7: lw $r11, 12($r2)
i8: add $r4, $r4, $r5
i9: add $r6, $r6, $r7
i10: add $r8, $r8, $r9
i11: add $r10, $r10, $r11
i12: sw $r4, 0($r3)
i13: sw $r6, 4($r3)
i14: sw $r8, 8($r3)
i15: sw $r10, 12($r3)
i16: addi $r3, $r3, 16
i17: addi $r2, $r2, 16
i18: addi $r1, $r1, 16
i19: bne $r3, $r0, i0
```

Apartado 5 La Tabla 10 muestra el correspondiente diagrama de tiempos.

Dependiendo del criterio elegido en el apartado 2, el número de ciclos para iteración desenrrollada será 33 o 35 y el número de ciclos por iteración será $\frac{33}{4} = 8{,}25$ o $\frac{35}{4} = 8{,}75$

Por tanto el speedup será:

$$S = \frac{18}{8{,}25} = 2{,}18$$

O bien:

$$S = \frac{20}{8{,}75} = 2{,}28$$

Ejercicio 7 *Examen de octubre de 2013.*

En un determinado procesador pretende ejecutar el siguiente segmento de código:

```
i0:     lw $r3, 0($r0)
i1:     lw $r1, 0($r3)
i2:     addi $r1, $r1, 1
i3:     sub $r4, $r3, $r2
i4:     sw $r1, 0($r3)
i5:     bnz $r4, i0
```

Asuma que el procesador tiene una arquitectura segmentada de 5 etapas (captación, decodificación, ejecución, memoria y post-escritura) sin envío adelantado. Todas las operaciones se ejecutan en un ciclo, excepto las operaciones de carga y almacenamiento que requieren dos ciclos adicionales de latencia en el acceso a memoria, y las instrucciones de salto que requieren un ciclo adicional de ejecución.

1. Determine los riesgos de datos RAW que presenta el código.
2. Muestre un diagrama tiempos con las fases de ejecución de cada instrucción para una iteración:
3. Determine cuantos ciclos requiere la ejecución de una iteración del bucle si no hay ningún tipo de predicción de saltos.
4. Determine cuantos ciclos requiere la ejecución de una iteración si se usa un predictor de saltos que predice siempre a tomado.

Solución 7

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
I0	IF	ID	EX	M1	M2	WB																
I1		IF	ID	EX	–	M1	M2	WB														
I2			IF	ID	–	EX	–	M1	M2	WB												
I3				IF	–	ID	–	EX	–	M1	M2	WB										
I4						IF	–	ID	–	EX	–	M1	M2	WB								
I5								IF	–	ID	–	EX	–	M1	M2	WB						
I6										IF	–	ID	–	EX	–	M1	M2	WB				
I7												IF	–	ID	–	EX	–	M1	M2	WB		
I8														IF	–	ID	–	EX	–	M	WB	
I9																IF	–	ID	–	EX	M	WB
I10																		IF	–	ID	M	WB
I11																				IF	ID	EX
I12																					F	D
I13																						F

Instrucción	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
I10	WB														
I11	M	WB													
I12	EX	M1	M2	WB											
I13	ID	EX	–	M1	M2	WB									
I14	IF	ID	–	EX	–	M1	M2	WB							
I15		IF	–	ID	–	EX	–	M1	M2	WB					
I16				IF	–	ID	–	EX	–	M	WB				
I17						IF	–	ID	–	EX	M	WB			
I18								IF	–	ID	EX	M	WB		
I19										IF	ID	EX	EX	M	WB

Cuadro 10: Segundo diagrama de tiempos del ejercicio 6.

Apartado 1

- $r3: i0 → i1
- $r1: i1 → i2
- $r1: i2 → i4
- $r4: I3 → i5

Apartado 2 La Tabla 11 muestra el correspondiente diagrama de tiempos.

- I0: Requiere tres ciclos de memoria
- I1: No puede empezar a decodificar hasta que no se hace WB de $r3. Requiere 3 ciclos de memoria.
- I2: No se puede captar hasta que no se libera la unidad por I1. No puede empezar a decodificar hasta que no se hace WB de $r1
- I4: No pude empezar a decodificar hasta que I2 no hace WB de $r1
- I5: No puede empezar captación hasta que no se inicia decodificación de I4. No puede empezar ciclo de memoria hasta que no termina el ciclo de memoria de I4.

Apartado 3 En ese caso la instrucción I0 de la siguiente iteración no puede captarse hasta el ciclo 18, por lo que una iteración requiere 18 ciclos.

Apartado 4 En ese caso, la predicción se realiza en la etapa de decodificación, que es cuando se sabe que se trata de una instrucción de salto, por lo que I0 comenzaría en el ciclo 16, y una iteración requiere 16 ciclos.

Ejercicio 8 *Examen de octubre de 2013.*

Se dispone del siguiente código escrito en el ensamblador de MIPS. Suponga que, antes de empezar la ejecución las instrucciones R3 y R5 contienen, respectivamente, las direcciones de memoria del primer y último elemento de un array de 9 entradas (Valor inicial de R1=0x010 y valor inicial de R5=0x018).

```
Loop:   LD      R4  0(R1)
        DIV     R2  R2  R4
        ADD     R1  R1  #1
        SUB     R5  R5  #1
        SD      R4  0(R5)
        SUB     R6  R1  R5
        BNEZ    R6  Loop
```

1. Indique todos los riesgos de datos RAW y WAW presentes en el código anterior.
2. Diagrama de temporización asumiendo que usamos un procesador segmentado de 5 etapas (*fetch*, *decode*, *execution*, *memory* y *write back*), se emite una instrucción por ciclo y **no se usa forwarding**. Asuma que existe congelación del pipe en el salto y que **hay 1 ciclo adicional por acceso a memoria en lectura (LD) (no ocurre en escrituras)**.
3. Número de ciclos que tardaría el bucle Loop (todas sus iteraciones) en ejecutarse.

Solución 8

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
I0	IF	ID	EX	M1	M2	M3	WB														
I1		IF	–	–	–	–	ID	EX	M1	M2	M3	WB									
I2							IF	–	–	–	–	ID	EX	M	WB						
I3												IF	ID	EX	M	WB					
I4													IF	–	ID	EX	M1	M2	M3	WB	
I5															IF	ID	E1	E2	–	M	WB

Cuadro 11: Diagrama de tiempos del ejercicio 7.

Apartado 1 Si se numeran las instrucciones de **I1** a **I7** (siendo **I1** la primera), se tienen los siguientes riesgos de tipo RAW.

- **R4**: **I1** → **I2**
- **R4**: **I1** → **I5**
- **R5**: **I4** → **I5**
- **R5**: **I4** → **I6**
- **R1**: **I3** → **I6**
- **R6**: **I6** → **I7**

Apartado 2 La Tabla 12 muestra el correspondiente diagrama de tiempos.

Apartado 3 Total ciclos de ejecución de una iteracción del bucle: 15

Como hay 5 iteraciones (valores de **r1** y **r5** respectivamente: (0x010, 0x018), (0x011, 0x017),(0x012, 0x016), (0x013, 0x015) y (0x014, 0x014)) y hay que esperar a que termine la instrucción **BNZ** en la última iteracción (que son 3 ciclos más) se tiene:

$$15 \text{ ciclos} \cdot 5 \text{ iteraciones} + 4 \text{ ciclos adicionales} = 79 \text{ ciclos}$$

Ejercicio 9 *Examen de octubre de 2013.*

Considere el siguiente fragmento de código:

```
Bucle:  LD R4,0(R2)
        LD R5,0(R3)
        ADD R6,R4,R5
        SD R6,0(R3)
        BNZ R6,Bucle
```

1. Número de ciclos necesarios para ejecutar una iteración del bucle en un procesador no segmentado.Las instrucciones de acceso a memoria tienen una latencia de 3 ciclos. La instrucción de salto tiene una latencia de 1 ciclo.
2. Identifique las dependencias de datos RAW existentes en el código.
3. Calcule el número de ciclos necesarios para ejecutar una iteración del bucle en un procesador segmentado en 5 etapas. Se usa la técnica de forwarding y la estrategia de predicción de salto Congelación del Pipeline. Complete el diagrama de temporización siguiente

Solución 9

Apartado 1 Estudiando por separado los distintos tipos de instrucciones:
3 Instrucciones de memoria · (1 emisión + 3 latencia) = 12 ciclos
1 Instrucción de ALU · 1 emisión = 1 ciclo
1 Instrucción de salto · (1 emisión + 1 latencia) = 2 ciclos
Total = 15 ciclos

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LD R4 0(R1)	IF	ID	EX	M	–	WB														
DIV R2 R2 R4		IF	–	–	–	ID	EX	M	WB											
ADD R1 R1 #1						IF	ID	EX	M	WB										
SUB R5 R5 #1							IF	ID	EX	M	WB									
SD R4 0(R5)								IF	–	–	ID	EX	M	WB						
SUB R6 R1 R5											IF	ID	EX	M	WB					
BNEZ R6 Loop												IF	–	–	ID	EX	M	WB		
LD R4 0(R1)															–	IF	ID	EX	M	WB

Cuadro 12: Diagrama de tiempos del ejercicio 8.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12
LD R4,0(R2)	IF	ID	EX	M	WB							
LD R5,0(R3)		IF	ID	EX	M	WB						
ADD R6,R4,R5			IF	ID	–	EX	WB					
SD R6,0(R3)				IF	–	ID	EX	M	WB			
BNZ R6, Bucle						IF	ID	EX	M	WB		
(**Next**)								IF	ID	EX	M	WB

Cuadro 13: Diagrama de tiempos del ejercicio 9.

Apartado 2 Se identifican las siguientes dependencias:

- **R4**: **LD** → **ADD**.
- **R5**: **LD** → **ADD**.
- **R6**: **ADD** → **SD**.

Apartado 3 La Tabla 13 muestra el correspondiente diagrama de tiempos.

Apartado 4

$$S = \frac{15 \cdot 2}{(7 \cdot 2) + 3} = \frac{30}{17} = 1{,}76$$

Ejercicio 10 *Examen de junio de 2013.*

Dado el siguiente código, donde cada instrucción como coste asociado un ciclo además de los indicados en la siguiente tabla. Suponga además que la máquina en la que ejecuta es capaz de emitir una instrucción por ciclo, salvo las esperas debidas a detenciones y que es un procesador segmentado con un único camino de datos (pipeline).

Debido a riesgos estructurales, las detenciones de una instrucción (stalls) se realizan siempre, independientemente de que haya o no dependencia de datos con las instrucciones siguientes. Inicialmente se tiene que R1=0, R2=24, R3=16.

```
Loop1:  LD F2, 0(R1)
        ADDD F2,F0,F2
Loop2:  LD F4,0(R3)
        MULTD F4,F0,F4
        DIVD F10,F4,F0
        ADDD F12,F10,F4
        ADDI R1,R1,#8
        SUB R18,R2,R1
        BNZ R18,Loop2
        SD F2,0(R3)
        ADDI R3, R3, #8
        SUB R20,R2,R3
        BNZ R20,Loop1
```

1. Calcular el número de ciclos necesarios para ejecutar una iteración del bucle exterior y treinta del bucle interior.
2. Desenrollar tres iteraciones del bucle interior, no desenrollar el bucle exterior y volver a realizar el cálculo anterior.

Cuadro 14: Costes adicionales por instrucción

Instrucción	Latencia
LD	3
SD	1
ADD	2
MULTD	4
DIVD	10
ADDI	0
SUB	0
BNZ	1

3. Comparar los resultados de tiempos (número de ciclos) del primer y segundo apartado. Justificar la respuesta.

Solución 10

Apartado 1 Análisis:

```
Loop1: LD F2, 0(R1)        1+3
ADD F2,F0,F2               1+2
Loop2: LD F4,0(R3)         1+3
MULTD F4,F0,F4             1+4
DIVD F10,F4,F0             1+10
ADDD F12,F10,F4            1+2
ADDI R1,R1,#8              1
SUB R18,R2,R1              1
BNZ R18,Loop2              1+1

Iteración interior         27

SD F2,0(R3)                1+1
ADDI R3,R3,#8              1
SUB R20,R2,R3              1
BNZ R20,Loop1              1+1

Iteración exterior:        13
```

Tres iteraciones interiores y una exterior $= 13 + (27 \cdot 3) = 94$ ciclos

Apartado 2 Análisis:

```
Loop1: LD F2, 0(R1)        1+3
ADD F2,F0,F2               1+2
Loop2: LD F4,0(R3)         1+3
LD F6,0(R3)                1+3
LD F8,0(R3)                1+3
MULTD F4,F0,F4             1+4
MULTD F6,F0,F6             1+4
MULTD F8,F0,F8             1+4
DIVD F10,F4,F0             1+10
DIVD F12,F6,F0             1+10
DIVD F14,F8,F0             1+10
ADDD F16,F10,F4            1+2
ADDD F18,F12,F6            1+2
ADDD F20,F14,F8            1+2
ADDI R1,R1,#24             1
SUB R18,R2,R1              1
BNZ R18,Loop2              1+1

Tres iteraciones interiores 73
```

```
SD  F2,0(R3)          1+1
ADDI R3,R3,#8         1
SUB  R20,R2,R3        1
BNZ  R20,Loop1        1+1
```

Tres iteraciones interiores y una exterior $= 13 + (73 \cdot 1) = 86$ ciclos

Apartado 3 Se puede ver que no hay una reducción importante de ciclos. Eso se debe a que el bucle interior tiene muy pocas iteraciones y que el peso del control del bucle (incremento índices y salto) es relativamente pequeño comparado con el resto de instrucciones.

Ejercicio 11 *Examen de enero de 2013.*

Se dispone del siguiente código escrito en el ensamblador de MIPS. En el mismo se lee y escribe la misma posición de memoria un número de veces. Inicialmente los valores de R1 y R2 son R1=1 y R2=1000.

```
        ADDI R3,R3,1
Loop:   LD  R4,0(16)
        MULT R5,R4,R4
        ADD  R5,R3,R5
        SD   R5,0(16)
        DADDI R3,R4,1
        DADDI R1,R1,1
        BNE  R1,R2,Loop
```

Se tiene un procesador segmentado tipo MIPS con las siguientes fases de ejecución: fetch, decodificación, ejecución, memoria y escritura. Dicho procesador emite una instrucción por ciclo y tiene capacidad de envío adelantado (*forwarding*). Todas las etapas del camino de datos de ejecutan en un ciclo salvo los siguientes casos: la operación SD utiliza un ciclo adicional para leer **el registro R5 del banco de registros**, la operación LD utiliza un ciclo adicional para escribir **el valor de memoria en el registro R4 del banco de registros** y las operaciones ADD y MULT utilizan un ciclo adicional para ejecutarse en ALU. Asumimos que la estrategia de predicción de salto es No Tomada.

1. Indique sólo los riesgos de datos WAW presentes en el código anterior indicando las instrucciones que lo causan y el registro. ¿En qué situación un riesgo WAW podría originar un resultado incorrecto del programa?

2. Diagrama de temporización asumiendo forwarding. Número de ciclos que tardaría el programa en ejecutarse.

Solución 11

Apartado 1

- **WAW**: **I4** con **I3** en **R5**.
- **WAW**: **I1** con **I6** en **R3**.

Se obtendría un resultado incorrecto si **I4** se ejecuta antes de **I3**. En aquellos procesadores segmentados donde las instrucciones pueden ser reordenadas.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ADDI R3 ,R3, 1	IF	ID	EX	M	WB											
LD R4 , 0(16)		IF	ID	EX	M	WB	WB									
MULT R5 ,R4 ,R4			IF	ID	–	EX	EX	M	WB							
ADD R5 ,R3 ,R5				IF	–	ID	–	EX	EX	M	WB					
SD R5 , 0(16)						IF	–	ID	ID	EX	M	WB				
DADDI R3 ,R4, 1								IF	–	ID		EX	M	WB		
DADDI R1 ,R1, 1										IF	ID	EX	M	WB		
BNE R1 ,R2 , Loop											IF	–	ID	EX	M	WB
(sig)													IF	IF	ID	EX

Cuadro 15: Diagrama de tiempos del ejercicio 11.

Apartado 2 La Tabla 15 muestra el correspondiente diagrama de tiempos.
Los ciclos requeridos para la ejecución del bucle se pueden dividir:

- Número de ciclos antes del bucle: 1.
- Número de ciclos por iteración del bucle: 12.
- Número de ciclos extra de la última instrucción del bucle: 3

El bucle se ejecuta un total de 999 veces. Por tanto el número de ciclos será:

$$\text{ciclos} = 1 + (12 \cdot 999) + 3 = 11902$$

Ejercicio 12 *Examen de junio de 2012.*

Dado el siguiente fragmento de código:

```
Bucle:  LD R4, 0(R2)
        LD R5, 0(R3)
        ADD R6, R4,R5
        SD R6, 0(R3)
        ADD R6, R6, R4
        ADDI R3, R3, #8
        SUB R20, R4, R3
        BNZ R20, Bucle
```

1. Indique las dependencias de datos existentes en el código. Indicando el dato que provoca cada dependencia.
2. Presente la temporización de esta secuencia para el pipeline RISC de 5 etapas sin hardware de *forwarding* o *bypassing*, pero asumiendo que una lectura y una escritura de un dato en el banco de registros se pueden realizar en el mismo ciclo de reloj (mediante el envío adelantado a través del banco de registros). Asuma que la bifurcación se trata vaciando el pipeline y que todos los accesos a memoria (incluidas el fetch) tardan dos ciclos. Justifique la respuesta.

Solución 12

Apartado 1 Se pueden identificar las siguientes dependencias de datos:

- **R4**: **I1** → **I3** (RAW).
- **R5**: **I2** → **I3** (RAW).
- **R6**: **I4** → **I3** (RAW).
- **R6**: **I5** → **I4** (WAR).
- **R6**: **I5** → **I3** (WAW).
- **R4**: **I5** → **I1** (RAW).
- **R3**: **I6** → **I4** (WAR).
- **R3**: **I7** → **I6** (RAW).
- **R20**: **I8** → **I7** (RAW).

Si una lectura y una escritura de un dato en el banco de registros se pueden realizar en el mismo ciclo de reloj, entonces se puede solapar la fase de decodificación (ID) y *write-back* (WB) en el mismo ciclo.

Apartado 2 La Tabla 16 muestra el correspondiente diagrama de tiempos.

Ejercicio 13 *Examen de mayo de 2012.*

Dada la siguiente sección de código:

```
        DADDUI R3, R1, #40      ; I1
BUCLE:  L.D F0, 0(R1)           ; I2
        L.D F2, 0(R2)           ; I3
        ADD.D F4, F0, F2        ; I4
        S.D F4, 0(R1)           ; I5
        DADDUI R1, R1, #8       ; I6
        DADDUI R2, R2, #8       ; I7
        BLE R1, R3, BUCLE       ; I8
```

Y teniendo en cuenta que se ejecuta en una máquina con las latencias adicionales entre instrucciones indicada por la tabla 17.

La instrucción de salto (*branch*) tiene una latencia de un ciclo y no tiene *delay slot*.

Suponga además que la máquina en la que ejecuta es capaz de emitir una instrucción por ciclo, salvo las esperas debidas a detenciones y que es un procesador segmentado con un único camino de datos (*pipeline*).

1. Determine todas las dependencias de datos.
2. Determine el número de ciclos total que se necesita para ejecutar la sección de código completa.
3. Modifique el código para reducir las detenciones mediante la técnica de planificación de bucle. Determine la aceleración obtenida respecto a la versión sin planificar (apartado 2).
4. Modifique el código realizando un desenrollamiento de bucle con dos iteraciones por bucle. Determine la aceleración obtenida respecto a la versión sin planificar (apartado 2).

Solución 13

Apartado 1 Se producen las siguientes dependencias:

- **I4** → **I2** (**F0**: RAW), **I4** → **I3** (**F2**: RAW)
- **I5** → **I4** (**F4**: RAW)
- **I6** → **I1** (**R1**: WAR), **I6** → **I2** (**R1**: WAR), **I6** → **I5** (**R1**: WAR)
- **I7** → **I3** (**R2**: WAR)
- **I8** → **I7** (**R2**: RAW), **I8** → **I1** (**R3**: RAW)

Apartado 2 A continuación se identifican las detenciones en la ejecución:

```
        DADDUI R3, R1, #40  ;Solamente la primera vez
BUCLE:  L.D F0, 0(R1)
        L.D F2, 0(R2)
        Detención
        Detención
        ADD.D F4, F0, F2
        Detención
        Detención
        Detención
        Detención
        S.D F4, 0(R1)
        DADDUI R1, R1, #8
        DADDUI R2, R2, #8
        BLE R2, R3, BUCLE
```

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
LD R4, 0(R2)	IF	–	ID	EX	M	–	WB																	
LD R5, 0(R3)		IF	–	ID	EX	M	–	WB																
ADD R6,R4,R5			IF	–	ID	EX	M	–	WB															
SD R6, 0(R3)				IF	–	–	–	IF	–	–	–	IF	–	–	–	–	–	–	–	–	–	–	–	
ADD R6,R6,R4					IF	–	–		IF	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
ADDI R3,R3, #8						IF	–			IF	–	–	–	–	–	–	–	–	–	–	–	–	–	
SUB R20,R4,R3							IF				IF	–	–	–	–	–	–	–	–	–	–	–	–	
BNZ R20, Bucle												IF	–	–	–	–	–	–	–	–	–	–	–	
(sig)													IF	–	–	–	–	–	–	–	–	–	–	

Cuadro 16: Diagrama de tiempos del ejercicio 12.

Cuadro 17: Latencias adicionales

Instrucción que produce el resultado (previa)	Instrucción que usa el resultado (posterior)	Latencia
Operación ALU FP	Operación ALU FP	5
Operación ALU FP	Cargar/almacenar double	4
Operación ALU FP	Instrucción de salto	4
Cargar doble	Operación ALU FP	2
Cargar doble	Cargar doble	1
Almacenar doble	Operación ALU FP	2

Se necesita un ciclo para el código de iniciación. Cada iteración necesita 13 ciclos. Como el bucle se ejecuta 5 veces, se tiene un total de:

$$1 + 5 \cdot 13 = 66$$

Debido a que no existe delay slot no es necesario añadir un ciclo de stall después de la instrucción de salto (**BLE**).

Apartado 3 A continuación se muestra el código modificado:

```
        DADDUI R3, R1, #40 ;Solamente la primera vez
BUCLE:  L.D F0, 0(R1)
        L.D F2, 0(R2)
        DADDUI R1, R1, #8
        DADDUI R2, R2, #8
        ADD.D F4, F0, F2
        Detención
        Detención
        Detención
        Detención
        S.D F4, -8(R1)
        BLE R1, R3, BUCLE
```

Ahora se requiere un tiempo total de $1 + 5 \cdot 11 = 56$. De nuevo, se necesita un ciclo para el código de iniciación. Cada iteración necesita 13 ciclos. Debido a que no existe delay slot no es necesario añadir un ciclo de stall después de la instrucción de salto (**BLE**).

$$\text{Speedup} = 66/56 = 1{,}17$$

Apartado 4 A continuación se muestra el código modificado:

```
        DADDUI R3, R1, #32  ;Solamente la primera vez
BUCLE:  L.D F0, 0(R1)
        L.D F2, 0(R2)
        Detención
        Detención
        ADD.D F4, F0, F2
        Detención
        Detención
        Detención
        Detención
        S.D F4, 0(R1)
        L.D F6, 8(R1)
        L.D F8, 8(R2)
        Detención
        Detención
        ADD.D F10, F6, F8
        Detención
        Detención
```

```
        Detención
        Detención
        S.D F10, 8(R1)
        DADDUI R1, R1, #16
        DADDUI R2, R2, #16
        BLE R2, R3, BUCLE
        L.D F0, 0(R1)
        L.D F2, 0(R2)
        Detención
        Detención
        ADD.D F4, F0, F2
        Detención
        Detención
        Detención
        Detención
        S.D F4, 0(R1)
```

En total ahora se tienen 2 iteraciones dentro del bucle. Además la quinta iteración del bucle original se hace ahora al final. El tiempo requerido es:

$$1 + 2 \cdot 23 + 10 = 57$$

Si además se planifica, se puede tener:

```
        DADDUI R3, R1, #32  ;Solamente la primera vez
BUCLE:  L.D F0, 0(R1)
        L.D F2, 0(R2)
        L.D F6, 8(R1)
        L.D F8, 8(R2)
        ADD.D F4, F0, F2
        Detención
        ADD.D F10, F6, F8
        DADDUI R1, R1, #16
        DADDUI R2, R2, #16
        S.D F4, -16(R1)
        Detención
        S.D F10, -8(R1)
        BLE R2, R3, BUCLE
        L.D F0, 0(R1)
        L.D F2, 0(R2)
        Detención
        Detención
        ADD.D F4, F0, F2
        Detención
        Detención
        Detención
        Detención
        S.D F4, 0(R1)
```

El tiempo pasa a:

$$1 + 2 \cdot 13 + 10 = 37\text{ciclos}$$