Arquitectura e Ingeniería de Computadores - 3r Grado Ing. Informática - ETSINF - UPV

Ejercicios y soluciones de la Unidad Temática 2. "Computadores segmentados" (2)

Gestión dinámica de instrucciones y ejecución especulativa

Ejercicio 2.15.

Un procesador MIPS aplica planificación dinámica de instrucciones con especulación para todas las instrucciones. Las instrucciones siguen las siguientes fases:

- **IF** Búsqueda de la instrucción.
- **I** Decodificación de la instrucción y lanzamiento a ejecución siguiendo el método de Tomasulo con especulación.
- **Ei** Ejecución en el operador correspondiente.
- **WB** Fase de transferencia de resultados por el bus interno.
- **C** Fase de confirmación. Escritura de resultados en el registro destino.

Todas las fases duran un ciclo de reloj, excepto la fase **Ei** cuya duración depende del operador. El procesador dispone de un predictor de saltos del tipo branch target buffer que obtiene la predicción en la fase **IF**.

Las características de las unidades funcionales del procesador son:

Tipo	Unidades	Latencia (ciclos)	Otras
Aritmética entera	1	1	4 estaciones de reserva
Multiplicación FP	1	3	Segmentada, 4 estaciones de reserva
Suma/Resta FP	1	2	Segmentada, 4 estaciones de reserva
Carga/Almac.	1	2	Segmentada 2 load buffers, 2 store buffers

Sobre dicho procesador se pretende ejecutar el siguiente fragmento de código:

```
      l.d f1,d(r0)
      l.d f3,n(r0)
loop: sub.d f4,f2,f1
      mul.d f1,f1,f4
      mul.d f3,f3,f4
      c.lt.d f1,f0
      bc1t loop      ; Salta si f1 < f0
      s.d f3,q(r0)
      ...
```

La instrucción `c.lt.d f1,f0` se evalúa en la unidad de suma/resta, y escribe su resultado en un registro interno (registro de estado de coma flotante). La instrucción `bc1t loop` salta si el registro de estado de coma flotante está a *true*, calculando la dirección de salto y evaluando la condición en la unidad entera.

En el momento de empezar a ejecutar este código, el *reorder buffer* y los operadores se encuentran vacíos, y el banco de registros y la memoria contienen los siguientes valores:

F0	F1	F2	F3	F4	d(r0)	n(r0)
1.0	0.0	2.0	0.0	0.0	1	0.25

Obsérvese que, para los datos indicados, sólo debe ejecutarse una vez cada instrucción del bucle. Sin embargo, supóngase que el predictor predice **incorrectamente** que la instrucción `bc1t loop` **salta**.

1. Dibuja un diagrama temporal en el que se indique qué fase está ejecutando cada instrucción en cada ciclo, desde el ciclo en que la instrucción `l.d f1,d(r0)` ejecuta la fase **IF** hasta el ciclo en que la instrucción `s.d f3, q(r0)` termina completamente.

2. Muestra el estado del *reorder buffer*, de las estaciones de reserva y buffers y del banco de registros al final del ciclo de reloj en que la instrucción `sub.d f4,f2,f1` de la primera iteración ejecuta la fase **C**.

Nota: Para representar las fases de ejecución $\mathbf{E}_{\mathrm{n}}$ en el diagrama temporal utilizar: EX para operaciones enteras, $\mathbf{A}_i$ para suma en coma flotante, $\mathbf{M}_i$ para multiplicación en coma flotante y **AC** $\mathbf{L}_i$ para las cargas y los almacenamientos; donde el subíndice indica el número de ciclos en ejecución (1, 2, ...).

Solución:

Diagrama temporal

PC	Instruc.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	l.d f1,d(r0)	IF	I	AC	L1	L2	WB	C																		
1	l.d f3,n(r0)		IF	I	AC	L1	L2	WB	C																	
2	sub.d f4,f2,f1			IF	I	–	–	A1	A2	WB	C															
3	mul.d f1,f1,f4				IF	I	–	–	–	–	M1	M2	M3	WB	C											
4	mul.d f3,f3,f4					IF	I	–	–	–	–	M1	M2	M3	WB	C										
5	c.lt.d f1,f0						IF	I	–	–	–	–	–	–	A1	A2	WB	C								
6	bc1t loop							IF	I	–	–	–	–	–	–	–	–	E1	WB	C						
2	sub.d f4,f2,f1								IF	I	–	–	–	–	–	A1	A2	WB	–	x						
3	mul.d f1,f1,f4									IF	I	–	–	–	–	–	–	–	M1	X						
4	mul.d f3,f3,f4										IF	I	–	–	–	–	–	–	–	X						
5	c.lt.d f1,f0											IF	I	–	–	–	–	–	–	x						
6	bc1t loop												IF	I	–	–	–	–	–	x						
2	sub.d f4,f2,f1													IF	I	–	–	–	–	x						
3	mul.d f1,f1,f4														IF	I	–	–	–	x						
4	mul.d f3,f3,f4															IF	I	–	–	x						
5	c.lt.d f1,f0																IF	I	–	x						
6	bc1t loop																	IF	I	x						
2	sub.d f4,f2,f1																		IF	X						
3	mul.d f1,f1,f4																			X						
7	s.d f3,16(r0)																				IF	I	AC	C	L1	L2

Estado en el ciclo 10:

ROB:

	busy	instr	completed	dest	value	pred	PC
0	NO	*l.d f1,d(r0)*	*SI*	*F1*	*1.00*		*0*
1	NO	*l.d f3,n(r0)*	*SI*	*F3*	*0.25*		*1*
2	NO	*sub.d f4,f2,f1*	*SI*	*F4*	*1.00*		*2*
3	SI	mul.d f1,f1,f4		F1			3
4	SI	mul.d f3,f3,f4		F3			4
5	SI	c.lt.d f1,f0		fpsr			5
6	SI	bc1t loop		loop		Salta	6
7	SI	sub.d f4,f2,f1		F4			2
8	SI	mul.d f1,f1,f4		F1			3
9	NO						
10	NO						
11	NO						
12	NO						
13	NO						
14	NO						
15	NO						

RS:

	busy	Op	Q1	V1	Q2	V2	rob	result
e1	SI	B	#5			0	#6	
e2	NO							
e3	NO							
e4	NO							
a1	SI	-		2.00	#3		#7	
a2	SI	$<$	#3			1.00	#5	
a3	NO							
a4	NO							
m1	SI	*		1.00		1.00	#3	
m2	SI	*		0.25		1.00	#4	
m3	SI	*	#3		#7		#8	
m4	NO							

LB:

	busy	Q1	V1	disp	addr	rob	result
l1	NO		*0*	*d*	*d*	*#0*	*1.00*
l2	NO		*0*	*n*	*n*	*#1*	*0.25*

SB:

	busy	Q1	V1	disp	addr	rob	Q2	V2	confirm
s1	NO								
s2	NO								

Registros

	F0	F1	F2	F3	F4	F5	F6	F7
rob		#8		#4	#7			
valor	1.00	1.00	2.00	0.25	1.00			

	R0	R1	R2	R3	R4	R5	R6	R7
rob								
valor								

□

Ejercicio 2.16. Un procesador compatible binario con el MIPS64 aplica gestión dinámica de instrucciones con especulación *hardware*. Las instrucciones siguen las siguientes fases:

- **IF** Búsqueda de la instrucción.
- **I** Decodificación de la instrucción y lanzamiento a ejecución siguiendo el método de Tomasulo con especulación.
- **Ei** Ejecución en el operador correspondiente.
- **WB** Fase de transferencia de resultados por el bus interno (el resultado transferido no estará disponible hasta el siguiente ciclo).
- **C** Fase de confirmación. Escritura de resultados en el registro destino.

Todas las fases duran un ciclo de reloj, excepto la fase **Ei** cuya duración depende del operador. El procesador dispone de un predictor de saltos perfecto que obtiene la predicción y dirección de destino en la fase **IF**.

Las características de las unidades funcionales del procesador son:

Tipo	Unidades	Latencia (ciclos)	Otras
Aritmética entera	1	1	3 estaciones de reserva (e1..e3)
Suma/Resta FP	1	2	Segmentada, 3 estaciones de reserva (a1..a3)
Multiplicación FP	1	4	Segmentada, 2 estaciones de reserva (m1..m2)
Carga/Almac.	1	2	No segmentada 3 load buffers (l1..l3), 3 store buffers (s1..s3)

Sobre dicho procesador se pretende ejecutar el siguiente fragmento de código:

```
        l.d      f0, A(r0)
salto:  l.d      f2, X(r1)
        mult.d   f4, f2, f0
        l.d      f6, Y(r1)
        sub.d    f4, f4, f6
        s.d      f4, Z(r1)
        dsub     r1, r1, #8
        bnez    r1, salto
```

En el momento de empezar a ejecutar este código, el *reorder buffer* y los operadores se encuentran vacíos. El valor inicial del registro R1 es 80. Los valores accedidos de los vectores X e Y se representarán como: `x1`, `x2`, ... e `y1`, `y2`, ... respectivamente. El valor que se encuentra en la dirección `Mem[r0 + A]` es `a`.

Mostrar el diagrama temporal de ejecución de las dos primeras iteraciones, y el estado de las estructuras de datos al final del ciclo en que el salto "`bnez r1, salto`" entra por **segunda** vez en la fase de búsqueda (IF). Calcular los MFLOPS que alcanzaría este código en un procesador con una frecuencia de reloj de 150 MHz.

Nota: Para representar las fases de ejecución **Ei** en el diagrama temporal utilizar: **EX** para operaciones enteras, $\mathbf{A}_i$ para la suma y resta flotante, $\mathbf{M}_i$ para la multiplicación flotante y **AC** $\mathbf{L}_i$ para las cargas y los almacenamientos; donde el subíndice indica el número de ciclos en ejecución (1, 2, ...).

Solución:

Diagrama temporal:

```
PC Instruc.       1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 0 l.d f0,A(r0)   IF I  AC L1 L2 WB C
 1 l.d f2,X(r1)      IF I  AC -  L1 L2 WB C
 2 mul.d f4,f2,f0       IF I  -  -  -  -  M1 M2 M3 M4 WB C
 3 l.d f6,Y(r1)            IF I  AC -  L1 L2 WB -  -  -  -  C
 4 sub.d f4,f4,f6             IF I  -  -  -  -  -  -  -  A1 A2 WB C
 5 s.d f4,Z(r1)                  IF I  AC -  -  -  -  -  -  -  -  C  L1 L2
 6 dsubi r1,r1,8                    IF I  E1 -  WB -  -  -  -  -  -  -  C
 7 bnez r1,salto                       IF I  -  -  E1 -  WB -  -  -  -  -  C
 1 l.d f2,X(r1)                           IF I  -  AC L1 L2 WB -  -  -  -  -  C
 2 mul.d f4,f2,f0                            IF I  -  -  -  -  M1 M2 M3 M4 WB -  C
 3 l.d f6,Y(r1)                                 IF I  AC -  L1 L2 WB -  -  -  -  -  C
 4 sub.d f4,f4,f6                                  IF I  -  -  -  -  -  -  -  A1 A2 WB C
 5 s.d f4,Z(r1)                                       IF I  AC -  -  -  -  -  -  -  -  C  L1 L2
 6 dsubi r1,r1,8                                         IF I  E1 -  WB -  -  -  -  -  -  -  C
 7 bnez r1,salto                                            IF I  -  -  E1 -  WB -  -  -  -  -  C
```

Estado de la unidad de ejecución en el ciclo 15:

ROB

	busy	instr	completed	dest	value	pred	PC
0	NO	*l.d f0,A(r0)*	*SI*	*F0*	*a*		*0*
1	NO	*l.d f2,X(r1)*	*SI*	*F2*	*x1*		*1*
2	NO	*mul.d f4,f2,f0*	*SI*	*F4*	*a*x1*		*2*
3	NO	*l.d f6,Y(r1)*	*SI*	*F6*	*y1*		*3*
4	SI	sub.d f4,f4,f6		F4			4
5	SI	s.d f4,Z(r1)		s1			5
6	SI	dsubi r1,r1,8	SI	R1	72		6
7	SI	bnez r1,salto	SI	salto	Salta	Salta	7
8	SI	l.d f2,X(r1)	SI	F2	x2		1
9	SI	mul.d f4,f2,f0		F4			2
10	SI	l.d f6,Y(r1)		F6			3
11	SI	sub.d f4,f4,f6		F4			4
12	SI	s.d f4,Z(r1)		s2			5
13	SI	dsubi r1,r1,8		R1			6
14	NO						
15	NO						

RS:

	busy	Op	Q1	V1	Q2	V2	rob	result
e1	SI	-		72		8	#13	
e2	NO	*B*		*72*		*0*	*#7*	*Salta*
e3	NO							
a1	SI	-		a		y1	#4	a*x1-y1
a2	SI	-	#9		#10		#11	
a3	NO							
m1	NO	***		*x1*		*a*	*#2*	*a*x1*
m2	SI	*		x2		a	#9	

LB:

	busy	Q1	V1	disp	addr	rob	result
l1	NO		*72*	*X*	*72+X*	*#8*	*x2*
l2	SI		72	Y	72+Y	#10	
l3	NO		*80*	*Y*	*80+Y*	*#3*	*y1*

SB:

	busy	Q1	V1	disp	addr	rob	Q2	V2	confirm
s1	SI		80	Z	80+Z	#5	#4		NO
s2	SI		72	Z	72+Z	#12	#11		NO
s3	NO								

Registros:

	F0	F1	F2	F3	F4	F5	F6	F7
rob			#8		#11		#10	
valor	5.00		x1		a*x1		y1	

	R0	R1	R2	R3	R4	R5	R6	R7
rob		#13						
valor		80						

Sin contar la primera instrucción, que no pertenece al bucle principal, el procesador propuesto lanza una instrucción por ciclo, luego tarda 7 ciclos en lanzar las instrucciones pertenecientes a una iteración. En cada iteración del bucle se realizan 2 operaciones de **coma flotante** (`multd` y `subd`). Así pues, los MFLOPs que puede alcanzar este procesador en la ejecución del código son:

$$R_\infty = \frac{2}{7}\ op./ciclo\ \cdot\ 150 \cdot 10^6\ ciclos/seg. = \frac{300}{7}\ \cdot\ 10^6\ op./seg. = 42{,}86\ MFLOPS$$

□

Ejercicio 2.17.

Se pretende utilizar el código correspondiente al bucle $\vec{y} = a \cdot \vec{x}$ para evaluar las prestaciones de cierto computador. Dicho computador posee un procesador similar al MIPS que aplica especulación hardware para todas las instrucciones. Las instrucciones siguen las siguientes fases:

- **IF** Búsqueda de la instrucción.
- **I** Decodificación de la instrucción y lanzamiento a ejecución siguiendo el método de Tomasulo con especulación.
- **Ei** Ejecución en el operador correspondiente.
- **WB** Fase de transferencia de resultados por el bus común de datos.
- **C** Fase de confirmación. Escritura de resultados en el registro destino. Comprobación de la predicción y cancelación de las intrucciones, en su caso.

Todas las fases duran un ciclo de reloj, excepto la fase **Ei** cuya duración depende del operador. El procesador dispone de un predictor de saltos de **dos** bits, del tipo *branch target buffer*, que obtiene la predicción en la fase **IF**.

Las características de las unidades funcionales del procesador son:

Tipo	Unidades	Latencia (ciclos)	Otras
Aritmética entera	1	1	3 estaciones de reserva
Multiplicación FP	1	3	Segmentada, 3 estaciones de reserva
Carga/Almac.	1	2	No segmentada 3 load buffers, 3 store buffers

El código que genera el compilador para dicho bucle es el siguiente:

```
loop:   l.d f2,x(r1)
        mult.d f2,f2,f0
        s.d f2, y(r1)
        dsub r1,r1,#8
        bnez r1,loop
        trap 0
```

El estado de los registros y la memoria, antes de la última iteración, es el siguiente:

Reg.	R1	F0	F2	Mem	x	x+8	y	y+8
Valor	8	3	2	Valor	100	102	10	12

1. Dibuja un diagrama instrucciones-tiempo en el que se muestre la ejecución de las instrucciones que se lanzan en la última iteración del bucle, hasta el ciclo en el que se busca la instrucción `trap 0`. Adviértase que el predictor predecirá **incorrectamente** la instrucción `bnez r1, loop` (predice que salta y finalmente no salta). Para representar las fases de ejecución $\mathbf{E}_{\mathrm{n}}$ en el diagrama temporal utilizar: EX para operaciones enteras, $\mathbf{A}_i$ para suma en coma flotante, $\mathbf{M}_i$ para multiplicación en coma flotante y **AC** $\mathbf{L}_i$ para las cargas y los almacenamientos; donde el subíndice indica el número de ciclos en ejecución (1, 2, ...). Por simplicidad, supóngase que el ROB y las estaciones de reserva están vacías antes de la última iteración y que no hay ninguna instrucción pendiente de ejecución.

2. En las mismas condiciones que el apartado anterior (*reorder buffer* y estaciones de reserva vacíos y sin instrucciones pendientes de ejecución), muestra cuál sería el estado del *reorder buffer*, el banco de registros y de la memoria al final del ciclo de reloj en el que la instrucción `s.d f2, y(r1)` ejecuta la fase **Commit**.

3. Supóngase ahora que el salto `bnez r1, loop` no se encuentra en la tabla al comenzar la ejecución del bucle, que el valor inicial de `R1` es 160. Indicar, justificando la respuesta, el tiempo de ejecución del bucle en ciclos. Considerar que el bucle acaba cuando la instrucción de salto de la última iteración llega a la etapa *Commit*.

Solución:

1. Diagrama instrucciones–tiempo

PC	Instruc.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	l.d f2,x(r1)	IF	I	AC	L1	L2	WB	C								
1	mul.d f2,f2,f0		IF	I	–	–	–	M1	M2	M3	WB	C				
2	s.d f2,y(r1)			IF	I	AC	–	–	–	–	–	–	C	L1	L2	
3	dsubi r1,r1,8				IF	I	E1	WB	–	–	–	–	–	C		
4	bnez r1,loop					IF	I	–	E1	WB	–	–	–	–	C	
0	l.d f2,x(r1)						IF	I	AC	L1	L2	WB	–	–	x	
1	mul.d f2,f2,f0							IF	I	–	–	–	M1	M2	X	
2	s.d f2,y(r1)								IF	I	AC	–	–	–	x	
3	dsubi r1,r1,8									IF	I	E1	WB	–	x	
4	bnez r1,loop										IF	I	–	E1	X	
0	l.d f2,x(r1)											IF	I	AC	X	
1	mul.d f2,f2,f0												IF	I	x	
2	s.d f2,y(r1)													IF	X	
3	dsubi r1,r1,8														X	
5	trap 0															IF

2. Estado al final del ciclo 12.

   ROB:

	busy	instr	completed	dest	value	pred	PC
0	NO	*l.d f2,x(r1)*	*SI*	*F2*	*102.00*		*0*
1	NO	*mul.d f2,f2,f0*	*SI*	*F2*	*306.00*		*1*
2	NO	*s.d f2,y(r1)*		*s1*			*2*
3	SI	dsubi r1,r1,8	SI	R1	0		3
4	SI	bnez r1,loop	SI	loop	No salta	Salta	4
5	SI	l.d f2,x(r1)	SI	F2	100.00		0
6	SI	mul.d f2,f2,f0		F2			1
7	SI	s.d f2,y(r1)		s2			2
8	SI	dsubi r1,r1,8	SI	R1	-8		3
9	SI	bnez r1,loop		loop		Salta	4
10	SI	l.d f2,x(r1)		F2			0

Registros:

	F0	F1	F2	F3	F4	F5	F6	F7
rob			#10					
valor	3.00		306.00					

	R0	R1	R2	R3	R4	R5	R6	R7
rob		#8						
valor		8						

Memoria:

Dir	Dato
x	100.00
x+8	102.00
...	...
y	10.00
y+8	12.00

3. Con el valor de R1 a 160, el bucle realizará 20 iteraciones. Cuando el predictor acierta, el tiempo para realizar una iteración es de 5 ciclos de reloj. Cada vez que se falla en la predicción (una vez, en la última iteración) o no se encuentra el salto en la tabla del predictor (en la primera iteración) se pierden 9 ciclos, por lo tanto:
   $T_{ejecucion} = 20 * 5 + 2 * 9 = 100 + 18 = 118$ ciclos

□

Lanzamiento múltiple de instrucciones

Ejercicio 2.18.

Se pretende utilizar el código correspondiente al bucle $\vec{y} = a\vec{x} + b\vec{y} + c$ para evaluar las prestaciones de cierto computador. Dicho computador posee un procesador superescalar de **dos vías** que aplica planificación dinámica de instrucciones con especulación para todas las instrucciones. Las instrucciones siguen las siguientes fases:

- **IF** Búsqueda de la instrucción.
- **I** Decodificación de la instrucción y lanzamiento a ejecución.
- **Ei** Ejecución en el operador correspondiente.
- **WB** Fase de transferencia de resultados por el bus interno.
- **C** Fase de confirmación. Escritura de resultados en el registro destino.

Todas las fases duran un ciclo de reloj, excepto la fase **Ei** cuya duración depende del operador. El procesador dispone de un predictor de saltos del tipo *branch target buffer* que obtiene la predicción al final de la fase **IF**. El tiempo de transferencia por los buses comunes de datos es de 1 ciclo de reloj.

Las características de las unidades funcionales del procesador son:

Tipo	Unidades	Latencia (ciclos)	Otras
Aritmética entera	2	1	6 estaciones de reserva
Multiplicación FP	1	4	Segmentada, 4 estaciones de reserva
Suma/Resta FP	1	2	Segmentada, 4 estaciones de reserva
Carga/Almac.	2	2	2 load buffers, 2 store buffers

El código que genera el compilador para dicho bucle es el siguiente:

```
loop:   l.d f2,x(r1)
        l.d f4,y(r2)
        mul.d f2,f2,f0
        mul.d f4,f4,f1
        add.d f6,f3,f2
        add.d f6,f6,f4
        s.d f6,y(r2)
        dsub r1,r1,#8
        dsub r2,r2,#8
        bnez r1,loop
        trap #0
```

El estado de los registros y la memoria es el siguiente:

R1	R2	F0	F1	F2	F3	F4	F5	F6	x+32	x+24	y+40	y+32
32	40	3.0	0.5	2.0	2.0	0.0	0.0	0.0	100	102	10	12

1. Dibuja un diagrama temporal en el que se muestre la ejecución de las instrucciones que se lanzan en las dos primeras iteraciones. Supóngase que el predictor predice **correctamente** que la instrucción `bnez r1, loop` **salta**. Se deberá indicar qué fase está ejecutando cada instrucción en cada ciclo. Para representar las fases de ejecución multiciclo $\mathbf{E}_{\mathrm{n}}$ en el diagrama temporal utilizar: EX para operaciones enteras, $\mathbf{A}_i$ para suma en coma flotante, $\mathbf{M}_i$ para multiplicación en coma flotante y **AC** $\mathbf{L}_i$ para las cargas y los almacenamientos; donde el subíndice indica el número de ciclos en ejecución (1, 2, ...).

2. Muestra el estado del *reorder buffer*, de las estaciones de reserva y buffers, del banco de registros y de memoria al final del ciclo de reloj en el que la instrucción `bnez r1,loop` de la segunda iteración ejecuta la fase **IF**.

Solución:

PC	Instruc.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	l.d f2,x(r1)	IF	I	AC	L1	L2	WB	C																		
1	l.d f4,y(r2)	IF	I	AC	L1	L2	WB	C																		
2	mul.d f2,f2,f0		IF	I	–	–	–	M1	M2	M3	M4	WB	C													
3	mul.d f4,f4,f1		IF	I	–	–	–	–	M1	M2	M3	M4	WB	C												
4	add.d f6,f3,f2			IF	I	–	–	–	–	–	–	–	A1	A2	WB	C										
5	add.d f6,f6,f4			IF	I	–	–	–	–	–	–	–	–	–	–	A1	A2	WB	C							
6	s.d f6,y(r2)				IF	I	AC	–	–	–	–	–	–	–	–	–	–	–	C	L1	L2					
7	dsubi r1,r1,8				IF	I	E1	WB	–	–	–	–	–	–	–	–	–	–	–	C						
8	dsubi r2,r2,8					IF	I	E1	WB	–	–	–	–	–	–	–	–	–	–	C						
9	bnez r1,loop					IF	I	–	E1	WB	–	–	–	–	–	–	–	–	–	–	C					
0	l.d f2,x(r1)						IF	I	AC	L1	L2	WB	–	–	–	–	–	–	–	–	C					
1	l.d f4,y(r2)						IF	I	–	AC	L1	L2	WB	–	–	–	–	–	–	–	–	C				
2	mul.d f2,f2,f0							IF	I	–	–	–	M1	M2	M3	M4	WB	–	–	–	–	C				
3	mul.d f4,f4,f1							IF	I	–	–	–	–	M1	M2	M3	M4	WB	–	–	–	–	C			
4	add.d f6,f3,f2								IF	I	–	–	–	–	–	–	–	A1	A2	WB	–	–	C			
5	add.d f6,f6,f4								IF	I	–	–	–	–	–	–	–	–	–	–	A1	A2	WB	C		
6	s.d f6,y(r2)									IF	I	AC	–	–	–	–	–	–	–	–	–	–	–	C	L1	L2
7	dsubi r1,r1,8									IF	I	E1	–	WB	–	–	–	–	–	–	–	–	–	C		
8	dsubi r2,r2,8										IF	I	E1	WB	–	–	–	–	–	–	–	–	–	C		
9	bnez r1,loop										IF	I	–	–	E1	WB	–	–	–	–	–	–	–	–	–	C
0	l.d f2,x(r1)											IF	I	–	AC	L1	L2	–	WB	–	–	–	–	–	–	C

El estado de la unidad en el ciclo 10 es el siguiente:

ROB:

	busy	instr	completed	dest	value	pred	PC
0	NO	*l.d f2,x(r1)*	*SI*	*F2*	*100.00*		*0*
1	NO	*l.d f4,y(r2)*	*SI*	*F4*	*10.00*		*1*
2	SI	mul.d f2,f2,f0		F2			2
3	SI	mul.d f4,f4,f1		F4			3
4	SI	add.d f6,f3,f2		F6			4
5	SI	add.d f6,f6,f4		F6			5
6	SI	s.d f6,y(r2)		s1			6
7	SI	dsubi r1,r1,8	SI	R1	24		7
8	SI	dsubi r2,r2,8	SI	R2	32		8
9	SI	bnez r1,loop	SI	loop	Salta	Salta	9
10	SI	l.d f2,x(r1)		F2			0
11	SI	l.d f4,y(r2)		F4			1
12	SI	mul.d f2,f2,f0		F2			2
13	SI	mul.d f4,f4,f1		F4			3
14	SI	add.d f6,f3,f2		F6			4
15	SI	add.d f6,f6,f4		F6			5
16	SI	s.d f6,y(r2)		s2			6
17	SI	dsubi r1,r1,8		R1			7

RS:

	busy	Op	Q1	V1	Q2	V2	rob	result
e1	SI	-		24		8	#17	
e2	NO	*-*		*40*		*8*	*#8*	*32*
e3	NO	*B*		*24*		*0*	*#9*	*Salta*
e4	NO							
e5	NO							
e6	NO							
a1	SI	+		2.00	#2		#4	
a2	SI	+	#4		#3		#5	
a3	SI	+		2.00	#12		#14	
a4	SI	+	#14		#13		#15	
m1	SI	*		100.00		3.00	#2	300.00
m2	SI	*		10.00		0.50	#3	
m3	SI	*	#10			3.00	#12	
m4	SI	*	#11			0.50	#13	

LB:

	busy	Q1	V1	disp	addr	rob	result
l1	SI		24	x	x+24	#10	102.00
l2	SI		32	y	y+32	#11	

SB:

	busy	Q1	V1	disp	addr	rob	Q2	V2	confirm
s1	SI		40	y	y+40	#6	#5		NO
s2	SI		32	y		#16	#15		NO

Registros:

	F0	F1	F2	F3	F4	F5	F6	F7
rob			#12		#13		#15	
valor	3.00	0.50	100.00	2.00	10.00			

	R0	R1	R2	R3	R4	R5	R6	R7
rob		#17	#8					
valor		32	40					

Memoria:

Dir	Dato
...	...
x+24	102.00
x+32	100.00
...	...
y+32	12.00
y+40	10.00

□

Ejercicio 2.19.

Se tiene un procesador superescalar compatible binario con el MIPS, capaz de buscar y lanzar a ejecución hasta dos instrucciones por ciclo de reloj. La frecuencia de reloj es 900 MHz, y posee instrucciones enteras y de coma flotante, aplicando gestión dinámica de instrucciones con especulación para todas las instrucciones. Las fases que atraviesan las instrucciones durante su ejecución son:

IF Fase de búsqueda de la instrucción.

I Fase de lanzamiento de la instrucción.

En Fase de ejecución en los operadores correspondientes.

WB Fase de transferencia de resultados por el bus interno (el resultado transferido no estará disponible hasta el siguiente ciclo).

C Fase de confirmación de la instrucción. Comprobación de las predicciones. Escritura en registros. Lanzamiento de las escrituras a memoria.

Todas las fases duran un ciclo de reloj, excepto la fase En que puede durar varios ciclos, en función del tipo de instrucción. La unidad de ejecución dispone de los siguientes operadores independientes:

Tipo	Unidades	Latencia (ciclos)	Otras
Entera/saltos	2	1	6 buffers
Carga/almac	2	2	Segmentada lineal, 6 buffers
Coma flotante	2	3	Segmentada lineal, 6 buffers

El procesador resuelve los riesgos de control mediante un predictor perfecto de cero ciclos de reloj de penalización. Cada uno de los bancos de registros tiene cuatro puertos de lectura y dos de escritura. Se necesita un ciclo de reloj para transferir un dato por los buses comunes de datos en la unidad de ejecución.

Sobre este procesador se pretende ejecutar el código obtenido de la compilación de $\vec{B}(i) := x + \vec{A}(i)$:

```
loop:  DSUB R1,R1,#8
       L.D F0, A(R1)
       ADD.D F4, F0, F2
```

```
S.D F4, B(R1)
BNEZ R1,loop
TRAP 0
```

Dibuja un diagrama en el que se indique, para cada instrucción y ciclo de reloj, qué fase de la instrucción se está ejecutando. Considera únicamente la primera y segunda iteración del bucle. Para representar las fases de ejecución multiciclo $\mathbf{E}_{\mathrm{n}}$ en el diagrama temporal utilizar: EX para operaciones enteras, $\mathbf{A}_i$ para suma en coma flotante, $\mathbf{M}_i$ para multiplicación en coma flotante y **AC** $\mathbf{L}_i$ para las cargas y los almacenamientos; donde el subíndice indica el número de ciclos en ejecución (1, 2, ...).

Suponiendo que la segunda iteración es representativa de lo que ocurre en el resto de iteraciones, calcula los CPI obtenidos y la velocidad en MFLOPS ejecutando ese fragmento de código.

Solución:

PC	Instruc.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	dsubi r1,r1,8	IF	I	E1	WB	C													
1	l.d f0,A(r1)	IF	I	–	–	AC	L1	L2	WB	C									
2	add.d f4,f0,f2		IF	I	–	–	–	–	–	A1	A2	A3	WB	C					
3	s.d f4,B(r1)		IF	I	–	AC	–	–	–	–	–	–	–	C	L1	L2			
4	bnez r1,loop			IF	I	E1	WB	–	–	–	–	–	–	–	C				
5	trap 0			IF	X														
0	dsubi r1,r1,8				IF	I	E1	WB	–	–	–	–	–	–	C				
1	l.d f0,A(r1)				IF	I	–	–	AC	L1	L2	WB	–	–	–	C			
2	add.d f4,f0,f2					IF	I	–	–	–	–	–	A1	A2	A3	WB	C		
3	s.d f4,B(r1)					IF	I	–	AC	–	–	–	–	–	–	–	C	L1	L2
4	bnez r1,loop						IF	I	E1	WB	–	–	–	–	–	–	–	C	
5	trap 0						IF	X											
0	dsubi r1,r1,8							IF	I	E1	WB	–	–	–	–	–	–	C	
1	l.d f0,A(r1)							IF	I	–	–	AC	L1	L2	WB	–	–	–	C

Cada iteración ejecuta 5 instrucciones y necesita 3 ciclos de reloj, realizando una operación en coma flotante. Por lo tanto:

CPI=$\frac{3}{5}$=0.6.

$v = \frac{1}{3} = 0{,}33$ op/ciclo @ 900 MHz = 297 MFLOPS.

□

Ejercicio 2.20. Cierto programa consume la mayor parte de su tiempo de ejecución llevando a cabo la siguiente operación sobre vectores:

$\vec{Y} = \vec{X} * \vec{Y} + b$

$\vec{X} = a * \vec{X}$

Dicho programa se pretende ejecutar sobre un procesador MIPS/S, que funciona a 3 GHz y aplica planificación dinámica de instrucciones basada en el algoritmo de Tomasulo con especulación *hardware*, siendo las etapas del ciclo de instrucción: IF (búsqueda de la instrucción), I (*Issue*), En (ejecución), WB (1 ciclo de transferencia por los buses comunes de datos) y C (*Commit*). El procesador es superescalar de dos vías y la cache de instrucciones (etapa IF) entrega a la etapa I dos instrucciones alineadas en una dirección par. En caso de que sólo se necesite acceder a una de ellas, la otra se cancela en la etapa I y no se decodifica.

El MIPS/S dispone de los operadores multiciclo segmentados mostrados en la tabla:

MIPS/S	
Uds. de carga/almacenamiento, segmentadas	2 operadores, 2 ciclos, 6 buffers de carga 6 buffers de almacenamiento
Sumadores, segmentados	2 operadores, 2 ciclos, 4 buffers
Multiplicadores, segmentados	2 operadores, 3 ciclos, 4 buffers
Ud. enteros/saltos	2 operadores, 1 ciclo, 6 buffers

y utiliza un predictor dinámico de saltos BTB de un bit que obtiene la predicción en la fase IF.

Suponed que el compilador ha ubicado las constantes en los registros `f0` y `f1` y ha generado el siguiente código para un juego de instrucciones semejante al MIPS:

```
        .text 0x400000
loop:   l.d f2,X(r1)
        l.d f3,Y(r1)
        mul.d f3,f3,f2
        add.d f3,f3,f1
        s.d f3,Y(r1)
        mul.d f4,f0,f2
        s.d f4,X(r1)
        dadd r1,r1,#8
        bne r1,r4,loop
        trap #0 ; Fin de programa
```

Antes de la última iteración, el buffer de reordenación está vacío y el estado de los registros y la memoria es el siguiente:

Reg.	r1	r4	f0	f1	f2	f3	Mem	x+16	x+24	x+32	y+16	y+24	y+32
Valor	24	32	10	20	0.5	-1	Valor	88	96	104	188	196	204

Se pide, justificando la respuesta:

1. Dibuja el diagrama instrucciones-tiempo correspondiente a la **última iteración** del bucle en el MIPS/S. Muestra únicamente las instrucciones que alcanzan la etapa *Commit*. Por simplicidad, supóngase que el ROB y las estaciones de reserva están vacías antes de la última iteración y que no hay ninguna instrucción pendiente de ejecución. Para representar las fases de ejecución $\mathbf{E}_{\mathrm{n}}$ en el diagrama temporal utilizar: EX para operaciones enteras, $\mathbf{A}_i$ para suma en coma flotante, $\mathbf{M}_i$ para multiplicación en coma flotante y **AC** $\mathbf{L}_i$ para las cargas y los almacenamientos; donde el subíndice indica el número de ciclos en ejecución (1, 2, ...).

2. Indica el número de ciclos consumido por una iteración cuando el predictor acierta y también cuando falla.

3. Indica en qué ciclo de reloj desde el comienzo de la iteración se consideraría que la variable `X` está actualizada en la memoria. Indica también en qué ciclo se buscará la instrucción `trap #0` que no se cancela.

4. Considera el ciclo de reloj en el que la instrucción `l.d f2,X(r1)` está en la fase *Commit* y muestra:

   a) El estado del registro f3 al final de dicho ciclo.

   b) El estado de los operadores de multiplicación al final de dicho ciclo.

   c) El contenido de la entrada del ROB que corresponde a la instrucción `bne r1,r4,loop` al final de dicho ciclo.

5. Si el bucle se ejecutase 500 veces, calcula el tiempo de ejecución en ciclos, los CPI obtenidos y la velocidad alcanzada en MFLOPS. Supón que, inicialmente, la instrucción de salto que cierra el bucle no se ha ejecutado antes.

Solución:

1. Dibuja el diagrama instrucciones-tiempo correspondiente a la **última iteración** del bucle en el MIPS/E. Muestra únicamente las instrucciones que alcanzan la etapa *Commit*.

```
PC Instruc.          1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18
 0 l.d f2,X(r1)     IF I  AC L1 L2 WB C
 1 l.d f3,Y(r1)     IF I  AC L1 L2 WB C
 2 mul.d f3,f3,f2      IF I  -  -  -  M1 M2 M3 WB C
 3 add.d f3,f3,f1      IF I  -  -  -  -  -  -  -  A1 A2 WB C
 4 s.d f3,Y(r1)           IF I  AC -  -  -  -  -  -  -  -  C  L1 L2
 5 mul.d f4,f0,f2         IF I  -  -  M1 M2 M3 WB -  -  -  -  C
 6 s.d f4,X(r1)              IF I  AC -  -  -  -  -  -  -  -  C  L1 L2
 7 daddi r1,r1,8             IF I  E1 WB -  -  -  -  -  -  -  -  C
 8 bne r1,r4,loop               IF I  -  E1 WB -  -  -  -  -  -  C
 9 trap 0                       IF X
 0 l.d f2,X(r1)                    IF I  AC L1 L2 WB -  -  -  -  x
 1 l.d f3,Y(r1)                    IF I  AC L1 L2 WB -  -  -  -  x
 2 mul.d f3,f3,f2                     IF I  -  -  -  M1 M2 M3 WB x
 3 add.d f3,f3,f1                     IF I  -  -  -  -  -  -  -  X
 4 s.d f3,Y(r1)                          IF I  AC -  -  -  -  -  x
 5 mul.d f4,f0,f2                        IF I  -  -  M1 M2 M3 WB x
 6 s.d f4,X(r1)                             IF I  AC -  -  -  -  x
 7 daddi r1,r1,8                            IF I  E1 WB -  -  -  x
 8 bne r1,r4,loop                              IF I  -  E1 WB -  x
 9 trap 0                                      IF X
 0 l.d f2,X(r1)                                   IF I  AC L1 L2 X
 1 l.d f3,Y(r1)                                   IF I  AC L1 L2 X
 2 mul.d f3,f3,f2                                    IF I  -  -  x
 3 add.d f3,f3,f1                                    IF I  -  -  x
 4 s.d f3,Y(r1)                                         IF I  AC x
 5 mul.d f4,f0,f2                                       IF I  -  x
 6 s.d f4,X(r1)                                            IF I  x
 7 daddi r1,r1,8                                           IF I  X
 8 bne r1,r4,loop                                             IF X
 9 trap 0                                                     IF X
 0 l.d f2,X(r1)                                                  X
 1 l.d f3,Y(r1)                                                  X
 8 bne r1,r4,loop                                                   IF X
 9 trap 0                                                           IF I
```

2. Indica el número de ciclos consumido por una iteración cuando el predictor acierta y también cuando falla.

   Si acierta: 5 ciclos

   Si falla: 16 ciclos

3. Indica en qué ciclo de reloj desde el comienzo de la iteración se consideraría que la variable `X` está actualizada en la memoria. Indica también en qué ciclo se buscará la instrucción `trap #0`.

   La variable `X` está actualizada en la memoria al final del ciclo 17

   La instrucción `trap #0` se buscará en el ciclo 17

4. Considera el ciclo de reloj en el que la instrucción `l.d f2,X(r1)` está en la fase *Commit* y muestra:

   Se trata del ciclo 7.

   a) El estado del registro f3 al final de dicho ciclo.
   Registros:

	F0	F1	F2	**F3**	F4	F5	F6	F7
rob			#9	#10	#5			
valor	10.00	20.00	96.00	196.00				
	R0	R1	R2	R3	R4	R5	R6	R7
rob		#7						
valor		24			32			

b) El estado de los operadores de multiplicación al final de dicho ciclo.
RS:

	busy	Op	Q1	V1	Q2	V2	rob	result
e1	NO	+		24		8	#7	32
e2	SI	B		32		32	#8	
e3	NO							
e4	NO							
e5	NO							
e6	NO							
a1	SI	+	#2			20.00	#3	
a2	NO							
a3	NO							
a4	NO							
m1	SI	*		196.00		96.00	#2	
m2	SI	*		10.00		96.00	#5	
m3	NO							
m4	NO							

c) El contenido de la entrada del ROB que corresponde a la instrucción `bnez r1,r4,loop`.
ROB:

	busy	instr	completed	dest	value	pred	PC
0	NO	*l.d f2,X(r1)*	*SI*	*F2*	*96.00*		*0*
1	NO	*l.d f3,Y(r1)*	*SI*	*F3*	*196.00*		*1*
2	SI	mul.d f3,f3,f2		F3			2
3	SI	add.d f3,f3,f1		F3			3
4	SI	s.d f3,Y(r1)		s1			4
5	SI	mul.d f4,f0,f2		F4			5
6	SI	s.d f4,X(r1)		s2			6
7	SI	daddi r1,r1,8	SI	R1	32		7
8	SI	bne r1,r4,loop		loop		Salta	8
9	SI	l.d f2,X(r1)		F2			0
10	SI	l.d f3,Y(r1)		F3			1

5. Si el bucle se ejecutase 500 veces, calcula el tiempo de ejecución en ciclos, los CPI obtenidos y la velocidad en MFLOPS.

   El predictor de saltos falla una vez al principio del bucle y otra más al final. Al principio del bucle, el salto no está en la tabla, por lo que se predice que "No Salta". Como es la primera iteración, el salto es efectivo y el predictor falla. Al final del bucle, el salto está en la tabla en estado "Salta". Como es la última iteración, el salto no es efectivo y el predictor falla. Por lo tanto:

$$T(50) = 498 * 5 + 2 * 16 = 2522 \text{ ciclos}$$

   Cada iteración ejecuta 9 instrucciones. Por tanto:

$$CPI = \frac{2522}{9 * 500} = 0,56 \text{ ciclos/instruc.}$$

   Cada iteración ejecuta 3 operaciones en coma flotante. Por tanto:

$$MFLOPS = \frac{3 * 500}{2522} = 0,595 \text{ op./ciclo @ 3 GHz} = 1786 \text{ MFLOPS.}$$

□

Ejercicio 2.21.

El siguiente código ensamblador para un procesador MIPS implementa la operación $\vec{Z} = a\vec{X} + b\vec{Y}$ condicionada al contenido del vector de máscara $\vec{M}$.

```
loop:   LD R3, M(R1)
        BEQZ R3, endif     ; Salta si M[i] == 0
        L.D F2, X(R1)      ; Valor inicial de R1 = 0
        L.D F3, Y(R1)
        MULT.D F3, F1, F3  ; F1 contiene b = 5.0
        MULT.D F2, F0, F2  ; F0 contiene a = 2.5
        ADD.D F4, F2, F3
        S.D F4, Z(R1)
endif:  DADD R1, R1, #8
        BNE R1, R2, loop   ; R2 contiene 64 * 8
```

El código se ejecuta sobre un procesador **superescalar de 2 vías** que aplica gestión dinámica de instrucciones con especulación hardware. Las instrucciones atraviesan durante su ejecución las siguientes etapas:

- IF - Búsqueda de instrucciones;
- I - Decodificación y lanzamiento de instrucciones;
- En - Ejecución en el operador correspondiente;
- WB - Transferencia de resultados;
- C - Etapa de confirmación.

Todas las etapas tienen una duración de 1 ciclo, excepto `Ei`, cuya duración varía de un operador a otro. Los riesgos de control se solucionan utilizando un predictor dinámico de saltos de tipo *Branch Target Buffer* de 1 bit. El predictor obtiene su predicción al final de la etapa IF y **actualiza su predicción en la etapa de confirmación**. Los bancos de registros poseen 4 puertos de lectura y 2 de escritura. El tiempo de transferencia a través de los buses de datos internos es de 1 ciclo.

Las características de los operadores existentes en el procesador son:

Tipo	Operadores	Latencia	Características
Entero	2	1	8 estaciones de reserva
Carga/Almacenamiento	1	2	Segmentado, 4 buffers de escritura y 4 de lectura
Suma/Resta CF	1	2	Segmentado, 4 estaciones de reserva
Multiplicación CF	1	4	Segmentado, 4 estaciones de reserva

Supongáse que el contenido inicial del vector $\vec{M}$ es $1, 0, 1, 0, ...$ Así pues, en la primera iteración el salto `BEQZ R3, endif` **efectivamente no saltará**. El estado inicial del predictor es:

Salto	Predicción
`BEQZ R3, endif`	Salta
`BNE R1, R2, loop`	Salta

Se solicita:

1. Representar el diagrama instrucciones-tiempo de la primera iteración del bucle (hasta que el salto `BNE R1, R2, loop` ejecuta la etapa Commit). Muestra sólo aquellas instrucciones que quepan en la hoja de respuesta. Para representar las fases de ejecución $\mathbf{E}_{\mathrm{n}}$ en el diagrama temporal utilizar: EX para operaciones enteras, $\mathbf{A}_i$ para suma en coma flotante, $\mathbf{M}_i$ para multiplicación en coma flotante y **AC** $\mathbf{L}_i$ para las cargas y los almacenamientos; donde el subíndice indica el número de ciclos en ejecución (1, 2, ...).

2. Considerando únicamente las instrucciones mostradas en el diagrama, indica el estado de los registros F2, F3 y F4 al final de la etapa de confirmación de la instrucción `L.D F2, X(R1)` de la primera iteración. El valor inicial de dichos registros es 2, 3 y 4, respectivamente. Mostrar el contenido de los campos `valor` y `rob`. El contenido del vector $\vec{X}$ es $100, 108, ...$ y el de $\vec{Y}$ es $10, 11, ...$

Solución:

1. Diagrama instrucciones-tiempo de la primera iteración del bucle.

```
PC Instruc.       1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 0 ld r3,M(r1)    IF I  AC L1 L2 WB C
 1 beqz r3,endif  IF I  -  -  -  -  E1 WB C
 8 daddi r1,r1,8     IF I  E1 WB -  -  -  x
 9 bne r1,r2,loop    IF I  -  -  E1 WB -  x
 0 ld r3,M(r1)          IF I  -  AC L1 L2 X
 1 beqz r3,endif        IF I  -  -  -  -  x
 8 daddi r1,r1,8           IF I  E1 WB -  x
 9 bne r1,r2,loop          IF I  -  -  E1 X
 0 ld r3,M(r1)                IF I  -  AC X
 1 beqz r3,endif              IF I  -  -  x
 8 daddi r1,r1,8                 IF I  E1 X
 9 bne r1,r2,loop                IF I  -  x
 0 ld r3,M(r1)                      IF I  x
 1 beqz r3,endif                    IF I  x
 8 daddi r1,r1,8                       IF X
 9 bne r1,r2,loop                      IF X
 0 ld r3,M(r1)                            X
 1 beqz r3,endif                          X
 2 l.d f2,X(r1)                              IF I  AC L1 L2 WB C
 3 l.d f3,Y(r1)                              IF I  -  AC L1 L2 WB C
 4 mul.d f3,f1,f3                               IF I  -  -  -  -  M1 M2 M3 M4 WB C
 5 mul.d f2,f0,f2                               IF I  -  -  -  M1 M2 M3 M4 WB -  C
 6 add.d f4,f2,f3                                  IF I  -  -  -  -  -  -  -  -  A1 A2 WB C
 7 s.d f4,Z(r1)                                    IF I  AC -  -  -  -  -  -  -  -  -  -  C  L1 L2
 8 daddi r1,r1,8                                      IF I  E1 WB -  -  -  -  -  -  -  -  -  C
 9 bne r1,r2,loop                                     IF I  -  -  E1 WB -  -  -  -  -  -  -  C
```

2. Estado de los registros al final de la etapa de confirmación de la instrucción `L.D F2, X(R1)` (ciclo 16).

Registro	F2	F3	F4
Valor	100	3	4
Reorder	#3	#2	#4

□

Ejercicio 2.22.

La figura siguiente muestra el diagramas instrucciones–tiempo simplificado correspondientes a la ejecución de dos aplicaciones A y B sobre un procesador superescalar de 3 vías. Se considera que ocurre un evento de alta latencia si se interrumpe la emisión de instrucciones durante dos o más ciclos:

0	1	2	3	4	5	6	7	8
		A						
A		A					A	A
A	A	A	A			A	A	A

0	1	2	3	4	5	6	7	8
				B				
	B			B	B	B		
B	B			B	B	B	B	B

Con el objeto de aumentar la utilización de los recursos del procesador, se plantea ejecutar múltiples hilos o *threads*.

1. Desde el punto de vista de cómo se comparten los recursos del procesador, explica cuáles son las diferencias principales entre un procesador multihilo de grano fino, de grano grueso y simultáneo *SMT*.

2. Muestra un posible diagrama instrucciones–tiempo, similar a los mostrados, correspondiente a la ejecución de las dos aplicaciones A y B sobre un procesador multihilo de grano fino, de grano grueso y simultáneo *SMT*.

Solución:

- Multihilo de grano fino.
  - Cambia de hilo cada ciclo de reloj, habitualmente round-robin.
  - Necesita poca sobrecarga en el cambio de hilo.
  - Oculta eventos de baja y alta latencia, pero puede retardar la ejecución de hilos que no tengan detenciones.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
				A		B							
A			B	A		B	B		B	A		A	
A	B	A	B	A	A	B	B	A	B	A	B	A	B

- Multihilo de grano grueso.
  - Cambia de hilo ante eventos de alta latencia.
  - Tolera mayor sobrecarga en el cambio de hilo.
  - No oculta eventos de baja latencia, pero no retarda la ejecución de hilos sin detenciones.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
		A							B				
A		A			B		A	A	B	B	B		
A	A	A	A	B	B	A	A	A	B	B	B	B	B

- *SMT*
  - Ejecuta simultáneamente varios hilos, tratando de ocupar todos los recursos ya disponibles en el procesador superescalar.
  - El número de hilos en ejecución varía a lo largo del tiempo según el ILP de cada hilo y los recursos disponibles.

0	1	2	3	4	5	6	7	8
B	B	A		B		B	B	B
A	B	A		B	B	B	A	A
A	A	A	A	B	B	A	A	A

□