borrador_correc_exam_teoria_3jul...

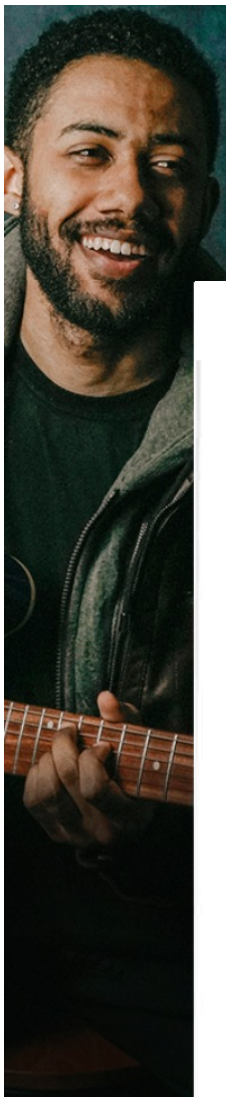
Jeremy

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada


ESCANEA Y PARTICIPA EN EL SORTEO DE UN ALIENWARE
LA PRIMERA RESIDENCIA GAMING EN EL MUNDO ABRE EN MADRID
GAMING RESIDENCES
gamingresidences.com
info@gamingresidences.com



2º curso / 2º cuatr. Grado en Ing. Informática	**Arquitectura de Computadores** Nombre y apellidos: **Puntuación**: 6 puntos **Duración**: 2:30 horas Grupo:

Cuestiones (1.75 puntos)

Cuestión 1. (0.1 puntos) Dado el bucle "`for i=1 to N do a(i)=k*(b(i)+c(i))`", en el que `a()`, `b()`, `c()`, y `k` son números en coma flotante, ¿cuántos GFLOPS consigue un computador que lo ejecuta en 2 segundos cuando $N=10^{11}$

Cuestión 2. (0.1 puntos) ¿Qué es NUMA?

Cuestión 3. (0.2 puntos) Si no puede predecir con precisión el volumen de trabajo de las tareas que constituyen un programa paralelo ¿Qué utilizaría, asignación de carga dinámica o estática? ¿Por qué? ¿Cuál de las dos conlleva más *overhead* o sobrecarga?

Cuestión 4. (0.1 puntos) ¿En qué consiste la comunicación colectiva *all-scatter*?

Cuestión 5. (0.2 puntos) Si el modelo de consistencia de memoria de un multiprocesador NO respeta los órdenes W→R y W->W (SÍ respeta todos los demás), e inicialmente `X=Y=Z=r1=0` (donde `X`, `Y` y `Z` son variables en memoria compartida y `r1` es un registro de P1), ¿qué valores podría tener `Y` al finalizar la ejecución del código paralelo de la derecha? Razonar respuesta.

```
P1:                          | P2:
  (1) while (Z==0) { };      | (a) X=1;
  (2) r1=X;                  | (b) Y=2;
  (3) Y=r1;                  | (c) Z=1;
```

Cuestión 6. (0.25 puntos) Para que la secuencia de instrucciones de la derecha implemente un cerrojo (`lock(k)`) en el que `k=1` significa que el cerrojo está cerrado y `k=0` que está abierto, ¿qué valores deben tener `r1`, `r2`, y `r3`, y qué debe hacer `compare&swap()`?

```
do
  b=r1; compare&swap(r2,b,k);
while (b==r3);
```

Cuestión 7. (0.2 puntos) Sin incluir instrucciones de salto, ¿cómo expresaría el siguiente código utilizando un repertorio típico de instrucciones con predicado en el que TODA instrucción se puede predicar: "`if ((r2>0)&&(r1>0) then r2=r2+r1 else r2=r2*r1;`" ¿(suponga que LOS PREDICADOS ESTÁN INICIALIZADOS A 0)

Cuestión 8. (0.2 puntos)¿En qué consiste la técnica de SEGMENTACIÓN SOFTWARE (SOFTWARE PIPELINING)? ¿Para qué se utiliza?

Cuestión 9. (0.2 puntos) En un procesador con registros de 64 bits (8 bytes), direccionamiento a nivel de byte, y modelo de consistencia que no garantiza W->R, ¿se podría adelantar `i+1` a `i` (`i` precede a `i+1` en el código)? ¿por qué?

```
(i)   | sw 0(r5), r2  // M(r5) ← r2
(i+1) | lw r4, 4(r5)  // r4 ←M(r5+4)
```

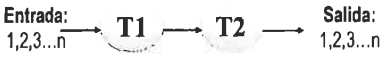
Cuestión 10. (0.2 puntos) Se quiere implementar un núcleo de procesamiento que sea capaz de emitir 4 instrucciones en paralelo cada ciclo de reloj de un flujo de instrucciones y ejecutar instrucciones que operan con vectores, además debe tener un consumo de potencia bajo. Para reducir el consumo de potencia se ha decidido que el paralelismo entre instrucciones lo detecte el compilador. Indicar qué arquitecturas cree que ha escogido el desarrollador para conseguir ajustarse a estos requisitos. ¿Cuándo (o si lo recuerda, en qué década) se extendieron comercialmente estas arquitecturas?

Ejercicios (4.25 puntos)

Ejercicio 1. (1.25 puntos) Los programas que ejecuta un procesador con una microarquitectura sin segmentación de cauce (pipelining) y una frecuencia de reloj de 2 GHz incluyen, por término medio, un 50% de instrucciones con operaciones con la ALU (de 4 ciclos por instrucción, CPI), un 20% de instrucciones BRANCH (5 CPI), un 20% de instrucciones de carga de datos de memoria, LOADs (4 CPIs), y un 10% de instrucciones de almacenamiento de datos en memoria, STOREs (3 CPIs). **(a)** ¿Cuántos nanosegundos tardaría en ejecutarse un programa con 2 millones de instrucciones? **(b)** ¿Es mejor utilizar una microarquitectura alternativa para ese procesador que reduce en uno los CPI de las instrucciones LOAD (3 CPI) y en otro ciclo las instrucciones BRANCH (4 CPI) a costa de aumentar el tiempo de ciclo de reloj un 20%. **(c)** Calcule los MIPS para la microarquitectura de partida y para la del apartado (b). ¿Cuál de las dos opciones tiene mayor número de MIPS? ¿Coincide con la mejor opción? **(d)** ¿Cuál es la máxima ganancia de velocidad que puede conseguirse con respecto a la microarquitectura de partida por mejoras en los CPI de las instrucciones BRANCH, sin que haya cambios en el tiempo de ciclo con respecto a la situación de partida? **(e)** Exprese el cálculo de dicha ganancia en términos de la ley de Amdahl.

(a),(b),(c),(d),(e) . 0.25

Ejercicio 2. (0.75 puntos) Se ha implementado un código paralelo que presenta una estructura de tareas segmentada (o de flujo de datos):



La tarea 1 requiere un tiempo de 1 unidad de tiempo y la 2 de 8 unidades de tiempo. La tarea 2 se puede ejecutar en paralelo a su vez distribuyendo por igual entre los procesadores (hasta un máximo de 8 procesadores) su carga de trabajo. La tarea 2 no se puede paralelizar en más de 8 procesadores.

Suponiendo que se aplica el código a una secuencia de `n` entradas, conteste a las siguientes preguntas: (a) ¿Cuál es la máxima ganancia que se puede conseguir y para qué número de procesadores se consigue? (b) ¿Cuál sería la eficiencia? (c) ¿Qué ganancia se obtendría con 5 procesadores? (d) ¿Qué requisito debe cumplir una aplicación para que se deba usar una estructura segmentada?

(a)-(b): 0.5 (c)-(d): 0.25

Ejercicio 3. (1 puntos) La secuencia de instrucciones de la tabla que acompaña a este ejercicio se ha ejecutado en un procesador superescalar de 32 bits con buffer de reordenamiento (ROB) y modelo de consistencia de memoria que relaja todos los órdenes, que dispone de cinco etapas: IF (captar instrucción), ID (decodificar), EX (ejecutar), ROB (escribir resultados en ROB), WB (retirada de instrucciones). Cada una de las etapas IF e ID pueden procesar TRES instrucciones por ciclo, se pueden emitir y retirar DOS instrucciones por ciclo, y almacenar en ROB DOS resultados procedentes de las unidades funcionales por ciclo. El procesador dispone de una unidad funcional de carga de memoria que consume 2 ciclos, una unidad de multiplicación de 3 ciclos, una unidad de suma/resta de 2 ciclos, y una de almacenamiento en memoria de 1 ciclo: **(a)** Complete la tabla de evolución temporal de las instrucciones. **(b)** ¿Qué valor promedio de CPI (ciclos por instrucción) se tiene para este código? **(c)** ¿Para qué se usa ROB en superescalares?

(a) 0.5 (b) (c): 0.5

Instrucción	Significado	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lw r4,0(r1)	r4←m(r1+0)																
mult r3,r4,r2	r3←r4*r2																
add r5,r4,r2	r5←r4+r2																
sw 0(r1),r3	m(r1+0) ←r3																
lw r6,4(r1)	r6←m(r1+4)																
add r3,r6,r5	r3←r6+r5																
sw 4(r1),r3	m(r1+4) ←r3																
addi r1,r1,#4	r1←r1+4																

Ejercicio 4. (1.25 puntos) Se dispone de un multiprocesador UMA con 4 procesadores o nodos (P1-P4) y una memoria de 64 GBytes conectada a los nodos a través de un bus. El multiprocesador implementa para mantener la coherencia de cache un protocolo MESI de espionaje. Cada procesador dispone de una cache de datos de último nivel de 4MBytes con marcos de bloque (también llamados líneas) de 32 bytes. En el multiprocesador se están ejecutando en paralelo tres *threads* que acceden a los elementos de dos vectores `X[]` e `Y[]` de 8 elementos de 64 bits cada uno. Los vectores se encuentran almacenados en posiciones consecutivas a partir de una dirección que es comienzo de bloque (o línea) en la memoria principal: primero están almacenados los componentes de `X[]`y, justo a continuación, los elementos de `Y[]`. **(a)** ¿Cuántos bloques de memoria ocupan los vectores `X[]` e `Y[]`? **(b)** Indique los estados de los bloques en las caches y las acciones que se generan para la secuencia de eventos de la tabla (considere que inicialmente los bloques que contienen ambos vectores no están en ninguna cache.

(a) 0.25 (b) 1

NOTA: Suponga que bloques distintos se almacenan en la cache de cada procesador en marcos de bloque (líneas) diferentes.

Orden	Evento	Bloque al que se accede	Acciones (paquetes generados)	Estado de los bloques en las cachés (después)
1	P1 lee X[0]			
2	P1 escribe X[1]			
3	P2 lee Y[1]			
4	P3 escribe X[2]			
5	P3 lee Y[2]			
6	P1 escribe Y[0]			
7	P3 lee Y[3]			

Respuestas a Cuestiones (1.75)

1.- $2 * 10^{11}$ FLOP (operaciones en coma flotante)

$T = 2$ s.

$$GFLOPS = \frac{FLOP}{T * 10^9} = \frac{2 * 10^{11}}{2 * 10^9} = 100$$

2.- Multiprocesador con memoria físicamente distribuida (acceso no uniforme a memoria ya que depende de la ubicación en donde se encuentre el dato: local o remoto)

3.- Asignación de carga dinámica, que lleva más sobrecarga el coste que hay que pagar para conseguir una distribución de carga equilibrada (load balancing)

4.- Todos comunican datos diferentes a todos.

5.-
(1) while (z==0) { }; (R)
(2) r1 = X; (R)
(3) Y = r1; (W)

RAW en r1 (no cambia el orden)

(a) X=1, (W)
(b) Y=2 (W)
(c) Z=1; (W)

Puede darse porque no se respeta W → W

El orden relativo (1)(2)(3) no cambia, pero el orden relativo (a)(b)(c) puede cambiar

Y=1 (para (a)(b)(c)(1)(2)(3))
Y=0 (para (c)(1)(2)(a)(b)(3))
Y=2 (para (c)(1)(2)(3)(a)(b))

} pueden verse

6.-
r1 = 1
r2 = 0
r3 = 1

7.-
p1, p2 cmp.gt r2,0
(p1) p1, p2 cmp.gt r1,0
(p1) add r2,r2,r1
(p2) mult r2,r2,r1

8.- Permite eliminar dependencias entre instrucciones dentro de las instrucciones en una iteración del bucle

Se introducen en el nuevo bucle instrucciones de iteraciones diferentes del bucle de partida.

9.- El procesador puede detectar que el dato en O(R5) está solapado con el dato en A(R5) ya que los datos tienen 8 bytes (64 bits). Por lo tanto no permitiría el adelantamiento ya que detecta una dependencia RAW (independientemente de que pueda permitir los adelantamientos especulativos, o de que no se garantice W→R

10.- Arquitectura VLIW

Se extendieron como arquitectura para procesadores de propósito general en la década de los 2000 (cuando se vió que intentar aprovechar más paralelismo a nivel de instrucción (ILP) en un hebra tenía sus límites y el consumo de energía crecía de manera inaceptable.

Ejercicios

Ej 1.- $f = 2\,GHz \rightarrow T_{ciclo} = 0.5\ ns$

	Nº instrucciones	CPI
Op ALU	0.5 NI	4
BRANCH	0.2 NI	5
LOAD	0.2 NI	4
STORE	0.1 NI	3
	NI	

(a) $T_{CPU_1} = NI_1 * T_{ciclo_1} * CPI_1 = 2*10^6 * 4.1 * 0.5\ ns = 4.1*10^6\ ns = 4.1\ ms$

$NI_1 = 2*10^6$ inst

$T_{ciclo_1} = 0.5\ ns$

$CPI_1 = \frac{0.5\,NI_1 * 4 + 0.2\,NI_1 * 5 + 0.2\,NI_1 * 4 + 0.1\,NI_1 * 3}{NI_1} = 2 + 1 + 0.8 + 0.3 = 4.1$

(b)

Op ALU	$0.5\,NI_1$	4
BRANCH	$0.2\,NI_1$	4
LOAD	$0.2\,NI_1$	3
STORE	$0.1\,NI_1$	3
	$NI_2 = NI_1$	

$T_{ciclo_2} = 1.2\,T_{ciclo_1} = 1.2 * 0.5\ ns = 0.6\ ns$

$T_{CPU_2} = NI_2 * T_{ciclo_2} * CPI_2 = 2*10^6 * 3.7 * 0.6\ ns = 4.44*10^6\ ns$

$CPI_2 = \frac{0.5\,NI_1 * 4 + 0.2\,NI_1 * 4 + 0.2\,NI_1 * 3 + 0.1\,NI_1 * 3}{NI_1} = 2 + 0.8 + 0.6 + 0.3 = 3.7$

3.7
0.6
2.22 * 2 = 4.44

No MEJORA

(c) $MIPS_1 = \frac{NI_1}{T_{CPU_1} * 10^6} = \frac{2 * 10^6}{4.1 * 10^{-3} * 10^6} = \frac{2000}{4.1}$

$MIPS_2 = \frac{NI_2}{T_{CPU_2} * 10^6} = \frac{2 * 10^6}{4.44 * 10^{-3} * 10^6} = \frac{2000}{4.44}$

$NI_2 = NI_1$

En este caso mayores MIPS SI significan mejores prestaciones.

(d) $T_{CPU_1} = 2 * 10^6 * 4.1 * 0.5 \text{ ns} = 4.1 * 10^6 \text{ ns}$.

$T_{CPU_{BRANCH}} = 0.2 * 2 * 10^6 * 5 * 0.5 \text{ ns} = 10^6 \text{ ns}$.

$T_{BRANCH} = 0 \Rightarrow T_{CPU_2} = T_{CPU_1} - T_{CPU_{BRANCH}} =$

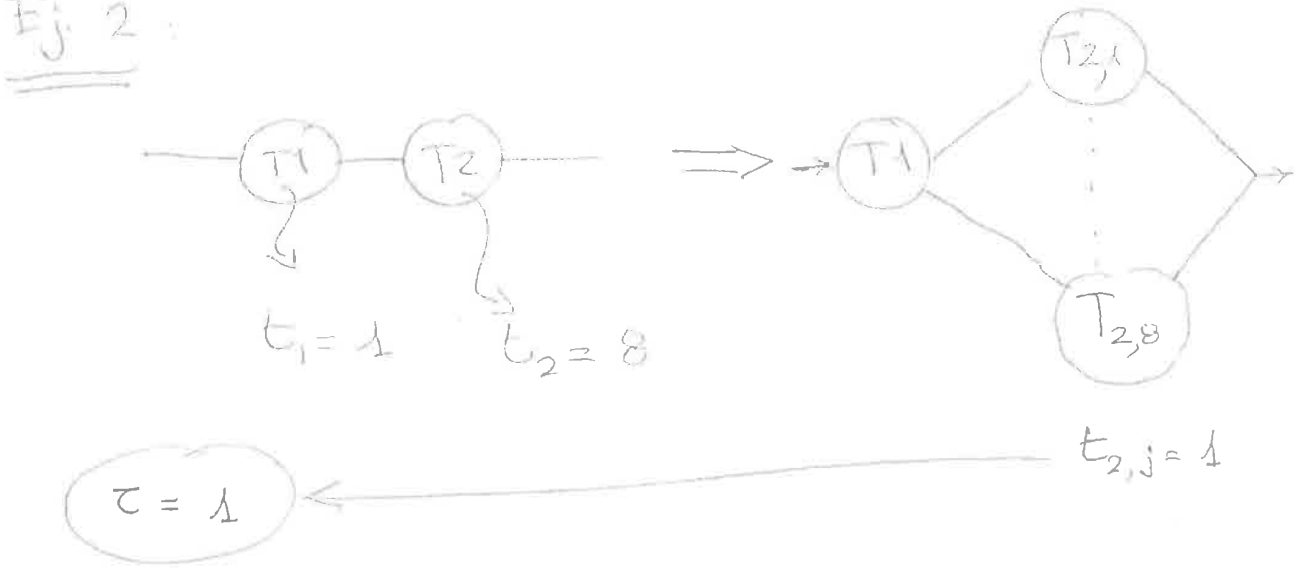
$= 4.1 * 10^6 - 10^6 = 3.1 * 10^6 \text{ ns}$.

$S_{max_{BRANCH}} = \frac{4.1}{3.1}$

(e) $f = \frac{3.1 * 10^6 \text{ ns}}{4.1 * 10^6 \text{ ns}}$ ← Tiempo en el que no se usa el recurso / ← Tiempo total

$S_{max_{BRANCH}} = \frac{1}{f} = \frac{4.1}{3.1}$ (lo que se obtenía en d)

Ej. 2:



a) $T_{sec}(n) = n * (t_1 + t_2) = 9n$

$T_{par}(n) = 2 + (n-1)\tau = 2 + (n-1)$

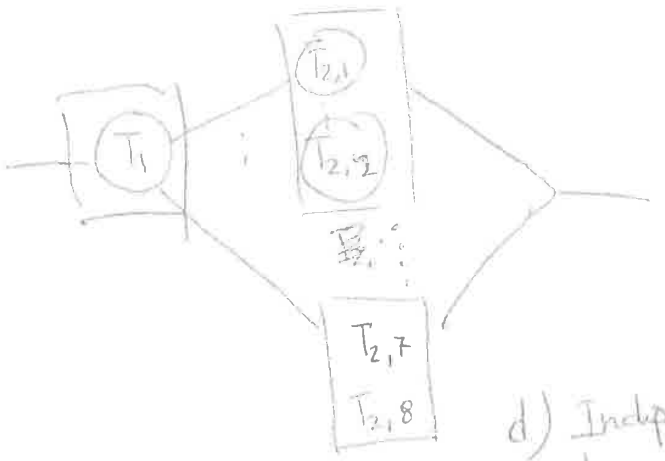
$(t_1 + t_{2/8})$

$S(n) = \frac{9n}{2 + (n-1)}$

Se consigue con 9 procesados

b) $E(n) = \left(\frac{9n}{2+(n-1)}\right) / 9 = \frac{n}{n+1} = \frac{S(n)}{9}$

c)



$\tau = 2$

$T_{per_5}(n) = 3 + (n-1)\tau =$

$= 3 + 2(n-1)$

$S = \frac{9n}{3 + 2(n-1)} \longrightarrow S_{max} = 4.5$

d) Independencia entre las etapas y las tareas que se introducen.

Ej. 3

b) $T = TLI + (N-1)\,CPI$

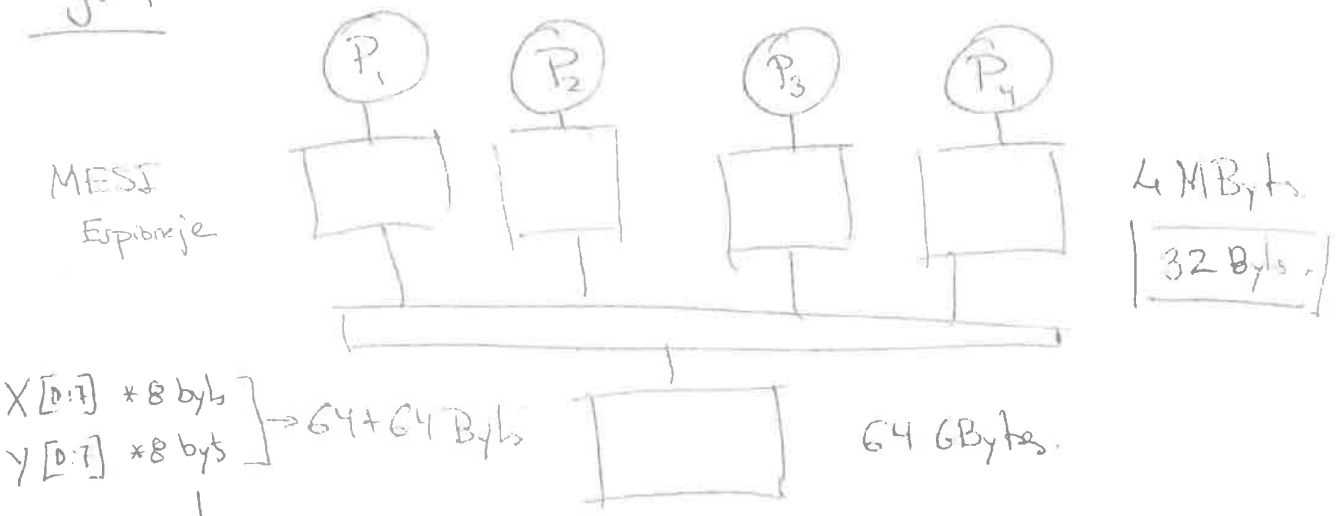
$13 = 6 + 7 * CPI \rightarrow CPI = \frac{7}{7} = 1$

c) - Reordenamiento => Finalizar ordenado a pesar de ejecutar desordenado

- Renombramiento

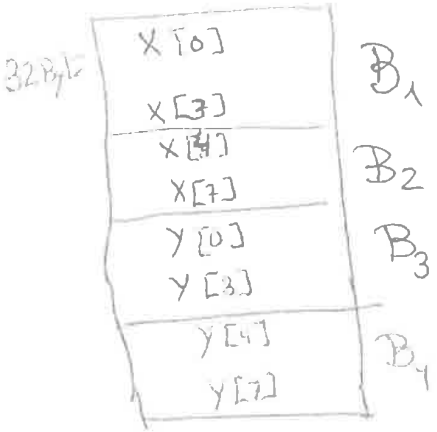
- En el procesamiento especulativo de saltos....

Ej. 4



a) 4 Bloques

b) En la tabla



Ejercicio 3. (1 puntos) La secuencia de instrucciones de la tabla que acompaña a este ejercicio se ha ejecutado en un procesador superescalar de 32 bits con buffer de reordenamiento (ROB) y modelo de consistencia de memoria que relaja todos los órdenes, que dispone de cinco etapas: IF (captar instrucción), ID (decodificar), EX (ejecutar), ROB (escribir resultados en ROB), WB (retirada de instrucciones). Cada una de las etapas IF e ID pueden procesar TRES instrucciones por ciclo, se pueden emitir y retirar DOS instrucciones por ciclo, y almacenar en ROB DOS resultados procedentes de las unidades funcionales por ciclo. El procesador dispone de una unidad funcional de carga de memoria que consume 2 ciclos, una unidad de multiplicación de 3 ciclos, una unidad de suma/resta de 2 ciclos, y una de almacenamiento en memoria de 1 ciclo: **(a)** Complete la tabla de evolución temporal de las instrucciones. **(b)** ¿Qué valor promedio de CPI (ciclos por instrucción) se tiene para este código? **(c)** ¿Para qué se usa ROB en superescalares?

Instrucción	Significado	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lw r4,0(r1)	r4←m(r1+0)	IF	ID	EX	EX	ROB	WB										
mult r3,r4,r2	r3←r4*r2	IF	ID			EX	EX	EX	ROB	WB							
add r5,r4,r2	r5←r4+r2	IF	ID			EX	EX	ROB		WB							
sw 0(r1),r3	m(r1+0) ←r3		IF	ID					EX		WB						
lw r6,4(r1)	r6←m(r1+4)		IF	ID			EX	EX	ROB		WB						
add r3,r6,r5	r3←r6+r5		IF	ID						EX	EX	ROB	WB				
sw 4(r1),r3	m(r1+4) ←r3			IF	ID							EX	WB				
addi r1,r1,#4	r1←r1+4			IF	ID			EX	EX	ROB				WB			

Ejercicio 4. (1.25 puntos) Se dispone de un multiprocesador UMA con 4 procesadores o nodos (P1-P4) y una memoria de 64 GBytes conectada a los nodos a través de un bus. El multiprocesador implementa para mantener la coherencia de cache un protocolo MESI de espionaje. Cada procesador dispone de una cache de datos de último nivel de 4MBytes con marcos de bloque (también llamados líneas) de 32 bytes. En el multiprocesador se están ejecutando en paralelo tres *threads* que acceden a los elementos de dos vectores `X[]` e `Y[]` de 8 elementos de 64 bits cada uno. Los vectores se encuentran almacenados en posiciones consecutivas a partir de una dirección que es comienzo de bloque (o línea) en la memoria principal: primero están almacenados los componentes de `X[]`y, justo a continuación, los elementos de `Y[]`. **(a)** ¿Cuántos bloques de memoria ocupan los vectores `X[]` e `Y[]`? **(b)** Indique los estados de los bloques en las caches y las acciones que se generan para la secuencia de eventos de la tabla (considere que inicialmente los bloques que contienen ambos vectores no están en ninguna cache.

NOTA: Suponga que bloques distintos se almacenan en la cache de cada procesador en marcos de bloque (líneas) diferentes.

Orden	Evento	Bloque al que se accede	Acciones (paquetes generados)	Estado de los bloques en las cachés (después)
1	P1 lee X[0]	B_0	Pr_{Lec} (C=0) / Pt_{Lec} (P_1)	C_1: $B_1 \equiv E$
2	P1 escribe X[1]	B_1	Pr_{Esc} / -	C_1: $B_1 \equiv M$
3	P2 lee Y[1]	B_3	Pr_{Lec} (C=0) / Pt_{Lec} (P_2)	C_1: $B_1 \equiv M$ C_2: $B_3 = E$
4	P3 escribe X[2]	B_1	Pr_{Esc} / $Pt_{Lec Ex}$ (P_3) R_p Bloque (P_3)	C_1: $B_1 \equiv I$ C_3: $B_1 \equiv M$ C_2: $B_3 \equiv E$
5	P3 lee Y[2]	B_3	$Pr_{Lec}^{(C=1)}$ / Pt_{Lec} (P_3)	C_1: $B_1 \equiv I$ C_3: $B_1 \equiv M$ C_2: $B_3 \equiv S$ C_3: $B_3 = S$
6	P1 escribe Y[0]	B_3	Pr_{Esc} / $Pt_{Lec Ex}$ (P_1)	C_1: $B_1 \equiv I$ C_3: $B_1 = M$ C_1: $B_3 \equiv M$ C_2: $B_3 \equiv I$ C_3: $B_3 \equiv I$
7	P3 lee Y[3]	B_3	Pr_{Lec} (C=1) / Pt_{Lec} (P_3) R_p Bloque (P_3)	C_1: $B_1 \equiv I$ C_3: $B_1 \equiv M$ C_1: $B_3 \equiv S$ C_2: $B_3 \equiv I$ C_3: $B_3 \equiv S$