Universidad Carlos III de Madrid

Departamento de Informática
Grado en Ingeniería Informática
Arquitectura de Computadores

Examen

ATENCIÓN:

- Lea atentamente todo el enunciado antes de comenzar a contestar.
- **La duración del examen es de 180 minutos.**

NOMBRE:
APELLIDOS:
NIA:

Ejercicio 1: (1.5 puntos) Indique cuál es el objetivo de cada una de las siguientes optimizaciones de memoria caché realizadas por los compiladores:

- Reordenación de procedimientos.
- Linearización de saltos.
- Alineación de bloques básicos.

NOTA: Se pide exclusivamente el objetivo de la optimización. No debe explicar en qué consiste la optimización.

SOLUCIÓN:

- Reordenación de procedimientos: Reducir los fallos por conflicto debidos a que dos procedimientos coincidentes en el tiempo se corresponden con la misma línea de caché.
- Linearización de saltos: Reducir los fallos de caché debidos a saltos condicionales.
- Alienación de bloques básicos: Reducir la posibilidad de fallos de caché para código secuencial.

Ejercicio 2: (1.5 puntos) Explique en qué consisten las siguientes relaciones de ordenación en el modelo de memoria del lenguaje C++:

- Relación *sincroniza-con*.
- Relación *ocurren-antes*.

SOLUCIÓN:

Relación sincroniza-con:

Es una relación entre operaciones sobre tipos atómicos.

Una escritura sobre un valor atómico sincroniza con una lectura sobre ese valor atómico que lee el valor:

- Almacenado por esa escritura.
- Almacenado por una escritura subsiguiente del mismo hilo que realizó la escritura.
- Almacenado por una secuencia de operaciones read-modify-write sobre el valor de cualquier hilo en la que la primera operación leyó el valor almacenado por la escritura.

Relación ocurre-antes-que:

Especifica qué operación ve los efectos de otra operación:

Dentro de un hilo, una operación ocurre-antes que otra se aparece en una sentencia anterior.

- No hay orden entre dos operaciones que aparecen en la misma sentencia.

Entre dos hilos, una operación en un hilo ocurre-antes que otra operación de otro hilo si:

- Existe una relación sincroniza-con entre ambas.
- Existe una cadena de relaciones ocurre-antes y sincroniza-con entre ellas.

Ejercicio 3: (1.5 punto): Sea un procesador con arquitectura **Intel P6 o posterior**, en el que se tienen dos variables (z y t) que inicialmente tienen el valor 42. En dicho procesador dos hilos ejecutan las siguientes instrucciones:

Hilo 1	Hilo 2

`mov [_z], 1`
`mov r1, [_z]`
`mov r2, [_t]`
	`mov [_t], 1`
	`mov r3, [_t]`
	`mov r4, [_z]`

¿Es posible que al final de la ejecución del hilo 2 los registros r2 y r4 tengan ambos el valor de 42? Justifique su respuesta.

SOLUCIÓN

Si es posible porque las escrituras pueden percibirse en distinto orden por cada procesador. De esta manera en el hilo 1, puede tenerse r1=1 y r2=42, mientras que en el hilo 2, puede tenerse r3=1 y r4=42

Ejercicio 4: (3 puntos) En un determinado procesador se dispone de la siguiente tabla de 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

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.
- 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

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

APARTADO 3:

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

APARTADO 4:

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 4: (2.5 puntos): Sea un multiprocesador con arquitectura de memoria compartida simétrica basado en bus con protocolo de espionaje o *snooping*. Cada procesador tiene una caché privada cuya coherencia se mantiene usando el protocolo MSI. Cada bloque caché tiene una única palabras.

La siguiente tabla muestra el estado inicial de cuatro variables distintas en cada una de las cachés.

	Estado inicial de las variables			
Procesador	**A**	**B**	**C**	**D**
P0	*Shared*	*Exclusive*	*Shared*	*Shared*
P1	*Invalid*	*Invalid*	*Invalid*	*Shared*
P2	*Invalid*	*Invalid*	*Shared*	*Shared*

La siguiente tabla muestra el estado final de estas variables tras realizar una serie de accesos a memoria.

	Estado final de las variables			
Procesador	**A**	**B**	**C**	**D**
P0	*Invalid*	*Invalid*	*Invalid*	*Shared*
P1	*Invalid*	*Invalid*	*Shared*	*Exclusive*
P2	*Exclusive*	*Exclusive*	*Invalid*	*Shared*

Se pide:

- Para cada variable (A, B, C y D) describir de forma justificada el/los accesos realizados y el/los procesos involucrados que han permitido alcanzar el estado final. Nota1: para alcanzar el estado final puede ser suficiente un solo acceso o una secuencia de accesos. Nota2: puede existir un estado final que sea inalcanzable (es decir, que no tiene solución).
- Para cada caso anterior describir el tráfico de bus generado

SOLUCIÓN:

Variable A

P2 Writes A (estado a exclusive) invalidando la copia en la caché de P0

P2 genera un write miss en el bus que lo captura P0 el cual no genera tráfico.

Variable B

P2 Writes B (estado a exclusive) invalidando la copia en la caché de P0

P2 genera un write miss en el bus que lo captura P0 el cual realiza un write-back del bloque a la caché de P2

Variable C

P1 writes B (estado a exclusive) invalidando las otras dos copias.

P1 genera un write miss en el bus que lo captura P0 y P1 los cuales no generan tráfico.

P1 reads o writes D otra variable cuyo bloque ocupa la misma línea caché que C generando un fallo de conflicto. Esto hace que el bloque de C se reemplace y como ha sido modificado se escribe en memoria. El bloque de C no está presente en caché (equivalente a estado invalid).

Esta acción de P1 genera de tráfico realizando el write-back del bloque en memoria.

P1 reads C. Genera un read miss y pasa a estado compartido.

Variable D

El estado final no se puede alcanzar porque un bloque no puede estar exclusivo y compartido a la vez.