Ejercicios de multiprocesadores

J. Daniel García Sánchez (coordinator)
David Expósito Singh
Javier García Blas

Computer Architecture
ARCOS Group
Computer Science and Engineering Department
University Carlos III of Madrid

1. Ejercicios recomendados

Se recomienda la realización del los siguienten ejercicios del libro:
Source: Computer Architecture: A Quantitative Approach. 5 Ed
Hennessy and Patterson. Morgan Kaufmann. 2012.

- Capítulo 5: 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.9, 5.10, 5.11, 5.12.

2. Ejercicios de examen

Exercise 1 *Examen de junio de 2015.*

Sea un computador con dos procesadores, cada uno con su memoria caché privada que usan política de post-escritura a memoria principal. Inicialmente todas las entradas de las cachés se encuentran marcadas como inválidas. La posición de memoria correspondiente a la variable **A**, contiene inicialmente el valor **255**.

En este computador se ejecuta la siguiente secuencia de operaciones:

Tiempo	Procesador 1	Procesador 2
1	**lw $t0, A**	
2		**lw $t1, A**
3	**sw $zero, A**	
4		**lw $t2, A**

Indique para cada una de estas operaciones el estado y el valor de la dirección de memoria **A** tanto en las cachés como en memoria principal.

Tiempo	Estado caché 1	Valor caché 1	Estado caché 2	Valor caché 2	Valor memroia
0	I	–	I	–	255
1					
2					
3					
4					

Exercise 2 *Examen de enero de 2015.*

Dados los siguientes fragmentos de código que ejecutan 3 hilos empleando el protocolo de coherencia MSI.

```
// Código del hilo 0
for(i=0;i<16;i++){
  a[i]= 16;
}

// Código del hilo 1
tmp=0;
for(i=0;i<16;i++){
  a[i]=tmp;
  tmp+=a[i];
}

// Código del hilo 2
cnt=0;
for(i=0;i<4;i++){
  cnt+=b[i];
}
```

El computador tiene una arquitectura CC-NUMA con:

- 2 procesadores de 32 bits con único nivel de memoria caché que es privado a cada procesador y tiene una correspondencia directa. El tamaño de línea caché es de 16 bytes.
- Las memorias caché están inicialmente vacías.
- Los hilos 0 y 2 se ejecutan en el procesador 0 mientras que el hilo 1 se ejecuta en el procesador 1.
- Las variables **i**, **tmp** y **cnt** se almacenan en registros (no se almacenan en memoria).
- El bloque que contiene **a[0]** está asociado a la misma línea caché que el bloque que contiene **b[0]**.

Se pide rellenar las siguientes tablas y justificar la respuesta para los siguientes apartados:

1. Partiendo de la situación inicial, indicar en la siguiente tabla las transiciones de estado del bloque que almacena **a[0]** cuando se ejecuta primero el hilo 0 completamente y a continuación el hilo 1. Indique el tráfico de bus asociado a cada hilo.
   - **Nota**: en el caso de haber varias transiciones asociadas a un hilo, indique todas ellas.

Código	Transición P0	Transición P1	Señales de bus
Hilo 0			
Hilo 1			

2. Partiendo de la situación inicial, indicar en la siguiente tabla las transiciones de estado del bloque que almacena **a[0]** cuando se ejecuta primero el hilo 1 completamente y a continuación el hilo 0 Indique el tráfico de bus asociado a cada hilo.
   - **Nota**: en el caso de haber varias transiciones asociadas a un hilo, indique todas ellas.

Código	Transición P0	Transición P1	Señales de bus
Hilo 0			
Hilo 1			

3. Partiendo de la situación inicial, indicar en la siguiente tabla las transiciones de estado del bloque que almacena **a[0]** cuando se ejecuta primero el hilo 0 completamente, a continuación el hilo 2 completamente y finalmente el hilo 1 Indique el tráfico de bus asociado a cada hilo.

   - **Nota**: en el caso de haber varias transiciones asociadas a un hilo, indique todas ellas.

Código	Transición P0	Transición P1	Señales de bus
Hilo 0			
Hilo 1			
Hilo 2			

4. Para cada uno de los escenarios anteriores, indique el número de fallos caché existente para cada proceso.

Escenario	Fallos caché P0	Fallos caché P1
Hilo 0 → hilo 1		
Hilo 1 → hilo 0		
Hilo 0 → hilo 2 → hilo 1		

Exercise 3 *Examen de enero de 2014.*

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 caché utiliza correspondencia directa y tiene cuatro bloques cada uno con dos palabras. Esta caché utiliza como campo de etiqueta la dirección de memoria completa.

Las siguientes tablas muestran el estado de cada memoria, con la palabra menos significativa a la izquierda.

Procesador P0			
Bloque	**Estado**	**Etiqueta**	**Datos**
B0	I	0x00100700	0x00000000 0x7FAABB11
B1	S	0x00100708	0x00000000 0x00001234
B2	M	0x00100710	0x00000000 0x0077AABB
B3	I	0x00100718	0x00000000 0x7FAABB11

Procesador P1			
Bloque	**Estado**	**Etiqueta**	**Datos**
B0	I	0x00100700	0x00000000 0x7FAABB11
B1	M	0x00100728	0x00000000 0xFF000000
B2	I	0x00100710	0x00000000 0xEEEE7777
B3	S	0x00100718	0x00000000 0x7FAABB11

Procesador P2			
Bloque	**Estado**	**Etiqueta**	**Datos**
B0	S	0x00100720	0x00000000 0x1111AAAA
B1	S	0x00100708	0x00000000 0X00001234
B2	I	0x00100710	0x00000000 0x7FAABB11
B3	I	0x00100718	0x00001234 0x1111AABB

Para cada uno de los apartados que a continuación se presentan, parta de la situación inicial del problema, sin tener en cuenta los cambios de los apartados anteriores. Indique los cambios que se producen en las cachés. En el caso de las lecturas, indique además cuál es el valor efectivamente leído.

1. P2: write 0x00100708, 0xFFFFFFFF
2. P2: read 0x00100708
3. P2: read 0x00100718

En cada apartado deberá rellenar una tabla con el siguiente formato, justificando la respuesta:

Procesador	Bloque	Estado anterior	Estado actual	Etiqueta	Datos	

Exercise 4 *Examen de junio de 2014.*

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 concurrentemente.

El hilo 1 ejecuta:

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

El hilo 2 ejecuta:

```
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.

Exercise 5 *Examen de junio de 2014.*

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 palabra.

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

Exercise 6 *Examen de enero de 2013.*

Un computador dispone de un procesador Intel Core Dúo de 32 bits con dos núcleos (cores) con la configuración mostrada en la figura. El computador utiliza el protocolo de coherencia de caché MSI, el cual se aplica en todos los niveles de memoria caché. El tamaño de bloque es de 16 Bytes y el de cada palabra de 4 Bytes, el cual aplica a todos los niveles de memoria caché.

La memoria caché tiene las siguientes características:

- **L1I**: Tamaño de 16 KB, tiempo de acceso de 1ns y política escritura directa.
- **L1D**: Tamaño de 32 KB, tiempo de acceso de 1ns y política escritura directa.
- **L2**: Tamaño de 2 MB, tiempo de acceso de 5ns y política post-escritura.
- **Memoria principal**: Tamaño de 4 GB, tiempo de acceso de 100 ns.

El código ejecutado por cada núcleo se muestra a continuación. Ambos núcleos empiezan a ejecutar su código al mismo tiempo pero debido a los sleeps ejecutarán cada bucle en distintos instantes de tiempo. Las etiquetas T0, T1, T2, T3, T4 y T5 referencian distintos puntos de ejecución en cada programa. Observe que T0<T1<T2<T3<T4<T5.

Los índices de los bucles **i** y **j** y las variables **tmp1** y **tmp2** se almacenan en registros. El vector **a** se puede almacenar en caché pero **está inicialmente ubicado sólo en la memoria principal** (no está en la caché). Cada elemento de a ocupa una palabra. Asumir que el tiempo de ejecución de cada bucle es tan pequeño que es despreciable.

```
// Código ejecutado en núcleo 1
sleep(seconds(2));
// T1
for (i=0;i<40;++i) {
  tmp1 = tmp1 + a[i];
}
sleep(seconds(3));
// T2
for (j=0;j<40;++j) {
  a[j] = j;
}
// T3


// Código ejecutado en núcleo 2
// T0
for (i=0;i<40;++i) {
  tmp2 = tmp2 + a[i];
}
sleep(seconds(10));
// T4
for (j=0;j<40;++j) {
  tmp2 = tmp2 + a[j];
}
// T5
```

Se pide:

1. Describir en qué estados del protocolo **MSI** está el bloque que almacena **a[0]** para la caché L1D del **núcleo 1** en los instantes T0, T1, T2, T3, T4 y T5. Razona la respuesta.

	T0	T1	T2	T3	T4	T5
Estado de **a[0]** en L1D del núcleo 1						

2. Describir en qué estados del protocolo **MSI** está el bloque que almacena **a[0]** para la caché L1D del **núcleo 2** en los instantes T0, T1, T2, T3, T4 y T5. Razona la respuesta.

	T0	T1	T2	T3	T4	T5
Estado de **a[0]** en L1D del núcleo 1						

3. Calcular el tiempo de ejecución del código ejecutado por el **núcleo 2** considerando únicamente los tiempos de acceso a los datos (asumir que el tiempo de acceso a las instrucciones y el de ejecución de los bucles es despreciable).

Exercise 7 *Examen de enero de 2013.*

Dada la siguiente implementación de *lock/unlock*:

```
lock(location) {
  acquired = 0;
  while (! acquired) { // espera si lock ya está adquirido
    acquired = test_and_set(location);
      if (!acquired) // si test_and_set no adquiere el lock espera de 2 ms
        sleep (2ms);
  }
}

unlock(location) {
  location = 0;
}
```

Cuatro procesadores (P0,P1,P2 y P3) intentan adquirir el lock ejecutando simultáneamente el siguiente código:

```
lock(location);
// Realizar cómputo durante 3 ms
unlock(location);
```

Se pide:

1. Calcular cuánto tiempo (en ms) tarda en adquirir el cerrojo cada uno de los cuatro procesadores asumiendo que lo adquieren de forma ordenada (primero P0, luego P1, luego P2 y finalmente P3). Asumir que el tiempo de ejecución de cada sentencia y que la sobrecarga del protocolo de coherencia son despreciables.

2. ¿Es eficiente la implementación del cerrojo? Justifica tu respuesta.

3. Indicar una mejora posible para esta implementación. Justifica tu respuesta.

Exercise 8 *Examen de junio de 2013.*

Considere 2 procesos ejecutando en sendos procesadores P1 y P2 en un multiprocesador de memoria compartida vía un bus de 100 Mbps. Los procesadores P1 y P2 disponen de una memoria caché no compartida inicialmente vacía y la consistencia se consigue utilizando el protocolo MSI. En el mismo instante de tiempo los procesos ejecutan el siguiente código, donde la variable **l** almacena la dirección de memoria de un cerrojo.

```
lock (l)
//cómputo por t=100 ms.
unlock(l)
```

Se pide:

1. Escribir la implementación para las primitivas de sincronización **lock(l)** y **unlock(l)** basada en la instrucción **test_and_set** cuya sintaxis es:

   ```
   test_and_set Registro Dir
   ```

2. Suponga que un fallo de caché implica una transferencia de 8 bytes para notificar el fallo y una invalidación implica una transferencia de 16 bytes. ¿Cuál sería el tiempo total de ejecución de este programa? Considere sólo los tiempos de fallo, invalidación y cómputo. Asuma que las cachés están inicialmente vacías.

3. ¿Qué diferencia de rendimiento existe entre las primitivas de sincronización **test_and_set** y **LL/SC** (*Load Linked* y *Store Condicional*)? Justifique su respuesta.

3. Ejercicios complementarios de consistencia

Exercise 9

El siguiente programa se ejecuta sobre una máquina que dispone de 3 procesadores. Sus variables se encuentran en memoria compartida e iniciadas a **0**.

El procesador 1 ejecuta:

```
X=1;
```

El procesador 2 ejecuta:

```
Print(X);
Print(Y);
```

El procesador 3 ejecuta:

```
X=1;
Print(X);
```

Asumiendo que todos los accesos a las variables se ejecutan de forma atómica, conteste a las siguientes preguntas:

- Para cada uno de los siguientes resultados indique:
  - ¿Es posible que se obtenga el resultado?
  - Si el resultado es posible ofrezca un orden total de las instrucciones que produzcan el resultado.

Print(X) [P2]	Print(Y)	Print(X) [P3]
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

- Indique un orden global de instrucciones (y su correspondiente resultado) que sea imposible bajo consistencia secuencial, pero posible bajo otros modelos más relajados.

Exercise 10

Repita el ejercicio 9 para el siguiente programa, con todas las variables inicialmente a cero.
Procesador 1:

```
A=1;
Print(flag);
```

Procesador 2:

```
while (flag==0) ;
Print(A);
```

Procesador 3:

```
flag=1;
Print(A);
```

Exercise 11

Repita el ejercicio 9 para el siguiente programa, con todas las variables inicialmente a cero.
Procesador 1:

```
A=1;
```

Procesador 2:

```
U=A;
B=1;
```

Procesador 3:

```
V=B;
W=A;
```

Exercise 12

El siguiente programa se ejecuta en una máquina con 2 procesadores. Sus variables están declaradas en memoria compartida e inicialmente valen 0.
Procesador 1:

```
X=1;
Y=X;
```

Procesador 2:

```
Y=2;
X=3;
```

- ¿Qué resultados para X e Y son posibles bajo consistencia secuencial?
- ¿Existe un resultado, que sea aceptable bajo consistencia relajada, pero no lo sea bajo consistencia secuencial?
- ¿Es el resultado (3,0) posible?