Examen-Resuelto-Tema-3.pdf

Zukii

Arquitectura de Computadores

2º Grado en Ingeniería Informática

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

V/F

• Un procesador multinúcleo no incluye memoria cache
F

• En un microprocesador SMT (multihebra simultánea), en un instante determinado se pueden enviar a ejecutar instrucciones de hebras diferentes
V

• En un microprocesador SMT (multihebra simultánea) se pueden enviar a ejecutar varias instrucciones de una misma hebra en un instante determinado
V

• En un microprocesador SMT (multihebra simultánea), se procesan varias hebras concurrentemente, aunque en un instante determinado solo se pueden enviar a ejecutar instrucciones de la misma hebra
F

• En el protocolo MESI para mantener la coherencia de cache una línea puede estar en el estado S solo en una de las caches del multiprocesador
F

• En el protocolo MESI para mantener la coherencia de cache, una línea puede estar en el estado E en varias caches del multiprocesador
F

• En el protocolo MESI para mantener la coherencia de cache, una línea Puede estar en el estado E solo en una cache del multiprocesador
V

• En el protocolo MESI para mantener la coherencia de cache, una línea puede estar en el estado M solo en una cache del multiprocesador
V

• En el protocolo MESI para mantener la coherencia de caché, una línea dada de memoria puede estar, en un momento dado, en el estado E(exclusivo) en una caché y en el estado S(compartido) en otras cachés
F

• En el protocolo MESI, si en la cache de un nodo N1 hay un bloque B en estado E(Exclusivo), y ese nodo detecta que otro procesador en el nodo N2 intenta leer un dato que está en el bloque B, dicho bloque pasa al estado S(Compartido) en las caches N1 y N2
V

• En el protocolo MESI, si en la cache de un nodo N1 hay un bloque B en estado S(Compartido), y ese nodo detecta que otro procesador en el nodo N2 intenta leer un dato que está en el bloque B, dicho bloque pasa al estado S(Compartido) en las caches N1 y N2
V

• En el protocolo MSI, si en la cache de un nodo N1 hay un bloque B en estado M(Modificado), y ese nodo detecta que otro procesador en el nodo N2 intenta escribir un dato que está en el bloque B, dicho bloque pasa al estado I(no válido) en la cache del N1 y a M(modificado) en N2
V

• En el protocolo MSI, si en la cache de un nodo N1 hay un bloque B en estado M(Modificado), y ese nodo detecta que otro procesador en el nodo N2 intenta leer un dato que está en el bloque B, dicho bloque pasa al estado I(no válido) en la cache del N1 y a M(modificado) en N2
F

• En el protocolo MSI, si en la cache de un nodo N1 hay un bloque en estado M(Modificado), y ese nodo detecta que otro procesador intenta leer un dato que está en ese bloque, el bloque pasa al estado I (Inválido) en el nodo N1
F

CALCULAR

• En un multiprocesador NUMA con 16 nodos, 4GBytes por nodo, y líneas de cache de 64Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión?

2^2 * 2^30/ 2^6= 2^26entradas

• En el multiprocesador NUMA descrito en la pregunta anterior ¿Cuántos bits tiene cada una de las entradas del directorio que se utiliza para mantener la coherencia de cache?

16 + 1 = 17

• En un multiprocesador NUMA con 8 nodos, 16GBytes por nodo, y líneas de cache de 64Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión?

16GBytes = 2^34 Bytes

2^34 / 2^6= 2^28 líneas

• En el multiprocesador NUMA descrito en la pregunta anterior, ¿Cuántos bits tienen cada una de las entradas del directorio que se utiliza para mantener la coherencia de cache en un protocolo MSI con directorio?

8 + 1 = 9bits

• En el mismo multiprocesador NUMA anterior con el mismo protocolo de coherencia de cache, se puede tener el estado 1 1 0 … 0 V (1: hay copia del bloque en la cache del nodo correspondiente al bit; 0: no hay copia en la cache del nodo correspondiente al bit; V:bloque válido en memoria principal) en alguna de las entradas de alguno de los directorios

V

• En un multiprocesador NUMA con 8 nodos, 8GBytes por nodo, y líneas de cache de 128Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizada en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión?

8GBytes = 2^33 Bytes

2^33 / 2^7 = 2^26 líneas

• En el multiprocesador NUMA descrito en la pregunta anterior, ¿Cuántos bits tienen cada una de las entradas del directorio que se utiliza para mantener la coherencia de cache en un protocolo MSI con directorio?

8 + 1 = 9bits

• En el mismo multiprocesador NUMA anterior con el mismo protocolo de coherencia de cache, se puede tener el estado 1 1 0 … 0 I (1: hay copia del bloque en la cache del nodo correspondiente al bit; 0: no hay copia en la cache del nodo correspondiente al bit; I:bloque no válido en memoria principal)
F

• En un multiprocesador NUMA con 64 nodos, 8GBytes por nodo, y líneas de cache de 128Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizada en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión?
2^3* 2^30 / 2^7 = 2^26 entradas

• En el multiprocesador NUMA descrito en la pregunta anterior ¿Cuántos bits tiene cada una de las entradas del directorio que se utiliza para mantener la coherencia de cache?
64+1 = 65 (Un bit por nodo más un bit de validez)

• En un multiprocesador NUMA con 32 nodos, 16GBytes por nodo, y líneas de cache de 128Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizada en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión?
2^4 * 2^30 / 2^7 = 2^27 entradas

• En el multiprocesador NUMA descrito en la pregunta anterior ¿Cuántos bits tiene cada una de las entradas del directorio que se utiliza para mantener la coherencia de cache?
32+1 = 33 (Un bit por nodo más un bit de validez)

¿Qué valores se puede observar en R si el modelo de consistencia de memoria del computador donde están los procesadores que ejecutan estos códigos no respeta el orden W→W (sí respeta los demás), e inicialmente X=Y=0?

P1:	X=2	P2:	R=1;
	Y=1		if (Y==1) R=X;

R=0; R=1; R=2

Si respeta el orden W→W: R=1; R=2

¿Qué valor deben r1, r2, y r3 para que la secuencia de instrucciones siguiente implemente un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y 0 que está abierto.

```
b=r1;
do
        compare&swap(r2,b,k);  // si r2==k k y b se intercambian
while (b==r3);
```

r1=1 r2=0 r3=1

8. Si el modelo de consistencia de memoria de un multiprocesador NO respeta el orden R→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), al final se podría tener Y=0

F

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

R(z) R(x) W(y) — no puede realizarlo por el RAW — RAW en r1 — W(x) W(y) W(z)

9. Cuando se utilizan instrucciones del tipo LL/SC (lectura enlazada/escritura condicional) para implementar un cerrojo, los recursos hardware asociados a dicha técnica permiten detectar si, entre la ejecución de la lectura enlazada (LL) y la ejecución de la escritura condicional (SC) a la dirección de memoria del cerrojo, algún otro procesador ha accedido a dicha dirección

V

10. Si en la secuencia de instrucciones siguiente se tiene que r1=1, r2=0, r3=0, dicha secuencia implementa un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y k=0 que está abierto.

F

```
b=r1;
do
        compare&swap(r2,b,k);  // si r2==k, k y b se intercambian
while (b==r3);
```

8. Si el modelo de consistencia de memoria de un multiprocesador NO respeta el orden R→W (SÍ respeta todos los demás), e inicialmente X=Y=Z=r1=0 (donde X, Y, Z son variables en memoria compartida y r1 es un registro de P1), al final SOLO se podría tener Y=1

V

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

No puede por el RAW — R(z) R(x) W(y) — RAW en r1 — W(x) W(y) W(z)

9. Cuando se utilizan instrucciones del tipo LL/SC (lectura enlazada/escritura condicional) para implementar un cerrojo, los recursos hardware asociados a dicha técnica impiden que, entre la ejecución de la lectura (LL) y la ejecución de la escritura (SC) a la dirección de memoria del cerrojo, ningún otro procesador pueda acceder a dicha dirección de memoria del cerrojo.

F

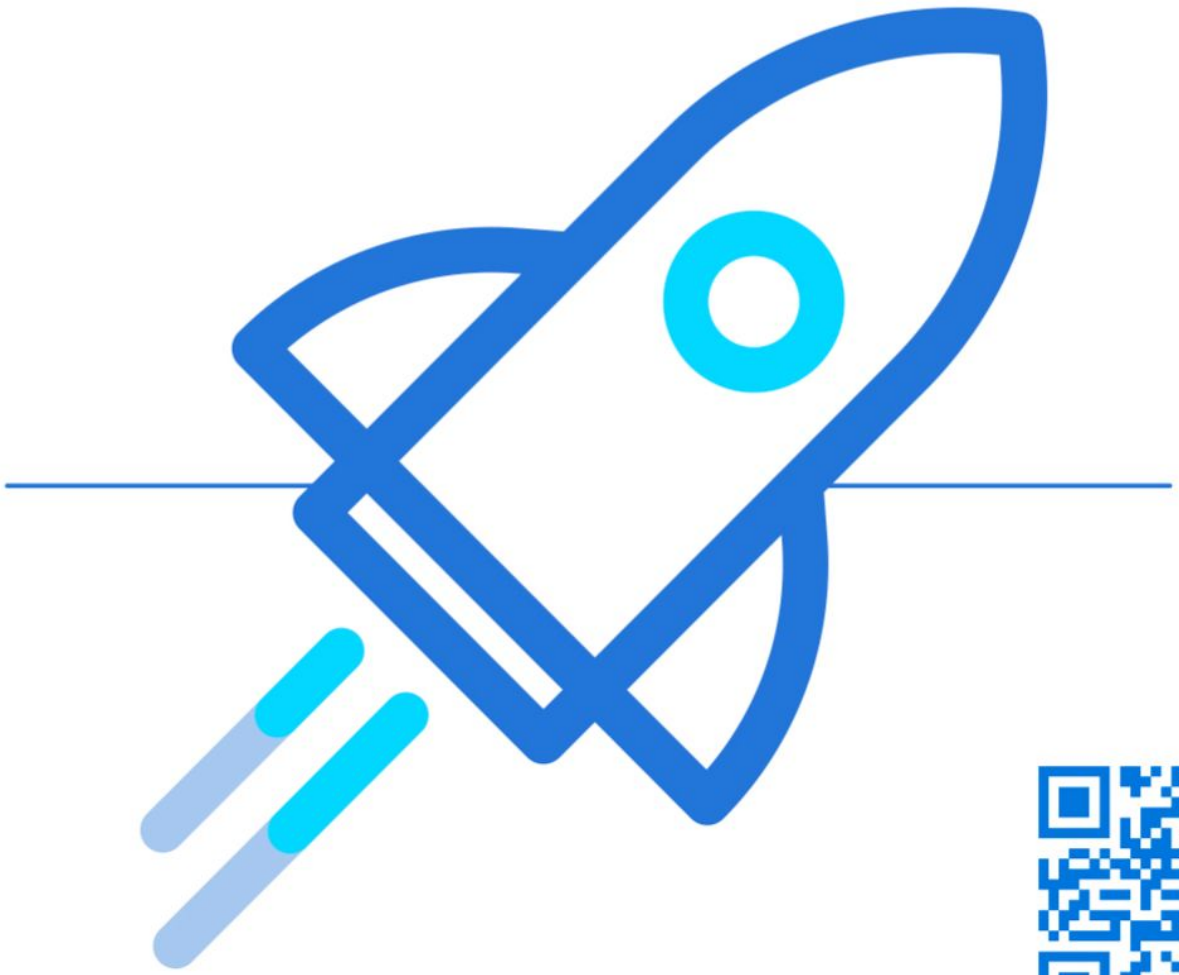
10. Si en la secuencia de instrucciones siguiente se tiene que r1=1, r2=0, r3=1, dicha secuencia implementa un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y k=0 que está abierto.

V

```
b=r1;
do
        compare&swap(r2,b,k);  // si r2==k k y b se intercambian
while (b==r3);
```

deja de imaginar que trabajarás de lo tuyo

hazlo posible.



descarga la app

randstad

partner for talent.

6. ¿Qué valores se puede observar en R si el modelo de consistencia de memoria del computador donde están los procesadores que ejecutan estos códigos tienen un modelo de consistencia secuencial, e inicialmente X=Y=0? (R es un registro del procesador donde se ejecuta P2 y X e Y son direcciones de memoria compartida)

P1:	X=2 Y=1	P2:	R=1; if (Y==1) R=X;

R=1 o R=2

8. ¿Qué valores se observarían en el registro R del problema anterior si no se garantiza el orden W→R?

Lo mismo (no cambia el orden de las escrituras en P1) R=1 o R=2

8. En el modelo de consistencia de liberación no se garantizan los órdenes W→W y W→ R, pero sí los R→ RW.

(F)

9. ¿Qué valor pondría en a para que la secuencia de instrucciones siguiente implemente un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y 0 que está abierto?

```
b=1;
a=0
do
        compare&swap(a,b,k); //  lect-mod-escritura atómica
while (b==1);
```

10. ¿Cómo implementaría un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y 0 que está abierto con una primitiva del tipo test_&_set?

while (test_&_set(k)==1) {}; // k se inicializa a 0

6. ¿Qué valores se puede observar en R si el modelo de consistencia de memoria del computador donde están los procesadores que ejecutan estos códigos tienen un modelo de consistencia secuencial, e inicialmente X=Y=0? (R es un registro del procesador donde se ejecuta P2 y X e Y son direcciones de memoria compartida)

P1:	X=2 Y=1	P2:	R=1; if (Y==1) R=X;

R=1 o R=2

7. ¿Qué valores se observarían en el registro R del problema anterior si no se garantiza el orden W→W?

R=1, R=2, o R=0

8. En el modelo de consistencia de liberación no se garantizan los órdenes W→W y W→ R, pero sí los R→ RW.

(F)

9. ¿Qué valor pondría en a para que la secuencia de instrucciones siguiente implemente un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y 0 que está abierto?

```
b=1;
a=0
do
        compare&swap(a,b,k); //  lect-mod-escritura atómica
while (b==1);
```

10. ¿Cómo implementaría un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y 0 que está abierto con una primitiva del tipo fetch_&_or?

while (fetch_&_or(k,1)==1) {}; // k se inicializa a 0

Cuando se utilizan instrucciones del tipo LL/SC (lectura enlazada/escritura condicional) para implementar un cerrojo, los recursos hardware asociados a dicha técnica impiden que, entre la ejecución de la lectura (LL) y la ejecución de la escritura (SC) a la dirección de memoria del cerrojo, ningún otro procesador pueda acceder a dicha dirección de memoria del cerrojo.
Si, se permite el acceso a la memoria, pero se modifica un bit de marca para tenerlo en cuenta en la instrucción SC posterior
(F)

Cuando se utilizan instrucciones del tipo LL/SC (lectura enlazada/escritura condicional) para implementar un cerrojo, los recursos hardware asociados a dicha técnica permiten detectar si, entre la ejecución de la lectura enlazada (LL) y la ejecución de la escritura condicional (SC) a la dirección de memoria del cerrojo, algún otro procesador ha accedido a dicha dirección
Precisamente en eso consiste la técnica. Gracias al uso de una marca en un registro auxiliar
(V)

El código siguiente permite implementar un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y k=0 que está abierto:
b=0; k=1;
while (fetch_and_add(k,b)==1) {};
b debería ser igual a 1 y utilizar fetch_and_or(k,b)
(F)

En el protocolo MESI para mantener la coherencia de cache, una línea puede estar en el estado E (exclusivo) solo en una cache del multiprocesador
De ahí, precisamente el nombre del estado, aunque ocurre lo mismo con una línea en el estado M
(V)

En el protocolo MESI, si en la cache de un nodo N1 hay un bloque B en estado M (Modificado), y ese nodo detecta que el nodo N2 intenta escribir en un dato del mismo bloque B, dicho bloque pasará al estado M (Modificado) en la caché del nodo N2, y se mantendrá en el estado M en la caché del nodo N1 tras actualizarse en la memoria principal.
El bloque en N1 se invalida
(F)

En el protocolo MESI, si en la cache de un nodo N1 hay un bloque B en estado S (Compartido), y ese nodo detecta que el nodo N2 intenta escribir un dato en el mismo bloque B (que no está en la caché de N2), dicho bloque pasará al estado E (exclusivo) en la caché del nodo N2, y se mantendrá en el estado S en la caché del nodo N1
Pasará al estado I en el nodo N1 y al estado M en el nodo N2
(F)

En el protocolo MESI, si en la cache de un nodo N1 hay un bloque B en estado S (Compartido), y ese nodo detecta que el nodo N2 intenta leer un dato del mismo bloque B, dicho bloque pasará al estado S (Compartido) en la caché del nodo N2, y se mantendrá en el estado S en la caché del nodo N1
Eso es precisamente lo que ocurre según el protocolo MESI
(V)

En el protocolo MSI de espionaje para la coherencia, si en la cache de un nodo N1 hay un bloque B en estado M (Modificado) y detecta que el nodo N2 intenta escribir en un dato del mismo bloque B, dicho bloque pasará al estado M (Modificado) en la caché del nodo N2, y al estado I en la caché del nodo N1 tras actualizarse en la memoria principal.
Es lo que ocurriría
(V)

En el protocolo MSI de espionaje para la coherencia, si en la cache de un nodo N1 hay un bloque B en estado M (Modificado) y detecta que el nodo N2 intenta escribir en un dato del mismo bloque B, dicho bloque pasará al estado S (Compartido) en las cachés de los nodos N1 y N2 tras actualizarse en la memoria principal.
Pasa a M en N2 y se invalida en N1
(F)

En el protocolo MSI de espionaje para la coherencia, si en la cache de un nodo N1 hay un bloque B en estado M (Modificado) y detecta que el nodo N2 intenta leer un dato del mismo bloque B, dicho bloque pasará al estado S (Compartido) en las cachés de los nodos N1 y N2 tras actualizarse en la memoria principal.
Efectivamente es lo que ocurre según el protocolo MSI
(V)

En un microprocesador SMT (multihebra simultánea) se pueden enviar a ejecutar instrucciones de hebras diferentes en cada ciclo
Eso ocurre precisamente en un SMT
(V)

En un microprocesador SMT (multihebra simultánea), se procesan varias hebras concurrentemente y en un instante determinado solo se pueden enviar a ejecutar instrucciones de una misma hebra.
Se pueden enviar a ejecutar instrucciones de hebras diferentes
(F)

En un multiprocesador NUMA con 16 nodos, 4 GBytes por nodo, y líneas de cache de 128 Bytes, el directorio de memoria utilizado en cada nodo para mantener la cache en un protocolo MSI sin difusión tiene 2^25 (2 elevado a 25) entradas
2^32 Bytes / 2^7 (Bytes/linea) = 2^25 líneas (el número de entradas en el directorio coincide con el número de bloques o líneas la memoria del nodo
(V)

En un multiprocesador NUMA con 16 nodos, 4 GBytes por nodo, y líneas de cache de 128 Bytes, el número de bits que tiene cada una de las entradas del directorio que se utiliza para mantener la coherencia de cache en un protocolo MSI con directorio y codificación de bit completo es igual a 9

Son 17: Un bit por cada nodo (16) más otro de válido/no válido
(F)

En un multiprocesador NUMA con 16 nodos, 4 GBytes por nodo, y líneas de cache de 64 Bytes, el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI de directorios tiene 2^25 (2 elevado a 25) entradas
2^32 Bytes / 2^6 (Bytes/linea) = 2^26 líneas (el número de entradas en el directorio coincide con el número de bloques o líneas la memoria del nodo
(F)

En un multiprocesador NUMA con protocolo MSI basado en directorios de vector de bits completo NO puede haber una entrada en uno de los directorios con todos sus bits de copias en caches a cero (no hay una ninguna copia del bloque correspondiente en las caches de la máquina) y el bit de estado del bloque en memoria igual a 0 (estado No Válido en memoria)
Si el bloque no se ha llevado a ninguna caché debe estar en estado válido en la memoria principal
(V)

En un multiprocesador NUMA con protocolo MSI basado en directorios de vector de bits completo NO puede haber una entrada en uno de los directorios con un único bit a uno (hay una copia del bloque correspondiente en una cache de la máquina) y el bit de estado del bloque en memoria igual a 0 (estado No válido en memoria)
Sí podría haberlo. El bloque estaría en estado modificado en la caché y no sería coherente con la memoria
(F)

En un multiprocesador NUMA con protocolo MSI basado en directorios de vector de bits completo NO puede haber una entrada en uno de los directorios con un único bit a uno (hay una copia del bloque correspondiente en una cache de la máquina) y el bit de estado del bloque en memoria igual a 1 (estado Válido en memoria)
Sí podría haberlo. El bloque podría estar en estado compartido en la correspondiente caché y sería coherente con la copia en memoria
(F)

En un multiprocesador NUMA con protocolo MSI basado en directorios de vector de bits completo NO puede haber una entrada en uno de los directorios con varios bits a uno (hay copias del bloque correspondiente en varias caches de la máquina) y el bit de estado del bloque en memoria igual a 0 (estado No válido en memoria)
Si hay varias copias deben estar en estado compartido y ser coherentes con la memoria principal del nodo correspondiente. Por lo tanto, el bit de estado del bloque en memoria debe estar en estado válido
(V)

En un multiprocesador NUMA con protocolo MSI basado en directorios de vector de bits completo puede haber una entrada en uno de los directorios con un único bit a uno (hay una copia del bloque correspondiente en una cache de la máquina) y el bit de estado del bloque en memoria igual a 1 (estado Válido en memoria)

El bloque estaría en estado compartido en la caché y sería coherente con la memoria

(V)

En un multiprocesador, el procesador P1 ejecuta las instrucciones
(1) while (Z==0) { };
(2) r1=Y;
en paralelo con las instrucciones que ejecuta el procesador P2:
(a) X=1;
(b) Y=2;
(c) Z=1;
Si el modelo de consistencia de memoria de un multiprocesador NO respeta el orden W→R (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), al final se podría tener r1=2

(1) estaría ejecutándose hasta que (c) se haya ejecutado y r1 se cargaría con el último valor almacenado en Y, que es 2. El orden W→W se respeta

(V)

En un multiprocesador, el procesador P1 ejecuta las instrucciones
(1) while (Z==0) { };
(2) r1=Y;
en paralelo con las instrucciones que ejecuta el procesador P2:
(a) X=1;
(b) Y=2;
(c) Z=1;
Si el modelo de consistencia de memoria de un multiprocesador NO respeta el orden 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), al final se podría tener r1=0

(1) estaría ejecutándose hasta que (c) se haya ejecutado, pero (c) puede adelantar a (a) y (b) y r1 se cargaría con el último valor almacenado en Y, que podría haberse mantenido a 0 si no se ha ejecutado (b). Hay que tener en cuenta que el orden W→W no se respeta

(V)

En un multiprocesador, el procesador P1 ejecuta las instrucciones
(1) while (Z==0) { };
(2) r1=Y;
en paralelo con las instrucciones que ejecuta el procesador P2:
(a) X=1;
(b) Y=2;
(c) Z=1;

Si el modelo de consistencia de memoria de un multiprocesador NO respeta el orden 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), al final SOLO se podría tener r1=2

(1) estaría ejecutándose hasta que (c) se haya ejecutado, pero (c) puede adelantar a (a) y (b) y r1 se cargaría con el último valor almacenado en Y, que, por ejemplo, podría haberse mantenido a 0 si no se ha ejecutado (b). Hay que tener en cuenta que el orden W→W no se respeta
(F)

Los comercialmente denominados procesadores HT de Intel (hyper-threading) son procesadores multihebra simultánea (SMT)
Hyper-Threading (HT) es la denominación comercial de Intel para sus microprocesadores multihebra simultánea
(V)

Los microprocesadores SMT (multihebra simultánea) pueden ejecutar varias instrucciones de una misma hebra en paralelo
Precisamente, pueden enviar varias instrucciones a ser ejecutadas en un ciclo. Incluso esas instrucciones pueden pertenecer a hebras diferentes
(V)

Si en la secuencia de instrucciones siguiente se tiene que r1=1, r2=0, r3=0, dicha secuencia implementa un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y k=0 que está abierto.
b=r1;
do
compare&swap(r2,b,k); // si r2==k, k y b se intercambian
while (b==r3);
r3 debe ser igual a 1
(F)

Si en la secuencia de instrucciones siguiente se tiene que r1=1, r2=0, r3=1, dicha secuencia implementa un cerrojo (lock(k)) en el que k=1 significa que el cerrojo está cerrado y k=0 que está abierto.
b=r1;
do
compare&swap(r2,b,k); // si r2==k, k y b se intercambian
while (b==r3);
Es un lock() con espera activa.
(V)

Si una línea de la cache del nodo N1 está en el estado M del protocolo MSI para mantener la coherencia de caché, el contenido de esa línea es coherente con su contenido en memoria principal.
No es coherente porque ha podido ser modificada.
(F)