TEST-Tema-3-soluciones-al-final.pdf

Dwight_Schrute

Arquitectura de Computadores

2º Grado en Ingeniería Informática

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

TEST T3 - Soluciones al Final

1 (no es muy importante)

1. 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. Se trata de las instrucciones Test&Set, Fetch&Oper(x,a) Compare&Swap(a,b,x)

2 (no es muy importante)

2. 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.

3

3. 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) {};
```

4

4. 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.

5

5. 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.

6

6. 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

7

7. 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 N1.

8

8. 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.

9

9. 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.

10

10. 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.

11

11. En un microprocesador SMT (multihebra simultánea) se pueden enviar a ejecutar instrucciones de hebras diferentes en cada ciclo.

12

12. 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.

13

13. 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

14

14. 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

15

15. 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)

16

16. 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)

17

17. 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).

18

18. 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)

19

19. 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)

20

20. 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.

21

21. 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.

22

22. 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.

23

24. Los microprocesadores SMT (multihebra simultánea) pueden ejecutar varias instrucciones de una misma hebra en paralelo.

24

25. 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);
```

25

26. 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);
```

26

27. 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.

27

28. Un procesador multinúcleo no incluye memoria cache.

28

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

29

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

30

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.

31

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.

32

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

33

En el protocolo MESI para mantener la coherencia de cache, una línea puede estar en el estado E sólo en una caché del multiprocesador.

34

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

35

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.

36

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.

37

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.

38

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

ESTE TEMA LO TIENES MASTICADÍÍÍÍÍSIMO

¡~~UNA HORA~~ UN TRIDENT MÁS Y YA LO TIENES!

Trident®

ESTIIIIIRA TUS MOMENTOS

39

40. 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

40

41. 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.

41

41 V/F El código siguiente permite implementar un cerrojo (lock(k)) en el que k=0 significa que el cerrojo está cerrado y k=1 que está abierto:

```
lock(k) {
while (test_and_set(k)==1) {};
}
```

42

42 V/F 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:

```
lock(k) {
while (fetch_and_and(k,0)==0) {};
}
```

43

43 V/F 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);
```

44

44 V/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

45

45 V/F 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.

46

46 • 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?

47

47 • 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?

48

48 • 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?

49

49 • 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?

50

50 • 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

51

51 • 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?

52

52 • 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?

53

• 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)

54

• 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?

55

• 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?

56

• 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?

57

• 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?

SOLUCIONES

1	**FALSO** → Se permite el acceso a la memoria, pero hay un bit que se modifica para tenerlo en cuenta en la SC posterior.
2	**VERDADERO** → Es precisamente en eso en lo que consiste la técnica
3	**FALSO** → **b** debería ser igual a **1** y utilizar **fetch_and_or(b,k)**
4	**VERDADERO** → Tanto en E como en M pasa esto
5	**FALSO** → El bloque NI se invalida
6	**FALSO** → N1:I , N2:M
7	**VERDADERO**
8	**VERDADERO**
9	**FALSO** → N2:M , N1:I
10	**VERDADERO**
11	**VERDADERO**
12	**FALSO** → Se pueden enviar a ejecutar instrucciones de hebras diferentes
13	**VERDADERO ¿?¿?¿? 2^32/2^7**
14	**FALSO** → **17** (un bit por nodo más otro de in/validación.
15	**VERDADERO** → Si el bloque no se ha llevado a ninguna caché debe estar en estado válido en la memoria principal.
16	**FALSO** → Sí podría haberlo, el bloque estaría en estado modificado en la caché y no sería coherente con la memoria
17	**FALSO** → El bloque podría estar compartido en la correspondiente caché y seguir siendo coherente con la copia en memoria

18	**VERDADERO** → 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 deberá estar en estado válido.
19	**VERDADERO** → El bloque estaría en estado compartido en la caché y sería coherente con la memoria
20	**VERDADERO** → Ya que por cojones a-b-c van en orden , mientras se respete el orden W->W
21	**VERDADERO** → C adelantaría a A y B
22	**FALSO** → Ya que también podría ser 0
23	**VERDADERO** → Precisamente pueden entrara varias instrucciones a ser ejecutadas en un ciclo. Incluso esas instrucciones pueden pertenecer a hebras diferentes.
24	**FALSO** → r3 debe ser igual a 1
25	**VERDADERO** → Básicamente implementa un lock() con espera activa
26	**FALSO** → No es coherente porque ha podido ser modificada
27	**FALSO**
28	**VERDADERO**
29	**VERDADERO**
30	**FALSO**
31	**FALSO**
32	**FALSO**
33	**VERDADERO**
34	**VERDADERO**
35	**FALSO**
36	**VERDADERO**
37	**VERDADERO**
38	**VERDADERO**

39	**FALSO**
40	**VERDADERO**
41	**FALSO**
42	**FALSO**
43	**FALSO**
44	**VERDADERO**
45	**FALSO**
CÁLCULOS	
46	2^2 * 2^30 / 2^6 = **2^26**
47	**16+1**
48	16 GB → 2^34 B 2^34 / 2^6 = **2^28**
49	**8+1**
50	**VERDADERO**
51	8 GB = 2^33 2^33 / 2^7 = **2^26**
52	**8+1**
53	**FALSO**
54	2^3*2^30 & 2^7 = **2 ^26**
55	**64+1**
56	2^4*2^30 / 2^7 = **2^27**
57	**32+1**