

Preguntas T3

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.

EXPLICACIÓN: Si, se permite el acceso a la memoria, pero se modifica un bit de marca para tenerlo en cuenta en la instrucción SC posterior.

RESPUESTA: FALSO

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.

EXPLICACIÓN: Precisamente en eso consiste la técnica. Gracias al uso de una marca en un registro auxiliar.

RESPUESTA: VERDADERO

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) {};
```

EXPLICACIÓN: b debería ser igual a 1 y utilizar fetch_and_or(k,b)

RESPUESTA: FALSO

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.

EXPLICACIÓN : De ahí, precisamente el nombre del estado, aunque ocurre lo mismo con una línea en el estado M.

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: El bloque en N1 se invalida

RESPUESTA: FALSO

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

EXPLICACIÓN: Pasará al estado I en el nodo N1 y al estado M en el nodo N2.

RESPUESTA: FALSO

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.

EXPLICACIÓN: Eso es precisamente lo que ocurriría según el protocolo MESI.

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: Es lo que ocurriría

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: Pasa a M en N2 y se invalida en N1

RESPUESTA: FALSO

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.

EXPLICACIÓN: Efectivamente es lo que ocurre según el protocolo MSI

RESPUESTA: VERDADERO

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

EXPLICACIÓN: Eso ocurre

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: Se pueden enviar a ejecutar instrucciones de hebras diferentes

RESPUESTA: FALSO

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

EXPLICACIÓN: 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).

RESPUESTA: VERDADERO

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

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

RESPUESTA: FALSO

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)

EXPLICACIÓN: Si el bloque no se ha llevado a ninguna caché debe estar en estado válido en la memoria principal.

RESPUESTA: VERDADERO

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)

EXPLICACIÓN: Sí podría haberlo. El bloque estaría en estado modificado en la caché y no sería coherente con la memoria

RESPUESTA: FALSO

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).

EXPLICACIÓN: Sí podría haberlo. El bloque podría estar en estado compartido en la correspondiente caché y sería coherente con la copia en memoria.

RESPUESTA: FALSO

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)

EXPLICACIÓN: 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.

RESPUESTA: VERDADERO

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)

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

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: (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.

RESPUESTA: VERDADERO

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 $W \rightarrow W$ (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 $r1=0$.

EXPLICACIÓN: (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 $W \rightarrow W$ no se respeta.

RESPUESTA: VERDADERO

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 $W \rightarrow W$ (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 $r1=2$.

EXPLICACIÓN: (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 $W \rightarrow W$ no se respeta.

RESPUESTA: FALSO

23. Los comercialmente denominados procesadores HT de Intel (hyper-threading) son procesadores multihebra simultánea (SMT).

EXPLICACIÓN: Hyper-Threading (HT) es la denominación comercial de Intel para sus microprocesadores multihebra simultánea.

RESPUESTA: VERDADERO

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

EXPLICACIÓN: Precisamente, pueden enviar varias instrucciones a ser ejecutadas en un ciclo. Incluso esas instrucciones pueden pertenecer a hebras diferentes.

RESPUESTA: VERDADERO

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 ($b==r3$);

EXPLICACIÓN: $r3$ debe ser igual a 1.

RESPUESTA: FALSO

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 ($b==r3$);

EXPLICACIÓN: Es un lock() con espera activa.

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: No es coherente porque ha podido ser modificada.

RESPUESTA: FALSO

28. Un procesador multinúcleo no incluye memoria cache.

RESPUESTA: FALSO

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

RESPUESTA: VERDADERO

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

RESPUESTA: VERDADERO

31. 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.

RESPUESTA: FALSO

32. 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.

RESPUESTA: FALSO

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

RESPUESTA: FALSO

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

RESPUESTA: VERDADERO

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

RESPUESTA: VERDADERO

36. 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.

RESPUESTA: FALSO

37. 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.

RESPUESTA: VERDADERO

38. 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.

RESPUESTA: VERDADERO

39. 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

RESPUESTA: VERDADERO

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

RESPUESTA: FALSO

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.

RESPUESTA: FALSO

42. 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.

EXPLICACION: Si, se permite el acceso a la memoria, pero se modifica un bit de marca para tenerlo en cuenta en la instrucción SC posterior

RESPUESTA: FALSO

43. 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

EXPLICACIÓN: Precisamente en eso consiste la técnica. Gracias al uso de una marca en un registro auxiliar

RESPUESTA: VERDADERO

44. 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) {};
```

EXPLICACIÓN: b debería ser igual a 1 y utilizar fetch_and_or(k,b)

RESPUESTA: FALSO

45. 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

EXPLICACIÓN: De ahí, precisamente el nombre del estado, aunque ocurre lo mismo con una línea en el estado M

RESPUESTA: VERDADERO

46. 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.

EXPLICACIÓN: El bloque en N1 se invalida

RESPUESTA: FALSO

47. 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

EXPLICACIÓN: Pasará al estado I en el nodo N1 y al estado M en el nodo N2

RESPUESTA: FALSO

48. 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

EXPLICACIÓN: Eso es precisamente lo que ocurre según el protocolo MESI

RESPUESTA: VERDADERO

49. 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.

EXPLICACIÓN: Es lo que ocurriría

RESPUESTA: VERDADERO

50. 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.

EXPLICACIÓN: Pasa a M en N2 y se invalida en N1

RESPUESTA: FALSO

51. 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.

EXPLICACIÓN: Efectivamente es lo que ocurre según el protocolo MSI

RESPUESTA: VERDADERO

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

EXPLICACIÓN: Eso ocurre precisamente en un SMT

RESPUESTA: VERDADERO

53. 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.

EXPLICACIÓN: Se pueden enviar a ejecutar instrucciones de hebras diferentes
RESPUESTA: FALSO

54. 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

EXPLICACIÓN: 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)
RESPUESTA: VERDADERO

55. 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

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

56. 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

EXPLICACIÓN: 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)
RESPUESTA: FALSO

57. 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)

EXPLICACIÓN: Si el bloque no se ha llevado a ninguna caché debe estar en estado válido en la memoria principal
RESPUESTA: VERDADERO

58. 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)

EXPLICACIÓN: Sí podría haberlo. El bloque estaría en estado modificado en la caché y no sería coherente con la memoria
RESPUESTA: FALSO

59. 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)

EXPLICACIÓN: Sí podría haberlo. El bloque podría estar en estado compartido en la correspondiente caché y sería coherente con la copia en memoria
RESPUESTA: FALSO

60. 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)

EXPLICACIÓN: 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

RESPUESTA: VERDADERO

61. 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)

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

RESPUESTA: VERDADERO

62. 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 $W \rightarrow R$ (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 $r1=2$

EXPLICACIÓN: (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 $W \rightarrow W$ se respeta

RESPUESTA: VERDADERO

63. 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 $W \rightarrow W$ (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 $r1=0$

EXPLICACIÓN: (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 $W \rightarrow W$ no se respeta

RESPUESTA: VERDADERO

64. 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 $W \rightarrow W$ (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 $r1=2$

EXPLICACIÓN: (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

RESPUESTA: FALSO

65. Los comercialmente denominados procesadores HT de Intel (hyper-threading) son procesadores multihebra simultánea (SMT)

EXPLICACIÓN: Hyper-Threading (HT) es la denominación comercial de Intel para sus microprocesadores multihebra simultánea

RESPUESTA: VERDADERO

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

EXPLICACIÓN: Precisamente, pueden enviar varias instrucciones a ser ejecutadas en un ciclo. Incluso esas instrucciones pueden pertenecer a hebras diferentes

RESPUESTA: VERDADERO

67. 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);
```

EXPLICACIÓN: $r3$ debe ser igual a 1

RESPUESTA: FALSO

68. 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);
```

EXPLICACIÓN: Es un lock() con espera activa.

RESPUESTA: VERDADERO

69. 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.

EXPLICACIÓN: No es coherente porque ha podido ser modificada.

RESPUESTA: FALSO

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

RESPUESTA: VERDADERO

71. En el protocolo MESI para mantener la coherencia de cache, una línea dada de memoria puede estar, en un momento dado, en el estado E(exclusivo) en una cache y en el estado S(compartido) en otras caches.

RESPUESTA: FALSO

72. 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 del N1 y N2.

RESPUESTA: VERDADERO

73. 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.

RESPUESTA: VERDADERO

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

RESPUESTA: $16\text{Gb} = 2^{34} \text{ Bytes}$. $2^{34}/2^6 = 2^{28}$ líneas.

75. En un 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?

RESPUESTA: $8+1 = 9$ bits. (8 para cada nodo + 1 extra)

76. 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.

RESPUESTA: VERDADERO

77. 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 $Y = 0$.

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

RESPUESTA: FALSO

78. 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.

RESPUESTA: VERDADERO

79. 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);
```

RESPUESTA: FALSO

80. 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.

RESPUESTA: FALSO

81. En el protocolo MESI para mantener la coherencia de cache, una línea puede estar en el estado E(Exclusivo) solo en la cache del multiprocesador.

RESPUESTA: VERDADERO

82. 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 del N1 y N2.

RESPUESTA: VERDADERO

83. 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 el procesador del 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.

RESPUESTA: FALSO

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

RESPUETA: $8\text{Gbytes} = 2^{33} \text{ Bytes}$. $2^{33}/2^7 = 2^{26}$ entradas

85. 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 en un protocolo MSI con directorio?

RESPUESTA: $8+1=9$

86. En el mismo multiprocesador NUMA anterior con el protocolo MSI de coherencia de cache, es posible que alguna de las entradas de alguno de los directorios esté en 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).

RESPUESTA: FALSO

87. 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 SOLO podría tener $Y = 1$.

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

RESPUESTA: VERDADERO

88. Cuando se utilizan instrucciones del tipo LL/SC 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.

RESPUESTA: FALSO

89. 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 esta cerrado y $k=0$ que esta abierto.

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

RESPUESTA: VERDADERO

90. Un microprocesador multinúcleo no incluye memoria caché.

RESPUESTA: FALSO

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

RESPUESTA: VERDADERO

92. 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.

RESPUESTA: FALSO

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

RESPUESTA: FALSO

94. 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 (no válido) en el nodo N1.

RESPUESTA: FALSO

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

RESPUESTA: $2^2 \cdot 2^{30} / 2^6 = 2^{26}$ entradas

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

RESPUESTA: $16 + 1 = 17$

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

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

R = 0; R = 1; R = 2;

Sí, respeta el orden W->W: R=1; R=2

98. En un multiprocesador NUMA con 16 nodos, 8 GBytes por nodo, y líneas de cache de 64 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

RESPUESTA: FALSO

99. Si una linea de la caché 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.

RESPUESTA: FALSO

100. 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 enlazada (LL) y la ejecución de la escritura condicional (SC) a la dirección de memoria del cerrojo, ningun otro procesador pueda acceder a dicha dirección de memoria del cerrojo

RESPUESTA: FALSO

101. 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 cache en un protocolo MSI sin difusión tiene 2^{25} (2 elevado a 25) entradas

RESPUESTA: FALSO

102. 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 a cero (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 No válido en memoria)

RESPUESTA: FALSO

103. 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 del mismo bloque B, 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

RESPUESTA: FALSO

104. 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) {};
```

RESPUESTA: FALSO

105. 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:

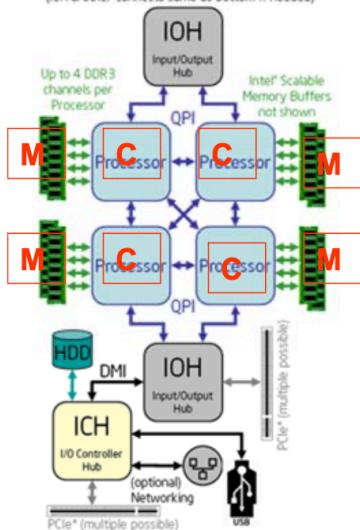
```
b=0; k=1;  
while (fetch_and_or(k,b)==1) {};
```

RESPUESTA: FALSO

106. 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 a cero (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 No válido en memoria)

RESPUESTA: FALSO

(ICH & other connects same as bottom if needed)



107. El servidor de la figura tiene una arquitectura de tipo UMA

RESPUESTA: FALSO

108. 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

RESPUESTA: FALSO

109. 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

RESPUESTA: VERDADERO

110. En un multiprocesador NUMA con 64 nodos, 8GBytes por nodo, y líneas de cache de 128Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión?

RESPUESTA: $2^3 * 2^{30} / 2^7 = 2^{26}$ entradas

111. 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?

RESPUESTA: $64+1 = 65$ (Un bit por nodo más un bit de validez)

112. En un multiprocesador NUMA con 32 nodos, 16GBytes por nodo, y líneas de cache de 128Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión?

RESPUESTA: $2^4 * 2^{30} / 2^7 = 2^{27}$ entradas

113. 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?

RESPUESTA: $32+1 = 33$ (Un bit por nodo más un bit de validez)

114. 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?

RESPUESTA: $2^2 * 2^{30} / 2^6 = 2^{26}$ entradas

115. 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?

RESPUESTA: $16\text{GBytes} = 2^{34}\text{Bytes}$
 $2^{34} / 2^6 = 2^{28}$ líneas

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

RESPUESTA: $8\text{GBytes} = 2^{33}\text{Bytes}$
 $2^{33} / 2^7 = 2^{26}$ líneas

117. 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

RESPUESTA: VERDADERO

118. ¿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 $W \rightarrow W$ (sí respeta los demás), e inicialmente $X=Y=0$?

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

R=0; R=1; R=2

Si respeta el orden $W \rightarrow W$: R=1; R=2

119. ¿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 ejecuta P2 y X e Y son direcciones de memoria compartida)

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

R=1 o R=2

120. ¿Qué valores se observarían en el registro R del problema anterior si no se garantiza el orden $W \rightarrow W$?

R=1, R=2 o R=0

121. En el modelo de consistencia de liberación no se garantizan los órdenes $W \rightarrow W$ y $W \rightarrow R$, pero si los $R \rightarrow RW$

RESPUESTA: FALSO

soluciones_prueba_Tema3_2018.pdf



Cinder



Arquitectura de Computadores



2º Grado en Ingeniería Informática



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



**Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.**



Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

ARQUITECTURA DE COMPUTADORES. Benchmark del Tema 3

Estudiante:

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

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

3. 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 del N1 y N2 (N)

4. 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 (N)

5. En un multiprocesador NUMA con 8 nodos, 16 GBytes por nodo, y líneas de cache de 64 Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión? $16\text{ GBytes} = 2^{34}\text{ Bytes}$. $\frac{2^{34}}{2^6} = 2^{28}$ líneas (F)

6. 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? $8+1 = 9$ bits. (N)

7. 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 (N)

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; W (X)
(2) r1=X;	(b) Y=2; W (Y)
(3) Y=r1; RAW en P1	(c) Z=1; 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 (N)

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);
```



Regístrate en el webinar. Dónde te contaremos cómo conseguirla

Fecha 26 de mayo · Hora 18:00 - 19:00



EDUCOWAY
EUROPEAN EXCELLENCE EDUCATION

WUOLAH

ARQUITECTURA DE COMPUTADORES. Benchmark del Tema 3

Estudiante:

1. 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.

F

2. 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

V

3. 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 procesador del nodo N2 intenta leer un dato que está en el bloque B, dicho bloque pasa al estado S (Compartido) en las caches del N1 y N2

N

4. 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

5. En un multiprocesador NUMA con 8 nodos, 8 GBytes por nodo, y líneas de cache de 128 Bytes.

¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la cache en un protocolo MSI sin difusión?

$$8 \text{ GBytes} = 2^{33} \text{ Bytes.}$$

$$\frac{2^{33}}{2^7} = 2^{26} \text{ entradas}$$

6. 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 en un protocolo MSI con directorio?

$$8+1=9$$

7. En el mismo multiprocesador NUMA anterior, con protocolo MSI de coherencia de cache, es posible que alguna de las entradas de alguno de los directorios esté en 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

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

Examen-Resuelto-Tema-3.pdf



Zukii



Arquitectura de Computadores



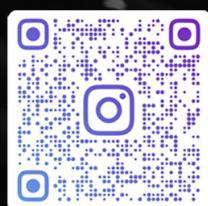
2º Grado en Ingeniería Informática



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



LA PRIMERA RESIDENCIA GAMING
EN EL MUNDO ABRE EN MADRID
ESCANEA Y PARTICIPA EN EL
SORTEO DE UN ALIENWARE



gamingresidences.com

info@gamingresidences.com

**CELEBRACIÓN:
¡FIN DE CURSO!**

2x1

UNIVERSITARIOS Y ESTUDIANTES

=+18=



INFO AQUÍ



**FOSTER'S
HOLLYWOOD**

LA RESIDENCIA DE ESTUDIANTES PERFECTA PARA TI

Amro Estudiantes, la residencia universitaria con todo listo para que vivas una época inolvidable



Tour Virtual

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



GRANADA

MÁLAGA

Estancias
Flexibles

PLAN AMIGO
120 €
Cheque Amazon

amroestudiantes.es

- 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?

$$2^2 * 2^{30} / 2^6 = 2^{26} \text{ entradas}$$

- 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?

$$16 + 1 = 17$$

- 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?

$$16\text{GBytes} = 2^{34} \text{ Bytes}$$

$$2^{34} / 2^6 = 2^{28} \text{ líneas}$$

- 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?

$$8 + 1 = 9 \text{ bits}$$

- 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?

$$8\text{GBytes} = 2^{33} \text{ Bytes}$$

$$2^{33} / 2^7 = 2^{26} \text{ líneas}$$

- 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?

$$8 + 1 = 9 \text{ bits}$$

- 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 Y=1	P2: R=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

LA RESIDENCIA DE ESTUDIANTES PERFECTA PARA TI

Amro Estudiantes, la residencia universitaria con todo listo para que vivas una época inolvidable



Tour Virtual



GRANADA

MÁLAGA

Estancias
Flexibles

PLAN AMIGO
120 €
Cheque Amazon

amroestudiantes.es

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 se podría tener $Y=0$

F

$\begin{array}{l} R(Z) \\ R(X) \\ W(Y) \end{array}$	P1: (1) while ($Z==0$) { }; (2) $r1=X$; (3) $Y=r1$; <i>o puede fallar en el RAW</i>	P2: (a) $X=1$; $W(X)$ (b) $Y=2$; $W(Y)$ (c) $Z=1$; $W(Z)$
---	--	--

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

$\begin{array}{l} R(Z) \\ R(X) \\ W(Y) \end{array}$	P1: (1) while ($Z==0$) { }; (2) $r1=X$; (3) $Y=r1$; <i>No puede fallar el RAW</i>	P2: (a) $X=1$; $W(X)$ (b) $Y=2$; $W(Y)$ (c) $Z=1$; $W(Z)$
---	--	--

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);
```

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 $W \rightarrow R$?

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 $W \rightarrow W$ y $W \rightarrow R$, pero sí los $R \rightarrow RW$.

(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 $W \rightarrow W$?

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

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

(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)

LA RESIDENCIA DE ESTUDIANTES PERFECTA PARA TI

Amro Estudiantes, la residencia universitaria con todo listo para que vivas una época inolvidable



Tour Virtual



GRANADA

MÁLAGA

Estancias
Flexibles

PLAN AMIGO
120 €
Cheque Amazon

amroestudiantes.es



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 cache 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 $W \rightarrow R$ (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 $W \rightarrow W$ 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 $W \rightarrow W$ (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 $W \rightarrow W$ 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 $W \rightarrow W$ (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)

ACT3PREGUNTASEXAMEN.pdf



danielsp10



Arquitectura de Computadores



2º Grado en Ingeniería Informática



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



**FORMACIÓN ONLINE Y
PRESENCIAL EN GRANADA**

**Clases de Inglés B1, B2, C1
DELF B1 y DELF B2 de Francés**

academia-granada.es



**ASIGNATURAS
DE UNIVERSIDAD:
HACEMOS GRUPOS
PARA CLASES DE APOYO**

Aprende Inglés

Con nuestros cursos **GRATUITOS**
para desempleados



**Comienza
Ahora!**

**Fórmate con nuestros cursos de Inglés para las titulaciones
A1 B1 A2 y B2 100% subvencionados para desempleados**



GOBIERNO
DE ESPAÑA

MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



SERVICIO PÚBLICO
DE EMPLEO ESTATAL

SEPE


Junta de Andalucía
Consejería de Empleo, Formación
y Trabajo Autónomo

LA RESIDENCIA DE ESTUDIANTES PERFECTA PARA TI

Amro Estudiantes, la residencia universitaria con todo listo para que vivas una época inolvidable

Daniel Pérez Ruiz - PREGUNTAS T3 - ARQUITECTURA DE COMPUTADORES

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.

EXPLICACIÓN: Si, se permite el acceso a la memoria, pero se modifica un bit de marca para tenerlo en cuenta en la instrucción SC posterior.

RESPUESTA: FALSO

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.

EXPLICACIÓN: Precisamente en eso consiste la técnica. Gracias al uso de una marca en un registro auxiliar.

RESPUESTA: VERDADERO

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) {};
```

EXPLICACIÓN: b debería ser igual a 1 y utilizar fetch_and_or(k,b)

RESPUESTA: FALSO

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.

EXPLICACIÓN: De ahí, precisamente el nombre del estado, aunque ocurre lo mismo con una línea en el estado M.

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: El bloque en N1 se invalida

RESPUESTA: FALSO

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

EXPLICACIÓN: Pasará al estado I en el nodo N1 y al estado M en el nodo N2.

RESPUESTA: FALSO

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.

EXPLICACIÓN: Eso es precisamente lo que ocurriría según el protocolo MESI.

RESPUESTA: VERDADERO



GRANADA

MÁLAGA

Estancias
Flexibles

PLAN AMIGO
120 €
Cheque Amazon

amroestudiantes.es

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.

EXPLICACIÓN: *Es lo que ocurre*

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: *Pasa a M en N2 y se invalida en N1*

RESPUESTA: FALSO

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.

EXPLICACIÓN: *Efectivamente es lo que ocurre según el protocolo MSI*

RESPUESTA: VERDADERO

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

EXPLICACIÓN: *Eso ocurre*

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: *Se pueden enviar a ejecutar instrucciones de hebras diferentes*

RESPUESTA: FALSO

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

EXPLICACIÓN: *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).*

RESPUESTA: VERDADERO

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

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

RESPUESTA: FALSO

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)

EXPLICACIÓN: *Si el bloque no se ha llevado a ninguna caché debe estar en estado válido en la memoria principal.*

RESPUESTA: VERDADERO

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)

EXPLICACIÓN: *Si podría haberlo. El bloque estaría en estado modificado en la caché y no sería coherente con la memoria*

RESPUESTA: FALSO

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).

EXPLICACIÓN: *Si podría haberlo. El bloque podría estar en estado compartido en la correspondiente caché y sería coherente con la copia en memoria.*

RESPUESTA: FALSO

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)

EXPLICACIÓN: *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.*

RESPUESTA: VERDADERO

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)

EXPLICACIÓN: *El bloque estaría en estado compartido en la caché y sería coherente con la memoria.*

RESPUESTA: VERDADERO

LA RESIDENCIA DE ESTUDIANTES PERFECTA PARA TI

Amro Estudiantes, la residencia universitaria con todo listo para que vivas una época inolvidable

Daniel Pérez Ruiz - PREGUNTAS T3 - ARQUITECTURA DE COMPUTADORES

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 $W \rightarrow R$ (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 $r1=2$.

EXPLICACIÓN: (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 $W \rightarrow W$ se respeta.

RESPUESTA: VERDADERO

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 $W \rightarrow W$ (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 $r1=0$.

EXPLICACIÓN: (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 $W \rightarrow W$ no se respeta.

RESPUESTA: VERDADERO

GRANADA

MÁLAGA

Estancias
Flexibles

PLAN AMIGO
120 €
Cheque Amazon

amroestudiantes.es



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 $W \rightarrow W$ (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 $r1=2$.

EXPLICACIÓN: (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 $W \rightarrow W$ no se respeta.

RESPUESTA: FALSO

23. Los comercialmente denominados procesadores HT de Intel (hyper-threading) son procesadores multihebra simultánea (SMT).

EXPLICACIÓN: *Hyper-Threading (HT) es la denominación comercial de Intel para sus microprocesadores multihebra simultánea.*

RESPUESTA: VERDADERO

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

EXPLICACIÓN: *Precisamente, pueden enviar varias instrucciones a ser ejecutadas en un ciclo. Incluso esas instrucciones pueden pertenecer a hebras diferentes.*

RESPUESTA: VERDADERO

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);
```

EXPLICACIÓN: *r3 debe ser igual a 1.*

RESPUESTA: FALSO

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);
```

EXPLICACIÓN: *Es un lock() con espera activa.*

RESPUESTA: VERDADERO

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.

EXPLICACIÓN: *No es coherente porque ha podido ser modificada.*

RESPUESTA: FALSO

28. Un procesador multinúcleo no incluye memoria cache.

RESPUESTA: FALSO

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

RESPUESTA: VERDADERO

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

RESPUESTA: VERDADERO

31. 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.

RESPUESTA: FALSO

32. 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.

RESPUESTA: FALSO

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

RESPUESTA: FALSO

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

RESPUESTA: VERDADERO

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

RESPUESTA: VERDADERO

LA RESIDENCIA DE ESTUDIANTES PERFECTA PARA TI

Amro Estudiantes, la residencia universitaria con todo listo para que vivas una época inolvidable

Daniel Pérez Ruiz - PREGUNTAS T3 - ARQUITECTURA DE COMPUTADORES

36. 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.

RESPUESTA: FALSO

37. 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.

RESPUESTA: VERDADERO

38. 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.

RESPUESTA: VERDADERO

39. 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

RESPUESTA: VERDADERO

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

RESPUESTA: FALSO

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.

RESPUESTA: FALSO

Estancias
Flexibles

PLAN AMIGO
120 €
Cheque Amazon

amroestudiantes.es



GRANADA

MÁLAGA

Reservados todos los derechos. Queda permitida la transformación de esta obra. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Tema-3-TEST-ACTUALIZADOS.pdf



Cooper_3



Arquitectura de Computadores



2º Grado en Ingeniería Informática



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

Aprende Inglés con nuestros cursos **GRATUITOS** para desempleados

 Junta de Andalucía
Consejería de Empleo, Formación
y Trabajo Autónomo



Fórmate con nuestros cursos de Inglés
para las titulaciones A1 B1 A2 y B2
100% subvencionados para desempleados



**Comienza
Ahora!**





LA RESIDENCIA DE ESTUDIANTES PERFECTA PARA TI

Amro Estudiantes, la residencia universitaria con todo listo para que vivas una época inolvidable



GRANADA

MÁLAGA

Estancias
Flexibles

PLAN AMIGO
120 €
Cheque Amazon

amroestudiantes.es

1-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 $W \rightarrow R$ (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 $r1=2$

V

2-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)

V

3-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)

V

4-En un multiprocesador NUMA con 8 nodos, 8 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 5

F

5-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

6-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_or(k,b)==1) {};

F

7-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.

V

8-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

V

9-En un multiprocesador NUMA con 8 nodos, 8 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^{26} (2 elevado a 26) entradas

V

10-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.

F

11-Los microprocesadores SMT(multihebra simultánea), pueden ejecutar varias instrucciones de una misma hebra en paralelo

V

12-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 0 (estado No válido en memoria)

V

13- 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=0

V

14- 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) {};
```

F

15- 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 del mismo bloque B, 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

F

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 todos sus bits de copias a cero (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 No válido en memoria)

F

17- 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 cache en un protocolo MSI sin difusión tiene 2^{25} (2 elevado a 25) entradas

F

18- 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 enlazada (LL) y la ejecución de la escritura condicional (SC) a la dirección de memoria del cerrojo, ningun otro procesador pueda acceder a dicha dirección de memoria del cerrojo

F

19- Si una linea de la caché 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.

F

20- En un multiprocesador NUMA con 16 nodos, 8 GBytes por nodo, y líneas de cache de 64 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

F

RespuestasAC.pdf



Anónimo



Arquitectura de Computadores



2º Grado en Ingeniería Informática



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



**FORMACIÓN ONLINE Y
PRESENCIAL EN GRANADA**

**Clases de Inglés B1, B2, C1
DELF B1 y DELF B2 de Francés**

academia-granada.es



**ASIGNATURAS
DE UNIVERSIDAD:
HACEMOS GRUPOS
PARA CLASES DE APOYO**

CURSO SUPERIOR EN

INTELIGENCIA EMOCIONAL, COACHING Y SOFTSKILLS.

• Descubre nuestros cursos **GRATUITOS.**

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

2.- En el protocolo MESI para mantener la coherencia de cache, una línea dada de memoria puede estar, en un momento dado, en el estado E(exclusivo) en una cache y en el estado S(compartido) en otras caches. F

3.- 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 del N1 y N2. V

4.- 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

5.- En un multiprocesador NUMA con 8 nodos, 16 GBytes por nodo, y líneas de caché de 64 Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión? $16Gb = 2^{34}$ Bytes. $2^{34}/2^6 = 2^{28}$ líneas.

6.- En un 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? $8+1 = 9$ bits.

7.- 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

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:

```
while(Z==0){};  
r1=X;  
Y=r1;
```

P2:

```
X=1;  
Y=2;  
Z=1;
```

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);
```

11.- 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. F

12.- En el protocolo MESI para mantener la coherencia de cache, una línea puede estar en el estado E(Exclusivo) solo en la cache del multiprocesador. V

13.- 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 del N1 y N2. V

14.- 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 el procesador del 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 a M(Modificado) en N2.

15.- En un multiprocesador NUMA con 8 nodos, 8GBytes por nodo, y líneas de caché de 128 Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la cache en un protocolo MSI sin difusión? $8\text{Gbytes} = 2^{33}$ Bytes. $2^{33}/2^7 = 2^6$ entradas

16.- 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 en un protocolo MSI con directorio? $8+1=9$

17.- En el mismo multiprocesador NUMA anterior con el protocolo MSI de coherencia de cache, es posible que alguna de las entradas de alguno de los directorios esté en 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

18.- 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 SÓLO podría tener $Y = 1$. V

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

19.- 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

20.- 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,j);
while(b==r3);
```

21.- Un microprocesador multinúcleo no incluye memoria caché. F

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

23.- 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

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

25.- 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 (no válido) en el nodo N1. F

26.- En un multiprocesador NUMA con 16 nodos, 4GBytes por nodo, y líneas de cache de 64 Bytes. ¿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} / $2^6 = 2^6$ entradas

27.- En el multiprocesador NUMA descrito en la pregunta anterior ¿Cuántos bits tiene cada una de las entradas del directorio que se utiliza para mantener

coherencia de cache? 16 + 1 = 17

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

P1:

```
X = 2;  
Y = 1;
```

P2:

```
R=1;  
if(Y==1)R=X;
```

R = 0; R = 1; R = 2;

Sí, respeta el orden W->W: R=1; R=2

29.- ¿Qué valor deben r1, r2, y r3 tener 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
```

soluciones_prueba_Tema3_2018.pdf



Cinder



Arquitectura de Computadores



2º Grado en Ingeniería Informática



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



**Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.**



Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

ARQUITECTURA DE COMPUTADORES. Benchmark del Tema 3

Estudiante:

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

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

3. 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 del N1 y N2 (N)

4. 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 (N)

5. En un multiprocesador NUMA con 8 nodos, 16 GBytes por nodo, y líneas de cache de 64 Bytes. ¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la coherencia de cache en un protocolo MSI sin difusión? $16\text{ GBytes} = 2^{34}\text{ Bytes}$. $\frac{2^{34}}{2^6} = 2^{28}$ líneas (F)

6. 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? $8+1 = 9$ bits. (N)

7. 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 (N)

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; W (X)
(2) r1=X;	(b) Y=2; W (Y)
(3) Y=r1; RAW en P1	(c) Z=1; 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 (N)

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);
```

ARQUITECTURA DE COMPUTADORES. Benchmark del Tema 3

Estudiante:

1. 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.

F

2. 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

V

3. 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 procesador del nodo N2 intenta leer un dato que está en el bloque B, dicho bloque pasa al estado S (Compartido) en las caches del N1 y N2

N

4. 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

5. En un multiprocesador NUMA con 8 nodos, 8 GBytes por nodo, y líneas de cache de 128 Bytes.

¿Cuántas entradas tiene el directorio de memoria utilizado en cada nodo para mantener la cache en un protocolo MSI sin difusión?

$$8 \text{ GBytes} = 2^{33} \text{ Bytes.}$$

$$\frac{2^{33}}{2^7} = 2^{26} \text{ entradas}$$

6. 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 en un protocolo MSI con directorio?

$$8+1=9$$

7. En el mismo multiprocesador NUMA anterior, con protocolo MSI de coherencia de cache, es posible que alguna de las entradas de alguno de los directorios esté en 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

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

<p>P1:</p> <p>(1) while (Z==0) {};</p> <p>(2) r1=X;</p> <p>(3) Y=r1;</p>	<p>P2:</p> <p>(a) X=1; W(X)</p> <p>(b) Y=2; W(Y)</p> <p>(c) Z=1; W(Z)</p>
--	---

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);
  
```

Tema-3.pdf



Blancabril



Arquitectura de Computadores



2º Grado en Ingeniería Informática



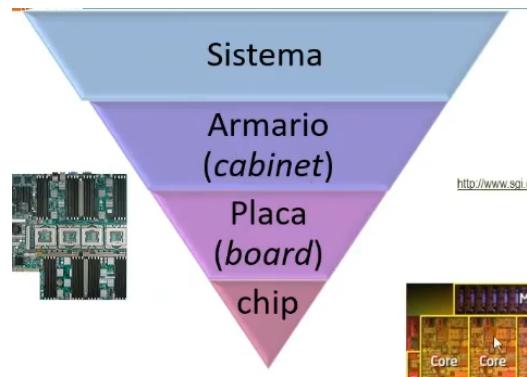
**Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada**



CLASIFICACIÓN DE ARQUITECTURAS CON TLP EXPLÍCITO Y UNA INSTANCIA DE SISTEMA OPERATIVO

Considerando que tenemos un único proceso que está constituido por diversas hebras que comparten la zona de memoria de ese proceso.

Multiprocesador: ejecuta varias hebras en paralelo en una máquina con varios procesadores. Existen diversos niveles de empaquetamiento del multiprocesador: dado, encapsulado, chasis y sistema.



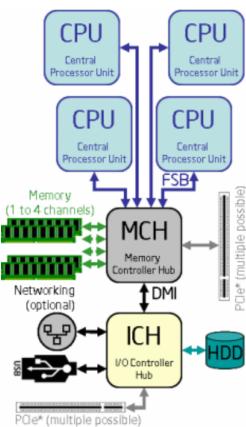
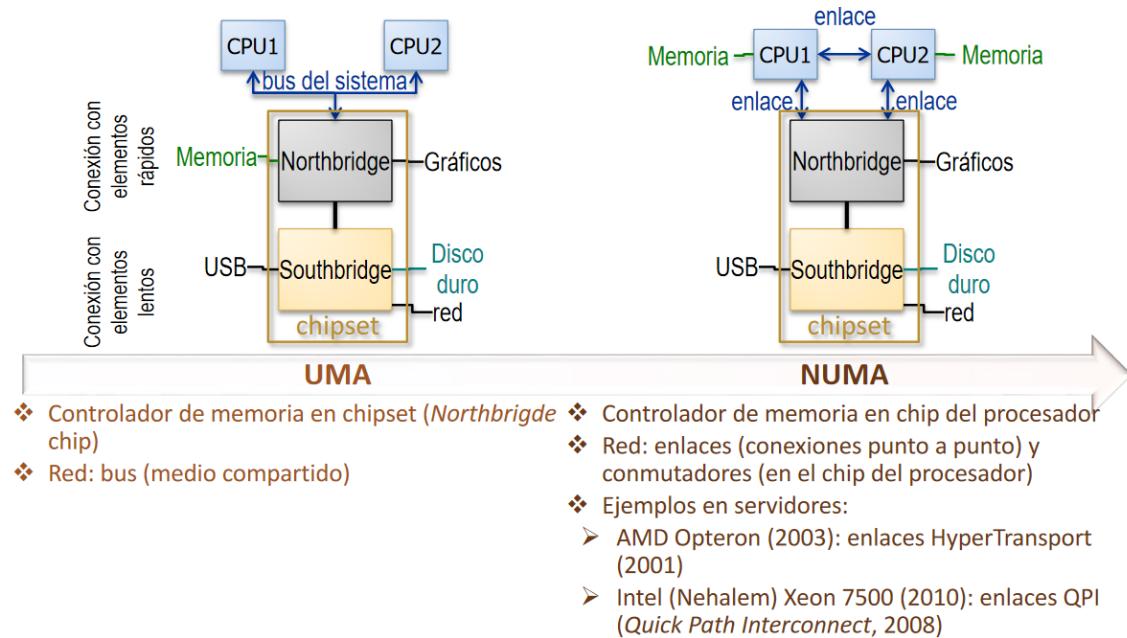
Multicore o CMP: se ejecutan varias hebras en paralelo en un chip de procesamiento multicore, donde cada hebra es un core distinto.

Core multithread: core que modifican su arquitectura ILP para ejecutar threads concurrentemente o en paralelo.

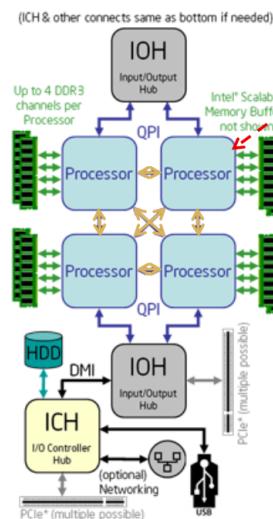
MULTIPROCESADORES SEGÚN LA ESTRUCTURA DE MEMORIA

- **Memoria Centralizada (UMA):** la memoria se encuentra centralizada pero pueden acceder a ella todos los procesadores. El acceso a la memoria es uniforme y el tiempo de acceso es similar. Recordamos que tiene mayor latencia y es poco estable.
- **Memoria Distribuida (NUMA):** todos los procesadores comparten el mapa de memoria, esa memoria está físicamente distribuida por módulos a cada procesador. Tiene menor latencia y es escalable.

Tanto en un caso como en otro, el mapa de memoria es compartido por todos los procesadores que están en la máquina. Lo que ocurre es que dependiendo donde se encuentre esa dirección de memoria, el tiempo de acceso será menor o mayor.



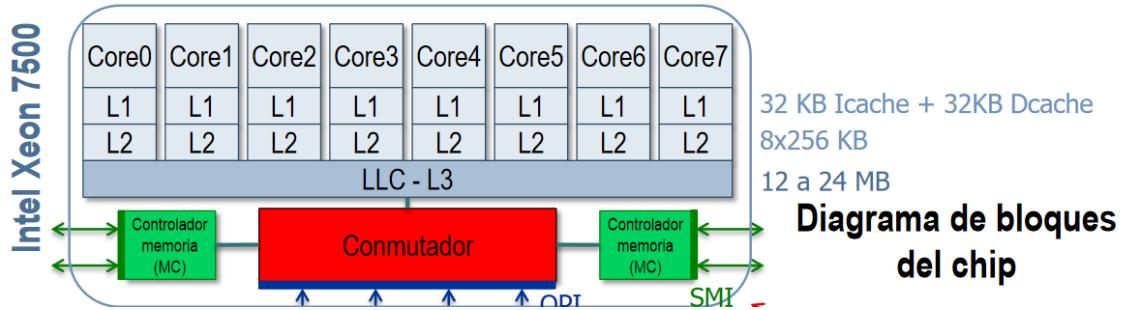
Se trata de un **multiprocesador UMA** donde podemos ver 4 CPUs que todas están conectadas a un puente norte que da acceso a la memoria compartida con el controlador de memoria.



Como podemos ver, se trata de un **multiprocesador NUMA** donde cada procesador tiene su memoria local. Los procesadores se comunican a través de los enlaces para poder acceder a direcciones de memoria que se encuentran en otro procesador.

MULTICORES

Los **multicore**s ejecutan hebras en paralelo en un **chip de procesamiento multicore**, entonces cada hebra va un core distinto.



La tecnología ha permitido integrar multiprocesadores dentro de un circuito integrado. Tenemos varios núcleos con su nivel de caché local y después en este ejemplo tienen un nivel compartido de caché (**LLC -- Last Level Cache**), incluyen también los controladores de memoria y un comutador que permite acceder a la memoria o bien acceder a través de un enlace (QPI) a otro multiprocesador.

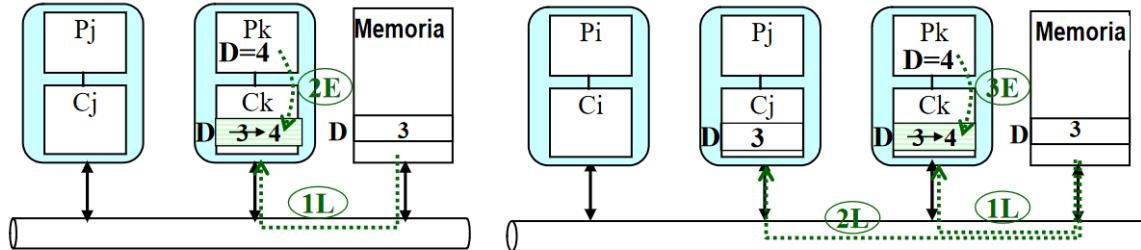
SISTEMA DE MEMORIA EN MULTIPROCESADORES

Cada procesador en un multiprocesador tiene:

- **Memoria caché:** memoria local.
- **Memoria principal.**
- **Controladores de memoria:** permiten determinar las señales que van a la memoria, mantener los ciclos de refresco de la memoria principal y determina si la dirección a la que quiere acceder se encuentra en la memoria caché o en otro sitio de memoria.
- **Buffer:**
 - Buffer de escritura/almacenamiento
 - Buffer que combinan escrituras/almacenamientos
- **Medio de comunicación entre los procesadores y la memoria:** red de interconexión (numa) o buses.

En un multiprocesador, la **comunicación de datos** entre procesadores se realiza a través de lectura y escritura en esa memoria que puede estar distribuida o localizada.

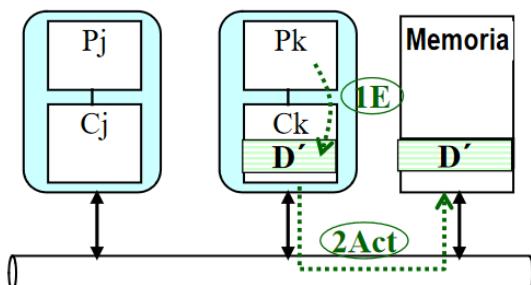
CONCEPTO DE COHERENCIA EN EL SISTEMA DE MEMORIA



Tenemos una memoria compartida por dos procesadores (J y K), los cuales tienen una memoria caché local y una memoria principal compartida. Pero claro, como podemos ver en la primera imagen, en la caché del procesador K aparece un 4 cuando en la memoria es un 3 y habría una incoherencia.

MÉTODOS DE ACTUALIZACIÓN DE MEMORIA PRINCIPAL

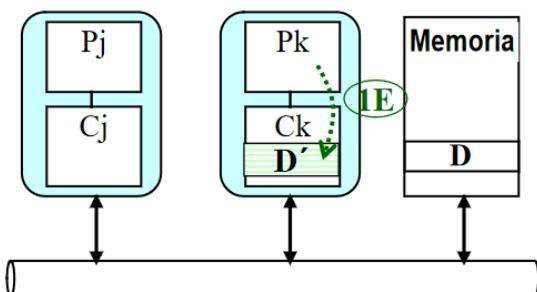
ESCRITURA INMEDIATA



Este método consiste en que cada vez que un procesador escribe en su caché, también la modifica en la memoria principal.

Problema: consume ancho de banda porque cualquier cambio en la caché tiene que pasar por el bus.

POSESCRITURA



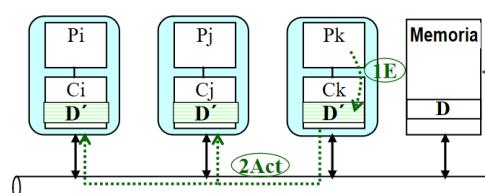
Este método es el más utilizado y consiste en que cuando se modifica la línea en la memoria caché, se permite que haya esa incoherencia hasta el momento en que esa línea se desaloja de la memoria caché.

Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

El problema viene cuando tenemos otros procesadores dentro de la máquina.

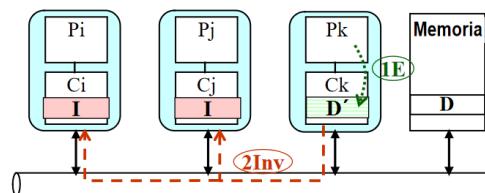
ESCRITURA CON ACTUALIZACIÓN



Cuando se produzca una escritura en su línea de caché, actualiza todas las cachés.

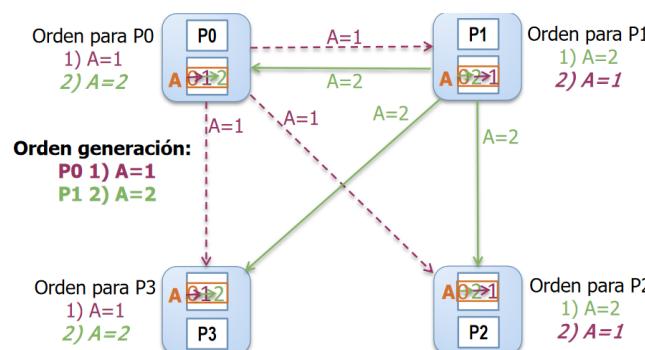
Problema: igual que antes, hay mucho tráfico.

ESCRITURA CON INVALIDACIÓN



En este método cuando se modifica una línea de caché, se produce un ciclo de invalidación que hace las copias de esa línea en otras cachés las pone como no válidas y cuando esos procesadores van a acceder a esa línea, tienen que acceder directamente a la memoria principal.

El problema principal viene cuando se hacen dos escrituras y se hace por escritura con actualización, según donde estén los nodos, pueden pisarse y cada uno puede acabar con un valor en la caché diferente.

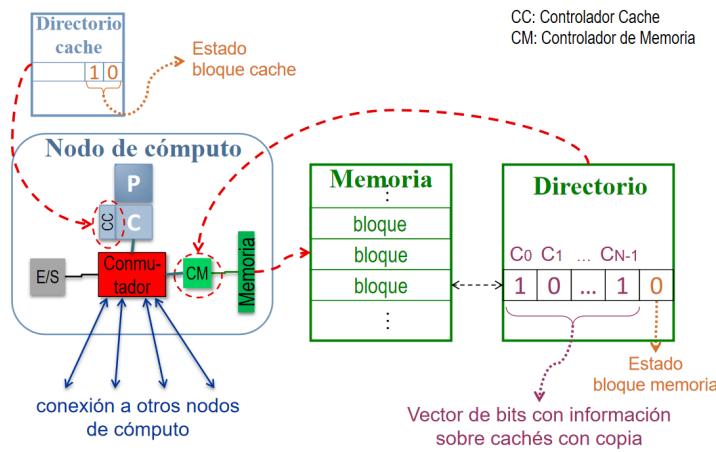


Para arreglar esto hay que:

- **Propagar las escrituras en una dirección** a todos los demás procesadores en un tiempo finito. En el caso de un bus, no hay problema ya que los paquetes de actualización son visibles a todos los nodos conectados.
- **Serializar las escrituras en una dirección** de manera que las escrituras deben verse en el mismo orden por todos los procesadores.

Regístrate en el webinar. Donde te contaremos cómo conseguirla

Fecha 26 de mayo · Hora 18:00 - 19:00



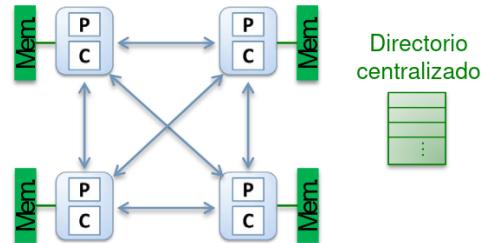
En el nodo de cómputo tenemos el **procesador** y la **caché** con el directorio donde se tiene la información de los bits de estado. Además tendremos un **conmutador** para acceder a la memoria local por el controlador de memoria y acceso a otros nodos de cómputo.

En el controlador nos encontramos con información de la ubicación posible de esos bloques de memoria que son los que se transfieren a caché cuando el nodo pide información y además nos encontramos con un **directorío de memoria** que proporciona información de la memoria, la ubicación en las distintas cachés de la máquina y otro bit que nos indica el estado del bloque de memoria para ver si hay incoherencias o no.

Como cada nodo tiene su memoria local, se introduce el **nodo home** que es aquel nodo donde se encuentra en su memoria local, la línea de caché que estamos considerando.

Hay dos tipos de directorios:

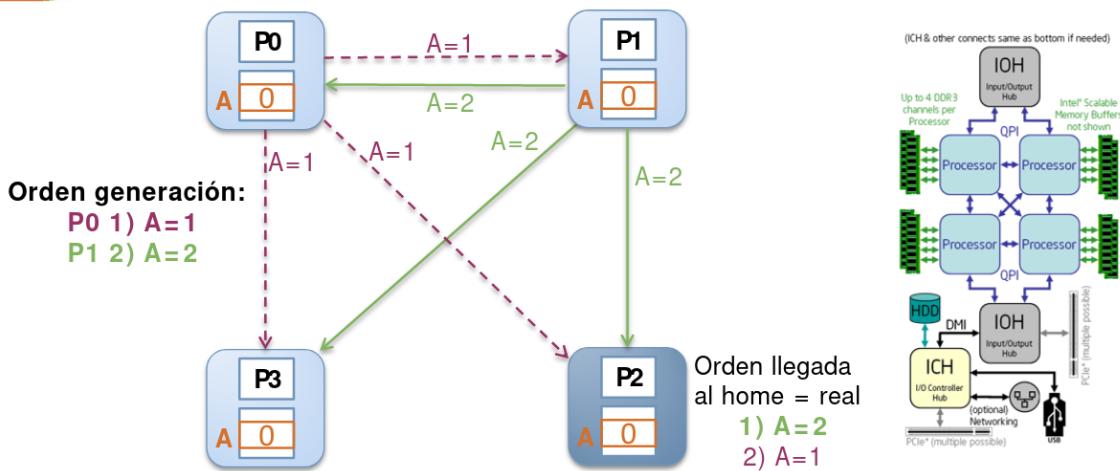
- **Centralizado:** este directorio está compartido por todos los nodos y contiene información de los bloques de todos los módulos de memoria.



- **Distribuido:** en este directorio las filas se distribuyen entre los nodos y el directorio de un nodo contiene información de los bloques de sus módulos de memoria.

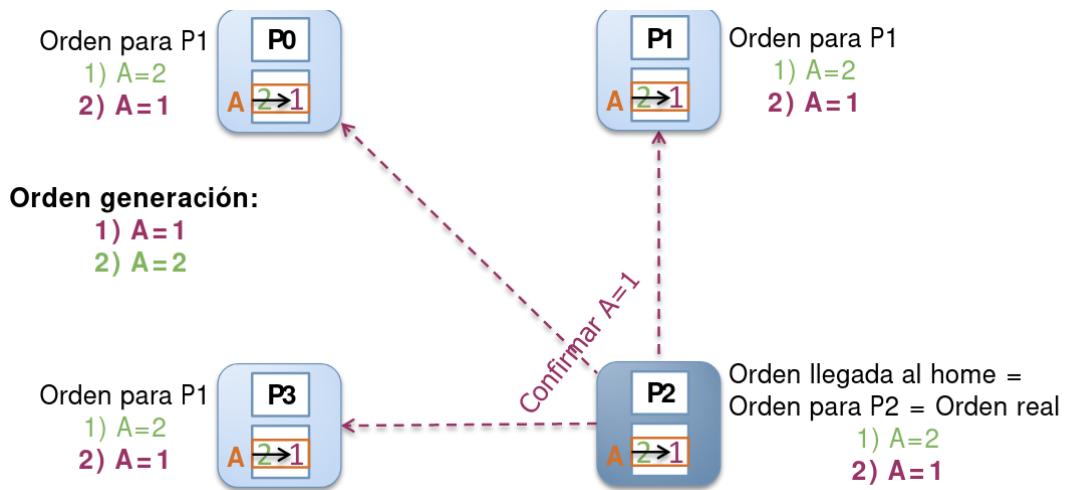


Para resolver el problema anterior con el nodo home lo haremos de la siguiente forma:



- Contenido inicial de las copias de la dirección A en **calabaza**. P0 escribe en A un 1 y, después, P1 escribe en A un 2.
- Se utiliza **actualización** para propagar las escrituras (las propagación se nota con flechas)
- El orden de llegada al home es el orden real para todos

Como es el nodo P2, el que tiene la linea que pertenece a A, pues le llega a el primero el A=2 y después el A=1 y es el encargado de transmitirlo a los otros nodos.



PROTOCOLOS DE COHERENCIA

PROTOCOLOS DE ESPIONAJE (SNOOPY)

Se utilizan cuando tenemos que todos los procesadores conectados a un bus o a una red de difusión, es decir, que lo que pasa en uno de los nodos, se conoce en el resto de los nodos.

PROTOCOLOS BASADOS EN DIRECTORIOS

Se utilizan para redes sin difusión o escalables.

Para definir un **protocolo de coherencia** hay que:

- Determinar qué **política de actualización** de la memoria vamos a usar: escritura inmediata, posescritura o mixta,
- Elegir un **tipo de coherencia entre cachés**: invalidación, actualización o mixta.
- Describir el comportamiento:
 - Posibles estados de los bloques en caché y memoria
 - Tipo de transferencias antes lecturas y escrituras del procesador al nodo y la consecuencia de la llegada de paquetes de otros nodos.
 - Definir transiciones de estados para un bloque en cache y en memoria.

PROTOCOLO MSI

Podemos definir el estado del bloque en caché:

- **Modificado (M)**: única copia del bloque válida en todo el sistema.
- **Compartido (C/S)**: ese bloque se ha traído a la caché y es válido (no ha sido modificado).
- **Inválido**: bloque que no está en la memoria caché o si estuvo en su momento, se ha invalidado.

Para definir el estado del bloque en memoria:

- **Válido**: todas las copias que hay en todas las cachés tienen el mismo valor.
- **Inválido**: hay un bloque en caché modificado.

Transferencias generadas por un nodo con caché:

- **Petición de lectura de un bloque (PtLec)**: por lectura con fallo de caché del procesador del nodo.

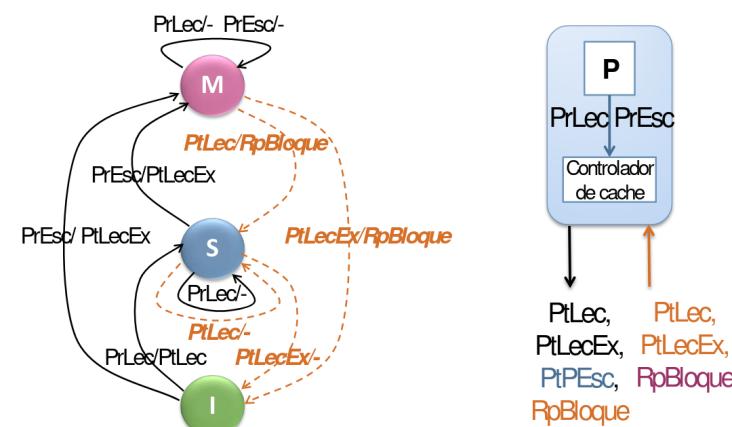
Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

Regístrate en el webinar. Donde te contaremos cómo conseguirla

Fecha 26 de mayo · Hora 18:00 - 19:00

Regístrate en el webinar. Donde te contaremos cómo conseguirla



EST. ACT.	EVENTO	ACCIÓN	SIGUIENTE
Modificado (M)	PrLec/PrEsc		Modificado
	PtLec	Genera paquete respuesta (RpBloque)	Compartido
	PtLecEx	Genera paquete respuesta (RpBloque) Invalida copia local	Inválido
	Reemplazo	Genera paquete posescritura (PtPEsc)	Inválido
Compart. (S)	PrLec		Compartido
	PrEsc	Genera paquete PtLecEx	Modificado
	PtLec		Compartido
	PtLecEx	Invalida copia local	Inválido
Inválido (I)	PrLec	Genera paquete PtLec	Compartido
	PrEsc	Genera paquete PtLecEx	Modificado
	PtLec/PtLecEx		Inválido

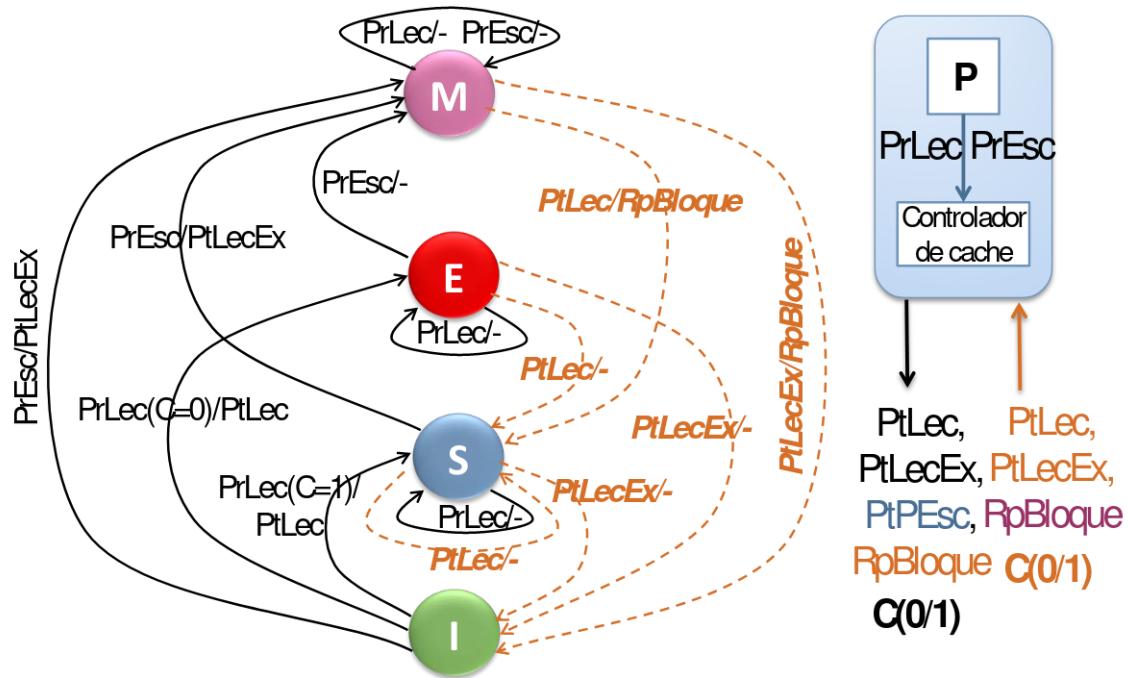
PROTOCOLO MESI

Está basado en el MSI pero definiendo otro estado:

- **Modificado (M):** única copia del bloque válida en todo el sistema.
- **Compartido (C/S):** ese bloque se ha traído a la caché y es válido (no ha sido modificado).
- **Exclusivo (E):** única copia válida en las cachés y esa linea es coherente con la memoria principal.
- **Inválido:** bloque que no está en la memoria caché o si estuvo en su momento, se ha invalidado.

Para definir el estado del bloque en memoria:

- **Válido:** todas las copias que hay en todas las cachés tienen el mismo valor.
- **Inválido:** hay un bloque en caché modificado.



Si es una petición de lectura y es la única copia ($C = 0$) entonces es exclusivo. Si es escritura se pasa directamente al estado modificado.

Modificado (M)	PrLec/PrEsc		Modificado
	PtLec	Genera RpBloque	Compartido
	PtLecEx	Genera RpBloque. Invalida copia local	Inválido
Exclusivo (E)	PtLec		Exclusivo
	PrEsc		Modificado
	PtLec		Compartido
Compartido (S)	PtLecEx	Invalida copia local	Inválido
	PrLec/PtLec		Compartido
	PrEsc	Genera PtLecEx	Modificado
Inválido (I)	PtLecEx	Invalida copia local	Inválido
	PrLec (C=1)	Genera PtLec	Compartido
	PrLec (G=0)	Genera PtLec	Exclusivo
	PrEsc	Genera PtLecEx	Modificado
	PtLec/PtLecEx		Inválido

PROTOCOLO MSI CON DIRECTORIOS

Podemos definir el estado del bloque en caché:

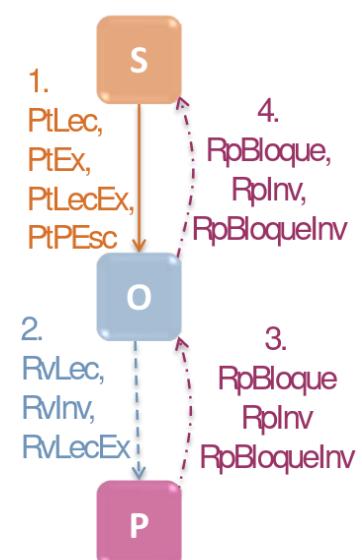
- **Modificado (M)**: única copia del bloque válida en todo el sistema.
- **Compartido (C/S)**: ese bloque se ha traído a la caché y es válido (no ha sido modificado).
- **Inválido**: bloque que no está en la memoria caché o si estuvo en su momento, se ha invalidado.

Para definir el estado del bloque en memoria:

- **Válido**: todas las copias que hay en todas las cachés tienen el mismo valor.
- **Inválido**: hay un bloque en caché modificado.

Transferencias de tipo de paquetes:

- **Nodos:**
 - **Solicitante (S)**
 - **Origen (O)**
 - **Modificado (M)**
 - **Propietario (P)**
 - **Compartidor (C)**
- **Petición de nodo S a O**: lectura de un bloque, lectura con acceso exclusivo, petición de acceso exclusivo, petición de acceso exclusivo sin lectura, posescritura.
- **Reenvío de petición de O a (P, C, M)**: invalidación, lectura.
- **Respuesta de P a O / O a S**: respuesta con bloque, respuesta con o sin bloque confirmado.



CONSISTENCIA DEL SISTEMA DE MEMORIA

La **consistencia** establece el orden en el cual las operaciones de memoria deben parecer haberse realizado a distintas variables que están en diferentes direcciones.

La **coherencia** sólo abarca las operaciones realizadas por múltiples componentes a una única dirección.

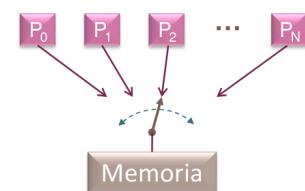
CONSISTENCIA SECUENCIAL

Es el modelo que espera el programador y requiere:

- Todas las operaciones de un único procesador parezcan ejecutarse en el orden descrito en el programa de entrada al procesador, es decir una hebra donde los accesos a memoria se realicen en el orden en el que aparecen en el código.
- Todas las operaciones de memoria parezcan ser ejecutadas una cada vez

En la consistencia secuencial, el sistema de memoria se presenta como una memoria global conectada a todos los procesadores por un conmutador central.

La memoria definirá un orden definido de acceso.



MODELOS DE CONSISTENCIA RELAJADOS

Los modelos de consistencia relajados son aquellos en los cuales no se respetan los requisitos para garantizar la consistencia secuencial, sino que relajan algunos requisitos como:

- **Orden de programa**, es decir que pueden cambiar el orden de las instrucciones.

ST R1, OX1C (R2) → escritura

LD R3, 0X2 D (R3) → lectura

MULT R3, R3, R6

Esto quiere decir que tengo primero una escritura y luego una lectura. En muchos casos, el procesador hace una escritura y tarda un determinado tiempo en hacerlo pero si detecta que acceden a direcciones diferentes, el procesador no espera y hace la lectura porque no hay dependencia entre ellas.

Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

- **Atomicidad**, que quiere decir que hay modelos que permiten a un procesador poder ver el valor escrito por otro.

Los modelos relajados de memoria comprenden:

- Las órdenes de acceso a memoria que no garantizan el sistema de memoria.
- Mecanismos que ofrece el hardware para garantizar un orden cuando sea necesario por si en algún momento necesito que se mantenga ese orden de manera que se fuerce cuando sea necesario.

Ejemplos:

Modelo	Orden del programa relajado W→R W→W R→RW			Orden global Lec. anticipada propia de otro	Instrucciones para garantizar los órdenes relajados por el modelo
Sparc-TSO, x86-TSO	Si			Si	I-m-e (instruc. lectura-modificación-escritura atómica)
Sparc-PSO	Si	Si		Si	I-m-e, STBAR (instrucción <i>STore BARrier</i>)
Sparc-RMO	Si	Si	Si	Si	MEMBAR (instrucción <i>MEMbry BARrier</i>)
PowerPC	Si	Si	Si	Si	SYNC, ISYNC (instrucciones <i>SYNChronization</i>)
Itanium	Si	Si	Si	Si	LD.ACQ, ST.REL, MF (ACQuisition LoaD, RElease STore, Memory Fence), y cmpxchg8.acq y otras I-m-e
ARMv7	Si	Si	Si	Si	DMB (Data Memory Barrier)
ARMv8	Si	Si	Si	Si	LDA LDAR, STL STLR (LoaD-Acquire, STore-reLease 32b 64b), LDAXR LDAXR, STLEX STLXR (LoaD-Acquire eXclusive, Store-reLease eXclusive 32b 64b), DMB

Inicialmente k1=k2=0 P1 k1=1; if (k2=0) { Sección crítica };	P2 k2=1; if (k1=0) { Sección crítica };	NO se comporta como SC los que relajan el orden W→R	①
P1 A=1; P2 if (A=1) B=1;	A=B=0 P3 if (B=1) reg1=A;	NO se comporta como SC los que no garantizan atomicidad	②
Inicialmente A= 0 P1 A=1; k=1;	P2 while (k=0) {}; copia=A;	NO se comporta como SC los que relajan el orden W→Wo R→R	③

Hay que tener cuidado porque igual para conseguir velocidad, perdemos el resultado final.

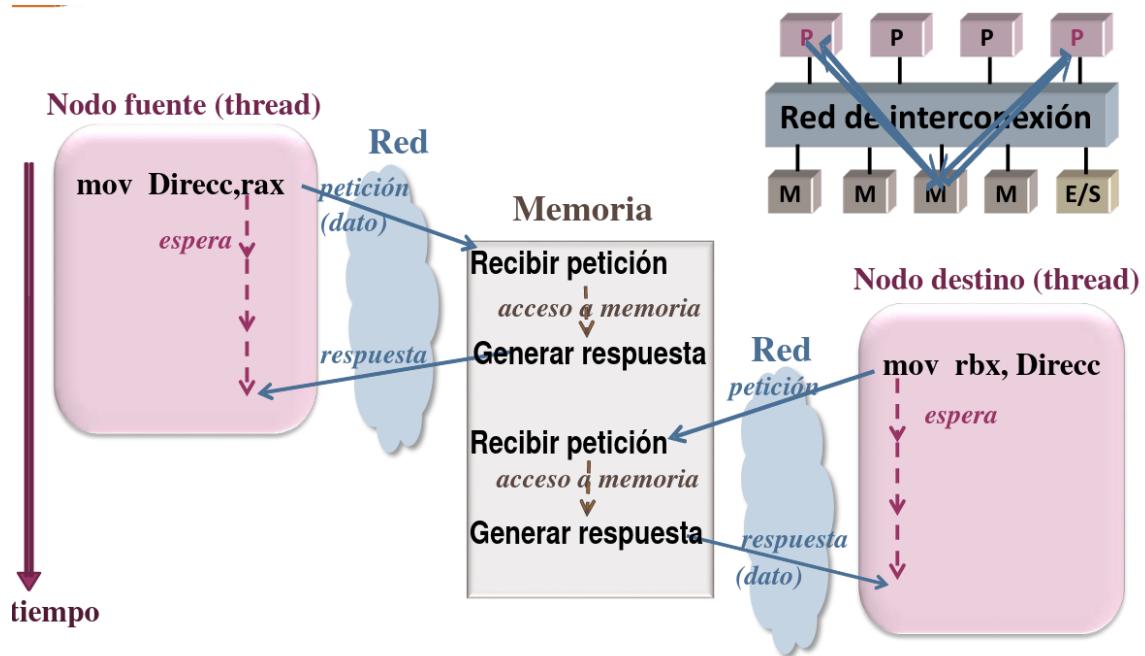
En el ejemplo 1 puede que se ejecute la lectura antes que la escritura, y se ejecuta el if antes de hacer el k1=1.

En el segundo no se garantiza atomicidad y que se ejecute malamente.

Por último, si no se respeta W → W o R → R, fallaría y no sabemos realmente el resultado final.



COMUNICACIÓN EN MULTIPROCESADORES



Se trata de una memoria compartida, a la cual pueden acceder todos los procesadores de la misma manera. El nodo fuente que está ejecutando una hebra y se escribe en el mapa de memoria compartido de la máquina, y otro procesador puede realizar una lectura de esa dirección de memoria y cargarlo en un registro interno. Es necesario sincronizar la escritura con la lectura para que la lectura lea el dato correcto.

COMUNICACIÓN UNO A UNO

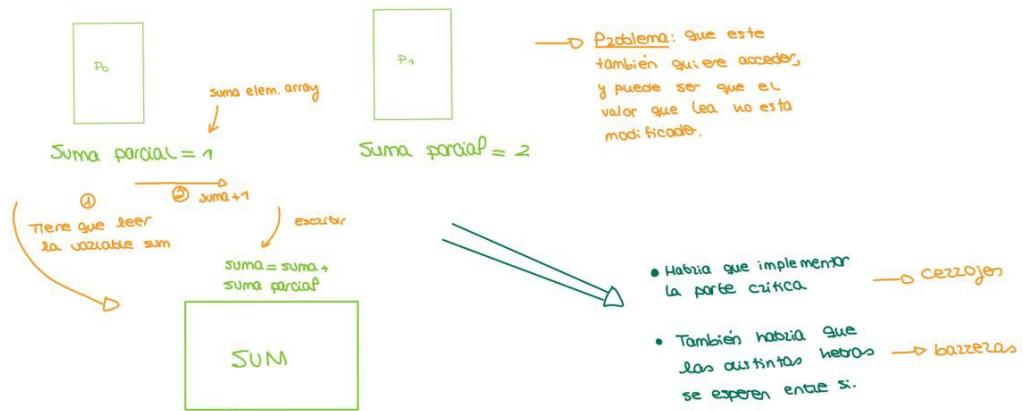
Paralela (K=0)	
P1	P2
...	...
A=1;	while (K=0) { };
K=1;	copia= A ;
...	...

Se debe garantizar que el proceso que recibe lea la variable compartida cuando el proceso 1 ya haya escrito en ella. Para ello utilizamos este bucle while que lo que hace es analizar la variable K mientras se cambia la variable A y cuando A cambia, K posteriormente también y por tanto sale del bucle.

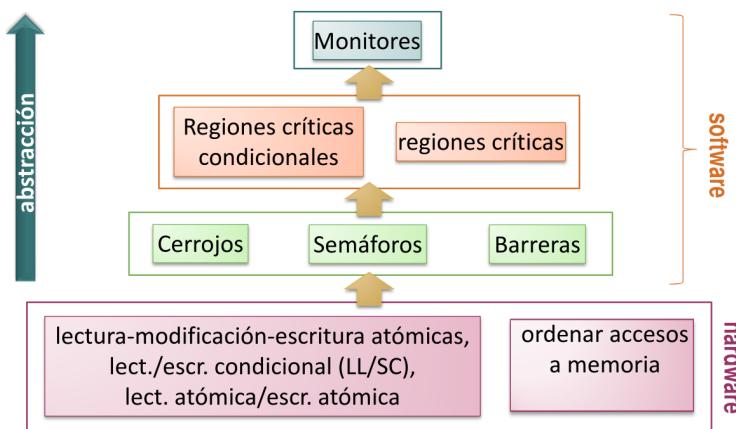
COMUNICACIÓN COLECTIVA

Secuencial	Paralela (sum=0)
<pre>for (i=0 ; i<n ; i++) { sum = sum + a[i]; }</pre>	<pre>for (i=ithread ; i<n ; i=i+nthread) { sump = sump + a[i]; } sum = sum + sump; /* SC, sum compart. */ if (ithread==0) printf(sum);</pre>

En paralelo se realiza un bucle mediante el número de hebras para hacer una distribución de la carga mediante Round Robin



SOPORTE HARDWARE Y SOFTWARE DE SINCRONIZACIÓN



La lectura-modificación-escritura atómica significa que entre que se empieza la lectura y se termina con la escritura, ningún otro procesador va a poder acceder a la posición de memoria a la que está accediendo.

Lecturas enlazadas y escritura condicionales, son instrucciones de escritura que están relacionadas entre sí que se usa para implementar cerrojos y barreras.

CERROJOS

Los cerrojos son variables que se encuentran en memoria compartida (variable K), que permiten la sincronización con dos operaciones.

La primera operación se denomina **cierre del cerrojo** donde se adquiere el derecho para acceder a la sección crítica y al mismo tiempo cierra el cerrojo para que las otras hebras no puedan acceder.

La siguiente operación es **apertura de cerrojo** que libera a una de las hebras que esperan el acceso a una sección crítica y en el caso de que no haya hebras, la siguiente que ejecute lock() entra sin espera.

Secuencial	Paralela
<pre>for (i=0 ; i<n ; i++) { sum = sum + a[i]; }</pre>	<pre>for (i=ithread ; i<n ; i=i+nthread) { sump = sump + a[i]; } lock(k); sum = sum + sump; /* SC, sum compart. */ unlock(k);</pre>

Tus codos aquí.

Ánimo con los exámenes.
Para lo que está por venir, cuenta con Infojobs.

InfoJobs

El portal líder de empleo.

Las alternativas para implementar la espera son la **espera ocupada**, donde las hebras siguen intentando cerrar el cerrojo y la **suspensión del proceso** en una cola.

MÉTODOS

- **Método de adquisición:** método por el cual la hebra trata de adquirir el derecho a utilizar las direcciones compartidas y se implementan normalmente con variables atómicas o bien utilizando LL/SC (carga enlazada, almacenamiento condicional).
- **Método de espera:** método por el cual cada hebra espera a adquirir el derecho a utilizar unas direcciones compartidas y se implementa con la espera ocupada o el bloqueo.
- **Método de liberación:** método utilizado por un thread para liberar a una o varias hebras.

Esto se implementa con la variables compartida K, cuando está a 0, significa que está abierto y cuándo se pone a 1, es que se encuentra cerrado.

Para cerrar el cerrojo se usa la función lock() y para la apertura se usa unlock(). Esto se debe realizar de forma atómica para que dos hebras no pasen a la sección crítica.

```
lock (k)
lock(k) {
    while (leer-asignar_1-escribir (k) == 1) {};
} /* k compartida */
```

```
unlock (k)
unlock(k) {
    k = 0 ;
} /* k compartida */
```

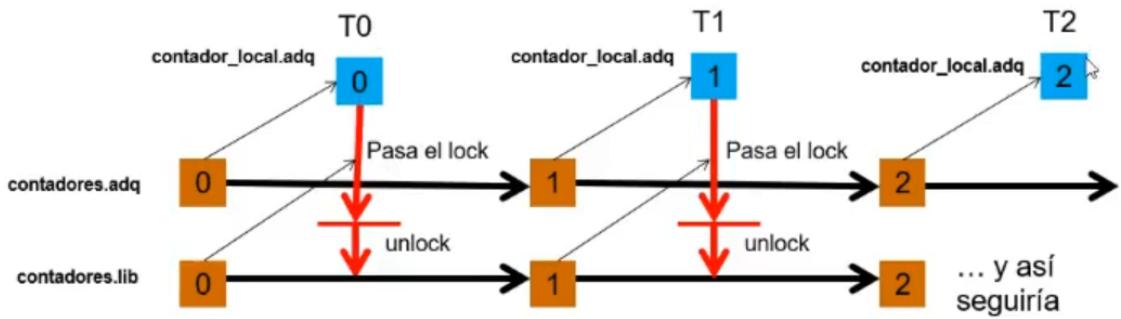
CERROJOS CON ETIQUETA

Este tipo de cerrojo busca garantizar que las hebras que vayan llegando y se esperen a la apertura del cerrojo, en el mismo orden se vayan liberando (FIFO), de manera que todas las hebras esperen un tiempo equivalente.

```
lock (contadores)
contador_local_adq = contadores.adq;
contadores.adq = (contadores.adq + 1) mod max_flujos_control;
while (contador_local_adq <> contadores.lib) {};
```

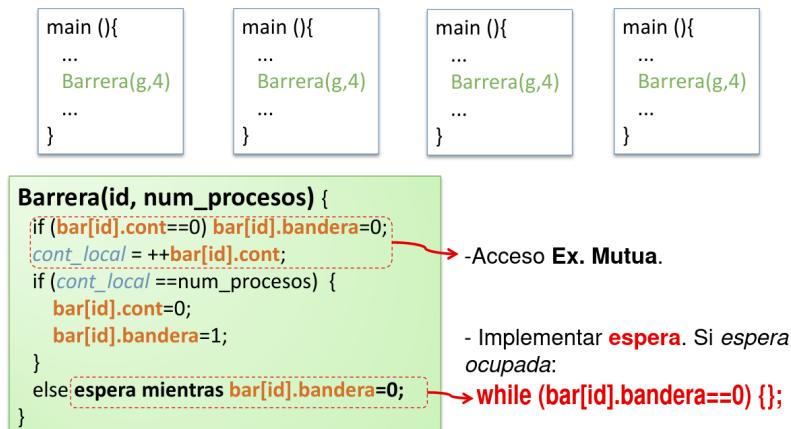
Azul: variable local
Naranja: variable compartida.

```
unlock (contadores)
contadores.lib = (contadores.lib + 1) mod max_flujos_control;
```



BARRERAS

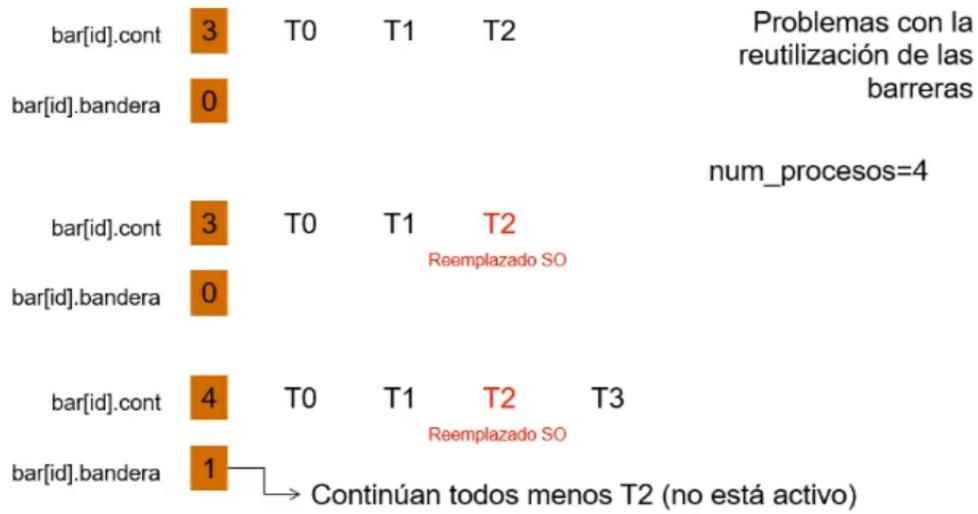
Las barreras permiten sincronizar varias hebras que se esperan hasta que han llegado todas y por tanto, podrían continuar.



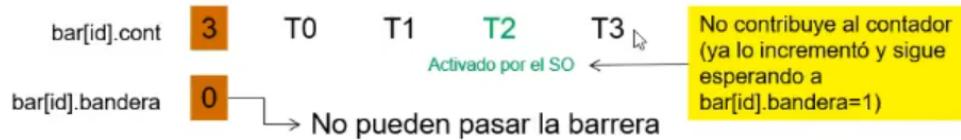
Cada barrera tiene un identificador y el número de procesos que tienen que sincronizarse. Las variables en azul son variables locales y las naranjas son variables de memoria compartida. Cuando llega una hebra lo primero que se comprueba es si el contador se encuentra a 0 y eso significa que no ha llegado ningún otro valor, entonces la bandera se pone a 0 para no permitir el paso y se incrementa el contador local.

Si el contador local es igual al número de procesos, la bandera se pone a 1 entonces pueden pasar todas las hebras y pone el contador de nuevo a 0.

BARRERA SIN REUTILIZACIÓN



Las hebras llegan otra vez a la misma barrera (ej. está en un bucle)



Para solucionar esto, utilizamos la barrera sin reutilización:

Barrera *sense-reversing*

```
Barrera(id, num_procesos) {
    bandera_local = !(bandera_local) //se complementa bandera local
    lock(bar[id].cerrojo);
    cont_local = ++bar[id].cont //cont_local es privada
    unlock(bar[id].cerrojo);
    if (cont_local == num_procesos) {
        bar[id].cont = 0; //se hace 0 el cont. de la barrera
        bar[id].bandera = bandera_local; //para liberar thread en espera
    }
    else while (bar[id].bandera != bandera_local) {}; //espera ocupada
}
```

Se hace con una variable de bandera local para saber si se puede continuar o no.

APOYO HARDWARE A PRIMITIVAS SOFTWARE

INSTRUCCIONES DE LECTURA-MODIFICACIÓN-ESCRITURA ATÓMICAS

Test&Set(x): lectura de una variable compartida x y se escribe un 1 en esa variable x y proporciona lo que hay escrito en la variable antes de ser cambiada.

Fetch&Oper(x,a): lectura de dos variables compartidas x y a, se guarda en una variable local temporal la primera variable y se realiza un sumatorio con la segunda variable, devolviendo esa sumatoria.

Compare&Swap(a,b,x): tres operandos, si x es igual a uno de los operandos, a, se almacena en la variable temporal x y el operando b se hace igual al operando x.

Test&Set (x)

```
Test&Set (x) {  
    temp = x ;  
    x = 1 ;  
    return (temp) ;  
}  
/* x compartida */
```

x86

```
mov reg,1  
xchg reg,mem  
reg ↔ mem
```

Fetch&Oper(x,a)

```
Fetch&Add(x,a) {  
    temp = x ;  
    x = x + a ;  
    return (temp)  
}/* x compartida,  
a local */
```

x86

```
lock xadd reg,mem  
reg ← mem |  
mem ← reg+mem
```

Compare&Swap(a,b,x)

```
Compare&Swap(a,b,x){  
    if (a==x) {  
        temp=x ;  
        x=b; b=temp ;  
    }/* x compartida,  
a y b locales */
```

x86

```
lock cmpxchg mem,reg  
if eax=mem  
then mem ← reg  
else eax ← mem
```

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?

$$2^2 * 2^{30} / 2^6 = 2^{26} \text{ entradas}$$

- 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?

$$16 + 1 = 17$$

- 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?

$$16\text{GBytes} = 2^{34} \text{ Bytes}$$

$$2^{34} / 2^6 = 2^{28} \text{ líneas}$$

- 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?

$$8 + 1 = 9 \text{ bits}$$

- 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?

$$8\text{GBytes} = 2^{33} \text{ Bytes}$$

$$2^{33} / 2^7 = 2^{26} \text{ líneas}$$

- 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?

$$8 + 1 = 9 \text{ bits}$$

- 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?

$$2^3 * 2^{30} / 2^7 = 2^{26} \text{ entradas}$$

- 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?

Mantener la coherencia de cache:

¿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 $W \rightarrow W$ (sí respeta los demás), e inicialmente $X=Y=0$?

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

R=0; R=1; R=2

Si respeta el orden $W \rightarrow W$: $R=1$: $R=2$

¿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.

h=r1:

do

compare&swap(r2,b,k): // si r2==k k y b se intercambian

```
while (b==r3):
```

r1=1

$$r^2=0$$

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, Z son variables en memoria compartida y r1 es un registro de P1), al final se podría tener $Y=0$

F

P1:	P2:
<i>o puede</i> <i>que el</i> <i>o el RAW</i> <i>W(Y)</i>	(1) while ($Z==0$) { }; (2) $r1=X$; <i>RAW en P1</i> (3) $Y=r1$;
<i>R(Z)</i>	(a) $X=1$; <i>W(X)</i> (b) $Y=2$; <i>W(Y)</i> (c) $Z=1$; <i>W(Z)</i>

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:
<i>No puedo</i> <i>por el</i> <i>RAW</i> <i>R(Z)</i> <i>R(X)</i> <i>W(Y)</i>	(1) while ($Z==0$) { }; (2) $r1=X$; <i>RAW en P1</i> (3) $Y=r1$;
<i>R(X)</i>	(a) $X=1$; <i>W(X)</i> (b) $Y=2$; <i>W(Y)</i> (c) $Z=1$; <i>W(Z)</i>

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);
```

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 $W \rightarrow R$?

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 $W \rightarrow W$ y $W \rightarrow R$, pero sí los $R \rightarrow RW$.

(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 $W \rightarrow W$?

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

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

(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 cache 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 $W \rightarrow R$ (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 $W \rightarrow W$ 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 $W \rightarrow W$ (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 $W \rightarrow W$ 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 $W \rightarrow W$ (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)

Tema-3.pdf



Blancabril



Arquitectura de Computadores



2º Grado en Ingeniería Informática



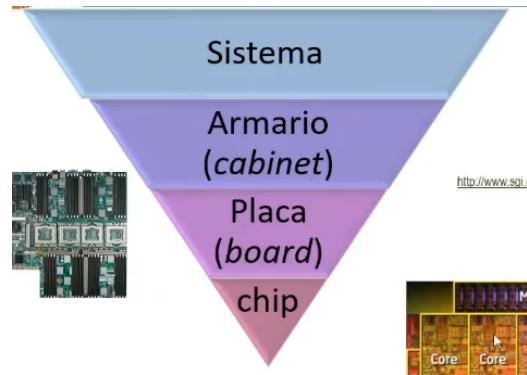
**Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada**



CLASIFICACIÓN DE ARQUITECTURAS CON TLP EXPLÍCITO Y UNA INSTANCIA DE SISTEMA OPERATIVO

Considerando que tenemos un único proceso que está constituido por diversas hebras que comparten la zona de memoria de ese proceso.

Multiprocesador: ejecuta varias hebras en paralelo en una máquina con varios procesadores. Existen diversos niveles de empaquetamiento del multiprocesador: dado, encapsulado, chasis y sistema.



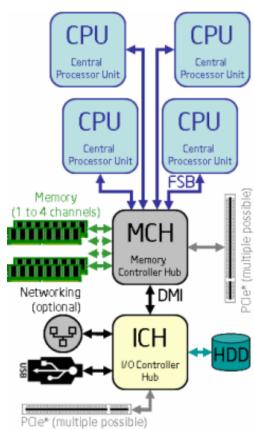
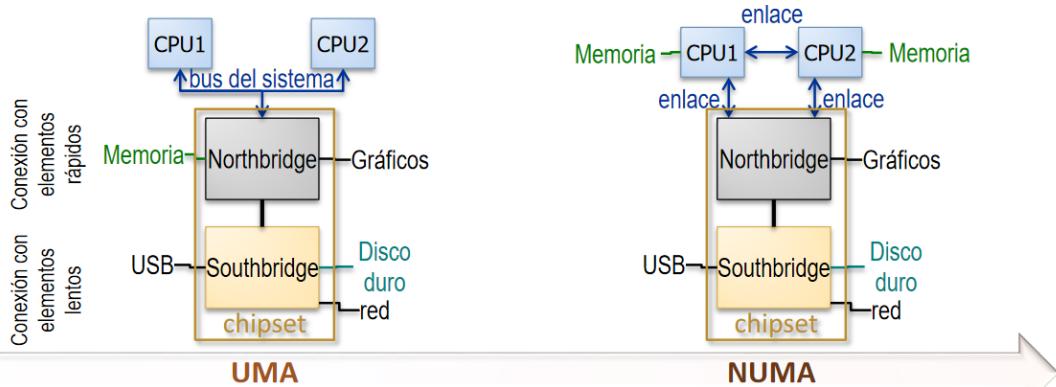
Multicore o CMP: se ejecutan varias hebras en paralelo en un chip de procesamiento multicore, donde cada hebra es un core distinto.

Core multithread: core que modifican su arquitectura ILP para ejecutar threads concurrentemente o en paralelo.

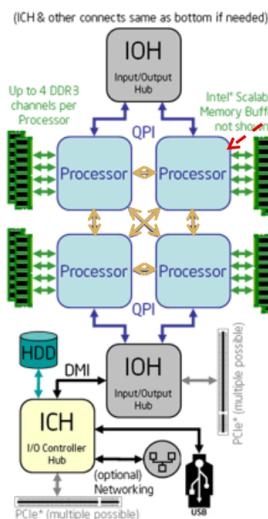
MULTIPROCESADORES SEGÚN LA ESTRUCTURA DE MEMORIA

- **Memoria Centralizada (UMA):** la memoria se encuentra centralizada pero pueden acceder a ella todos los procesadores. El acceso a la memoria es uniforme y el tiempo de acceso es similar. Recordamos que tiene mayor latencia y es poco estable.
- **Memoria Distribuida (NUMA):** todos los procesadores comparten el mapa de memoria, esa memoria está físicamente distribuida por módulos a cada procesador. Tiene menor latencia y es escalable.

Tanto en un caso como en otro, el mapa de memoria es compartido por todos los procesadores que están en la máquina. Lo que ocurre es que dependiendo donde se encuentre esa dirección de memoria, el tiempo de acceso será menor o mayor.



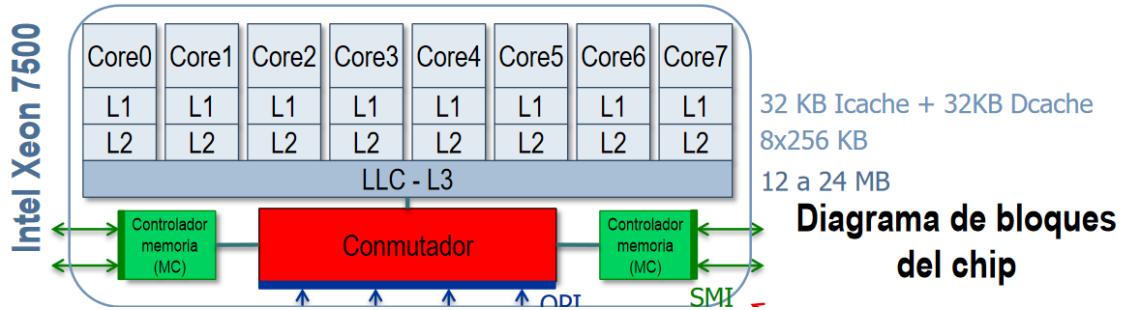
Se trata de un **multiprocesador UMA** donde podemos ver 4 CPUs que todas están conectadas a un puente norte que da acceso a la memoria compartida con el controlador de memoria.



Como podemos ver, se trata de un **multiprocesador NUMA** donde cada procesador tiene su memoria local. Los procesadores se comunican a través de los enlaces para poder acceder a direcciones de memoria que se encuentran en otro procesador.

MULTICORES

Los **multicore**s ejecutan hebras en paralelo en un **chip de procesamiento multicore**, entonces cada hebra va un core distinto.



La tecnología ha permitido integrar multiprocesadores dentro de un circuito integrado. Tenemos varios núcleos con su nivel de caché local y después en este ejemplo tienen un nivel compartido de caché (**LLC -- Last Level Cache**), incluyen también los controladores de memoria y un conmutador que permite acceder a la memoria o bien acceder a través de un enlace (QPI) a otro multiprocesador.

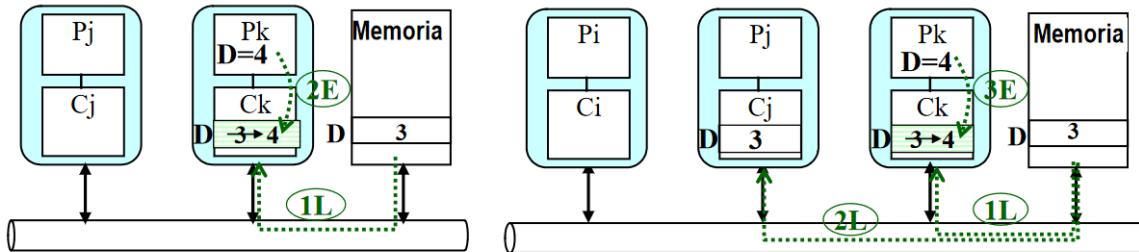
SISTEMA DE MEMORIA EN MULTIPROCESADORES

Cada procesador en un multiprocesador tiene:

- **Memoria caché:** memoria local.
- **Memoria principal.**
- **Controladores de memoria:** permiten determinar las señales que van a la memoria, mantener los ciclos de refresco de la memoria principal y determina si la dirección a la que quiere acceder se encuentra en la memoria caché o en otro sitio de memoria.
- **Buffer:**
 - Buffer de escritura/almacenamiento
 - Buffer que combinan escrituras/almacenamientos
- **Medio de comunicación entre los procesadores y la memoria:** red de interconexión (numa) o buses.

En un multiprocesador, la **comunicación de datos** entre procesadores se realiza a través de lectura y escritura en esa memoria que puede estar distribuida o localizada.

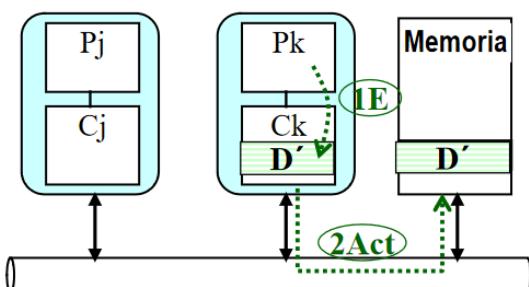
CONCEPTO DE COHERENCIA EN EL SISTEMA DE MEMORIA



Tenemos una memoria compartida por dos procesadores (J y K), los cuales tienen una memoria caché local y una memoria principal compartida. Pero claro, como podemos ver en la primera imagen, en la caché del procesador K aparece un 4 cuando en la memoria es un 3 y habría una incoherencia.

MÉTODOS DE ACTUALIZACIÓN DE MEMORIA PRINCIPAL

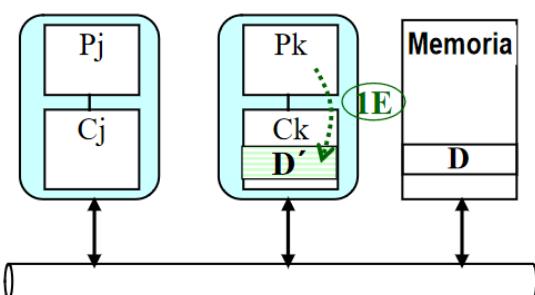
ESCRITURA INMEDIATA



Este método consiste en que cada vez que un procesador escribe en su caché, también modifica en la memoria principal.

Problema: consume ancho de banda porque cualquier cambio en la caché tiene que pasar por el bus.

POSESCRITURA



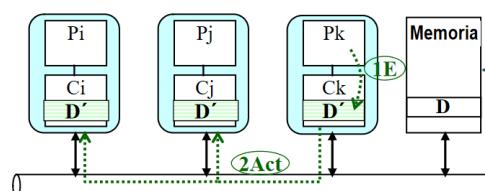
Este método es el más utilizado y consiste en que cuando se modifica la línea en la memoria caché, se permite que haya esa incoherencia hasta el momento en que esa línea se desaloja de la memoria caché.

Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

El problema viene cuando tenemos otros procesadores dentro de la máquina.

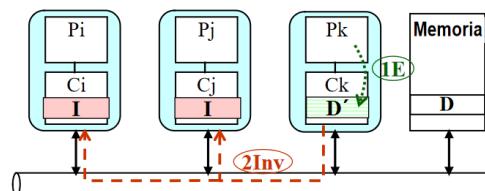
ESCRITURA CON ACTUALIZACIÓN



Cuando se produzca una escritura en su línea de caché, actualiza todas las cachés.

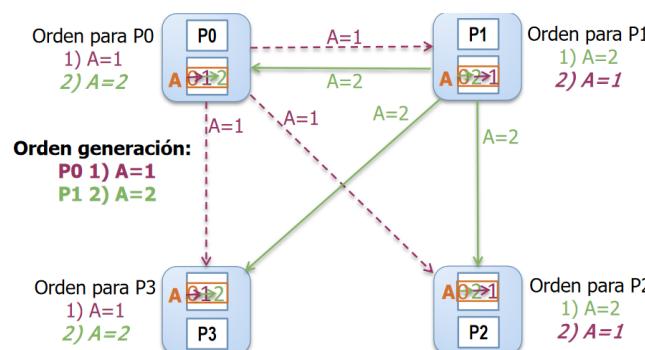
Problema: igual que antes, hay mucho tráfico.

ESCRITURA CON INVALIDACIÓN



En este método cuando se modifica una línea de caché, se produce un ciclo de invalidación que hace las copias de esa línea en otras cachés las pone como no válidas y cuando esos procesadores van a acceder a esa línea, tienen que acceder directamente a la memoria principal.

El problema principal viene cuando se hacen dos escrituras y se hace por escritura con actualización, según donde estén los nodos, pueden pisarse y cada uno puede acabar con un valor en la caché diferente.

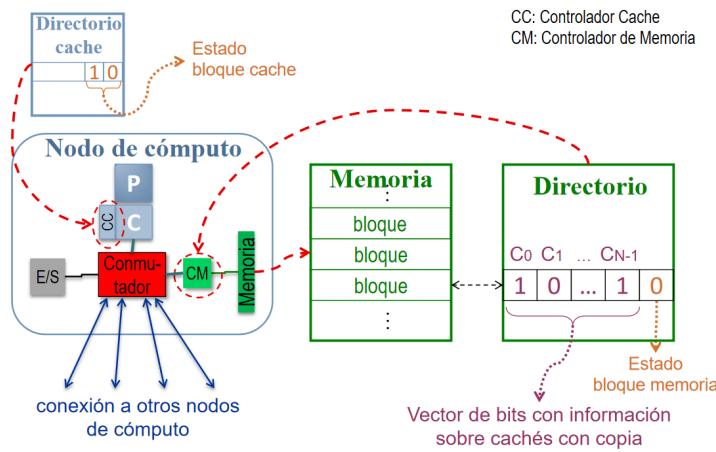


Para arreglar esto hay que:

- **Propagar las escrituras en una dirección** a todos los demás procesadores en un tiempo finito. En el caso de un bus, no hay problema ya que los paquetes de actualización son visibles a todos los nodos conectados.
- **Serializar las escrituras en una dirección** de manera que las escrituras deben verse en el mismo orden por todos los procesadores.

Regístrate en el webinar. Donde te contaremos cómo conseguirla

Fecha 26 de mayo · Hora 18:00 - 19:00



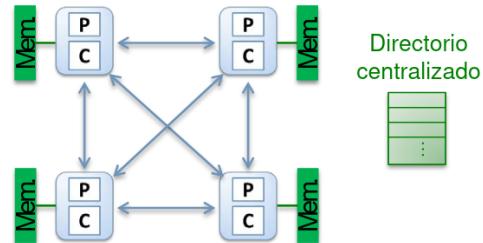
En el nodo de cómputo tenemos el **procesador** y la **caché** con el directorio donde se tiene la información de los bits de estado. Además tendremos un **conmutador** para acceder a la memoria local por el controlador de memoria y acceso a otros nodos de cómputo.

En el controlador nos encontramos con información de la ubicación posible de esos bloques de memoria que son los que se transfieren a caché cuando el nodo pide información y además nos encontramos con un **directorío de memoria** que proporciona información de la memoria, la ubicación en las distintas cachés de la máquina y otro bit que nos indica el estado del bloque de memoria para ver si hay incoherencias o no.

Como cada nodo tiene su memoria local, se introduce el **nodo home** que es aquel nodo donde se encuentra en su memoria local, la línea de caché que estamos considerando.

Hay dos tipos de directorios:

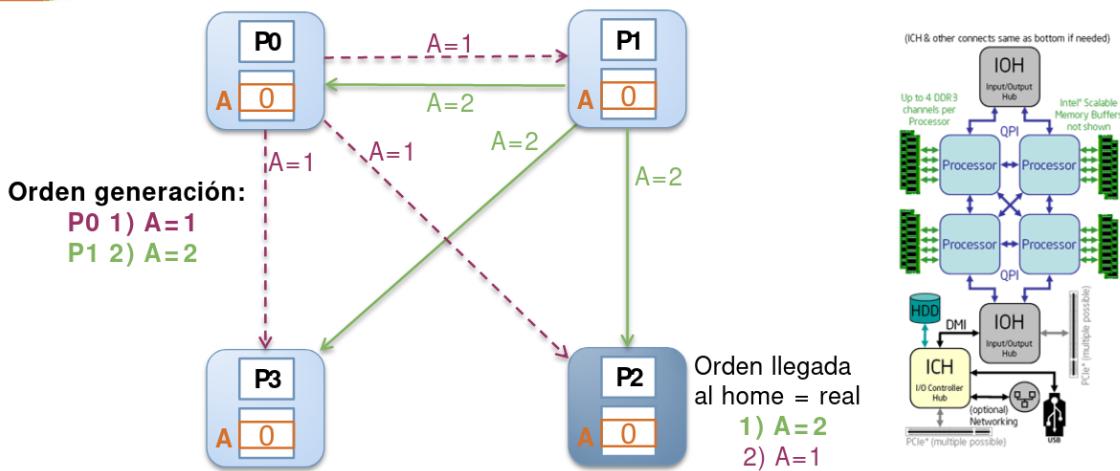
- **Centralizado:** este directorio está compartido por todos los nodos y contiene información de los bloques de todos los módulos de memoria.



- **Distribuido:** en este directorio las filas se distribuyen entre los nodos y el directorio de un nodo contiene información de los bloques de sus módulos de memoria.

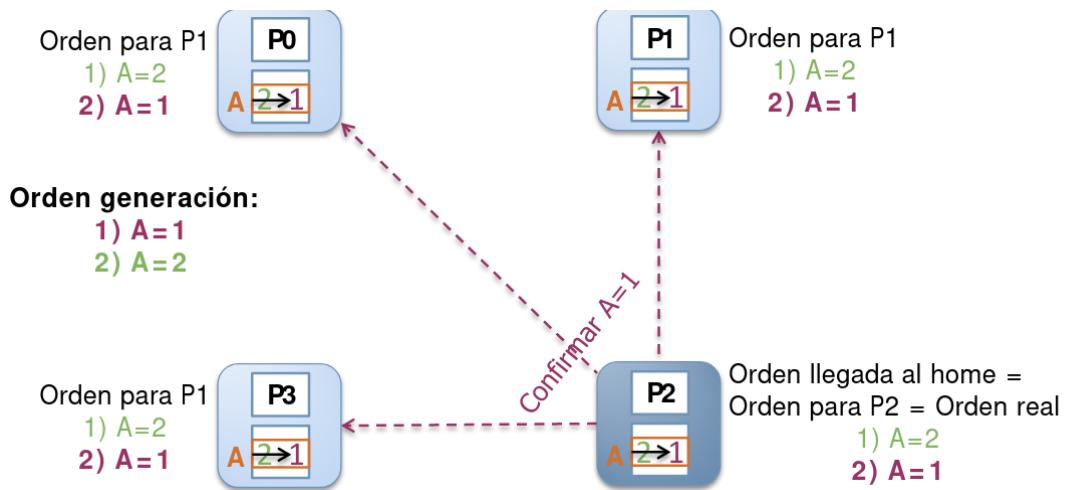


Para resolver el problema anterior con el nodo home lo haremos de la siguiente forma:



- Contenido inicial de las copias de la dirección A en **calabaza**. P0 escribe en A un 1 y, después, P1 escribe en A un 2.
- Se utiliza **actualización** para propagar las escrituras (las propagación se nota con flechas)
- El orden de llegada al home es el orden real para todos

Como es el nodo P2, el que tiene la linea que pertenece a A, pues le llega a el primero el A=2 y después el A=1 y es el encargado de transmitirlo a los otros nodos.



PROTOCOLOS DE COHERENCIA

PROTOCOLOS DE ESPIONAJE (SNOOPY)

Se utilizan cuando tenemos que todos los procesadores conectados a un bus o a una red de difusión, es decir, que lo que pasa en uno de los nodos, se conoce en el resto de los nodos.

PROTOCOLOS BASADOS EN DIRECTORIOS

Se utilizan para redes sin difusión o escalables.

Para definir un **protocolo de coherencia** hay que:

- Determinar qué **política de actualización** de la memoria vamos a usar: escritura inmediata, posescritura o mixta,
- Elegir un **tipo de coherencia entre cachés**: invalidación, actualización o mixta.
- Describir el comportamiento:
 - Posibles estados de los bloques en caché y memoria
 - Tipo de transferencias antes lecturas y escrituras del procesador al nodo y la consecuencia de la llegada de paquetes de otros nodos.
 - Definir transiciones de estados para un bloque en cache y en memoria.

PROTOCOLO MSI

Podemos definir el estado del bloque en caché:

- **Modificado (M)**: única copia del bloque válida en todo el sistema.
- **Compartido (C/S)**: ese bloque se ha traído a la caché y es válido (no ha sido modificado).
- **Inválido**: bloque que no está en la memoria caché o si estuvo en su momento, se ha invalidado.

Para definir el estado del bloque en memoria:

- **Válido**: todas las copias que hay en todas las cachés tienen el mismo valor.
- **Inválido**: hay un bloque en caché modificado.

Transferencias generadas por un nodo con caché:

- **Petición de lectura de un bloque (PtLec)**: por lectura con fallo de caché del procesador del nodo.

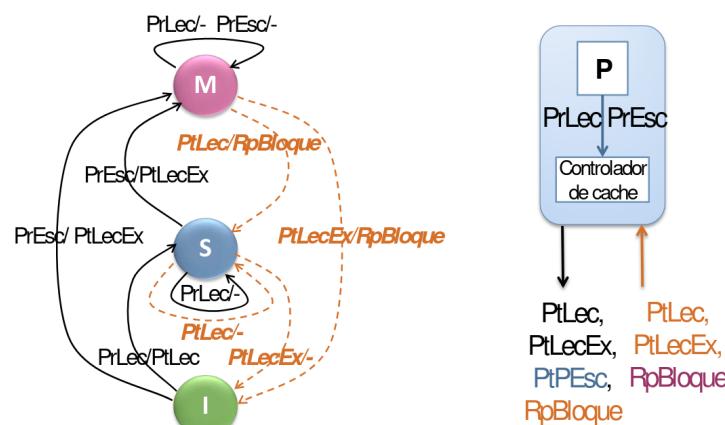
Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

Regístrate en el webinar. Donde te contaremos cómo conseguirla

Fecha 26 de mayo · Hora 18:00 - 19:00

- **Petición de acceso exclusivo (PtLecEx/ PtEsc):** ha solicitado el procesador una lectura o escritura en la caché pero como no la tiene, activa esa señal para avisar de que ese nodo quiere acceder para traérsela y modificarla.
- **Petición de postescritura:** se activa para indicar que se está poniendo el bloque a actualizar en el bus para que lo coja el nodo que lo necesite.
- **Respuesta con bloque:** al tener en estado modificado el bloque solicitado por una PtLec o PtLecEx recibida.



EST. ACT.	EVENTO	ACCIÓN	SIGUIENTE
Modificado (M)	PrLec/PrEsc		Modificado
	PtLec	Genera paquete respuesta (RpBloque)	Compartido
	PtLecEx	Genera paquete respuesta (RpBloque) Invalida copia local	Inválido
	Reemplazo	Genera paquete posescritura (PtPEsc)	Inválido
Compart. (S)	PrLec		Compartido
	PrEsc	Genera paquete PtLecEx	Modificado
	PtLec		Compartido
	PtLecEx	Invalida copia local	Inválido
Inválido (I)	PrLec	Genera paquete PtLec	Compartido
	PrEsc	Genera paquete PtLecEx	Modificado
	PtLec/PtLecEx		Inválido

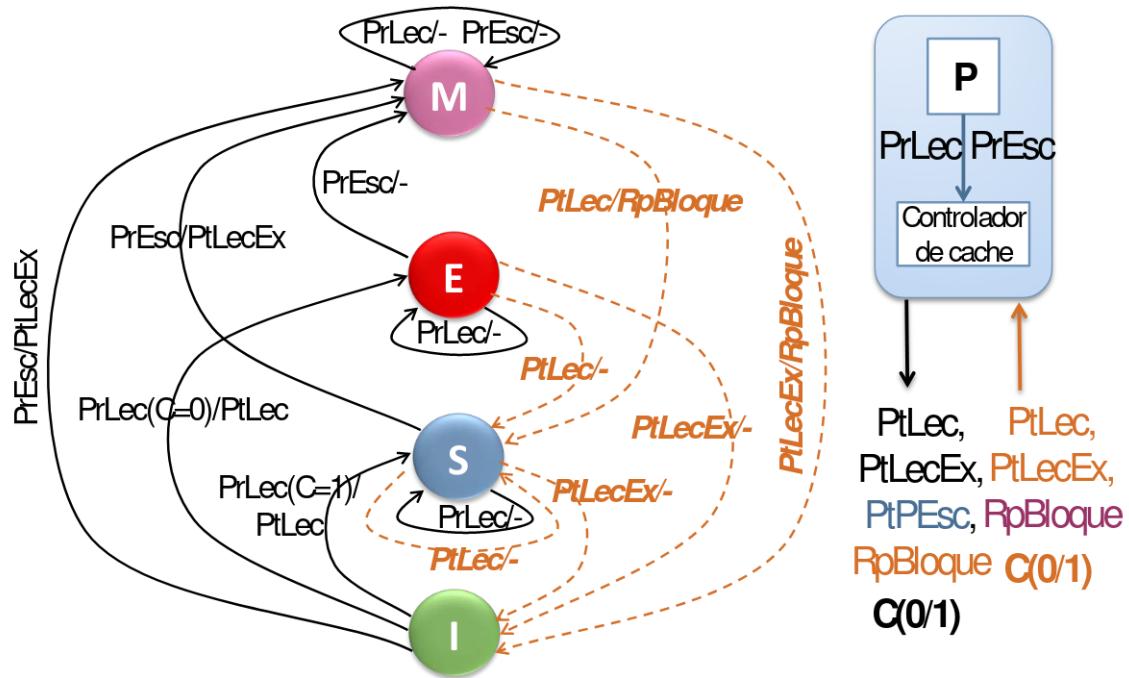
PROTOCOLO MESI

Está basado en el MSI pero definiendo otro estado:

- **Modificado (M):** única copia del bloque válida en todo el sistema.
- **Compartido (C/S):** ese bloque se ha traído a la caché y es válido (no ha sido modificado).
- **Exclusivo (E):** única copia válida en las cachés y esa linea es coherente con la memoria principal.
- **Inválido:** bloque que no está en la memoria caché o si estuvo en su momento, se ha invalidado.

Para definir el estado del bloque en memoria:

- **Válido:** todas las copias que hay en todas las cachés tienen el mismo valor.
- **Inválido:** hay un bloque en caché modificado.



Si es una petición de lectura y es la única copia ($C = 0$) entonces es exclusivo. Si es escritura se pasa directamente al estado modificado.

Modificado (M)	PrLec/PrEsc		Modificado
	PtLec	Genera RpBloque	Compartido
	PtLecEx	Genera RpBloque. Invalida copia local	Inválido
Exclusivo (E)	PtLec		Exclusivo
	PrEsc		Modificado
	PtLec		Compartido
Compartido (S)	PtLecEx	Invalida copia local	Inválido
	PrLec/PtLec		Compartido
	PrEsc	Genera PtLecEx	Modificado
Inválido (I)	PtLecEx	Invalida copia local	Inválido
	PrLec (C=1)	Genera PtLec	Compartido
	PrLec (G=0)	Genera PtLec	Exclusivo
	PrEsc	Genera PtLecEx	Modificado
	PtLec/PtLecEx		Inválido

PROTOCOLO MSI CON DIRECTORIOS

Podemos definir el estado del bloque en caché:

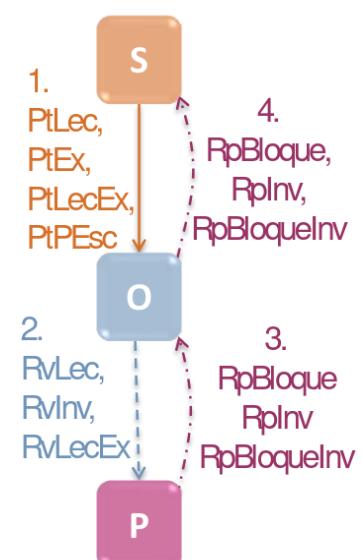
- **Modificado (M)**: única copia del bloque válida en todo el sistema.
- **Compartido (C/S)**: ese bloque se ha traído a la caché y es válido (no ha sido modificado).
- **Inválido**: bloque que no está en la memoria caché o si estuvo en su momento, se ha invalidado.

Para definir el estado del bloque en memoria:

- **Válido**: todas las copias que hay en todas las cachés tienen el mismo valor.
- **Inválido**: hay un bloque en caché modificado.

Transferencias de tipo de paquetes:

- **Nodos:**
 - **Solicitante (S)**
 - **Origen (O)**
 - **Modificado (M)**
 - **Propietario (P)**
 - **Compartidor (C)**
- **Petición de nodo S a O**: lectura de un bloque, lectura con acceso exclusivo, petición de acceso exclusivo, petición de acceso exclusivo sin lectura, posescritura.
- **Reenvío de petición de O a (P, C, M)**: invalidación, lectura.
- **Respuesta de P a O / O a S**: respuesta con bloque, respuesta con o sin bloque confirmado.



CONSISTENCIA DEL SISTEMA DE MEMORIA

La **consistencia** establece el orden en el cual las operaciones de memoria deben parecer haberse realizado a distintas variables que están en diferentes direcciones.

La **coherencia** sólo abarca las operaciones realizadas por múltiples componentes a una única dirección.

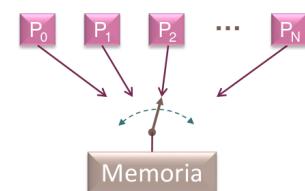
CONSISTENCIA SECUENCIAL

Es el modelo que espera el programador y requiere:

- Todas las operaciones de un único procesador parezcan ejecutarse en el orden descrito en el programa de entrada al procesador, es decir una hebra donde los accesos a memoria se realicen en el orden en el que aparecen en el código.
- Todas las operaciones de memoria parezcan ser ejecutadas una cada vez

En la consistencia secuencial, el sistema de memoria se presenta como una memoria global conectada a todos los procesadores por un conmutador central.

La memoria definirá un orden definido de acceso.



MODELOS DE CONSISTENCIA RELAJADOS

Los modelos de consistencia relajados son aquellos en los cuales no se respetan los requisitos para garantizar la consistencia secuencial, sino que relajan algunos requisitos como:

- **Orden de programa**, es decir que pueden cambiar el orden de las instrucciones.

ST R1, OX1C (R2) → escritura

LD R3, 0X2 D (R3) → lectura

MULT R3, R3, R6

Esto quiere decir que tengo primero una escritura y luego una lectura. En muchos casos, el procesador hace una escritura y tarda un determinado tiempo en hacerlo pero si detecta que acceden a direcciones diferentes, el procesador no espera y hace la lectura porque no hay dependencia entre ellas.

Estudia tu Grado Universitario en Berlín en inglés y 100% financiado

Escoge entre las 37 carreras universitarias oficiales en inglés que SRH Berlín ofrece y haz despegar tu futuro profesional.

- **Atomicidad**, que quiere decir que hay modelos que permiten a un procesador poder ver el valor escrito por otro.

Los modelos relajados de memoria comprenden:

- Las órdenes de acceso a memoria que no garantizan el sistema de memoria.
- Mecanismos que ofrece el hardware para garantizar un orden cuando sea necesario por si en algún momento necesito que se mantenga ese orden de manera que se fuerce cuando sea necesario.

Ejemplos:

Modelo	Orden del programa relajado W→R W→W R→RW			Orden global Lec. anticipada propia de otro	Instrucciones para garantizar los órdenes relajados por el modelo
Sparc-TSO, x86-TSO	Si			Si	I-m-e (instruc. lectura-modificación-escritura atómica)
Sparc-PSO	Si	Si		Si	I-m-e, STBAR (instrucción <i>STore BARrier</i>)
Sparc-RMO	Si	Si	Si	Si	MEMBAR (instrucción <i>MEMbry BARrier</i>)
PowerPC	Si	Si	Si	Si	SYNC, ISYNC (instrucciones <i>SYNChronization</i>)
Itanium	Si	Si	Si	Si	LD.ACQ, ST.REL, MF (ACQuisition LoaD, RElease STore, Memory Fence), y cmpxchg8.acq y otras I-m-e
ARMv7	Si	Si	Si	Si	DMB (Data Memory Barrier)
ARMv8	Si	Si	Si	Si	LDA LDAR, STL STLR (LoaD-Acquire, STore-reLease 32b 64b), LDAXR LDAXR, STLEX STLXR (LoaD-Acquire eXclusive, Store-reLease eXclusive 32b 64b), DMB

Inicialmente k1=k2=0 P1 k1=1; if (k2=0) { Sección crítica };	P2 k2=1; if (k1=0) { Sección crítica };	NO se comporta como SC los que relajan el orden W→R	①
P1 A=1; P2 if (A=1) B=1;	A=B=0 P3 if (B=1) reg1=A;	NO se comporta como SC los que no garantizan atomicidad	②
Inicialmente A= 0 P1 A=1; k=1;	P2 while (k=0) {}; copia=A;	NO se comporta como SC los que relajan el orden W→Wo R→R	③

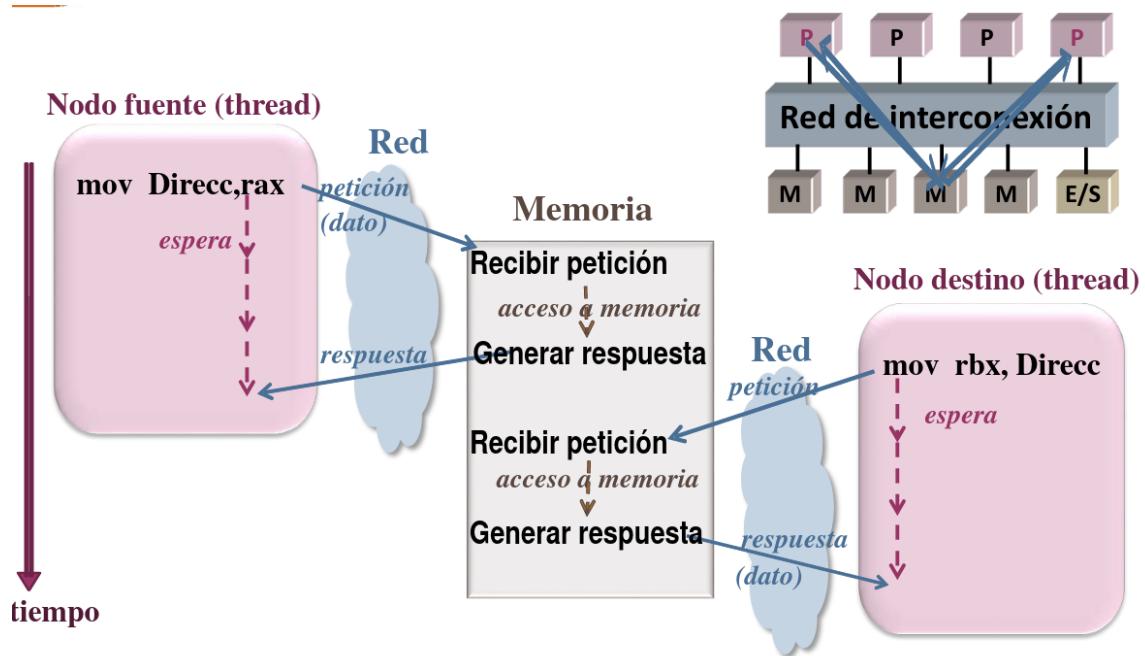
Hay que tener cuidado porque igual para conseguir velocidad, perdemos el resultado final.

En el ejemplo 1 puede que se ejecute la lectura antes que la escritura, y se ejecuta el if antes de hacer el k1=1.

En el segundo no se garantiza atomicidad y que se ejecute malamente.

Por último, si no se respeta W → W o R → R, fallaría y no sabemos realmente el resultado final.

COMUNICACIÓN EN MULTIPROCESADORES



Se trata de una memoria compartida, a la cual pueden acceder todos los procesadores de la misma manera. El nodo fuente que está ejecutando una hebra y se escribe en el mapa de memoria compartido de la máquina, y otro procesador puede realizar una lectura de esa dirección de memoria y cargarlo en un registro interno. Es necesario sincronizar la escritura con la lectura para que la lectura lea el dato correcto.

COMUNICACIÓN UNO A UNO

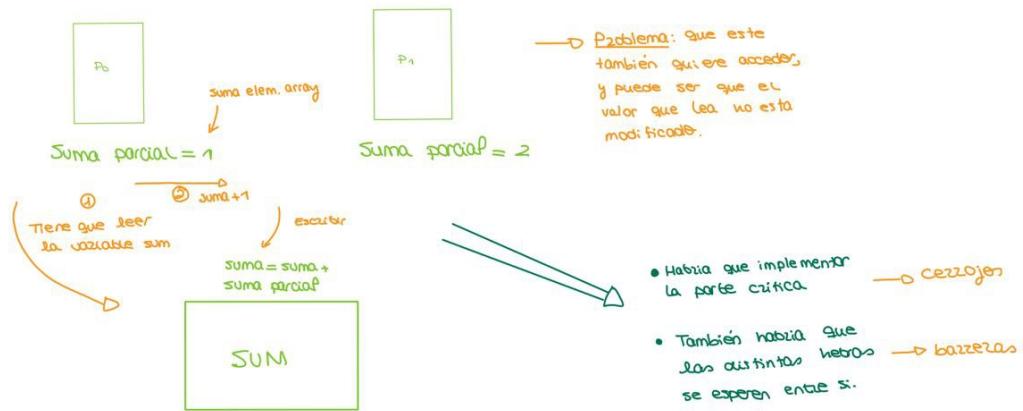
Paralela (K=0)	
P1	P2
...	...
A=1;	while (K=0) { };
K=1;	copia= A ;
...	...

Se debe garantizar que el proceso que recibe lea la variable compartida cuando el proceso 1 ya haya escrito en ella. Para ello utilizamos este bucle while que lo que hace es analizar la variable K mientras se cambia la variable A y cuando A cambia, K posteriormente también y por tanto sale del bucle.

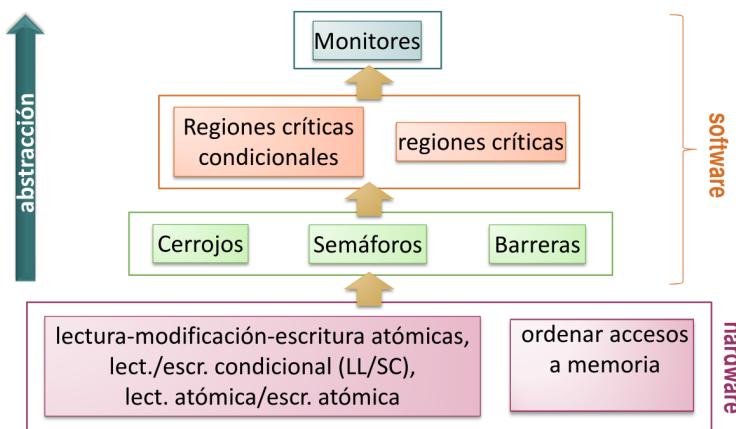
COMUNICACIÓN COLECTIVA

Secuencial	Paralela (sum=0)
<pre>for (i=0 ; i<n ; i++) { sum = sum + a[i]; }</pre>	<pre>for (i=ithread ; i<n ; i=i+nthread) { sump = sump + a[i]; } sum = sum + sump; /* SC, sum compart. */ if (ithread==0) printf(sum);</pre>

En paralelo se realiza un bucle mediante el número de hebras para hacer una distribución de la carga mediante Round Robin



SOPORTE HARDWARE Y SOFTWARE DE SINCRONIZACIÓN



La lectura-modificación-escritura atómica significa que entre que se empieza la lectura y se termina con la escritura, ningún otro procesador va a poder acceder a la posición de memoria a la que está accediendo.

Lecturas enlazadas y escritura condicionales, son instrucciones de escritura que están relacionadas entre sí que se usa para implementar cerrojos y barreras.

CERROJOS

Los cerrojos son variables que se encuentran en memoria compartida (variable K), que permiten la sincronización con dos operaciones.

La primera operación se denomina **cierre del cerrojo** donde se adquiere el derecho para acceder a la sección crítica y al mismo tiempo cierra el cerrojo para que las otras hebras no puedan acceder.

La siguiente operación es **apertura de cerrojo** que libera a una de las hebras que esperan el acceso a una sección crítica y en el caso de que no haya hebras, la siguiente que ejecute lock() entra sin espera.

Secuencial	Paralela
<pre>for (i=0 ; i<n ; i++) { sum = sum + a[i]; }</pre>	<pre>for (i=ithread ; i<n ; i=i+nthread) { sump = sump + a[i]; } lock(k); sum = sum + sump; /* SC, sum compart. */ unlock(k);</pre>

Tus codos aquí.

Ánimo con los exámenes.
Para lo que está por venir, cuenta con Infojobs.

InfoJobs

El portal líder de empleo.

Las alternativas para implementar la espera son la **espera ocupada**, donde las hebras siguen intentando cerrar el cerrojo y la **suspensión del proceso** en una cola.

MÉTODOS

- **Método de adquisición:** método por el cual la hebra trata de adquirir el derecho a utilizar las direcciones compartidas y se implementan normalmente con variables atómicas o bien utilizando LL/SC (carga enlazada, almacenamiento condicional).
- **Método de espera:** método por el cual cada hebra espera a adquirir el derecho a utilizar unas direcciones compartidas y se implementa con la espera ocupada o el bloqueo.
- **Método de liberación:** método utilizado por un thread para liberar a una o varias hebras.

Esto se implementa con la variables compartida K, cuando está a 0, significa que está abierto y cuándo se pone a 1, es que se encuentra cerrado.

Para cerrar el cerrojo se usa la función lock() y para la apertura se usa unlock(). Esto se debe realizar de forma atómica para que dos hebras no pasen a la sección crítica.

```
lock (k)
lock(k) {
    while (leer-asignar_1-escribir (k) == 1) {};
} /* k compartida */
```

```
unlock (k)
unlock(k) {
    k = 0 ;
} /* k compartida */
```

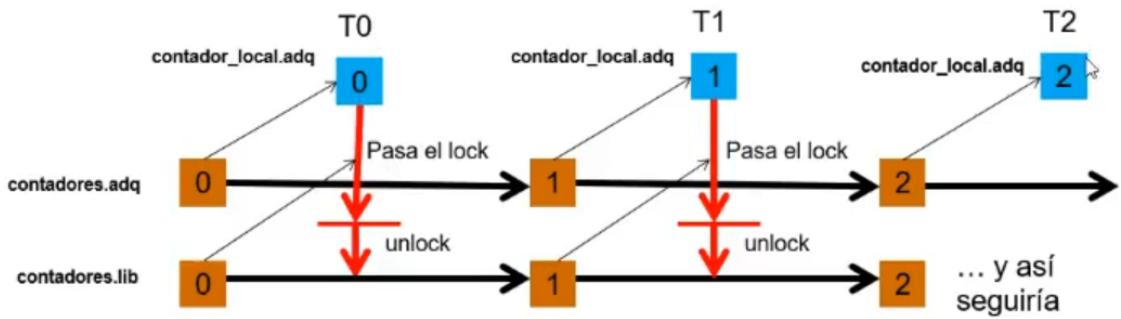
CERROJOS CON ETIQUETA

Este tipo de cerrojo busca garantizar que las hebras que vayan llegando y se esperen a la apertura del cerrojo, en el mismo orden se vayan liberando (FIFO), de manera que todas las hebras esperen un tiempo equivalente.

```
lock (contadores)
contador_local_adq = contadores.adq;
contadores.adq = (contadores.adq + 1) mod max_flujos_control;
while (contador_local_adq <> contadores.lib) {};
```

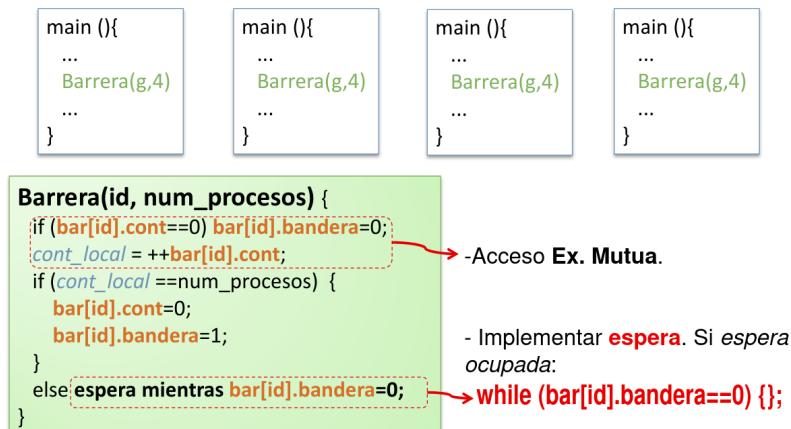
Azul: variable local
Naranja: variable compartida.

```
unlock (contadores)
contadores.lib = (contadores.lib + 1) mod max_flujos_control;
```



BARRERAS

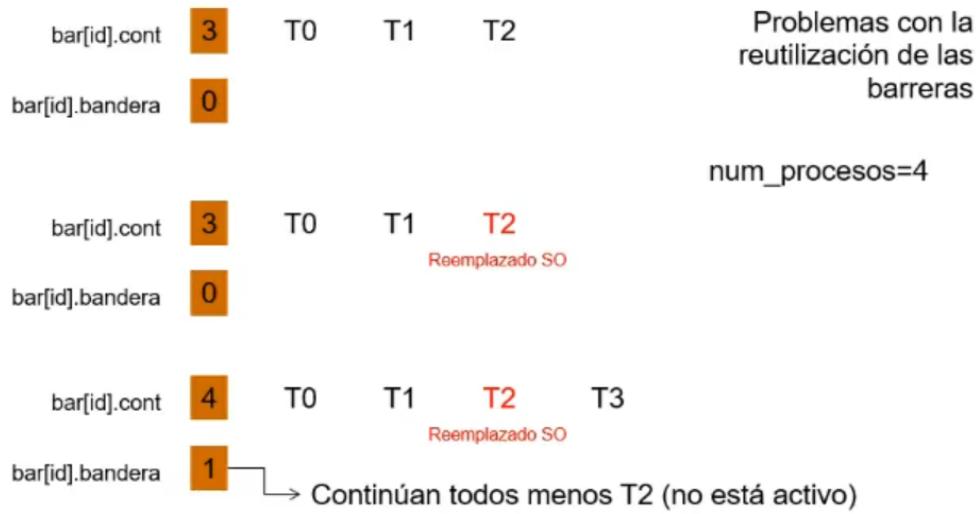
Las barreras permiten sincronizar varias hebras que se esperan hasta que han llegado todas y por tanto, podrían continuar.



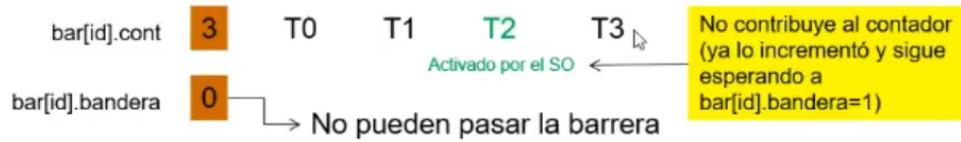
Cada barrera tiene un identificador y el número de procesos que tienen que sincronizarse. Las variables en azul son variables locales y las naranjas son variables de memoria compartida. Cuando llega una hebra lo primero que se comprueba es si el contador se encuentra a 0 y eso significa que no ha llegado ningún otro valor, entonces la bandera se pone a 0 para no permitir el paso y se incrementa el contador local.

Si el contador local es igual al número de procesos, la bandera se pone a 1 entonces pueden pasar todas las hebras y pone el contador de nuevo a 0.

BARRERA SIN REUTILIZACIÓN



Las hebras llegan otra vez a la misma barrera (ej. está en un bucle)



Para solucionar esto, utilizamos la barrera sin reutilización:

Barrera *sense-reversing*

```
Barrera(id, num_procesos) {
    bandera_local = !(bandera_local) //se complementa bandera local
    lock(bar[id].cerrojo);
    cont_local = ++bar[id].cont //cont_local es privada
    unlock(bar[id].cerrojo);
    if (cont_local == num_procesos) {
        bar[id].cont = 0; //se hace 0 el cont. de la barrera
        bar[id].bandera = bandera_local; //para liberar thread en espera
    }
    else while (bar[id].bandera != bandera_local) {}; //espera ocupada
}
```

Se hace con una variable de bandera local para saber si se puede continuar o no.

APOYO HARDWARE A PRIMITIVAS SOFTWARE

INSTRUCCIONES DE LECTURA-MODIFICACIÓN-ESCRITURA ATÓMICAS

Test&Set(x): lectura de una variable compartida x y se escribe un 1 en esa variable x y proporciona lo que hay escrito en la variable antes de ser cambiada.

Fetch&Oper(x,a): lectura de dos variables compartidas x y a, se guarda en una variable local temporal la primera variable y se realiza un sumatorio con la segunda variable, devolviendo esa sumatoria.

Compare&Swap(a,b,x): tres operandos, si x es igual a uno de los operandos, a, se almacena en la variable temporal x y el operando b se hace igual al operando x.

Test&Set (x)

```
Test&Set (x) {
    temp = x ;
    x = 1 ;
    return (temp) ;
}
/* x compartida */
```

↓ x86

```
mov reg,1
xchg reg,mem
reg ↔ mem
```

Fetch&Oper(x,a)

```
Fetch&Add(x,a) {
    temp = x ;
    x = x + a ;
    return (temp)
}/* x compartida,
a local */
```

↓ x86

```
lock xadd reg,mem
reg ← mem | 
mem ← reg+mem
```

Compare&Swap(a,b,x)

```
Compare&Swap(a,b,x){
    if (a==x) {
        temp=x ;
        x=b; b=temp ;
    }
}/* x compartida,
a y b locales */
```

↓ x86

```
lock cmpxchg mem,reg
if eax=mem
then mem ← reg
else eax ← mem
```