Multiprocessadors

FACULTAT D'INFORMÀTICA DE BARCELONA

Genís Bosch | MP | 2018

EJERCICIOS 1

Ejercicio 1.1

Considere la ejecución de los siguientes cuatro programas en cada una de las tres máquinas que se indican

	Tiempo de ejecución (segundos)		
Programa	computador A	computador B	computador C
programa 1	1	10	20
programa 2	1000	100	20
programa 3	500	1000	50
programa 4	100	800	100

Suponga que se han ejecutado 100.000.000 de instrucciones en cada uno de los 4 programas.

Pregunta 1: *Calcule los MIPS de cada programa en cada una de las 3 máquinas.*

Considere que los 4 programas representan una carga de trabajo. Además, suponga que los programas se ponderan con el mismo peso al considerar la carga de trabajo.

Pregunta 2: *Calcule el tiempo medio de ejecución por instrucción para cada máquina.*

Pregunta 3: *Calcule los MIPS para cada máquina. Efectúe el calculo a partir de los MIPS de cada programa.*

Ejercicio 1.2

Disponemos de 3 versiones de un procesador: A, B y C y de un conjunto de programas de prueba que se utiliza para evaluarlos. El procesador B es una versión mejorada del procesador C y el procesador A es una versión mejorada del procesador B. En los procesadores A y B la cache es no bloqueante y los load son bloqueantes. En el procesador C la cache es bloqueante. Todos los procesadores funcionan a la misma frecuencia.

El CPI medio medido en la versión A es 1.25. Por otro lado conocemos que el CPI medio de A es un 30% menor que el CPI medio de B.

Pregunta 1: *Calcule el CPI de B. Así mismo calcule la ganancia en rendimiento de A respecto de B.*

El lenguaje máquina de los procesadores A y B se ha ampliado con instrucciones de prebúsqueda, las cuales no estan disponibles en el procesador C. Conocemos que en ambos procesadores el número de instrucciones ejecutadas es un 3% mayor que en el procesador C.

En los procesadores A y B el CPI de las instrucciones de prebúsqueda es uno.

Pregunta 2: *Calcule el CPI medio del resto de las instrucciones (que no son de prebúsqueda) en el procesador B.*

Sea G la ganancia en rendimiento de A respecto de B y sea RR la reducción en rendimiento de B respecto de A. Ambos procesadores funcionan a la misma frecuencia.

Pregunta 3: *Exprese G y RR en función del tiempo de ejecución de A y B. Muestre una expresión que permita calcular G en función de RR y otra expresión que permite el cálculo recíproco.*

El mayor IPC de A respecto de B está determinado por una mejora arquitectónica que reduce la frecuencia de fallos por instrucción (f), sin incrementar el número de instrucciones ejecutadas. Todos los otros parámetros arquitectónicos permanecen invariables. En concreto, el CPI de la unidad de proceso (acierto siempre en cache, CPI_{UP}) es el mismo.

En las siguientes preguntas suponga que el CPI de la unidad de proceso (CPI_{UP}) es uno y la penalización por fallo son 10 ciclos.

Pregunta 4: *Calcule los fallos por instrucción en los procesadores A y B.*

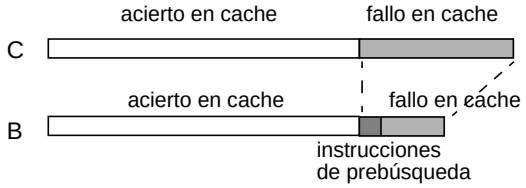
Pregunta 5: *Calcule la reducción en fallos por instrucción del procesador A respecto del procesador B.*

El cálculo de la ganancia mediante la ley de Amdahl utiliza los parámetros g y m, siendo m la fracción de tiempo, del caso sin mejora, donde se utiliza la mejora y g la ganancia en esta fracción de tiempo.

Pregunta 6: *Para los procesadores A y B determine expresiones para los parámetros g y m en función de factores tales como: CPI medio (con jerarquía de memoria), CPI de la unidad de proceso*

(acierto siempre en cache, CPI_{UP}), la frecuencia de fallo por instruccion, la penalización por fallo o el número de instrucciones. Corrobore el valor de la ganancia de la pregunta a).

Al comparar el procesador B y el procesador C hay que considerar que las instrucciones de prebúsqueda se incluyen en el tramo de tiempo donde se observa la mejora.



Suponga que el CPI del procesador C es 3.

Pregunta 7: *Para los procesadores B y C determine expresiones para los parámetros g y m en función de factores tales como: CPI medio (con jerarquía de memoria), CPI de la unidad de proceso (acierto siempre en cache, CPI_{UP}), la frecuencia de fallo por instruccion, la penalización por fallo o el número de instrucciones.*

Ejercicio 1.3

En un procesador A el CPI de las instrucciones de tipo multimedia es 2.5 y en una nueva generación del procesador (B), que funciona a la misma frecuencia, se consigue reducir el CPI de estas instrucciones a 1.5, ya que internamente la instrucción de lenguaje máquina multimedia se interpreta mediante una sola microinstrucción en lugar de las 2 microinstrucciones utilizadas en la versión A del procesador. En un conjunto de programas de prueba (P) las instrucciones multimedia representan un 20% de las ejecutadas. Al ejecutar este conjunto de programas de prueba en el procesador A se mide un CPI medio de 1.9. El CPI mínimo o ideal en los dos procesadores es 1.

Nota: los valores de CPI están calculados utilizando las instrucciones de lenguaje máquina.

Pregunta 1: *Calcule la ganancia de reducir el CPI en las instrucciones multimedia.*

Partiendo del procesador B se evalúa introducir la técnica multihilo para mejorar la productividad. Supondremos que la técnica multihilo es capaz de utilizar todos los ciclos perdidos, por

riesgos de datos y de secuenciamiento, al ejecutar un programa para ejecutar otro programa de forma concurrente. Esto es, se aprovechan todos los ciclos perdidos por riesgos de datos y de secuenciamiento. Además, no se incrementan los riesgos estructurales al utilizar la técnica multihilo.

En el procesador B, suponga que el tiempo de ejecución de otro conjunto de programas de prueba (P1) sin utilizar la técnica multihilo es 10 segundos, el CPI medido es 2.5 y los ciclos perdidos por instrucción por riesgos estructurales representan un 6% de los ciclos perdidos totales.

Pregunta 2: *En el procesador B calcule el número de ciclos por instrucción perdidos por riesgos de datos y de secuenciamiento.*

Pregunta 3: *Calcule la latencia media de inicio al utilizar la técnica multihilo en el procesador B. ¿Cuál es la ganancia potencial en productividad al utilizar la técnica multihilo en el procesador B?.*

Pregunta 4: *En 10 segundos cuantas veces se puede ejecutar el conjunto de programas de prueba P1 cuando se utiliza la técnica multihilo en el procesador B.*

Suponga que, cuando se utiliza la técnica multihilo en el procesador B, el conjunto de programa de prueba tarda en ejecutarse 4.37 segundos.

Pregunta 5: *Introducir la técnica multihilo representa incrementar la potencia consumida en un 10%. Calcule la ganancia energética potencial de introducir la técnica multihilo en el procesador B al ejecutar el conjunto de programas de prueba P1.*

Partiendo del procesador B, sin la técnica multihilo, también se evalúan dos posibilidades para incrementar la productividad en una nueva versión tecnológica: a) utilizar los transistores disponibles para reducir el CPI en un 10% (versión C) o b) utilizar los transistores disponibles para crear un chip con 2 procesadores idénticos (versión D). En los dos casos la frecuencia se mejora un 40%.

Note que en la versión D cada procesador puede estar ejecutando uno de los programas del conjunto de programas de prueba. Suponga que todos los procesadores inician y finalizan el trabajo al mismo tiempo y que el CPI de los programas es el mismo que en la versión B (uniprocesador).

En el procesador B, sin utilizar la técnica multihilo, suponga que el tiempo de ejecución de otro conjunto de programas de prueba (P2) es 10 segundos y el CPI medido es 2.5.

Pregunta 6: *Al ejecutar el conjunto de programas de prueba P2, calcule el tiempo de ejecución del conjunto de programas de prueba en la versión C.*

Pregunta 7: *Al ejecutar el conjunto de programas de prueba P2, calcule el tiempo de ejecución del conjunto de programas de prueba en la versión D.*

En la nueva versión tecnológica del procesador el consumo de potencia de la versión C es 10/16 el consumo de potencia de la versión B. En cambio en la versión D el consumo de potencia se mantiene igual al de la versión B.

Pregunta 8: *Al ejecutar el conjunto de programas de prueba P2, indique cuál de las versiones C o D es energéticamente mejor.*

Ejercicio 1.4

Un procesador utiliza una jerarquía de memoria con un nivel de cache. En este nivel se utiliza una cache para instrucciones y otra cache para datos. Las dos cache utilizan mapeo 4 asociativo, el tamaño de bloque es de 16 bytes y el tamaño de cada cache es 32Kbytes. La penalización por fallo de cache son 10 ciclos. Las caches son bloqueantes.

En este procesador se ejecuta el siguiente código.

L.A.N
do I = 1,100
A(I) = B(I) + C(I)
enddo

dirección		instrucción
1$:	200	load **R1**, 0(R2)
	204	load **R3**, 0(R4)
	208	add **R5**, R1, R3
	220	store R5, 0(R6)
	224	add **R2**, R2, #8
	228	add **R4**, R4, #8

direc.	instrucción
232	add **R6**, R6, #8
236	sub **R7**, R7, #1
240	bne R7, 1$

El 1º elemento de los vectores A, B y C está alineado a tamaño de bloque. El tamaño de un elemento de los vectores es de 8 bytes.

La 1ª instrucción del bucle está alineada a tamaño de bloque. El tamaño de una instrucción son 4 bytes.

Cuando empieza a ejecutarse el bucle las caches están vacías. Esto es, la información de estado de todos los contenedores indica bloque inválido.

Pregunta 1: *Calcule los fallos que se producen en la cache de instrucciones al ejecutar el bucle.*

Cuando se ejecuta el bucle y se acierta en cache, tanto al acceder al código como a los datos se mide un CPI de 4 / 3.

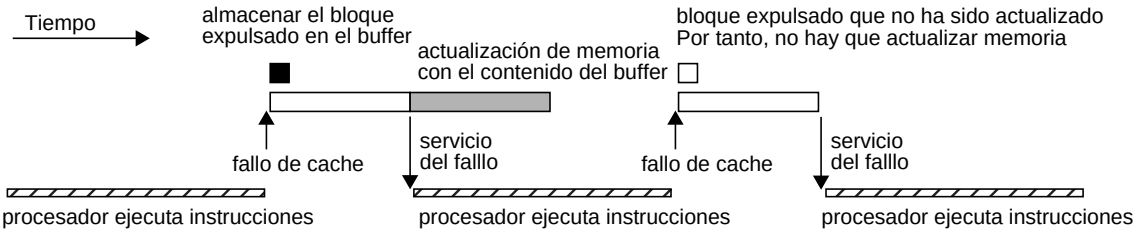
Pregunta 2: *Calcule el número de ciclos que tarda en ejecutarse la 1ª iteración del bucle.*

Pregunta 3: *Calcule el número de ciclos que tarda en ejecutarse la 2ª iteración del bucle.*

Pregunta 4: *Calcule el número de ciclos que tarda en ejecutarse el bucle.*

En la cache de datos se utiliza escritura retardada para mantener la coherencia en la jerarquía de memoria. Además, en caso de fallo en una operación de escritura se trae el bloque al 1º nivel de la jerarquía. Esto es, en una operación de escritura se actualiza el bloque en el 1º nivel de cache (si era fallo se ha traído antes) y el bloque se marca como actualizado. Posteriormente cuando el bloque debe expulsarse del contenedor para almacenar otro bloque (acción de reemplazo) se actualiza memoria.

El tiempo de actualizar memoria son 10 ciclos. Para no retardar el servicio del fallo de cache, el cual ha provocado la expulsión de un bloque que ha sido actualizado previamente, el bloque expulsado se almacena en un buffer y concurrentemente se inicia el servicio del fallo de cache (siguiente figura). Posteriormente cuando finaliza el servicio del fallo de cache se actualiza memoria con el contenido del buffer mientras concurrentemente el procesador sigue ejecutando instrucciones.



En las siguientes preguntas supondremos que todos los contenedores de cache almacenan bloques válidos y que los vectores A y B están almacenados en cache cuando se inicia la ejecución del bucle. Sólo se falla al acceder al vector C, lo cual en ocasiones requiere actualizar memoria.

Pregunta 5: *Justifique si en el bucle previo el tiempo entre dos fallos consecutivos de cache es mayor que el tiempo necesario para actualizar memoria.*

La potencia consumida por el procesador y el nivel de cache cuando no hay fallos es P julios/ciclo y cuando está bloqueado esperando el servicio de un fallo de cache es 0.2 x P. La potencia consumida cuando se accede a memoria (fallo o actualización de memoria) es 0.1 x P.

Pregunta 6: *Suponga que un 30% de los bloques reemplazados al ejecutar el bucle han sido actualizados previamente y por tanto hay que actualizar memoria. Calcule la energía consumida al ejecutar el bucle.*

Ejercicio 1.5

Sea un procesador de 10MIPS que efectua 1.3 referencias a memoria por instrucción. La frecuencia de fallo de cache por referencia es 0.05 y la probabilidad de que el fallo se produzca en una linea que debe actualizarse en memoria principal es 0.5. Cada vez que se accede a memoria se ocupa el bus durante 300 ns que es el tiempo que se tarda en acceder a una linea de cache; durante este tiempo el procesador está parado.

Pregunta 1: *Calcule la frecuencia de fallo por instrucción ejecutada (considerar el efecto del reemplazo como un fallo de cache).*

Pregunta 2: *Calcule el tiempo de procesador transcurrido entre accesos a memoria que producen fallo.*

Pregunta 3: *Calcule el número de transacciones por segundo en el bus.*

Pregunta 4: *Calcule los MIPS efectivos del procesador.*

Pregunta 5: *Calcule la utilización (ocupación) del bus por parte del procesador. Es decir, la probabilidad de que una instrucción utilice el bus.*

Ejercicio 1.6

El tamaño del primer nivel de la cache de datos (L1D) de un procesador es 32Kbytes y la asociatividad es 4, siendo el tamaño de bloque 64 bytes. Respecto a la cache de instrucciones de primer nivel (L1I) supondremos que siempre se acierta. El segundo nivel de la jerarquía es una cache compartida (datos e instrucciones, L2). La penalización en fallo de L1D son 5 ciclos de procesador. Supondremos que siempre se acierta en L2 y que la cache L1D es bloqueante. Al computador con las características descritas lo denominamos A.

El tiempo de interpretación de cualquier instrucción es 1 ciclo a menos que se produzca fallo en cache.

Como programa de prueba utilizaremos el siguiente código. Al iniciarse la ejecución del programa de prueba ninguno de los elementos de los vectores A y B está almacenado en la cache L1D.

```
do K = 1 , M
      S = S + A(I) * B(I)
enddo
```

El tamaño de un elemento de los vectores A y B es 8 bytes.

R1 y R3 almacenan la dirección del primer elemento de los vectores A y B respectivamente. R8 almacena el contenido de la variable S. El valor de M es múltiplo de 8. Los vectores A y B están alineados a tamaño de bloque.

```
1$:  load R2, 0(R1)
     load R4, 0(R3)
     add R1, R1, #8
     add R3, R3, #8
     add R5, R5, # -1
     mul R6, R2, R4
     add R8, R6, R8
     bne R5, 1$
```

Pregunta 1: *Calcule los fallos totales que se producen al ejecutar el bucle, los fallos por instrucción, los accesos a memoria por instrucción y los fallos por acceso a memoria.*

Pregunta 2: *Muestre en un diagrama temporal las dos primeras iteraciones del programa de prueba. Para indicar la ejecución de varias instancias de una misma instrucción utilice la misma fila, indicando la ejecución en los ciclos correspondientes. Calcule los ciclos totales de penalización debido a fallos en L1D al ejecutar el bucle. Así mismo, calcule la fracción de tiempo que representa esta penalización en el tiempo de ejecución del programa.*

El consumo de potencia cuando se acierta en la cache L1D es P vatios y cero cuando el procesador está bloqueado. El consumo de potencia de la cache L2 es un 15% del consumo del procesador.

Pregunta 3: *Calcule la energía consumida al ejecutar el programa de prueba en el computador A.*

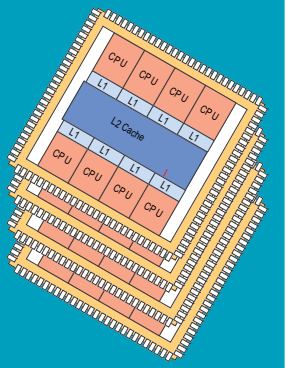
En un nuevo diseño del computador, al que denominamos B, la cache es no bloqueante y la instrucción load tampoco es bloqueante. Así mismo, el procesador soporta varios fallos concurrentes. Además, el segundo nivel de la jerarquía de memoria (L2) tiene dos bancos que pueden servir accesos concurrentes. Los bloques de memoria se almacenan de forma entrelazada en los bancos. Para ello se utiliza el bit menos significativo de los bits de la dirección que identifican el bloque. Esto es, bloques consecutivos en memoria se almacenan en bancos distintos.

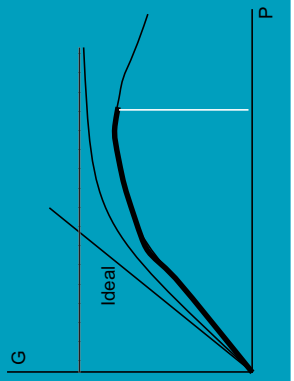
Suponga que el primer bloque de cada vector del programa de prueba se almacena en bancos distintos de L2.

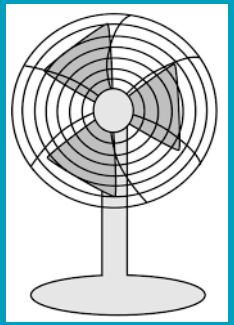
Pregunta 4: *Muestre en un diagrama temporal las dos primeras iteraciones del programa de prueba. Para indicar la ejecución de varias instancias de una misma instrucción utilice la misma fila, indicando la ejecución en los ciclos correspondientes. Calcule el tiempo de ejecución del programa de prueba en ciclos y la ganancia respecto al computador A.*

Pregunta 5: *El consumo de potencia de un banco de la cache L2 es un 15% del consumo del procesador. Muestre en un diagrama temporal la potencia consumida en cada ciclo durante las dos primeras iteraciones del programa de prueba. Calcule la energía*

consumida al ejecutar el programa en el diseño B. ¿Cuál de los computadores A y B tiene la mejor métrica MIPS^2/Potencia media?.







Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 2
Sistema multiprocesador

La necesidad de incrementar el rendimiento de un sistema de cómputo es siempre un horizonte en arquitectura de computadores. El objetivo es reducir el tiempo de ejecución de un programa o incrementar la productividad.

Una técnica ampliamente utilizada para mejorar la productividad es el paralelismo. En este contexto se implementa utilizando varios procesadores que ejecutan de forma autónoma instrucciones, pero con mecanismos que permiten comunicar información entre ellos y sincronizar esa comunicación. Estos sistemas de cómputo se denominan multiprocesadores.

Por otro lado, restricciones tecnológicas favorecen la utilización de multiprocesadores, ya que dedicar más recursos en mejorar el rendimiento de un procesador no representa un incremento sustancial del mismo y empeora otras métricas como el consumo de energía, necesidades de disipación térmica y relaciones entre rendimiento y estas últimas métricas.

Factores tecnológicos favorables por un lado, como la escala de integración y la necesidad de mayor capacidad de procesado por otro lado, teniendo en cuenta las restricciones tecnológicas, han determinado la introducción masiva en el mercado de chips multiprocesador.

Ejemplos de paralelismo son el paralelismo de datos y el paralelismo de función (Figura 2.1). En el primero de ellos varios procesadores efectúan la misma tarea sobre conjuntos de datos disjuntos. Un ejemplo del paralelismo de datos es un bucle cuyas iteraciones son independientes entre si. Cada iteración se puede estar calculando en un procesador. Otro ejemplo son búsquedas independientes en una base de datos. Un ejemplo de paralelismo de función es el procesado de imagen. En el procesado se distinguen varias fases que se encadenan, siendo cada fase una función. Los datos de entrada de una fase son los datos de salida de la fase previa.

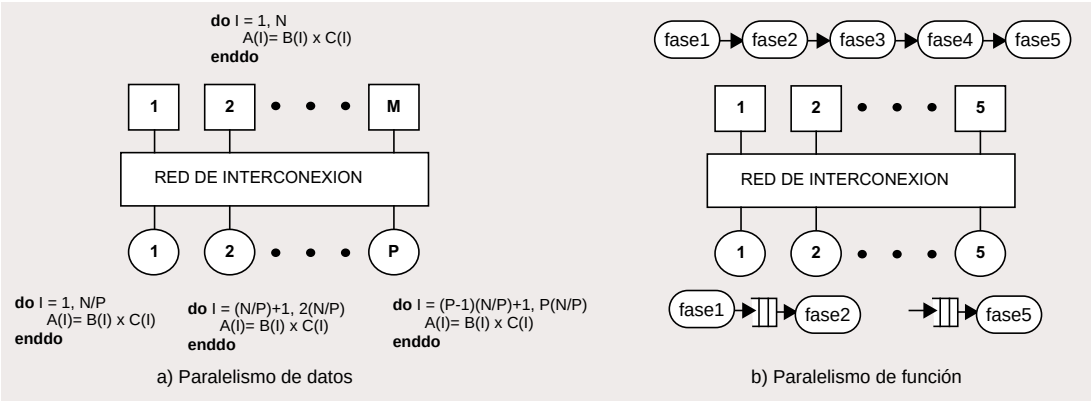


Figura 2.1 Ejemplos de paralelismo.

MODELOS DE PROGRAMACION

La explotación de la capacidad de procesado de un multiprocesador no es transparente al programador, como pueden ser mejoras a nivel de microarquitectura en un procesador. Por modelo de programación entendemos la forma de expresar, en un lenguaje de alto nivel, el paralelismo.

Podemos distinguir dos modelos básicos de programación paralela, que se diferencian en la forma de comunicar información y sincronizar la utilización de la misma entre varias tareas paralelas (Figura 2.2). Esto es, la coordinación del trabajo efectuado en cada tarea.

- Memoria compartida: se utiliza un único espacio de direcciones. La comunicación y sincronización se efectúa mediante instrucciones load y store que ejecutan los hilos[1] para acceder a las posiciones de memoria.
- Paso de mensajes: los espacios de direcciones de cada proceso son disjuntos. La comunicación y sincronización entre los procesos se efectúa mediante el intercambio explícito de mensajes, utilizando un canal de comunicación.

El modelo de memoria compartida facilita partir de un programa serie y paralelizarlo. Todos los hilos tienen acceso directo, mediante instrucciones load y store, a las estructuras de datos. Es necesario utilizar primitivas explí-

1. En el contexto de memoria compartida utilizaremos el término hilo para identificar una tarea de un programa paralelo, ya que el espacio lógico es el mismo. Ahora bien, varios procesos, cada uno con su espacio lógico, pueden acceder al mismo espacio físico utilizando el mecanismo de traducción de direcciones de lógica a física (Figura 2.2). En el contexto de paso de mensajes utilizaremos el término proceso.

citas para sincronizar la comunicación de información entre los hilos. La escalabilidad de estos multiprocesadores está limitada por la tecnología disponible.

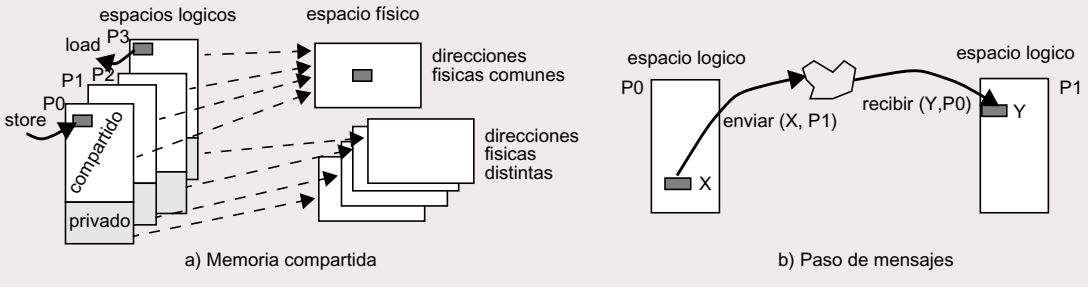


Figura 2.2 Modelos de programación paralela.

El modelo de paso de mensajes requiere particionar explícitamente o distribuir las estructuras de datos del programa entre procesos. Al finalizar hay que recolectar los resultados de cada proceso. Facilita la escalabilidad del multiprocesador, ya que la comunicación se explicita mediante primitivas donde se identifica el emisor y los receptores. La utilización de estas primitivas identifica los puntos de comunicación y sincronización.

En este capítulo y en los restantes nos centraremos en multiprocesadores con memoria compartida.

ORGANIZACIÓN DE MULTIPROCESADORES CON MEMORIA COMPARTIDA

Distinguimos dos organizaciones básicas en función de la latencia de acceso a memoria (Figura 2.3).

- Latencia de acceso uniforme: El tiempo de acceso a memoria de cualquier procesador a cualquier módulo de memoria es el mismo. Los nodos o elementos conectados a la red de interconexión son módulos de memoria o procesadores. La escalabilidad del diseño está limitada por la necesidad de que la latencia de acceso a memoria esté dentro de ciertos rangos. Esta latencia está determinada principalmente por la red de interconexión utilizada.
- Latencia de acceso no uniforme: El tiempo de acceso a memoria desde un procesador a los módulos de memoria es distinto. Los nodos son pares procesador-memoria. La latencia de acceso de un procesador al módulo de memoria con el que está emparejado es mucho menor que la latencia de acceso a otros módulos de memoria. Esta característica facilita la escalabilidad del multiprocesador, aunque requiere una distribución

adecuada de las estructuras de datos, para aprovechar la menor latencia de acceso al módulo de memoria con el que está emparejado un procesador.

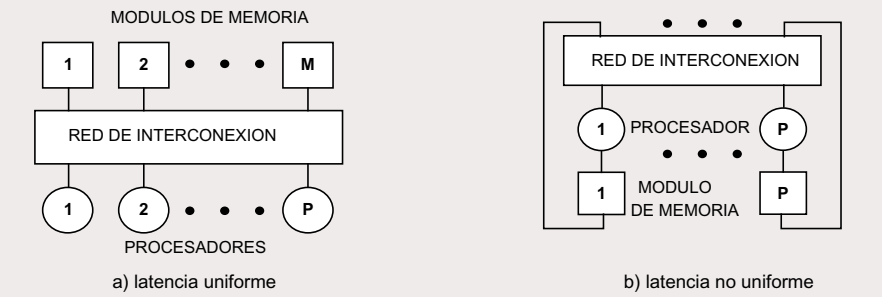


Figura 2.3 Organizaciones del multiprocesador.

En los multiprocesadores se explota la propiedad de localidad de los programas para reducir la latencia de acceso a memoria. La memoria se organiza de forma jerárquica, siendo transparente al programador. Por otro lado, para reducir la demanda de ancho de banda de los procesadores a la red de interconexión, se interponen niveles de la jerarquía entre el procesador y la red de interconexión.

Red de interconexión

La función de la red de interconexión es transportar información entre los nodos conectados a la misma. La red se utiliza siempre que se produce un fallo de cache. Entonces, la latencia de atravesar la red puede ser una fracción significativa de la latencia de acceso a memoria.

Por otro lado, en un multiprocesador la demanda para utilizar la red de interconexión es proporcional al número de procesadores. Entonces, la serialización de los accesos a memoria, que está determinada por el encaminamiento de las peticiones de los procesadores y las repuestas de los módulos de memoria, incrementa la latencia efectiva de acceso a memoria.

Seguidamente se describen algunas de las redes que se utilizan típicamente.

Bus. Es una extensión natural de un sistema uniprocesador. Al elemento que conecta el procesador con memoria se conectan más procesadores. La escalabilidad[2] del bus está limitada por la carga eléctrica que representan los procesadores y la longitud necesaria del bus para conectar los procesadores (Figura 2.4).

2. Número de procesadores y frecuencia de funcionamiento del bus.

Crossbar. Se distinguen puertos de entrada y puertos de salida. En los puertos de entrada se conectan los procesadores y en los puertos de salida los módulos de memoria (Figura 2.4). Puede observarse como p buses que están conectados a los procesadores y m buses que están conectados a los módulos de memoria. Cada uno de los p buses está conectado a cada uno de los m buses mediante un conmutador. En un intervalo de tiempo uno de los m buses sólo puede conectarse a uno de los p buses[3]. Si cada procesador solicita acceso a un módulo de memoria distinto se pueden efectuar p accesos en paralelo (p menor igual que m)[4]. La escalabilidad de la red está limitada por el área que ocupa, la cual también influye en la latencia necesaria para atravesar la red de interconexión.

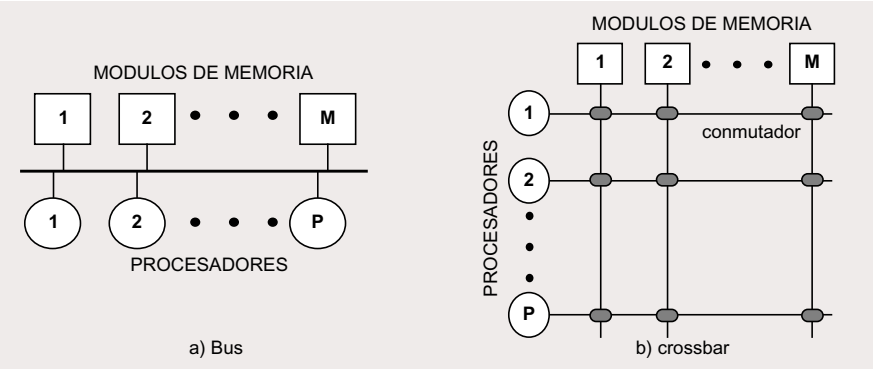


Figura 2.4 Redes de interconexión: a) bus y b) crossbar.

Anillo. La red puede observarse como conexiones punto a punto entre nodos del multiprocesador (Figura 2.5). Los nodos pueden ser módulos de memoria, procesadores o el emparejamiento de ambos. Entre dos nodos adyacentes en la red existe comunicación punto a punto. Sin embargo, para establecer una comunicación entre dos nodos no adyacentes hay encaminar la información pasando por los nodos ubicados entre el nodo fuente y el nodo destino. Entonces, cada nodo debe disponer de un componente que encamine los mensajes cuyo destinatario es otro nodo. En un anillo, pueden producirse tantas comunicaciones en paralelo como conexiones punto a punto hay en el anillo. La escalabilidad del anillo está limitada por la latencia de comunicación entre los dos nodos más alejados entre sí[5].

3. Diremos que es una conexión punto a punto. Una red crossbar puede utilizarse en otros contextos, donde interesa difundir información desde un emisor a varios receptores. Para ello, uno de los p buses se conecta a varios de los m buses. Entonces diremos que es una conexión para difundir información.
4. Un bus puede considerarse un caso degenerado de crossbar (m = 1).
5. Diámetro (número de nodos), el cual es N/2 para N nodos.

Malla. Para reducir la latencia de comunicación debido a la lejanía entre nodos, la red de interconexión se organiza en dos dimensiones. Los nodos están en los cruces de los enlaces de comunicación. Un nodo, no periférico, puede estar comunicándose con los cuatro nodos a los que está conectado[6].

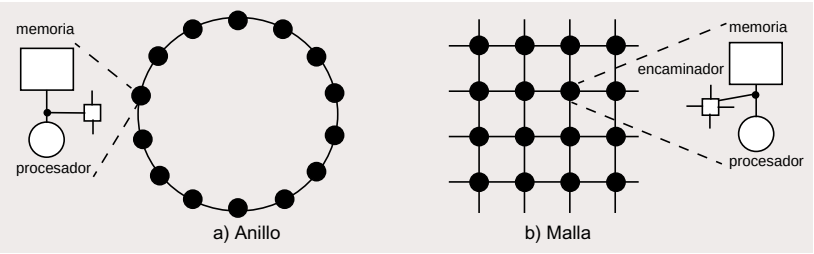


Figura 2.5 Redes de interconexión: a) anillo y b) malla.

OPERACIONES DE SINCRONIZACIÓN

Por comunicación entendemos la transmisión de información entre hilos, la cual, en un multiprocesador de memoria compartida, se efectúa mediante posiciones de almacenamiento en memoria y la ejecución de instrucciones load y store. Las sincronizaciones se efectúan utilizando posiciones de almacenamiento en memoria. Entonces, una sincronización puede observarse como una forma especial de comunicación en la que los datos son información de control.

En un multiprocesador la sincronización entre hilos se utiliza para satisfacer, en tiempo de ejecución, las restricciones impuestas por las dependencias entre hilos. Esto es, un objetivo de una operación de sincronización es proveer un mecanismo para que un hilo acceda a una variable después de conocer que ha sido actualizada (dependencia verdadera) o viceversa, esperar a que una variable haya sido leída para actualizarla (antidependencia). Otro objetivo de una operación de sincronización es garantizar que sólo un hilo, entre varios, está actualizando un conjunto de variables en un instante determinado. El primer caso se denomina sincronización mediante eventos y el segundo acceso excluyente.

Finalmente se describe la operación de sincronización barrera que fuerza las dependencias entre fases de cálculo de grupos de operaciones. Todos los hilos ejecutan la operación barrera y esperan hasta que el último hilo ejecuta la operación.

6. El diámetro de una malla cuadrada es 2 x (N -1), siendo N^2 el número total de nodos.

Soporte de la arquitectura

La característica clave que debe tener una instrucción, para ser utilizada en operaciones de sincronización, es que el acceso a memoria sea una operación atómica.

A nivel de lenguaje máquina, la unidad de ejecución indivisible para acceder a memoria son las instrucciones load y store. Si dos instrucciones acceden concurrentemente a la misma posición de memoria el hardware serializa el acceso.

Las dos operaciones de sincronización descritas, evento y acceso exclusivo, se pueden implementar mediante instrucciones load y store. Seguidamente se presentan ejemplos para mostrar la necesidad de sincronización y el mecanismo para realizarla.

En la Figura 2.6 se muestra una sincronización mediante evento y en la Figura 2.10 se muestra el acceso exclusivo a un conjunto de variables utilizando instrucciones load y store.

Sincronización mediante evento

Los hilos que se muestran en la parte izquierda de la Figura 2.6 se comunican un valor utilizando la variable A. En concreto, el hilo H1 escribe un valor en la variable A y el hilo H2 lee el valor.

Hilo H1	Hilo H2	Comentarios
. . .	. . .	R1 = R6 = dirección de la variable A
store R3, 0(R1)	load R4, 0(R6)	
. . .	. . .	

Hilo H1	Hilo H2	Inicialización
A = 3.14	While (aviso = 0) { };	aviso = 0
aviso = 1	T = A	
. . .	. . .	

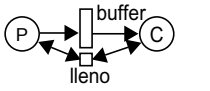
Figura 2.6 Sincronización mediante evento.

Para garantizar que el hilo H2 lee la variable A después de que el hilo H1 la haya actualizado hay que incluir una operación de sincronización, denominada por evento. Para ello se utiliza otra variable, denominada aviso, que el hilo H1 actualiza después de escribir en la variable cuyo valor quiere comunicar. El hilo H2 consulta el valor de la variable aviso. Mientras el valor de variable aviso no tiene un valor predeterminado no prosigue la ejecución (parte derecha de la Figura 2.6).

Ejercicio

Suponga el siguiente código que se utiliza para comunicar información entre dos hilos. Uno de los hilos se denomina productor y el otro hilo se denomina consumidor.

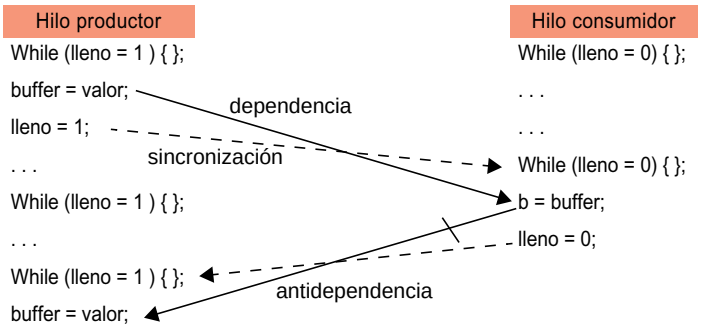
Hilo productor	Hilo consumidor
While (lleno = 1) { };	While (lleno = 0) { };
buffer = valor;	b = buffer;
lleno = 1;	lleno = 0;



Indique las operaciones de sincronización que se utilizan y el tipo de dependencia de datos entre hilos que se quieren garantizar en tiempo de ejecución.

Respuesta

La operación de sincronización es de tipo evento. Se utiliza únicamente una variable para efectuar dos operaciones de sincronización y la variable es lleno. En el sentido productor-consumidor garantiza que el hilo consumidor no lee el contenido de la variable buffer hasta que el hilo productor ha escrito un valor. Esto es, una dependencia de datos verdadera. En el sentido consumidor-productor garantiza que el hilo productor no actualice el valor de la variable buffer antes de que el hilo productor la haya leído. Esto es, una antidependencia.



Comunicación entre dos hilos

Una forma de paralelismo es el paralelismo a nivel de tarea o paralelismo funcional, en el cual funciones independientes se ejecutan en procesadores distintos. Por ejemplo, se pueden encadenar varias funciones de forma segmentada y el objetivo es procesar un flujo de datos.

Para conectar un hilo productor y un hilo consumidor usualmente se utiliza un buffer con varias entradas. Ello permite que el hilo productor pueda producir ráfagas de información más rápidamente que la rapidez de consumo del hilo consumidor, aunque en media la rapidez de producción sea menor igual que la rapidez de consumo.

El buffer se utiliza para acomodar las ráfagas de producción con la rapidez media del hilo consumidor. El hilo productor almacena la información en el buffer mientras el hilo consumidor procesa la información previa. En la Figura 2.7 se muestra un diseño de los dos hilos con un buffer que se gestiona de forma circular. El hilo productor almacena la información en entradas consecutivas del buffer, utilizando la variable cola que indica la primera entrada libre. De forma similar, el hilo consumidor extrae información de entradas consecutivas del buffer, utilizando la variable cabeza que indica la primera entrada con información.

Hilo productor	Hilo consumidor	Inicialización
while (cola = (cabeza + N -1) mod N) { };	While (cabeza = cola) { };	cabeza = cola = 0
buffer(cola) = item;	item = buffer(cabeza);	
cola = (cola + 1) mod N;	cabeza = (cabeza + 1) mod N;	
	PROCESADO (item);	

Figura 2.7 Buffer circular que utilizan un hilo productor y un hilo consumidor.

Para determinar la condición de buffer lleno o vacío se utilizan las variables cola y cabeza[7]. Observemos que estas variables sólo son actualizadas por un hilo. Por tanto, no es necesario establecer una serialización al acceder a ellas.

Acceso exclusivo

En el ejemplo del buffer circular, cuando la rapidez con que el hilo productor produce datos es mayor que la rapidez con la cual las consume el hilo consumidor, una alternativa es utilizar varios hilos consumidores, hasta equiparar la rapidez de producción con la de consumo[8].

Ahora bien, un entrelazado de las instrucciones de dos hilos consumidores (Figura 2.7), como el mostrado en la Figura 2.8, puede producir resultados no esperados. Por ejemplo, el entrelazado muestra que los dos hilos leen la misma entrada del buffer.

tiempo	Hilo consumidor 1	Hilo consumidor 2
1	while (cabeza = cola) { };	While (cabeza = cola) { };
2	item = buffer(cabeza);	
3		item = buffer(cabeza);

Figura 2.8 Entrelazado de sentencias de dos hilos consumidores que se ejecutan concurrentemente.

7. Observe que el buffer como máximo puede contener N-1 elementos sin extraer antes de que el consumidor se bloquee.
8. Paralelismo de datos accediendo a una única estructura de datos para obtener la información.

Para que el entrelazado sea correcto hay que garantizar que el acceso a algunas variables compartidas está serializado. Esto es, hasta que un hilo no ha efectuado todas las operaciones de lectura y escritura, ningún otro hilo consumidor puede acceder a las variables compartidas. Esta acción de sincronización se denomina garantizar acceso exclusivo. Notemos que un hilo consumidor necesita incrementar el valor de la variable cabeza, para indicar que puede leerse información de la siguiente entrada. Por tanto, sólo un hilo consumidor puede acceder al buffer. En la Figura 2.9 se muestra un ejemplo de código donde se indican las instrucciones que acceden de forma exclusiva a las variables compartidas por los hilos productores. Notemos que una vez obtenido el acceso exclusivo se vuelve a comprobar si existe alguna entrada que debe procesarse.

Hilo productor	Hilo consumidor	Inicialización
while (cola = (cabeza + N -1) mod N) { };	repeat	cabeza = cola = 0
buffer(cola) = item;	While (cabeza = cola) { };	
cola = (cola + 1) mod N;	volver = 0	
	Adquirir acceso exclusivo	
	if (not (cabeza = cola)) then	
	item = buffer(cabeza);	
	cabeza = (cabeza + 1) mod N;	
	else	
	volver = 1	
	endif	
	Liberar acceso exclusivo	
	until volver = 0	
	PROCESADO (item);	

Figura 2.9 Buffer circular que utilizan un hilo productor y varios hilos consumidores.

Implementación del acceso exclusivo mediante instrucciones load y store

En este apartado se muestra una forma de garantizar acceso exclusivo mediante instrucciones load y store. Notemos que se utilizan dos sincronizaciones tipo evento (aviso1 y aviso2) y una variable turno que determina el hilo que accede a la zona de exclusión cuando intentan entrar los dos hilos concurrentemente. En este caso, obtiene el acceso exclusivo el hilo que no ejecuta la sentencia “turno = “ el último.

Comentarios	Hilo H1	Hilo H2	Inicialización
	aviso1 = 1;	aviso2 = 1;	aviso1 = aviso2 = 0
adquirir el acceso	turno = 1;	turno = 2;	
	while ((aviso2 = 1) and (turno = 1)) {};	while ((aviso1 = 1) and (turno = 2)) {};	
	acceso exclusivo	acceso exclusivo	
liberar el acceso	aviso1 = 0;	aviso2 = 0;	

Figura 2.10 Obtención de acceso exclusivo, algoritmo de Peterson, utilizando instrucciones load y store.

La implementación de acceso excluyente mediante instrucciones load y store tiene deficiencias y no escala cuando se incrementa el número de procesadores. Por ello el lenguaje máquina de los procesadores se extiende con instrucciones atómicas que realizan las siguientes operaciones de forma indivisible o atómica: a) lectura, b) modificación y c) escritura. Estas instrucciones facilitan la implementación de acceso exclusivo a variables compartidas y las denominaremos de forma genérica instrucciones LME.

Instrucciones atómicas de lectura-modificación-escritura

Una característica clave de la ejecución de una instrucción LME, que accede a una posición de memoria, es que la secuencia de lectura, modificación y escritura es atómica. Esto es, además de que no se gestionan cambios del flujo de ejecución, forzados[9] o no, durante la ejecución de una instrucción LME el hardware serializa el acceso de otros procesadores a la posición de memoria que referencia.

En la Figura 2.11 se muestra la especificación de varias instrucciones atómicas. Una operación típica es un intercambio atómico. Una instrucción que implementa esta operación intercambia el contenido de un registro con el contenido de una posición de memoria. Cuando dos procesadores en un multiprocesador ejecutan concurrentemente una instrucción intercambio, estas serán ordenadas por el mecanismo de serialización de escrituras.

intercambio (llave, R)	Test&set (llave)	fetch&inc (dir)	Comentario
tmp = llave	tmp = llave	tmp = dir	Secuencia de instrucciones que se ejecutan de forma atómica.
llave = R	llave = 1	dir = tmp + 1	
return R = tmp	return tmp	return tmp	

Figura 2.11 Ejemplos de instrucciones atómicas.

9. Por ejemplo, una interrupción.

Otras operaciones atómicas son: a) test&set y b) fetch&inc. En una implementación de la instrucción test&set se lee el valor de una posición de memoria y posteriormente se escribe un uno. En una instrucción fetch&inc se lee el valor de una posición de memoria y atómicamente se almacena el valor leído incrementado en una unidad.

En la Figura 2.12 se muestra la utilización de la instrucción atómica intercambio para implementar la primitiva adquirir, que junto con la primitiva liberar permite garantizar el acceso exclusivo a un conjunto de datos compartidos de una secuencia de instrucciones.

adquirir (llave)	liberar (llave)	acceso exclusivo	Inicialización
R = 1	llave = 0	. . .	llave = 0
repeat	return	adquirir (llave)	
intercambio (llave, R)		acceso exclusivo	
until R = 0		liberar (llave)	
return		. . .	

Figura 2.12 Primitivas adquirir y liberar y su utilización para garantizar acceso exclusivo.

En la Figura 2.13 se muestra la utilización de la instrucción fetch&inc para obtener el número de iteración que debe ejecutar un hilo, perteneciente al conjunto de hilos que ejecutan un bucle paralelo.

Bucle paralelo	Autoplanificación	Comentarios
doall I = 1, N	LI = fetch&inc (GI)	GI: variable compartida
. . .	while (LI =< N)	LI: variable local. Número de iteración
endoall	. . .	
	LI = fetch&inc (GI)	Inicialmente GI = 1
	endwhile	

Figura 2.13 Obtención del número de iteración por parte de los hilos que calculan un bucle paralelo. Utilización de la instrucción atómica fetch&inc para autoplanificación.

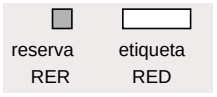
Instrucciones load con enlace y almacenamiento condicional

La implementación de una instrucción de lectura-modificación-escritura de forma atómica no es sencilla. Por ello se han añadido en algunos lenguaje maquina dos instrucciones que implementan individualmente las operaciones de lectura y actualización, pero conjuntamente implementan una operación LME. Para ello se incrementa la semántica las instrucciones load y store conocidas[10]. Estas instrucciones se denominan load con enlace (LL) y almacenamiento condicional (SC)[11].

Para garantizar que la ejecución de la secuencia de instrucciones LL y SC es atómica el hardware dispone de recursos para: a) identificar la secuencia de instrucciones LL y SC y b) observar las escrituras de otros procesadores a la posición de memoria a la que acceden. En estas condiciones, la operación de actualización no se efectúa si, en el lapso de tiempo transcurrido entre la ejecución de la instrucción LL y la instrucción SC, se ha detectado una escritura a la posición de memoria.

Para la implementación de las instrucciones LL y SC se utiliza un registro denominado registro de enlace (RE). El registro contiene la dirección de la posición de memoria que lee la instrucción LL. El contenido del registro RE se invalida cuando se observa una escritura, en el lapso de tiempo transcurrido entre la ejecución de la instrucción LL y la ejecución de la instrucción SC[12]. Al interpretar la instrucción SC se analiza si el registro RE es inválido. Si es el caso, la instrucción SC se convierte en una instrucción nop (se anula). En caso contrario se actualiza la posición de memoria.

En la Figura 2.14 se especifican las instrucciones load con enlace y almacenamiento condicional.

load con enlace: LL R_d, X(R_f)	Descripción	Registro de enlace (RE)
$RED^v = X + R_f^v$ $RER^v = 1$ $R_d^v = M [X + R_f^v]$ el superíndice v indica valor	Almacena en R_d el contenido de la posición de memoria cuya dirección se calcula como $X + R_f^v$ y efectúa la acción de reserva almacenando la dirección $X + R_f^v$ en un registro denominado registro de enlace (RE).	

almacenamiento condicional: SC R_{fd}, X(R_f)	

if ($RED^v = X + R_f^v$ and $RER^v = 1$) then
cancelar la reserva de otros procesadores de la dirección $X + R_f^v$
$M [X + R_f^v] = R_{fd}^v$
$R_{fd}^v = 1$
else
$R_{fd}^v = 0$
	Comprueba si la reserva de la dirección de memoria en el registro RE es válida. Si es el caso, el contenido del registro R_{fd} se almacena en la posición de memoria cuya dirección se calcula como $X + R_f^v$ y en el registro R_{fd} se almacena el valor uno. En otro caso, la instrucción se anula y en el registro R_{fd} se almacena un cero

Figura 2.14 Especificación de las instrucciones load con enlace y almacenamiento condicional.

10. En una instrucción atómica la operación de lectura no es observable por otros procesadores y como no modifica el estado de memoria no es necesario excluirla en el lapso de tiempo requerido para ejecutar una operación atómica.
11. En inglés, usualmente, load linked y store conditional.
12. También debe invalidarse, por ejemplo, en operaciones de cambio de contexto del procesador. En los capítulos de coherencia de cache se entra en mayor detalle.

En la Figura 2.15 se muestra la utilización de las instrucciones LL y SC para implementar la operación atómica intercambio. Después de ejecutar las instrucciones LL y SC, para efectuar el intercambio, se comprueba si se ha ejecutado la instrucción SC de forma efectiva. Si este no es el caso, se vuelve a intentar, ya que la ejecución atómica de un intercambio no se ha realizado.

intercambio (llave, R)		Intercambio (0(R1), R4)	comentario
tmp = llave	1$:	mov R3, R4	mover valor de intercambio
llave =R		LL R2, 0(R1)	Load Linked
return R = tmp		SC R3, 0(R1)	Store condicional
		beq R3, 1$	repetir si se anula
		mov R4, R2	R1: dirección de la variable llave
		return	

Figura 2.15 Operación intercambio implementada mediante instrucciones LL y SC.

Observemos que entre las instrucciones LL y SC se pueden ejecutar instrucciones que utilizan como operandos valores almacenados en el banco de registros y que almacenan el resultado en un registro del banco de registros[13]. Una recomendación usual es que entre una instrucción LL y una instrucción SC no se efectúen accesos a memoria. También, no es usual soportar el entrelazado o imbricado del secuencia de instrucciones LL y SC.

Consideraciones en una operación de sincronización

Seguidamente se efectúan algunas consideraciones sobre una operación de sincronización. En una sincronización distinguimos tres fases:

- Fase de adquisición u obtención. Se establece un valor en una variable compartida para indicar la adquisición de una sincronización.
- Fase de espera. Nos centraremos en lo que se denomina espera activa. Esto es, la espera se efectúa mediante la ejecución de instrucciones cuyo fin es esperar la adquisición de la sincronización.
- Fase de liberación. Se establece un valor en una variable compartida para indicar la liberación de una sincronización.

Cada una de las fases anteriores presenta características distintas en función de la operación de sincronización. Sin embargo, de forma genérica interesa tener en cuenta las siguientes características:

- Fase de adquisición: la latencia debe ser reducida cuando no existen otros hilos que efectúan la misma fase.

13. Para que no se incremente la probabilidad de que otro procesador escriba en la posición de memoria accedida, el número de instrucciones entre una instrucción LL y una instrucción SC tiene que ser reducido.

- Fase de espera: el tráfico que se inyecte en la red de interconexión debe ser bajo.
- Necesidades de almacenamiento: el número de posiciones de memoria utilizadas debe ser reducido.
- Escalabilidad: tanto la latencia como el tráfico y el almacenamiento necesario en las fases de adquisición y espera debe incrementarse linealmente con el número de hilos.
- Equitativo: hay que garantizar que no se produce inanición. Esto es, un hilo no se queda relegado de efectuar una adquisición de forma indefinida.

Ejercicio

En la operación de sincronización por evento, mostrada en la Figura 2.6, indique las fases de adquisición y espera y las necesidades de almacenamiento. En una operación de acceso exclusivo, como la mostrada en la Figura 2.12, indique las fases de adquisición y espera y las necesidades de almacenamiento.

Respuesta

En la operación de sincronización que se muestra seguidamente, la fase de adquisición es la ejecución de la sentencia aviso = 1, la fase de espera es el bucle while y la necesidad de almacenamiento es una posición de memoria.

Hilo H1	Hilo H2	Inicialización
A = 3.14	While (aviso = 0) { };	aviso = 0
aviso = 1	T = A	
. . .	. . .	

En una operación de sincronización, para garantizar acceso exclusivo, la operación intercambio en la primitiva adquirir y la actualización de la variable llave en la primitiva liberar son respectivamente las fases de adquisición y liberación. La necesidad de almacenamiento es una posición de memoria (variable llave). La fase de espera es el bucle repeat.

```
adquirir (llave)          liberar (llave)      acceso exclusivo
R = 1                     llave = 0            llave = 0
repeat                    return               . . .
   intercambio (llave,R)                       adquirir (llave)
until R = 0                                       acceso exclusivo
return                                         liberar (llave)
```

Operacion barrera

Una barrera es una operación de sincronización que fuerza las dependencias entre fases de cálculo de grupos de operaciones. Todos los hilos ejecutan la operación barrera y esperan hasta que el último hilo ejecuta la operación. Podemos decir que una operación barrera es un sincronización por evento.

En la Figura 2.16 se muestra un ejemplo de utilización de la operación BARRERA. Los cálculos efectuados en el primer bucle se utilizan en el segundo bucle. Esto es, hay dependencias de datos. La planificación de iteraciones entre los procesadores no garantiza que un procesador ejecute las mismas iteraciones en los dos bucles. Entonces, antes de empezar a ejecutar el segundo bucle hay que esperar a que finalicen todas las iteraciones del primer bucle[14].

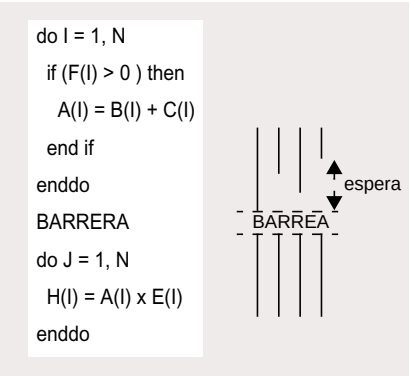


Figura 2.16 Ejemplo de utilización de la operación de sincronización BARRERA.

En la implementación de la operación BARRERA se utiliza una llave y un evento. La llave se utiliza para garantizar el acceso exclusivo a un contador, que contabiliza los hilos que han ejecutado la operación barrera. El evento se utiliza para que los hilos esperen hasta que el contador toma el valor cero.

En la parte izquierda de la Figura 2.17 se muestra un esquema del código de una operación barrera. Para contabilizar el número de hilos que han ejecutado la operación se utiliza un contador. En la parte derecha se muestra una implementación, utilizando las primitivas adquirir y liberar, para obtener acceso exclusivo al contador. Para efectuar el bloqueo se utiliza la variable liberar, mediante la cual se implementa una sincronización tipo evento.

14. Podrían utilizarse sincronizaciones por evento individualizadas por dependencia. Ahora bien, el número de eventos necesarios es proporcional al número de iteraciones.

BARRERA
b = b+1
if b < N **then**
bloquear hilo en la cola
else
desbloquear todos los hilos de la cola
b := 0
endif
return

	BARRERA	comentario
1$:	adquirir (llave)	acceso exclusivo
	if (contador = 0) liberar =0	inicializar evento de finalización
	contador = contador +1	
	liberar (llave)	
	if (contador = N) then	evento de finalización
	contador =0	
	liberar =1	activar evento de finalización
	else	
	repeat	esperar a la activación
	until (liberar = 1)	del evento de finalización
	endif	
	return	

Figura 2.17 Operación de sincronización B RRER .N es el número de procesadores.

Si el valor de la variable contador es cero se desactiva el evento liberar, ya que se inicia la ejecución de la operación atómica barrera. Previamente se ha adquirido el acceso exclusivo a la variable contador. Después de incrementar el contenido de la variable contador se libera el acceso exclusivo. Si el valor de la variable contador es menor que N los hilos se esperan, utilizando espera activa, en el bucle repeat. En caso contrario se establece el valor cero en la variable contador y se habilita la liberación[15].

La operación barrera descrita tiene contención, tanto en la fase de contabilizar el número de procesadores que han llegado a ella como en la fase de dejar la barrera. En la fase de contar el número de procesadores, la contención es importante ya que hay que obtener acceso exclusivo a la variable contador. La fase de dejar la barrera solo tiene contención de lecturas. Una alternativa es utilizar una estructura de datos en árbol para implementar la operación barrera.

En la Figura 2.18 se muestra un esquema. Las hojas son los hilos y la llegada de los hilos a la barrera se contabiliza de forma distribuida; en sentido ascendente. En la Figura 2.18 cada nodo del primer nivel del árbol contabiliza a 2 hilos. En el siguiente nivel, se contabilizan 2 grupos de 2 hilos. En el último nivel, que es la raíz, se contabilizan 4 grupos de 2 hilos. Cuando el último de

15. Esta implementación de la operación barrera (parte derecha de la Figura 2.17) no funciona correctamente si se utiliza varias veces en un código. Por ejemplo, entre varios bucles paralelos consecutivos con dependencias. En un ejercicio se propone su modificación.

los 2 hilos llega a un nodo se lo notifica al nodo del siguiente nivel. Posteriormente la liberación también puede efectuarse de forma distribuida siguiendo el árbol en sentido inverso.

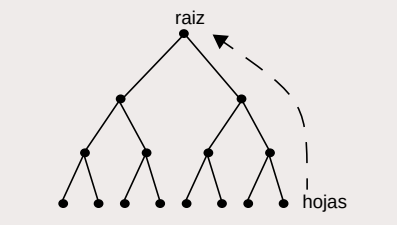


Figura 2.18 Barrera con estructura en árbol para reducir la contención.

LEY DE AMDAHL

La ley de Amdahl permite calcular la ganancia que se obtiene al ejecutar un programa partiendo de: a) la fracción de tiempo en la cual puede explotarse una mejora y b) la ganancia que se obtiene cuando se utiliza la mejora.

En la Figura 2.19 se muestra un diagrama donde, en el tiempo de ejecución de una programa en el procesador original, se distingue la proporción de tiempo en la cual se pueden utilizar los P procesadores. Para simplificar el dibujo, los instante de tiempos en los cuales se pueden utilizan los P procesadores se han agrupado de forma contigua (t_2). La ganancia depende de dos factores: a) la fracción de tiempo en el procesador original donde pueden utilizarse los P procesadores ($f_m = t_2/T_o$)[16] y b) la ganancia cuando se utilizan los P procesadores el 100% ($P = t_2/t_3$).

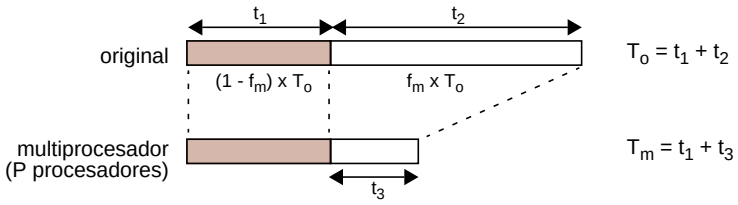


Figura 2.19 Relación entre el tiempo de ejecución en el procesador original y el tiempo de ejecución en el multiprocesador.

Teniendo en cuenta los tiempos de ejecución, la ganancia se calcula como

$$G = \frac{T_o}{T_m} = \frac{T_o}{t_1 + t_3}$$

16. A esta fracción de tiempo también se la denomina, en ocasiones, fracción de código que se paraleliza.

Como $f_m = t_2/T_o$ y $P = t_2/t_3$, efectuando manipulaciones algebraicas obtenemos.

$$G = \frac{1}{(1 - f_m) + \frac{f_m}{P}}$$

Cuando el número de procesadores tiende a infinito obtenemos la siguiente expresión, que indica la máxima ganancia que se puede obtener.

$$G|_{P \to \infty} = \frac{1}{1 - f_m}$$

Revisión I

La ley de Amdahl supone un tamaño fijo del problema que se resuelve o unas funcionalidades que no se modifican. Sin embargo, cuando se dispone de más capacidad de procesado, es típico añadir más funcionalidades o mejorar la precisión del cálculo, lo cual incrementa el tamaño del problema[17].

En este contexto la carga de trabajo se incrementa pero se espera que, al disponer de más procesadores, el tiempo de ejecución sea el mismo que antes o se reduzca (Figura 2.20). Adicionalmente se espera que la carga de trabajo adicional se pueda ejecutar en paralelo.

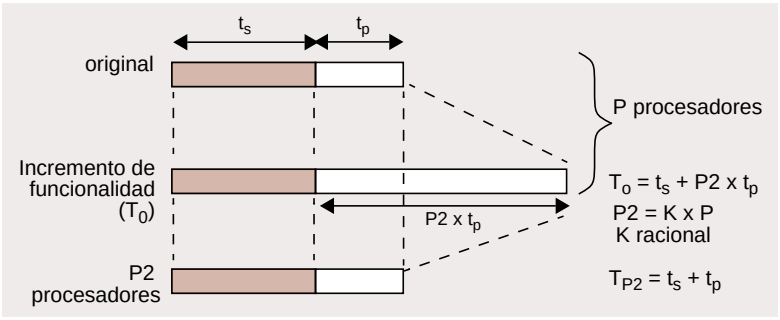


Figura 2.20 Ganancia teniendo en cuenta un incremento de las funcionalidades de la aplicación.

En el contexto descrito tenemos que

$$G = \frac{T_o}{T_{P2}} = \frac{t_s + t_p \times P_2}{t_s + t_p} = (1 - \beta) + \beta \times P_2 \qquad \text{siendo} \qquad \beta = \frac{t_p}{t_s + t_p}$$

Entonces, si la funcionalidad añadida[18] también utiliza los procesadores adicionales ($P_2 > P$), se obtiene una ganancia que es lineal con la parte que utiliza el incremento de capacidad de procesado.

17. Este hecho no es exclusivo del contexto multiprocesador. Se ha estado produciendo al incrementar las prestaciones de un sistema uniprocesador.

Desde el punto de vista del usuario se dispone de mayor funcionalidad y el tiempo de respuesta es el mismo.

Revisión II

En una ejecución paralela la inicialización de los hilos es una tarea serie. Por otro lado, las operaciones de sincronización no son necesarias en una ejecución serie del programa. Estos tiempos representan una penalización al ejecutar un programa paralelo, respecto su ejecución serie, y pueden reducir la ganancia esperada al ejecutar el programa en paralelo.

Para que el efecto de la penalización de una sincronización sea reducido, una operación de sincronización debe permitir realizar un número significativo de cálculos. Si además esta sincronización determina mucha comunicación de información, puede ser otro factor que reduzca la ganancia. Por ejemplo, no es lo mismo efectuar 10 cálculos con 2 procesadores, antes de efectuar una sincronización, que utilizar 20 procesadores, donde cada uno de ellos realiza un cálculo por operación de sincronización.

En la Figura 2.21 se muestra una representación del tiempo de ejecución de un programa en un procesador y el tiempo de ejecución en un multiprocesador, donde se explicita el tiempo de sincronización.

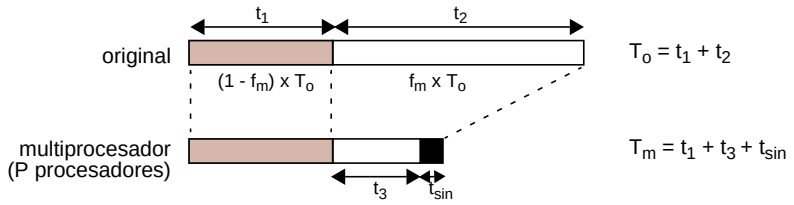


Figura 2.21 Relaciones de tiempos de ejecución teniendo en cuenta la sincronización.

La expresión de la ganancia cuando se tiene en cuenta la penalización debida a la comunicación y sincronización es

$$G = \frac{1}{(1 - f_m) + \frac{f_m}{P} + \frac{t_{sin}}{T_0}}$$

18. El tiempo requerido por el incremento de funcionalidades es $(P_2 - 1) \times t_p$.

En la Figura 2.22 se muestra una gráfica con valores representativos de f_m y t_{sin}. Observemos que cuando t_{sin} es distinto de cero, un incremento del número de procesadores puede reducir la ganancia.

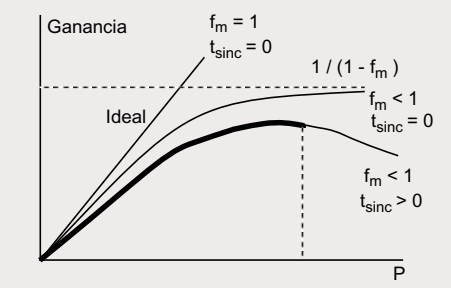
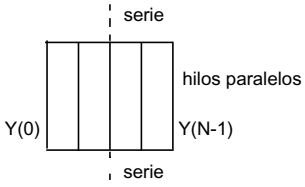


Figura 2.22 Ganancia teniendo en cuenta el tiempo de comunicación y sincronización. El valor $1 / (1 - f_m)$ se calcula suponiendo un número ilimitado de procesadores, siendo $f_m < 1$ y $t_{sin} = 0$.

Como ejemplo en la Figura 2.23 se muestra la ejecución del producto matriz por vector. En la parte izquierda, cada procesador calcula el producto de una fila de la matriz por el vector, para calcular un elemento del vector resultado. Se pueden estar calculando todos los elementos del vector resultado en paralelo. En la parte derecha se utilizan todos los procesadores para calcular un elemento del vector. Se produce una serialización debido a la operación de reducción que hay que realizar[19]. La operación de reducción se efectúa en paralelo. Para ello los cálculos, en concreto las sumas, se efectúan en un orden distinto al especificado por el programador.

Cálculo de un elemento del vector Y en cada procesador

```
doall i = 0, N-1
  do j = 0, N-1
    Y(i) = Y(i) + A(i, j) * X (j)
```



Cálculo de un elemento del vector Y utilizando todos los procesadores

```
do i = 0, N-1
  S = Y(i)
  doall j = 0, N-1
    S = S + A(i, j) * X (j)
  Y(i) = S;
```

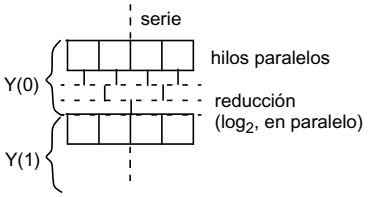


Figura 2.23 Cálculo del producto matriz por vector en paralelo. El bucle en negrita es el que se ejecuta en paralelo. Los dibujos no están en la misma escala.

19. Una operación de reducción requiere que los procesadores accedan (lectura y escritura) a variables compartidas. Por tanto, hay que garantizar acceso exclusivo.

Otro ejemplo es la operación de sincronización barrera que hay que utilizar entre dos bucles paralelos, cuando el primero calcula datos que se utilizan en el segundo y no se utiliza una planificación específica para asignar procesadores a hilos, si ello es posible.

Eficiencia

Se entiende por eficiencia la capacidad de utilizar los recursos disponibles. En este contexto, los recursos son los procesadores. Entonces, la eficiencia se calcula mediante la siguiente expresión.

$$E = \frac{G}{P}$$

EJERCICIOS

Ejercicio 2.1

Considere la ejecución paralela de un programa, con 200.000 instrucciones, en un multiprocesador de memoria compartida con 4 procesadores que funcionan a una frecuencia de 40 Mhz. El programa consiste de 4 tipos de instrucciones. La frecuencia de los tipos de instrucciones y el número de ciclos (CPI) necesarios para cada tipo de instrucción se han obtenido de una traza experimental del programa.

tipo de instrucción	CPI	frecuencia
aritméticas y lógicas	1	60 %
load/store con acierto en cache	2	18 %
secuenciamiento explícito	4	12 %
referencias a memoria con fallo de cache	8	10%

Pregunta 1: *Calcule el CPI medio cuando el programa se ejecuta en un solo procesador.*

Pregunta 2: *Calcule los MIPS cuando el programa se ejecuta en un solo procesador.*

Para la ejecución con 4 procesadores el programa se divide en 4 partes. Suponga que la frecuencia de instrucciones en cada parte es la misma que en el programa original. El CPI para las instrucciones que referencian a memoria y producen un fallo de cache se ha incrementado de 8 a 12 ciclos, debido a colisiones en la red de interconexión o en los módulos de memoria. El CPI para el resto de instrucciones no se modifica.

El programa se divide en cuatro partes de 30.000, 40.000, 60.000 y 70.000 instrucciones cada una.

Pregunta 3: *Calcule el CPI cuando el programa se ejecuta en el multiprocesador con 4 procesadores.*

Pregunta 4: *Calcule la ganancia cuando el programa se ejecuta en un multiprocesador con 4 procesadores respecto a una ejecución serie.*

Pregunta 5: *Calcule la eficiencia del multiprocesador con 4 procesadores.*

Pregunta 6: *¿Por qué no se obtiene máxima eficiencia?*

Para balancear la carga de trabajo en los 4 procesadores, el programa se divide en cuatro parte iguales (50.000 instrucciones en cada parte). Debido a las necesidades de sincronización entre las cuatro partes, en el tiempo de

ejecución debe considerarse el equivalente a la ejecución de 5000 instrucciones extras en cada parte individualmente (son instrucciones que no pertenecen al cálculo original, tiempos de espera, etc.).

Pregunta 7: *Calcule el CPI cuando el programa se ejecuta en un multiprocesador con 4 procesadores.*

Pregunta 8: *Calcule los MIPS cuando el programa se ejecuta en un multiprocesador con 4 procesadores.*

Pregunta 9: *Calcule la ganancia al ejecutar el programa en un multiprocesador con 4 procesadores respecto de una ejecución en un sólo procesador.*

Pregunta 10: *Calcule la eficiencia del multiprocesador con 4 procesadores.*

Pregunta 11: *¿Por qué no se obtiene máxima eficiencia?.*

Ejercicio 2.2

Se define ganancia de rendimiento como

$$G(P) = \frac{T_1(n)}{T_P(n)}$$

siendo $T_1(n)$ el tiempo de ejecución del programa en 1 procesador y $T_P(n)$ el tiempo de ejecución con P procesadores. Se define eficiencia como

$$E(P) = \frac{G(P)}{P} = \frac{T_1(n)}{T_P(n) \times P}$$

lo cual es una indicación del grado actual de ganancia que se obtiene comparado con el máximo valor de la ganancia.

Pregunta 1: *Determine el valor máximo de la ganancia.*

Pregunta 2: *Determine los valores extremos de la eficiencia.*

Pregunta 3: *¿Cómo debe variar la ganancia en función de P para que la eficiencia sea constante?.*

Ejercicio 2.3

El tiempo de ejecución de un programa en un procesador es

$$T_1 = T_{seq} + T_{par}$$

donde T_{seq} es el tiempo empleado en cálculos no paralelizables y T_{par} es el tiempo empleado en cálculos perfectamente paralelizables entre el número de procesadores disponibles.

El tiempo de ejecución en un multiprocesador con P procesadores es

$$T_p = T_{seq} + \frac{T_{par}}{P}$$

Entonces, el incremento de rendimiento que se obtiene al ejecutar el programa en un multiprocesador respecto de una ejecución serie se puede calcular utilizando la ley de Amdahl.

$$G = T_1 / T_p = \frac{1}{(1-\alpha) + \alpha/P}$$

donde α es la fracción de tiempo en una ejecución serie que se paraleliza.

Un multiprocesador puede operar en modo serie o paralelo. Suponga que en modo paralelo se utilizan 9 procesadores. Posteriormente se observa que el 25% de T_p se atribuye al modo paralelo. Durante el tiempo restante el programa se ha ejecutado en modo serie.

Pregunta 1: *Calcule la ganancia efectiva en las condiciones anteriores cuando se compara con una ejecución exclusivamente en modo serie. Además, calcule la fracción α de código que ha sido paralelizada.*

Pregunta 2: *Suponga que doblamos el número de procesadores. Calcule la ganancia efectiva que se obtiene.*

Pregunta 3: *Suponga que mediante técnicas de compilación y 9 procesadores se puede obtener la misma ganancia efectiva que en el apartado anterior, donde se había duplicado el número de procesadores. Determine la fracción α de código que ha sido paralelizada por el compilador.*

Ejercicio 2.4

El siguiente código resuelve un sistema triangular de ecuaciones.

```
do J=1,N
    X(J) = B(J) / A(J, J)
    do I = J+1, N                          bucle paralelo
        B(I) = B(I) - A(I,J) * X(J)
    enddo
enddo
```

En las siguientes preguntas considere sólo las dos sentencias de cálculo al evaluar el tiempo. Suponga que una sentencia de cálculo tarda un ciclo en ejecutarse.

Pregunta 1: *Calcule la fracción α de código que no se paraleliza.*

Pregunta 2: *Dibuje un perfil del paralelismo. Esto es, numero de cálculos paralelos en función de la variable de iteración J.*

Pregunta 3: *Suponga que se dispone de N procesadores. Calcule la ganancia respecto de una ejecución serie. Calcule también la eficiencia.*

Ejercicio 2.5

Al ejecutar una aplicación en un multiprocesador se distinguen 3 modos: a) se utilizan todos los procesadores, b) se utiliza la mitad de los procesadores y c) se utiliza un procesador. Suponga que el 2% del tiempo se ejecuta en un solo procesador y que hay 100 procesadores.

Pregunta 1: *Queremos obtener una ganancia de 80. Calcule la fracción máxima de tiempo en la que se solo se utilizan la mitad de los procesadores.*

Ejercicio 2.6

Considere la ejecución del siguiente código en un multiprocesador con 8 procesadores.

```
doall I = 1, N
    A(I) = A(I) + C(I)
enddo
```

Cada procesador tiene una cache que podemos considerar de tamaño infinito y con un tamaño de bloque de 32 bytes, ocupando cada elemento de los vectores 8 bytes. Solo existen fallos de carga o forzosos y no existen fallos de capacidad ni de conflicto debidos a la función de mapeo. En otras palabras, solo existe fallo de cache cuando se referencia por primera vez un dato si el bloque no está en la cache.

La red de interconexión entre la memoria y las caches privadas de los procesadores soporta un número ilimitado de transferencias en paralelo. En cuanto a la memoria es capaz de suministrar en paralelo un número ilimitado de peticiones de acceso. Es decir, no existen retardos debido a que varios procesadores soliciten acceso a memoria de forma concurrente.

Suponga un multiprocesador ideal donde las acciones de coherencia de cache se realizan en tiempo cero. Además las referencias de un procesador a memoria no están interferidas por los accesos de otros procesadores a variables almacenadas en el mismo bloque.

El tiempo de acierto en cache es de un ciclo de procesador. La penalización por fallo de cache son 100 ciclos de procesador.

Suponga que el tiempo de ejecución de las operaciones aritméticas y de las instrucciones de control del bucle es despreciable. Es decir, considere que el cuerpo del bucle son las instrucciones

```
load A(I)
load C(I)
store A(I)
```

Pregunta 1: *Calcule el tiempo de ejecución en un procesador.*

El bucle anterior se ejecuta en el multiprocesador utilizando 2 algoritmos de planificación estática.

A) Planificación simple: El bucle doall se convierte en los siguientes bucles

```
doall p=1, NP
  do I = (p-1)×⌈N/(NP)⌉+1, min(N, p×⌈N/(NP)⌉)
```

B) Planificación entrelazada: El bucle doall se convierte en los siguientes bucles

```
doall p=1, NP
    do I = p , N, NP
```

donde NP es el número de procesadores

Suponga que N es múltiplo del número de procesadores y que NP es igual a 8.

Pregunta 2: *Calcule, en el caso A, el tiempo de ejecución y la ganancia que se obtiene respecto a la ejecución en un procesador.*

Pregunta 3: *Calcule, en el caso B, el tiempo de ejecución y la ganancia que se obtiene respecto a la ejecución en un procesador.*

Pregunta 4: *Calcule la relación entre la ganancia en el caso A y el caso B y justifique el valor haciendo referencia a algún parámetro del multiprocesador.*

Ejercicio 2.7

Suponga un sistema multiprocesador con cinco procesadores. Al ejecutar un programa en el multiprocesador comprobamos que durante 1/6 del tiempo se han utilizado los 5 procesadores.

Pregunta 1: *Calcule la ganancia respecto a una ejecución serie.*

Pregunta 2: *Calcule el porcentaje de código (α) que ha sido paralelizado.*

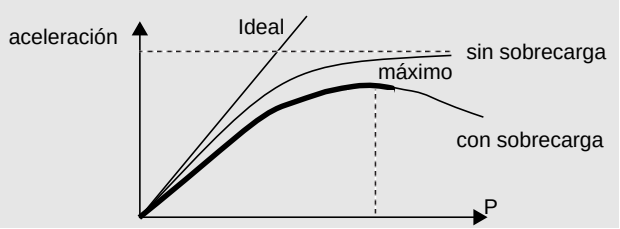
Pregunta 3: *Calcule la ganancia si suponemos que el código paralelo se puede ejecutar con un número ilimitado de procesadores.*

Suponga que se añaden 5 procesadores más. En estas condiciones el 80% del código paralelizable se puede ejecutar en 10 procesadores y el resto de código paralelizable se sigue ejecutando en 5 procesadores.

Pregunta 4: *Calcule la ganancia respecto a una ejecución serie.*

Pregunta 5: *Utilizando solo 5 procesadores, calcule el porcentaje de código (α_1) que debe ser paralelizado (mejora de la capacidad de paralelización del compilador) para obtener la misma ganancia que en el apartado 4).*

Suponga que el coste (sobrecarga) de iniciar los procesos en los procesadores y otros efectos debidos a la sincronización y comunicación determinan que hay que considerar un coste adicional en el tiempo de ejecución paralelo.



Pregunta 6: *En las condiciones de la pregunta 5) (α_1) y suponiendo que la sobrecarga, por el concepto expuesto anteriormente, es 0.005 veces el tiempo de ejecución en serie por el número de procesadores (0.005 x P x T_{serie}), calcule el número de procesadores con el que se obtiene la máxima ganancia.*

Nota: La derivada del producto de dos funciones es igual al primer factor por la derivada del segundo más el segundo factor por la derivada del primero. La derivada del cociente de dos funciones es igual a la derivada del numerador por el denominador menos la derivada del denominador por el numerador, divididas por el cuadrado del denominador.

Ejercicio 2.8

Para dar soporte a la sincronización se han diseñado dos instrucciones específicas denominadas Load Linked (LL) y Store Conditional (SC).

Load Linked (LL rd, (rf)): además de efectuar el load, tiene el efecto lateral de activar un bit denominado bit de enlace (bit LL) que se asocia al bloque de cache en el cual se mapea la dirección.

Este bit forma un enlace rompible entre la instrucción LL y la siguiente instrucción SC (store condicional).

La instrucción SC (SC rfd, (rf)), cuando se ejecuta, efectúa un store simple si el bit de enlace (LL bit) está activado. Si el bit de enlace no está activado, entonces el store no se ejecuta. La ejecución o no del store se indica en el registro (rfd) de la instrucción SC. Por tanto, este registro es fuente y destino.

El bit de enlace (bit LL) se desactiva si, mientras se ejecuta la secuencia de código entre LL y SC, se produce cualquier evento que potencialmente modifica el bloque accedido utilizando LL. Esto puede ser debido, entre otras causas: a) a una actualización externa del bloque que contiene la variable accedida mediante LL, b) a una invalidación externa del bloque que contiene la variable accedida mediante LL.

Codifique las siguientes operaciones atómicas utilizando las instrucciones LL y SC.

Pregunta 1: *Test&set (llave).*

test&set (llave)	comentarios
tmp = llave	llave es una variable global
llave = 1	
return tmp	

Pregunta 2: *Fetch&inc (dir).*

fetch&inc (dir)	comentario
tmp = dir	dir es una variable global
dir = tmp + 1	
return tmp	

Pregunta 3: *Compare&swap (viejo, nuevo, dir).*

Compare&swap (viejo, nuevo, dir)	comentario
tmp = dir	dir es una variable global
if (tmp = viejo) then	nuevo y viejo son valores almacenados en variables locales
dir = nuevo	
z =1	
else	
viejo = dir	
z =0	
endif	
return z	

Pregunta 4: *Fetch&and (dir, a).*

fetch&and (dir, a)	comentario
tmp = dir	dir es una variable global
dir = tmp and a	a es un valor almacenado en una variable local
return tmp	

Pregunta 5: *Sawp (dir, a).*

swap (dir, a)	comentario
tmp = dir	dir es una variable global
dir = a	a es un valor almacenado en una variable local
return tmp	

Ejercicio 2.9

En el siguiente programa paralelo la actualización de la variable global Total se efectúa creando una zona de exclusión mediante las primitivas atómicas obtener y liberar.

```
doall II =1 , P
   do I = II, N, P
      local (II) = local (II) + X(I)
   enddo
   obtener (llave)
      Total = Total + local (II)
   liberar (llave)
enddo
```

Modifique el programa de forma que se utilice la instrucción indivisible compare&swap para efectuar la actualización de la variable Total.

Compare&swap (viejo, nuevo, dir)	comentario
tmp = dir	dir es una variable global
if (tmp = viejo) then	nuevo y viejo son valores almacenados en variables locales
dir = nuevo	
z =1	
else	
viejo = dir	
z =0	
endif	
return z	

Ejercicio 2.10

El siguiente algoritmo efectúa una planificación del bucle por trozos de tamaño fijo.

```
region critica
  LI = GI
  GI = GI + K
end region critica
While (LI < N)
  do I = LI, min (LI+K-1, N)
    . . .
  enddo
  region critica
    LI = GI
    GI = GI + K
  end region critica
endwhile
```

comentario
GI: variable global que indica la primera iteración no ejecutada
K: número fijo de iteraciones asignadas. El valor ha sido determinado antes de iniciar la ejecución paralela.
LI: variable local que indica la primera iteración que ejecuta el hilo
GI está inicializado con el valor de la primera iteración del bucle
N es el número de iteraciones del bucle

Los procesadores disponen de las instrucciones atómicas fetch&add y compare&swap.

fetch&add (dir, a)	comentario
tmp = dir	dir es una variable global
dir = tmp + a	a es un valor almacenado en una variable local
return tmp	

Compare&swap (viejo, nuevo, dir)	comentario
tmp = dir	dir es una variable global
if (tmp = viejo) then	nuevo y viejo son valores almacenados en variables locales
dir = nuevo	
z =1	
else	
viejo = dir	
z =0	
endif	
return z	

Pregunta 1: *Reescriba el algoritmo utilizando la primitiva fetch&add.*

Pregunta 2: *Reescriba el algoritmo utilizando la primitiva compare&swap.*

Ejercicio 2.11

La operación de sincronización barrera se puede implementar utilizando la primitiva fetch&inc mediante el siguiente código.

barrera (bar)	inicialización	comentario
fetch&inc (bar) repeat until bar = N return	variable global bar = 0	N es el número de hilos que ejecutan la operación barrera

Un programador quiere reutilizar la posición de memoria bar en sucesivas utilizaciones de la operación barrera, de la forma que se muestra en los siguientes códigos paralelos (parte izquierda). Para ello ha diseñado el siguiente código de la operación barrera (parte derecha).

código paralelo A	código paralelo B	barrera (bar)
bar = 0	do = . . .	if (fetch&inc (bar) = N-1) then
doall I = 1, N	doall = . . .	bar = 0
. . .	. . .	else
enddoall	endoall	repeat
barrera (bar)	barrera (bar)	until bar = 0
doall I = 1, N	enddo	endif
. . .		return
enddoall		
barrera (bar)		
. . .		

Pregunta 1: *Muestre que esta implementación de la operación barrera no funciona correctamente.*

Nota: suponga que algún hilo llega a la segunda barrera antes de que el resto de hilos hayan acabado de ejecutar la primera barrera.

Pregunta 2: *Proponga una implementación de la operación barrera que funcione correctamente en el ejemplo anterior. Especifique si las variables utilizadas son globales o locales.*

Nota: desacople la espera activa de la variable que se utiliza para contabilizar los hilos que han llegado a la barrera. Además, la nueva variable debe actuar como un conmutador entre dos instancias de la operación barrera.

Ejercicio 2.12

En el diseño de un algoritmo de planificación de bucles se tiene en consideración la siguiente característica: los hilos del bucle paralelo compiten con otros procesos o hilos en el sistema multiprocesador y es posible que no estén disponibles todos los procesadores necesarios en el mismo instante de tiempo. Por tanto, los hilos inician la ejecución en tiempos distintos.

Para balancear la carga de trabajo la idea es asignar un número de iteraciones variable. El primer trozo que se asigna tiene un tamaño $\lceil N/P \rceil$, siendo N el número de iteraciones. Los siguientes trozos se van decrementando hasta que no quedan iteraciones. Este tipo de planificación se denomina guiada.

Una acción de planificación determina la asignación de un número de iteraciones K_i a un hilo, considerando que el resto de hilos (P-1) también se planifica en el mismo instante. Sea R_i el número de iteraciones que quedan por planificar, entonces en la siguiente planificación se asignan $K_i = \lceil R_i/P \rceil$ iteraciones a un hilo, en la siguiente planificación se asignan $K_i = \lceil R_{i+1}/P \rceil$ a otro hilo, donde $R_{i+1} = R_i - K_i$ y así sucesivamente. Inicialmente tenemos $R_0 = N$.

Pregunta 1: *Calcule para los siguientes valores la asignación de iteraciones: a) N=100, P =5 y b) N=1000, P =4.*

Pregunta 2: *Dibuje un diagrama de tiempos para el caso de la primera pregunta, suponiendo que los procesadores empiezan a trabajar en los siguientes instantes de tiempo y que cada iteración equivale a una unidad de tiempo.*

P1	P2	P3	P4	P5
0	5	3.5	10	17.5

Otros algoritmos que se utilizan para planificar bucles paralelos son: autoplanificación y planificación por trozos de tamaño fijo. Por autoplanificación se entiende que un hilo obtiene una iteración para ejecutar y al finalizar la ejecución compite con otros hilos por obtener otra iteración. Por planificación por trozos de tamaño fijo se entiende que el número total de iteraciones se distribuye entre los hilos antes de empezar la ejecución paralela. En este último caso suponga que a un hilo se le asignan iteraciones contiguas.

Pregunta 3: *Compare planificación guiada con autoplanificación y planificación por trozos de tamaño fijo. Razone sobre el número de planificaciones y la distribución de la carga.*

Pregunta 4: *Diseñe un algoritmo que implemente planificación guiada al ejecutar un bucle. Para ello dispone de las primitivas atómicas obtener (llave) y liberar (llave). Los valores de la primera y última iteración son 1 y N respectivamente.*

Ejercicio 2.13

En el diseño de un algoritmo de planificación de bucles se tiene en consideración la siguiente característica: en el cuerpo del bucle existen condicionales y existe una alta probabilidad de que los trozos asignados a los procesadores finalicen antes del tiempo previsto.

La idea, utilizada en la planificación, es dejar suficiente trabajo para los trozos que finalizan antes (alisar). A este tipo de planificación se le denomina factorización.

Las iteraciones se asignan en P trozos de igual tamaño K_i. El tamaño del trozo se calcula mediante la siguiente expresión $K_i = \lceil R_i/(2 \times P) \rceil$, siendo $R_{i+1} = R_i - P * K_i$, donde $R_0 = N$ y N es el número de iteraciones.

Pregunta 1: *Calcule para el siguiente caso la asignación de iteraciones: N=100, P = 5.*

Pregunta 2: *Escriba el esqueleto de un programa que muestra la planificación de un bucle utilizando planificación mediante factorización. Para ello dispone de las primitivas atómicas obtener (llave) y liberar (llave).*

Ejercicio 2.14

Un programa paralelo utiliza P procesadores de un multiprocesador. El encargado de recursos informáticos se plantea, manteniendo el mismo tiempo de ejecución, reducir el consumo de energía, reduciendo la frecuencia y utilizando más procesadores. Supondremos para simplificar que el CPI medio de los procesadores al reducir la frecuencia no se modifica (se mantiene el número de ciclos en fallo).

Sea α la fracción de código paralelizable medida en una ejecución serie del programa. La reducción en frecuencia representa multiplicar por r > 1 el tiempo de ciclo.

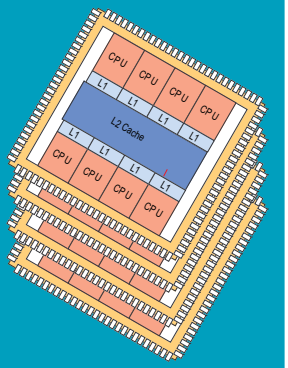
Pregunta 1: *Desarrolle una expresión que evalúe el número de procesadores necesarios (PP) en función de α, r y P.*

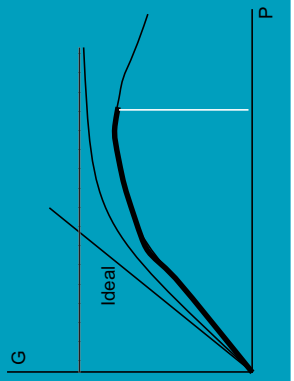
Pregunta 2: *Calcule la ganancia en energía (C x V^2) en función de P, PP y r. Para ello suponga que la tensión de alimentanción se reduce $\beta = 1 - (1/r)$ al reducir la frecuencia en la misma magnitud. Suponga también, para simplificar, que todos los procesadores consumen la misma energía durante todo el tiempo de ejecución.*

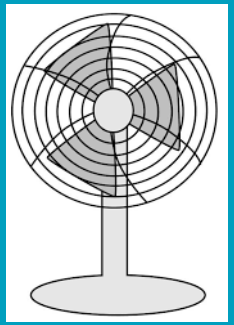
Seguidamente se quiere evaluar la influencia del tiempo de iniciación de los hilos del programa al ejecutarlo en paralelo. Para simplificar, supondremos que el programa es totalmente paralelizable y que el tiempo de iniciación de un hilo es T_i. Notemos que los hilos se inician de forma serie.

Pregunta 3: *Calcule el número máximo de procesadores (P) a partir del cual el tiempo de ejecución en paralelo se incrementa (mínimo tiempo de ejecución). El tiempo de ejecución en serie del programa es T_S.*

Pregunta 4: *Calcule el número de procesadores (P) a partir del cual el tiempo de ejecución en paralelo es mayor que el tiempo en serie (T_S). Suponga que $P >> 1$ y que $T_i << T_S$. El resultado no es infinito.*







Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 3
Consistencia de memoria y coherencia de cache

Los multiprocesadores han sido introducidos después de una amplia utilización de sistemas uniprocesador. Estos últimos se han utilizado para ejecutar programas serie. También mediante la multiplexación del procesador, denominada usualmente tiempo compartido, un sistema uniprocesador se ha utilizado para ejecutar varios procesos serie concurrentemente o varios procesos o hilos pertenecientes al mismo programa paralelo.

Un multiprocesador tiene como objetivos reducir el tiempo de ejecución de una aplicación paralela y/o incrementar la productividad al ejecutar varias tareas serie. Esto es, el número de tareas serie procesadas por unidad de tiempo.

Organización. En la Figura 3.1 se muestran organizaciones básicas de sistemas multiprocesador. Estas organizaciones pueden considerarse una extensión o transición natural de un sistema uniprocesador (máquina von-Neumman), donde mediante una red de interconexión se conectan varios procesadores y módulos de memoria.

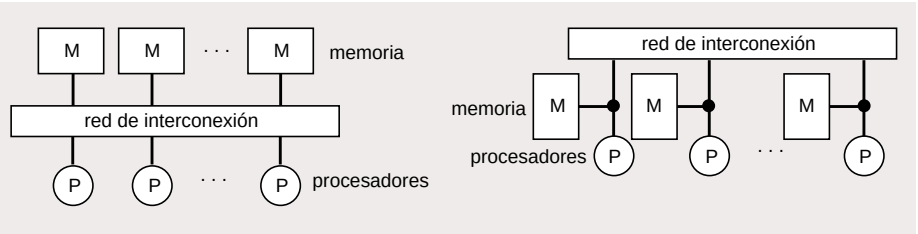


Figura 3.1 Multiprocesador de memoria compartida. a) Multiprocesador con memoria en un lado de la red de interconexión y procesadores en el otro lado y b) Multiprocesador con memoria físicamente distribuida.

En el multiprocesador mostrado en la parte izquierda de la Figura 3.1, en un lado de la red de interconexión se ubican procesadores y en el otro lado de la red se ubican módulos de memoria. En este tipo de multiprocesador la memoria podría estar constituida por un único módulo dependiendo de la red de interconexión utilizada. En el caso de que se utilicen varios módulos se puede acceder a cada uno de ellos de forma concurrente.

En el multiprocesador de la parte derecha de la Figura 3.1 los nodos son parejas módulo de memoria y procesador. En esta organización, notemos que uno de los módulo de memoria está más cerca de un procesador que los otros módulos de memoria y por tanto, la latencia de acceso es menor.

Espacio de direcciones global. Un sistema multiprocesador con memoria compartida dispone de un espacio de direcciones físicas global al que se tiene acceso desde cualquier procesador. El acceso al espacio de direcciones global se efectúa mediante instrucciones de load y store. Por ejemplo, esta característica es útil para los procesos del sistema operativo que acceden a las mismas estructuras de datos (estructuras de datos compartidas) y también facilita que los procesos puedan migrar entre procesadores. Por otro lado simplifica tareas tales como la distribución de la carga de trabajo entre los procesadores.

Programación. Los programas tanto serie como paralelos utilizan instrucciones store y load sobre una posición de memoria para comunicar información entre instrucciones distintas del mismo proceso o entre procesos. En este último caso, las instrucciones load y store se utilizan también para establecer ordenación entre dos procesos o hilos que se ejecutan concurrentemente, pero que cooperan para realizar una misma tarea. Como ejemplo, en la Figura 3.2 se muestra una sincronización punto a punto entre dos hilos. La variable aviso se inicializa a cero y permite sincronizar la lectura de la variable A en el hilo H2 con la escritura de la misma variable en el hilo H1.

Hilo H1	Hilo H2
A = 2.36	While (aviso <>1) {}
aviso = 1	T = A

Figura 3.2 Sincronización punto a punto entre dos hilos utilizando eventos. El valor inicial de la variable aviso es cero.

Un programador intuitivamente observa las operaciones de memoria (una en cada instante) en el orden especificado en el programa. Dada una posición de memoria, un load devuelve el último valor escrito en la posición de memoria accedida. Una instrucción store determina el valor que será devuelto por una instrucción load posterior o más joven hasta que la siguiente instrucción store actualice la posición de memoria.

En el trozo de código mostrado la Figura 3.2 se utiliza la variable aviso para establecer un orden entre la escritura de la variable A en el hilo H1 con la lectura de la misma variable en el hilo H2. Mediante la asignación del valor uno, a la variable aviso en el hilo H1, estamos indicando el instante a partir del cual se puede leer la variable A en el hilo H2. Mientras la variable aviso sea distinta de uno, el hilo H2 ejecuta un bucle, cuyo cuerpo es una instrucción load que lee la variable aviso. El programador espera que el hilo H2 lea el valor 2.36 al acceder a la variable A.

En programas paralelos que se ejecutan en un multiprocesador, la idea es aplicar razonamientos conocidos por diseñadores de sistemas operativos y bases de datos en un uniprocesador con tiempo compartido. Por tanto, interesa que la semántica del espacio de direcciones global, de un sistema multiprocesador, ofrezca un modelo de programación (visión de la memoria) compatible con los sistemas uniprocesador con tiempo compartido.

Semántica del espacio de direcciones global (consistencia de memoria).

En un procesador que se multiplexa o en un multiprocesador, entendemos por semántica del espacio de direcciones global compartido, cómo los procesos o hilos observan el entrelazado de los accesos a memoria, efectuado por cada uno de ellos de forma autónoma. Esta semántica es la que se utiliza al desarrollar programas paralelos y es la que permite razonar y predecir el comportamiento de los programas. Por tanto, hay que conocer la semántica del espacio de direcciones.

En la Figura 3.3 se muestra mediante un esquema la relación entre algoritmo, modelo abstracto de memoria global y acceso a posiciones de memoria en una máquina de cómputo. El modelo abstracto de memoria es una especificación formal de cómo el sistema de memoria es observado por el programador. Esto elimina la posible brecha que puede existir entre el comportamiento esperado del programa y cómo se comporta el sistema. En concreto, el modelo de memoria impone restricciones sobre el valor que puede ser devuelto en una lectura al ejecutar un programa.

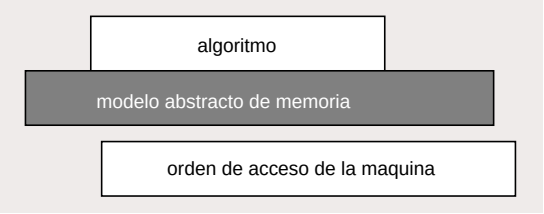


Figura 3.3 Modelo abstracto de memoria global. Esquema de relaciones entre niveles.

El modelo abstracto de memoria o modelo de consistencia de memoria influye en todo el sistema de cómputo, desde la programación, pasando por el rendimiento y finalizando por la portabilidad. El programador utiliza el modelo de consistencia para razonar acerca de la corrección de los programas. Los arquitectos y diseñadores de compiladores desarrollan optimizaciones que se explotan a nivel hardware y software, con el objetivo de incrementar el rendimiento. La portabilidad indica si un software desarrollado para un sistema de cómputo se puede ejecutar en otro sistemas de cómputo.

Consistencia de memoria. Determina cuándo un valor escrito en una posición de memoria será devuelto en una lectura.

La especificación del lenguaje máquina de un procesador incluye la semántica del espacio de direcciones global compartido o modelo de consistencia de memoria.

Organización con jerarquía de memoria. En la Figura 3.4 se muestran las organizaciones mostradas en la Figura 3.1 con cache privadas. La función de la cache es reducir el tiempo medio de acceso a instrucciones y datos por parte de cada procesador y reducir el ancho de banda demandado por cada procesador en la red de interconexión y en la memoria.

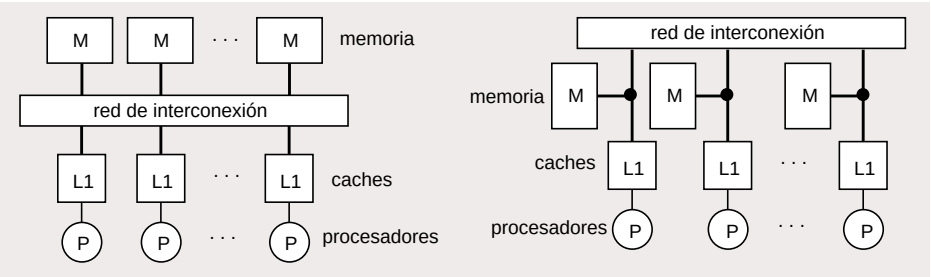


Figura 3.4 Organizaciones de sistemas multiprocesador con cache privadas.

Además, las caches privadas en los multiprocesadores mostrados en la Figura 3.4, habilitan la posibilidad de que existan copias de un mismo bloque de memoria en cada una de ellas, y por tanto, aparece la necesidad de que estas copias sean coherentes.

Coherencia de cache. Determina qué valor puede ser devuelto en una lectura a una posición de memoria.

El mecanismo utilizado en un multiprocesador para mantener la coherencia es transparente al lenguaje máquina. Debe ser eficiente, con coste reducido y no intrusivo.

Podemos decir que:

- Consistencia de memoria: está relacionada con el orden de todas las lecturas y escrituras a todas las posiciones de memoria.
- Coherencia de cache: está relacionada con el orden de todas las lecturas y escrituras a cada posición de memoria individual.

En este capítulo, en primer lugar se desarrolla el modelo de consistencia secuencial de memoria. Posteriormente se analiza como mantener coherentes las caches privadas (coherencia de cache).

Finalmente se muestra que el modelo de consistencia secuencial de memoria establece unas restricciones que son más estrictas que las restricciones necesarias que espera un programador al ejecutar un programa paralelo. Por ello se han desarrollado modelos de consistencia de memoria relajados, cuyo objetivo es posibilitar diseños hardware que incrementen el rendimiento y habilitar optimizaciones del compilador, que en caso contrario deben inhibirse. En contrapartida el programador debe prestar mayor atención al programar. En concreto, debe especificar en el código las zonas donde debe respetarse consistencia secuencial.

CONSISTENCIA SECUENCIAL DE MEMORIA

Recordemos que, al escribir programas paralelos que se ejecutan en un multiprocesador, la idea es aplicar razonamientos conocidos por diseñadores de sistemas operativos y bases de datos en un uniprocesador con tiempo compartido.

En la parte izquierda de la Figura 3.5 se muestra un modelo simplificado del proceso de multiplexación de un procesador. En este ejemplo, un procesador se multiplexa para ejecutar concurrentemente dos procesos serie. Un conmutador conecta un proceso a la memoria global y la posición del conmutador se determina de forma aleatoria después de cada acceso. Cada proceso efectúa los accesos a memoria en el orden especificado por el programador y el conmutador determina la serialización global entre todos los accesos. La multiplexación del procesador se produce entre acciones atómicas. En particular, las instrucciones load y store a una posición de memoria son operaciones (acciones) atómicas.

Informalmente, al ejecutar una programa serie o paralelo en un uniprocesador, que se multiplexa, esperamos que una lectura de una posición de memoria devuelva el valor más reciente escrito en esta posición de memoria.

Esto es, el valor escrito en una posición de memoria es observado por el siguiente load (del mismo o distinto hilo) que accede a esa posición de memoria, si antes no se ha escrito otra vez en esa posición de memoria.

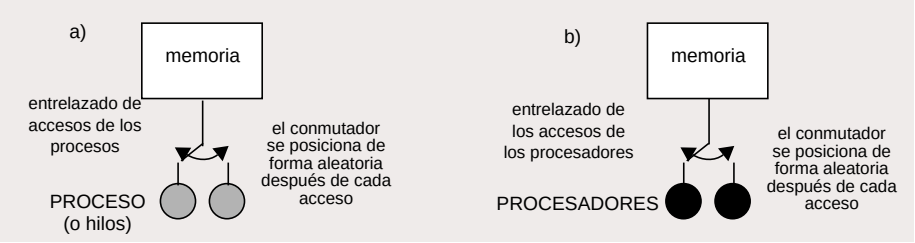


Figura 3.5 Modelo de consistencia de memoria. a) sistema uniprocesador con tiempo compartido y b) sistema multiprocesador.

El modelo de consistencia de memoria en un procesador con tiempo compartido se denomina consistencia secuencial. Informalmente, disponer de consistencia secuencial en un multiprocesador requiere que la ejecución de un programa paralelo se comporte igual que una ejecución entrelazada de los hilos del programa paralelo en un uniprocesador con tiempo compartido.

Conceptualmente, un multiprocesador con consistencia secuencial se comporta como si todos los procesadores tuvieran un turno para acceder a memoria, creando un flujo entrelazado de accesos, en el que los accesos de cada procesador individual están ordenados (Figura 3.5 parte derecha). Los procesadores efectúan accesos a memoria uno cada vez y el flujo de accesos a memoria es el entrelazado de accesos de los procesadores. Esto es, los accesos a memoria de todos los procesadores son atómicos y se puede construir un orden serie del entrelazado de accesos a memoria.

En la parte izquierda de la Figura 3.6 se muestra un modelo simplificado de multiprocesador. En la parte derecha de la figura se muestra un entrelazado de accesos cuando se ejecutan los dos hilos mostrados en la parte izquierda. En este entrelazado de accesos, se puede identificar el orden en el cual se han especificado los accesos en cada hilo. Por otro lado, una instrucción load lee el valor establecido por la instrucción store previa en el entrelazado.

Consistencia secuencial [Lamport]. Un multiprocesador es secuencialmente consistente si, el resultado de cualquier ejecución es el mismo que se obtendría cuando las operaciones de todos los procesadores se hubieran ejecutado en un orden serie (una cada vez), y las operaciones de cada uno de los procesadores aparecen en dicha secuencia en el orden especificado por el programador[1].

1. Desde el punto de vista del programador es un multiprocesador sin caches privadas y todos los accesos a posiciones de almacenamiento se encaminan a memoria, la cual sirve un acceso cada vez.

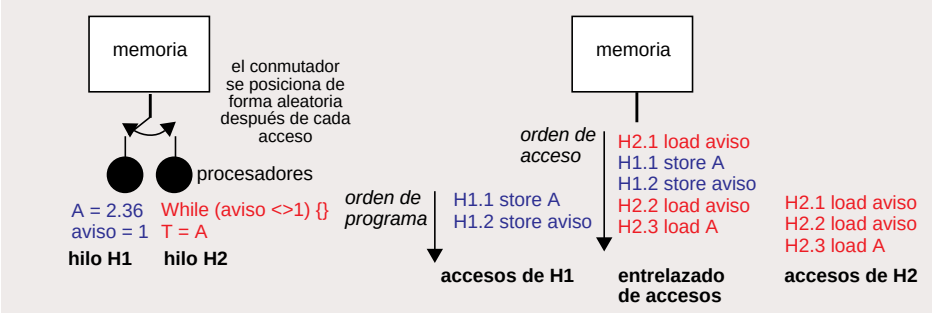


Figura 3.6 Entrelazado de accesos de dos hilos que se sincronizan. La variable aviso ha sido inicializada a cero. Notación: Hx.y, donde x es el número de hilo e y identifica el orden de programa.

De la definición previa de consistencia secuencial se extraen las siguientes condiciones.

Orden del programa. Dado un procesador, orden en el cual las operaciones de memoria o accesos a memoria son especificados por el programador (*las operaciones de cada uno de los procesadores aparecen en dicha secuencia en el orden especificado por el programador*).

Respetar el orden de programa indica que hay que mantener el orden de acceso en las cuatro combinaciones posibles de instrucciones load y store a posiciones de almacenamiento distintas (Figura 3.7)[2].

orden de programa				
↓	lectura	lectura	escritura	escritura
	lectura	escritura	lectura	escritura

Figura 3.7 Modelo de consistencia secuencial. Ordenes que hay que respetar en los accesos a posiciones de memoria.

Atomicidad. Todos los procesadores observan todos los accesos a memoria en el mismo orden (*cuando las operaciones de todos los procesadores se hubieran ejecutado en un orden serie, una cada vez*)[3].

Atomicidad indica una única operación indivisible o que se realiza instantaneamente.

2. Por otro lado, dado un hilo, para preservar su semántica, las dependencias al acceder a una posición de memoria deben respetarse.
3. Un acceso a memoria ha finalizado antes de que se efectúe el próximo acceso a memoria de cualquier procesador. En otras palabras, un acceso a memoria es visible instantáneamente a todos los procesadores.

Análisis del modelo de consistencia secuencial. En la Figura 3.8 se muestran dos hilos que efectúan dos accesos a memoria cada uno[4]. Para representar la acción del conmutador de la Figura 3.5 utilizamos un árbol binario[5]. Un orden de ejecución empieza en la raíz y finaliza en una hoja. Dada una rama, en cada nivel se representa la sentencia ejecutada, teniendo en cuenta que las sentencias se ejecutan en orden de programa y las sentencias que se han ejecutado previamente.

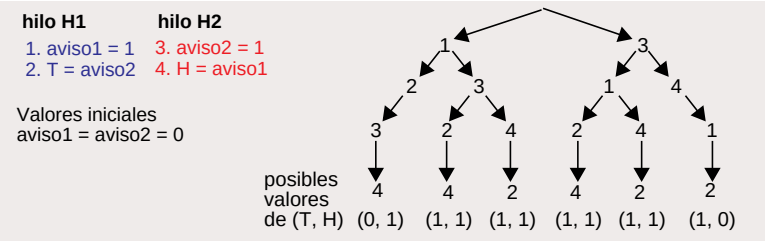


Figura 3.8 Posibles entrelazados secuencialmente consistentes.

Deducimos que hay seis posibles entrelazados que respetan el orden de programa y los posibles valores del par de variables (T, H) son (0,1) (1, 0) y (1, 1). Estos entrelazados son secuencialmente consistentes. Notemos que el valor del par (0, 0), al iniciar la ejecución de los hilos, no es posible observarlo al finalizar la ejecución. Un entrelazado que produzca el par (0, 0) no es secuencialmente consistente.

En una ejecución de un programa paralelo se observa un entrelazado. En ejecuciones distintas del programa paralelo pueden observarse entrelazados distintos.

Necesidades del modelo de consistencia secuencial

Orden de programa. Para garantizar orden de programa, un procesador debe recibir una respuesta del sistema de memoria.

- En una instrucción load la respuesta es el dato.
- En una instrucción store el sistema de memoria responde explícitamente, mediante una confirmación de la escritura[6].

Atomicidad. Un acceso a memoria es atómico (indivisible)[7]. Entonces, una vez una escritura llega a memoria, la posición de memoria accedida se actualiza y es visible por un load posterior, a la misma posición de memoria,

4. El código se corresponde con un esqueleto del algoritmo de Dekker para exclusión mutua.
5. En cualquiera de los dos esquemas mostrados en la Figura 3.5.
6. En un uniprocesador no es necesario una respuesta explícita en una instrucción store, ya que sólo hay un camino a memoria que mantiene el orden de los accesos a memoria. Además en este único camino se pueden gestionar dependencias de datos. Por tanto, en un uniprocesador al emitir una instrucción store no se espera respuesta del sistema de memoria.

efectuado por cualquier procesador, si previamente no se ha ejecutado otro store a la misma posición de memoria. Respecto a una instrucción load, el valor que se lee de memoria es el valor almacenado por la instrucción store previa, en el entrelazado de accesos de todos los procesadores, a esa posición de memoria.

- Paralelismo en memoria. En un sistema de memoria construido con un conjunto de módulos de memoria entrelazados también se cumple la condición de atomicidad[8]. Si los accesos son al mismo módulo de memoria, el orden está determinado por el instante en el que acceden al módulo de memoria. Accesos concurrentes a módulos de memoria distintos se efectúan de forma paralela y pueden ordenarse arbitrariamente, entre ellos, mientras se mantenga el orden en cada módulo de memoria.

En la Figura 3.9 se muestra un ejemplo donde el sistema de memoria dispone de dos módulos de memoria. La variable aviso1 está ubicada en el módulo M1 y la variable aviso2 en el módulo M2. En el primer instante de tiempo cada procesador accede a un módulo de memoria. Después de recibir la confirmación de las respectivas escrituras, cada procesador emite el siguiente acceso a memoria. Estos accesos son instrucciones load y también acceden a memoria de forma paralela. En la parte inferior derecha de la figura se muestran dos de los cuatro posibles entrelazados, secuencialmente consistentes, que pueden construirse a partir de los entrelazados parciales.

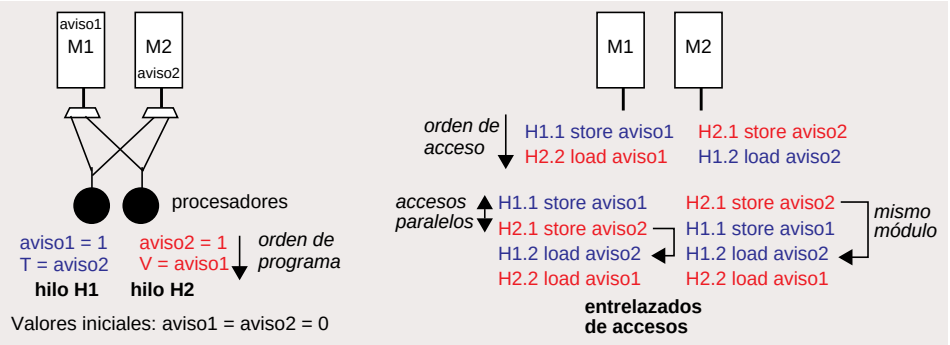


Figura 3.9 Sistema de memoria con dos módulos. Dos entrelazados posibles de accesos. Notación: Hx.y, donde x es el número de hilo e y identifica el orden de programa.

7. El acceso consolida con respecto a todos los procesadores antes de que el próximo acceso, en el entrelazado global, se inicie.
8. Estamos suponiendo orden de programa. Entonces, en un procesador el siguiente acceso a memoria se inicia después de que haya consolidado el previo. Por tanto, garantiza que los accesos de un procesador, a módulos de memoria distintos, se efectúan en orden de programa. También garantiza que una red con encaminamiento adaptativo (no siempre se utiliza el mismo camino para acceder desde un procesador a un módulo de memoria) no desordena, respecto al orden de programa, accesos efectuados por un procesador al mismo módulo de memoria.

Una propiedad que se puede extraer, de un sistema con varios módulos de memoria, es que la atomicidad de las escrituras[9] no requiere una serialización estricta de todos las escrituras. Es suficiente con la atomicidad de las escrituras a cada posición de memoria[10]. A partir de estos órdenes parciales se puede construir un orden global secuencialmente consistente.

Orden de programa

El orden, en que el programador ha especificado los accesos a memoria en un hilo, debe ser observado por todos los procesadores. Ningún componente del multiprocesador tiene que modificar el orden, ya sea el compilador, el procesador, la red de interconexión o memoria. Para ello hay que utilizar mecanismos explícitos o implícitos que permitan garantizar el orden de programa.

Implicaciones en compilación

En la parte izquierda de la Figura 3.6 se muestra un modelo simplificado de multiprocesador. En la parte derecha se muestra un entrelazado de accesos cuando se ejecutan los dos hilos mostrados en la parte izquierda.

El objetivo del código es comunicar el valor de A en el hilo H1 al hilo H2. La forma de establecer una relación de orden entre una escritura y una lectura, efectuadas por procesadores distintos, es utilizar una sincronización mediante eventos. Para ello se utiliza más de una posición de almacenamiento. Para preservar el orden, entre accesos a la misma posición de memoria, efectuados por procesadores distintos, hay que respetar el orden entre accesos a posiciones de almacenamiento distintas, efectuados por el mismo procesador[11].

En el ejemplo de la Figura 3.6 es importante el orden de programa, aunque si los dos hilos se analizan por separado no se puede observar esta necesidad. En el hilo H1, entre la asignación de la variable A y la asignación de la variable aviso no existe ninguna dependencia de datos. Igualmente ocurre en el hilo H2 entre la lectura de la variable aviso y la lectura de la variable A. Ahora bien, al considerar el programa paralelo, se observa que mediante la variable aviso se establece una dependencia de control dentro de cada hilo (Figura 3.10).

En la Figura 3.10 se muestra mediante relaciones de orden qué espera el programador al ejecutar el programa. El programador espera que se respete el orden de programa, aunque no existen dependencias de datos dentro de cada

9. Orden global de todos los stores a todas las posiciones de memoria.
10. Notemos que un módulo de memoria puede ser una única posición de almacenamiento.
11. Notemos que esta restricción también se aplica en el caso de un procesador que se multiplexa.

hilo. Podemos decir que son dependencias de control dentro de cada hilo en particular y en general en el programa paralelo. El orden de programa determina a -> b y c -> d. La ordenación b -> c implica la ordenación a -> d, donde -> indica relación de precedencia.

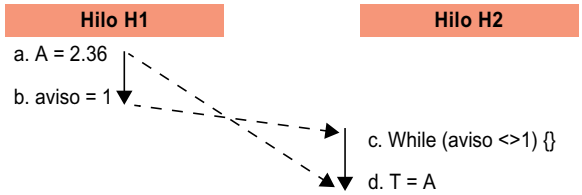


Figura 3.10 Sincronización punto a punto. Relaciones de dependencia entre dos hilos. El valor inicial de la variable aviso es cero.

El compilador no debe de utilizar algunas optimizaciones que se aplican en programas serie. Por ejemplo, el compilador no debe reordenar el orden de los accesos a memoria especificado por el programador[12]. Si en el ejemplo de la Figura 3.10 el compilador modifica el orden de acceso a las variables A y aviso, en cualquiera de los dos hilos, el programa no se comporta como esperamos.

Red de interconexión

La red de interconexión es un elemento que puede modificar la observación del orden de los accesos efectuados por un procesador por parte de otro procesador. En la Figura 3.11 se muestra un sistema multiprocesador que utiliza una red en malla. Cada nodo de la red tiene tres elementos: a) procesador, b) memoria y c) un encaminador, que encamina los accesos a otros nodos si no se sirven en este nodo. La memoria está entrelazada utilizando los bits menos significativos de la dirección. En el módulo de memoria del nodo 0 se almacena el bloque de memoria cero, en el nodo 1 se almacena el bloque de memoria uno y así sucesivamente.

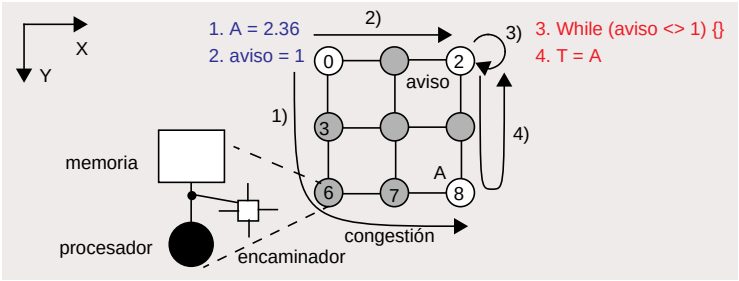


Figura 3.11 Multiprocesador que utiliza una red en malla.

12. En un apartado posterior se expone la necesidad de disponer de una primitiva específica para esta funcionalidad. Esta primitiva se utiliza cuando el modelo de consistencia de un lenguaje no es el modelo de consistencia secuencial.

En esta red, un acceso de un procesador a la memoria ubicada en otro nodo se encamina primero en la dirección Y y después en la dirección X. Por ejemplo, un acceso del procesador P0 a la memoria del nodo 8 se encamina en primer lugar por los nodos 3 y 6 y posteriormente por el nodo 7. En estas condiciones, los accesos entre dos nodos cualesquiera se transmiten de forma ordenada. Esto es, se mantiene orden punto a punto.

En el ejemplo de la Figura 3.11 el procesador P0 ejecuta un trozo de código de un hilo y el procesador P2 ejecuta otro trozo de código de otro hilo, los cuales pertenecen a un programa paralelo (Figura 3.10). Las variables A y aviso están almacenadas respectivamente en la memoria de los nodos 8 y 2.

Supongamos que se están transmitiendo por la red accesos de varios procesadores. Por congestión entendemos que el acceso de uno de ellos se ve retrasado por los accesos de los otros procesadores. Por ejemplo, en la Figura 3.11 existe mucha comunicación entre los nodos 3, 6 y 7. Entonces, supondremos que los accesos del procesador P0 al módulo de memoria ubicado en el nodo 8 experimentan retrasos.

Seguidamente se muestra la necesidad de que memoria confirme un acceso de escritura. El objetivo de esta señal de confirmación es garantizar que los accesos a memoria de un procesador sean observados por los otros procesadores en orden de programa.

No se confirma una escritura. Un procesador, después de emitir una instrucción store, ejecuta la siguiente instrucción sin esperar una confirmación. El procesador P0 emite una escritura que accede a la memoria del nodo 8. Seguidamente emite la siguiente escritura que accede a la memoria del nodo 2. El segundo store actualiza la variable aviso. El camino que siguen los dos accesos en la red de interconexión es distinto y también lo es el retardo en llegar al módulo de memoria

En la Figura 3.12.a se muestra un diagrama temporal del instante de inicio de una operación en un procesador, el instante en que se accede a memoria y en el caso de un load, el instante en el cual el procesador dispone del dato. En este último caso se utiliza una línea quebrada que parte del procesador, llega al módulo de memoria y vuelve al procesador. En el caso de una instrucción store la línea finaliza en el módulo de memoria.

En el procesador P2, al ejecutar el bucle se determina que la variable aviso ha tomado como valor uno (3). Por tanto, se lee la variable A (4). En el camino desde el nodo cero al nodo 8 hay congestión. El store emitido por el procesador P0 aún no ha actualizado memoria cuando se lee la variable A desde el

procesador P2. Entonces, el load del procesador P2 no lee el valor 2.36 sino un valor previo. Por tanto, la ejecución no cumple el modelo de consistencia secuencial.

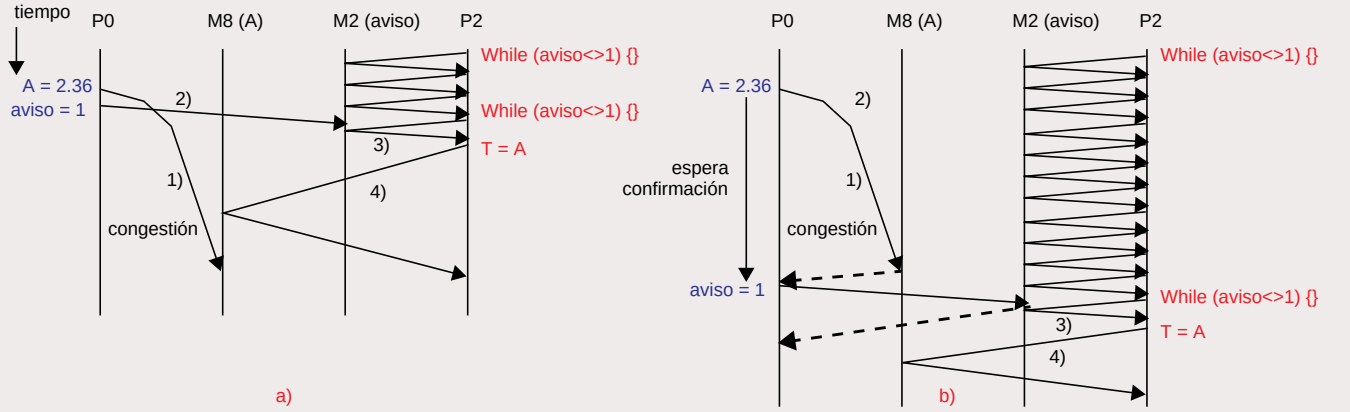


Figura 3.12 Diagrama temporal de los accesos mostrados en la Figura 3.11: a) No se confirman las escrituras, b) Se confirman las escrituras. Una línea de trazo discontinuo grueso es la confirmación de una escritura. La numeración de las acciones se corresponde con la numeración de la Figura 3.11.

El procesador P0 ha iniciado los accesos a memoria en secuencia, pero no son observados por otros procesadores en orden de programa. Como el procesador no espera confirmación de memoria cuando ejecuta una instrucción store, la red de interconexión determina que el orden observado por otros procesadores sea distinto del orden de programa[13].

Confirmación de una escritura. En la Figura 3.12.b se muestra un diagrama temporal donde un procesador, al emitir una instrucción store, espera confirmación de memoria, antes de ejecutar la siguiente instrucción de acceso a memoria[14]. De esta forma garantiza que el orden de programa sea observado, sin modificación, por los otros procesadores.

COHERENCIA DE CACHE

Para reducir la latencia de acceso a memoria se explota la propiedad de localidad de los programas mediante una jerarquía de memoria.

En un multiprocesador interesa también reducir el ancho de banda demandado por los procesadores en la red de interconexión. Por ello se ubican niveles de caches antes de la red de interconexión (caches privadas). Por otro

13. Caminos distintos, longitud distinta, latencia distinta y congestión en los enlaces.
14. Esta confirmación se representa mediante una línea gruesa de trazo discontinuo desde el módulo de memoria al procesador.

lado, esta disposición habilita la posibilidad de que lecturas simultáneas a la misma posición de memoria, por parte de varios procesadores, se efectúen en paralelo.

En la parte izquierda de la Figura 3.13 se muestra un ejemplo donde dos procesadores tiene copia en su cache privada de la variable A. Los dos procesadores pueden leer el valor de la variable A concurrentemente, utilizando la copia en la cache privada. No se utiliza la red de interconexión y no se accede a memoria. La latencia de acceso al dato es la latencia de lectura en la cache privada.

Ahora bien, en un multiprocesador cada procesador observa la memoria global a través de una jerarquía de memoria distinta. En la parte derecha de la Figura 3.13 se muestra que la jerarquía de memoria del procesador P1 es L11-Memoria y la jerarquía del procesador P2 es L12-Memoria. En estas condiciones es factible, aunque no es lo que se desea, que una lectura no devuelva el último valor que se ha escrito en una posición de memoria.

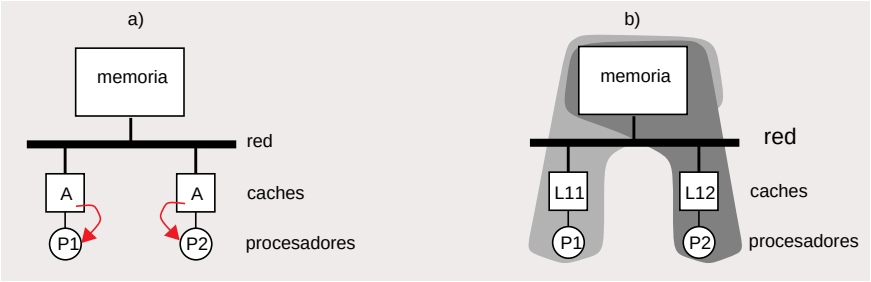


Figura 3.13 Sistema multiprocesador con procesadores que tienen cache privada. a) Lecturas concurrentes a copias de la variable en la cache privada. b) Jerarquía de memoria utilizada por cada procesador para observar la memoria global.

El problema que se plantea no se resuelve con las técnicas utilizadas en la jerarquía de memoria de un sistema uniprocesador (escritura retardada o escritura inmediata). Tampoco se resuelve utilizando los mecanismos empleados en un uniprocesador cuando se efectúa comunicación con el exterior del computador. Esto es, entrada/salida de información del computador. Los mecanismos que se suelen utilizar son groseros y no son eficaces en un sistema multiprocesador por la penalización que introducen.

Ejemplo. Para mostrar la necesidad de la coherencia de cache, utilizaremos como ejemplo la migración de procesos entre procesadores en un multiprocesador con cache privadas. En la Figura 3.14 se muestra un multiprocesador con dos procesadores y un proceso que migra entre los procesadores. Las caches privadas utilizan escritura retardada.

Supongamos que la variable T no está almacenada en ninguna cache inicialmente. Cuando el proceso se ejecuta en el procesador P1 lee la variable T y se produce un fallo de cache. El bloque que contiene la variable se lee de memoria y se almacena en un contenedor de su cache (Figura 3.14.a). Posteriormente el proceso migra al procesador P2 y lee la variable T. Se produce un fallo, el bloque que contiene la variable se lee de memoria y se almacena en un contenedor de su cache (Figura 3.14.b). Ahora hay 2 copias del bloque en las caches y las 2 copias tienen el mismo valor. Seguidamente, mientras el proceso se ejecuta en el procesador P2, se escribe la variable T (Figura 3.14.c). Ahora las 2 copias tienen valores distintos.

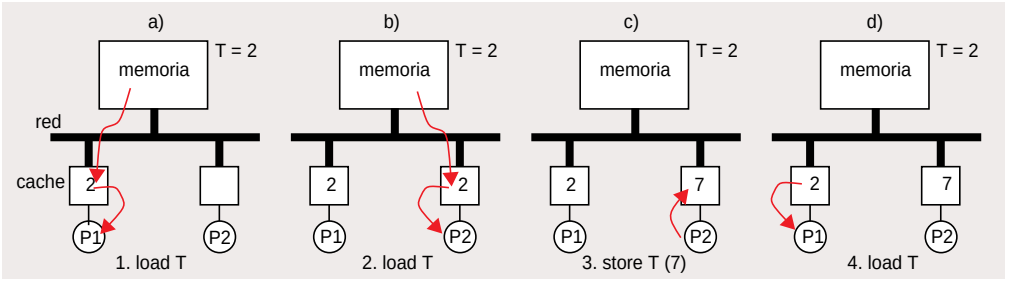


Figura 3.14 Sistema multiprocesador donde un proceso migra entre 2 procesadores.

Después de un intervalo de tiempo, el proceso migra al procesador P1 y lee la variable T. La variable T está almacenada en un contenedor de su cache y por tanto es un acierto (Figura 3.14.d). Ahora bien, el valor de la variable T ya no es el último valor que ha escrito el proceso en la variable T. Este valor está almacenado en la copia residente en la cache del procesador P2. Por tanto, la ejecución será incorrecta, ya que se lee un valor antiguo.

En la Figura 3.15 se muestra en cada fila una fotografía de las caches y memoria para cada acceso a memoria. Las filas están en orden temporal de arriba a abajo, representando en la primera fila el primer acceso.

Actividad del procesador	Actividad de la red	Cache de P1 (valor)	Cache de P2 (valor)	Memoria (valor)	Coherencia en copias
				2	
1. P1 load T	fallo	2	no hay copia	2	coherente
2. P2 load T	fallo	2	2	2	coherente
3. P2 store T		2	7	2	incoherente
4. P1 load T		2	7	2	incoherente

Figura 3.15 Tabla que muestra temporalmente el valor de las copias en un sistema multiprocesador.

En resumen, en un multiprocesador hay que diseñar un mecanismo que mantenga la coherencia entre las jerarquías de memoria utilizadas por cada procesador para observar la memoria global. En concreto, mantener la coherencia de los datos en las caches privadas. Este mecanismo debe ser transparente al lenguaje máquina y debe permitir la replicación y migración de datos entre caches privadas.

Protocolo de coherencia de cache

El objetivo de un protocolo de coherencia de cache hardware es detectar y eliminar las incoherencias. Para ello, las incoherencias deben reconocerse en ejecución.

Propagación de escrituras. Un protocolo de coherencia de cache es simplemente un mecanismo que propaga una operación de escritura a las copias en cache de la posición de memoria actualizada.

Políticas de actuación para propagar una escritura. Distinguiremos dos políticas, que se diferencian en el tipo de actuación al propagar una operación de escritura.

- Actualización (actualización en escritura). En una operación de escritura se actualizan todas las copias en las caches de otros procesadores y en la posición de almacenamiento en memoria.
- Invalidación (invalidación en escritura). En una operación de escritura se invalidan todas las copias en las caches de otros procesadores y en la posición de almacenamiento en memoria.

En la siguiente descripción de las políticas de coherencia, se supone que en cada instante sólo se está efectuando la acción que se describe. La idea no es mostrar una implementación, sino efectuar una descripción a nivel conceptual de la política.

En los dos casos que se muestran supondremos un multiprocesador donde los procesadores utilizan escritura retardada en la jerarquía de memoria observada. Como ejemplo se utilizará el mismo que en la Figura 3.14, donde un proceso migra entre procesadores.

Propagación de escrituras: política de actualización

En una política de actualización la granularidad de la acción de coherencia es la palabra. En la Figura 3.16.a y la Figura 3.16.b el proceso efectúa la lectura de la variable T y produce un fallo de cache en los procesadores P1 y P2. Cuando el procesador P2 efectúa la operación de escritura se transmite por la red esta intención y el valor que debe utilizarse para actualizar la variable T. La

cache del procesador P1 tiene una copia de la variable T y al observar, o conocer de alguna forma, esta acción actualiza su copia, utilizando el valor transmitido por la red (la escritura se propaga). Suponemos que memoria también se actualiza.

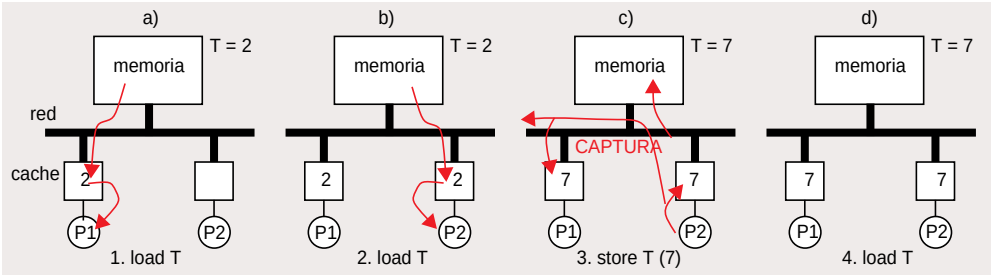


Figura 3.16 Política de actualización para mantener la coherencia.

Lecturas posteriores de la variable T, por parte de cualquiera de los dos procesadores, son acierto en cache. Una escritura de la variable T por parte del procesador P1 o P2, desencadena nuevamente una transmisión de la intención por la red, junto con el valor con el cual se actualiza la variable T. La otra cache, al conocer la intención actualiza la variable T.

Propagación de escrituras: política de invalidación

En una política de invalidación la granularidad de la acción de coherencia es el bloque. En la Figura 3.17.a y la Figura 3.17.b el proceso efectúa la lectura de la variable T y produce un fallo de cache en los procesadores P1 y P2. Cuando el procesador P2 efectúa la operación de escritura se transmite por la red esta intención, la cual es observada, o conocida de alguna forma, por la otra cache y memoria. Esta intención desencadena la invalidación de la copia del bloque, que contiene la variable T, en la cache de otros procesadores e implícitamente en memoria (Figura 3.17.c). Esto es, la escritura se propaga. Posteriormente se actualiza en la cache de P2.

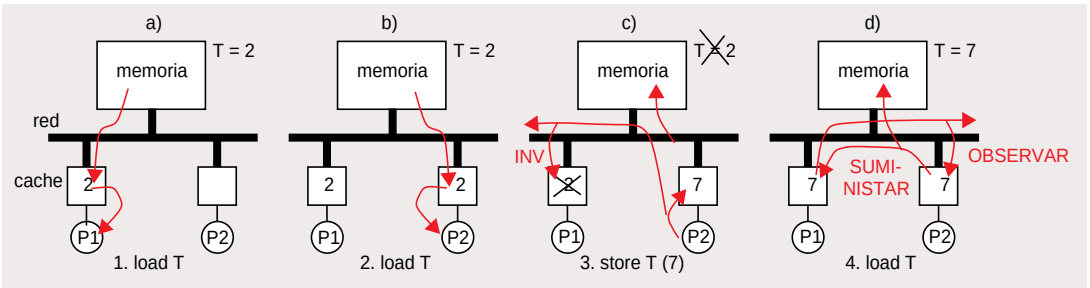


Figura 3.17 Política de invalidación para mantener la coherencia.

Posibles operaciones posteriores, de lectura o escritura, del procesador P2 no desencadenan ninguna otra acción, ya que en la cache de P2 se dispone de información, para determinar que es la única copia válida del bloque y se dispone de derechos de acceso para leer y escribir.

La política de invalidación requiere de un mecanismo que suministre la copia válida del bloque en una operación de lectura. Cuando el procesador P1 quiere leer la variable T, se detecta un fallo de cache y se transmite por la red la intención de lectura del bloque que contiene la variable T. Esta intención es observada, o conocida de alguna forma, por la cache del procesador P2, la cual suministra una copia del bloque de memoria a la cache del procesador P1. Adicionalmente, se actualiza el bloque en memoria. Lecturas posteriores de la variable T, por parte de los procesadores P1 y P2, son aciertos en sus respectivas cache. Una escritura de la variable T, por parte del procesador P1 o P2, desencadena nuevamente una invalidación de la otra copia del bloque que contiene la variable.

Notemos que para propagar un valor, la política de invalidación induce un fallo en las caches que tienen copia del bloque, que contiene la variable que se escribe. Entonces, al servirse el fallo se suministra una copia del bloque con el valor actualizado.

Tipos de protocolos de coherencia de cache

Los protocolos de coherencia se sustentan en seguir la pista del estado de cualquier bloque de memoria[15]. Se utilizan básicamente dos clases de protocolos de coherencia, los cuales utilizan técnicas distintas para seguir la pista del estado de un bloque de memoria.

En la Figura 3.18 se muestra un esquema de un sistema basado en directorio y de un esquema basado en observación o difusión.

Basado en directorio. El estado de un bloque de memoria se mantiene en una sola ubicación, denominada directorio. El directorio tiene información de la ubicación de las copias y el estado[16]. Las peticiones de los controladores de cache o de coherencia se encaminan mediante una red de interconexión al directorio. El directorio, a partir de la información de estado e identificación de posibles réplicas, toma las acciones oportunas para servir la solicitud[17]. Sólo

15. La granularidad del estado es a nivel de bloque, aunque un acceso a memoria tiene menor granularidad. Entonces, un acceso a memoria puede modificar el estado del bloque.
16. Por estado entendemos, en general, si hay copias y las operaciones que pueden realizarse en cada una de estas copias.
17. Peticiones a los controladores de coherencia que éstos responden.

se establece comunicación con los controladores de cache que tienen copia del bloque de memoria (Figura 3.18.a). El directorio, después de recibir las respuestas de los controladores de cache, responde al solicitante.

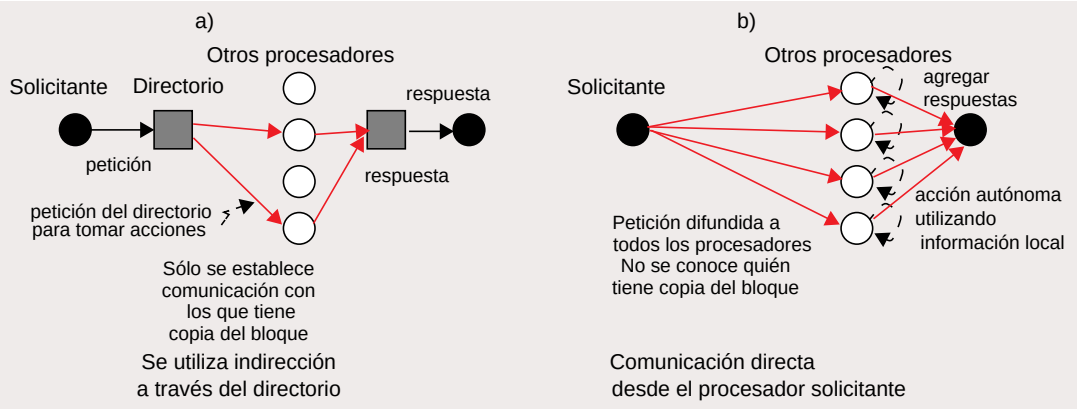


Figura 3.18 Tipos de protocolo de coherencia de cache: a) basado en directorio y b) basado en difusión u observación.

Basado en observación. Cada cache, que tiene una copia de un bloque de memoria, también tiene una copia del estado de compartición del bloque. No se mantiene ningún estado centralizado. Por tanto, la información de estado está distribuida. Las caches son accesibles mediante una red de interconexión, que permite difundir información (petición, respuesta). Todos los controladores de cache monitorizan u observan la información transmitida por la red de interconexión para determinar, a partir de la información difundida, si ellos tienen o no copia del bloque que es accedido. El resultado de la observación se agrega (agrupa) y se suministra al solicitante para que determine el estado de su copia del bloque. Además, de forma autónoma y utilizando información local, cada controlador de cache toma la acción oportuna, para actualizar el estado y servir la petición difundida, si es el caso (Figura 3.18.b). Memoria suministra el bloque cuando no es suministrado por una cache.

Consideraciones de implementación

La jerarquía de memoria es transparente al programador. Entonces, el comportamiento que observan los hilos, al acceder a una posición de memoria, debe ser el mismo que cuando no existen caches privadas. Una alternativa es permitir sólo una copia. Otra alternativa es propagar una escritura instantáneamente a todas las copias. Esto es, una escritura se propaga de forma atómica.

La propagación de una escritura de forma atómica establece un orden global serie de las escrituras a una posición de memoria[18]. En una lectura de una posición de memoria se devuelve el valor de la última escritura serializada, cuando está consolidada. Esto es, el valor de la escritura es observable por todos los procesadores[19].

En la Figura 3.19 se muestra un ejemplo con cuatro hilos. Todos los accesos son a la misma posición de memoria. Las instrucciones load en los hilos H3 y H4 deben observar las escrituras en el mismo orden. Por ejemplo, si el primer load del hilo H3 lee el valor escrito por el hilo H2 (8) y el segundo load, del mismo hilo, lee el valor escrito por el hilo H1 (3), no es posible que el primer load del hilo H4 lea el valor 3 y el segundo load, del mismo hilo, lea el valor 8.

Hilo H1	Hilo H2	Hilo H3	Hilo H4
store R4, A	store R2, A	load R6, A	load R9, A
		load R7, A	load R10, A

Figura 3.19 Propagación de escrituras. Cuatro hilos que acceden a la misma posición de memoria. El valor de los registros R4 y R2 es 3 y 8 respectivamente.

La propagación de una escritura no puede considerarse una acción instantánea o atómica, ya que en un multiprocesador la transmisión de información no es en tiempo cero[20].

Una forma de implementar una operación atómica es no permitir que los cambios que se producen progresivamente (subacciones) sean observados individualmente[21]. Todos los cambios serán visibles por todos los procesadores utilizando una única acción, cuando la operación ha finalizado (consolidado).

Son necesarios dos mecanismos para implementar la atomicidad de una escritura a una posición de memoria: a) identificar un punto o lugar de serialización[22] y b) una técnica para determinar la consolidación de una escritura (la propagación ha finalizado).

18. De forma idéntica a un multiprocesador sin cache, donde las escrituras consolidan en el módulo de memoria.
19. La atomicidad en cada escritura es una condición suficiente desde el punto de vista de coherencia de cache. La propagación de una escritura de forma atómica establece un orden global serie de las escrituras a una posición de memoria. Ahora bien, es posible relajar la condición y seguir manteniendo coherencia de cache. Esta relajación es útil en multiprocesadores con consistencia relajada de memoria.
20. Las lecturas cumplen la propiedad de idempotencia. El resultado de varias lecturas a la misma posición de memoria es el mismo que el de una lectura.
21. Por ejemplo, el valor que establece una escritura puede ser observado por alguno o algunos procesadores mientras que otros están observan un valor distinto.
22. Este punto o lugar de ordenación puede ser el mismo para todas las posiciones de memoria o un lugar distinto para cada posición de memoria.

Punto, entidad o lugar de serialización. Dada una posición de memoria, ésta tiene un lugar que se utiliza para serializar las escrituras[23]. Todos los procesadores encaminan los accesos a memoria a este punto de serialización. Esto es, esta entidad observa los accesos de todos los procesadores a la posición de memoria. Desde este punto de serialización se inician las acciones necesarias para mantener la coherencia.

Consolidación o finalización de la propagación de una escritura. Todas las caches deben responden a la solicitud de coherencia efectuada desde el lugar de serialización. Una vez se han recibido todas las respuestas puede considerarse que la escritura ha consolidado[24] [25].

La consolidación de una escritura se utiliza para[26]:

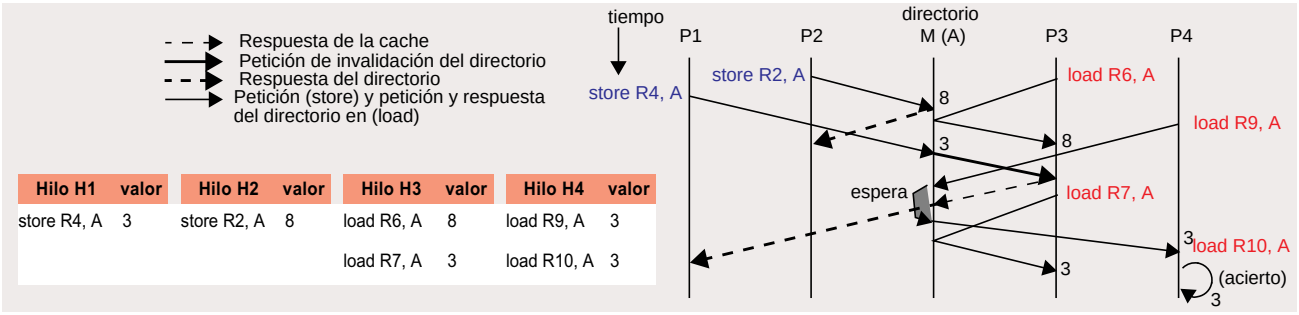
- Conocer cuándo el valor de una escritura es observable por todos los procesadores[27]. A partir de este instante se puede suministrar el valor a una instrucción load.
- Cuándo puede iniciarse el procesado de otra escritura.

Seguidamente se muestran dos ejemplos. Se utiliza un multiprocesador con caches privadas, que utilizan escritura inmediata sin asignación de contenedor en un fallo de escritura.

Para mantener la coherencia se utiliza un directorio y la política es de invalidación. El directorio confirma la escritura a la cache solicitante cuando la escritura ha sido consolidada. Para ello, el directorio espera las respuestas a las peticiones de invalidación, que ha efectuado a las caches que tienen copia. Suponemos que el directorio envía la señal de consolidación o confirmación, aunque conozca que no hay copia del bloque en el solicitante. También, suponemos que siempre se utiliza el mismo camino entre un emisor y un receptor y que en el camino se mantiene el orden de los mensajes.

23. En un multiprocesador sin cache privadas el punto de serialización es la memoria. En un multiprocesador con cache y un protocolo de directorio se puede utilizar el directorio como punto de serialización.
24. En función de propiedades de la red de interconexión no es necesario una respuesta explícita, está implícita en alguna acción efectuada en el punto de ordenación. Por otro lado, en una red que no mantiene el orden de mensajes entre emisor y destino pueden ser necesarios más mensajes o gestionar el desorden de la red en la implementación del protocolo de coherencia.
25. Un controlador de cache en una operación de lectura, mientras no recibe una acción de coherencia, suministra el valor almacenado en cache.
26. Desde el punto de vista de consistencia secuencial se utiliza para implementar orden de programa.
27. En un protocolo de coherencia con política de invalidación, el valor de una escritura no puede ser observado por un load hasta que han sido confirmadas todas las invalidaciones. En un protocolo de coherencia con política de actualización son necesarios dos pasos. En el primero se reciben las confirmaciones y posteriormente se indica a los nodos, involucrados en la acción de coherencia, que el nuevo valor es observable. En el intervalo temporal hay que esperar.

Ejemplo. En la Figura 3.20 se muestra una ejecución de los hilos de la Figura 3.19. Las caches no tiene copia de la variable A. Al procesar en el directorio la instrucción store de P1 hay que invalidar la copia en la cache de P3. El directorio espera, a que se consolide la escritura en curso, para responder a la primera instrucción load del procesador P4. La segunda instrucción load en el procesador P3 es un fallo de cache. La segunda instrucción load en el procesador P4 es un acierto en cache.

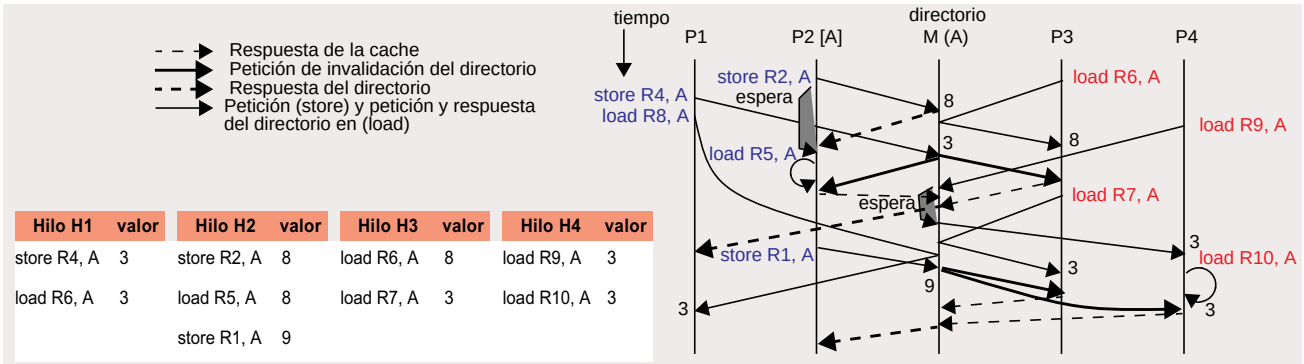


Hilo H1	valor	Hilo H2	valor	Hilo H3	valor	Hilo H4	valor
store R4, A	3	store R2, A	8	load R6, A	8	load R9, A	3
				load R7, A	3	load R10, A	3

Figura 3.20 Coherencia de cache. Propagación de escrituras I.

Ejemplo. En la Figura 3.21 se parte del código de la Figura 3.19 y los hilos H1 y H2 ejecutan instrucciones load después de las instrucciones store. Adicionalmente el hilo H2 ejecuta una instrucción store antes de finalizar. Por otro lado, el procesador P2 tiene copia de la variable A en la cache.

En el procesador P2 se espera la confirmación de que el store ha consolidado antes de ejecutar la instrucción load, la cual es un acierto en cache. La cache se actualiza al recibir la confirmación (la escritura es globalmente visible). La instrucción load del procesador P1 es un fallo en cache. El procesador efectúa la petición del bloque antes de recibir la confirmación de la instrucción store[28].



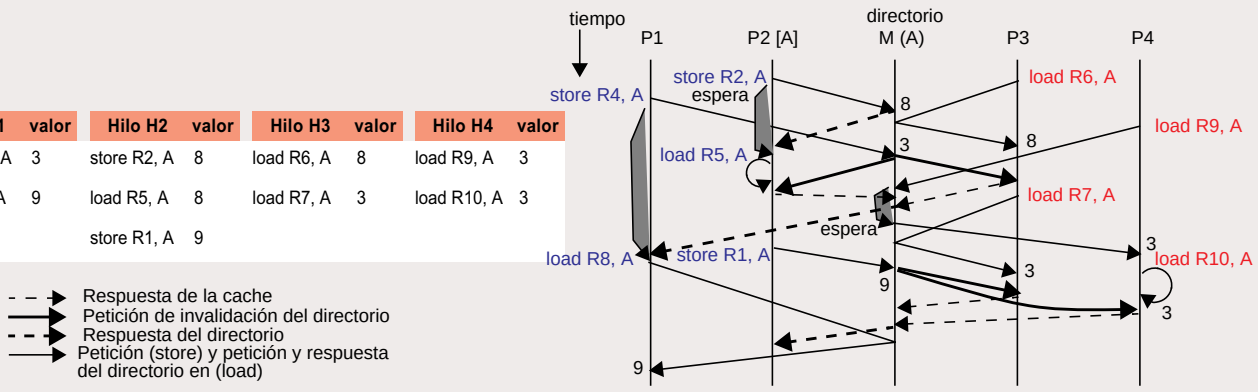
Hilo H1	valor	Hilo H2	valor	Hilo H3	valor	Hilo H4	valor
store R4, A	3	store R2, A	8	load R6, A	8	load R9, A	3
load R6, A	3	load R5, A	8	load R7, A	3	load R10, A	3
		store R1, A	9				

Figura 3.21 Coherencia de cache. Propagación de escrituras II.

Ejercicio *En el multiprocesador descrito previamente suponga que el camino utilizado entre un emisor y un receptor puede ser distinto para cada petición. Muestre un diagrama temporal de la ejecución del código de la Figura 3.21.*

Respuesta En la siguiente figura se muestra el diagrama temporal de ejecución.

Hilo H1	valor	Hilo H2	valor	Hilo H3	valor	Hilo H4	valor
store R4, A	3	store R2, A	8	load R6, A	8	load R9, A	3
load R6, A	9	load R5, A	8	load R7, A	3	load R10, A	3
		store R1, A	9				



Como la red puede utilizar caminos distintos, el acceso debido a la instrucción load del hilo H1 podría llegar al directorio antes que el acceso debido a la instrucción store. Para respetar las dependencias de datos en el hilo H1 es necesario esperar la confirmación de la escritura.

MULTIPROCESADOR SECUENCIALMENTE CONSISTENTE CON CACHE PRIVADAS

En el apartado de consistencia de memoria se ha mostrado que, para establecer un orden de accesos a una posición de memoria, efectuados por procesadores distintos, el sistema de memoria debe respetar el orden de lecturas y escrituras a diferentes posiciones de memoria, efectuadas desde un hilo (por ejemplo en la Figura 3.10). Coherencia de cache no establece relaciones entre accesos a posiciones de memoria distintas. Entonces, las condiciones de coherencia de cache son insuficientes para disponer de consistencia secuencial

Seguidamente se muestran las condiciones de consistencia secuencial en un multiprocesador con cache privadas, detallando los requisitos.

28. Notemos que no se espera a que la instrucción store esté consolidada. Ahora bien, hay que respetar el orden de accesos a la misma posición de memoria. Son dependencias de datos del hilo. Como se utiliza el mismo camino entre emisor y receptor y se mantiene el orden de los mensajes, está garantizado que se respetan las dependencias del hilo. La instrucción load se procesa en el directorio después de la instrucción store del mismo procesador. En general, entre ellas dos, en el directorio, pueden procesarse otros accesos a la misma posición de memoria.

Orden de programa. Un procesador debe esperar a que el anterior acceso a memoria se haya consolidado antes de iniciar el siguiente acceso a memoria en orden de programa.

- Finalización de una lectura. Cuando el dato llega al procesador.
- Finalización de una escritura. Conocer que la escritura ha sido consolidada. Esto es, han sido confirmadas todas las acciones de propagación de la escritura por las caches destinatarias[29]. Por tanto, la escritura ha sido serializada.

Es un mecanismo local del procesador que requiere conocer las confirmaciones de las acciones de propagación de una escritura.

Atomicidad. Todos los procesadores observan los accesos a memoria en el mismo orden.

- Atomicidad de una escritura. Conocer que una escritura ha sido consolidada, para procesar la siguiente escritura si es el caso. Por otro lado, una lectura obtiene el valor de la última escritura consolidada[30]. Esto es, el valor de una escritura sólo es observable cuando todas las acciones de coherencia han sido confirmadas. Por tanto, todos los procesadores observan las escrituras a la misma posición de memoria en el mismo orden[31].

Es un mecanismo disponible para gestionar cada bloque (unidad de gestión).

Orden de programa

Seguidamente se muestra, con un ejemplo, que el inicio del procesado en el directorio de una escritura no es una condición suficiente para garantizar que el orden de programa en un procesador es observado por los otros procesadores.

En la Figura 3.22 se muestra la organización del multiprocesador. La red y el encaminamiento de mensajes es el mismo que el descrito para el multiprocesador de la Figura 3.11. Los nodos disponen de un procesador, de una cache privada y de un módulo de memoria. La cache privada utiliza escritura inmediata. Los módulos de memoria se entrelazan de la misma forma que en la Figura 3.11.

29. Por ejemplo, el directorio puede ser el encargado de recolectar las respuestas de las caches a las peticiones de invalidación del directorio. Una vez han respondido todas las caches, el directorio confirma la escritura al procesador solicitante.
30. Si una escritura está pendiente de consolidación, una alternativa es esperar la consolidación para suministrar el valor a la lectura.
31. En el apartado “Necesidades del modelo de consistencia secuencial” se concluye que es suficiente con que las escrituras a cada posición de memoria sean atómicas.

Suponemos un protocolo de coherencia de tipo directorio y con política de invalidación. Un acceso a memoria, que no puede servirse localmente, accede al directorio[32], el cual está ubicado en el mismo nodo que el módulo de memoria donde se ubica la variable.

El código que se ejecuta en el multiprocesador es una sincronización por evento (Figura 3.10). La variable aviso está ubicada en la memoria del nodo cero y la variable A en la memoria del nodo 8. La cache del nodo 2 almacena las dos variables.

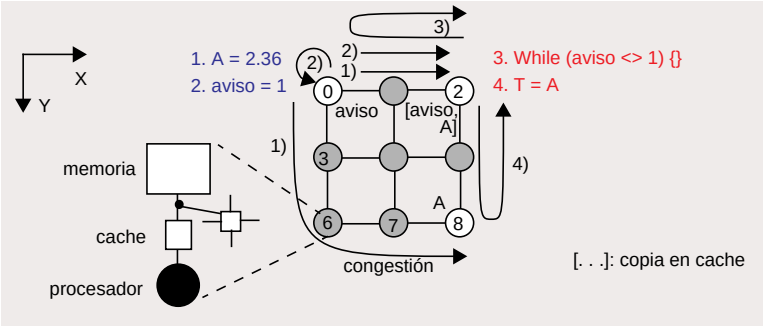


Figura 3.22 Multiprocesador cuyos nodos tienen cache privada. Orden de programa. Necesidad de esperar las respuestas a una acción de coherencia (no se muestran).

Para simplificar el diagrama temporal no se muestran los accesos debidos a la espera activa en el procesador P2.

En la parte izquierda de la Figura 3.23 se muestra el caso en el cual el directorio responde al emitir la acción de coherencia a la cache involucrada. Esto es, sin esperar la respuesta de la cache involucrada en la acción de coherencia. El procesador P0 utiliza esta respuesta para ejecutar la siguiente escritura (aviso). El resultado en el diagrama temporal de la Figura 3.23.a no es secuencialmente consistente. El valor leído de A en el procesador P2 es un valor anterior al escrito por el procesador cero[33]. La escritura de la variable aviso es observada en el procesador P2 antes que la escritura de la variable A. Las dos escrituras son efectuadas por el procesador P0, pero el orden de programa observado ha sido intercambiado por la red de interconexión.

En la parte derecha de la misma Figura 3.23 se muestra el caso en el cual el directorio espera la respuesta, de la cache involucrada, antes de emitir la respuesta a la cache que ha efectuado la petición. Esto es, la propagación ha consolidado y la escritura ha consolidado. Cuando el procesador P0 ejecuta la siguiente escritura, la escritura previa es globalmente visible. El resultado en el

32. Punto o entidad de serialización.
33. Notemos que las caches se mantienen coherentes. En algún instante la copia en la cache del nodo 2 es invalidada.

diagrama temporal de la Figura 3.23.b es secuencialmente consistente. El procesador P2 observa el orden de programa del hilo que se ejecuta en el procesador P0[34].

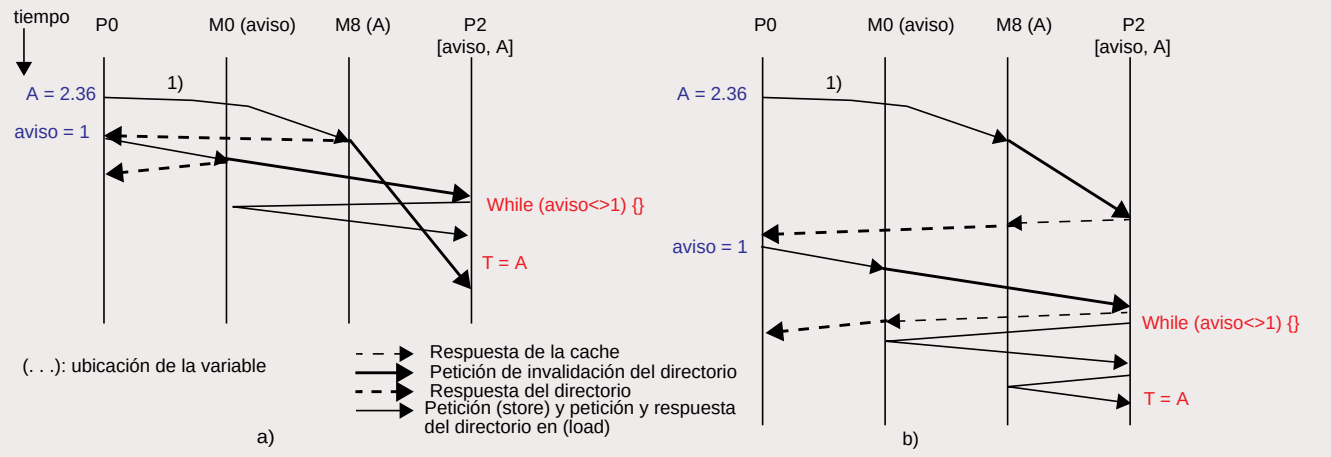


Figura 3.23 Necesidad de esperar las respuestas de las caches a una acción de coherencia: a) no se espera la respuesta de las caches, b) se espera la respuesta de las caches.

Atomicidad de la escritura

La atomicidad de los accesos a una posición de memoria, inducido por el modelo de consistencia secuencial, determina que el valor leído en una instrucción load es el valor escrito por la última instrucción store consolidada.

Suministro del valor de una escritura. El valor que establece una escritura (visibilidad) sólo puede suministrarse después de que todas las acciones de coherencia han sido confirmadas. Esto es, la propagación de la escritura ha finalizado[35].

Para mostrar la necesidad de atomicidad en las escrituras utilizaremos como ejemplo el código de la Figura 3.24. El multiprocesador que se utiliza, la red y el encaminamiento de mensajes es el mismo que el descrito para el multiprocesador de la Figura 3.11 (Figura 3.24). Los nodos disponen de un procesador, de una cache privada y de un módulo de memoria. La cache privada utiliza escritura inmediata. Los módulos de memoria se entrelazan de la misma forma que en la Figura 3.11.

34. Otra alternativa es que el directorio no responda a la lectura, de la variable aviso del procesador P2, hasta que recibe la respuesta de invalidación del bloque que contiene la variable A (Figura 3.23.a). Esta alternativa reduce la serialización en el procesador P0.
35. En un protocolo de coherencia con política de invalidación, el valor de una escritura no puede ser observado por un load hasta que han sido confirmadas todas las invalidaciones.

Suponemos un protocolo de coherencia de tipo directorio y con política de invalidación. Un acceso a memoria, que no puede servirse localmente, accede al directorio, el cual está ubicado en el mismo nodo que el módulo de memoria donde se ubica la variable.

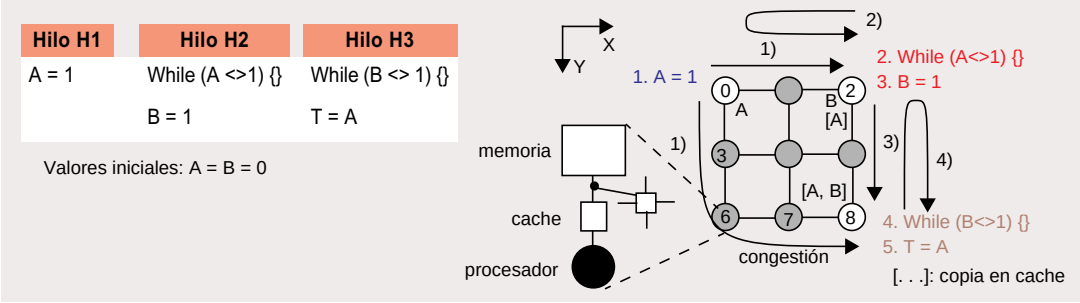


Figura 3.24 Requisito de atomicidad de las escrituras. Secuencias de código y organización del multiprocesador. Las respuestas a una acción de coherencia no se muestran.

El código consta de tres hilos (parte izquierda de la Figura 3.24). El hilo H1 se ejecuta en el procesador del nodo cero y los hilos H2 y H3 en los procesadores de los nodos 2 y 8 respectivamente.

La variables A y B están ubicadas en la memoria de los nodos 0 y 2 respectivamente. En las cache de los nodos 2 y 8 hay copia de la variable A y en la cache del nodo 8 hay copia de la variable B.

En el camino que comunica el nodo cero con el nodo 8 hay congestión. Esto es, los mensajes se retrasan por la comunicación existente entre otros nodos.

En la Figura 3.25 se muestran los casos relativos al suministro de un dato en una instrucción load: a) no se espera la confirmación de la propagación de la escritura a todas las cache involucradas en la acción de coherencia previa (la escritura aún no ha consolidado, Figura 3.25.a) y b) se espera la confirmación de la propagación de la escritura a todas las cache involucradas en las acción de coherencia previa (la escritura ha consolidado, Figura 3.25.b).

Para reducir el tamaño de los diagramas temporales no se muestran con la misma escala de tiempo (mensajes de invalidación desde el nodo 0). Tampoco se muestran los accesos debidos a la espera activa en los procesador P2 y P8[36].

36. Notemos que hasta que la escritura de un procesador no es propagada a todos los otros procesadores (ha consolidado), una lectura de estos últimos lee el valor previo que hay almacenado en cache. Una vez consolidada la propagación de la escritura todos los procesadores leen el mismo valor.

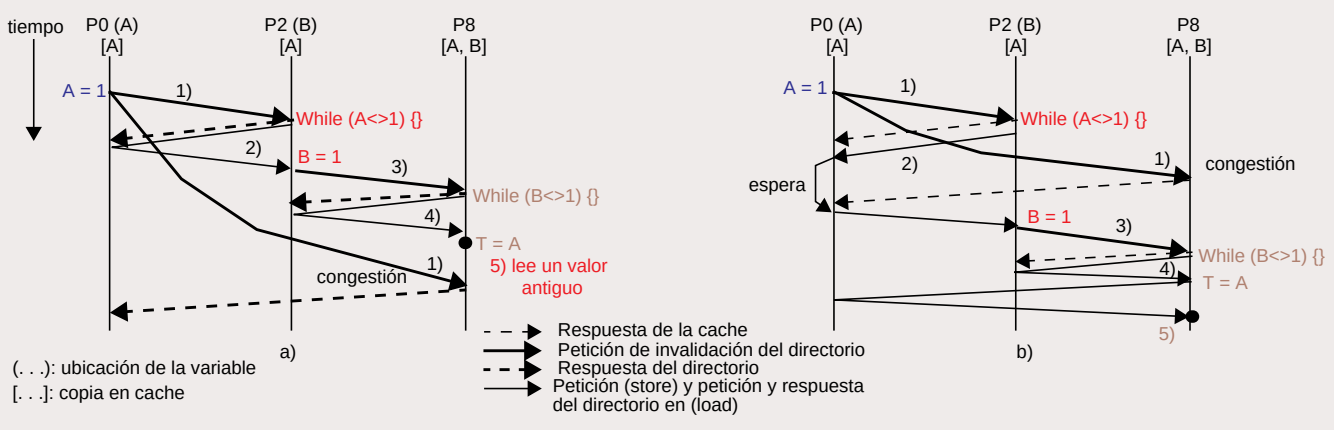


Figura 3.25 Diagrama temporales. Requisito de atomicidad de las escrituras.

En el nodo cero, al ejecutarse la instrucción store, el directorio emite mensajes de invalidación, del bloque que contiene la variable A, a los nodos 2 y 8. En el nodo 2, después de recibir y procesar la invalidación se produce un fallo de cache al leer la variable A. El controlador de coherencia emite un mensaje de petición de bloque al nodo cero, en cuyo módulo de memoria está ubicada la variable A. En el caso a) la memoria del nodo cero responde inmediatamente a la petición del nodo 2 (Figura 3.25.a). En cambio, en el caso b) la respuesta a la petición del nodo 2 se efectúa después de que se hayan recibido todas las confirmaciones a las peticiones de invalidación (nodos 2 y 8, Figura 3.25.b).

En el diagrama temporal de la Figura 3.25.a el valor leído de A en el procesador P8 es un valor previo al escrito por el procesador P0. No existe atomicidad en las escrituras, ya que el procesador P2 ha leído el valor uno[37]. En el diagrama temporal de la Figura 3.25.b se garantiza que la escritura de A por el procesador P0 se observa de forma atómica. No se suministra el valor hasta que se han recibido todas las confirmaciones a las invalidaciones emitidas.

CONSISTENCIA DE MEMORIA RELAJADA

La semántica de consistencia secuencial es intuitiva[38], pero es demasiado estricta: a) mantener orden de programa y b) atomicidad de las escrituras. Todo ello limita: a) implementaciones hardware y b) optimizaciones del compilador, cuyo objetivo es mejorar el rendimiento. Por ello, la mayoría de procesadores implementan un modelo más relajado de consistencia de memoria[39].

37. Observemos que la cache de P8 será coherente una vez llegue la invalidación.

En la Figura 3.26 se muestra un ejemplo donde el modelo de consistencia secuencial es demasiado estricto desde el punto de vista del resultado que espera obtener el programador (semántica). El análisis lo efectuamos desde el punto de vista de las posibilidades de reordenación de instrucciones de un compilador.

En la parte izquierda de la Figura 3.26 se muestran, mediante líneas finalizadas con flecha, las restricciones de orden del modelo de consistencia secuencial. En la parte derecha de la misma figura se muestran las restricciones de orden que garantizan la semántica que espera el programador. En este caso, las escrituras y las lecturas de las variables A y B pueden reordenarse entre ellas. Las únicas restricciones son que: a) las escrituras se especifiquen antes que la escritura a la variable aviso en H1 y b) las lecturas se especifiquen después de la lectura de la variable aviso en H2.

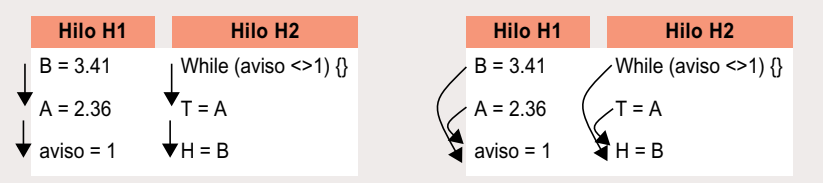


Figura 3.26 Código para mostrar que la semántica de modelo de consistencia secuencial es demasiado estricta. El valor inicial de la variable aviso es cero.

Lenguajes y compilación

Supondremos que el lenguaje de alto nivel utilizado para especificar los programas paralelos implementa un modelo relajado de consistencia[40].

Para establecer consistencia secuencial entre regiones de código utilizaremos la directiva LNbarrera[41]. Esta directiva define un punto, en la secuencia de instrucciones, en el cual se garantiza que un hilo consolida sus accesos a

38. Se adecua al modelo de un sistema uniprocesador multiprogramado.
39. En procesadores donde se utiliza especulación al ejecutar instrucciones, las implementaciones hardware, cuyo objetivo es mejorar el rendimiento del procesador, pueden hacerse transparentes al programador. Esto es, el modelo de memoria es secuencialmente consistente.
40. No se respeta ninguno de los ordenes de programa (Figura 3.7).
41. Lenguajes como OpenMP implementan un modelo relajado de consistencia. Para establecer consistencia secuencial entre regiones de código se utiliza el pragma o directiva FLUSH. !$OMP FLUSH (list) en Fortran y #pragma omp flush (list) en C/C++. Esta directiva está implícita en otras directivas. Un hilo pueden mantener una visión de la memoria que temporalmente no es secuencialmente consistente con la que observan otros hilos. Estas visiones temporales se pueden hacer secuencialmente consistentes en ciertos puntos del programa. La operación que fuerza que los accesos a memoria de un hilo, hasta ese punto, sean observados por otros hilos se denomina flush.

memoria: a) todos los accesos previos a memoria han consolidado y son observables por los otros hilos, b) no se han ejecutado accesos a memoria especificados posteriormente en el hilo.

La directiva LNbarrera además de limitar las optimizaciones del compilador genera una instrucción de lenguaje máquina, denominada INbarrera[42] (fence, sync, etc). Esta instrucción es utilizada por la arquitectura, si el modelo de consistencia que soporta no es secuencialmente consistente.

Nosotros no analizaremos en detalle la semántica de ningún lenguaje. Unicamente nos centraremos en los puntos donde interesa que se disponga de consistencia secuencial. Por ello, utilizaremos la primitiva LNbarrera para indicar que el compilador: a) no puede reordenar instrucciones o asignar una variable a un registro y b) genere una instrucción INbarrera.

Para garantizar que la escritura de una variable en un hilo sea consolidada y otro hilo observe este valor, deben de efectuarse las siguientes operaciones: a) el hilo H1 escribe la variable V, b) el hilo H1 ejecuta la instrucción generada al incluir la directiva LNbarrera (instrucción INbarrera), c) el hilo H2 ejecuta la instrucción generada al incluir la directiva LNbarrera (instrucción INbarrera) y d) el hilo H2 lee la variable V.

Seguidamente utilizamos un ejemplo para mostrar la utilización de la directiva LNbarrera. En el código de la parte izquierda de la Figura 3.27, por ejemplo, el compilador puede reordenar las lecturas y escrituras a cualquier variable. También puede asignar la variable aviso a un registro.

Hilo H1	Hilo H2
a. A = 2.36	c. While (aviso <> 1) {}
b. aviso = 1	d. T = A
Valor inicial : aviso = 0	

Hilo H1	Hilo H2
a. A = 2.36	LNbarrera
LNbarrera	c. While (aviso <> 1) { }
b. aviso = 1	LNbarrera
LNbarrera	d. T = A

Figura 3.27 Secuencias de código para mostrar la utilización de la directiva LNbarrera.

En el código de la parte derecha de la Figura 3.27 se muestra la inserción de la directiva LNbarrera. En el hilo H1 la primera LNbarrera asegura que la escritura de la variable A consolida antes de efectuar la escritura de la variable aviso. La segunda LNbarrera asegura que la escritura de la variable aviso consolida antes de ejecutar alguna instrucción más joven[43]. En el hilo H2, la

42. No hay que confundir la directiva LNbarrera y la instrucción INbarrera con la operación de sincronización que se denomina BARRERA.

43. La inserción en este punto de la primitiva LNbarrera y la generación de la instrucción correspondiente es una acción conservadora. Depende del código que se especifica posteriormente.

primera LNbarrera asegura que la variable aviso se lee de memoria[44]. La segunda LNbarrera asegura que la variable A se lee de memoria, después de que consolide la lectura de la variable aviso.

En función del modelo de consistencia de memoria de la arquitectura, puede no ser necesario la generación de la instrucción INbarrera cuando se especifica la primitiva LNbarrera. Notemos que una arquitectura puede garantizar algunos de los ordenes de programa (Figura 3.7). En el siguiente apartado se analiza un caso.

Optimización hardware

El modelo de consistencia secuencial determina que la latencia de una escritura puede ser significativa. Hay que esperar la confirmación de las acciones de coherencia. Ello representa una pérdida de rendimiento, ya que un procesador no puede ejecutar el siguiente acceso a memoria hasta que ha sido consolidado la escritura.

La relajación del modelo de consistencia, en el cual un acceso de lectura a memoria no debe esperar la consolidación de la escritura previa, en orden de programa, se denomina Total Store Order (TSO) y permite la introducción de un buffer de escrituras (BE) en la microarquitectura del procesador[45] (Figura 3.28).

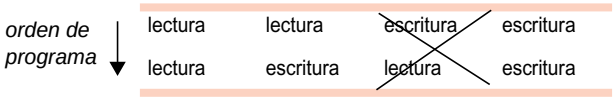


Figura 3.28 Modelo relajado de consistencia de memoria. Total Store Order,

Buffer de escrituras

Los procesadores ejecutan las instrucciones en orden de programa. Para que las escrituras no bloqueen al procesador se utiliza un buffer denominado de escrituras (BE). Una vez una instrucción store ha finalizado la ejecución en el procesador, en orden de ejecución[46], se almacena en el BE. Posteriormente se actualiza memoria.

44. Tengamos en cuenta que el mismo hilo podría haber escrito previamente la variable y estar almacenada en el buffer de escrituras. Se consolida la escritura.
45. En estas condiciones el buffer de escritura no es transparente al programador desde el punto de vista del modelo de consistencia. Si es transparente desde el punto de vista de las dependencias entre instrucciones de acceso a memoria ejecutadas en el procesador. Esto es, en la microarquitectura de un procesador hay hardware que gestiona las dependencia de datos entre las instrucciones de acceso a memoria que ejecuta el procesador.
46. Orden en el cual el procesador ejecuta las instrucciones

Cada entrada del BE tiene un campo de dirección y un campo de dato y el orden en que se almacenan sucesivas instrucciones store en el BE mantiene el orden de programa (parte derecha del la Figura 3.29).

El objetivo de un BE es soportar la latencia de las escrituras. El procesador sigue ejecutando instrucciones. Una instrucción de cálculo actualiza el estado del procesador. Una instrucción store se almacena en el BE. Si la instrucción es un load, se inicia el acceso a memoria adelantando a las instrucciones store almacenadas en el BE. La actualización de memoria desde el BE se efectúa en orden de programa de las instrucciones store.

Para que el BE sea transparente a la arquitectura del procesador[47] se añade circuitería a su alrededor. Si una instrucción load más joven accede a una posición de memoria almacenada en el BE, el procesador obtiene el valor de la instrucción store más joven, que accede a la misma posición de memoria almacenada en el BE[48]. Si el BE está lleno y se inicia la interpretación de una nueva instrucción store, se bloquea la interpretación de instrucciones hasta que se ha vaciado el BE. En la parte izquierda de la Figura 3.29 se muestra un esquema de parte del camino de datos relativo al BE.

Multiprocesador con procesadores que utilizan un BE

En la parte derecha de la Figura 3.29 se muestra un esquema de un multiprocesador con dos procesadores. El multiprocesador que utilizaremos como ejemplo tiene un bus como red de interconexión. El bus está ocupado durante la latencia de acceso a memoria. En cada procesador pueden competir por el bus la entrada en la cabeza del BE y una instrucción load. En un procesador una instrucción load obtiene de forma prioritaria el acceso al bus.

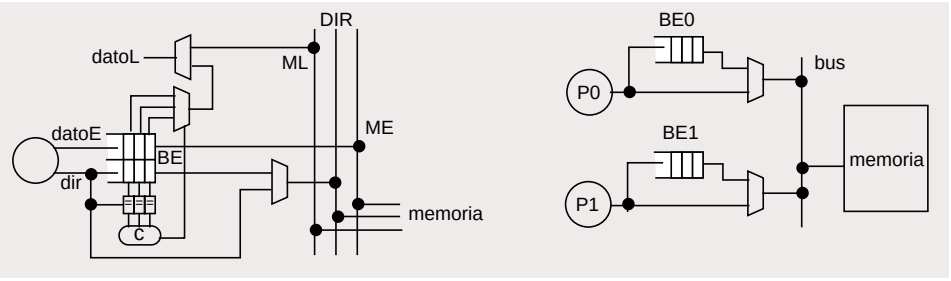


Figura 3.29 Esquema del camino de datos del BE. Multiprocesador cuyos procesadores utilizan un buffer de escrituras (BE).

47. Respeten las dependencia de datos entre instrucciones de acceso a memoria que ejecuta el procesador.
48. Otra alternativa es que el procesador se bloquee mientras perdura el riesgo de datos. El riesgo desaparece cuando la instrucción store, almacenada en el BE, actualiza memoria y se extrae del BE.

La actualización de memoria desde el BE se efectúa en orden de programa de las instrucciones store. Una escritura emitida desde el BE no consolida hasta que recibe la respuesta de memoria[49]. Entonces, se extrae del BE y la siguiente entrada puede competir por acceder al bus[50].

Consistencia secuencial de memoria

Mientras no se emita al sistema de memoria una escritura, a una posición de memoria, ésta no es serializada y en consecuencia no está consolidada. Por tanto, no es observable por otros procesadores. En estas condiciones, la utilización del valor de una escritura para suministrar un valor a una instrucción load, que accede a la misma posición de memoria, determina que la escritura no sea atómica. Por tanto no se cumple consistencia secuencial.

Seguidamente utilizamos un ejemplo para mostrar la utilización de la instrucción INbarrera con el objetivo de establecer consistencia secuencial en puntos del código donde sea necesario.

En el multiprocesador de la Figura 3.29 se ejecuta el esquema de código que se muestra en la parte izquierda de la Figura 3.30, que es el mismo que el mostrado en la Figura 3.8 con la inclusión de la directiva LNBARRERA. En la parte derecha se muestra el código en lenguaje ensamblador.

Hilo H1	Hilo H2
aviso1 = 1	aviso2 = 1
LNbarrera	LNbarrera
T = aviso2	H = aviso1

Valores iniciales: aviso1 = aviso2 = 0

Hilo H1	Hilo H2
store R4, aviso1	store R5, aviso2
INbarrera	INbarrera
load R6, aviso2	load R7, aviso1

Figura 3.30 Esquemas de código para mostrar la gestión del BE cuando se interpreta una instrucción barrera. Los registros R4 y R5 almacenan el valor uno.

En la parte izquierda de la Figura 3.31 se muestra un diagrama temporal de los accesos a memoria al ejecutar el código de la Figura 3.30 sin las instrucciones INbarrera. El primer acceso de cada procesador es una escritura. Cada una de ellas se almacena en el BE del procesador que la ejecuta. El almacenamiento en los BE se efectúa de forma paralela. El siguiente acceso, de cada uno de los procesadores, es una lectura. El procesador P1 lee la variable aviso2 y el procesador P2 lee la variable aviso1. Las lecturas tienen prioridad respecto de la cabeza del BE. Entonces, las lecturas obtienen el bus.

49. El tipo de red utilizada (bus) permite que la respuesta pueda considerarse implícita al obtener acceso al bus.
50. En un sistema uniprocesador es necesario vaciar el BE en un cambio de contexto del procesador. Pensemos que aunque se utilicen direcciones físicas, el SO puede modificar la correspondencia entre el espacio lógico y el espacio físico.

El bus está ocupado durante todo el acceso a memoria. Entonces, los accesos se efectúan de forma serie. El valor leído de las variables aviso1 y aviso2 es cero. En consecuencia no se mantiene consistencia secuencial. Después de las lecturas se actualiza memoria con la información del BE[51].

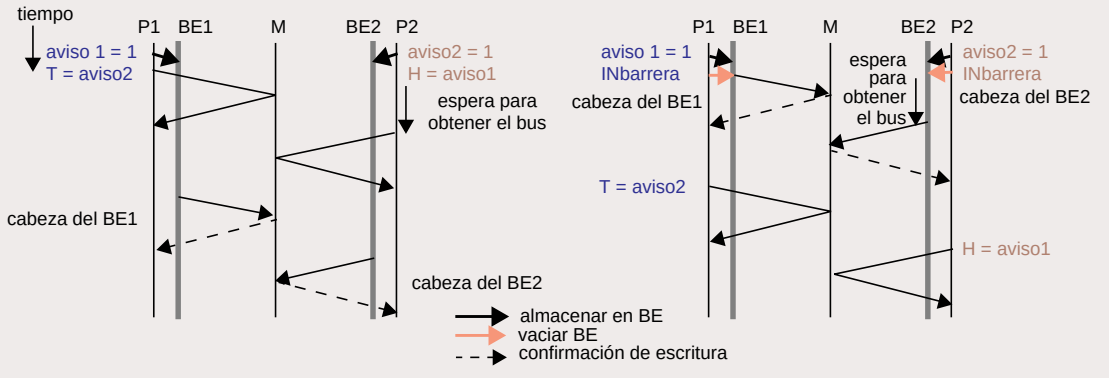


Figura 3.31 Multiprocesador con procesadores que utilizan un BE. Diagrama temporal de accesos a memoria.

En la parte derecha de la Figura 3.31 se muestra el diagrama temporal de los accesos a memoria al ejecutar el código de la parte derecha de la Figura 3.30 con las instrucciones INbarrera[52].

La microarquitectura utiliza la instrucción INbarrera para consolidar todos los accesos a memoria pendientes y no iniciar ningún acceso posterior hasta que se haya consolidado la instrucción INbarrera. Esto es, el BE debe vaciarse. Una vez consolidada la última entrada ocupada del BE se consolida la ejecución de la instrucción INbarrera. Entonces, el procesador puede ejecutar la siguiente instrucción de acceso a memoria. El valor leído de las variables aviso1 y aviso2 es uno. En consecuencia se mantiene consistencia secuencial.

Coherencia de cache

Respecto a coherencia de cache, la condición de atomicidad de las escrituras puede relajarse. Un procesador puede utilizar el valor de una escritura, antes de ser consolidada (propagada), en una instrucción load más joven. La coherencia de cache se mantiene, ya que otros procesadores no observan el valor. La lectura se puede ordenar, en el orden de accesos a la posición de memoria, inmediatamente después de que la escritura consolide[53] y el sistema es coherente.

51. Notemos que las escrituras se consolidan y lecturas posteriores leen el valor establecido por estas escrituras.
52. Notemos que la acción de una instrucción INbarrera es local. Establece ordenación de accesos a memoria en un hilo.

En estas condiciones, la condición de atomicidad de escrituras se relaja a la condición de serialización de escrituras, la cual es subyacente en la condición de atomicidad.

Serialización de escrituras. Todas las escrituras a una posición de memoria son observadas en el mismo orden por todos los procesadores.

Notemos que un procesador, después de ejecutar una instrucción store, puede ejecutar inmediatamente una instrucción load, a la misma posición de memoria, sin tener que esperar la confirmación de la escritura, es suficiente garantizar que la instrucción store y la instrucción load se ejecutan en orden de programa[54].

Ejemplo. Suponemos un multiprocesador que utiliza un bus como red de interconexión y cada procesador tiene un BE y una cache privada. Las cache privadas utilizan escritura inmediata y el protocolo de coherencia es de tipo invalidación (Figura 3.32).

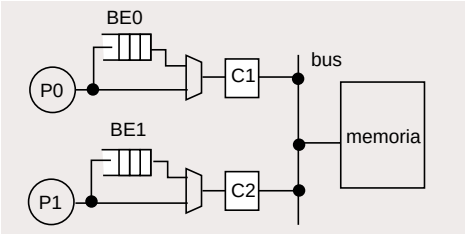


Figura 3.32 Multiprocesador con procesadores que utilizan un BE y tienen cache privada.

En la Figura 3.33 se muestra un ejemplo de código y su ejecución en el multiprocesador de la Figura 3.32. Los procesadores almacenan la instrucción store en el BE. Cuando ejecutan las instrucciones load obtienen el valor del BE correspondiente. En paralelo se propagan las escrituras.

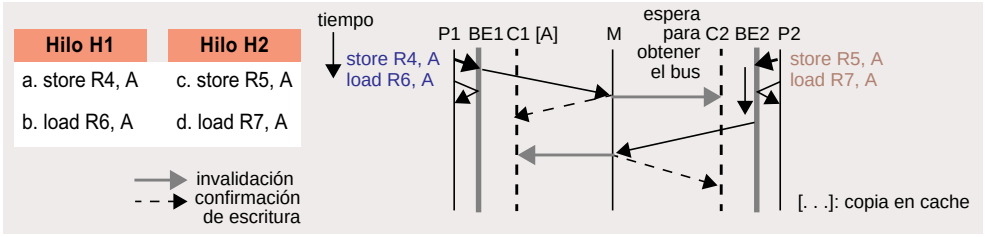


Figura 3.33 Serialización de escrituras.

53. Desde el punto de vista de coherencia, esta relajación permite la utilización de BE o que una cache se actualice antes de recibir la confirmación de la consolidación.
54. Es una dependencia de datos del hilo. Por ejemplo, en un multiprocesador donde los procesadores no tienen BE y el camino a memoria de las instrucciones store y de las instrucciones load es distinto hay que garantizar el orden. Para ello, la instrucción load debe esperar la confirmación de la escritura.

Al utilizarse un bus como red de interconexión las caches observan la transacción e invalidan la copia en cache. La cache C2 invalida la copia al propagarse la escritura del procesador P1. La cache C1 se actualiza al recibir la confirmación de la escritura.

La cache C1 invalida la copia al propagarse la escritura del procesador P2. La cache C2 no se actualiza al recibir la confirmación de la escritura, ya que el bloque está invalidado. Al finalizar la ejecución, la cache no tienen copia del bloque y memoria almacena el valor establecido por la escritura del procesador P2. El orden de los accesos a la posición de memoria A es: a, b, c, d.

Condiciones de coherencia de cache. A partir de ahora, las condiciones para coherencia de cache son: a) propagación de escritura y b) serialización de escrituras, en lugar de atomicidad de escrituras.

En estas condiciones, mediante el mecanismo de coherencia no se conoce el instante a partir del cual una escritura es observable por otros procesadores. El procesador que ejecuta una instrucción load puede obtener el valor de un entorno local como un BE o una cache.

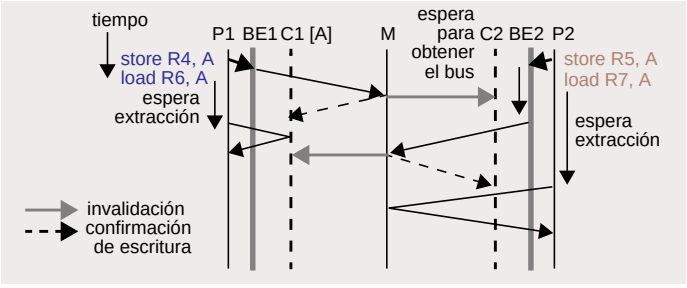
Ejercicio

Suponga que el BE dispone de lógica para detectar dependencias, pero no dispone de lógica para suministrar el dato. Razone la actuación que debe tomarse en caso de detectar una dependencia. Utilice el código de la Figura 3.33 para mostrar en un diagrama temporal el funcionamiento en el multiprocesador de la Figura 3.32*. Suponga que el orden temporal de ejecución de los accesos a memoria es: a, c, b, d.*

Respuesta

La lógica debe bloquear la instrucción load hasta que en el BE no exista una dirección coincidente. Una instrucción load obtendrá el valor de la escritura emitida por el mismo procesador o de una escritura emitida por otro procesador, en función de la serialización en el entrelazado de accesos a la misma posición de memoria.

En la siguiente figura se muestra el diagrama temporal.



La ejecución de la instrucción load espera a que se extraiga del BE la instrucción store con dirección coincidente. En el procesador P1 la instrucción load es acierto en cache. En el procesador P2 la instrucción load es fallo de cache, ya que la copia ha sido invalidada por la escritura P1.

EJEMPLOS

Orden de programa: compilador

Para mantener consistencia secuencial es necesario mantener los siguientes ordenes de programa en instrucciones de acceso a memoria que acceden a posiciones distintas de memoria.

orden de programa ↓			
lectura	lectura	escritura	escritura
lectura	escritura	lectura	escritura

En los trozos de código de cada pregunta razone sobre los ordenes que deben mantenerse.

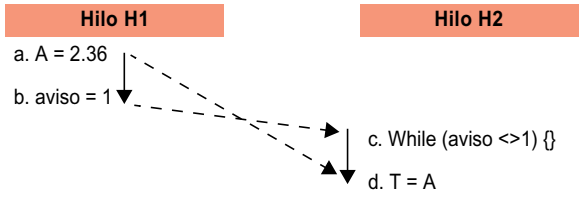
Pregunta 1: *Sincronización punto a punto mediante eventos. La variable aviso ha sido inicializada a cero.*

Hilo H1	Hilo H2
A = 2.36	While (aviso <>1) {}
aviso = 1	T = A

Respuesta: Deben mantenerse los siguientes dos ordenes.

lectura	escritura
lectura	escritura

El primero de ellos en el hilo H2 y el segundo en el hilo H1. El orden de programa determina a -> b (E-E) y c -> d (L-L). La ordenación b -> c implica la ordenación a -> d.



Pregunta 2: *Sincronización de tres hilos mediante eventos. Las variables A y B han sido inicializadas a cero.*

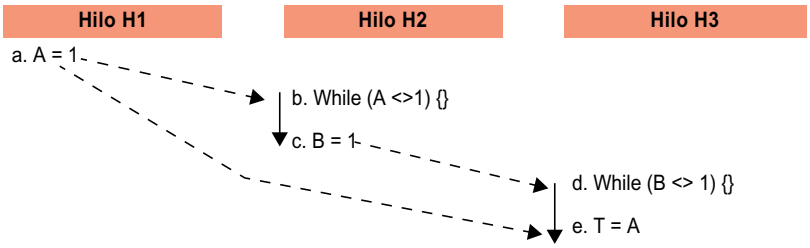
Hilo H1	Hilo H2	Hilo H3
A = 1	While (A <>1) {}	While (B <> 1) {}
	B = 1	T = A

Respuesta: Debe mantenerse el siguiente orden.

lectura

escritura

Los ordenes de programa determinan b -> c (L-E) y d -> e (L-E). La ordenación a -> b implica la ordenación a -> c. La ordenación c -> d implica la ordenación b -> c -> e. Por tanto se cumple la ordenación a -> e.



Pregunta 3: *Algoritmo de Dekker para exclusión mutua. Las variables aviso1 y aviso2 han sido inicializadas a cero. La variable turno ha sido inicializada a uno.*

	Hilo H1	Hilo H2
petición	aviso1 = 1	aviso2 = 1
pregunta	While (aviso2 = 1) {	While (aviso1 = 1) {
decisión si hay	While (turno <> 1) {	While (turno <> 2) {
competencia	aviso1 = 0	aviso2 = 0
	}	}
petición	aviso1=1	aviso2=1
	}	}
exclusión	Acceso exclusivo	Acceso exclusivo
cambio de turno	turno =2	turno = 1
liberación	aviso1 = 0	aviso2 = 0

En este algoritmo se pone a uno una variable (aviso1 o aviso2), para indicar que el hilo quiere conseguir acceso exclusivo. Un hilo, antes de entrar, comprueba si el otro hilo también pretende entrar. En caso de que observe que el otro hilo quiere entrar y no sea su turno declina de su pretensión y lo vuelve a intentar posteriormente.

Respuesta: En el algoritmo de Dekker debe mantenerse el orden.

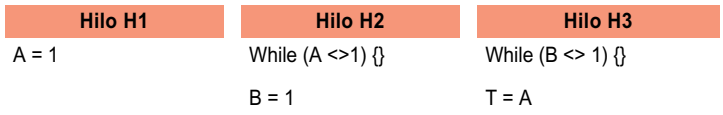
escritura

lectura

El orden de programa determina a -> b (E-L) y c -> d (E-L). Las ordenaciones a -> d y c -> b implican tomar un camino u otro después de d o b.

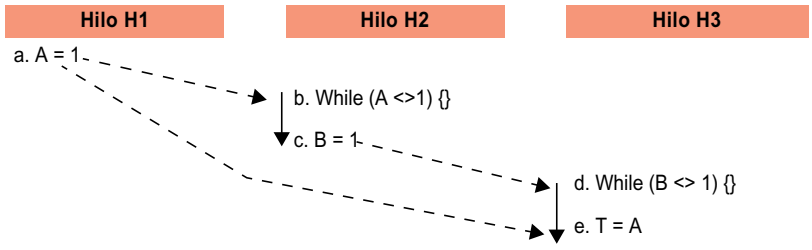
Atomicidad de las escrituras

Sincronización de tres hilos mediante eventos. Las variables A y B han sido inicializadas a cero.



Pregunta 1: *Justifique la necesidad de atomicidad en las escrituras utilizando el ejemplo.*

Respuesta: Los ordenes de programa determinan b -> c y d -> e. La ordenación a -> b implica la ordenación a -> c. La ordenación c -> d implica la ordenación b -> c -> e. Por tanto se cumple la ordenación a -> b -> e. Las sentencias b y e leen la misma variable. Entonces, el valor debe ser el mismo.



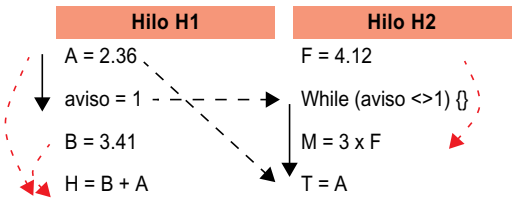
Semántica que espera el programador

Muestre, utilizando los siguientes trozos de código, que las condiciones enumeradas para disponer de consistencia secuencial son más restrictivas que las necesarias para mantener la semántica que espera el programador. Para ello, razone sobre si se pueden reescribir los trozos de código y seguir efectuando el mismo cálculo.

Pregunta 1: *Las variables B, H, F, T Y M están en el espacio de direcciones compartido pero son accedidas exclusivamente por uno sólo de los procesos.*

Hilo H1	Hilo H2
B = 3.41	F = 4.12
A = 2.36	While (flag <>1) {}
flag = 1	T = A
H = B + A	M = 3 x F

Respuesta: Las únicas variables involucradas en la comunicación y sincronización entre hilos son las variables A y aviso. Las otras variables no se comparten. Por tanto, se puede modificar el orden mientras se respeten las dependencias de datos dentro de cada hilo. Una posible ordenación que efectúa el mismo cálculo es la siguiente.



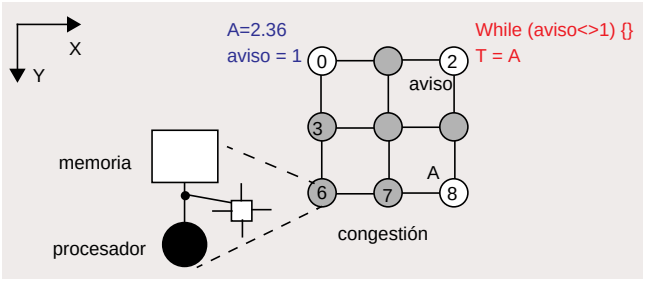
Las variables T y M pueden considerarse locales.

Por tanto, las condiciones enumeradas para consistencia secuencial son suficientes, pero no son necesarias para mantener la semántica que espera el programador.

Tiempo de ejecución

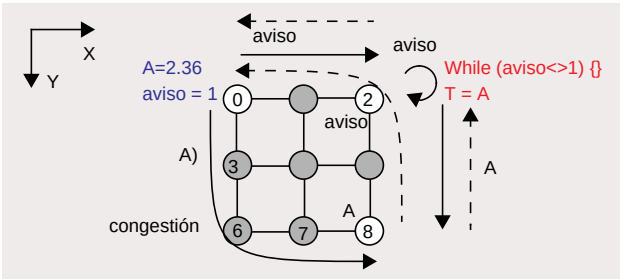
Suponga un sistema multiprocesador donde la red de interconexión es una malla. Un acceso se encamina en primer lugar siguiendo la dirección Y y posteriormente se sigue por la dirección X.

En los procesadores de los nodos cero y 2 se ejecutan los trozos de código mostrados. Los dos hilos pertenecen a un programa paralelo. Suponga que el acceso a una memoria local tarda 1 ciclo y que, al efectuar un acceso a un módulo de memoria no local, la transmisión de información entre dos nodos adyacentes también tarda 1 ciclo. La confirmación en una instrucción store tarda en generarse 1 ciclo y se efectúa en paralelo con la actualización de memoria. Suponga que en el camino de ida, en el acceso a memoria relativo a la variable A, hay congestión y se tardan 2 ciclos más de lo imprescindible.



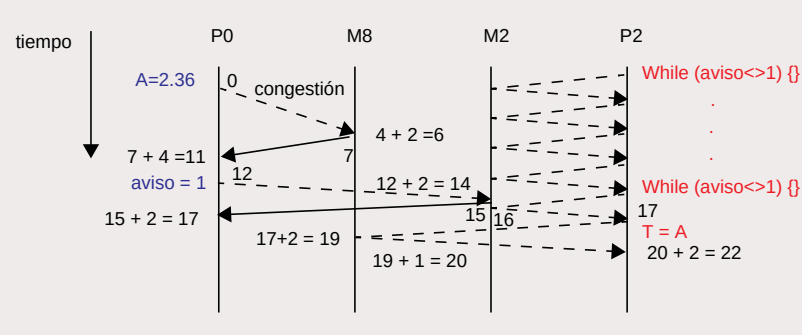
Pregunta 1: *Calcule el tiempo de ejecución.*

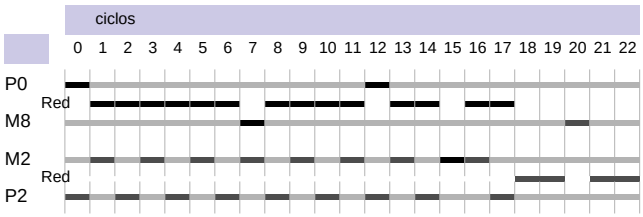
Respuesta: En el siguiente diagrama se muestran el recorrido de los accesos en la ida y en la vuelta. La línea continua indica ida y la línea discontinua indica vuelta.



En la siguiente figura se muestra el cálculo de los tiempos.

En el camino de ida de la instrucción store A se tardan 4 saltos (4 comunicaciones entre nodos adyacentes) más 2 ciclos debidos a la congestión del camino. Por tanto, el acceso llega a la memoria del nodo 8 en el ciclo 6, si empezamos a contar desde el ciclo 0. En el camino de vuelta no se encuentra congestión y son 4 saltos, lo que equivale a 4 ciclos. Entonces la confirmación llega al procesador en el ciclo 11, ya que ha salido en el ciclo 7.





El acceso a la variable aviso (store aviso) se envia en el ciclo 12. Este acceso llega a la memoria del nodo 2 en el ciclo 14 y la confirmación llega al procesador en el ciclo 17 (15 + 2).

Suponemos que la variable aviso se lee en el ciclo 16, ya que el acceso de actualización de esta variable ha llegado a memoria en el ciclo 14 y el ciclo 15 se utiliza para actualizar memoria. Como es un acceso a memoria local el procesador efectúa el siguiente acceso (load A) en el siguiente ciclo (17). Este acceso llega a la memoria del nodo 8 en el ciclo 19 y después de leer (1 ciclo) se envía el dato al procesador. El dato llega al procesador en el ciclo 22.

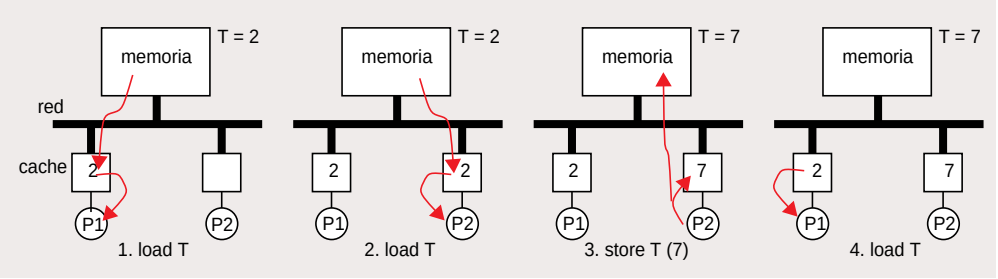
Coherencia de cache

Un multiprocesador dispone de dos procesadores. Las caches privadas utilizan escritura inmediata para mantener la coherencia. Suponga que se está ejecutando un proceso que migra entre los procesadores.

Pregunta 1: *Muestre que en este multiprocesador los resultados que se calculan pueden ser incorrectos.*

Respuesta: Supongamos que la variable T no está almacenada en ninguna cache inicialmente. El proceso lee la variable T cuando se ejecuta en el procesador P1 y se produce un fallo de cache. El bloque que contiene la variable se trae a un contenedor de su cache. Posteriormente el proceso migra al procesador P2 y lee la variable T. Se produce un fallo y el bloque que contiene la variable se almacena en un contenedor de su cache. Ahora hay 2 copias del bloque en las caches y son idénticas. Seguidamente, mientras el proceso se ejecuta en el procesador P2 se escribe la variable T. Como se utiliza escritura inmediata se actualiza la cache y memoria. Ahora las copias ya no son idénticas.

Después de un intervalo de tiempo el proceso migra al procesador P1 y lee la variable T. La variable T está almacenada en un contenedor de su cache y por tanto es un acierto. Ahora bien, el valor de la variable T no es el último valor que ha escrito el proceso en la variable T. Este valor está almacenado en la copia residente en la cache del procesador P2 o en memoria. Por tanto, la ejecución será incorrecta, ya que se lee un valor antiguo.



En la siguiente tabla se muestra en cada fila una fotografía de las caches y memoria en cada acceso a memoria. Las filas están en orden temporal de arriba a abajo, representado en la primera fila el primer acceso.

Actividad del procesador	Actividad de la red	Cache de P1	Cache de P2	Memoria	Coherencia en copias
				2	
1. P1 load T	fallo	2	no hay copia	2	coherente
2. P2 load T	fallo	2	2	2	coherente
3. P2 store T	acierto	2	7	7	incoherente
4. P1 load T	acierto	2	7	7	incoherente

APENDICE A: SISTEMA UNIPROCESADOR Y ENTRADA/SALIDA

El problema de coherencia de cache también existe, en sistemas uniprocesador, cuando se efectúa comunicación con el exterior. Las transferencias de datos, debidas a la entrada/salida de información, se efectúan mediante canales de acceso directo a memoria (direct memory access, DMA) o procesadores especializados de entrada/salida (Figura 3.34). Estos dispositivos mueven datos entre la memoria y los dispositivos exteriores, en ambos sentidos, sin intervención del procesador. Cuando un canal de DMA escribe en memoria el procesador puede seguir leyendo un valor antiguo (más viejo), almacenado en cache si no se efectúa ninguna acción. Igualmente, cuando un canal de DMA lee de memoria puede leer un valor antiguo si, se utiliza escritura retardada para mantener la coherencia en la jerarquía y el procesador ha actualizado el dato en cache.

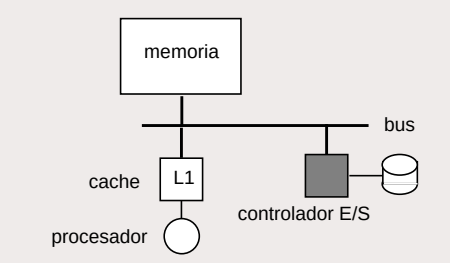


Figura 3.34 Sistema uniprocesador con comunicación con el exterior.

Las operaciones de entrada/salida son menos frecuentes que los accesos a memoria mediante instrucciones load y store desde el procesador. Entonces, pueden adoptarse soluciones más groseras que en un multiprocesador. Por ejemplo, páginas de memoria utilizadas por la entrada/salida se pueden marcar como no cacheables (los bloques de memoria que contienen no se almacenan en cache). También se pueden utilizar instrucciones load y store marcadas de forma que el dato accedido no se almacena en cache. Estas instrucciones se utilizan al comunicarse con los dispositivos de entradas/salida para indicar las operaciones o comprobar si se ha producido algún error. Otra posibilidad para mejorar la eficiencia es involucrar al sistema operativo. El sistema operativo elimina de la cache de los procesadores, usualmente mediante instrucciones específicas, los bloques de las páginas de memoria sobre las que se efectúa la operación de entrada/salida, antes de que se efectúe propiamente la operación de entrada/salida.

EJERCICIOS

Ejercicio 3.1

Considere la ejecución de 2 programas por 2 procesadores con memoria compartida. Suponga que las variables A, B, C, D se inicializan a cero y que la sentencia PRINT imprime los dos argumentos en el mismo ciclo de forma indivisible. Las salidas son 4-tuplas que se obtienen al ejecutar las sentencias c y f. El primer par de valores de la 4-tupla se corresponde con la sentencia que se ejecuta en primer lugar.

P0	P1
a. A = 1	d. C = 1
b. B = 1	e. D = 1
c. PRINT A, D	f. PRINT B,C

Pregunta 1: *Enumere todos los ordenes de ejecuciones entrelazadas de las 6 sentencias que preservan el orden del programa.*

Pregunta 2: *Suponga que el orden del programa se preserva y todos los accesos a memoria son atómicos. Es decir, un store efectuado por un procesador es observado inmediatamente por todos los procesadores restantes. Enumere todas las posibles 4-tuplas de salida.*

Pregunta 3: *Suponga que el orden del programa se preserva pero los accesos a memoria no son atómicos; es decir un store efectuado por un procesador se almacena en un buffer de forma que otros procesadores no pueden observar inmediatamente la actualización. Enumere todas las posibles 4-tuplas de salida.*

Ejercicio 3.2

Considere la ejecución de 3 programas por 3 procesadores con memoria compartida. Suponga que las variables A, B, C se inicializan a cero y que la sentencia PRINT imprime los dos argumentos en el mismo ciclo de forma indivisible. Las salidas son 6-tuplas de vectores de bits, que se obtienen al ejecutar las sentencias b, d y f. El primer par de valores de la 6-tupla se corresponde con la sentencia que se ejecuta en primer lugar y el segundo par de valores con la que se ejecuta en segundo lugar.

P0	P1	P2
a. A =1	c. B=1	e. C = 1
b. PRINT B, C	d. PRINT A,C	f. PRINT A, B

Pregunta 1: *Suponga que todos los procesadores ejecutan las instrucciones en el orden que especifica su programa. Determine que las salidas 001011 y 111111, pertenecen a una ejecución de los programas.*

Pregunta 2: *Suponga que todos los procesadores ejecutan las instrucciones en el orden que especifica su programa. Determine que la salida 000000 no es posible.*

Pregunta 3: *Suponga que todos los procesadores ejecutan las instrucciones en el orden que especifica su programa. Determine que la salida 011001 es posible si procesadores distintos observan los eventos en ordenes distintos.*

Ejercicio 3.3

El siguiente código implementa el algoritmo de Dekker para la exclusión mutua entre dos procesos.

P0	P1	valores iniciales
A =1	B =1	A = B = 0
while (B =1) {. . .};	while (A =1) {. . .};	
. . .	. . .	
R.Critica	R.Critica	
. . .	. . .	
A = 0	B = 0	

Una simplificación del código para efectuar un análisis de consistencia secuencial es

P0	P1	valores iniciales
1. A =1	3. B = 1	A = B = 0
2. x = B	4. y = A	

Pregunta 1: *Muestre que si se cumple consistencia secuencial las variables (x, y) pueden tomar los valores (1,0), (0,1) o (1,1) pero no (0,0).*

Ejercicio 3.4

El siguiente código utiliza una instrucción que efecua las operaciones leer-modificar-escribir de forma indivisible.

P0	P1	valores iniciales
1.1) x = A	2. A = 1	A = 0
1.2) A = x + 2	3. y = A	

Las instrucciones 1.1) y 1.2) se ejecutan de forma indivisible.

Pregunta 1: *Muestre que si se cumple consistencia secuencial las variables (x, y) pueden tomar los valores (0,1), (1,1) o (1,3) pero no (0,2).*

Ejercicio 3.5

Un hilo productor y un hilo consumidor utilizan un buffer circular para comunicarse información. Cada uno de ellos se ejecuta en un procesador distinto. El hilo productor inserta un dato en un buffer compartido si el buffer no esta lleno. El hilo consumidor extrae un dato del buffer si no esta vacio.

El buffer tiene N posiciones de almacenamiento y se implementa mediante un vector compartido y tres variables compartidas denominadas cola, cabeza y cont, las cuales indican respectivamente la próximas posiciones de inclusión y extracción y el número de posiciones ocupadas del buffer.

Seguidamente se muestran tres versiones del trozo de código que nos interesa. La actualización de la variable cont se efectúa de forma atómica mediante una instrucción. En todas las versiones, las variables están inicializadas a cero:

A)	productor	consumidor
	while (cont = N) {};	While (cont = 0) {};
	buffer(cola) = item;	item = buffer(cabeza);
	cola = (cola + 1) mod N;	cabeza = (cabeza + 1) mod N;
	cont = cont +1;	cont = cont - 1;

B)	productor	consumidor
	while (cont = N) {};	While (cont = 0) {};
	cont = cont +1;	cont = cont - 1;
	buffer(cola) = item;	item = buffer(cabeza);
	cola = (cola + 1) mod N;	cabeza = (cabeza + 1) mod N;

C)	productor	consumidor
	while (cont = N) {};	While (cont = 0) {};
	buffer(cola) = item;	item = buffer(cabeza);
	cont = cont +1;	cont = cont - 1;
	cola = (cola + 1) mod N;	cabeza = (cabeza + 1) mod N;

Suponga que tanto el compilador como el hardware garantizan consistencia secuencial.

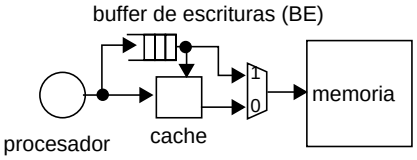
Pregunta 1: *Indique la versiones que funcionan de forma correcta y las que no lo hacen. Justifique la respuesta*

Suponga que el compilador o el hardware no garantizan consistencia secuencial. La necesidad de garantizar consistencia secuencial se puede efectuar mediante la inserción de la directiva LNbarrera.

Pregunta 2: *En las versiones que ha identificado que funcionan correctamente indique la posición en el código en la cual debe incluirse la directiva LNbarrera. Justifique la respuesta.*

Ejercicio 3.6

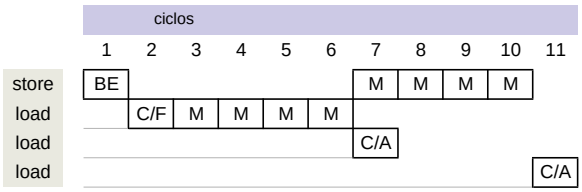
Un procesador utiliza una cache con escritura inmediata y sin asignación de contenedor en un fallo de escritura. Para que las escrituras no bloqueen al procesador se utiliza un buffer denominado de escrituras (BE). En la siguiente figura se muestra el camino de datos cuando se produce un fallo de lectura o una escritura (acierto o fallo). En una escritura la cache se actualiza si se produce un acierto.



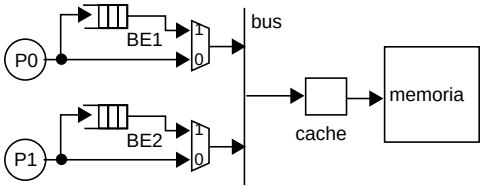
El dato en una instrucción store se almacena en el BE y el procesador sigue interpretando instrucciones (store no bloqueante). En una entrada del BE se distinguen los campos dirección y dato. Una operación store actualiza memoria cuando el camino de acceso está libre. La gestión del BE es FIFO.

Para mejorar el rendimiento se permite que instrucciones load más jóvenes adelanten a operaciones store almacenada en el BE. La justificación es que el procesador necesita los datos para efectuar cálculos y proseguir interpretando instrucciones. En cambio, la actualización determinada por una instrucción store se puede retardar. En el caso de que la dirección a la que accede una instrucción load más joven coincida con una dirección en el BE el dato se suministra desde el BE (esta parte del camino de datos no se muestra en la figura previa).

Si el buffer de escrituras está lleno y se inicia la interpretación de una nueva instrucción store se bloque la interpretación de instrucciones hasta que se ha vaciado el BE. En la figura se muestra un ejemplo donde se observa como una instrucción load adelanta a una instrucción store más vieja. Los acrónimos C, A, F y M indican respectivamente acceso a cache, acierto en cache, fallo en cache y acceso a memoria.



Este procesador se utiliza en el diseño de un multiprocesador que utiliza un bus como red de interconexión y la cache de primer nivel se comparte por todos los procesadores. En estas condiciones en cada procesador pueden competir por el bus la entrada en la cabeza del BE y una instrucción load. En este caso el fallo es prioritario. En la siguiente figura se muestra un esquema de un multiprocesador con dos procesadores.



En este multiprocesador se ejecutan los siguientes dos trozos de código (esqueleto incompleto) que implementan el algoritmo de Dekker para acceso exclusivo.

P0	P1	valores iniciales
aviso1 =1	aviso2 =1	aviso1 = aviso2 = 0
while (aviso2 =1) {...};	while (aviso1 =1) {...};	
. . .	. . .	
R.Critica	R.Critica	
. . .	. . .	
aviso1 = 0	aviso2 = 0	

Una simplificación del código para efectuar un análisis de consistencia secuencial es

P0	P1	valores iniciales
1. aviso1 = 1	3. aviso2 = 1	aviso1 = aviso2 = 0
2. x = aviso2	4. y = aviso1	

Inicialmente, en contenedores de cache están almacenados los bloques que contienen las variables aviso1 y aviso2.

Pregunta 1: *Muestre que este multiprocesador no cumple las condiciones de consistencia secuencial al ejecutarse los códigos previos.*

La necesidad de garantizar consistencia secuencial se puede efectuar mediante la inserción de la directiva LNbarrera.

Pregunta 2: *Indique las posiciones en el código en las cuales debe incluirse la directiva LNbarrera. Justifique la respuesta.*

Un segundo código que se ejecuta en el multiprocesador es el siguiente.

Hilo H1	Hilo H2
A = 2.36	While (aviso <>1) {}
aviso = 1	T = A

Inicialmente los bloques que contienen las variables están almacenados en cache. El valor de los variables es cero.

Pregunta 3: *Justifique si se cumple consistencia secuencial.*

Un cuarto código que se ejecuta en el multiprocesador es el siguiente.

Hilo H1	Hilo H2	Hilo H3
A = 1	While (A <>1) {}	While (B <> 1) {}
	B = 1	T = A

Inicialmente los bloques que contienen las variables están almacenados en cache. El valor de los variables es cero.

Pregunta 4: *Justifique si se cumple consistencia secuencial.*

Finalmente, otro código que se ejecuta en el multiprocesador es el siguiente.

P0	P1	valores iniciales
1. aviso1 = 1	4. aviso2 = 1	aviso1 = aviso2 = 0
2. P = aviso1	5. Q = aviso2	
3. T = aviso2	6. V = aviso1	

Inicialmente los bloques que contienen las variables están almacenados en cache.

Pregunta 5: *Indique los valores que almacenan las variables P, T, Q y V.*

Ejercicio 3.7

En un multiprocesador se permite migración de procesos entre procesadores. Este multiprocesador no dispone de un mecanismo para mantener la coherencia entre cache privadas. Suponga escritura inmediata en las caches privadas.

Pregunta 1: *Muestre que puede producirse un problema de coherencia de cache al ejecutar un programa secuencial. Para ello utilice una traza de accesos a memoria.*

Ejercicio 3.8

Un hilo productor y un hilo consumidor utilizan un buffer circular para comunicarse información. Cada uno de ellos se ejecuta en un procesador distinto. El hilo productor inserta un dato en un buffer compartido si el buffer no esta lleno. El hilo consumidor extrae un dato del buffer si no esta vacio.

El buffer tiene N posiciones de almacenamiento y se implementa mediante un vector compartido y tres variables compartidas denominadas cola, cabeza y cont, las cuales indican respectivamente las posiciones de inclusión y extracción y el número de posiciones ocupadas del buffer.

El código relevante del productor y del consumidor se muestra seguidamente. La variable cont se actualiza mediante una instrucción atómica. Todas las variables están inicializadas a cero.

productor (P)	consumidor (C)
while (cont = N) {};	While (cont = 0) {};
buffer(cola) = item;	item = buffer(cabeza);
cola = (cola + 1) mod N;	cabeza = (cabeza + 1) mod N;
cont = cont +1;	cont = cont - 1;

Suponga que los hilos P y C insertan o extraen K veces consecutivas elementos del buffer. Es decir, el hilo P inserta K elementos y el hilo C no extrae ningún elemento. Posteriormente, el hilo C extrae K elementos y el hilo P no inserta ningún elemento.

Suponga que el tamaño de bloque es exactamente igual a un elemento del buffer, que las variables compartidas ocupan un bloque cada una de ellas y que el tamaño de la cache es infinito. También suponga que todas las posiciones del buffer han sido accedidas al menos una vez por el hilo productor y el hilo consumidor.

Pregunta 1: *Considere un esquema de coherencia de cache con política de invalidación en las escrituras. Calcule el número de fallos de cache y el número de invalidaciones en función del valor de K.*

Pregunta 2: *Considere un esquema de coherencia de cache con política de actualización en las escrituras. Calcule el número de fallos de cache y el número de actualizaciones en función del valor de K.*

Pregunta 3: *Razone, en función de K, el efecto que tiene el tamaño de bloque sobre el número de fallos, número de invalidaciones y el tráfico de bus.*

Ejercicio 3.9

Un algoritmo iterativo para resolver sistemas lineales de ecuaciones es el siguiente.

productor

```
repeat
    X_{i+1} = A X_i + B
until (. . . .)
```

donde X_{i+1}, X_i y B son vectores de tamaño N y A es una matriz de tamaño N x N. El calculo finaliza (acaba de iterar) cuando la diferencia entre X_{i+1} y X_i es menor que un valor predeterminado. La condición de terminación no se ha incluido ya que no es importante para el propósito del problema.

Suponga que cada iteración (X_{i+1}) se calcula utilizando N procesos, donde cada proceso calcula un elemento del vector. El código utilizado es el siguiente.

productor

```
repeat
    doall J =1 , N
         xtemp(J) = B(J)
        do K =1, N
            xtemp(J) = xtemp(J) + A(J,K) * X(K)
        endo
    endoall
    BARRERA
    doall J =1, N
        X(J) = xtemp(J)
    endoall
    BARRERA
until false
```

El bucle doall inicia N procesos. Cada proceso calcula un nuevo valor, el cual se almacena en xtemp. El segundo bucle paralelo copia los elementos del vector xtemp en el vector x. Esta operación requiere una sincronización tipo BARRERA.

Observe que los elementos del vector B y la matriz A solo se leen y los elementos de los vectores X y xtemp se leen y escriben

Suponga que el tamaño del bloque es exactamente igual a un elemento de la matriz o de los vectores y que el tamaño de cache es infinito.

Pregunta 1: *Considere un esquema de coherencia de cache con política de invalidación en las escrituras. Calcule, en el peor caso, el número de fallos de cache y el número de invalidaciones para cada elemento de las estructuras de datos del programa.*

Pregunta 2: *Considere un esquema de coherencia de cache con política de actualización en las escrituras. Calcule el número de fallos de cache y el número de actualizaciones para cada elemento de las estructuras de datos del programa.*

Pregunta 3: *Razone el efecto que tiene el tamaño de bloque sobre la frecuencia de fallo y el tráfico de bus.*

Suponga que el compilador o el hardware no garantizar consistencia secuencial. La necesidad de garantizar consistencia secuencial se puede efectuar mediante la inserción de la directiva LNbarrera.

Pregunta 4: *Indique las posiciones en el código en las cuales debe incluirse la directiva LNbarrera. Justifique la respuesta.*

Ejercicio 3.10

"Seqlock" es un mecanismo de sincronización que permite que un proceso actualice variables compartidas (proceso actualizador) y otros procesos lean estas variables (procesos lectores). El proceso actualizador nunca se espera para efectuar la actualización, mientras que los procesos lectores tiene que esperar si el proceso actualizador está accediendo a las variables.

El mecanismo "seqlock" utiliza una variable compartida para almacenar un número de secuencia (numsec). La variable numsec es actualizada por el proceso actualizador para indicar que está actualizando las variables compartidas. Los procesos lectores utilizan la variable numsec para determinar si los datos leídos de las variables compartidas son consistentes.

El proceso actualizador incrementa la variable numsec antes y después de actualizar las variables compartidas.

Un proceso lector lee el número de secuencia (numsec). Si el número de secuencia es impar el proceso actualizador está modificando las variables compartidas. Si los números de secuencia leídos por el proceso lector antes y después de leer las variables compartidas son distintos, entonces el proceso actualizador ha modificado los datos compartidos mientras se estaban leyendo. El proceso lector debe de volver a efectuar las acciones de leer las variables

compartidas y el número de secuencia antes y después de leer las variables compartidas hasta que los números de secuencia leídos, que deben ser par, sean el mismo.

Los códigos del proceso actualizador y de un proceso lector son los siguientes:

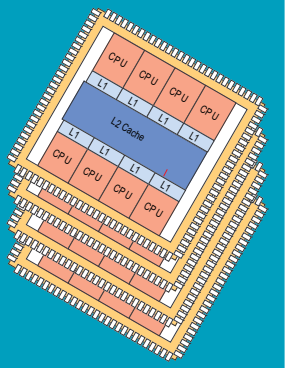
proceso actualizador	proceso lector	inicialización
numsec = numsec +1	repeat	numsec = 0
dato1 = . . .	repeat	numsecini es local
dato2 = ...	numsecini = numsec	
numsec = numsec +1	until (numsecini and 1 = 0)	
	. . . = dato1	
	. . . = dato 2	
	until (numsecini = numsec)	

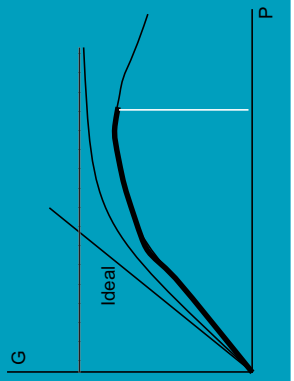
Pregunta 1: *Suponga que los códigos previos se ejecutan en un sistema multi-procesador donde se garantiza consistencia secuencial (compilador, hardware). Indique si las dos siguientes versiones de los procesos actualizador y lector producen los mismos resultados que el código previo. Justifique la respuesta.*

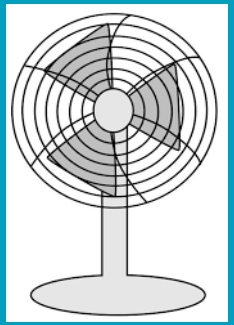
VERSION A			VERSION B	
proceso actualizador	proceso lector	inicialización	proceso actualizador	proceso lector
numsec = numsec +1	repeat	numsec = 0	numsec = numsec +1	repeat
dato2 = . . .	repeat		dato2 = . . .	repeat
dato1 = ...	numsecini = numsec		dato1 = ...	numsecini = numsec
numsec = numsec +1	until (numsecini and 1 = 0)		numsec = numsec +1	until (numsecini and 1 = 0)
	. . . = dato1			. . . = dato2
	. . . = dato2			. . . = dato1
	until (numsecini = numsec)			until (numsecini = numsec)

Los códigos previos se ejecutan en un sistema multiprocesador que no garantiza consistencia secuencial. Para garantizarla utilizaremos la directiva LNbarrera. Al insertar esta directiva entre dos sentencias se inhibe en el compilador la posibilidad de reordenar operaciones de acceso a memoria. El compilador no puede ubicar después de la directiva LNbarrera operaciones de acceso a memoria que el programador ha especificado antes de la directiva LNbarrera y viceversa. Desde el punto de vista hardware la directiva se traduce en una instrucción INbarrera que garantiza que las operaciones de acceso a memoria, previas a la instrucción INbarrera, han sido consolidadas antes de iniciar la ejecución de las operaciones de acceso a memoria especificadas después de la instrucción INbarrera.

Pregunta 2: *Indique las posiciones en el código en las cuales hay que insertar la directiva LNbarrera para garantizar consistencia secuencial. Justifique la respuesta. El número de inserciones de la directiva LNbarrera debe ser el mínimo posible.*







Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 4
Protocolo de observación en una red ordenada

En la Figura 4.1 se muestra un sistema multiprocesador con memoria compartida y procesadores con cache privada. La red de interconexión es un elemento al que están conectadas las caches y la memoria. Tanto las caches como la memoria disponen de un controlador que observa todas las transacciones transmitidas por la red de interconexión. Por tanto, la red de interconexión se puede utilizar como un medio para dar a conocer las intenciones de lecturas y escrituras de un procesador a los otros procesadores y caches asociadas y estas últimas actuar en consecuencia.

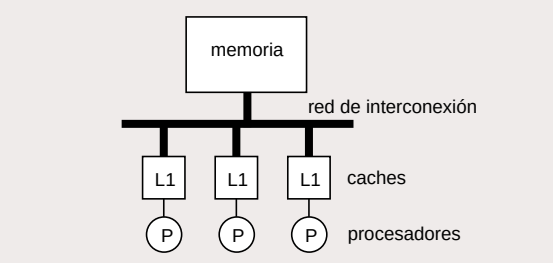


Figura 4.1 *Sistema multiprocesador de memoria compartida.*

En estas condiciones el protocolo de coherencia debe determinar qué operaciones de lectura y escritura deben ser observadas por todos los procesadores, para mantener coherentes las copias de bloques de memoria en las caches privadas de los procesadores. Igualmente, tiene que garantizar que la memoria almacena un copia válida cuando no existe ninguna copia en las caches de los procesadores. Adicionalmente el protocolo debe especificar las acciones que efectúan los controladores de coherencia en cada transacción observada en la red de interconexión.

En este capítulo nos centraremos en protocolos de observación en redes ordenadas, las cuales facilitan la implementación de un protocolo de observación. Como política para mantener la coherencia utilizaremos invalidación [1].

Granularidad de una acción de invalidación. El bloque de memoria.

En particular, en el desarrollo de los conceptos, utilizaremos el bus como red de interconexión ordenada. Esta red es el medio más simple que permite difundir información.

Respecto a las políticas, utilizadas en las caches privadas para mantener coherente la memoria, analizaremos escritura inmediata y escritura retardada.

La descripción de un protocolo se efectúa en dos pasos. En primer lugar se presenta una descripción funcional. Posteriormente se describe una implementación, suponiendo un procesador que ejecutas las instrucciones en orden de programa y la cache es bloqueante[2].

RED ORDENADA

Propiedad de una red ordenada. Una red totalmente ordenada garantiza que todas las peticiones, aunque sean de fuentes distintas o se envíen a destinos distintos, se observen en un orden global (Figura 4.2).

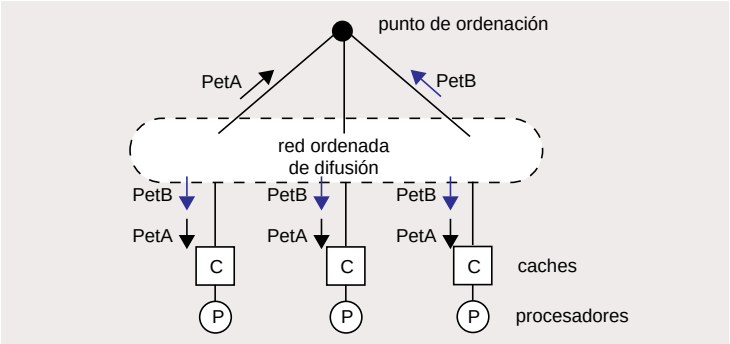


Figura 4.2 Esquema conceptual de una red ordenada.

Para establecer el orden global de las peticiones comúnmente se utiliza un conmutador raíz o un mecanismo de arbitraje, que se denomina punto de ordenación. Las peticiones se transmiten en primer lugar al punto de ordenación, el cual entonces difunde las peticiones a todos los nodos[3] del sistema. La

1. En los ejercicios se enuncian y plantean protocolos de coherencia que utilizan la política de actualización para mantener la coherencia entre caches. También se plantean alternativas, con una política de coherencia de invalidación, que mejoran el rendimiento.
2. En los ejercicios se propone el desarrollo de otras implementaciones que tienen en cuenta procesadores donde se introducen mejoras para incrementar su rendimiento.
3. Elementos conectados a las red: a) par procesador-cache, b) memoria.

red de interconexión tiene que asegurar que el orden en el cual las peticiones dejan el punto de ordenación es el mismo orden en el cual los nodos observan las peticiones.

La implementación de la consistencia y la coherencia de memoria en este capítulo se sustenta en la propiedad de ordenación global de la red.

La implementación de un protocolo de observación en una red ordenada depende del orden lógico y no del tiempo físico en el cual las peticiones son procesadas. El tiempo físico en el cual una petición de observación llega a un nodo no es importante, mientras el orden global en el que todos los nodos del sistema observan una petición sea el mismo. En estas condiciones, un procesador puede inferir el orden en que otro procesador observa las transacciones. Por tanto, no es necesario responder explícitamente las peticiones de invalidación incluidas en una transacción.

Un ejemplo de red ordenada sencilla es un bus (Figura 4.3). Un bus es un conjunto de cables que permite transmitir información (transacción) y que en un instante determinado, es utilizado de forma exclusiva por un nodo para iniciar una transacción. Para ello, se utiliza un árbitro que serializa la utilización del bus por parte de los nodos, para que estos emitan peticiones. Entonces, el árbitro es el punto de ordenación y los nodos observan las peticiones en el orden en el cual el árbitro concede el bus.

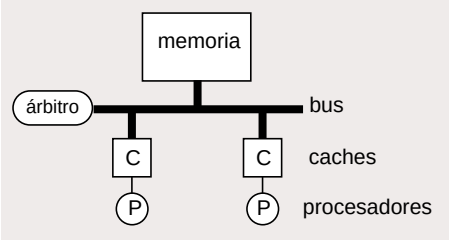
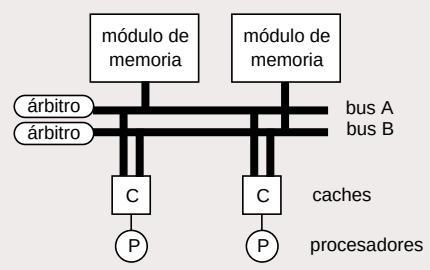


Figura 4.3 Multiprocesador con un bus como red de interconexión.

Ejercicio

Un multiprocesador dispone de una red de interconexión con dos buses. Cada bus permite el acceso a un conjunto disjunto de direcciones, lo cual garantiza estructuralmente que no pueden coexistir dos transacciones al mismo bloque. En cada bus, el punto de ordenación es el árbitro que tiene asociado. Proponga una forma de obtener una ordenación global de las transacciones al ser observadas por los controladores de coherencia (CC).



Respuesta Se consigue un orden global si todos los nodos observan las transacciones transportadas por los buses en el mismo orden. Para ello, se numeran los buses cuando hay transacciones paralelas y todos los CC analizan las transacciones en el mismo orden.

El mecanismo más sencillo es que la numeración sea fija. Por ejemplo, primero el bus A y después el bus B.

PROTOCOLO DE OBSERVACIÓN

En un multiprocesador pueden existir copias de un bloque en varias cache y en cada cache la copia tiene asociado un estado[4], el cual permite inferir los posibles estados de otras copias. El estado de cada copia del bloque está gestionado por un controlador de coherencia (autómata de estados finitos) distinto, aunque el diagrama de transición de estados de un bloque es el mismo para todos los controladores de coherencia.

En un sistema multiprocesador, para mantener la coherencia de las copias de un bloque, hay que conocer las operaciones de acceso a memoria de los otros procesadores sobre copias del mismo bloque. Para ello se utiliza la red de interconexión, la cual es el medio de difusión de las operaciones relevantes para mantener la coherencia de las copias en las caches.

Un protocolo de coherencia de cache de tipo observación es un algoritmo distribuido, implementado por un conjunto de autómatas que cooperan para mantener la coherencia de las copias de los bloques. La coordinación entre los distintos autómatas se efectúa mediante transacciones en la red de interconexión, las cuales son observadas por todos los autómatas de coherencia.

4. Contenido del campo de estado en el contenedor de cache.

Descripción funcional

Una copia de un bloque, almacenada en un contenedor de cache, tiene asociado el estado de compartición del bloque. Podemos decir que la información de estado de un bloque está distribuida. Cada cache tiene información del estado de los bloques que están almacenado en ella y puede inferir, a partir de las transacciones observadas, el estado de las copias del bloque en otras caches.

Cuando un procesador efectúa un acceso a memoria todas las otras caches deben de observar los fallos y las escrituras[5] (propagación de escritura). Para ello las caches son accesibles mediante una red de interconexión que permite difundir información (transacción: petición y respuesta) (Figura 4.4).

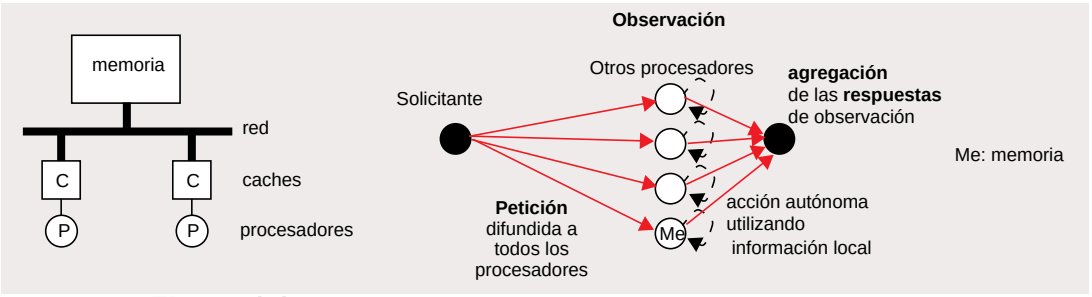


Figura 4.4 Protocolo de observación.

Los controladores de coherencia (CC) monitorizan (observan) la petición de acceso, a una posición o bloque de memoria, transmitida por la red de interconexión y determinan si tienen o no copia del bloque[6].

Una petición de un procesador se difunde utilizando la red y cada CC, de forma autónoma y utilizando información local, realiza las acciones oportunas y responde a la petición. Las respuestas se agregan en una única respuesta, que recibe el CC que ha efectuado la petición. Este CC, a partir de la respuesta, determina el estado de la copia del bloque que almacenará en cache (Figura 4.4).

Cuando un procesador escribe en un bloque, el estado del bloque en las otras caches, que tienen copia, debe modificarse a estado inválido (propagación de una escritura). Notemos que es una forma indirecta de actualizar la copia del bloque. Cuando el procesador, cuya copia ha sido invalidada, vuelva a referenciar el bloque se producirá un fallo y obtendrá una copia actualizada.

5. En ocasiones no es explícitamente necesario. El controlador de coherencia de la cache conoce que no hay más copias del bloque.
6. Si es una posición de memoria, es el bloque que la contiene.

En un protocolo de invalidación, además de invalidar las copias de un bloque que se actualiza, hay que determinar la ubicación del bloque cuando se produce un fallo. En el caso de escritura inmediata la memoria siempre está actualizada y se encarga de suministrar el bloque.

Sin embargo, en el caso de escritura retardada, una cache puede tener la única copia actualizada del bloque (exclusividad). Por tanto, el sistema debe disponer de un mecanismo que suministre esta copia al procesador que la solicita y que también actualice la memoria, si es el caso. En el fase de observación de una transacción una cache reconoce que tiene el bloque en exclusividad y se encargará de suministrarlo.

Recordemos que al utilizar una red de interconexión ordenada, no es necesario responder de forma explícita a la acción de invalidación de un bloque, inducida por la observación de la transacción.

ORGANIZACION DEL MULTIPROCESADOR

En un sistema uniprocesador existe un bus que comunica la cache con la memoria. Cuando se produce un fallo de cache se accede a memoria. Para ello el CC inicia una transacción en el bus. El controlador de memoria (CM) interpreta la petición y efectúa las acciones oportunas: a) suministrar el bloque o actualizar la memoria (Figura 4.5).

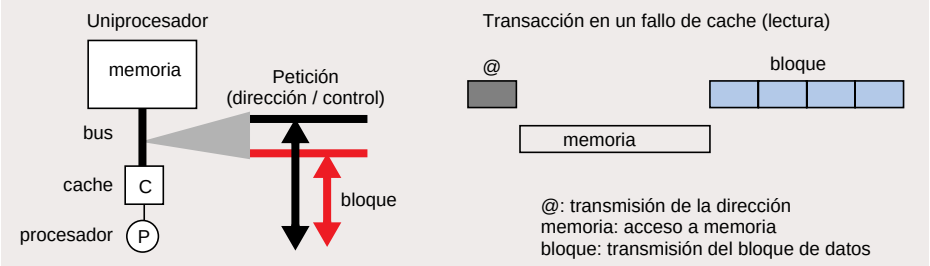


Figura 4.5 Uniprocesador y transacción en un fallo de cache.

En una transacción de bus debido a un fallo de cache distinguimos las siguientes fases: a) transmisión por el bus de la dirección del bloque accedido y del tipo de acceso, además de otras señales de control, b) acceso a memoria y c) transmisión del bloque (Figura 4.5).

Para disponer de un multiprocesador se conectan al bus las caches de otros procesadores, siendo el objetivo acceder a memoria e interrogar el estado del bloque en las caches, cuando el bus transporta una petición correspondiente a una transacción (Figura 4.6).

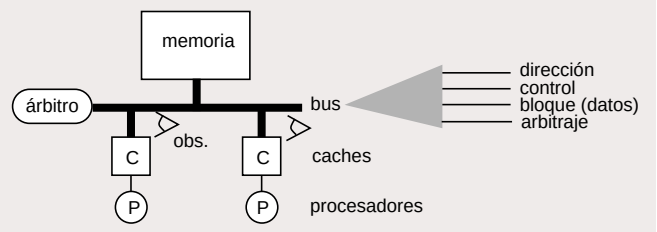


Figura 4.6 Multiprocesador que utiliza un bus como red de interconexión.

Un bus es la red de interconexión más simple y puede considerarse la extensión natural de un sistema uniprocesador, siendo el coste de implementación reducido. La implementación usual de un bus es mediante un conjunto de cables, que transportan señales, a los que se conectan todos los nodos, lo cual facilita la difusión de información[7].

Cuando un CC quiere difundir una petición solicita al árbitro el acceso al bus. En un instante determinado sólo un nodo puede emitir información por el bus. Una vez concedido el bus, este permanece ocupado durante todo el tiempo que dura una transacción (bus atómico). Entonces, el orden en el que los nodos obtienen el bus, para emitir las peticiones (iniciar una transacción), es el mismo orden en el cual los nodos observan las peticiones.

Cada CC monitoriza el bus (observación) para determinar si el bloque, al que se refiere la petición correspondiente a la transacción, está almacenado en un contenedor de la cache que gestiona. El resultado de la observación en cada cache se agrega y se transmite por el bus, dentro del intervalo de tiempo que dura la transacción cuya petición han observado.

En estas condiciones, el número de eventos que debe gestionar un CC se incrementa respecto del caso de un sistema uniprocesador. Además de las peticiones del procesador, un CC debe gestionar las operaciones de memoria difundidas por los otros CC mediante transacciones por el bus.

7. Una de las deficiencias del bus es su escalabilidad. El número de elementos que se pueden conectar está limitado por la longitud del bus y la carga eléctrica que representan los nodos que están conectados. Por otro lado, la longitud del bus y el número de nodos determinan la frecuencia de funcionamiento del bus.

Transacción de bus

En un multiprocesador distinguiremos varias fases en una transacción de bus (Figura 4.7). Estas fases se agrupan en dos. Un grupo corresponde a las fases utilizadas para efectuar la petición y el otro grupo corresponde a las fases de la respuesta. En la Figura 4.7 se muestra la ocupación de memoria en los casos de lectura (L) y escritura (E).

El bus es síncrono y todas las transacciones tienen la misma duración a menos que se especifique lo contrario.

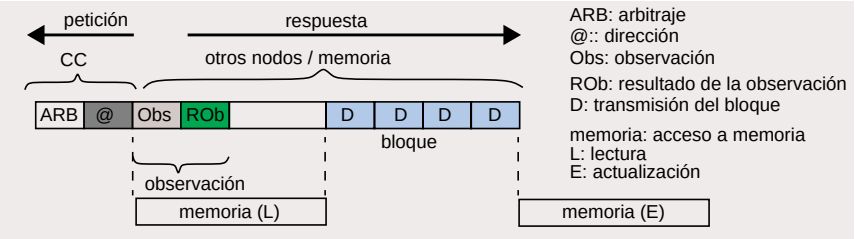


Figura 4.7 Fases en una transacción de bus. La ocupación del bus es independiente del tipo de acceso (lectura o escritura).

En la Figura 4.8 se muestran las señales en el bus. Seguidamente se describe cada una de las fases en una transacción de bus.

- Arbitraje (ARB): los CC solicitan acceso al bus y el árbitro concede el acceso a uno de ellos. Los otros CC deben esperar a que se les conceda el bus en otra fase de arbitraje.
- Difusión de la petición (@): transmisión, por las señales dedicadas en el bus, de la dirección del bloque accedido y del tipo de acceso además de otras señales de control.
- Observación (Obs): los CC comprueban si el bloque, cuya dirección se transmite por el bus, está almacenado en su cache[8].
- Respuesta de la observación (ROb)[9]: los CC emiten una respuesta, si es el caso, en función de si el bloque está almacenado en cache y de su estado. Para ello utilizan las señales del bus dedicadas para este menester.
- Transmisión de datos (bloque): por las señales dedicadas en el bus, para este propósito se transmite: a) el bloque solicitado ó b) el dato o el bloque que se utilizará para actualizar la memoria. La transmisión puede requerir una duración de varios ciclos, en función de la relación en bits entre el tamaño de bloque y el número de cables disponibles[10].

8. En función del tipo de transacción no se efectúa la fase de observación. Por ejemplo, en una petición de actualización de memoria al expulsar un bloque.
9. En función del protocolo de coherencia no existe, en las señales del bus, una fase de respuesta explícita.

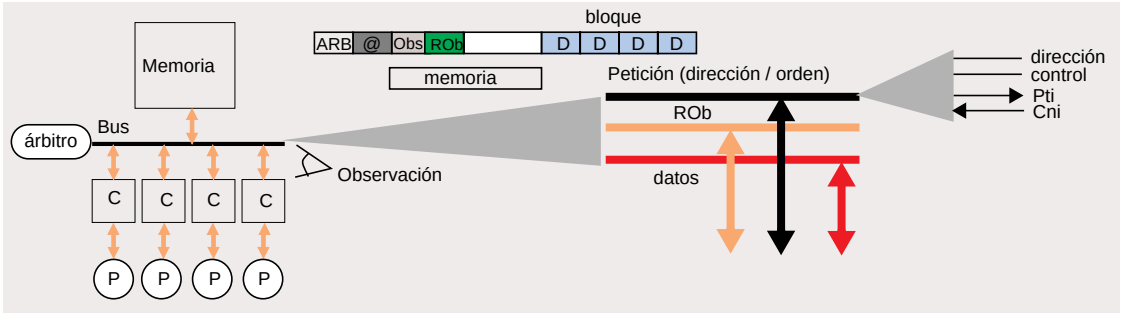
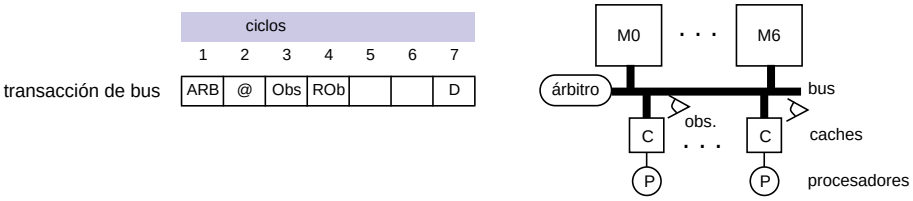


Figura 4.8 Grupos de señales en un bus. Las señales Pti y Cni son respectivamente las señales de petición de bus y concesión de bus. Hay un par de ellas por cada procesador.

En la fase de respuesta un CC confirma la invalidación de forma implícita, si es el caso. Esto es, no existe una señal en el bus para este propósito. Así mismo, la memoria u otro CC suministra el bloque solicitado, en función de la política de actualización de memoria que se utiliza y de la implementación. En el caso de expulsión de un bloque, almacenado en una cache, se actualiza la memoria y la memoria confirma la actualización. La confirmación se supone implícita al finalizar la transacción.

Ejercicio

Un multiprocesador dispone de un bus donde las transacciones se transmiten de forma segmentada. Mediante la segmentación, un bus puede transportar concurrentemente varias transacciones. Todas las transacciones tiene la misma duración. En la siguiente figura se muesta la segmentación de una transacción en siete etapas.



En el bus hay conectados tantos módulos de memoria como número de etapas en que se ha segmentado una transacción. Los módulos de memoria almacenan conjuntos disjuntos de direcciones, lo cual garantiza estructural-mente que no pueden coexistir dos transacciones al mismo bloque. En cada acción de arbitraje se arbitra para acceder a un módulo de memoria que no tiene una transacción en curso. En la siguiente figura se muestra un ejemplo de

10. Suponemos que la ocupación del bus es la misma tanto en acciones de lectura de bloque como en acciones de actualización de dato o bloque en la memoria (Figura 4.8)

utilización del bus. El acceso a un módulo de memoria sólo puede iniciarse si la transacción previa ha finalizado. Esto es, transacciones al mismo módulo de memoria se efectúan de forma serie. En la figura se supone una latencia de iniciación igual a uno. En la parte izquierda se muestra el módulo de memoria accedido en cada transacción.

	ciclos												
Módulo	1	2	3	4	5	6	7	8	9	10	11	12	13
0	ARB	@	Obs	ROb			D						
2		ARB	@	Obs	ROb			D					
1			ARB	@	Obs	ROb			D				
4				ARB	@	Obs	ROb			D			
5					ARB	@	Obs	ROb			D		
3						ARB	@	Obs	ROb			D	
6							ARB	@	Obs	ROb			D

Proponga una forma de obtener una ordenación global de las transacciones al ser observadas por los CC.

Respuesta

La intersección de los conjuntos de direcciones de los módulos de memoria es el conjunto vacio. Los accesos a un módulo de memoria se efectúan de forma serie. En un ciclo determinado se arbitra para acceder a un módulo de memoria desocupado. Todas las transacciones son iguales. La misma fase de distintas transacciones se efectúa en el orden en que se ha accedido al bus. Por ejemplo, la fase Obs de la primera y segunda transacción se efectúan en los ciclos 3 y 4 respectivamente. El orden global se obtiene a partir de las decisiones de arbitraje en cada ciclo. En el ejemplo, el orden global es el acceso a bloques almacenados en los módulos de memoria: 0, 2, 1, 4, 5, 3, 6.

Protocolo de coherencia de cache

El protocolo de coherencia es un algoritmo distribuido formado por un conjunto de autómatas que cooperan. La coordinación entre los autómatas (CC) se efectúa mediante las transacciones de bus.

En la fase de observación (Obs) se utiliza la lógica disponible en el CC para determinar acierto o fallo: utilizando la dirección, incluida en la transacción, se accede a los campos de etiqueta y estado de los contenedores de cache que pueden almacenar el bloque.

En la fase de respuesta de observación (ROb) se actualiza el estado del bloque, si está almacenado en cache y se activan (o no) en consecuencia las señales dedicadas para este menester en el bus.

En las caches distinguimos dos agentes (Figura 4.9):

- Procesador: El agente procesador utiliza los campos de etiqueta y estado para comprobar: a) si el bloque referenciado está almacenado en cache y b) si se puede realizar el acceso a memoria solicitado por el procesador. Si no se puede realizar el tipo de acceso solicitado, efectúa una petición para acceder al bus e iniciar una transacción[11].
- Observador de bus: utiliza la dirección de bloque, incluida en la transacción que transporta el bus, para comprobar si el bloque está almacenado en cache (fase Obs). En respuesta, actualiza el estado del bloque almacenado en cache, emite una respuesta (fase RObs) y en su caso, suministra el bloque (dependiendo de la política de actualización de memoria y el protocolo de coherencia).

Si un bloque no está presente en una cache lo consideraremos como inválido[12].

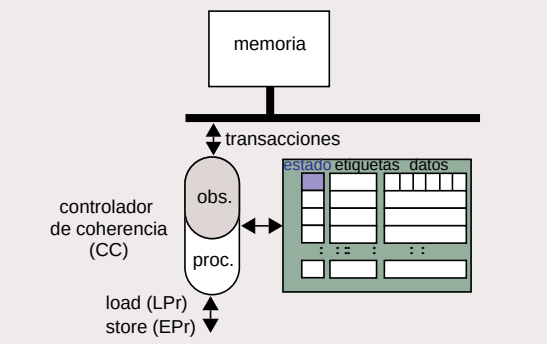


Figura 4.9 Controlador de coherencia: agentes observador y procesador.

Punto de ordenación

El bus es el medio de difusión y el punto de ordenación de los accesos difundidos a memoria. Todas las transacciones (petición-respuesta) dirigidas a memoria se encaminan por este camino común (Figura 4.10).

11. Incluido en el agente procesador está el autómata de reemplazo. Su función es seleccionar un bloque, almacenado en cache, para liberar un contenedor en el cual ubicar el bloque referenciado en una lectura y que no está en cache.
12. Notemos que el campo estado está asociado a un contenedor de cache. Por tanto, en una cache sólo se dispone del estado de bloques almacenados en la misma.

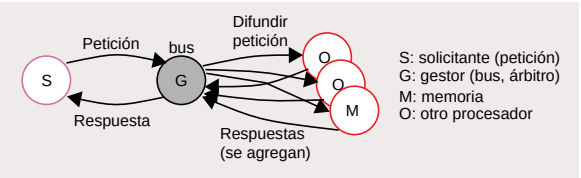


Figura 4.10 Punto de ordenación y fases en una transacción de bus.

Recordemos que el bus, en un instante determinado de tiempo, sólo está ocupado por una transacción. Por tanto, el árbitro que gestiona el acceso al bus establece un orden global entre concesiones del bus. Esta ordenación que determina el árbitro sustenta de forma natural:

- Coherencia (misma posición de memoria): a) propagación a todos los nodos de una escritura cuando esta es difundida por el bus (solicitud de invalidaciones), b) fuerza la serialización de las escrituras difundidas por el bus (arbitraje para ocupar el bus) y c) localización del bloque solicitado.
- Consistencia secuencial: garantiza la atomicidad de todas las escrituras. Serializa todas las escrituras a todas las posiciones de memoria (arbitraje) que hayan sido difundidas. Todos los nodos observan en la misma posición, en el orden global ordenado de transacciones de bus, una transacción de escritura. En una transacción de escritura las caches que tienen copia del bloque invalidan el bloque. En un fallo de cache se obtiene el bloque actualizado por la última escritura a una palabra del bloque[13].

HIPÓTESIS EN LA DESCRIPCIÓN DE UN PROTOCOLO

Un procesador inicia la ejecución de instrucciones en orden de programa y suspende la interpretación de instrucciones cuando detecta un riesgo. Sólo existe un nivel de cache privada, antes de la red de interconexión.

La cache es bloqueante. Esto es, el procesador suspende la interpretación de instrucciones (se bloquea) si es un fallo, no se dispone de los derechos necesarios para acceder a la copia del bloque o la cache utiliza escritura inmediata[14].

13. Ultima escritura consolidada que accede al bloque
14. En algunos ejercicios se relajan algunas de estas hipótesis.

Los procesadores disponen de un mecanismo que garantiza que una operación de acceso a memoria de un procesador no entra en conflicto con ninguna otra operación de acceso a memoria de otros procesadores. Esto es, un acceso a memoria es atómico: el acceso a la cache y si es necesario, la utilización del bus y la ocupación de memoria son indivisibles (Figura 4.11)[15].

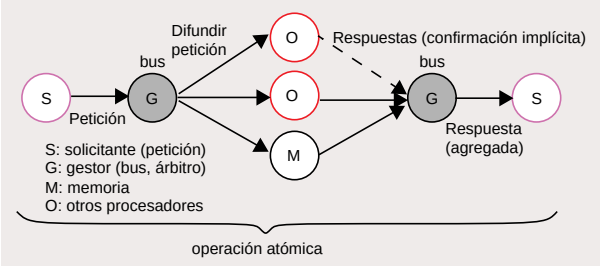


Figura 4.11 Acceso a memoria que requiere una transacción de bus. Operación atómica.

Los procesadores, que no efectúan el acceso a memoria descrito, pueden estar ejecutando instrucciones de cálculo o un acceso a memoria que es acierto en cache. En este último caso, el mecanismo garantiza la gestión de riesgos estructurales cuando hay una transacción en curso en el bus.

En otras palabras, el mecanismo, disponible en cada procesador, garantiza que en un procesador no existe una petición de acceso a memoria pendiente de utilizar el bus si hay una transacción en curso. En la Figura 4.12 se muestra un esquema simplificado.

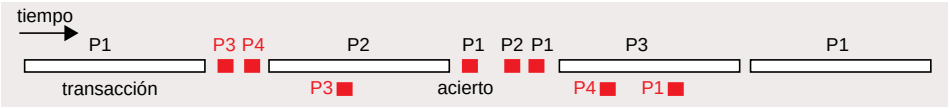


Figura 4.12 Diagrama temporal de accesos a memoria (load o store). Un mecanismo garantiza que sólo existe una transacción en curso en un instante determinado.

PROTOCOLO DE INVALIDACIÓN CON ESCRITURA INMEDIATA (VI)

Para mantener la coherencia de cache en un multiprocesador es suficiente conocer las escritura de los otros procesadores, además de las propias. Por otro lado, cuando se produce un fallo de cache es necesario obtener la copia actualizada del bloque.

15. En el apéndice B de este capítulo se muestra una implementación simple del mecanismo.

En una cache con escritura inmediata todas las escrituras se propagan a memoria. Como la red de interconexión es un bus, todos los nodos observan la propagación. Por tanto, podemos partir del autómata de cambio de estado de un bloque utilizado en un uniprocesador. Es suficiente añadir las transiciones entre estados que tengan en cuenta las operaciones de escritura de los otros procesadores.

Este protocolo lo denominaremos protocolo VI.

Descripción funcional

En un protocolo de invalidación se mantiene conceptualmente un invariante global. Para cada bloque:

- varios procesadores pueden leer el bloque.
- sólo un procesador tiene permiso para escribir[16].

Utilizando escritura inmediata distinguimos dos estados en un bloque almacenado en cache: Inválido (I) y Válido (V). Por otro lado, en un fallo de escritura no se asigna contenedor en cache para almacenar el bloque.

En la descripción que se efectúa del autómata funcional del CC se distingue entre: a) las acciones de acceso a memoria del propio procesador y b) las acciones de acceso a memoria efectuadas por otros CC.

Procesador. Las acciones de un procesador son load (LPr, lectura) y store (EPr, escritura). En una instrucción load, si se produce acierto en cache, el CC suministra el dato leído de la cache. En caso contrario, el CC iniciará las acciones necesarias para obtener el bloque de memoria. En una instrucción store el CC inicia las acciones necesarias para actualizar memoria, lo cual se utiliza para propagar la escritura. Además, si el bloque está almacenado en cache, ésta es actualizada (Figura 4.13)[17].

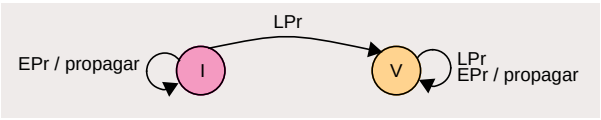


Figura 4.13 Protocolo VI: Transiciones debidas a accesos a memoria del propio procesador.

Otro CC (procesador). Las acciones de lectura (LPr) o escritura (EPr) de otro procesador pueden determinar cambios de estado en un bloque almacenado en la cache de un procesador. En una operación de lectura de otro procesador no existe cambio de estado. En cambio, en una operación de

16. Cuando las caches utilizan escritura inmediata es el procesador cuyo CC obtiene acceso al bus.
17. El permiso de escritura se obtiene al difundir la escritura y consolidar la misma.

escritura de otro procesador, si el bloque está almacenado en cache, el estado cambia de válido (V) a inválido (I) (Figura 4.14). Cuando el bloque no está almacenado en cache consideramos que el estado es inválido.

Figura 4.14 Protocolo VI. Transiciones debidas a accesos a memoria de otros procesadores.

Notemos que la granularidad de las operaciones en el caso de fallo de lectura y en el caso de escritura son distintas. En una operación de lectura se lee un bloque de memoria para almacenarlo en cache. En cambio, en una operación de escritura la granularidad, con la que se actualiza memoria, es la palabra[18]. Ahora bien, la posible invalidación inducida por una escritura tiene la granularidad de bloque.

Para describir los posibles estados de un bloque en las caches y memoria utilizaremos una tabla (Tabla 4.1). Dado el estado de un bloque en una cache, se enumeran los posibles estados en otras cache, aunque estos últimos sean incompatibles entre si. En la primera columna se indica el estado del bloque en una cache. En la segunda columna se indican los posibles estados en otras cache. En la tercera columna se indica si el bloque es válido o no en memoria.

Estado	Posible estado en otras cache	Memoria
I	I, V	válido
V	I, V	válido

Tabla 4.1 Protocolo VI. Descripción de los posibles estados de un bloque en las caches y memoria.

En el protocolo VI pueden existir varias copias de un bloque en las cache de los procesadores, pero cuando se observa una escritura se invalidan todas las copias, excepto, si existe, la copia del procesador que efectúa la escritura. La memoria siempre tiene el bloque actualizado.

Modificaciones al autómata del uniprocesador. Observemos que el autómata (agente procesador, Figura 4.13) que atiende las acciones del procesador es idéntico al utilizado en un sistema uniprocesador. Para la acción de propagación de una escritura se utiliza la transacción que actualiza memoria.

18. Suponemos que es el menor tamaño de datos que se actualiza con una instrucción store

Respuesta después de una observación. En este protocolo la única respuesta del agente observador, a una acción de observación, es que ha realizado la invalidación, si es el caso. Sin embargo, como la red de interconexión es ordenada no es necesario emitir la respuesta (Figura 4.14). Está implícita en la transacción (acción de observación).

La memoria tampoco emite una respuesta de actualización al finalizar la transacción. Está implícita en la finalización de la transacción.

Camino de datos

En la Figura 4.15 se muestra un esquema simplificado del camino de datos de la cache. En ambos lados de la cache, para reducir los cruces en el trazado, se dibujan las señales pertinentes del bus. Por tanto, la misma señal puede estar representada en ambos lados. Las señales son: a) dirección (DIR) y b) datos (DAT). El rectángulo del dibujo se corresponde con el camino de datos de una cache con escritura inmediata en un uniprocesador.

Para efectuar la acción de observación se añade el multiplexor MO. La salida de este multiplexor es la dirección emitida por el procesador o la dirección que se transmite por el bus. Las señales EST, AF e INV indican respectivamente estado, acierto o fallo e invalidación. Esta última señal es activada por el agente observador cuando observa una escritura y el bloque es válido en la cache (AF = EST = 1). No se muestran las señales de control del camino de datos.

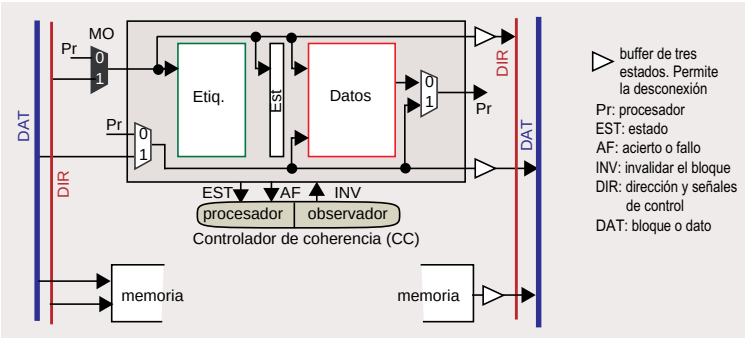


Figura 4.15 Camino de datos simplificado de una cache con escritura inmediata.

Eventos y transacciones

Para mantener la coherencia de cache en un multiprocesador, donde las cache utilizan la política de escritura inmediata, es suficiente observar las escrituras por el bus y actuar en consecuencia, ya que cualquier escritura se

transmite por el bus. Un agente observador al observar una escritura, a un bloque que tiene almacenado en su cache, realiza la acción de invalidar el bloque.

En la Tabla 4.2 se describen las peticiones del procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). En caso de acierto de lectura sólo se accede a la cache. En un fallo de lectura y en una escritura se accede a memoria.

Procesador	Controlador de coherencia (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato	dato	Si es un acierto en cache se lee el dato de cache. En caso contrario, antes de suministrar el dato, se inicia una transacción de lectura del bloque (Pt) y se asigna un contenedor para almacenar el bloque.
EPr (store): escritura de un dato	confirmación de la escritura (implícita al finalizar la transacción)	En cualquier caso se actualiza la memoria (PtE). Si es un acierto y el estado del bloque es válido se escribe el dato en cache al finalizar la transacción. No se asigna un contenedor en caso de fallo.

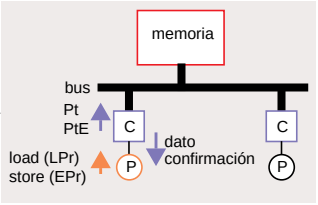


Tabla 4.2 Protocolo VI. Peticiones del procesadores y respuestas.

En la Tabla 4.3 se describen las peticiones del CC a memoria y las acciones del CC para gestionar la ubicación de bloques en la cache (conflictos, CcRe). Se distinguen las acciones de petición de lectura de bloque (Pt) y petición de escritura de dato (PtE). También se describe la acción de expulsión de un bloque debido a un conflicto.

Controlador de coherencia (CC)	Memoria / otros CC	Comentario
Peticiones a memoria y acciones	**Respuestas**	
Pt: petición de lectura de un bloque	bloque de datos	Se lee el bloque de datos de memoria y se suministra.
PtE: petición de escritura de un dato	confirmación de la consolidación de la escritura de forma implícita (un CC invalida el bloque si tiene copia)	Se actualiza la memoria con el dato
CcRe: expulsión de un bloque.		Se invalida la información del contenedor. No se notifica a memoria, ya que está actualizada

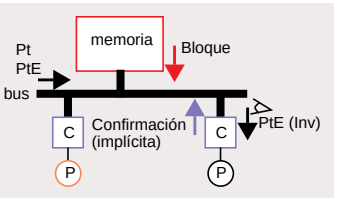


Tabla 4.3 Protocolo VI. Peticiones del CC a memoria, respuestas de memoria y de los otros CC y acciones del CC.

Estados y transiciones

La memoria siempre está actualizada y los posibles estados en un contenedor de cache son: a) Inválido (I) y Válido (V). En el estado inválido el contenedor almacena un bloque no válido, mientras que en el estado V el bloque es válido.

En la descripción de las transiciones entre estados distinguiremos explícitamente las que son debidas a las peticiones del procesador (agente procesador) y las inducidas por una acción de observación (agente observador). En primer lugar describimos las transiciones utilizando diagramas de estado. Posteriormente utilizaremos una tabla para representar los estados, eventos y transiciones.

Al suponer que las operaciones de memoria son atómicas, en una transición entre estados se identifica el evento fuente que inicia la transición y la correspondiente petición en el bus, si es el caso. Cuando el evento fuente es una transacción indicaremos la petición, y no se especifica la respuesta del CC, ya que en este protocolo, está implícita cuando finaliza la transacción.

Eventos del procesador y reemplazo

En una operación de lectura (LPr) el estado final es siempre válido (V, Figura 4.16). Si se ha detectado un fallo hay que iniciar previamente una petición de bloque (Pt).

En una operación de escritura (EPr) el estado final es el mismo del que se parte. Por tanto, si es un acierto se actualiza la cache. En cualquier caso se actualiza la memoria.

Cuando hay que ubicar un bloque en cache, debido a un fallo, y el contenedor está ocupado, se invalida el contenido antes de ubicar el bloque leído de memoria.

En una transacción de bus Pt la respuesta es el bloque, que suministra memoria. En una transacción PtE la respuesta es la confirmación de que se ha actualizado memoria y la escritura ha consolidado. Estas confirmaciones están implícitas al finalizar la transacción.

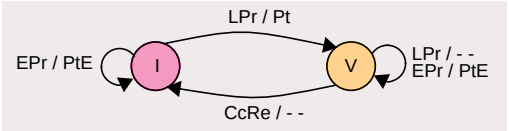


Figura 4.16 Protocolo VI. Transiciones entre estados inducidas por peticiones del procesador.

Eventos externos: acciones inducidas por observaciones

Una observación de una petición de escritura (PtE), por parte de un agente observador, siempre conduce a que el bloque que contiene la palabra se invalide (I, Figura 4.17). Esta petición es la propagación de una escritura. Una observación de una petición de lectura no modifica el estado del bloque.

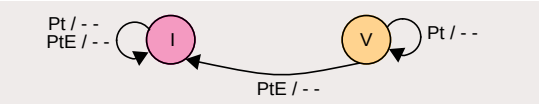


Figura 4.17 Protocolo VI. Transiciones entre estados inducidas por observaciones.

Tabla de estados y transiciones

Las transiciones entre estados también se pueden describir mediante una tabla de transiciones entre estados (Tabla 4.4)[19].

En las filas de la tabla se especifican los estados y en las columnas los eventos. Estos se agrupan en dos conjuntos. Eventos del procesador y reemplazo y eventos externos o transacciones observadas.

En una casilla, que es el cruce de un estado y un evento, se especifica, si es el caso, la acción inducida y el nuevo estado (o estado final). Cuando en un estado determinado no se puede producir un evento la casilla está vacía. Dos guiones indican que no se efectúa ninguna acción.

		Eventos del procesador y reemplazo			Eventos externos (transacciones de bus)	
		LPr	EPr	CcRe	Pt	PtE
Estado	I	Pt; V	PtE; I		- -; I	- -; I
	V	- - ; V	PtE; V	- -; I	- - ; V	- -; I

Tabla 4.4 Protocolo VI. Tabla de transiciones entre estados.

Representación de transacciones y transiciones entre estados

En este apartado se muestran dos formas de representar las transacciones y transiciones al ejecutar una secuencia de accesos a memoria: a) en formato tabla y b) mediante un diagrama temporal. Finalmente se muestra, mediante varios gráficos, una animación.

19. En una tabla se pueden especificar con mayor detalle algunas acciones que complican la comprensión de los diagramas de transición entre estados.

Representación en formato tabla

Utilizaremos una tabla, donde cada fila representa un acceso a memoria y no se gestiona el siguiente acceso hasta que ha finalizado el anterior. En una fila, de izquierda a derecha, después de la instrucción de acceso a memoria[20], se especifica:

1. La transacción de bus (petición), si es el caso.
2. El nombre de la variable y el valor en memoria.
3. Quién suministra el dato o bloque (cache o memoria)
4. Para las cache donde se modifica la información almacenada, el contenedor, el nombre de la variable, el valor y el estado del bloque.

Ejemplo. En la Tabla 4.5 se muestra una secuencia de accesos a memoria realizada por tres procesadores. La variable t no está almacenada en ninguna cache y el valor en memoria es dos.

	bus	mem.			C 1				C 2				C 3			
acceso	trans.	var.	val.	sum.	cont.	var.	val.	est.	cont.	var.	val.	est.	cont.	var.	val.	est.
1. P1 load t	Pt	t	2	mem.	1	t	2	V								
2. P3 load t	Pt	t	2	mem.									1	t	2	V
3. P3 store t (21)	PtE	t	21	mem.	1	t	2	I					1	t	21	V
4. P1 load t	Pt	t	21	mem.	1	t	21	V								
5. P2 store t (8)	PtE	t	8	mem.	1	t	21	I					1	t	21	I

Tabla 4.5 Protocolo VI. Formato tabla: transacciones y cambios de estado del bloque en las caches en una secuencia de accesos a memoria.

Los dos primeros accesos son instrucciones load y producen fallos en cache. El siguiente acceso es un acierto y al ser propagada la escritura: a) se invalida la copia en el otro procesador y b) se actualiza la copia del procesador donde se realiza la escritura (P3) y la memoria. Seguidamente en el procesador P1 se produce un fallo de cache debido a una instrucción load. Finalmente el procesador P2, que no tiene almacenado el bloque en cache, ejecuta una instrucción store. La propagación de la escritura da lugar a que se invaliden las copias en las caches de los otros procesadores y se actualice la memoria.

20. En el caso de una instrucción store se indica el valor con el cual se actualiza la posición de memoria.

Diagrama temporal

En un diagrama temporal se distinguen tres grupos de información. En el primer grupo de filas se representan las transacciones o aciertos en cache si es el caso. Para cada transacción se utiliza una fila y se identifican las fases de una transacción (arb, @, Obs, ROb, espera y D)[21]. Un acierto en cache se representa con una duración de un ciclo y el acrónimo A (Figura 4.18).

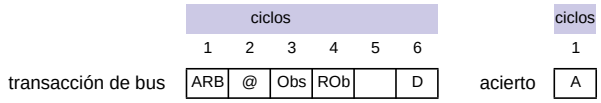


Figura 4.18 Especificación temporal de una transacción y de un acierto en cache.

En el segundo grupo de filas se representan los estados de los bloques en las caches. Cuando no se indica explícitamente el estado, este es el mismo que el último estado indicado en la misma fila. El cambio de estado debido a una transacción se indica en la fase D. El cambio de estado, inducido en la fase de observación de una transacción, en otras cache se representa en la columna correspondiente a la fase ROb de la transacción. Este cambio de estado se indica mediante una línea finalizada con una flecha, que se inicia en la fase ROb y finaliza en la fila donde se representa el estado del bloque en la cache correspondiente.

En el tercer grupo de filas se representan las peticiones que determinan que se inicie una transacción.

Ejemplo. Para la secuencia de accesos a memoria de la Tabla 4.5, en la Figura 4.19 se muestra un diagrama donde se observa la secuencia de transacciones de bus con las fases.

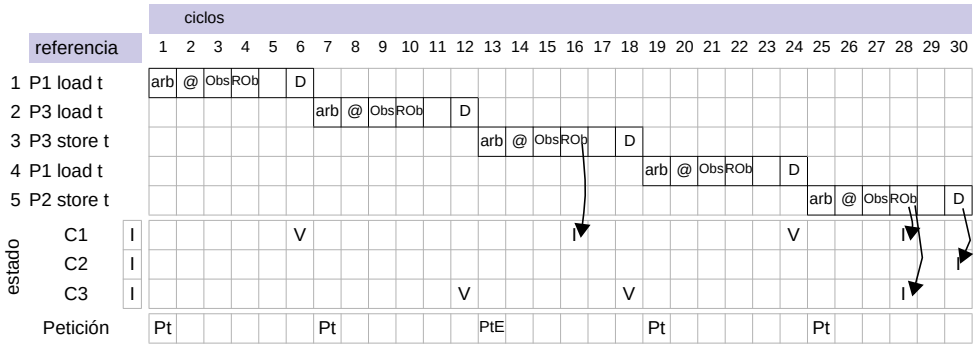


Figura 4.19 Protocolo VI. Diagrama temporal de transacciones y cambios de estado.

21. Aunque se asocian con ciclos no es relevante desde un punto de vista temporal de cada fase.

Animación

En la Figura 4.20 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 4.5. Para cada CC se muestran los estados y las transiciones en cada acceso a memoria.

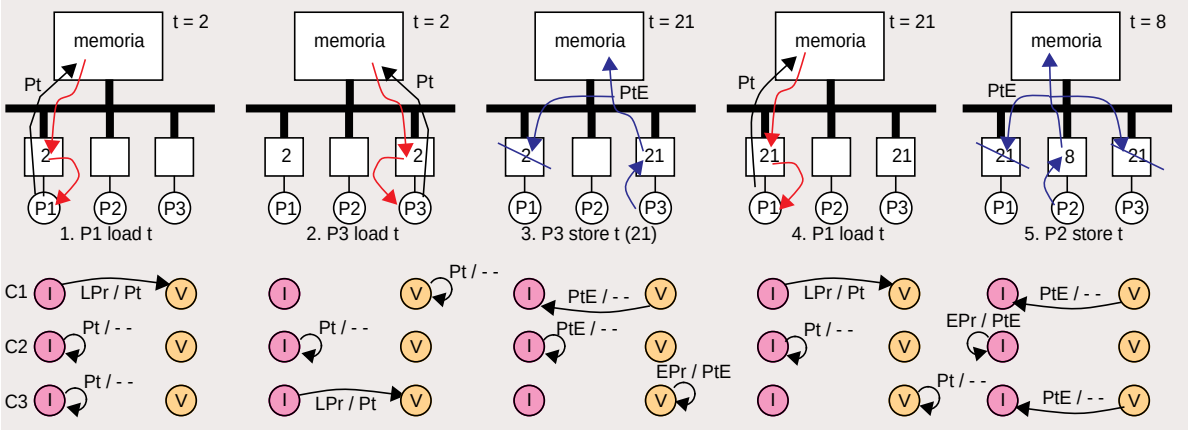


Figura 4.20 Protocolo VI. Fotogramas de las peticiones y respuestas y cambios de estado.

Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido.

Verificación no formal de coherencia y consistencia

En las figuras que se utilizan un acceso de lectura se representa mediante un circulo negro y una escritura mediante un circulo con fondo blanco. Cada uno de los accesos puede requerir efectuar una transacción o no, en función del protocolo de coherencia. El bus está representado por una línea gruesa con fondo gris. Un acierto en cache está representado por el círculo descrito. Una línea quebrada desde el procesador al bus indica que el acceso a memoria produce una transacción de bus.

Coherencia de cache

En la Figura 4.21 se muestra una secuencia de lecturas y escrituras a la misma posición de memoria por parte de tres procesadores. En el inicio de la secuencia se supone que las caches de los procesadores tienen copia en cache del bloque al que se accede.

Orden de programa en accesos a la misma posición de memoria. El procesador efectúa los accesos en el orden determinado por el L.M. El procesador se bloquea en un fallo de lectura o en una escritura, hasta que finaliza la transacción correspondiente[22].

Propagación de escrituras. Al utilizar escritura inmediata todas las escrituras se transmiten por el bus. Una escritura se propaga mediante una transacción y es observada por todos los CC. Las invalidaciones se efectúan durante la transacción y la respuesta de cada invalidación está implícita en la transacción de bus. En la Figura 4.21, la escritura del procesador P3 invalida las copias de P1 y P2.

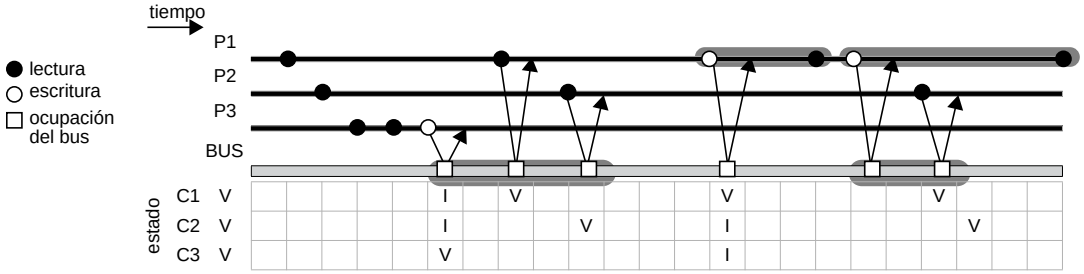


Figura 4.21 Protocolo VI: Verificación no formal de la coherencia de cache. Accesos a la misma posición de memoria.

Serialización de escrituras (atomicidad de una escritura).

- Punto de serialización: El bus es el punto de serialización de las escrituras a la misma posición de memoria[23]. En el bus sólo hay una transacción en curso y es observada por todos los controladores de coherencia. El arbitraje del bus determina el orden de serialización de las transacciones de escritura a una posición de memoria. En la Figura 4.21 las escrituras de P3, P1 y P1, en este orden, están serializadas por el orden en que se les concede el bus.
- Consolidación de una escritura: Una escritura está consolidada al finalizar la transacción. Por otro lado, una escritura es atómica, ya que, una vez finalizada la transacción correspondiente, una instrucción load posterior lee el valor establecido por la escritura previa en el orden de bus. Las copias del bloque son invalidadas en cada transacción de escritura. Un fallo de lectura lee el valor establecido por la escritura previa en el orden de bus (en Figura 4.21 las lecturas de P1 y P2 después de la escritura de P3). Un acierto de lectura lee el valor obtenido en el

22. Hay que garantizar las dependencias de datos en el hilo que se ejecuta.
23. Aún más, es el punto de serialización de las escrituras y los fallos de lectura a cualquier posición de memoria.

fallo de lectura previo (en la Figura 4.21 la tercera lectura de P2) o por la escritura previa, que acierta en cache, del mismo procesador (en la Figura 4.21 la tercera lectura de P1).

Consistencia secuencial de memoria

En la Figura 4.22 se muestran los accesos a memoria efectuados por tres hilos que se sincronizan mediante eventos. En el inicio de la secuencia se supone que las caches de los procesadores tienen copia en cache de los bloque a los que se accede.

Orden de programa. El procesador efectúa los accesos a cualquier posición de memoria en el orden determinado por el L.M.

- Lectura: Espera hasta que se obtiene el dato.
- Escritura: El bus es el punto de ordenación de todas las escrituras. Todas las escrituras requieren una transacción de bus. La finalización de la transacción es una indicación de que la escritura está consolidada[24]. Durante una transacción de bus las caches, que tienen copia del bloque, responden de forma implícita a la petición de invalidación.

Atomicidad de las escrituras. Una escritura siempre produce una transacción de bus.

- Consolidación de una escritura: La finalización de la transacción es una indicación de que la escritura está consolidada.
- Suministro del valor en una lectura: Un fallo de lectura requiere utilizar el bus. El valor devuelto en la transación es el valor establecido en la última escritura consolidada, a la misma posición de memoria (tercera lectura de P2). Un acierto de lectura lee el valor de la copia en cache. Este valor ha sido establecido en el fallo de lectura previo o en la escritura previa, que acierta en cache, del mismo procesador (cinco primeros accesos de lectura de P3).

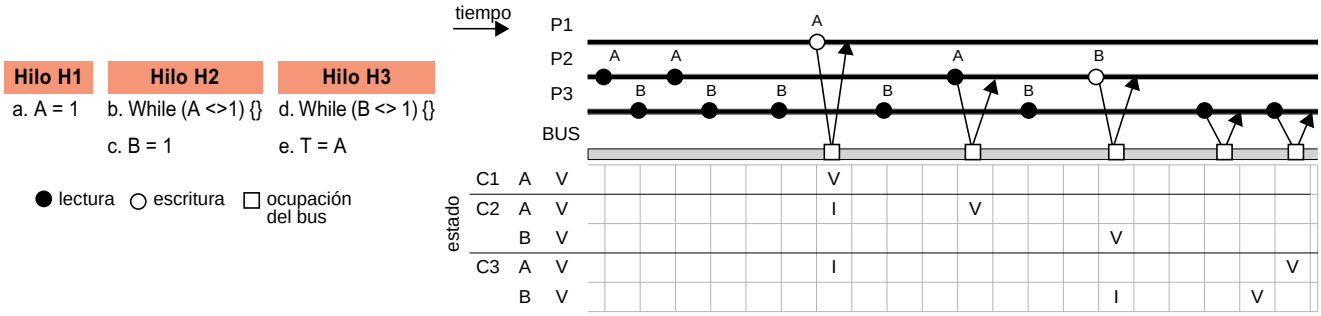


Figura 4.22 Protocolo VI. Verificación no formal de la consistencia de memoria.

24. Esta decisión es conservadora. En este contexto, puede considerarse que una escritura está consolidada cuando se inicia la transacción de bus.

PROTOCOLO DE INVALIDACION CON ESCRITURA RETARDADA (MLI)

Cuando se utiliza escritura inmediata el tráfico de bus generado es proporcional a los fallos de cache y al número de escrituras, ya que todas las escrituras actualizan la memoria.

Como ejemplo, para evaluar el tráfico, supongamos un procesador que funciona con una frecuencia de 1 GHz, tiene un CPI de 1 y un 15% de las instrucciones son stores de 8 bytes. El número de instrucciones store por unidad de tiempo es 0.15 x 1 GHz = 150 x 10^6 stores/s, lo que representa una demanda de ancho de banda (AB) de 1.2 Gbytes/s (150 x 10^6 store/s x 8 bytes/store). Por tanto, si el multiprocesador tiene 4 procesadores la demanda de AB de las instrucciones store es del orden de 5.0 Gbytes/s ((1.2 Gbytes/s) x 4 proc.).

La idea es filtrar el tráfico debido a las escrituras utilizando escritura retardada. En una operación de escritura se asigna un contenedor en cache, el bloque se tiene en exclusividad y las escrituras sólo actualizan cache. La memoria sólo se actualiza cuando se expulsa de cache un bloque que ha sido modificado.

Este protocolo lo denominaremos protocolo MLI.

Descripción funcional

En un protocolo de invalidación se mantiene conceptualmente un invariante global. Para cada bloque

- varios procesadores pueden leer el bloque.
- sólo un procesador tiene permiso para escribir[25].

Utilizando escritura retardada distinguimos tres estados en un bloque almacenado en cache: Inválido (I), Lectura (L) y Modificado (M)[26]. El estado M indica que se tiene copia del bloque en exclusividad para poder modificarla. El estado L sólo permite operaciones de lectura. Por otro lado, en un fallo de escritura se asigna un contenedor en cache para almacenar el bloque.

Procesador. Las acciones de un procesador son load (LPr, lectura) y store (EPr, escritura). En una instrucción load, si se produce acierto en cache, el CC suministra el dato leído de la cache. En caso contrario, el CC iniciará las acciones necesarias para obtener el bloque de memoria. En una instrucción

25. Cuando las caches utilizan escritura retardada es el procesador que tiene acceso al bloque en exclusividad.
26. En inglés las siglas del protocolo son MSI, donde M, S e I indican respectivamente “modified”, “shared” e “Invalid”.

store, si el estado es L o I, el CC iniciará las acciones necesarias para obtener la exclusividad de acceso al bloque y el estado final del bloque será M. Esto es, se propaga la escritura. En caso contrario, el estado es M y se dispone de la exclusividad de acceso al bloque, el CC actualizará el bloque de cache.

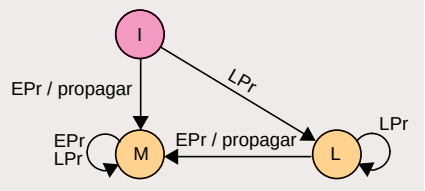


Figura 4.23 Protocolo MLI. Transiciones debidas a accesos a memoria del propio procesador.

Otro CC (procesador.) Las acciones de lectura (LPr) o escritura (EPr) de otro procesador pueden determinar cambios de estado en un bloque almacenado en cache. En una operación de lectura de otro procesador no existe cambio de estado, si el bloque referenciado está almacenado en cache en estado L. En caso contrario, estado M, el protocolo de coherencia debe disponer de un mecanismo para suministrar el bloque y cambiar el estado del bloque a estado L. En una operación de escritura de otro procesador, si el bloque está almacenado en cache, el estado cambia a inválido (I). Además, si el bloque estaba en estado M, el protocolo de coherencia deberá utilizar el mecanismo que permite suministrar el bloque.

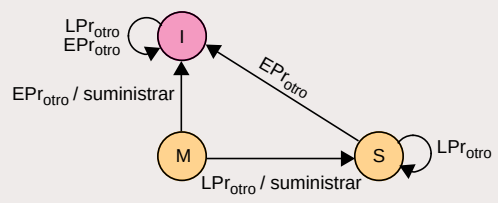


Figura 4.24 Protocolo MLI. Transiciones entre estados inducidas por accesos a memoria de otros procesadores.

Dado el estado de un bloque en un cache, en la Tabla 4.6 se enumeran los posibles estados de un bloque en otras cache, aunque estos últimos sean incompatibles entre si. También se indica la validez del bloque en memoria.

Un bloque puede estar presente (válido, L o M) en cualquiera de las caches y/o en la memoria. Si el bloque está presente en una cache en estado M, esta cache tiene acceso al bloque en exclusividad. Como se utiliza escritura retardada, cuando un bloque está estado M en una cache, la memoria no está actualizada. Al ser expulsado este bloque, debido a una acción de reemplazo, debe actualizarse memoria.

Estado	Posible estado en otras cache	Memoria
I	I, L, M	no se conoce
L	I, L	válido
M	I	inválido

Tabla 4.6 Protocolo MLI. Descripción de los posibles estados de un bloque en las caches y memoria.

Modificaciones al autómata del uniprocesador. Observemos que el autómata (agente procesador) que atiende las acciones del procesador es ligeramente distinto al caso uniprocesador. En concreto, en el caso uniprocesador, una escritura del procesador a un bloque en estado I determina que el CC solicite el bloque a memoria. En el multiprocesador hay que propagar la escritura. En este protocolo la acción de propagar una escritura determina disponer de exclusividad en el acceso al bloque (Figura 4.23). Sólo puede existir esta copia en el multiprocesador y la memoria no estará actualizada. En un uniprocesador, la transición de un bloque del estado L al estado M es con el objetivo de actualizar la memoria cuando se expulse el bloque. En cambio, en un multiprocesador, además de la razón previa y el cambio de estado asociado, es necesario dar a conocer a las otras caches la acción de escritura (propagar la escritura). Entonces, una alternativa cuando el bloque está en estado L y el procesador efectúa una escritura, a una palabra del bloque es actuar como en un fallo de cache.

Respuesta después de una observación. En este protocolo las respuestas de un CC a una acción de observación pueden ser (Figura 4.24): a) se ha efectuado la invalidación, b) el suministro de un bloque de cache y c) ambas. Como se utiliza una red de interconexión ordenada no es necesario emitir una respuesta indicando que se invalida el bloque. Respecto al suministro del bloque desde un CC, el multiprocesador debe disponer de un mecanismo para que la memoria se inhiba. Supondremos que un CC suministra el bloque en los mismos ciclos que los suministraría memoria. En la Figura 4.25 se muestra una transacción donde un CC suministra el bloque. En memoria se inicia un acceso especulativo para obtener el bloque. Después de la fase de observación, el CC que debe suministrar el bloque accede al campo de datos. Finalmente inyecta el bloque en el bus.

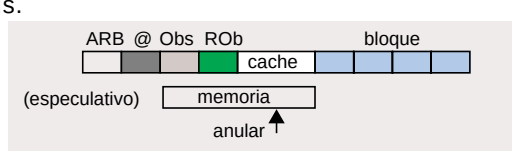


Figura 4.25 Transacción con acceso especulativo a memoria y suministro desde una cache.

Camino de datos

En la Figura 4.26 se muestra un esquema simplificado del camino de datos de una cache con escritura retardada y las conexiones al bus. Las señales en el bus son: a) dirección (DIR), datos en el bus (DAT) y suministro del bloque desde una cache (MOD). Un agente observador activa la señal MOD cuando suministra el bloque en una transacción. Memoria observa la señal MOD. Si la señal MOD está activada, memoria determina que el acceso ha sido especulativo y no suministra el bloque.

Para efectuar la acción de observación se utiliza el multiplexor MO. Las señales EST, AF y ESTn indican respectivamente, estado, acierto o fallo y nuevo estado. El agente observador establece la señal ESTn y el permiso de escritura correspondiente al campo de cache, cuando una transacción referencia un bloque almacenado en cache y es necesario modificar el estado del bloque. Notemos que la señal MOD es una salida de los CC y una entrada en memoria.

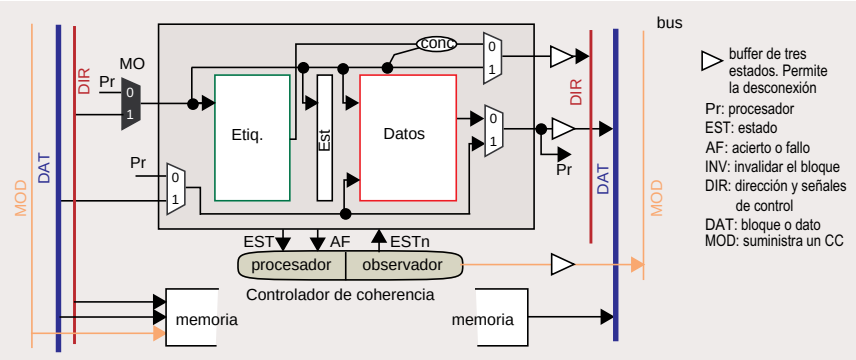


Figura 4.26 Señales de bus y camino de datos de una cache con escritura retardada.

Eventos y transacciones

Para mantener la coherencia en un multiprocesador, donde las cache utilizan la política de escritura retardada, es necesario garantizar la exclusividad de acceso al bloque antes de que un procesador actualice un bloque. Para ello, es necesario disponer de una transacción específica.

Propagación de la escritura. Para obtener un bloque en exclusividad se añade una petición de bloque con intención de modificación: PtIm.

Modificaciones al autómata del uniprocesador. En un acierto de escritura a un bloque en estado L se actúa como si fuera un fallo de cache. Esto es, aunque ya se tenga copia del bloque, se solicita el suministro del bloque, indicando además, que se quiere utilizar en exclusividad.

Respuesta después de una observación. En el bus existe una señal, denominada MOD, que utilizan los CC, para inhibir el suministro del bloque desde memoria (Figura 4.27). Las respuestas de observación de todos los CC se combinan utilizando una OR cableada. La señal MOD es observada por el CM[27]. Cuando la señal MOD está activada la memoria no suministra el bloque, en los ciclos asignados en la transacción, ya que el bloque es suministrado por un CC.

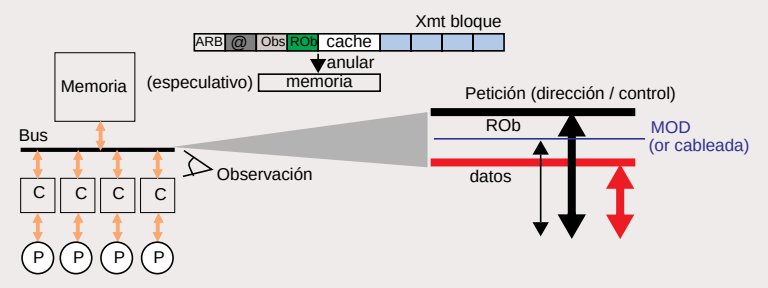


Figura 4.27 Bus: cable de respuesta de observación.

En la Tabla 4.7 se describen las peticiones del procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). En una lectura, si es acierto en cache sólo se accede a cache. En una escritura si es acierto y el estado es M se actualiza el bloque almacenado en cache. En los otros casos se solicita el bloque a memoria y si se pretende actualizar una palabra de bloque, la petición es con intención de modificación.

Procesador	Controlador de cache (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato	dato.	Si es un acierto en cache se lee el dato de cache. En caso contrario se inicia una transacción de lectura de bloque (Pt) y se asigna un contenedor para almacenar el bloque.
EPr (store): escritura de un dato	confirmación de la escritura.	Si es un acierto y el estado del bloque es M se escribe el dato en cache. En caso contrario, en primer lugar, se solicita el bloque con intención de modificarlo (PtIm).

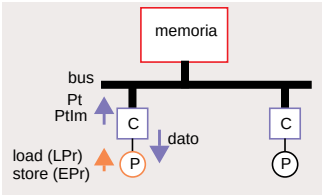


Tabla 4.7 Protocolo MLI. Peticiones del procesador y respuestas.

En la Tabla 4.8 se describen las peticiones y respuestas del CC y acciones del mismo para gestionar la ubicación de bloques en cache. Se distinguen las peticiones: a) lectura de bloque (Pt) y b) lectura de bloque con intención de modificación (PtIm). Las respuestas son: a) suministro del bloque (CaC) y b)

27. La memoria ha iniciado la lectura del bloque de forma especulativa (Figura 4.27)

señal para inhibir el suministro de memoria (MOD). Notemos que en cada CC la confirmación de una acción de invalidación, inducida por la transacción, está implícita al finalizar la transacción. Finalmente se describe la petición asociada con la expulsión de un bloque modificado (PtX) y la acción inducida de actualización de memoria (Dev).

Controlador de cache Peticiones a memoria y acciones del CC y memoria	Memoria / otros CC Respuestas	Comentario
Pt: petición de lectura de un bloque	bloque de datos.	Se obtiene el bloque de datos.
PtIm: petición de lectura de un bloque con intención de modificación	bloque de datos y confirmación de las invalidaciones (implícito al finalizar la transacción).	Se obtiene el bloque de datos en exclusividad.
	CaC: suministro del bloque.	Bloque en estado M. La cache suministra el bloque solicitado en una transacción.
	MOD: bloque en estado M.	Inhibe el suministro desde memoria.
CcRe: expulsión de un bloque		Se invalida la información del contenedor.
PtX: petición de actualización de memoria		Actualización de memoria con un bloque en estado M.
Dev: actualización de memoria		Actualización de memoria en CaC, si es el caso.

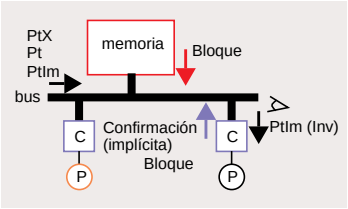


Tabla 4.8 Protocolo MLI. Peticiones de un CC a memoria, respuestas de memoria y de los otros CC y acciones del CC y memoria.

Estados y transiciones

La memoria puede no estar actualizada y los posibles estados en un contenedor de cache son: a) Inválido (I), b) Lectura (L) y c) Modificado (M).

Los estados L y M indican que el bloque almacenado en el contenedor de cache es válido y en particular, en el estado M se dispone de exclusividad en el acceso al bloque, con el objetivo de poder efectuar actualizaciones del mismo.

En la descripción de las transiciones entre estados se utilizan dos diagramas de transiciones. En uno de ellos se muestran las transiciones debidas a peticiones del procesador y en el otro diagrama, las transiciones inducidas por acciones de observación.

Eventos del procesador y reemplazo

En una operación de lectura (LPr, load), si se detecta un fallo de cache, el CC efectúa una petición de lectura de bloque (Pt) y al finalizar la transacción el estado final es L. En caso de acierto de lectura, el estado final es el mismo que el estado inicial (Figura 4.28).

En una operación de escritura (EPr, store) el estado final es siempre M. Si el estado inicial no es M, el CC solicita el bloque con intención de modificación (PtIm). En otras palabras, obtenerlo en exclusividad. Notemos que si el estado inicial es L, aunque ya se tiene copia del bloque, se solicita el bloque para obtener la exclusividad y con ello el permiso de escritura.

En una acción de reemplazo supondremos que en primer lugar se efectúa la expulsión del bloque que ocupa el contenedor y posteriormente se procesa el fallo que produce la expulsión[28]. Una expulsión en estado L no da lugar a ninguna transacción ya que memoria está actualizada. En cambio, una expulsión de un bloque en estado M requiere actualizar memoria (PtX).

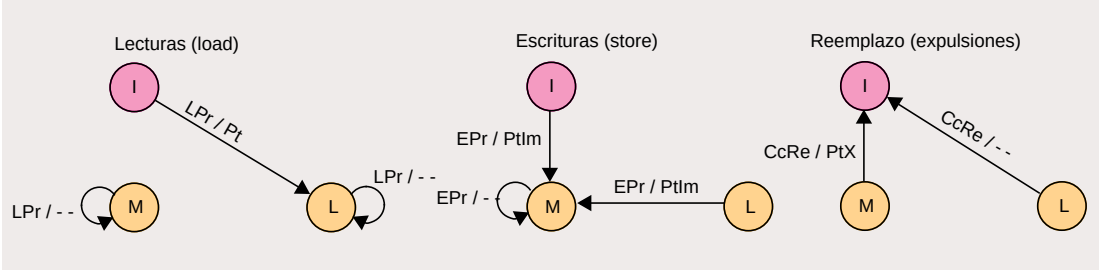


Figura 4.28 Protocolo MLI. Transiciones entre estados debidas a eventos del procesador y acciones de reemplazo.

Eventos externos: acciones inducidas por observaciones

En una observación de una petición de lectura (Pt) un CC suministra el bloque si el estado del bloque es M (Figura 4.29). La respuesta del agente observador es: a) indicación de que una cache suministra el bloque (MOD) y b) suministro del bloque (CaC). En la transición del estado M al estado L indicamos también que la memoria se actualiza (Dev). Una observación de una petición de lectura con intención de modificación (PtIm) requiere invalidar el bloque. Si el bloque está en estado M, el agente observador efectúa las mismas acciones que en la observación de una petición de lectura (Pt) antes de invalidar el bloque.

Notemos que la petición PtX también bien es una transacción. Sin embargo, los CC identifican que la petición PtX es una petición para actualizar memoria y no se efectúan las acciones relacionadas con una operación de observación.

28. El secuenciamiento está gestionado por el controlador de coherencia.

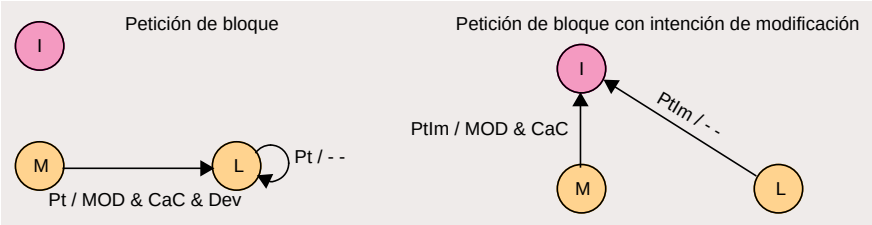


Figura 4.29 Protocolo MLI. Transiciones entre estados inducidas por observaciones.

Tabla de estados y transiciones

En la Tabla 4.9 se describen mediante una tabla los diagramas de transiciones ente estados de la Figura 4.28 y la Figura 4.29.

		Eventos del procesador y reeemplazo		Controlador	Eventos externos (transacciones de bus)	
		LPr	EPr	CcRe	Pt	PtIm
Estado	I	Pt; L	PtIm; M		- - ; I	- - ; I
	L	- - ; L	PtIm; M	- - ; I	- - ; L	- - ; I
	M	- - ; M	- -; M	PtX; I	MOD&CaC&Dev; L	MOD&CaC; I

Tabla 4.9 Protocolo MLI. Tabla de transiciones entre estados.

Representación de transacciones y transiciones entre estados

En este apartado se muestran dos formas de representar las transacciones y las transiciones entre estados al ejecutar una secuencia de accesos a memoria: a) en formato tabla y b) mediante un diagrama temporal. Finalmente se muestra, mediante varios gráficos, una animación.

Representación en formato tabla

Cada fila de la tabla representa un acceso a memoria y no se gestiona el siguiente acceso hasta que ha finalizado el anterior. En una fila, de izquierda a derecha, después de la instrucción de acceso a memoria, se especifica:

1. La transacción de bus y la activación de la señal MOD, si es el caso.
2. El nombre de la variable y el valor en memoria.
3. Quién suministra el dato o bloque (cache o memoria)

4. Para las cache donde se modifica la información almacenada, el contenedor, el nombre de la variable, el valor y el estado del bloque.

Para representar la actualización de la memoria debido a una expulsión de un bloque, inducida por un acceso a memoria, se utilizan dos filas. En la primera fila se representa la acción de expulsión y en la segunda fila la petición que ha determinado la expulsión.

Ejemplo. En la Tabla 4.5 se muestra una secuencia de accesos a memoria realizada por tres procesadores. La variable t no está almacenada en ninguna cache y el valor en memoria es dos.

Los dos primeros accesos son instrucciones load y los CC detectan fallos en cache. El siguiente acceso (P3) es una escritura que acierta en cache, pero es necesario obtener la exclusividad. La transacción de bus, emitida por el CC para obtener la exclusividad, invalida la copia en el otro procesador. Al finalizar la transacción se obtiene la copia del bloque y se actualiza.

	bus		mem.			C 1				C 2				C 3			
acceso	trans.	señal	var.	val.	sum.	cont.	var.	val.	est.	cont.	var.	val.	est.	cont.	var.	val.	est.
1. P1 load t	Pt		t	2	mem.	1	t	2	L								
2. P3 load t	Pt		t	2	mem.									1	t	2	L
3. P3 store t (21)	PtIm		t	2	mem.	1	t	2	I					1	t	21	M
4. P1 load t	Pt	MOD	t	21	C3	1	t	21	L					1	t	21	L
5. P2 store t (8)	PtIm		t	21	mem.	1	t	21	I	1	t	8	M	1	t	21	I

Tabla 4.10 Protocolo MLI. Secuencia de accesos a memoria: transacciones y cambios de estado en las caches.

Seguidamente, en el procesador P1 se produce un fallo de cache, debido a una instrucción load. La cache C3 almacena en su cache el bloque actualizado y es la encargada de suministrar el bloque en los ciclos dedicados de la transacción. Notemos que la memoria no tiene una copia actualizada del bloque y se actualiza cuando el bloque se transfiere por el bus. Finalmente el procesador P2, que no tiene almacenado el bloque en cache, ejecuta una instrucción store. La petición emitida para obtener el bloque indica exclusividad y da lugar a que se invaliden las copias en las caches de los otros procesadores, además de leer el bloque de memoria.

Diagrama temporal

En un diagrama temporal se distinguen tres grupos de información. En el primer grupo de filas se representan las transacciones o aciertos en cache si es el caso. Para cada transacción se utiliza una fila y se identifican las fases de una transacción (arb, @, Obs, ROb, espera y D)[29]. Un acierto en cache se representa con una duración de un ciclo y el acrónimo A (Figura 4.18).

Para representar la actualización de la memoria debido a una expulsión de un bloque, inducida por un acceso a memoria, se utilizan dos filas. En la primera fila se representa la acción de expulsión y en la segunda fila la petición que ha determinado la expulsión. Ahora bien, en un instante determinado sólo hay una transacción en curso.

En el segundo grupo de filas se representan los estados de los bloques en las caches. Cuando no se indica explícitamente el estado, este es el mismo que el último estado indicado en la misma fila. El cambio de estado debido a una transacción se indica en la fase D. El cambio de estado, inducido en la fase de observación de una transacción, en otras cache se representa en la columna correspondiente a la fase ROb de la transacción. Este cambio de estado se indica mediante una línea finalizada con una flecha, que se inicia en la fase ROb y finaliza en la fila donde se representa el estado del bloque en la cache correspondiente.

En el tercer grupo de filas se representan las peticiones que determinan que se inicie una transacción.

Ejemplo. Para la secuencia de accesos a memoria de la Tabla 4.10, en la Figura 4.30 se muestra un diagrama donde se observa la secuencia de transacciones de bus con las fases.

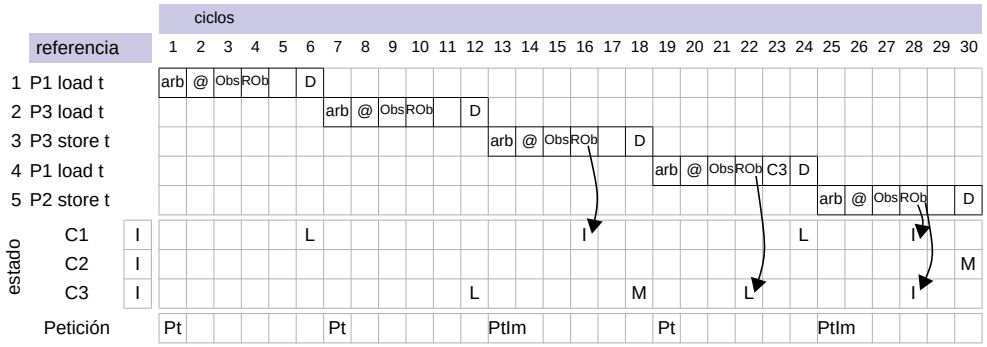


Figura 4.30 Protocolo MLI. Diagrama temporal: transacciones y cambios de estado en una secuencia de accesos a memoria.

29. Aunque se asocian con ciclos no es relevante desde un punto de vista temporal de cada fase.

Animación

En la Figura 4.31 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 4.10.

Para cada CC se muestran los estados y las transiciones en cada acceso a memoria. Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido.

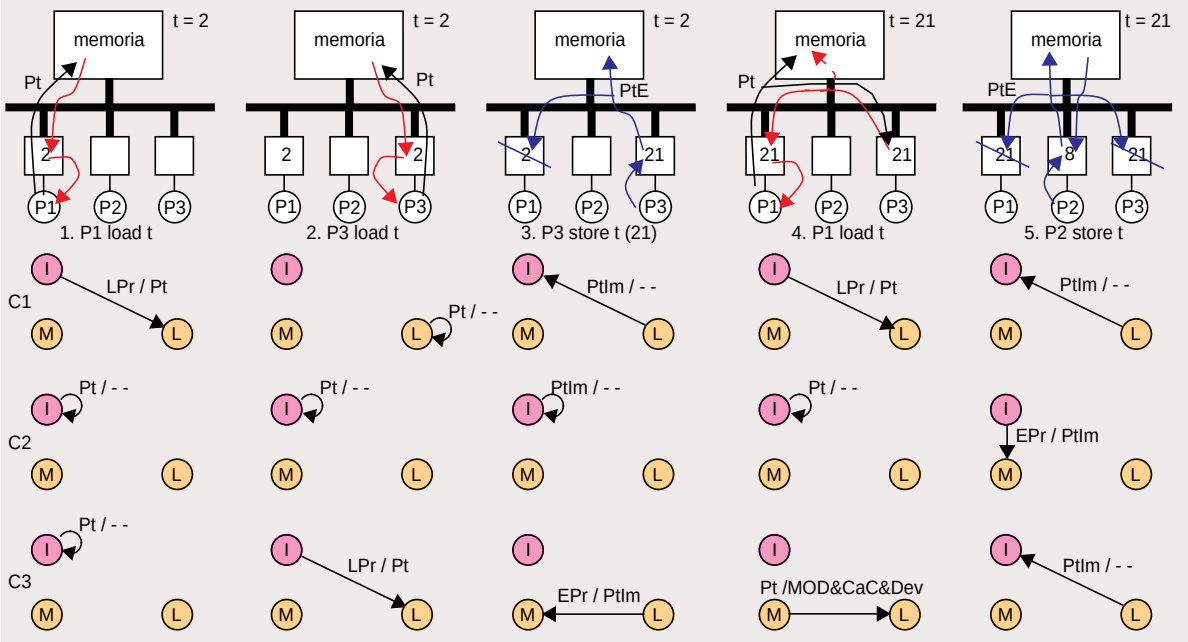


Figura 4.31 Protocolo MLI. Fotogramas de las peticiones y respuestas y cambios de estado en una secuencia de accesos a memoria.

Verificación no formal de coherencia y consistencia

En las figuras que se utilizan un acceso se representa mediante un circulo negro si es una lectura y un circulo con fondo blanco si es una escritura. Cada uno de los accesos puede requerir efectuar una transacción o no en función del protocolo de coherencia. El bus está representado por una línea gruesa con fondo gris. Un acierto en cache está representado por el círculo descrito. Una línea quebrada desde el procesador al bus indica que el acceso a memoria produce una transacción de bus.

Coherencia de cache

En la Figura 4.32 se muestra una secuencia de lecturas y escrituras a la misma posición de memoria por parte de tres procesadores. En el inicio de la secuencia se supone que las caches de los procesadores tienen copia en cache del bloque al que se accede.

Orden de programa en accesos a la misma posición de memoria. El procesador efectúa los accesos en el orden determinado por el L.M. El procesador se bloquea en un fallo de lectura o cuando requiere obtener la exclusividad de acceso al bloque, hasta que finaliza la transacción correspondiente.

Propagación de escrituras. Para efectuar una escritura en un bloque es necesario tener acceso al bloque en exclusividad (estado M). Ello garantiza que no hay copias del bloque. La exclusividad de acceso a un bloque se obtiene mediante una transacción PtIm. Las copias del bloque en otras caches se invalidan durante la transacción PtIm (primera escritura de P3). La respuesta de cada invalidación está implícita en la transacción y se efectúan en orden de bus.

Serialización de escrituras (atomicidad de escritura).

- Punto de serialización. El árbitro del bus es el punto de serialización de las transacciones PtIm. Las escrituras de un procesador, que tiene un bloque en exclusividad, quedan serializadas por el orden de programa. Se tiene garantía de que no existen copias del bloque.
- Consolidación de una escritura. Una transacción PtIm está consolidada al finalizar la transacción[30]. Una escritura a un bloque en exclusividad está consolidada después de actualizar la cache. Por otro lado, una escritura es atómica. Una vez finalizada una transacción PtIm y actualizada la cache, una lectura posterior del bloque (con o sin intención de modificar) obtiene el valor establecido por la escritura previa (segunda lectura de P1). En el lapso de tiempo entre obtener la exclusividad de acceso al bloque y el suministro del bloque, el procesador puede haber actualizado el bloque varias veces (segunda escritura de P1 y cuarta lectura de P2). Una lectura del procesador que tiene el bloque en exclusividad lee un valor consolidado.

30. Esta decisión es conservadora. En este contexto, puede considerarse que una escritura está consolidada cuando se inicia la transacción de bus.

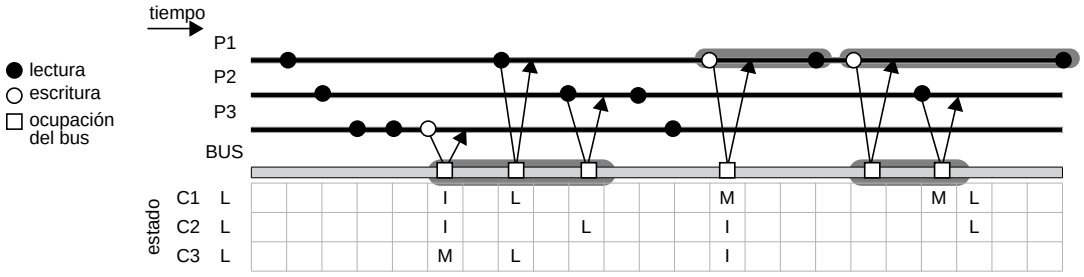


Figura 4.32 Protocolo MLI. Verificación no formal de la coherencia de cache. Accesos a la misma posición de memoria.

Consistencia secuencial de memoria

En la Figura 4.33 se muestran los accesos a memoria efectuados por tres hilos que se sincronizan mediante eventos. En el inicio de la secuencia se supone que las caches de los procesadores tienen copia en cache de los bloque a los que se accede.

Orden de programa. El procesador efectúa los accesos a cualquier posición de memoria en el orden determinado por el L.M.

- Lectura: Espera hasta que se obtiene el dato.
- Escritura: El árbitro del bus es el punto de ordenación de todas las transacciones PtIm. La finalización de la transacción es una indicación de que la escritura está consolidada. Durante la transacción de bus se responde de forma implícita a una posible petición de invalidación (si se tiene copia del bloque). En caso de acierto a un bloque en estado M, la escritura consolida al ser actualizada la cache (existe garantía de que no hay copias del bloque).

Atomicidad de las escrituras.

- Consolidación de una escritura: Para efectuar una escritura es necesario disponer de acceso exclusivo al bloque. El acceso exclusivo garantiza que no hay copias del bloque y se obtiene mediante una transacción PtIm. Una vez obtenida la exclusividad se actualiza la cache y la escritura está consolidada.
- Suministro del valor en una lectura: Un fallo de lectura de un bloque (con o sin intencion de modificarlo) requiere utilizar el bus. El valor devuelto en la transación es el valor establecido en la última escritura consolidada, a la misma posición de memoria (tercera lectura de P2). Un acierto de lectura lee el valor de la copia en cache. Este valor ha sido establecido en un fallo de lectura previo o en una escritura previa del mismo procesador (cinco primeros accesos de lectura de P3).

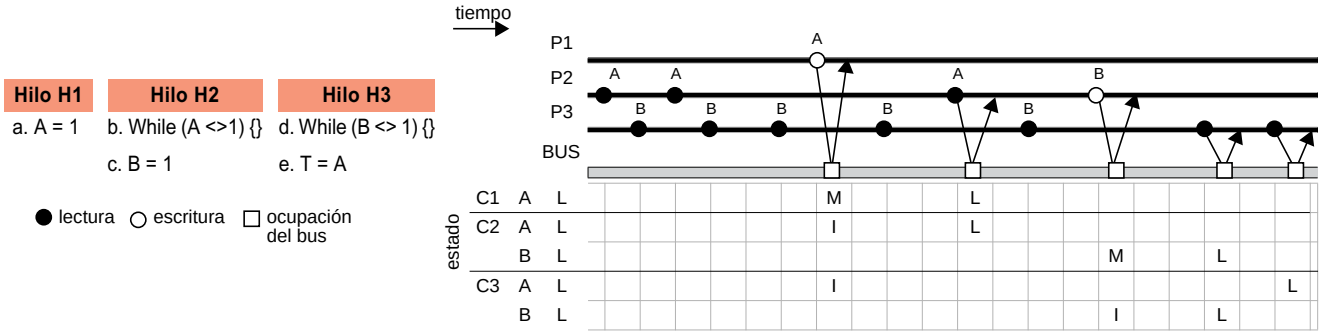


Figura 4.33 Protocolo MLI. Verificación no formal de la consistencia de memoria.

COMPARTICION FALSA

Al ejecutar un programa paralelo puede existir compartición verdadera o compartición falsa. Por compartición verdadera entendemos que varios procesos o hilos acceden a una misma palabra para comunicarse o sincronizarse.

En un protocolo de invalidación existe compartición falsa cuando el tamaño de bloque de memoria es mayor que una palabra. Se produce cuando varios procesos o hilos, que se ejecutan en procesadores distintos, acceden a distintas palabras ubicadas en el mismo bloque y al menos uno de ellos actualiza la palabra a la que accede. Puede decirse que se comparte el bloque, pero no las palabras ubicadas en el bloque. Sin embargo, como el protocolo es del tipo invalidación, cuando un procesador escribe en una palabra invalida las copias del bloque en las otras cache, debido a que la granularidad de la invalidación es el tamaño de bloque. Después de la invalidación, en los procesadores que acceden a las otras palabras se producirá un fallo de cache cuando acceden a una palabra almacenada en el bloque.

Ejemplo Supongamos un bloque de memoria del tamaño de dos palabras que son accedidas desde dos procesadores durante el mismo intervalo de tiempo. Un procesador sólo acede a una palabra. Denominamos A1 y A2 a cada una de las palabras. En la Tabla 4.11 se muestra una secuencia de accesos a las variables A1 y A2 por parte de dos procesadores. Notemos que en la tabla, en la columna correspondiente a variable en cache, se indican las dos palabras contenidas en el bloque.

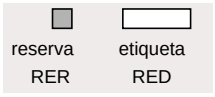
	bus	mem.		C 1			C 2		
acceso	trans.	var.	sum.	cont.	var.	est.	cont.	var.	est.
1. P1 load A1	Pt	A1	mem.	1	A1, A2	L			
2. P2 load A2	Pt	A2	mem.				1	A1, A2	L
3. P1 store A1	PtIm	A1	mem.	1	A1, A2	M	1	A1, A2	I
4. P2 load A2	Pt	A2	C 1	1	A1, A2	L	1	A1, A2	L
5. P2 store A2	PtIm	A2	mem.	1	A1, A2	I	1	A1, A2	M

Tabla 4.11 Compartición falsa.

Los dos primeros accesos son lecturas y los CC determina que son fallos. En los dos casos el bloque se almacena en cache. El tercer acceso es una escritura del procesador P1 a la variable A1. El CC determina que no se dispone de exclusividad para acceder al bloque y efectúa una petición PtIm. Esta transacción invalida la copia del bloque en la cache del procesador P2. EL siguiente acceso a memoria es una lectura del procesador P2 a la variable A2. El bloque está invalidado en cache. Por tanto, el CC efectúa una petición Pt. El siguiente acceso es una escritura del procesador P2 que, al obtener la exclusividad en el acceso al bloque, invalida la copia del bloque en la cache del procesador P1. En estas condiciones, en un acceso posterior del procesador P1 a la variable A1 el CC detectaría que se produce un fallo.

APENDICE A: IMPLEMENTACIÓN DEL PAR DE INSTRUCCIONES LL Y SC

En este apéndice se describe una implementación de las instrucciones load con enlace (LL) y almacenamiento condicional (SC). Por completitud en la Figura 4.34 se muestra la especificación de las instrucciones LL y SC especificada en Figura 2.14.

load con enlace: LL R_d, X(Rf)	Descripción	Registro de enlace (RE)
$RED^v = X + R_f^v$ $RER^v = 1$ $R_d^v = M [X + Rf^v]$ el superíndice v indica valor	Almacena en Rd el contenido de la posición de memoria cuya dirección se calcula como $X + R_f^v$ y efectúa la acción de reserva almacenado la dirección $X + R_f^v$ en un registro denominado registro de enlace (RE).	

almacenamiento condicional: SC R_{fd}, dir	Descripción

if ($RED^v = X + R_f^v$ and $RER^v = 1$) then
cancelar la reserva de otros procesadores de la dirección $X + R_f^v$
$M [X + R_f^v] = R_{fd}^v$
$R_{fd}^v = 1$
else
$R_{fd}^v = 0$
	Comprueba si la reserva de la dirección de memoria en el registro RE es válida. Si es el caso, el contenido del registro Rfd se almacena en la posición de memoria cuya dirección se calcula como $X + R_f^v$ y en el registro R_{fd} se almacena el valor uno. En otro caso, la instrucción se anula y en el registro R_{fd} se almacena un cero

Figura 4.34 Especificación de las intrucciones load con enlace y almacenamiento condicional.

Un CC sólo dispone de un registro de enlace (RE). La reserva se asocia al bloque de cache que contiene la palabra accedida en las instrucciones LL y SC y está soportada por el protocolo de coherencia. Esto es, se utiliza el protocolo de coherencia de cache para invalidar la reserva, cuando otro procesador ejecuta una instrucción store o una instrucción SC a una palabra contenida en el bloque, que contiene la dirección reservada por un LL.

En la Figura 4.35 se muestra un esquema donde se muestra la actuación del agente observador del CC durante el lapso de tiempo que transcurre entre una instrucción LL y una instrucción SC.

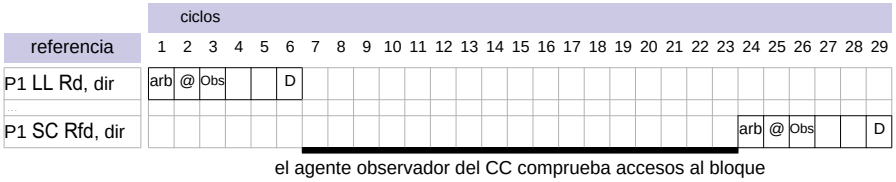


Figura 4.35 Controlador de coherencia: actuación del agente observador para determinar si la ejecución en secuencia de las instrucciones LL y SC se puede considerar atómica.

Adicionalmente, si el procesador deja de ejecutar el flujo de instrucciones en el cual están incluidas las intrucciones LL y SC, debido a un evento[31], también se invalida la reserva. También, se anula la reserva si el bloque, en el que está incluida la palabra accedida, es expulsado de cache.

En la Figura 4.36 se muestra un esquema del camino de datos.

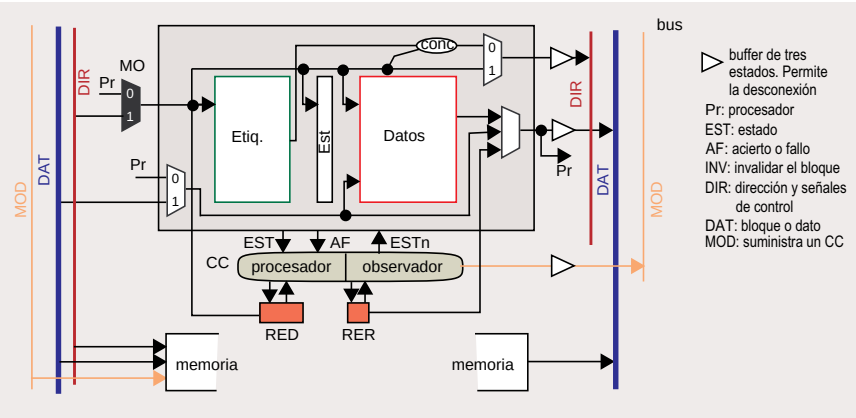


Figura 4.36 Esquema del camino de datos de una cache con registro de enlace.

31. Como puede ser una interrupción o una excepción.

APENDICE B: ACCESOS A MEMORIA CONCURRENTES

Un procesador accede a cache cuando se dispone de los derechos de acceso necesarios, aunque exista una transacción en curso.

La idea es que cualquier acceso a memoria que no obtiene acceso al bus vuelve a acceder a la cache para comprobar los campos etiqueta y el estado del bloque. Notemos que una transacción previa puede haber modificado el estado del bloque. Para ello se utiliza el mecanismo de reinterpretación de intrucciones. La activación de este mecanismo utiliza la señal de concesión de bus.

Por otro lado, el agente observador es prioritario frente al agente procesador para acceder a las estructuras de datos de la cache. Esto es, el procesador suspende la interpretación de instrucciones cuando tiene que acceder a la cache, en el mismo instante de tiempo que está siendo accedida por el agente observador, para procesar una transaccción en curso en el bus[32] (fase Obs y fase RObb en función del protocolo).

En estas condiciones un procesador se bloquea o suspende la interpretación de instrucciones cuando: a) se produce un riesgo estructural, b) se produce un riesgo de datos y c) cuando requiere utilizar el bus en un acceso a memoria[33].

En la Figura 4.37 se muestra un esquema temporal.

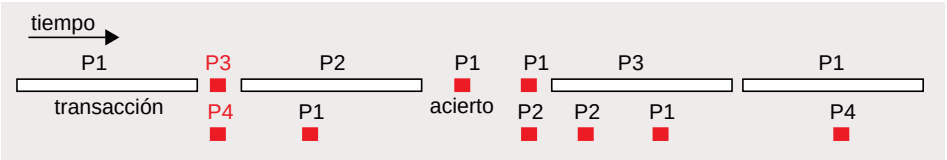


Figura 4.37 Concurrencia de un acceso a memoria y aciertos en cache.

En los siguientes diagramas temporales se muestra la utilización del mecanismo descrito en un procesador segmentado. Suponemos un procesador segmentado con latencia de iniciación de instrucciones igual a un ciclo cuando no se detectan riesgos. Las etapas del procesador son: a) determinar la dirección de la instrucción (CP), b) búsqueda de la instrucción (B), c) decodificación de la instrucción y lectura de operandos en el banco de registros (DL), d) cálculo en la ALU, e) acceso a memoria si es el caso (C1 y C2, dos ciclos) y f) actualización del banco de registros (ES). En los diagramas temporales, la anulación o descarte de una instrucción se indica con el acrónimo “nop” desde la etapa en la cual se decide descartar la instrucción.

32. En un capítulo posterior se analizan en detalle los recursos necesarios.
33. La utilización de un buffer de escrituras (BE) es compatible con el funcionamiento que se describe.

En el primer ciclo del acceso a cache (C1) se accede a los campos etiqueta y estado y en el segundo ciclo (C2) se accede al campo de datos.

Suponemos que una transacción de bus son seis ciclos y las fases son: arbitraje (ARB), transmisión de la dirección (@), observación (Obs), respuesta de observación (ROb), espera y transmisión de datos (D). Notemos que aunque, en un diagrama temporal, se representen todas las fases de una transacción en la misma fila, en la cual se explicitan las fases de interpretación de una instrucción, algunas de las fases utilizan hardware ubicado en otros CC, como por ejemplo la fase de observación.

Por otro lado, para simplificar los diagramas temporales suponemos que los procesadores y el bus funcionan a la misma frecuencia y están sincronizados.

En la primera columna de un diagrama temporal (Figura 4.38) se indica el valor de la dirección que almacenará el registro CP al final del ciclo, en el cual se identifica la etapa CP en la misma fila.

En primer lugar se muestra el diagrama temporal del procesador donde se detecta un fallo de cache y obtiene el acceso al bus (P1, Figura 4.38). Seguidamente se muestra el diagrama de otro procesador (P2, Figura 4.39) que detecta un fallo de cache en el mismo ciclo que el procesador P1. Finalmente se muestra el diagrama temporal de un procesador donde se interpretan instrucciones de acceso a memoria que aciertan en cache durante una transacción en curso. Posteriormente se detecta un fallo de cache y el bus está ocupado por una transacción (P3, Figura 4.40).

El CC del procesador P1 detecta un fallo de cache en el ciclo 5 (Figura 4.38). En el ciclo 6 se solicita al árbitro acceso al bus. El árbitro responde en el siguiente ciclo. Este procesador obtiene acceso al bus e inicia la transacción. Por otro lado, en el ciclo 6 la instrucción load y siguientes se anulan o descartan y hay que volver a reinterpretarlas. En el ciclo 7 se inicia la acción de reinterpretación. En la etapa CP se establece como contador de programa la dirección de la instrucción load que ha detectado un fallo al acceder a cache. Esta nueva interpretación se suspende en el ciclo 9 y se reanuda al finalizar la transacción (ciclo 13, Figura 4.38).

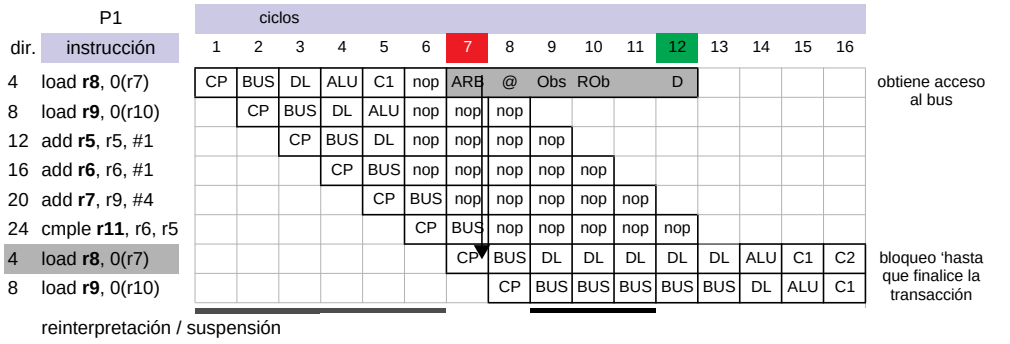


Figura 4.38 Diagrama temporal de un procesador que detecta un fallo de cache y obtiene acceso al bus.

El CC del procesador P2 detecta un fallo de cache en el mismo ciclo que el procesador P1 y efectúa las mismas acciones que el procesador P1 (Figura 4.39). Sin embargo, no obtiene el acceso al bus y esta denegación la reconoce en el ciclo 7. En este ciclo inicia la reinterpretación y suspende la interpretación de instrucciones en el ciclo 9[34]. En la primera fila del diagrama temporal de la Figura 4.39 se muestra la transacción de bus iniciada por el procesador P1.

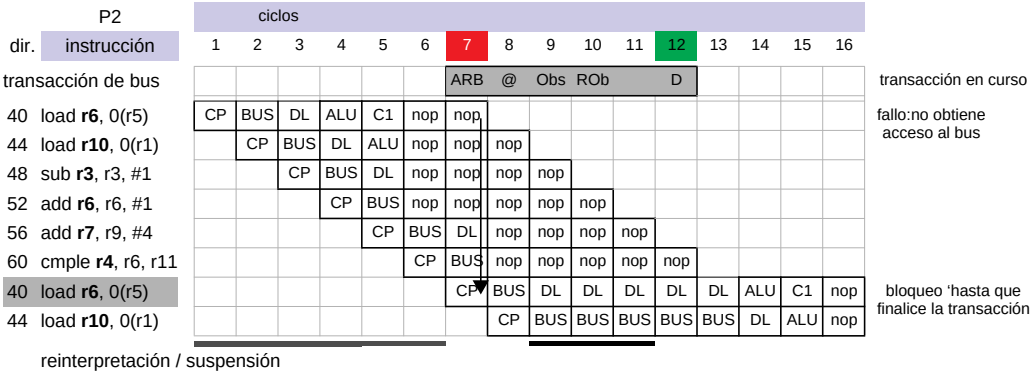


Figura 4.39 Diagrama temporal de un procesador que detecta un fallo de cache y no obtiene acceso al bus.

34. El control del procesador conoce que se ha producido un fallo de cache y no se ha obtenido el bus. Espera a que el bus esté libre. Otra alternativa es no suspender la interpretación y posteriormente volver a reinterpretar la instrucción si no se obtiene el acceso al bus al volver a detectar el fallo en el ciclo 15.

El procesador P3 interpreta instrucciones de acceso a memoria. Cuando se dispone de los derechos de acceso al bloque referenciado se finaliza la interpretación de la instrucción. Ahora bien, cuando se produce un fallo de cache y no se obtiene el acceso al bus (ciclo 10) se inicia el proceso de reinterpretación.

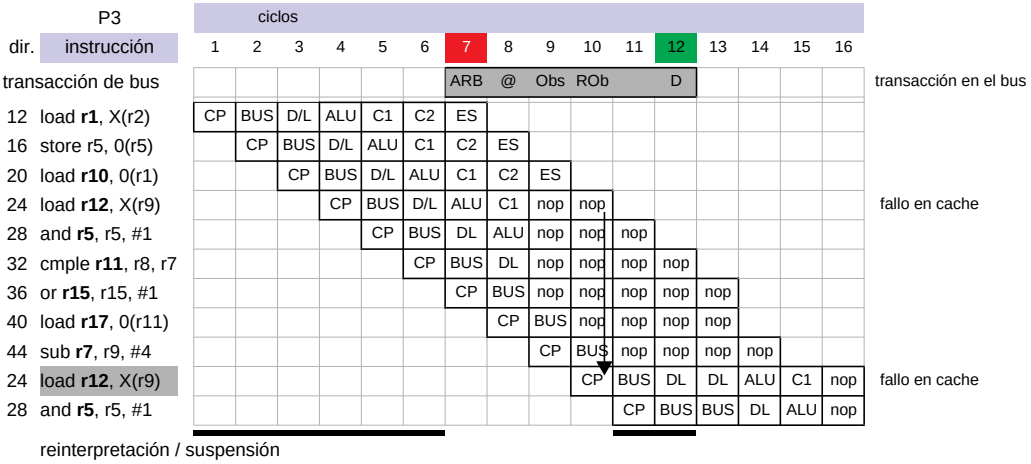


Figura 4.40 Diagrama temporal de un procesador que interpreta instrucciones hasta que detecta un fallo de cache y hay una transacción en curso.

En una cache con escritura retardada, en ocasiones, es necesario actualizar memoria para disponer de un contenedor libre. La instrucción de acceso a memoria que determina la expulsión y siguientes se reinterpretan y se quedan bloqueadas hasta que el controlador de coherencia ha obtenido el bus para expulsar el bloque[35].

En la Figura 4.41 se muestra la actuación cuando se produce un riesgo estructural al acceder a cache. En el ciclo 9 se detecta un riesgo estructural. El CC da prioridad de accesos al agente observador. Entonces, el procesador inicia la reinterpretación de instrucciones en el siguiente ciclo.

Otra posibilidad es disponer en la etapa DL de información sobre si existe una transacción en curso y el ciclo en el cual está. En estas condiciones, se puede suspender la interpretación de instrucciones durante varios ciclos para que no se produzca un riesgo estructural.

35. La utilización de un buffer de expulsiones (BEX) es compatible con el funcionamiento que se describe.

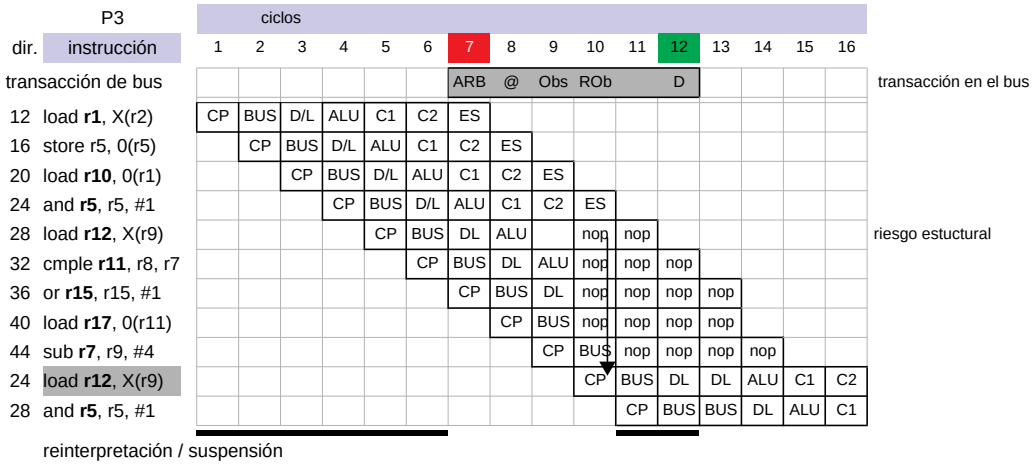


Figura 4.41 Diagrama temporal de un procesador que interpreta instrucciones hasta que detecta un riesgo estructural con una transacción en curso.

EJEMPLOS

Protocolo VI

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Todas las cache utilizan la misma organización (correspondencia directa) y tienen el mismo tamaño. Los bloques que contienen las variable u y t, al almacenarse en las caches, utilizan el mismo contenedor de cache. Las variables no están inicialmente almacenadas en las caches y los valores en memoria son u = 3 y t = 7.

Pregunta 1: *Utilice una tabla para mostrar las transacciones de bus y estado de los bloques en las caches. La secuencia de accesos a memoria se indica en la tabla mostrada en la respuesta.*

Respuesta: En la siguiente tabla se muestra la secuencia de accesos a memoria. El primer acceso a memoria lo efectúa el procesador P1 y es un fallo de lectura. El bloque se almacena en el contenedor uno de cache. El siguiente acceso a memoria (2. P2) es una escritura que también es fallo. La memoria se actualiza. Como no se asigna contenedor en un fallo de escritura, el bloque no se almacena en cache. El tercer acceso a memoria, efectuado por el procesador P2, lee el bloque que él mismo acaba de actualizar. El bloque se ubica en el contenedor uno de cache, ya que el enunciado indica que los bloques correspondientes a las variables u y t utilizan el mismo contenedor de cache.

	bus	mem.			C 1				C 2				C 3			
acceso	trans.	var.	val.	sum.	cont.	var.	val.	est.	cont.	var.	val.	est.	cont.	var.	val.	est.
1. P1 load t	Pt	t	7	mem.	1	t	7	V								
2. P2 store u (41)	PtE	u	41	mem.												
3. P2 load u	Pt	u	41	mem.					1	u	41	V				
4. P3 load t	Pt	t	7	mem.									1	t	7	V
5. P1 store u (17)	PtE	u	17	mem					1	u	41	I				
6. P1 load t					1	t	7	V								
7. P3 load u	Pt	t	17	mem.									1	u	17	V

El cuarto acceso a memoria lo efectúa el procesador P3, siendo un fallo de lectura. Por tanto, el bloque se almacena en cache y en concreto en el contenedor uno de cache. El quinto acceso a memoria lo efectúa el procesador P1. Es una escritura, el bloque no esta almacenado en C1, pero si está almacenado en C2. Por tanto, se invalida el bloque en C2 y se actualiza memoria.

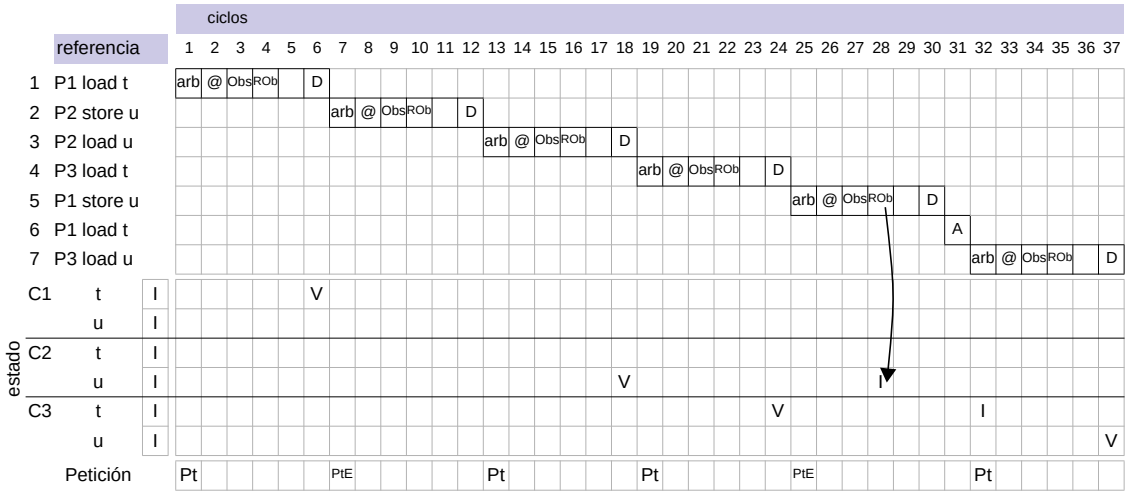
El sexto acceso a memoria es de lectura, lo efectúa el procesador P1 y es un acierto. El séptimo acceso a memoria también es una lectura, efectuada por el procesador P3. Como hay que asignar un contenedor y está ocupado por otro bloque, hay que expulsar el bloque que almacena la variable t. Después de la expulsión, que no requiere ninguna transacción de bus, se solicita el bloque a memoria y se almacena en el contenedor uno de cache.

Pregunta 2: *Indique el número de expulsiones que se producen.*

Respuesta: El número total de expulsiones es uno y se produce en el séptimo acceso a memoria, el cual lo realiza el procesador P3.

Pregunta 3: *Muestre en un diagrama temporal la secuencia de accesos. Suponga que una expulsión no requiere tiempo. La invalidación de un bloque expulsado se indica en la fase "arb" de la transacción cuya petición ha determinado la expulsión.*

Respuesta: En la siguiente figura se muestra el diagrama temporal.



Protocolo MLI

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Todas las cache utilizan la misma organización (correspondencia directa) y tienen el mismo tamaño. Los bloques que

contienen las variable u y t al almacenarse en las caches se ubican en el mismo contenedor de cache. Las variables no están inicialmente almacenadas en cache y los valores en memoria son u = 4 y t = 5.

Pregunta 1: *Utilice una tabla para mostrar las transacciones de bus y estado de los bloques en las caches. La secuencia de accesos a memoria se indica en la tabla mostrada en la respuesta.*

Respuesta: Los dos primeros accesos a memoria generan una petición de bloque y el estado del bloque al finalizar cada una de las transacciones es L. Para realizar el tercer acceso a memoria, el CC del procesador P1 debe obtener la exclusividad en el acceso al bloque. Para ello, el CC genera una petición de bloque con intención de modificación. El cuarto acceso a memoria es mimético al tercer acceso, estando la diferencia en el procesador que efectúa el acceso.

	bus		mem.			C 1				C 2			
acceso	trans.	señal	var.	val.	sum.	cont.	var.	val.	est.	cont.	var.	val.	est.
1. P1 load t	Pt		t	5	mem.	1	t	5	L				
2. P2 load u	Pt		u	4	mem.					2	u	4	L
3. P1 store t (21)	PtIm		u	4	mem.	1	t	21	M				
4. P2 store u (8)	PtIm		t	5	mem.					2	u	8	M
5. P2 load t	PtX		u	8	C 2					2	u	8	I
	Pt	MOD	t	21	C 1	1	t	21	L	2	t	21	L
6. P2 store u (12)	PtIm		u	21	mem.					2	u	12	M
7. P1 load t						1	t	21	L				
8. P2 load u										2	u	12	M

En el quinto acceso el CC detecta un fallo y es necesaria una acción de reemplazo. El bloque que se expulsa está en estado M. Por tanto, es necesario actualizar memoria. Después de la transacción de actualización de memoria se efectúa la petición de bloque. El bloque lo suministra la cache C1, ya que tiene el bloque en exclusividad. El estado del bloque al finalizar la transacción es L en las caches C1 y C2.

El sexto acceso es una escritura y el CC determina que es un fallo. La acción de reemplazo requiere expulsar un bloque en estado L. Como memoria está actualizada la acción de expulsión invalida el bloque (no se representa en la tabla).

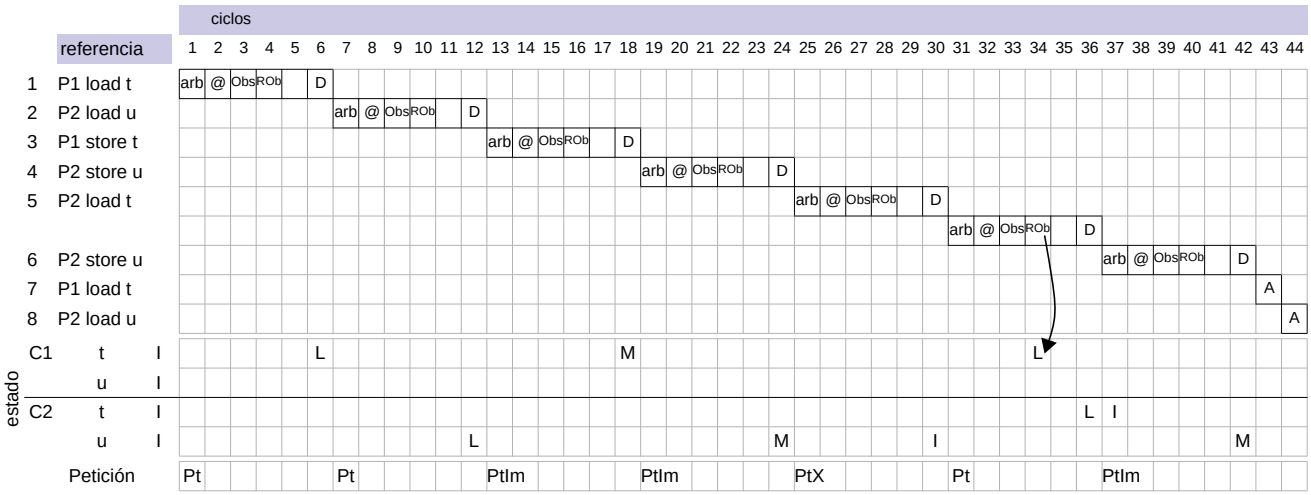
Seguidamente se inicia una transacción de petición de bloque con intención de modificación. Al finalizar la transacción el estado del bloque es M. Los dos últimos accesos a memoria son operaciones de lectura que son aciertos en cache. En el primer caso el bloque está en estado L y en el segundo el bloque está en estado M.

Pregunta 2: *Indique el número de expulsiones que se producen.*

Respuesta: En total se efectúan dos expulsiones. Una corresponde a un bloque en estado M (quinto acceso) y otra corresponde a un bloque en estado L (sexto acceso). Las dos expulsiones se efectúan en la cache del procesador P2.

Pregunta 3: *Muestre en un diagrama temporal la secuencia de accesos. Suponga que una expulsión de un bloque en el estado L no requiere tiempo. En caso contrario utilice una fila para representar sólo la expulsión y en la siguiente fila represente la transacción correspondiente a la petición que ha inducido la expulsión. La invalidación de un bloque expulsado en estado L se indica en la fase "arb" de la transacción cuya petición ha determinado la expulsión.*

Respuesta: La siguiente figura muestra el diagrama temporal que se solicita.



Compartición falsa

Suponga el siguiente bucle que se ejecuta de forma paralela en un multiprocesador. En la parte derecha de la figura se muestran las instrucciones que debe considerar al ejecutar una iteración del bucle.

Alto nivel	Ensamblador
do I = 1, N	load R2, B
A(I) = B(I) + C(I)	load R3, C
endo	add R4, R2, R3; considere tiempo cero
	store R4, C

El multiprocesador dispone de dos procesadores. Suponga que al ejecutar las instrucciones en los procesadores el funcionamiento es síncrono y en un ciclo determinado sólo se ejecuta una instrucción de un procesador. En ciclos consecutivos se ejecutan instrucciones de procesadores cuyo ordinal es consecutivo.

Suponga que las cache tienen un tamaño ilimitado. Esto es, sólo se producen fallos denominado de carga y fallos denominados de coherencia (debido a una invalidación previa producida por una transacción de otro procesador).

El tamaño del bloque permite que contenga cuatro elementos de un vector. Los vectores están alineados a tamaño de bloque.

Suponga que los vectores B y C están inicialmente almacenado en todas las caches y que el vector A no está almacenado en ninguna cache. El valor de N es divisible por cuatro.

Considere un protocolo de invalidación con escritura retardada.

Pregunta 1: *El bucle se ejecuta asignando iteraciones consecutivas a procesadores distintos. Calcule el número de fallos total de carga y de coherencia debido a accesos a datos al ejecutar el programa paralelo.*

Respuesta: Un procesador ejecuta iteraciones distanciadas el número de procesadores. Al ser el tamaño de bloque mayor que una palabra se produce compartición falsa.

El multiprocesador tiene dos procesadores. Un procesador accede a dos elementos del vector A contenidos en un mismo bloque. Entonces, para un procesador, el número de fallos por bloque:

- carga: 1
- coherencia: 1

Por procesador, el número total de fallos por bloque es dos.

El número de bloques es (número de elementos del vector / número de elementos contenidos en un bloque): $\left\lceil \frac{N}{4} \right\rceil$

Entonces, el numero total de fallos es:

- carga: $\left\lceil \frac{N}{4} \right\rceil$ x 2
- coherencia: $\left\lceil \frac{N}{4} \right\rceil$ x 2

Pregunta 2: *El bucle se ejecuta asignado iteracione consecutivas al mismo procesador. Calcule el número de fallos total debido a accesos a datos al ejecutar el programa paralelo y el número de fallos por procesador.*

Respuesta: Un procesador ejecuta iteraciones contíguas. No se produce compartición falsa.

El multiprocesador tiene dos procesadores. Entonces, para un procesador, el número de fallos por bloque:

- carga: 1
- coherencia: 0

Por procesador, el número total de fallos por bloque es uno.

El número de bloques es: $\left\lceil \frac{N}{4} \right\rceil$

Entonces, el numero total de fallos es:

- carga: $\left\lceil \frac{N}{4} \right\rceil$ x 2
- coherencia: 0

Considere un protocolo de invalidación con escritura inmediata.

Pregunta 3: *Calcule, en los dos supuestos que se han indicado para escritura retardada, el número de fallos y el número total de accesos a memoria.*

Respuesta: En los dos supuestos el número de accesos a memoria es N, ya que se utiliza escritura inmediata. También en los dos supuestos el número de fallos, aunque no se asigna contenedor, es N, ya que el vector A no está almacenado en cache.

Protocolo de espera para obtener una llave

Por protocolo de espera se entiende la forma de actuar de los hilos cuando encuentran que la llave está ocupada.

Una forma de espera para obtener la llave es estar constantemente intentando obtenerla. Esta forma de actuar es lo que se denomina espera activa. La espera activa típicamente se utiliza cuando: a) se espera que la llave este ocupada durante intervalos muy cortos de tiempo y b) cuando queremos que la acción de obtener la llave tenga una latencia reducida si la llave está libre.

En la siguiente figura se muestran las primitivas adquirir y liberar, que se utilizan para iniciar y finalizar el acceso exclusivo a datos compartidos por parte de una secuencia de código. Las primitivas adquirir y liberar se muestran en un lenguaje de alto nivel y en LM. En la primitiva adquirir se utiliza espera activa.

```
adquirir (llave)
R=1
repeat
   intercambio (llave,R)
until R = 0
return
```

```
liberar (llave)
llave = 0
return
```

```
      Intercambio (R1, R4)        liberar (R1)
1$:   mov R3, R4                  mov R4, #0
      LL R2, 0(R1)                store R4, 0(R1)
      SC R3, 0(R1)                return
      beq R3, 1$
      mov R4, R2
      return
```

```
acceso exclusivo
llave = 0
. . .
adquirir (llave)
   . . .
   acceso exclusivo
   . . .
liberar (llave)
```

En las siguientes preguntas se analizan características de las primitivas adquirir y liberar. El análisis se efectúa en un multiprocesador que utiliza un bus como red de interconexión. El protocolo de coherencia es de invalidación con escritura retardada (MLI).

Suponga la siguiente secuencia de acceso, correspondiente a varios hilos que intentan obtener una llave que está libre.

accesos	accesos	memoria (valor inicial)	cache (almacena)
1. P1 LL Rd, llave	6. P3 SC Rfd, llave	llave: 0	llave: no
2. P1 SC Rfd, llave	7. P3 LL Rd, llave		
3. P3 LL Rd, llave	8. P3 SC Rfd, llave		
4. P2 LL Rd, llave	9. P2 LL Rd, llave		
5. P2 SC Rfd, llave			

En la secuencia previa, cada procesador que intenta adquirir la llave ejecuta el par de instrucciones LL y SC.

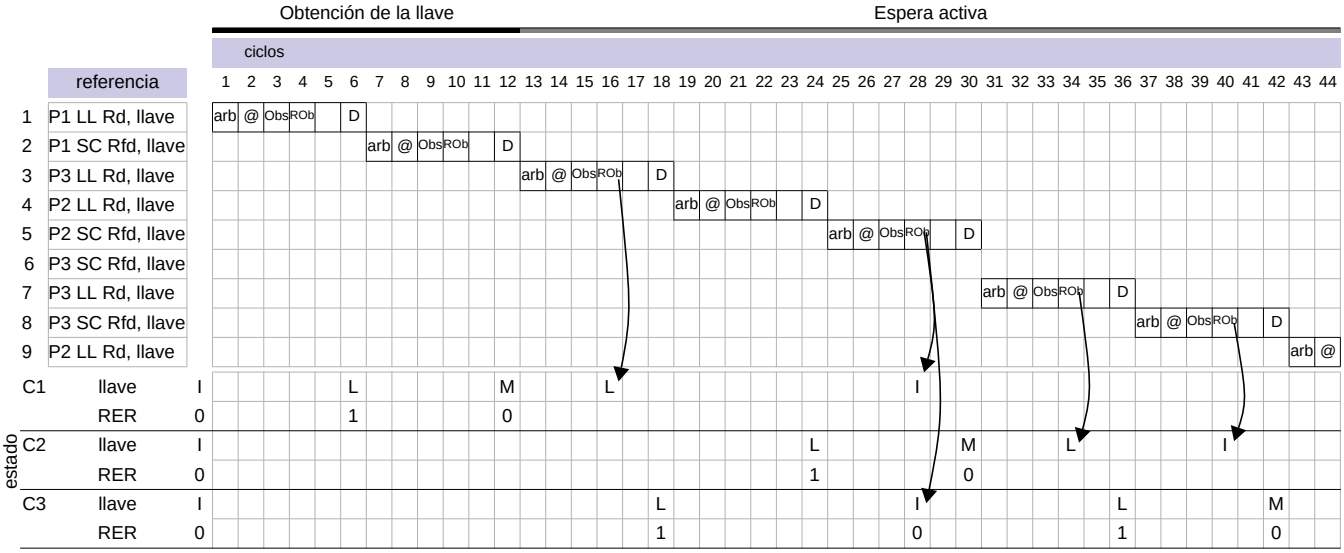
Pregunta 1: *Muestre, mediante una tabla, las transacciones de bus y estado de los bloques en las caches. Muestre también el valor del bit de enlace en cada cache (RER).*

Respuesta: El procesador P1 efectúa la operación intercambio de forma atómica mediante los dos primeros accesos a memoria. Los otros procesadores, debido al entrelazado en la ejecución de instrucciones, compiten por ejecutar la operación intercambio de forma atómica. El procesador P2 ejecuta la operación atómica en los accesos a memoria 4 y 5. El procesador P3 consigue ejecutar la operación en los accesos a memoria 7 y 8. Notemos que previamente la ejecución de una instrucción SC, por parte del procesador P3, ha sido anulada, debido a que ha sido desactivado el RER por una instrucción SC previa del procesador P2.

	bus		mem.			C 1			C 2			C 3		
acceso	trans.	señal	var.	val.	sum.	RER	val.	est.	RER	val.	est.	RER	val.	est.
1. P1 LL Rd, llave	Pt		llave	0	mem.	1	0	L						
2. P1 SC Rfd, llave	PtIm		llave	0	mem.	0	1	M						
3. P3 LL Rd, llave	Pt	MOD	llave	1	C1	0	1	L				1	1	L
4. P2 LL Rd, llave	Pt		llave	1	mem.				1	1	L			
5. P2 SC Rfd, llave	PtIm		llave	1	mem.	0	1	I	0	1	M	0	1	I
6. P3 SC Rfd, llave														
7. P3 LL Rd, llave	Pt	MOD	llave	1	C2				0	1	L	1	1	L
8. P3 SC Rfd, llave	PtIm		llave	1	mem.				0	1	I	0	1	M
9. P2 LL Rd, llave	Pt	MOD	llave	1	C3				1	1	L	0	1	L

Pregunta 2: *Muestre mediante diagrama temporal las transacciones de bus y estado de los bloques en las caches. Muestre también el valor del bit de enlace en cada cache (RER).*

Respuesta: En la siguiente figura se muestra el diagrama temporal.



En el código previo, cada procesador que está esperando adquirir la llave ejecuta el par de instrucciones LL-SC. Esta forma de operar genera mucho tráfico de bus. Cuando una instrucción SC genera una transacción de bus se invalidan las copias del bloque en las otras caches.

Ahora bien, es suficiente que los procesadores que están esperando obtener la llave consulten el valor. Una consulta se efectúa mediante una instrucción load, la cual no genera tráfico de bus, una vez el bloque está almacenado en cache.

Una llave se libera ejecutando una instrucción store. Esta instrucción invalida las copias del bloque que contiene la variable llave, en los procesadores que están consultando su valor. En la siguiente consulta se producen fallos de cache y consultan el valor actualizado, el cual indica que la llave ha sido liberada.

En la siguiente figura se muestra una modificación de la primitiva adquirir. Antes de intentar una operación intercambio se consulta el valor de la variable llave mediante una instrucción load. Cuando se tiene copia del bloque en cache, la ejecución de una instrucción load no produce tráfico. Una vez que un procesador detecta que el acceso exclusivo ha sido liberado ejecutará la operación atómica intercambio.

```
adquirir (llave)
repeat
   repeat
   until llave = 0
   R = 1
   intercambio (llave,R)
until R = 0
return
```

```
adquirir (R1, R4)
1$:  load R4, 0(R1)
     bnez R4, 1$
     mov R4, #1
     intercambio (0(R1),R4)
     bnez R4, 1$
     return
```

```
Intercambio (R1, R4)
1$:  mov R3, R4
     LL R2, 0(R1)
     SC R3, 0(R1)
     beq R3, 1$
     mov R4, R2
     return
```

Suponga la siguiente secuencia de acceso correspondiente a varios hilos que intentar obtener una llave que está libre. Previamente a la secuencia mostrada se ha ejecutado P1 load Rd, llave.

accesos	accesos	memoria (valor inicial)	cache (almacena)		
			C1	C2	C3
1. P1 LL Rd, llave	4. P2 load Rd, llave	llave: 0	llave: si	llave: no	llave: no
2. P1 SC Rfd, llave	5. P3 load Rd, llave				
3. P3 load Rd, llave	6. P2 load Rd, llave				

Pregunta 3: *Muestre mediante una tabla las transacciones de bus y el estado de los bloques en las caches. Muestre también el valor del bit de enlace en cada cache (RER).*

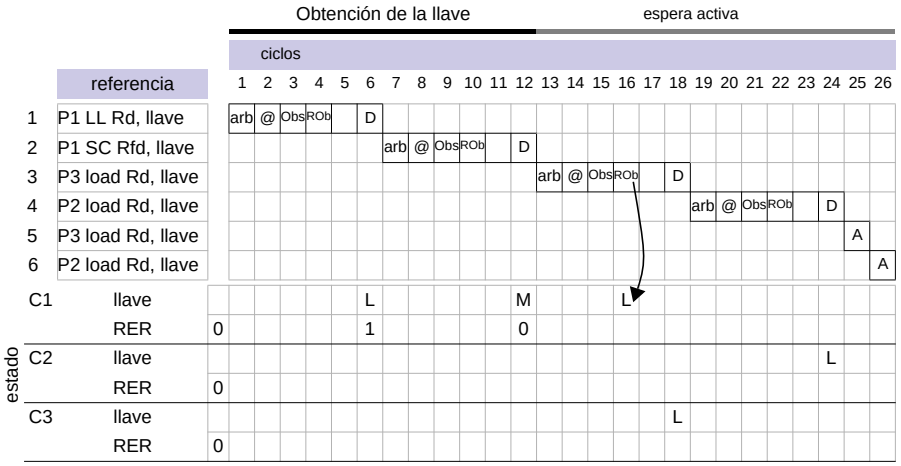
Respuesta: La instrucción LL del procesador P1 acierta en cache y activa el bit de enlace. Al ejecutar el mismo procesador la instrucción SC efectúa la operación intercambio.

Los procesadores P2 y P3, al ejecutar la instrucción load, comprueban que la llave está ocupada. Siguen ejecutando la instrucción load. En estas ejecuciones de las instrucciones load se producen aciertos en cache.

	bus		mem.			C 1			C 2			C 3		
acceso	trans.	señal	var.	val.	sum.	RER	val.	est.	RER	val.	est.	RER	val.	est.
1. P1 LL Rd, llave			llave	0		1	0	L						
2. P1 SC Rfd, llave	PtIm		llave	0	mem.	0	1	M						
3. P3 load Rd, llave	Pt	MOD	llave	1	C1.	0	1	L				0	1	L
4. P2 load Rd, llave	Pt		llave	1	mem.				0	1	L			
5. P3 load Rd, llave														
6. P2 load Rd, llave														

Pregunta 4: *Muestre mediante diagrama temporal las transacciones de bus y estado de los bloques en las caches. Muestre también el valor del bit de enlace en cada cache (RER).*

Respuesta: En la siguiente figura se muestra el diagrama temporal.



Pregunta 5: *Proponga un código de la primitiva adquirir que no utilice la operación intercambio como una subrutina. En su lugar inserte el código directamente ("inline"). En este contexto optimice el código de la primitiva adquirir de forma que el número de instrucciones sea el menor posible.*

Respuesta: Durante la espera activa se está ejecutando una instrucción LL. Cuando se detecta que se ha liberado el acceso exclusivo se ejecuta inmediatamente una instrucción SC.

```
                adquirir (R1, R4)
1$:   LL R4, 0(R1)
      bne R4, 1$
      mov R3, #1
      SC R3, 0(R1)
      beq R3, 1$
      return
```

EJERCICIOS

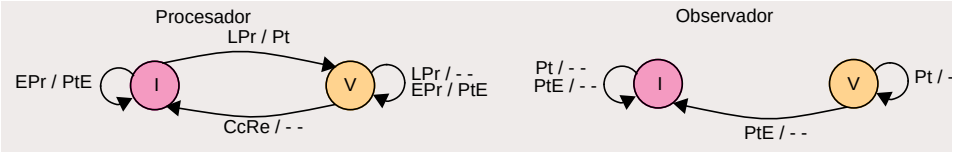
Descripción de un protocolo de observación VI denominado A

Un multiprocesador utiliza un bus como red de interconexión. Las caches privadas utilizan escritura inmediata sin asignación de contenedor en fallo de escritura. El multiprocesador utiliza la técnica de invalidación para mantener la coherencia. En cada contenedor de cache se identifican dos posibles estados: inválido (I) y válido (V).

Las peticiones de procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache	
Peticiones	**Transacciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque
EPr : escritura del proc.	PtE: petición de escritura de un dato	

Los diagramas de transiciones entre estados son los siguientes:



Un procesador inicia los accesos a cache en orden de programa y la cache es bloqueante.

Descripción de un protocolo de observación MLI denominado B

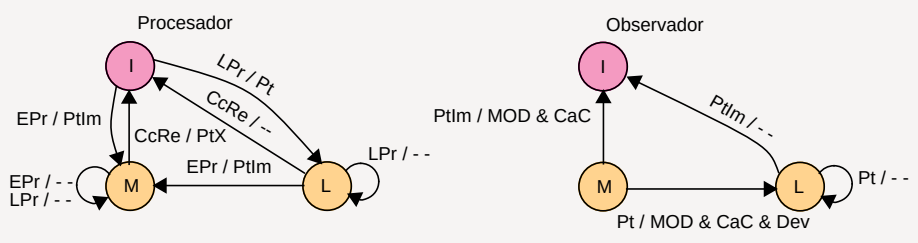
Un multiprocesador utiliza un bus como red de interconexión. Las caches privadas utilizan escritura retardada con asignación de contenedor en fallo. El multiprocesador utiliza la técnica de invalidación para mantener la coherencia. En cada contenedor de cache se dispone de dos bits para identificar los tres posibles estados: inválido (I), lectura (L) y modificado (M).

Las peticiones de procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: almacenar en memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificación	MOD: señal que indica bloque en estado M en una cache	
	PtX: actualización de memoria	CaC: suministro del bloque	

Una cache puede suministrar directamente el dato dentro de una transacción de bus iniciada por otro procesador (CaC). Además, en este caso se actualiza memoria, si es el caso. Cuando una cache tiene un bloque en estado M activa la señal denominada MOD. En el bus se dispone de un cable que es la función OR de las señales MOD de cada una de las caches.

El diagrama de transiciones entre estados es el siguiente.



Cuando es necesario efectuar una acción de reemplazo, primero se actualiza memoria, si es el caso. Posteriormente se efectuan las acciones relacionadas con el acceso que determina el reemplazo.

Un procesador inicia los accesos a cache en orden de programa y la cache es bloqueante.

Descripción de un protocolo de observación MELI denominado C

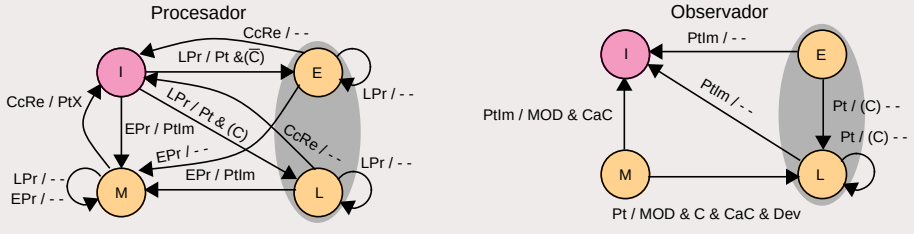
Un multiprocesador dispone de caches privadas que utilizan escritura retardada y la red de interconexión es un bus. Para mantener la coherencia entre las cache privadas se utiliza la técnica de invalidación. El protocolo de coherencia de cache se denomina MELI y tiene cuatro estados (inválido (I), exclusivo (E), lectura (L), modificado (M)).

Las peticiones de procesador, las transacciones de bus y las acciones son las siguientes.

Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificación	C: hay una copia del bloque en cache	
	PtX: petición para actualizar memoria	MOD: bloque en estado M	
		CaC: suministro del bloque	

En el bus se dispone de las señales C y MOD que indican respectivamente, si hay copias del bloque en otra cache o si el bloque lo suministra una cache. Estas señales en el bus son la función OR de las señales correspondiente de cada una de las caches.

Los diagramas de transiciones entre estados son los siguientes:



Cuando un controlador de cache detecta que tiene una copia actualizada (M) suministra (CaC) el bloque en los ciclos correspondientes de la transacción; en paralelo se actualiza la memoria (Dev), si es el caso. En los otros casos el bloque lo suministra la memoria.

Cuando es necesario efectuar una acción de reemplazo, primero se actualiza memoria, si es el caso. Posteriormente se efectuan las acciones relacionadas con el acceso que determina el reemplazo.

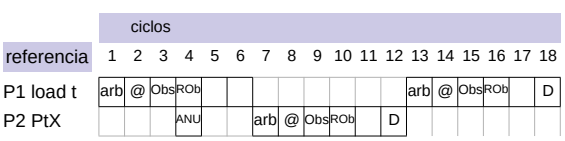
Un procesador inicia los accesos a cache en orden de programa y la cache es bloqueante.

Intervención indirecta

Los protocolos de coherencia B y C descritos utilizan transferencia entre caches para satisfacer la solicitud de un bloque (CaC). Esta característica se denomina intervención directa. Ahora bien, también se puede utilizar lo que se denomina intervención indirecta.

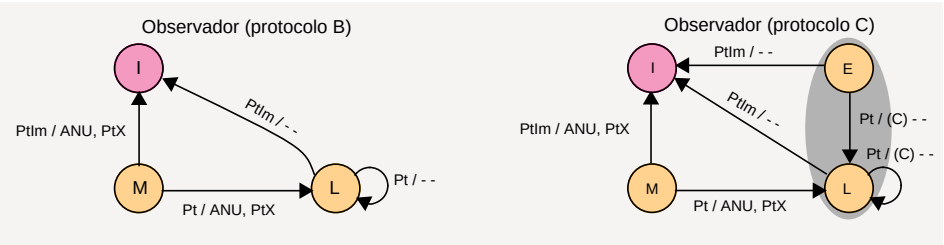
Por intervención indirecta se entiende que una cache nunca suministra un bloque directamente a otra cache. En su lugar, la cache que tiene el bloque en estado M anula la transacción en curso y posteriormente actualiza memoria. Se espera que la actualización de memoria sea antes de que la cache, cuya transacción ha sido anulada, vuelva a iniciar la transacción. En caso contrario, volverá a anularse la transacción. La implementación de este mecanismo requiere de una señal en el bus que indica que una transacción debe repetirse (ANU). Observemos que la señal ANU es el equivalente de la señal MOD cuando se utiliza intervención directa.

En estas condiciones, las transacciones Pt y PtIm siempre obtienen el bloque de datos de memoria. En la figura se muestra un ejemplo de la acción descrita mediante un diagrama temporal.



arb: fase de arbitraje
@: fase de transmisión de la dirección
Obs: fase de observación
ROb: fase de respuesta
D: fase de suministro del bloque

El diagrama de estados del agente observador en los protocolos de coherencia B y C se muestra en la siguiente figura.



Cuando se utiliza una tabla para representar las transacciones de bus y estados de los bloques, la acción de anulación de una transacción, la actualización de memoria por parte del controlador de memoria que anula la transacción y la reemisión de la petición anulada se representa en tres filas consecutivas.

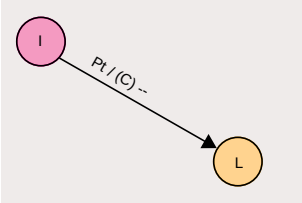
	bus				
acceso	**trans.**	**señal**			**comentario**
P5 load t	Pt	anu C4	. . .		transacción de P5 anulada por C4
	PtX		. . .		actualización de memoria por parte de C4
	Pt		. . .		se repite la transacción de P5

Cuando hay que actualizar memoria, al expulsar un bloque, se utilizan dos filas. En la primera fila se representa la acción de expulsión y en la segunda fila la petición que ha determinado la expulsión.

Lectura con actualización

Lectura con actualización (snarfing, cazar al vuelo) es una mejora del protocolo de coherencia, que utiliza la inherente capacidad del bus de difundir o radiar información; todos los procesadores están conectados al bus y pueden observar cualquier transacción. Cuando una transacción de bus es una lectura (Pt), los CC comparan la dirección de bus con las etiquetas de su cache. Si la dirección del bus se corresponde con la etiqueta de un bloque almacenado en

la cache, que esta en el estado I, se captura el bloque que transporta el bus y se almacena en cache en el estado L. Al diagrama de eventos del observador se añade la siguiente transición.



En estas condiciones, referencias pendientes al mismo bloque (fallos que no han obtenido el bus) se sirven inmediatamente como aciertos y referencias posteriores se comportan como aciertos en cache.

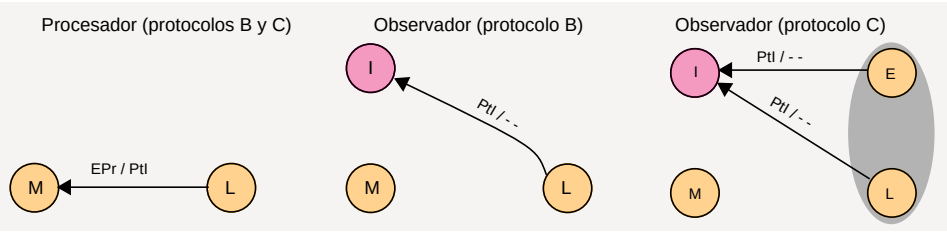
Transacción de exclusividad (petición PtI). Solicitud de acceso a un bloque en exclusividad partiendo del estado L

Para reducir el ancho de banda de bus se utiliza una transacción específica cuando un bloque en cache está en el estado L y el procesador quiere obtener la exclusividad de acceso al bloque. Esta transacción se denomina PtI. Su objetivo es que las otras caches, que tienen copia del bloque en el estado L, invaliden la copia. La diferencia entre una transacción PtI y una transacción PtIm es que no se transfiere el bloque de datos por el bus.

La transacción de bus adicional es la siguiente.

Controlador de cache
Transacción
PtI: petición de invalidación para obtener el permiso de escritura

En la siguiente figura se muestra la transición del estado L al estado M en los protocolos de coherencia B y C. También se muestra, en el diagrama de estados del agente observador, la transición inducida por la observación de una petición PtI.



Ejercicio 4.1

En un uniprocesador la cache de primer nivel utiliza escritura inmediata para mantener la coherencia en la jerarquía de memoria. En un fallo de escritura se lee el bloque de memoria y se almacena en un contenedor de cache (asignación de contenedor en un fallo de escritura). Posteriormente se actualiza el bloque en cache y en memoria. El controlador de coherencia secuencia las dos peticiones necesarias.

Pregunta 1: *Detalle los eventos o peticiones del procesador, las acciones del controlador de cache y las transacciones de bus.*

Pregunta 2: *Indique los estados de un bloque en un contenedor de cache y diseñe el diagrama de transición de estados.*

Suponga la siguiente secuencia de accesos a las variables u y t que están contenidas en bloques de memoria distintos. Cuando estos bloques se ubican en cache ocupan contenedores distintos.

accesos	accesos	memoria (valor inicial)	cache (almacena)
1. lee u	3. escribe t (10)	u: 5	u: no
2. escribe u (7)	4. lee t	t: 20	t: no

Pregunta 3: *Muestre, mediante una tabla, el estado en cache de los bloques que contienen las variables u y t. Así mismo, muestre el valor de las variables en cache y en memoria y las transacciones de bus en cada acceso del procesador.*

acceso	bus	valor en memoria	contenedor / variable	valor	estado

El procesador y la cache descrita se utilizan para diseñar un multiprocesador que utiliza un protocolo de invalidación y como red de interconexión un bus. En un fallo de escritura se lee el bloque de memoria y se almacena en un contenedor de cache. Posteriormente se actualiza el bloque en cache y en memoria. El secuenciamiento de las dos transacciones lo efectúa el CC.

Pregunta 4: *Indique los estados de un bloque en un contenedor de cache y diseñe el diagrama de transición entre estados.*

Suponga la siguiente secuencia de accesos a las variables u y t que están contenidas en bloques de memoria distintos. Cuando estos bloques se ubican en cache ocupan contenedores distintos. El multiprocesador dispone de 3 procesadores.

accesos	accesos	memoria (valor inicial)	cache(almacena)		
			C1	C2	C3
1. P1 lee u	5. P1 lee u	u: 5	u: no	u: no	u: no
2. P3 lee u	6. P2 lee t	t: 20	t: no	t: no	t: no
3. P3 escribe u (7)	7. P2 escribe u (9)				
4. P2 escribe t (10)					

Pregunta 5: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Bus	Memoria			Cache 1			Cache 2			Cache 3		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado	cont. / varia	valor	estado

Ejercicio 4.2

Un multiprocesador utiliza caches con escritura retardada, asignación de contenedor en un fallo de escritura y un protocolo de coherencia de tipo invalidación con dos estados. La red de interconexión utilizada es un bus.

Estados	Descripción
I	inválido
V	válido

Las peticiones del procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de coherencia		Memoria
Peticiones	**Peticiones**	**Acciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr : escritura del proc.	PtIm : petición de bloque con intención de modificación	CaC: suministro del bloque	
	PtX: petición para actualizar memoria.		

En una transacción de bus el suministro del bloque puede efectuarlo memoria o una cache (CaC). El bus dispone de una señal denominada MOD. Esta señal es activada por el CC, cuya cache tiene el bloque en exclusividad, al observar una transacción que referencia el bloque. El CC que activa la señal MOD suministra el bloque en la transacción y memoria inhibe el suministro.

Pregunta 1: *Indique cuántas copias de un bloque puede haber en las caches del multiprocesador.*

Pregunta 2: *¿Es necesario distinguir entre petición de bloque (Pt) y petición de bloque con intención de modificación (PtIm)?.*

Pregunta 3: *Diseñe el diagrama de transición entre estados.*

Suponga la siguiente secuencia de accesos a las variables u y t que están contenidas en bloques de memoria distintos. Cuando estos bloques se ubican en cache ocupan contenedores distintos. El multiprocesador dispone de 3 procesadores.

accesos	accesos	memoria (valor inicial)	cache(almacena)		
			C1	C2	C3
1. P1 lee u	5. P1 lee u	u: 5	u: no	u: no	u: no
2. P3 lee u	6. P2 lee t	t: 20	t: no	t: no	t: no
3. P3 escribe u (7)	7. P2 escribe u (9)				
4. P2 escribe t (10)					

Pregunta 4: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Bus	Memoria			Cache 1			Cache 2			Cache 3		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado	cont. / varia	valor	estado

Ejercicio 4.3

Un multiprocesador utiliza caches con escritura inmediata, no se asigna contenedor en un fallo de escritura y un protocolo de coherencia de tipo actualización con dos estados. La red de interconexión utilizada es un bus.

Estados	Descripción
I	inválido
V	válido

Las peticiones del procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache	
Peticiones	**Transacciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque
EPr : escritura del proc.	PtE: petición de escritura	Actu: actualizar la palabra

Pregunta 1: *Diseñe el diagrama de transición entre estados.*

Suponga la siguiente secuencia de accesos a las variables u y t que están contenidas en bloques de memoria distintos. Cuando estos bloques se ubican en cache ocupan contenedores distintos. El multiprocesador dispone de 3 procesadores.

accesos	accesos	memoria (valor inicial)	cache(almacena)		
			C1	C2	C3
1. P1 lee u	5. P1 lee u	u: 5	u: no	u: no	u: no
2. P3 lee u	6. P2 lee t	t: 20	t: no	t: no	t: no
3. P3 escribe u (7)	7. P2 escribe u (9)				
4. P2 escribe t (10)					

Pregunta 2: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Bus	Memoria			Cache 1			Cache 2			Cache 3		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado	cont. / varia	valor	estado

Suponga que en otro diseño del multiprocesador se utiliza asignación de contenedor en un fallo de escritura. En un fallo de escritura, en primer lugar se almacena el bloque en cache y después se actualiza. El CC secuencia las dos peticiones necesarias.

Pregunta 3: *Diseñe el diagrama de transición entre estados.*

Pregunta 4: *Utilice la secuencia de accesos a memoria previa. Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Bus	Memoria			Cache 1			Cache 2			Cache 3		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado	cont. / varia	valor	estado

Ejercicio 4.4

Un multiprocesador utiliza caches con escritura retardada, asignación de contenedor en un fallo de escritura y un protocolo de coherencia de tipo actualización con tres estados. La red de interconexión utilizada es un bus.

Estados	Descripción
I	inválido
L	el bloque sólo se puede leer
M	el bloque se puede leer y escribir

Las peticiones del procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr : escritura del proc.	PtA: petición de actualización	CaC: suministro del bloque	
	PtX: petición para actualizar memoria.	Actu: actualización de la palabra	

En el bus existe una señal denominada C que se activa, después de la observación, cuando alguna cache tiene el bloque almacenado en un contenedor de cache y el estado es válido.

En una transacción el suministro del bloque puede efectuarlo memoria o una cache. La memoria se actualiza en un suministro desde una cache. También se actualiza en una petición de actualización.

En un fallo de escritura, en primer lugar se almacena el bloque en cache y posteriormente se actualiza. El CC secuencia las dos peticiones necesarias.

Pregunta 1: *Indique cuántas copias de un bloque puede haber en las caches del multiprocesador.*

Pregunta 2: *Diseñe el diagrama de transición entre estados.*

Suponga la siguiente secuencia de accesos a las variables u y t que están contenidas en bloques de memoria distintos. Cuando estos bloques se ubican en cache ocupan contenedores distintos. El multiprocesador dispone de 3 procesadores.

accesos	accesos	memoria (valor inicial)	cache(almacena) C1	C2	C3
1. P1 lee u	5. P1 lee u	u: 5	u: no	u: no	u: no
2. P3 lee u	6. P2 lee t	t: 20	t: no	t: no	t: no
3. P3 escribe u (7)	7. P2 escribe u (9)				
4. P2 escribe t (10)					

Pregunta 3: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Cache 1			Cache 2			Cache 3				
acceso	cont. / varia	valor	estado	cont. / varia	valor	estado	cont. / varia	valor	estado	suministro	trans. de bus/ val. memor.

En otra versión del multiprocesador se utiliza un protocolo con política de invalidación, las peticiones del procesador y las transacciones de bus son las siguientes.

Procesador	**Controlador de cache**		**Memoria**
Peticiones	**Transacciones**	**Acciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificar	CaC: suministro del bloque	
	PtX: petición para actualizar memoria.		

Memoria no se actualiza si la tranferencia entre caches es debida a una petición de bloque con intención de modificarlo. En el bus hay una señal, que se activa después de la observación, para indicar suministro desde una cache (MOD). Este suministro sólo se efectúa si el bloque está en estado M.

Pregunta 4: *Diseñe el protocolo de invalidación.*

Pregunta 5: *Utilice la secuencia de accesos a memoria previa. Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Bus	Memoria			Cache 1			Cache 2			Cache 3		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado	cont. / varia	valor	estado

Pregunta 6: *Compare el protocolo de actualización con el protocolo de invalidación: a) número de transferencias de bloque, b) número de transferencias de palabra y c) número de suministros de bloque desde una cache.*

Ejercicio 4.5

Un multiprocesador utiliza caches privadas de mapeo directo, tienen 32 contenedores, el tamaño del campo de datos de cada contenedor es de 16 bytes y utilizan escritura retardada. El protocolo de coherencia tiene 3 estados

(Inválido, Lectura, Modificado) y utiliza la técnica de invalidación para mantener la coherencia. Las operaciones de load y store del procesador son de tamaño fijo igual a 4 bytes. En todas las preguntas debe justificar la respuesta.

Pregunta 1: *¿Pueden estar los contenedores 22 de varias caches en estado de lectura (L) en el mismo instante?.*

- *a) SI*
- *b) No*

Pregunta 2: *¿Pueden estar los contenedores 14 de varias caches en estado modificado (M) en el mismo instante?.*

- *a) SI*
- *b) No*

Pregunta 3: *¿Puede estar un bloque de memoria en varias caches en estado modificado (M) en el mismo instante?.*

- *a) SI*
- *b) No*

Pregunta 4: *¿Puede estar un bloque de memoria en varias caches en el estado modificado (M) en una de ellas y en el estado lectura (L) en el resto de caches en el mismo instante?.*

- *a) SI*
- *b) No*

Pregunta 5: *Indique, si existe, un tamaño de bloque que elimine la compartición falsa*

- *a) no existe ninguno*
- *b) 4 bytes*
- *c) 8 bytes*
- *d) 16 bytes*
- *e) 32 bytes*

Ejercicio 4.6

La compartición de datos en cache puede etiquetarse como verdadera o falsa. Compartición verdadera se refiere a compartir la misma palabra de memoria por varios procesadores. Compartición falsa se detecta en caches con bloques multipalabra y diferentes procesadores que acceden (escritura) a palabras distintas ubicadas en el mismo bloque.

Un modelo de multiprocesador utiliza cache privadas con escritura inmediata y un protocolo de coherencia de observación con invalidación en las escrituras. Como se utiliza escritura inmediata, cada operación de escritura (store) se

trata como un fallo de cache. El dato se escribe en memoria y si el bloque está presente en la cache de cualquier otro procesador la entrada de este bloque se invalida.

En un fallo de cache, se lee el bloque de memoria y se almacena en cache, reemplazando previamente otro bloque si el contenedor está ocupado.

El análisis se efectúa sobre bucles doall y supondremos auto-planificación, con iteraciones consecutivas del bucle asignadas a procesadores distintos. En los análisis que se soliciten estudie el peor caso y suponga que no se producen conflictos en cache.

Pregunta 1: *Analice los efectos del mecanismo de coherencia de cache en el siguiente bucle.*

L.A.N

```
doall I = 1,N
      Y(I) = Y(I) + a x X(I)
enddoall
```

Para reducir el efecto de la compartición falsa un bloque debe ser sólo accedido por un procesador. Para ello debe reescribirse el código de forma que, además de autoplanificación, se utilice planificación estática. La planificación que se obtiene es una planificación por afinidad de datos (affinity scheduling).

Suponga que la dirección base de cada vector es la primera dirección de un bloque de cache. El bucle paralelo se parte en dos bucles (strip-mining). El bucle interno recorre iteraciones contiguas del bucle original. El bucle externo recorre la primera iteración de grupos de iteraciones consecutivas. Entonces, el bucle externo es el bucle paralelo y mediante el bucle interno se asignan iteraciones consecutivas del bucle original al mismo procesador. El código después de efectuar la transformación es el siguiente.

L.A.N

```
doall II = 1, N, Tbloque
      do I = II, min (N, II + Tbloque -1)
            Y(I) = Y(I) + a x X(I)
      endo
enddoall
```

Pregunta 2: *Analice los efectos del mecanismo de coherencia de cache en el bucle transformado.*

Pregunta 3: *Suponga que no se asigna contenedor en una fallo de escritura. Repita el análisis efectuado en las dos preguntas previas.*

Otro modelo del mutiprocesador utiliza escritura retardada en las caches privadas y un protocolo de coherencia de observación con invalidación en las escrituras. Como utiliza escritura retardada, las escrituras no se propagan cuando se tiene copia del bloque en propiedad o exclusividad. Si este no es el caso, obtener el permiso de escritura determina que se invalidan las copias del bloque en otras caches. Una cache que tiene el bloque en propiedad suministra el bloque en la transferencia donde se solicita.

Pregunta 4: *Repita el análisis efectuado en las dos primeras preguntas para el sistema multiprocesador que se acaba de describir.*

Ejercicio 4.7

La compartición de datos en cache puede etiquetarse como verdadera o falsa. Compartición verdadera se refiere a compartir la misma palabra de memoria por varios procesadores. Compartición falsa se refiere a que diferentes procesadores acceden (escritura) a palabras distintas ubicadas en el mismo bloque y estas palabras no se utilizan para comunicarse entre los procesadores.

Se dispone de un multiprocesador con cache privadas que utilizan escritura retardada y un protocolo de coherencia de tipo invalidación MLI. La red de interconexión utilizada es un bus.

Como programa de prueba se utiliza el siguiente bucle paralelo. Al ejecutarse, se utiliza autoplanificación. Esto es, iteraciones consecutivas del bucle paralelo son asignadas a distintos procesadores.

L.A.N

```
do J = 1, N
      X(J) = B(J) / A(J, J)
      doall I = J + 1, N
            B(I) = B(I) - A(I,J) x X(J)
      endoall
enddo
```

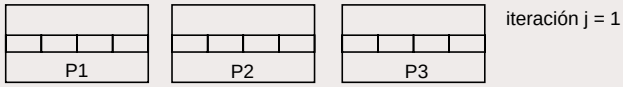
El tamaño de las caches es suficientemente grande para que no existan fallos de capacidad (todas las estructuras de datos caben en la cache). El valor de N es 16, la matriz se almacena por columnas y las direcciones de inicio de las estructuras de datos son 0 para la matriz A(N,N), 256 para el vector X(N) y 272 para el vector B(N). El tamaño del bloque es Tbloque = 4 y el número de procesadores es P = 3.

Suponga que las operaciones que no dan lugar a compartición falsa se ejecutan en tiempo cero. Esto es, se aproxima la ejecución por sólo instrucciones load o store a variables que muestran compartición falsa.

Pregunta 1: *Indique las variables que producen compartición falsa.*

Cuando se solicite información sobre el comportamiento del programa de prueba estudie la iteración J = 1 y las tres primeras iteraciones del bucle paralelo (I).

Pregunta 2: *Para las variables que muestran compartición falsa, indique la información en los contenedores de cache de los tres procesadores. Indique la variable y elemento de la misma que es accedida en cada procesador.*



Suponga que el arbitraje es equitativo y se asigna el bus a los procesadores de forma cíclica.

Pregunta 3: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables que muestran compartición falsa, en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Bus	Memoria		Cache 1		Cache 2		Cache 3	
acceso	trans.	var.	suministro	cont. / varia	estado	cont. / varia	estado	cont. / varia	estado

Pregunta 4: *Proponga una reescritura del algoritmo para reducir el efecto de la compartición falsa. Compruebelo para el valor J = 1. El esqueleto del código es el siguiente.*

L.A.N

```
do J = 1, N
    . . .
    doall II = . . .
        do I = . . .
            . . .
        enddo
    enddoall
enddo
```

En la siguiente pregunta estudie la iteración J = 1 y las 4 primeras iteraciones del bucle II.

Pregunta 5: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables que muestran compartición falsa, en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

	Bus	Memoria		Cache 1		Cache 2		Cache 3	
acceso	trans.	var.	suministro	cont. / varia	estado	cont. / varia	estado	cont. / varia	estado

Pregunta 6: *Calcule la ganancia del algoritmo reescrito respecto del original.*

Ejercicio 4.8

El protocolo de coherencia que se utiliza es el denominado B. Este protocolo se modifica con el mecanismo de transacción de exclusividad y con el mecanismo de intervención indirecta, los cuales han sido descritos junto con los protocolos.

Las cache utilizadas son de mapeo directo y el tamaño de bloque es de 2 palabras (2 variables).

Suponga la siguiente secuencia de accesos a las variables u y t que están contenidas en bloques distintos del espacio lógico. Cuando los bloques se almacenan en cache no hay conflictos.. El multiprocesador dispone de 2 procesadores.

accesos	accesos	memoria (valor inicial)	cache(almacena) C1	C2
1. P1 lee t	5. P2 lee t	u: 4	u: no	u: no
2. P2 lee u	6. P2 escribe u (12)	t: 5	t: no	t: no
3. P1 escribe t (21)	7. P1 lee t			
4. P2 escribe u (8)	8. P2 lee t			

Pregunta 1: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque. En transacción de bus indique también si se anula una transacción (ANU) y en suministro la cache que anula.*

	Bus	Memoria			Cache 1			Cache 2		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado

En una nueva versión del multiprocesador se utiliza exclusivamente el protocolo denominado B.

Pregunta 2: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

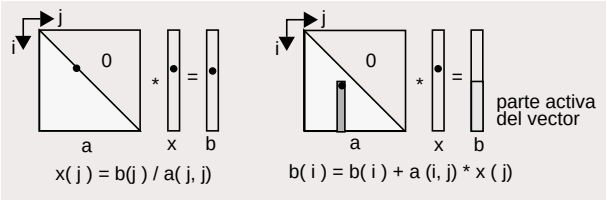
	Bus	Memoria			Cache 1			Cache 2		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado

Pregunta 3: *Calcule la ganancia.*

Ejercicio 4.9

El siguiente programa resuelve un sistema triangular de ecuaciones.

```
do J = 1,N
   X(J) = B(J) / A(J, J)
   do I = J+1, N
      B(I) = B(I) - A(I,J) * X(J)
   enddo
enddo
```



Una paralelización del código anterior es la siguiente.

```
do J = 1,N
   X(J) = B(J) / A(J, J)
   doall I = J+1, N
      B(I) = B(I) - A(I,J) * X(J)
   enddo
enddo
```

Posibles planificaciones del bucle paralelo son las siguientes

A) Planificación estática simple

El bucle doall se convierte en los siguientes bucles

```
doall p = 1, NP
   do i = (p − 1) × ⌈N/(NP)⌉ + j + 1, min(N, p × ⌈N/(NP)⌉)
```

donde NP es el número de procesadores

B) Planificación estática entrelazada

El bucle doall se convierte en los siguientes bucles

```
doall p = 1, NP
   do i = p + j, N, NP
```

donde NP es el número de procesadores

Pregunta 1: *A medida que J se incrementa el número de cálculos que hay que efectuar disminuye. ¿Algunás de las 2 planificaciones (A, B) distribuye mejor la carga de trabajo?. Justifique la contestación. NOTA: observe como se reparten los cálculos cuando el número de operaciones no es múltiplo de NP.*

Pregunta 2: *Modifique la planificación estática entrelazada para que un hilo acceda a un subconjunto de las filas accedidas en la iteración anterior del bucle J. Supondremos que en cada iteración del bucle J un hilo se asigna al mismo procesador. Complete la siguiente estructura del código.*

```
do J =
. . . .
    doall i = N, j, - NP
        . . .
```

Los algoritmos tradicionales de planificación de bucles ignoran la ubicación de datos cuando se asignan iteraciones a los procesadores, lo cual da lugar a una pérdida de rendimiento en multiprocesadores con memorias cache.

Mientras que el movimiento de datos producido por la compartición verdadera no se puede eliminar, es posible minimizar el movimiento de datos producido por una mala asignación de iteraciones a los procesadores.

La idea de planificación por afinidad es asignar la ejecución repetida de iteraciones de un bucle al mismo procesador, asegurando con ello que la mayoria de los accesos a datos acierten en cache (aprovechando la localidad espacial o temporal).

Por ejemplo, si los hilos p = 1:NP en cada iteración del bucle j se asignan a procesadores distintos, datos almacenados en las caches de cada procesador en la iteración anterior del bucle j no se van a utilizar o viceversa (si se asignan al mismo procesador)

Pregunta 3: *El tamaño de bloque es de varios elementos. Indique para las 3 planificaciones (A, B, pregunta 2)) mostradas si es bueno seguir asignando el hilo al mismo procesador en cada iteración del bucle j, en función de si la matriz a(1:N, 1:N) se almacena por filas (lenguaje C) o por columnas (lenguaje FORTRAN). Justifique la contestación.*

planificación	A)	B)	pregunta 2)
filas			
columnas			

Ejercicio 4.10

El protocolo de coherencia que se utiliza es el denominado B. Este protocolo se modifica con el mecanismo de transacción de exclusividad y con el mecanismo de intervención indirecta, los cuales han sido descritos junto con los protocolos.

Las cache utilizadas son de mapeo directo y el tamaño de bloque es de 2 palabras (2 variables).

Suponga la siguiente secuencia de accesos a las variables u, t, v y r. Las variables **u** y **t** están contenidas en el bloque que denominamos **A** y las variables **v** y **r** están contenidas en el bloque que denominamos **B.** Al mapearse en cache los dos bloques ocupan el mismo contenedor de cache.

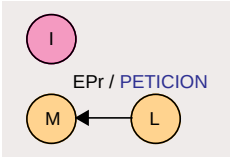
acceso	acceso	memoria (valor inicial)	Caches (almacena)	
			C1	C2
1. P1 load t	6. P1 load t	u: 4	no	no
2. P2 store r (34)	7. P1 store v (37)	t: 5	no	no
3. P1 store t (25)	8. P2 store u (17)	v: 12	no	no
4. P1 store u (71)	9. P1 store r (64)	r: 13	no	no
5. P2 store t (42)	10. P2 load t			

Pregunta 1: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u, t, v y r en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque. En la columna señal indique si se anula una transacción (ANU) y la cache que la anula.*

	bus		mem.						C 1							C 2						
acceso	trans.	señal	blo.	var.	val.	var.	val.	sum.	cont.	blo.	var.	val.	var.	val.	est.	cont.	blo.	var.	val.	var.	val.	est.

Pregunta 2: *En la secuencia de accesos a memoria mostrada previamente indique: a) el número de expulsiones debido a reemplazos, b) el número de expulsiones que requieren actualizar memoria, c) el número de transacciones anuladas y d) el número de accesos a memoria que no generan transacción de bus.*

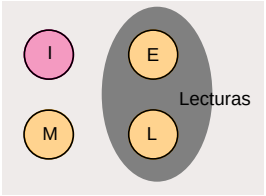
Ejercicio 4.11



Un multiprocesador utiliza caches con escritura retardada y un protocolo de coherencia de tres estados del tipo invalidación. A partir de ahora a este protocolo lo denominaremos protocolo MLI. En el protocolo MLI no se distingue si un dato está realmente compartido por varios procesadores cuando se quiere actualizar en cache. Si un dato se lee y poste-

riormente se escribe hay que notificarlo a las otras caches y obtener el permiso de escritura. En la figura adjunta se muestra la transición involucrada en el diagrama de transiciones entre estados del protocolo MLI.

Está característica determina que el rendimiento sea proporcional, además de a la frecuencia de fallo, a la frecuencia de escrituras. La idea es modificar el protocolo para que el rendimiento sea proporcional a la frecuencia de fallo y al grado de compartición. Para ello, el estado L en el protocolo MLI se expande en dos estados (L y E) en el nuevo protocolo, denominado MELI. El estado E (exclusivo) es factible cuando al producirse un fallo de lectura no existe ninguna copia del bloque en otras caches. El estado L es factible cuando al producirse un fallo de lectura hay alguna copia del bloque en otras caches. En el protocolo de coherencia MELI no se considera, en el diseño, la posibilidad de una transición del estado L al estado E. En la siguiente figura se muestran los estados en el protocolo MELI. Los dos estados (L y E) agrupados utilizando un tramado de fondo, se corresponden con el estado L en el protocolo MLI.



En el protocolo MELI memoria se actualiza cuando una cache suministra el bloque en una petición Pt o cuando un bloque que ha sido modificado es expulsado de una cache.

Para contestar a las siguientes preguntas analice el protocolo de coherencia MLI. La mayoria de transiciones en el protocolo MELI son idénticas.

Pregunta 1: *Describa, utilizando una tabla, para cada estado del protocolo de coherencia MELI el posible estado del bloque en otras caches y memoria.*

Cuando se almacena un bloque en cache, después de un fallo de lectura, el protocolo de coherencia MELI necesita conocer si hay copias del bloque en otras caches. Suponga que a esta información la denominamos C y está disponible para decidir el estado de un bloque en cache.

Pregunta 2: *Muestre de forma funcional las transiciones en un diagrama de estados. Suponga eventos del procesador (LPr, EPr) y eventos de otros procesadores (LPr_{otro}, EPr_{otro}). Indique la acción de propagación y la acción de suministro.*

El protocolo de coherencia MELI se implementa en un multiprocesador que utiliza como red de interconexión un bus. En este bus, además de las señales disponibles en el protocolo MLI, existe una señal denominada C. En una transacción de bus la señal C se activa/desactiva después de la fase de observación. Todas las caches están conectadas a la señal C y en el bus se efectúa la operación OR de las señales individuales. Una cache activa la señal C si ha observado que tiene una copia del bloque en su cache.

Las peticiones del procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificarlo	C: hay una copia del bloque en cache	
	PtX: petición para actualizar memoria	MOD: bloque en estado M	
		CaC: suministro del bloque	

Pregunta 3: *Muestre el diagrama de transiciones entre estados de un bloque en un contenedor de cache con el protocolo de coherencia MELI, cuando la red de interconexión es un bus. Para ello utilice varios diagrama parciales. Considere operaciones de lectura, operaciones de escritura, acción de reemplazo y observaciones de forma separada.*

Suponga la siguiente secuencia de accesos a memoria. Las variables u y t están contenidas en bloques distintos. Al almacenarse los bloque en cache no hay conflictos.

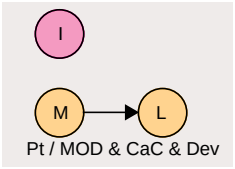
accesos	accesos	memoria (valor inicial)	cache(almacena)		
			C1	C2	C3
1. P1 lee u	5. P3 escribe u (18)	u: 4	u: no	u: no	u: no
2. P2 lee t	6. P1 lee u	t: 5	t: no	t: no	t: no
3. P1 escribe u (32)	7. P2 lee u				
4. P2 escribe t (67)					

Pregunta 4: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mimo, muestre las transacciones de bus y quién suministra el bloque.*

Ejercicio 4.12

En un protocolo de coherencia de cache con tres estados y del tipo invalidación, denominado MLI, siempre se actualiza memoria cuando una cache suministra el bloque en una petición Pt. Posteriormente, si no ha existido otra escritura al bloque, en un fallo de cache la memoria suministra el bloque. Esta

característica puede reducir el rendimiento en multiprocesadores donde la latencia de una transferencia entre caches sea menor que la latencia de acceso a memoria.

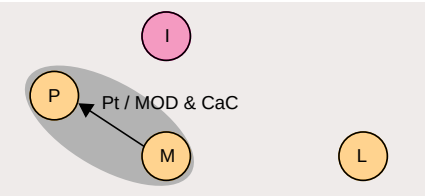


En este ejercicio supondremos que todos los protocolos son del tipo invalidación.

La idea es modificar el protocolo MLI para suministrar más veces desde cache. Esto es, incrementar las transferencia entre caches. El nuevo protocolo se denomina MPLI. La implementación de esta idea se efectúa tipicamente extendiendo el protocolo MELI y se denomina MPELI. En este ejercicio nos centraremos exclusivamente en la idea, ya que no se ven afectadas las transiciones entre los estados adicionales.

El nuevo estado, etiquetado como P (propietario), debe suministrar el bloque en un fallo de cache. La memoria no está actualizada y tampoco se actualiza cuando se suministra desde el estado P. Al estado P se llega exclusivamente desde el estado M, al suministrar el bloque en un fallo de lectura. Al efectuar este suministro la memoria no se actualiza.

En el protocolo MPLI se necesita conocer cuando una cache suministra el bloque. Suponga que a esta información la denominamos MOD y está disponible para determinar si el bloque lo suministra memoria o una cache.



Para contestar a las siguientes preguntas analice el protocolo de coherencia MLI. La mayoria de transiciones en el protocolo MPLI son idénticas.

Pregunta 1: *Describa, mediante una tabla, para cada estado del protocolo de coherencia MPLI el posible estado del bloque en otras caches y memoria.*

Pregunta 2: *Muestre de forma funcional las transiciones en un diagrama de estados. Suponga eventos del procesador (LPr, EPr) y eventos de otros procesadores (LPr_{otro}, EPr_{otro}). Indique la acción de propagación y la acción de suministro.*

Al implementar el protocolo MPLI, en un multiprocesador que utiliza un bus como red de interconexión, se utiliza la señal MOD, disponible en el protocolo MLI, para indicar suministro desde una cache.

Las peticiones de procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificarlo	MOD: bloque en estado M	
	PtX: petición para actualizar memoria	CaC: suministro del bloque	

Pregunta 3: *Muestre el diagrama de transiciones entre estados de un bloque en un contenedor de cache con el protocolo de coherencia MPLI, cuando la red de interconexión es un bus. Para ello utilice varios diagrama parciales. Considere operaciones de lectura, operaciones de escritura, acción de reemplazo y observaciones de forma separada.*

Pregunta 4: *Identifique las transacciones en el diagrama de estados donde deja de ser suministrado el bloque por una cache.*

Suponga la siguiente secuencia de accesos a memoria. Las variables u y t están contenidas en bloques distintos. AL almacenarse los bloques en cache no hay conflictos.

accesos	accesos	memoria (valor inicial)	cache(almacena)		
			C1	C2	C3
1. P1 lee u	5. P3 escribe u (18)	u: 4	u: no	u: no	u: no
2. P2 lee t	6. P1 lee u	t: 5	t: no	t: no	t: no
3. P1 escribe u (32)	7. P2 lee u				
4. P2 escribe t (67)					

Pregunta 5: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mimo, muestre las transacciones de bus y quién suministra el bloque.*

Ejercicio 4.13

Suponga un multiprocesador que utiliza como red de interconexión un bus. Las caches utilizan escritura retardada con un protocolo de coherencia del tipo invalidación.

Queremos evaluar de forma cuantitativa el efecto de la compartición falsa. Para ello utilizaremos el siguiente bucle.

LAN	**Ensamblador**	traducción simplificada
doall I =1,N	1$: L A(I)	load
C(I) = A(I) + 31	S C(I)	store
enddo		jmp 1$

Un procesador no empieza a ejecutar la siguiente instrucción hasta que ha finalizado la anterior.

Cuando se produce un acierto en cache una instrucción tarda un ciclo en ejecutarse. Si se produce un fallo el número de ciclos es uno más el tiempo de servir el fallo. La penalización por fallo de cache es el tiempo de cualquier transacción de bus: 2 ciclos.

Mientras un procesador está utilizando el bus los otros procesadores pueden acceder a sus caches mientras se produzcan aciertos.

Suponga que el árbitro del bus concede el bus de forma cíclica a los procesadores (proc : 1,2,3,4,1,2,3, . . .).

En un fallo de cache el bloque lo suministra memoria u otra cache dentro del tiempo de la transacción.

El número de procesadores es P= 4 y el tamaño de bloque son 4 palabras. El tamaño de cache es suficientemente grande para que se puedan almacenar los vectores A y C conjuntamente y que no se producen conflictos entre los bloques debido al mapeo y algoritmo de reemplazo utilizados.

Los vectores A y C están alineados a tamaño de bloque (A(1) y C(1) son la primera palabra de un bloque).

Supondremos que el vector A está almacenado en cache antes de empezar a ejecutarse el bucle.

Supondremos que N es divisible por 16 (4 procesadores y tamaño de bloque 4 palabras) y que la instrucción de salto se ejecuta en 0 ciclos.

El bucle se planifica asignando N/4 iteraciones consecutivas a cada procesador

```
doall II = 1 , N, N/4
    do I = II , min (II+(N/4)-1, N)
```

Pregunta 1: *Indique si se produce compartición falsa. Muestre en el diagrama que se adjunta la evolución temporal de ejecución de instrucciones en cada procesador. Como ejemplo se muestra la ejecución de algunas instrucciones.*

ciclos	P1	P2	P3	P4	BUS
1	L A1	L AN/4 +1	L AN/2+1	L A3N/4+1	
2	S C1				PtIm
3	1 ciclo				
4	2 ciclo				
5	L A2	S C1			PtIm
6		1 ciclo			
7		2 ciclo			
8					
9					

Pregunta 2: *Calcule el número de ciclos que tarda en ejecutarse el bucle.*

Pregunta 3: *Calcule la ganancia respecto a la ejecución en un solo procesador.*

El bucle se planifica asignando iteraciones consecutivas a procesadores distintos

```
doall p =1 ,4
    do I = p, N, 4
```

Pregunta 4: *Indique si se produce compartición falsa. Muestre en el diagrama que se adjunta la evolución temporal de ejecución de instrucciones en cada procesador. Como ejemplo se muestra la ejecución de algunas instrucciones.*

ciclos	P1	P2	P3	P4	BUS
1	L A1	L A2	L A3	L A4	
2	S C1				PtIm
3	1 ciclo				
4	2 ciclo				
5	L A5	S C2			PtIm
6		1 ciclo			
7		2 ciclo			CaC
8					

Pregunta 5: *Calcule el número de ciclos que tarda en ejecutarse el bucle*

Pregunta 6: *Calcule la ganancia respecto a la ejecución en un sólo procesador*

Ejercicio 4.14

El protocolo de coherencia que se utiliza es el denominado B. Este protocolo se modifica con el mecanismo de transacción de exclusividad, el cual ha sido descrito junto con los protocolos.

Las caches utilizan mapeo directo y el tamaño de bloque es de 2 palabras (2 variables).

Suponga la siguiente secuencia de accesos a las variables u y t que están contenidas en bloques distintos del espacio lógico y se mapean en el mismo contenedor de cache. El multiprocesador dispone de 2 procesadores.

accesos	accesos	memoria (valor inicial)	cache(almacena) C1	C2
1. P1 lee t	6. P2 escribe u (12)	u: 4	u: no	u: no
2. P2 lee u	7. P1 lee t	t: 5	t: no	t: no
3. P1 escribe t (21)	8. P2 lee t			
4. P2 escribe u (8)	9. P1 escribe t (18)			
5. P2 lee t				

Pregunta 1: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y t en las caches de cada procesador. Así mimo, muestre las transacciones de bus y quién suministra el bloque. En transacción de bus indique también el valor de las variables en memoria.*

	Bus	Memoria			Cache 1			Cache 2		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado

Ejercicio 4.15

Suponga un multiprocesador que tiene ocho procesadores con caches privadas. Cada cache tiene 1024 conjuntos y cada conjunto tiene asociatividad cuatro.

El multiprocesador utiliza como red de interconexión un bus y un mecanismo de observación para mantener la coherencia de cache. El protocolo de coherencia utiliza la técnica de invalidación y las caches privadas utilizan escritura retardada (MLI). En este multiprocesador un controlador de coherencia utiliza el tipo de petición que observa en el bus para determinar las comparaciones que debe realizar en la fase de observación de una transacción (Pt, PtIm, PtX). Tenga en cuenta en las respuestas que, en ocasiones, el mínimo puede ser igual al máximo.

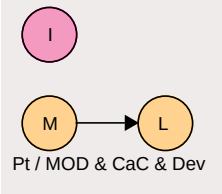
Pregunta 1: *En una cache indique el número mínimo y máximo de comparaciones que se efectúan en la fase de observación de una transacción.*

Pregunta 2: *En el multiprocesador indique el número mínimo y máximo de las comparaciones, que se efectúan en la fase de observación de una transacción, que pueden detectar coincidencia.*

Pregunta 3: *En el multiprocesador indique el número mínimo y máximo de respuestas (activación o desactivación de señales) en la fase de respuesta de una transacción.*

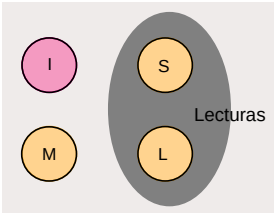
Ejercicio 4.16

En un protocolo de coherencia de cache con tres estados y del tipo invalidación, denominado MLI (M: Modificado, L: lectura, I: Inválido), siempre se actualiza memoria cuando una cache suministra el bloque y la transacción es de lectura. Posteriormente, si no ha existido otra escritura al bloque, en un fallo de cache la memoria suministra el bloque. Esta característica puede reducir el rendimiento en multiprocesadores donde la latencia de una transferencia entre caches sea menor que la latencia de acceso a memoria.



La idea es modificar el protocolo MLI para suministrar más veces desde cache estando memoria actualizada. Esto es, incrementar las transferencia entre caches. El nuevo protocolo tiene un estado adicional que indica suministro (S) y el protocolo se denomina MLIS. La implementación de esta idea se efectúa típicamente extendiendo el protocolo MELI y se denomina MELIS[36]. En este ejercicio nos centraremos exclusivamente en la idea, ya que no se ven afectadas las transiciones entre los estados adicionales.

El nuevo estado, etiquetado como S (suministrador), se utiliza para identificar la cache que suministra el bloque en un fallo de cache de otro procesador. Al estado S se llega exclusivamente al solicitar el bloque para lectura, después de un fallo de cache. Por tanto, el estado S de un bloque se transfiere entre caches a medida que solicitan el bloque. La esperanza subyacente es que la última cache donde se almacena el bloque sea la que lo expulse más tarde.



36. Las siglas en inglés son MESI y MESIF respectivamente, M: Modified, E: Exclusive, S: Shared, I: Invalid, F: Forwarding

Cuando, en un fallo de lectura, una cache en estado M suministra un bloque la memoria se actualiza y en la cache que ha generado la transacción el bloque se almacena en el estado es S.

Para contestar a las siguientes preguntas analice el protocolo de coherencia MLI. La mayoría de transiciones en el protocolo MLIS son idénticas.

Pregunta 1: *Describa, utilizando una tabla, para cada estado del protocolo de coherencia MLIS el posible estado del bloque en otras caches y memoria.*

En el protocolo MLIS, igual que en el MLI, se necesita conocer cuándo una cache suministra el bloque. Sin embargo, en el protocolo MLIS el suministro se efectúa cuando el bloque está en el estado M y en el estado S. Para el caso del estado M, como en el protocolo MLI, se utiliza la información denominada MOD que indica que una cache suministra. Para el caso del estado S se utiliza otra información denominada SUM. Notemos que en los dos casos de suministro sólo se actualiza memoria cuando se activa la señal MOD, si es el caso.

Pregunta 2: *Muestre de forma funcional las transiciones entre los estados de un bloque en un diagrama de estados. Suponga eventos del procesador (LPr, EPr) y eventos de otros procesadores (LPr_{otro}, EPr_{otro}). Indique si existe acción de suministro mediante MOD o SUM.*

El protocolo de coherencia MLIS se implementa en un multiprocesador que utiliza como red de interconexión un bus. En este bus, además de la señal disponible en el protocolo MLI (MOD), existe otra señal denominada SUM. En una transacción de bus, la señal SUM se activa/desactiva en la fase de respuesta de observación. Todas las caches están conectadas a la señal SUM y en el bus se efectúa la operación OR de las señales individuales. Una cache activa la señal SUM si tiene el bloque en estado S. Cuando se activa la señal SUM la memoria se inhibe de suministrar el bloque, como ocurre cuando se activa la señal MOD.

Las peticiones de procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificarlo	SUM: bloque en estado S, suministra el bloque	
	PtX: petición de actualización de memoria	MOD: bloque en estado M, suministra el bloque	
		CaC: suministro del bloque	

Pregunta 3: *Muestre el diagrama de transiciones entre los estados de un bloque en el protocolo de coherencia MLIS cuando la red de interconexión es un bus. Para ello utilice varios diagrama parciales. Considere operaciones de lectura, operaciones de escritura, acción de reemplazo y observaciones de forma separada.*

Suponga la siguiente secuencia de accesos a las variables u y r. Las variables u y r están contenidas en los bloques que denominamos A y B. Al mapearse en cache los dos bloques ocupan el mismo contenedor de cache.

accesos	accesos
1. P1 load u	6. P2 load r
2. P2 load r	7. P3 store u (65)
3. P1 store u (17)	8. P2 load u
4. P2 store r (8)	9. P1 load u
5. P3 store u (24)	10. P2 load r

memoria (valor inicial)	cache(almacena) C1	C2	C3
u: 4	u: no	u: no	u: no
r: 13	r: no	r: no	r: no

Pregunta 4: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables u y r en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

Pregunta 5: *En la secuencia de accesos a memoria mostrada previamente indique: a) número de expulsiones debido a reemplazos, b) número de expulsiones que requieren actualizar memoria, c) número de aciertos en cache, d) número de suministros desde el estado M y e) número de suministros desde el estado S.*

Suponga que cuando el bloque es suministrado por memoria el número de ciclos de bus de una transacción es NMC y cuando el bloque es suministrado por una cache el número de ciclos de bus de una transacción es NCC (NCC < NMC). En el caso de una expulsión suponga que el número de ciclos de una transacción es NCC.

Pregunta 6: *Calcule la ganancia del protocolo MLIS respecto del protocolo MLI utilizando la secuencia previa de accesos a memoria.*

Ejercicio 4.17

El protocolo de coherencia que se utiliza es el denominado C.

El lenguaje máquina de un procesador dispone de las primitivas Load Linked (LL) y Store Conditional (SC).

instrucción	descripción
LL Rd, dir	la palabra leída en la posición de memoria dir se almacena en el registro Rd y se activa el bit de enlace (bit=1). En el registro de enlace se almacena el valor dir
SC Rfd, dir	si el bit de enlace está activo (bit=1) el contenido del registro Rfd se almacena en la posición de memoria dir y se devuelve un 1 en el registro Rfd. En caso contrario no se ejecuta el store y se devuelve un 0 en el registro Rfd.

El bit de enlace se desactiva si el bloque que contiene la palabra referenciada en la instrucción LL se invalida.

Suponga que la variable llave se utiliza para garantizar acceso excluyente. El código de las primitivas obtener (llave) y liberar (llave), implementadas con las instrucciones Load Linked y Store Conditional es:

	obtener (llave)	liberar (llave)
1$:	mov R2, #1	store #0, llave
	LL R1, llave	return
	SC R2, llave	
	beq R2, 1$	
	return	

Tres procesadores, P1, P2 y P3 acceden a una variable compartida B que está protegida mediante una variable llave. Considere el siguiente entrelazado de referencias a la variable llave al competir los tres procesadores por el acceso a la zona de exclusión.

accesos	accesos
1. P1 LL llave	7. P2 LL llave
2. P3 store llave	8. P1 SC llave
3. P2 LL llave	9. P1 LL llave
4. P1 SC llave	10. P2 SC llave
5. P1 LL llave	11. P2 LL llave
6. P2 SC llave	

donde los números indican orden. Es decir, no se procesan en paralelo dos referencias correspondientes a procesadores distintos. Suponga que inicialmente ninguna cache almacena el bloque que contiene la variable llave.

Pregunta 1: *Suponga para esta pregunta que el controlador de coherencia gestiona una instrucción SC como una instrucción Store, independientemente del valor del "bit de enlace". Esto es, genera una transacción de bus para obtener la exclusividad, si es el caso. Muestre que en la anterior secuencia entrelazada de accesos a memoria ninguno de los procesadores (P1 y P2) obtiene la llave después de que el procesador P3 la libere. Esta situación se denomina falta de vivacidad*

(livelock). Para ello utilice la tabla que se adjunta, donde se especifica la secuencia de estados de los bloques en cada cache y transacciones que se producen en el bus.

	Bus		Cache 1		Cache 2		Cache 3	
acceso	trans.	suministro	bit	estado	bit	estado	bit	estado
				I		I		I

Pregunta 2: *Nota: en suministro indique si el dato lo suministra memoria o una cache. En estado indique el estado del bloque y en bit indique el valor del bit de enlace (0: no existe enlace, 1: existe enlace).*

Pregunta 3: *Muestre que no se produce falta de vivacidad si el controlador de coherencia no genera una transacción de bus cuando el bit de enlace está desactivado. Para ello utilice una tabla como la indicada en la pregunta anterior. Indique el número de acceso a memoria donde un procesador obtiene el acceso exclusivo. Note que a partir de este acceso a memoria la secuencia de accesos no es consistente. Por tanto, no siga rellenando la tabla.*

Ejercicio 4.18

El protocolo de coherencia que se utiliza es el denominado C. Al protocolo se le añade el mecanismo denominado lectura con actualización, descrito junto con los protocolos.

El lenguaje máquina de los procesadores dispone de la instrucción atómica fetch&inc(M) que incrementa en una unidad el contenido de la posición de memoria M y devuelve el valor que almacenaba previamente.

```
fetch&inc (M)
tmp = M
M = tmp + 1
return tmp
```

La instrucción fetch&inc se ejecuta en un módulo de memoria especializado para este tipo de operaciones atómicas. El propio módulo se encarga de incrementar la posición de memoria; es decir, no existe flujo de información para que el procesador actualice la posición de memoria mediante una operación. Esta posición de memoria no se mapea en cache en ningún caso y por tanto, no actúa el mecanismo de coherencia. Para ello, previamente se ha marcado la página que contiene la posición de memoria como no cacheable. Esto es, el acceso a esta variable siempre representa sólo una transferencia de bus, ya que se puede implementar como un load en el que los bits menos significativos de la dirección codifican la operación que se quiere efectuar.

La instrucción atómica fetch&inc se utiliza en la implementación de una barrera, cuyo código simplificado, para el propósito de la evaluación que se propone, es el siguiente.

```
    barrera (M)
if (fech&inc (M) = P-1) then
    M = 0
    aviso = 1
else
    repeat
    until (aviso = 1)
endif
return
```

donde P es el número de procesadores que ejecutan la instrucción barrera.

Suponga que cuatro procesadores ejecutan un bucle paralelo y que para sincronizarse antes de proseguir la ejecución utilizan una barrera. La siguiente secuencia de referencias muestra un posible entrelazado de referencias a memoria de los procesadores.

accesos	accesos
1. P1 fetch&inc M	7. P4 fetch&inc M
2. P1 load aviso	8. P4 store aviso
3. P2 fetch&inc M	9. P1 load aviso
4. P2 load aviso	10. P2 load aviso
5. P3 fetch&inc M	11. P3 load aviso
6. P3 load aviso	

donde fetch&inc indica la ejecución de la instrucción atómica en el módulo de memoria, load aviso representa la lectura relativa al bucle de espera y store aviso es la actualización que efectúa el último procesador que llega a la barrera.

Suponga que la instrucción atómica fetch&inc se implementa mediante un load. Por tanto, al ejecutar el procesador este load se produce un fallo de cache en un acceso de tipo lectura.

En la tabla que se solicita, cuando en una lectura se actualicen varias cache (lectura con actualización) indique las modificaciones de estado en la linea de la tabla correspondiente al procesador que efectúa la lectura.

Las variables M y aviso están ubicadas en bloques distintos de memoria. Suponga que inicialmente las caches de los procesadores P1, P2, P3 y P4 no almacenan el bloque que contiene la variable aviso y que no la han referenciado nunca.

Pregunta 1: *Cuando se considera el entrelazado de las referencias de los procesadores P1, P2, P3 y P4 mostrado anteriormente, muestre en la tabla que se adjunta, la secuencia de estados del bloque que contiene la variable aviso en cada cache y las transacciones en el bus correspondientes a los bloques que contienen las variables aviso y M.*

	bus		mem.		C 1			C 2			C 3			C 4		
acceso	trans.	señal (MOD, C)	var.	sum.	cont.	var.	est.	cont.	var.	est.	cont.	var.	est.	cont.	var.	est.
1. P1 fetch&inc																

Pregunta 2: *A partir de los resultados de los apartados anteriores evalúe el tráfico de bus generado cuando P procesadores deben sincronizarse mediante una barrera. Para ello indique en la siguiente tabla el número de transacciones de bus.*

	Transacciones (Pt, PtIm)
no cacheables	
cacheables	
TOTAL	

La frecuencia de reloj de los procesadores es 500MHz, ejecutan 4 instrucciones por ciclo y el IPC = (1 / CPI) efectivo de un procesador es 2.5. Así mismo, suponga que N es el número de instrucciones del programa. Al ejecutar el programa en el multiprocesador el número de instrucciones se distribuye uniformemente entre los procesadores.

El tiempo de ejecución de la barrera se asimila al número de transacciones de bus y supondremos que cada transacción de bus son 100 ciclos de procesador.

Pregunta 3: *Para evaluar la sobrecarga que representa la ejecución de una barrera, calcule el número mínimo de instrucciones (N) para que la ejecución paralela con cuatro procesadores sea provechosa frente a una ejecución serie.*

Ejercicio 4.19

El protocolo de coherencia que se utiliza es el denominado C.

El lenguaje máquina de los procesadores dispone de la instrucción atómica swap (M,R) que intercambia el contenido de la posición de memoria M con el contenido del registro R. Desde el punto de vista del protocolo de coherencia se comporta como un store (cambios de estado y transferencias de bus).

swap (M, R)	comentario
load R2, M	M es una variable global
ltore R, M	R es un registro que almacena un valor
mov R, R2	
return	

Suponga que la variable llave se utiliza para garantizar acceso excluyente. El código de las primitivas obtener (llave) y liberar (llave), implementadas con la instrucción swap es:

	obtener (llave)	liberar (llave)
2$:	mov R2, #1	store #0, llave
1$:	load R1, llave	return
	bne R1, 1$	
	swap (llave , R2)	
	bne R2, 2$	
	return	

Suponga que el procesador P5 está en una zona de exclusión y los procesadores P1, P2, P3 y P4 están en espera activa. Seguidamente P5 libera la zona de exclusión y la siguiente secuencia de referencias muestra un posible entrelazado de referencias a memoria de los procesadores que compiten para obtener el acceso excluyente.

accesos	accesos
0. P5 store llave	6. P2 LOA/STO llave
1. P1 load llave	7. P3 LOA/STO llave
2. P2 load llave	8. P4 LOA/STO llave
3. P3 load llave	9. P2 load llave
4. P4 load llave	10. P3 load llave
5. P1 LOA/STO llave	11. P4 load llave

donde LOA/STO indica la ejecución indivisible de un LOAD y STORE sobre la posición de memoria llave y los números indican orden; es decir, no se procesan en paralelo dos referencias correspondientes a procesadores distintos.

En la anterior secuencia se ha supuesto que el arbitraje del bus concede el acceso a los procesadores que lo solicitan de forma cíclica; es decir, antes de volver a conceder el acceso a un procesador determinado debe de haberse concedido el acceso a todos los otros procesadores que lo solicitan.

Suponga que inicialmente las caches de los procesadores P1, P2, P3 y P4 almacenan el bloque que contiene la variable llave en estado I y la cache del procesador P5 tiene el bloque en estado M ya que se acaba de ejecutar la referencia: 0. P5 store llave.

Pregunta 1: *Muestre en la tabla que se adjunta, la secuencia de estados del bloque en cada cache y transacciones que se producen en el bus, cuando se considera el entrelazado de las referencias de los procesadores P1, P2, P3 y P4 mostrada anteriormente.*

acceso	bus trans.	suministro	Cache 1 estado	Cache 2 estado	Cache 3 estado	Cache 4 estado	Cache 5 estado

El protocolo de coherencia denominado C se mejora utilizando el mecanismo de lectura con actualización, el cual ha sido descrito junto con los protocolos.

Pregunta 2: *Para lectura con actualización muestre en una tabla, la secuencia de estados del bloque en cada cache y transacciones que se producen en el bus, cuando se considera el entrelazado de las referencias de los procesadores P1, P2, P3 y P4 mostrado anteriormente. Cuando en una lectura se actualicen varias cache (lectura con actualización) indique las modificaciones de estado en la linea de la tabla correspondiente al procesador que efectúa la lectura.*

Pregunta 3: *A partir de los resultados de los apartados anteriores, evalúe el tráfico de bus generado cuando un procesador libera el acceso y hay P procesadores que compiten por acceder a la zona de exclusión, hasta que se llega a una situación estable en que P-1 procesadores están en espera activa. Para ello indique en una tabla el número de transacciones de bus en cada caso. La fase de competencia se corresponde desde la referencia 1 hasta la 8 ambas inclusive y la de espera activa el resto.*

Ejercicio 4.20

El protocolo de coherencia que se utiliza es el denominado B.

Por competencia para obtener una llave se entiende el comportamiento de los hilos, que están esperando obtener la llave, cuando un hilo libera la llave.

Para observar el comportamiento utilizaremos la siguiente implementación de la primitiva adquirir.

	adquirir (R1, R4)	Comentarios
1$:	LL R4, 0(R1)	LL: load con enlace
	bne R4, 1$	SC: almacenamiento condicional
	mov R3, #1	
	SC R3, 0(R1)	
	beq R3, 1$	
	return	

Suponga la siguiente secuencia de acceso correspondiente a varios hilos que están en espera activa e intentar obtener una llave cuando es liberada. El acceso exclusivo ha sido liberado por el procesador P4.

accesos	accesos	memoria (valor inicial)	cache (almacena) C1	C2	C3
1. P1 LL Rd, llave	7. P2 LL Rd, llave	llave: 0	llave: no	llave: no	llave: no
2. P2 LL Rd, llave	8.P3 LL Rd, llave				
3. P3 LL Rd, llave	9. P2 LL Rd, llave				
4. P1 SC Rfd, llave	10. P3 LL Rd, llave				
5. P2 SC Rfd, llave	11. P1 store Rd, llave				
6. P3 SC Rfd, llave					

Pregunta 1: *Muestre mediante una tabla las transacciones de bus y estado de los bloques en las caches. Muestre también el valor del bit de enlace en cada cache (RER).*

Pregunta 2: *Muestre mediante diagrama temporal las transacciones de bus y estado de los bloques en las caches. Muestre también el valor del bit de enlace en cada cache (RER).*

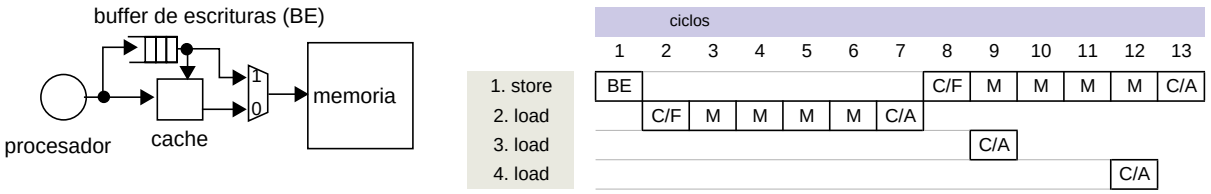
Ejercicio 4.21

Un procesador interpreta las instrucciones en orden de programa y utiliza una cache con escritura retardada. Para que las escrituras no bloqueen al procesador se utiliza un buffer denominado buffer de escrituras (BE). Una vez una instrucción store ha finalizado la ejecución en el procesador, en orden de programa, se almacena en el BE hasta que la jerarquía de memoria procese la instrucción. Esto es, cuando el bloque en la cache tiene los permisos necesarios, se extrae del BE y a la vez se actualiza el bloque en la cache. Cada entrada del BE tiene un campo de dirección y un campo de dato y el orden en que se almacenan varias instrucciones store en el BE mantiene el orden de programa.

El objetivo de un BE es soportar la latencia cuando es necesario acceder a memoria para obtener el bloque en exclusividad. El procesador, después de almacenar la instrucción store en el BE, sigue ejecutando instrucciones. Si la instrucción es de cálculo actualiza el estado del procesador. Si la instrucción es un store se almacena en el BE. Si la instrucción es un load y es acierto en cache se actualiza el estado del procesador. En caso de fallo se inicia el acceso a memoria y puede adelantar a una instrucción store almacenada en el BE.

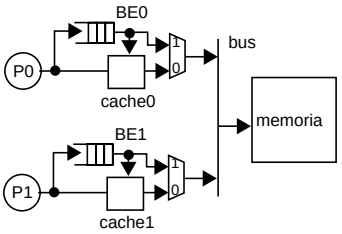
Para que el BE sea transparente a la arquitectura se añade circuitería a su alrededor. Si una instrucción load más joven accede a una posición de memoria almacenada en el BE, el procesador obtiene el valor de la instrucción store más joven que accede a la misma posición de memoria almacenada en el BE. Si el BE está lleno y se inicia la interpretación de una nueva instrucción store se bloquea la interpretación de instrucciones hasta que se ha vaciado el BE.

En la parte izquierda de la figura se muestra un esquema del camino de datos cuando se produce un fallo de lectura o una escritura (acierto o fallo). En la parte derecha de la figura se muestra un ejemplo donde se observa cómo una instrucción load, que falla al acceder a cache, adelanta a una instrucción store más vieja. Los acrónimos C, A, F y M indican respectivamente acceso a cache, acierto en cache, fallo en cache y acceso a memoria. También se muestran dos instrucciones load más jóvenes que aciertan en cache antes de que se extraiga el store del BE.



Pregunta 1: *Razone de forma sucinta si un procesador con un BE respeta las dependencias de datos al acceder a memoria (load y store).*

Este procesador se utiliza en el diseño de un multiprocesador cuya red de interconexión es un bus, las caches privadas utilizan escritura retardada y la coherencia se mantiene con un protocolo de invalidación (MLI). En la figura se muestra un esquema de un multiprocesador con dos procesadores.



Para extraer una instrucción store del BE, el bloque debe disponer de los permisos necesarios.

En las dos siguientes preguntas se limita la funcionalidad del BE descrito.

Pregunta 2: *Suponga que el BE sólo dispone de una entrada. Después de insertar una instrucción store en el BE, un procesador sigue interpretando instrucciones hasta que debe ejecutar una instrucción de acceso a memoria (load / store). En este caso, el procesador se bloquea si en el BE hay pendiente una actualización. Justifique de forma razonada si se mantiene consistencia secuencial.*

Después de insertar una instrucción store en el BE, un procesador sigue interpretando instrucciones. Si una de ellas es un store se almacena en el BE. Sin embargo, si interpreta una instrucción load (acierto o fallo) y en el BE hay pendiente alguna actualización el procesador se bloquea.

Pregunta 3: *Suponga que el BE dispone de varias entradas y la gestión del BE es FIFO. Esto es, la actualización de memoria y extracción del BE es en orden de programa. Justifique de forma razonada si se mantiene consistencia secuencial.*

Pregunta 4: *Suponga que el BE dispone de varias entradas y la gestión es no es FIFO. Esto es, si el bloque al que accede una instrucción store, cuando se inserta en el BE, tiene los permisos necesarios se actualiza inmediatamente la cache, aunque existan entradas ocupadas. Justifique de forma razonada si se mantiene consistencia secuencial.*

El protocolo de coherencia que se utiliza es el denominado B.

En el multiprocesador se ejecutan los siguientes dos esquemas de código que implementan un algoritmo para acceso exclusivo (izquierda, no se muestra la parte else). En el centro se muestra una simplificación del código para efectuar un análisis de consistencia secuencial. En la parte derecha se muestra el entrelazado de accesos que debe utilizarse en las preguntas.

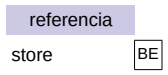
P 0	P 1	valores iniciales
aviso1 = 1 if (aviso2 = 0) R.Critica aviso1 = 0	aviso2 = 1 if (aviso1 = 0) R.Critica aviso2 = 0	aviso1 = 0 aviso2 = 0

Simplificación del código para efectuar un análisis de consistencia secuencial	
P0. store #1, aviso1	P1. store #1, aviso2
P0. load R4, aviso2	P1. load R6, aviso1

Entrelazado de accesos a memoria
P0 store #1, aviso1
P0 load R4, aviso2
P1 store #1, aviso2
P1 load R6, aviso1

Las variables aviso1 y aviso2 están contenidas en bloques distintos de memoria y su valor es cero. Al almacenarse en cache los bloques ocupan contenedores distintos. Inicialmente la cache C0 almacena la variable aviso1 en estado L y la cache C1 almacena la variable aviso2 en estado L.

En los diagramas temporales que se soliciten un almacenamiento en el BE se representa de la forma que se muestra en la figura.

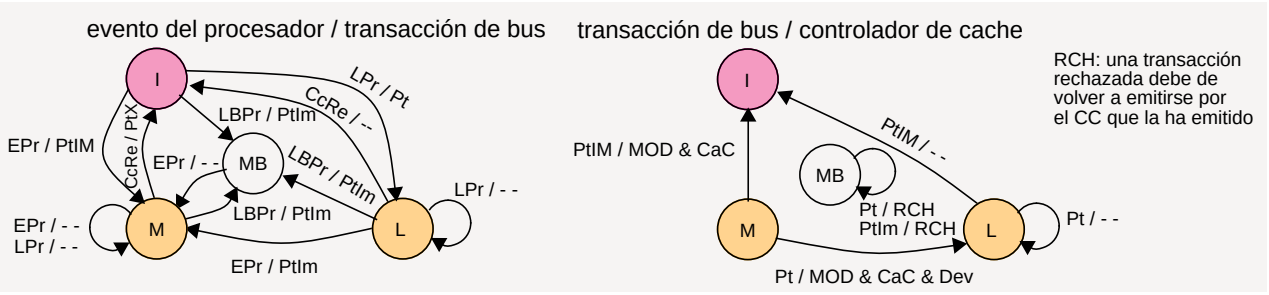


En las preguntan que restan se supone que el BE tiene la misma funcionalidad que en el caso descrito de un procesador. En estas condiciones, en cada procesador pueden competir por el bus: a) la entrada en la cabeza del BE y b) una instrucción load que ha detectado un fallo al acceder a cache. En este caso, el fallo de un load es prioritario.

Pregunta 5: *Muestre, mediante un diagrama temporal, que este multiprocesador no cumple las condiciones de consistencia secuencial al ejecutarse el entrelazado de accesos a memoria previo. La traza de accesos finaliza con las instrucciones store en los BE. Razone que no se cumple consistencia secuencial a la vista del resultado (valores almacenados en R4 y R6).*

El lenguaje máquina de los procesadores dispone de la instrucción atómica swap (swap r_{f1}, X(r_{f2})). La instrucción swap lee y actualiza (con el contenido de r_{f1}) de forma atómica una posición de memoria (X(r_{f2})). La parte store de la instrucción se ubica en el BE y la parte load utiliza el camino de datos de una instrucción load. Ahora bien, como la instrucción es atómica debe ejecutarse de forma indivisible. En consecuencia, antes de ejecutar la parte load hay que vaciar las entradas del BE ocupadas por instrucciones store previas, ya que el BE se gestiona de forma FIFO. En otras palabras, el procesador se bloquea hasta que se han vaciado las entradas previas en el BE.

Para soportar la instrucción atómica swap se añade otra petición del procesador en el CC, denominada Lectura con Bloqueo (LBPr). El diagrama de cambio de estados es el siguiente. Mientras se está en el estado MB se rechaza cualquier transacción de bus que acceda al mismo bloque.



Un programador propone sustituir las instrucciones store en el código previo por instrucciones swap.

En un diagrama temporal utilice 3 filas para mostrar una instrucción swap. En la primera fila indique el almacenamiento en el BE. En la segunda fila indique la parte load y en la tercera fila indique la parte store.

Pregunta 6: *Muestre, mediante el mismo entrelazado de accesos y con la sustitución descrita, que se cumplen las condiciones de consistencia secuencial al ejecutarse los códigos previos.*

Ejercicio 4.22

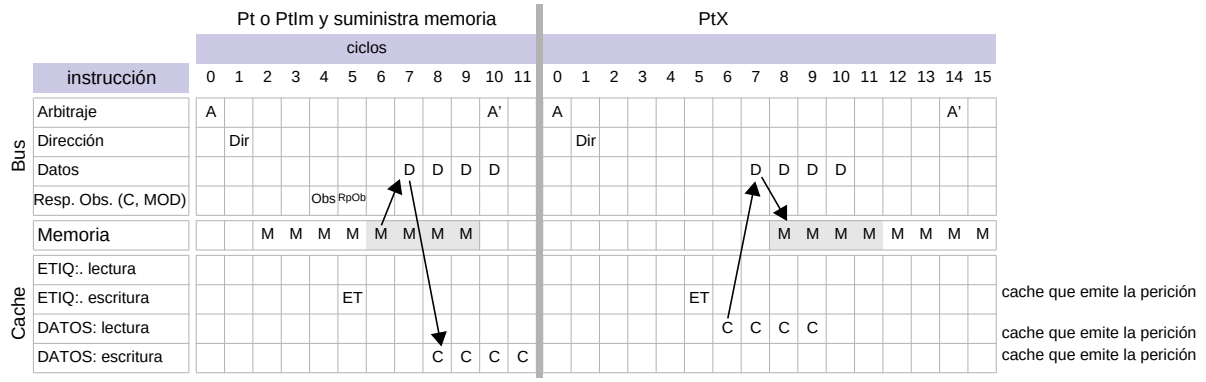
El protocolo de coherencia que se utiliza es el denominado B.

El multiprocesador utiliza un bus segmentado, en el cual todas las fases (abitraje (A), observación (Obs), respuesta de observación (RpOb), ...) se efectúan en un tiempo prefijado relativo al tiempo en el cual se inicia la petición.

El tiempo de ciclo del procesador son 5ns y la duración de una transacción para acceder a memoria (M) son 200ns, que son 10 ciclos de bus (t_{bus} = 20 ns). La frecuencia de reloj del sistema de memoria es 100 Mhz. Por tanto, un ciclo de bus son dos ciclos del reloj de sistema.

Los cables del bus que transportan los datos permiten transmitir 64 bits en cada ciclo (D) y el tamaño de bloque son 32 bytes. Si la memoria no está ocupa se puede arbitrar cada 10 ciclos (A', en la siguiente figura).

Los ciclos concretos en los cuales se realizan las operaciones y se utilizan los recursos (memoria, campos de etiquetas de cache (ETIQ) y campos de datos de cache (DATOS)) se muestran en los siguientes diagramas temporales para cada una de las transacciones.

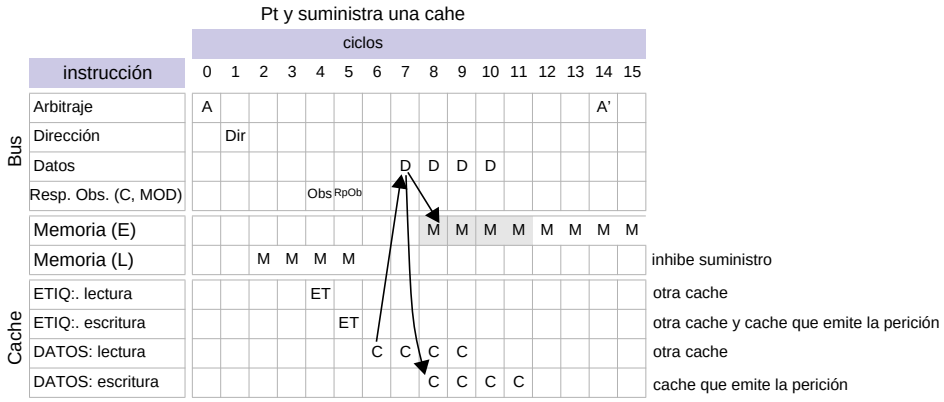


En una petición Pt siempre se accede a memoria aunque posteriormente no suministre el bloque, ya que lo suministra una cache. En una petición PtIm el comportamiento es el mismo que en una petición Pt, pero las caches involucradas en la acción de coherencia invalidan la copia del bloque. En una petición PtX se lee el bloque de cache y se actualiza memoria.

Notemos que se utilizan los mismos ciclos de bus, relativos al inicio de la transacción, tanto para transmitir los datos en una petición Pt o PtIm, como en una petición PtX.

En los diagramas temporales se indica como A’ el primer posible ciclo de arbitraje de la siguiente petición, si existe. El arbitraje de una petición se puede solapar con el último ciclo de la transferencia de datos de la transacción previa, si no se produce conflicto de recursos. La memoria está ocupada durante 8 ciclos (M) y se dispone de los datos leídos durante los últimos 4 ciclos. Los datos se transmiten por el bus durante 4 ciclos (D) y se almacenan en cache en el ciclo siguiente al de transmisión (C). En la figura previa se muestra, mediante flechas, el flujo del primer paquete de datos.

En una petición Pt, cuando el bloque lo suministra una cache, debe actualizarse memoria. El siguiente diagrama temporal muestra la ocupación de los recursos. Las filas correspondientes a ETIQ y “Datos:lectura” se corresponden con la cache que suministra el bloque.



Mientras no se indique lo contrario no se utiliza la capacidad de solapar transacciones que permite un bus segmentado.

Pregunta 1: *Suponiendo una ráfaga de transacciones consecutivas Pt, calcule el número de datos transmitidos por unidad de tiempo (ancho de banda en Mbytes/seg.). El bloque lo suministra memoria.*

Pregunta 2: *Suponiendo una ráfaga de transacciones consecutivas Pt, calcule el número de datos transmitidos por unidad de tiempo (ancho de banda en Mbytes/seg.). El bloque lo suministra una cache.*

Pregunta 3: *Deduzca, a partir de los diagramas de ocupación de recursos mostrados, el diagrama para una petición PtIm, cuando el bloque lo suministra otra cache. Indique también cuándo se arbitra para la siguiente transacción. Suponiendo una ráfaga de transacciones consecutivas PtIm, calcule el número de datos transmitidos por unidad de tiempo (ancho de banda en Mbytes/seg.).*

Pregunta 4: *Suponga una ráfaga de transacciones consecutivas donde se entrelazan peticiones Pt y PtX (alternandose). Justifique si con un módulo de memoria se puede obtener el mismo ancho de banda que con una ráfaga de peticiones consecutivas Pt . Para ello calcule el ancho de banda.*

Suponga la siguiente secuencia de accesos correspondiente a varios hilos.

accesos	memoria	cache (almacena)		
		C1	C2	C3
1. P1 load A 2. P2 load A 3. P2 store A 4. P3 load A 5. P1 store B 6. P2 store B	Las variables A y B están contenidas en bloque distintos Estos bloques al ubicarse en cache no producen conflictos	A, B: no	A, B: no	A, B: no

Pregunta 5: *Muestre, mediante una tabla, el estado de los bloques que contienen las variables A y B en las caches de cada procesador y el contenedor de cache donde se almacena. Además, muestre en cada transacción si la cache activa (1) o deactiva (0) la señal MOD. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

Pregunta 6: *Teniendo en cuenta los diagramas de ocupación de recursos, calcule el número de ciclos que tarda la secuencia de accesos previa.*

Al sistema multiprocesador se le añade otro módulo de memoria. Esto es, dispone de dos módulos de memoria. Los bloque se almacenan de forma entrelazada utilizando el bit menos significativo de la dirección de bloque.

Para utilizar al máximo el bus se permiten dos transacciones concurrentes cuando los bloques se almacenen en módulos de memoria distintos. Este diseño garantiza, de forma conservadora, que transacciones concurrente referencian bloques distintos. Por tanto, entre dos transacciones concurrentes no hay dependencias.

El arbitraje de peticiones a un módulo de memoria está distanciado como mínimo 10 ciclos de reloj, si lo permite la ocupación de memoria. Se solapa el ciclo de arbitraje con el último ciclo de transferencia de datos por el bus de la transacción previa.

Las transacciones que se muestran son peticiones Pt

	instrucción	ciclos 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Bus	Arbitraje	A0					A1					A0'					A1'		
	Dirección		Dir0					Dir1											
	Datos								D0	D0	D0	D0		D1	D1	D1	D1		
	Resp. Obs. (C, MOD)					Obs0	RpOb0				Obs1	RpOb1							
	Módulo de memoria 0			M0	M0	M0	M0	M0	M0	M0	M0								
	Módulo de memoria 1								M1	M1	M1	M1	M1	M1	M1	M1			

Entre dos transacciones de bus tiene que existir un ciclo en el cual no se transmiten datos (línea etiquetada como Datos en el grupo de recursos del Bus). Entonces, la distancia mínima entre dos arbitrajes a módulos de memoria distintos son 5 ciclos.

Pregunta 7: *Suponga que los bloques que contienen las variables A y B se almacenan en módulos de memoria distintos. Calcule el número de ciclos que tarda la secuencia de accesos previa.*

Pregunta 8: *Suponiendo una ráfaga de transacciones consecutivas Pt a bancos de memoria distintos de forma alternada, calcule el número de datos transmitidos por unidad de tiempo (ancho de banda en Mbytes/seg.). El bloque lo suministra memoria.*

Pregunta 9: *Suponiendo una ráfaga de transacciones consecutivas Pt a bancos de memoria distintos de forma alternada, calcule el número de datos transmitidos por unidad de tiempo (ancho de banda en Mbytes/seg.). El bloque lo suministra una cache.*

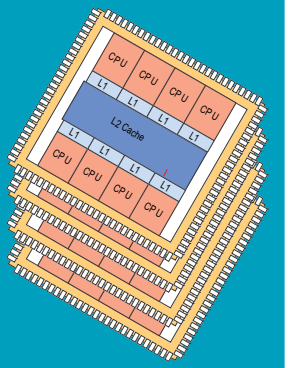
Pregunta 10: *Justifique si es necesario que el almacenamiento de los datos en la cache disponga de dos puertos lógicos (cada uno permite lectura y escritura) o es suficiente con un puerto que permita leer y escribir. Para ello analice una petición Pt de un CC1 donde el bloque lo suministra un CC2 y una petición Pt del CC2 y el bloque lo suministra memoria. Los bloques accedidos en cada transacción se ubican en módulos de memoria distintos.*

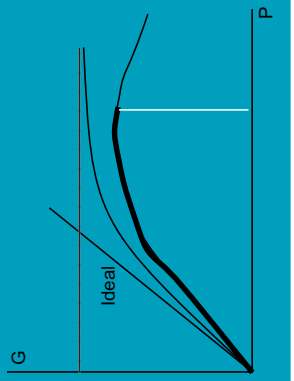
Pregunta 11: *Si es necesario más de un puerto lógico de acceso, proponga una organización de la cache que no requiera varios puertos físicos a la misma agrupación de celdas de memoria.*

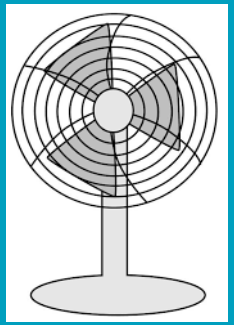
Pregunta 12: *Justifique si los bloques que se almacenan en un conjunto de cache se ubican en el mismo módulo de memoria o en distinto módulo de memoria.*

Pregunta 13: *Determine la latencia de servicio de un acceso a memoria que requiere liberar un contenedor de cache. El bloque seleccionado por el algoritmo de reemplazo está en el estado M.*

.







Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 5
Protocolo de observación en un bus atómico

En el Capítulo 4, cuando un procesador requiere acceder a memoria y no obtiene el acceso al bus suspende la interpretación de instrucciones[1]. En este capítulo, un procesador, que requiere acceder a memoria y no obtiene el bus, sigue interpretando instrucciones mientras: a) no se produzca un riesgo estructural, b) no se produzca un riesgos de datos o c) inicie la interpretación de una instrucción de acceso a memoria[2].

Por otro lado, en el Capítulo 4 se producen riesgos estructurales entre el agente procesador y el agente observador de un CC. Se ha supuesto que los recursos del camino de datos de acceso a la cache sólo permiten un acceso. La decisión que se ha tomado en el Capítulo 4, para gestionar el riesgo, ha sido que el acceso del agente observador es prioritario y se suspende la interpretación de instrucciones. En este capítulo se añaden recursos para reducir los ciclos perdidos por riesgos estructurales. Un CC puede estar sirviendo un acierto en cache, en la mayoría de las situaciones, mientras el agente observador del CC está efectuando una acción de observación.

En el Capítulo 4 no existen peticiones pendiente de obtener acceso al bus, ya que un acceso a memoria es una acción atómica. En este capítulo pueden existir peticiones pendientes de obtener el acceso al bus. Esta característica da lugar a que se cree una ventana temporal entre la petición (necesidad) de acceder al bus y la obtención del acceso al bus (Figura 5.1). Por tanto, a diferencia del Capítulo 4, un acceso a memoria no puede considerarse una petición atómica (acceso a cache y si es necesario acceso al bus y a memoria).

1. En concreto, reinterpreta la instrucción de acceso a memoria e instrucciones más jovenes hasta obtener el acceso al bus. Este mecanismo se utiliza para que todas las acciones en un acceso a memoria se efectúen de forma atómica: a) acceso a la cache y b) utilización del bus y acceso al módulo de memoria.
2. Este funcionamiento se relaja en algunos ejercicios. Por ejemplo, en un procesador se soportan aciertos en cache, cuando se dispone de los derechos de acceso necesarios, existiendo un acceso a memoria pendiente.

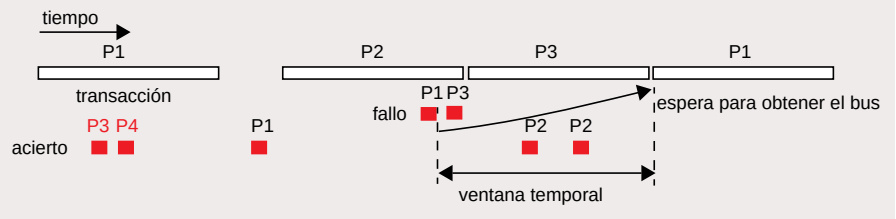


Figura 5.1 Ventana temporal entre petición de acceso al bus y concesión del bus para iniciar la transacción.

Por ejemplo, dos controladores de coherencia (CC1 y CC2), en un protocolo de invalidación con escritura retardada, solicitan el bus en el mismo ciclo. Uno de ellos obtiene el bus (CC1) y el otro debe esperarse (CC2). Mientras el CC2 está pendiente de obtener acceso al bus, debe servir el bloque de datos solicitado en la transacción que ocupa el bus (debida a CC1). Esto es, el agente observador del CC2 debe servir peticiones, mientras el agente procesador del CC2 espera obtener el bus para iniciar una transacción. Además, si las peticiones de los dos controladores de coherencia (CC1 y CC2) acceden al mismo bloque, es posible que las acciones previstas, para la petición pendiente de obtener el bus (CC2), deban modificarse en función de la transacción en curso. Esta última posibilidad da lugar a que, usualmente, la ventana temporal se denomine ventana de vulnerabilidad.

En este Capítulo una transacción de bus es atómica. Esto es, una transacción ocupa el bus desde que se le concede a un CC hasta que finaliza la transacción[3].

Ejercicio

En un protocolo de invalidación con escritura inmediata, describa una situación donde exista una ventana de vulnerabilidad, cuando se considera la posibilidad de peticiones pendientes de obtener el bus.

Respuesta

Un bloque está en estado válido en dos procesadores y los dos procesadores ejecutan concurrentemente una instrucción de escritura a una palabra del mismo bloque. Los dos controladores de coherencia solicitan el bus en el mismo ciclo. El controlador de coherencia del procesador P1 (CC1) obtiene el bus y el controlador de coherencia del procesador P2 (CC2) debe esperarse. La ventana de vulnerabilidad está en CC2. El bus ordena en primer lugar la petición de CC1. Por tanto, la copia del bloque en la cache del procesador P2

3. En los ejercicios se relaja la hipótesis. El bus se utiliza de forma segmentada y existen transacciones concurrentes. Ahora bien, en un primer paso, se garantiza que dos transacciones concurrentes no acceden al mismo bloque. Posteriormente, se analiza el caso de transacciones concurrentes al mismo bloque.

debe invalidarse. En consecuencia, al obtener el CC2 el bus, el estado del bloque es distinto del estado que tenía cuando se efectuó la petición de bus. En ese instante de tiempo se detectó acierto en cache.

En la descripción de las deficiencias en los apartados genéricos no se tiene en cuenta un único protocolo (VI, MLI). Los comentarios y mecanismo descritos pueden ser aplicables a uno de los protocolos descritos en el Capítulo 4 o a ambos. La aplicación concreta a cada protocolo se efectúa en el apartado dedicado al protocolo.

RECURSOS NECESARIOS EN UNA TRANSACCIÓN DE BUS

En un multiprocesador, en una transacción de bus distinguimos varias fases (Figura 5.2): arbitraje (ARB), difusión de la petición (@), observación (Obs), respuesta de observación (ROb) y transmisión de los datos (bloque).

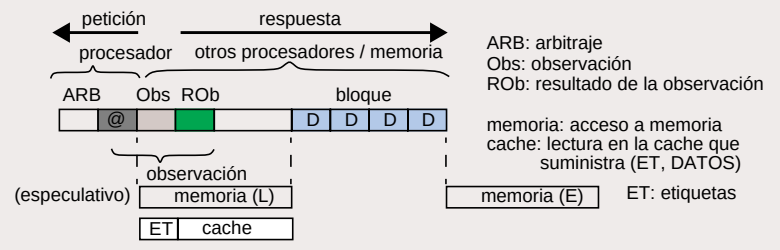


Figura 5.2 Fases en una transacción de bus.

En la Figura 5.3 se muestra un esquema de un multiprocesador donde se identifican las señales en el bus: a) señales de arbitraje, b) dirección (DIR), c) datos (DAT) y c) respuesta de observación (ROb, parada). Notemos que la mayoría de elementos del bus se han dibujado en ambos lados de los nodos, para que la mayoría de las señales fluyan de izquierda a derecha.

Arbitraje (ARB). Los CC solicitan acceso al bus y el árbitro concede el acceso a uno de ellos. Son necesarias señales de petición de bus, el árbitro y señales de concesión de bus.

Difusión de la petición (DIR). Transmisión por el bus de la dirección del bloque accedido y del tipo de acceso además de otras señales de control[4]. Para transmitir por el bus las señales descritas se utilizan conjuntos de cables.

Observación (Obs). Los CC comprueban si el bloque, cuya dirección se transmite por el bus, está almacenado en su cache. Es necesario acceder a los campos etiqueta y estado de la cache. El acceso del agente observador a estos recursos entra en competencia con un posible acceso del agente procesador.

4. No mostradas en la figura.

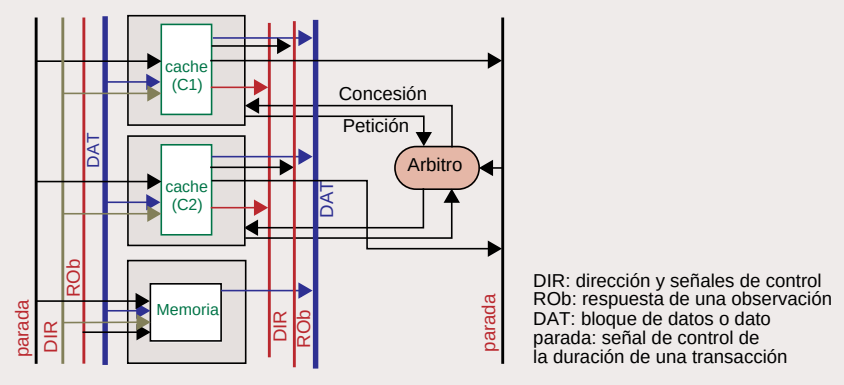


Figura 5.3 Esquema de los elementos utilizados en una transacción de bus.

Respuesta de la observación (ROb, parada). En función de si el bloque está almacenado en cache y de su estado, los controladores de cache emiten una respuesta en las señales del bus dedicadas para este menester (Figura 5.4). En esta fase un CC también puede activar la señal de parada de la transacción cuando debe suministrar el bloque y determina que no puede hacerlo en los ciclos prefijados de la transacción.

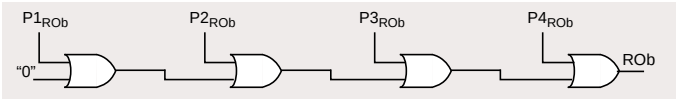


Figura 5.4 Señal de respuesta que es la función OR de las señales activadas por cada procesador.

En la fase de respuesta de cada CC se considera implícita la confirmación de la invalidación al finalizar la transacción, si es el caso. Por otro lado, la memoria u otro CC suministra el bloque solicitado en función del protocolo de coherencia que se utiliza. Cuando el agente observador necesita acceder al campo de datos de la cache, puede entrar en competencia con un posible acceso del agente procesador.

Transmisión de datos (DAT). Por las señales dedicadas en el bus, para este propósito, se transmite: a) el bloque solicitado, b) el dato o c) bloque que se almacenará en memoria. Suponemos que la ocupación del bus es la misma tanto en acciones de lectura de bloque como de actualización de memoria.

Cuando se expulsa de una cache un bloque actualizado se actualiza memoria y la confirmación de memoria se considera implícita al finalizar la transacción de bus.

El bus es síncrono y todas las transacciones tienen la misma duración a menos que se especifique lo contrario.

Riesgos estructurales en el camino de datos de acceso a la cache

El análisis se efectúa para la fase de observación y la fase de respuesta.

Fase de observación

En la fase de observación el agente observador puede tener un riesgo estructural con el agente procesador. El agente observador necesita leer el campo de etiqueta y el campo de estado y efectuar una comparación de direcciones (Figura 5.2)[5]. Para eliminar este riesgo estructural, el cual determina ciclos perdidos, se duplica el campo etiqueta, el campo estado[6] y la lógica de comparación necesaria (Figura 5.5). La copia es accedida para lecturas por el agente observador. Notemos que, mediante el duplicado, hay dos caminos de lectura, uno para cada agente, pero sólo uno de escritura.

En un fallo de cache hay que actualizar las dos copias del campo de etiqueta y el campo de estado. También, en una acción de observación, que determine modificar el estado, hay que modificar las dos copias del campo estado.

El duplicado de los campos etiqueta y estado filtra la mayoría de las observaciones. Esto es, transacciones de otros procesadores que referencian bloques que no están almacenados en la cache o bloques cuyo estado no necesita modificarse.

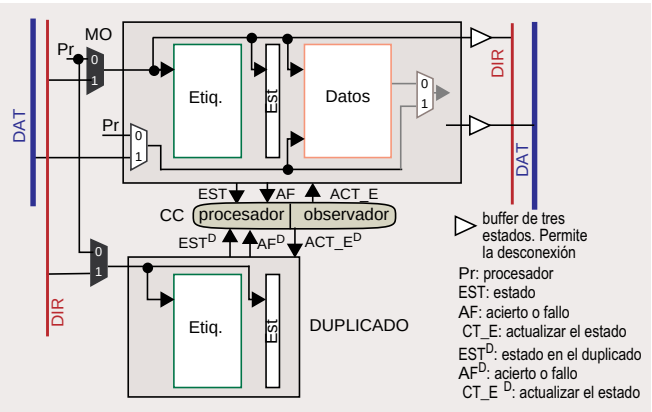


Figura 5.5 Reducción de riesgos estructurales: duplicación del campo etiqueta y estado de la cache.

5. El agente procesador puede estar interpretando una instrucción de acceso a memoria.
6. El campo estado usualmente se implementa utilizando una única estructura con dos puertos de acceso.

Fase de respuesta

En la fase de respuesta a una observación puede ser necesario, dependiendo del protocolo de coherencia, que una cache suministre el bloque. Por tanto, hay que leer en el campo de datos de la cache. Esta circunstancia puede producir un riesgo estructural entre el agente observador y el agente procesador. Cuando no es posible suspender la interpretación de instrucciones se utiliza una señal de "parada", disponible en el bus. Esta señal mantiene la transacción suspendida hasta que el agente observador puede acceder al campo de datos de la cache y suministrar el bloque (Figura 5.2)[7]. En otras palabras, se alarga o extiende la duración de una transacción (Figura 5.6)[8].

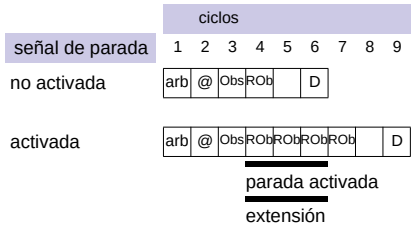


Figura 5.6 Señal de parada para extender la duración de una transacción de bus. Activación y observación en la fase ROb.

VENTANA TEMPORAL ENTRE PETICIÓN E INICIO DE LA TRANSACCIÓN

En este capítulo se permite que existan peticiones pendientes de utilizar el bus mientras el bus está ocupado. Esta característica da lugar a la existencia de una ventana temporal entre la necesidad de acceder al bus (petición) y la obtención del bus. Por tanto, un acceso a memoria, que utiliza el bus, no puede considerarse atómico.

Durante la ventana de tiempo las transacciones de bus que se observan pueden referenciar:

- un bloque distinto que el bloque que referencia la petición que está esperando obtener el bus.
- el mismo bloque que el bloque que referencia la petición que está esperando obtener el bus (ventana de vulnerabilidad).

7. La señal de parada se activa cada ciclo. También puede indicar que se mantiene la transacción suspendida un número determinado de ciclo. Cuando no es suficiente una activación se puede volver a activar al finalizar el periodo de suspensión.
8. Si el bus está segmentado de la forma descrita en el Capítulo 4, todas las transacciones concurrentes se suspenden. Notemos que cualquiera de los CC pueden activar la señal de parada y todos la observan. Memoria también observa la señal parada.

En el primer caso, si la cache debe de responder a la transacción en curso, se produce un riesgo estructural en el autómata del CC. El CC está gestionando un acceso a memoria, que está esperando obtener el bus, y es necesario que atienda la petición de observación. Para eliminar el riesgo estructural el autómata debe ser reentrante. Esto es, debe ser capaz de almacenar el estado de la petición pendiente, gestionar la petición de observación y posteriormente reanudar la gestión de la petición pendiente en el punto donde estaba.

Los agentes en el CC que gestionan la petición pendiente y la observación de la transacción son distintos. Por tanto, es suficiente que el agente procesador no utilice los recursos que necesita el agente observador, durante el lapso de tiempo que dura la transacción. En el peor caso son todos los campos de la cache. Notemos que la respuesta a una observación puede necesitar actualizar el campo estado y leer el campo de datos. En la Figura 5.7 se muestra de forma esquemática el camino de datos de un multiprocesador y las fases: a) arbitraje y b) observación. La transacción de bus corresponde al CC1 y existe una transacción pendiente del CC2 (Figura 5.7 a). Este último CC debe efectuar una observación (Figura 5.7 b).

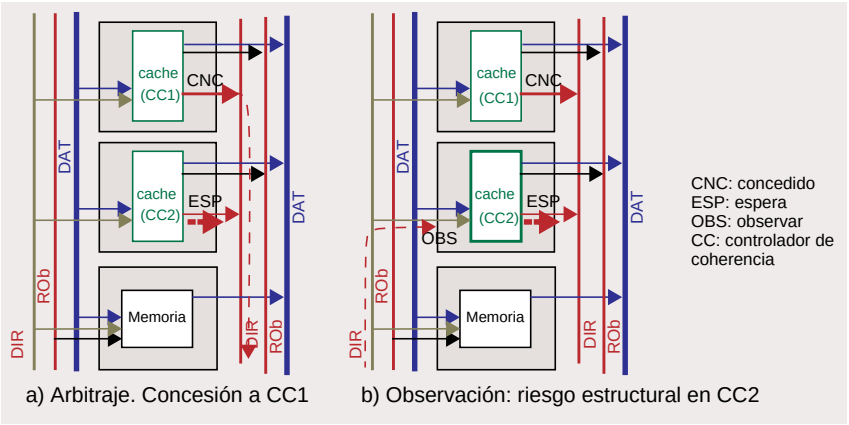


Figura 5.7 Riesgo estructural en el controlador de coherencia: a) arbitraje, b) riesgo estructural.

En el segundo caso, la existencia de la ventana temporal o de vulnerabilidad permite que, mientras se está esperando la concesión de acceso al bus, se observe una transacción al mismo bloque. Esta transacción, en el orden de bus, es previa a la que está esperando. Por tanto, hay que tenerla en cuenta, ya que puede determinar modificaciones del estado del bloque. Por otro lado, al efectuar la petición de bus se han previsto unas acciones cuando se conceda el bus, sin tener en cuenta la transacción que hay en el bus. En consecuencia, en la petición pendiente hay que tener en cuenta la transacción

en curso y es posible que haya que modificar las acciones previstas. Si este es el caso diremos que se produce un riesgo de datos.

En estas condiciones, la detección de la necesidad de acceder al bus (petición de bus) y la realización de la transacción (concesión de bus) no puede considerarse una acción atómica (Figura 5.1). Entonces, para gestionar esta característica se introducen los denominados estados transitorios. Estos estados se utilizan para separar la acción de solicitar acceso al bus de la acción de ocupar el bus, para iniciar y finalizar una transacción, la cual es atómica. Cada una de estas dos acciones es individualmente atómica, pero no lo son conjuntamente como en el Capítulo 4. El estado transitorio se utilizará para gestionar una espera entre acciones atómicas.

En la Figura 5.8 se muestra un diagrama de transiciones entre estados con dos estados estables y dos estados transitorios. La forma de etiquetar un estado transitorio es utilizar como prefijo el estado fuente y como sufijo el estado destino. Las acciones de petición de bus (PtBus) y concesión de bus (CnBus) se utilizan para identificar: a) el inicio de la ventana de vulnerabilidad y b) el final de la ventana de vulnerabilidad. Cuando se efectúa la petición de bus se realiza una transición desde el estado A al estado AB. En el estado AB puede que se observen transacciones que no requieran modificar las acciones previstas. Entonces, cuando se recibe la señal de concesión de bus se inicia la transacción prevista y se efectúa la transición al estado estable B. Si en el estado AB se observa una transacción, que requiere modificar las acciones previstas, se efectúa una transición a un estado transitorio que la identifique (ABB). En el estado transitorio ABB, cuando se recibe la señal de concesión de bus, se actúa en consecuencia.

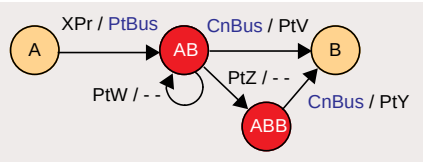


Figura 5.8 Estados transitorios para distinguir entre petición e inicio de la transacción de bus.

PROTOCOLO DE INVALIDACION CON ESCRITURA INMEDIATA

La descripción funcional del protocolo se ha efectuado en el Capítulo 4. En cuanto al camino de datos es el descrito en el mismo capítulo, teniendo en cuenta la replicación de los campos etiqueta y estado de la cache (Figura 5.5).

Eventos y transacciones

Además de los eventos descritos en el Capítulo 4 hay que tener en cuenta los eventos relacionados con el árbitro del bus: a) petición de bus y b) concesión de bus. En la Tabla 5.1 se describen estos eventos.

Bus		Comentario
Peticiones del CC	**Respuestas al CC**	
PtBus: petición de acceso al bus	CnBus: concesión de acceso al bus	Cada CC tiene este par de señales. Cuando un CC determina que necesita utilizar el bus (escritura o fallo de lectura) efectúa una petición al árbitro. Cuando el árbitro concede el acceso a un CC este utiliza el bus para iniciar la transacción.

Tabla 5.1 Peticiones al árbitro y respuestas.

Cada CC está conectado al árbitro mediante señales dedicadas de petición y concesión. El árbitro concede una de las peticiones activadas en cada fase de arbitraje. Una fase de arbitraje se realiza después de que finalice la transacción en curso[9] o cuando no hay transacción en curso.

Por razones de completitud de este capítulo, en la Tabla 5.2 y en la Tabla 5.3 se vuelven a describir los eventos descritos en el Capítulo 4. En las figuras incluidas en las tablas se muestra de forma explícita el árbitro de bus (Arb) y las señales de petición (PtBus) y concesión (CnBus) de bus.

En la Tabla 5.2 se describen las peticiones del procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). Cuando es acierto de lectura sólo se accede a la cache. En un fallo de lectura y en una escritura se accede a memoria.

Procesador	Controlador de coherencia (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato	dato	Si es un acierto en cache se lee el dato de cache. En caso contrario, antes de suministrar el dato, se inicia una transacción de lectura del bloque (Pt) y se asigna un contenedor para almacenar el bloque.
EPr (store): escritura de un dato	confirmación de la escritura (implícita al finalizar la transacción)	En cualquier caso se actualiza la memoria (PtE). Si es un acierto y el estado del bloque es válido se escribe el dato en cache al finalizar la transacción. No se asigna un contenedor en caso de fallo.

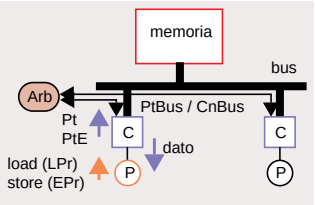


Tabla 5.2 Protocolo VI. Peticiones del procesador y respuestas.

9. Cuando hay una transacción en curso, la fase de arbitraje se suele solapar con la finalización de la transacción.

En la Tabla 5.3 se describen las peticiones del CC a memoria (Pt y PtE) y acciones del CC para gestionar la ubicación de bloques en la cache (conflictos, CcRe). Se distinguen las acciones de lectura de bloque (Pt) y escritura de dato (PtE). También se describe la acción de expulsión de un bloque debido a un conflicto.

Controlador de coherencia (CC)	Memoria / otros CC	Comentario
Peticiones a memoria y acciones	**Respuestas**	
Pt: petición de lectura de un bloque	bloque de datos	Se lee el bloque de datos de memoria y se suministra.
PtE: petición de escritura de un dato	confirmación de la consolidación de la escritura de forma implícita (un CC invalida el bloque si tiene copia)	Se actualiza la memoria con el dato
CcRe: expulsión de un bloque.		Se invalida la información del contenedor. No se notifica a memoria, ya que está actualizada

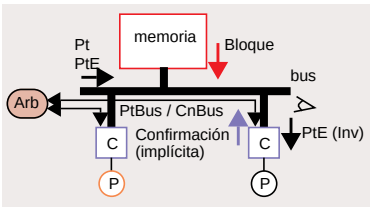


Tabla 5.3 Protocolo VI. Peticiones del CC a memoria, respuestas de memoria y de los otros CC y acciones del CC.

Estados y transiciones

En este protocolo hay dos estados estables: inválido (I) y válido (V). El número de estados transitorios es cuatro, que denominamos: IV, VV, II y VVI.

La descripción de las transiciones entre estados la efectuaremos en tres partes. En primer lugar tendremos en cuenta las transiciones debidas a las peticiones del procesador entre estados estables. En segundo lugar las transiciones inducidas por el observador entre estados estables[10]. Seguidamente analizaremos las transiciones inducidas por el observador en estados transitorios. Finalmente, como existe relación entre estas últimas y las primeras transiciones descritas se presentan el diagrama completo de estados y transiciones.

Eventos del procesador y reemplazo

El procesador se bloquea cuando hay que acceder a memoria y hay un acceso a memoria pendiente. Por otro lado, no hay buffer de escrituras[11]. Por tanto, un CC no gestiona peticiones del procesador cuando un bloque está en un estado transitorio.

10. Estos dos conjuntos de transiciones son los descritos en el Capítulo 4 utilizando los estados transitorios para identificar las acciones de petición y concesión de bus.
11. Posteriormente se relaja esta hipótesis.

En un estado estable (I, V), después de determinar que hay que acceder a memoria y a la vez que efectúa la petición de bus, el agente procesador cambia el estado estable por un estado transitorio. Al obtener la concesión de bus, el agente procesador emite la petición por el bus. En la fase de datos de la transacción, correspondiente a la respuesta, se efectúa la transición del estado transitorio al estado estable final (Figura 5.9). En una acción de reemplazo se invalida el bloque.

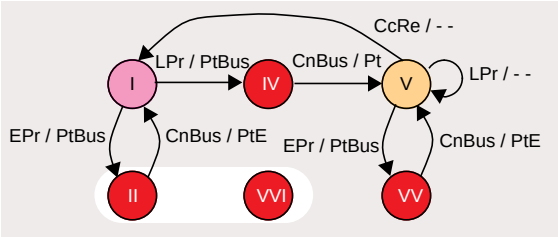


Figura 5.9 Protocolo VI. Transiciones desde estados estables iniciadas por el agente procesador y acción de reemplazo.

Eventos externos: acciones inducidas por observaciones

En la Figura 5.10 se muestra el diagrama de transiciones entre estados. Sólo hay una transición entre dos estados estables distintos. El agente observador al observar una transacción de escritura invalida el bloque, si está almacenado en cache.

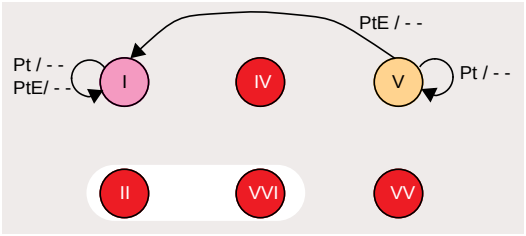


Figura 5.10 Protocolo VI. Transiciones desde estados estables iniciadas por el agente observador.

Eventos externos en estados transitorios

Mientras se está esperando obtener el bus (ventana de vulnerabilidad) se pueden observar transacciones al mismo bloque. En estos casos usualmente se dice que se produce una condición de carrera (“race condition”) que hay que resolver para que el estado final sea consistente[12].

En los estados transitorios IV e II estas observaciones no determinan ningún cambio de estado. Se llega a ellos después de detectar un fallo de cache. Por tanto, no se tiene copia del bloque en cache.

12. Dos transacciones referencian el mismo bloque y hay que serializarlas.

Sin embargo, en el estado VV se tiene copia del bloque en cache. Notemos que si no hubiera la petición pendiente, el bloque estaría en estado V y se invalidaría cuando el agente observador observara una transacción PtE en el bus (Figura 5.10). En otras palabras, la petición de escritura que ocupa el bus ha sido ordenada por el árbitro, en el orden global, antes que la petición pendiente.

En el estado VV, las acciones pendientes, después de obtener el bus son: a) actualizar memoria y b) actualizar la cache (Figura 5.11). Ahora bien, si se observa una escritura en la ventana de vulnerabilidad el bloque no contiene información válida. Por tanto, cuando se obtenga la concesión del bus sólo hay que actualizar memoria[13]. En consecuencia, al observar una transacción PtE en el estado transitorio VV, se cambia a otro estado transitorio denominado VVI, el cual se utiliza para recordar que previamente el árbitro ha ordenado una escritura emitida desde otro CC. Desde este estado transitorio, cuando se obtenga la concesión de bus, se actualizará memoria y se pasará al estado estable I.

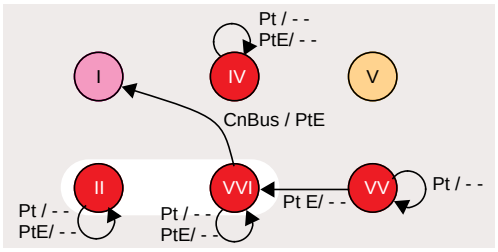


Figura 5.11 Protocolo VI. Transiciones desde estados transitorios iniciadas por el agente observador y la concesión del bus.

Diagrama completo de estados y transiciones

En la Figura 5.12 se muestran los dos estados estables y los cuatro estados transitorios con todas las posibles transiciones debidas a eventos del agente procesador, eventos del agente observador y a acciones de reemplazo.

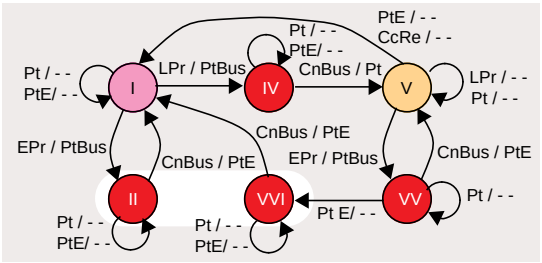


Figura 5.12 Bus atómico. Protocolo VI. Diagrama de estado y transiciones.

13. Recordemos que en una escritura, si no se tiene copia del bloque no se asigna contenedor de cache.

Notemos que todas las acciones y respuestas de salida en los estados II y VVI son las mismas (Pt / - -, PtE / - -, CnBus / PtE, Figura 5.12). Por tanto, los dos estados transitorios pueden agruparse en un único estado transitorio, lo cual no realizaremos por razones de claridad en la descripción.

Tabla de estados y transiciones

En la Tabla 5.4 se muestra la descripción de los estados, estables y transitorios, eventos y transiciones entre estados en formato tabla. Las casillas que no contienen información indican un error: en un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

			Eventos del procesador y reemplazo			Bus	Eventos externos (transacciones de bus)	
			LPr	EPr	CcRe	CnBus	Pt	PtE
Estados	Estables	I	PtBus; IV	PtBus; II			- - ; I	- - ; I
		V	- - ; V	PtBus; VV	- - ; I		- - ; V	- - ; I
	transitorios	IV				Pt; V	- - ; IV	- - ; IV
		VV				PtE; V	- - ; VV	- - ; VVI
		VVI				PtE; I	- - ; VVI	- - ; VVI
		II				PtE; I	- - ; II	- - ; II

Tabla 5.4 Bus atómico. Protocolo VI. Tabla de estados y transiciones.

Representación de transacciones y transiciones entre estados

En este apartado se muestran dos formas de representar las transacciones de bus y transiciones entre estados al ejecutar una secuencia de accesos a memoria: a) en formato tabla y b) mediante un diagrama temporal. Finalmente se muestra, mediante varios gráficos, una animación.

Representación en formato tabla

Utilizaremos una tabla, donde cada fila representa un acceso a memoria y no se gestiona el siguiente acceso hasta que ha finalizado el anterior. En una fila, de izquierda a derecha, después de la instrucción de acceso a memoria, se especifica:

1. El estado transitorio, si es el caso.

2. La transacción de bus, si es el caso.
3. El nombre de la variable y el valor en memoria.
4. Quién suministra el dato o bloque (cache o memoria).
5. Para las cache donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable del bloque.

Ejemplo. En la Tabla 5.5 se muestra una secuencia de accesos a memoria realizada por tres procesadores. La variable t no está almacenada en ninguna cache y el valor en memoria es dos.

Los dos primeros accesos son instrucciones load y producen fallos en cache. El estado transitorio del bloque en las caches, que gestionan los controladores de coherencia CC1 y CC3, es IV y el estado final es V. El siguiente acceso, efectuado por el procesador P3, es un acierto de escritura, se invalida la copia en la cache del procesador P2 y se actualiza la copia de cache (C3) y memoria. Notemos que la actualización de memoria queda reflejada en la fila. El estado transitorio del bloque en la cache C3 es VV y el estado final es V.

	C 1	C 2	C 3	Bus	mem.			C 1			C 2			C 3		
acceso	est.	est.	est.	trans.	var.	val.	sum.	var.	val.	est.	var.	val.	est.	var.	val.	est.
1. P1 load t	IV			Pt	t	2	mem.	t	2	V						
2. P3 load t			IV	Pt	t	2	mem							t	2	V
3. P3 store t (21)			VV	PtE	t	21	C3	t	2	I				t	21	V
4. P1 load t	IV			Pt	t	21	mem.	t	21	V						
5. P2 store t (8)		II		PtE	t	8	C2	t	21	I				t	21	I

Tabla 5.5 Bus atómico. Protocolo VI. Formato tabla: transacciones y cambios de estado del bloque en las caches en una secuencia de accesos a memoria.

Seguidamente en el procesador P1 se produce un fallo de cache debido a una instrucción load. Finalmente el procesador P2, que no tiene almacenado el bloque en cache, ejecuta una instrucción store. Ello da lugar a que se invaliden las copias en las caches de los otros procesadores y se actualice memoria. El estado transitorio del bloque en la cache C2 es II y el estado final es I. Recordemos que no se asigna contenedor.

Diagrama temporal

EL diagrama temporal y la representación de una transacción son como en el Capítulo 4.

En el segundo grupo de filas del final de la tabla también se representa el estado transitorio al solicitar acceso al bus. En la fase arb de una transacción se especifica el estado transitorio del bloque. Al finalizar la transacción se especifica el estado estable del bloque.

Ejemplo. Para la secuencia de accesos a memoria de la Tabla 5.5, en la Figura 5.13 se muestra un diagrama donde se observa la secuencia de transacciones de bus con las fases.

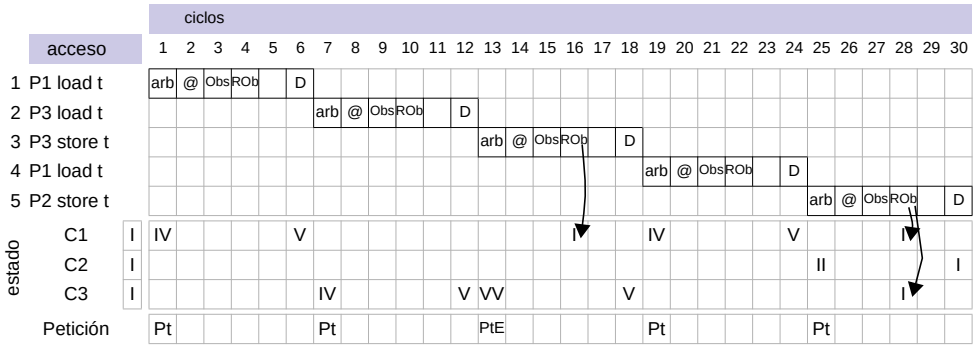


Figura 5.13 Bus atómico. Protocolo VI. Diagrama temporal y cambios de estado de una secuencia de accesos.

Animación

En la Figura 5.14 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 5.5. Además de las peticiones y respuestas, se muestran los cambios de estado utilizando diagramas de transición entre estados. Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido. Para reducir la información representada en la figura, no se muestran los casos en que un agente observador no modifica el estado de un bloque al observar una transacción.

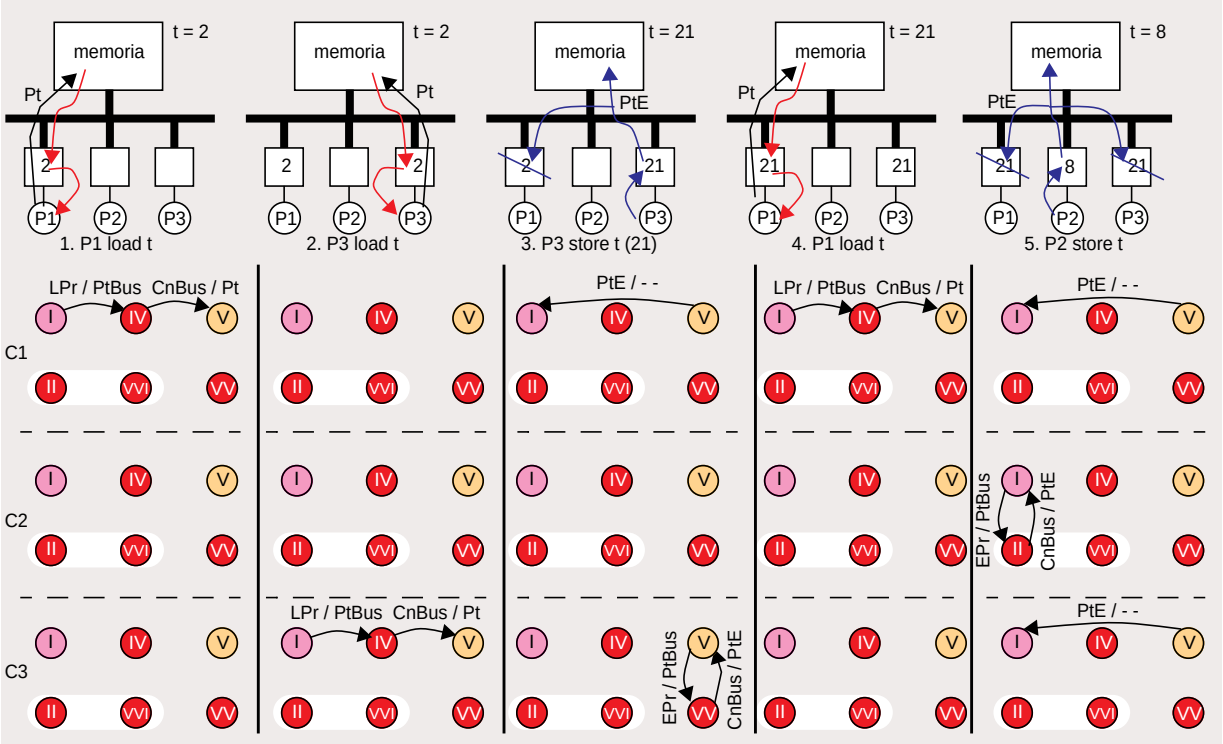


Figura 5.14 Bus atómico. Protocolo VI. Secuencia de accesos a memoria y fotogramas de las peticiones y respuestas y cambios de estado. En una observación, cuando el estado es inválido no se muestra la transacción.

Representación en formato tabla de peticiones concurrentes

En una secuencia se agrupan los accesos a memoria y la agrupación se muestra mediante líneas horizontales continuas. Usualmente se agrupan de dos en dos.

Cada grupo de peticiones se gestiona independientemente. No se empieza a gestionar el siguiente grupo de peticiones hasta que ha finalizado el anterior. Cuando los accesos a memoria de un grupo necesitan acceder al bus se supone que las acciones de petición se realizan concurrentemente en el mismo ciclo.

En una fila de la tabla, de izquierda a derecha, después de la instrucción de acceso a memoria se especifica:

1. El orden de concesión del bus a la petición (1ª, 2ª, . . .). Supondremos usualmente que en un grupo de peticiones, el bus se concede en el orden en el cual se especifican los accesos a memoria en la secuencia de accesos.

2. El estado transitorio al efectuar la petición de bus.

3. La petición que ocupa el bus en la columna correspondiente al orden de concesión.

4. El nombre de la variable y el valor en memoria.

5. Quién suministra el dato o bloque (cache o memoria).

6. Para las caches donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable o transitorio del bloque.

Ejemplo. Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenarse en cache se ubican en contenedores distintos. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 7 y t = 2.

Los dos primeros accesos concurrentes son fallos de lectura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es IV y el estado estable final es V.

	arb.	C 1	C 2	C 3	Bus (trans.)		mem.			C 1			C 2		
acceso	ord.	est.	est.	est.	1º	2º	var.	val.	sum.	var.	val.	est.	var.	val.	est.
1. P1 load t	1º	IV			Pt		t	2	mem	t	2	V			
1. P2 load u	2º		IV			Pt	u	7	mem				u	7	V
2. P1 store u (9)	1º	II			PtE		u	9	C1				u	7	I
2. P2 store t (1)	2º		II			PtE	t	1	C2	t	2	I			
3. P2 load u	1º	IV			Pt		u	9	mem				u	9	V
3. P1 load u	2º		IV			Pt	u	9	mem	u	9	V			
4. P1 store u (12)	1º	VV			Pt		u	12	C1	u	12	V	u	9	VVI
4. P2 store u (3)	2º		VV			Pt	u	3	C2	u	12	I	u	9	I

Tabla 5.6 Bus atómico. Protocolo VI. Formato tabla. transacciones y cambios de estado en una secuencia de accesos a memoria concurrentes.

El segundo grupo de accesos concurrentes son fallos de escritura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es II. Al ser fallo en escritura no se asigna contenedor en cache. Cada acceso actualiza una variable distinta. El acceso del procesador P1 invalida la copia del bloque en la cache del procesador P2 y el

acceso del procesador P2 invalida la copia del bloque en la cache del procesador P1. El tercer grupo de accesos concurrentes son fallos de lectura. Este caso es semejante al primer grupo de accesos concurrentes.

El cuarto grupo de accesos concurrentes son aciertos de escritura cuando se efectúa la petición de bus. Por tanto, el estado transitorio en los dos casos es VV. Notemos que los dos accesos son a la misma variable. La concesión de bus la obtiene en primer lugar el CC1. El estado estable del bloque en la cache C1 al finalizar la transacción es V y el estado transitorio del bloque en la cache C2 es VVI. Notemos que la transacción en curso y la petición pendiente de CC2 referencian la misma variable. Por tanto, el CC2 debe tener en cuenta la ordenación determinada por el bus. El estado transitorio en la cache C2 se muestra en la columna correspondiente a la cache C2.

Al finalizar la transacción iniciada por el CC2 el estado estable del bloque en las caches C1 y C2 es I. La transacción del CC2 invalida la copia en la cache de C1 y previamente la transacción del CC1 había indicado que debía invalidarse la copia en la cache C2 (estado transitorio VVI).

Diagrama temporal de peticiones concurrentes

En un grupo de peticiones suponemos que se solapa la fase de arbitraje de la próxima transacción con la finalización de la transacción previa.

En las Figura 5.15 se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado en una transacción.

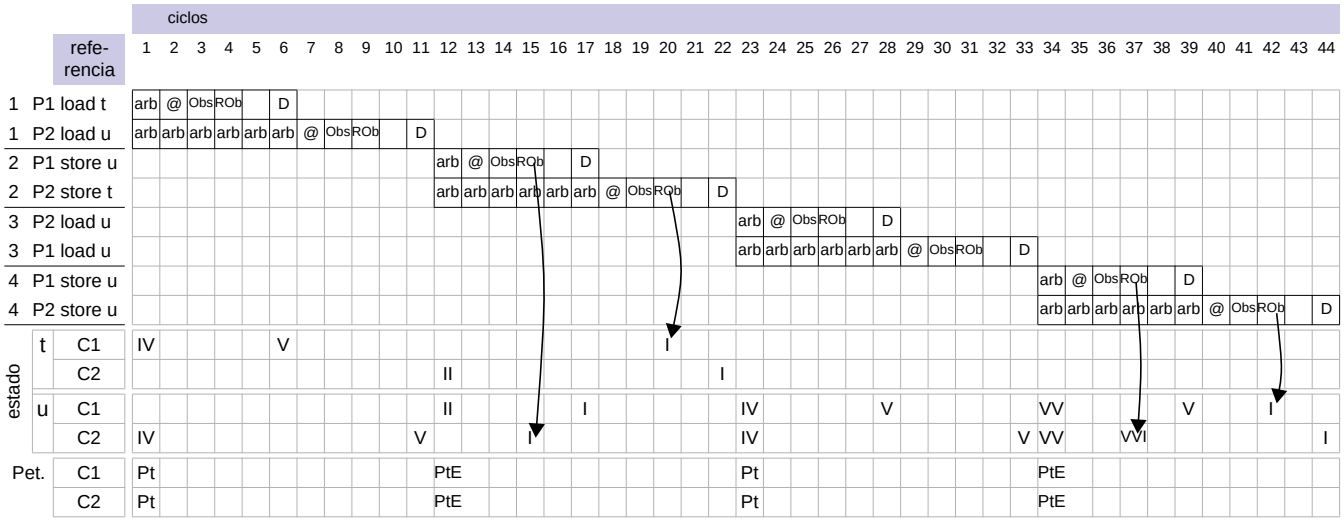


Figura 5.15 Bus atómico. Protocolo VI. Diagrama temporal de una secuencia de accesos concurrentes.

Verificación no formal de coherencia y consistencia

La base es la descripción efectuada en el Capítulo 4. La diferencia se centra en los estados transitorios de un bloque en cache. Durante la ventana temporal se memoriza localmente y por bloque en el CC, la observación de una escritura que haya ordenado previamente el árbitro del bus.

Coherencia de cache

Orden de programa en accesos a la misma posición de memoria. El procesador efectúa los accesos en el orden determinado por el L.M. El procesador se bloquea en un acceso a memoria, hasta que finaliza la transacción pendiente[14].

Propagación de escrituras. Al utilizar escritura inmediata todas las escrituras se transmiten por el bus. Una escritura se propaga mediante una transacción y es observada por todos los CC. Las invalidaciones se efectúan durante la transacción y la respuesta de cada invalidación está implícita en la transacción de bus.

Serialización de escrituras (atomicidad de una escritura).

- Punto de serialización: El bus es el punto de serialización de las escrituras a la misma posición de memoria[15]. En el bus sólo hay una transacción en curso y es observada por todos los CC. El arbitraje del bus determina el orden de serialización de las transacciones de escritura a una posición de memoria. Si una cache tiene el bloque en un estado estable la acción de invalidación es inmediata. En caso contrario se memoriza mediante un estado transitorio del bloque. Este estado transitorio indica que previamente ha sido serializada por el árbitro una escritura iniciada por otro CC. Entonces, las escrituras a una posición de memoria son serializadas por el árbitro en coordinación, o con ayuda, de los CC de forma local durante la ventana de vulnerabilidad de una petición de acceso al bus.
- Consolidación de una escritura: Una escritura está consolidada al finalizar la transacción[16]. Las acciones que se toman localmente en un CC al obtener el bus tienen en cuenta si previamente ha sido serializada la escritura de otro CC al mismo bloque. Por otro lado, una escritura es atómica, ya que, una vez finalizada la transacción correspondiente, una instrucción load posterior lee el valor establecido por la escritura previa en el orden de bus. Las copias del bloque son invalidadas en cada

14. Además, hay que garantizar las dependencias de datos en el hilo que se ejecuta.
15. Aún más, es el punto de serialización de las escrituras y los fallos de lectura a cualquier posición de memoria.
16. Esta decisión es conservadora. La escritura puede considerarse consolidada cuando ocupa el bus.

transacción de escritura. Un fallo de lectura lee el valor establecido por la escritura previa en el orden de bus. Un acierto de lectura lee el valor obtenido en el fallo de lectura previo por la escritura previa, que acierta en cache, del mismo procesador.

Consistencia secuencial de memoria

Se utilizan los mismo razonamientos que en el Capítulo 4, ya que una escritura puede considerarse consolidada al finalizar la transacción.

PROTOCOLO DE INVALIDACION CON ESCRITURA RETARDADA

La descripción funcional del protocolo se ha efectuado en el Capítulo 4. El camino de datos es el descrito en el mismo capítulo teniendo en cuenta la replicación de los campos etiqueta y estado de la cache (Figura 5.5).

Eventos y transacciones

Además de los eventos descritos en el Capítulo 4 hay que tener en cuenta los eventos relacionados con el árbitro del bus: a) petición de bus y b) concesión de bus. En la Tabla 5.1 se describen estos eventos.

Por razones de completitud de este capítulo se vuelven a describir los eventos descritos en el Capítulo 4 en la Tabla 5.7 y en la Tabla 5.8. En las figuras incluidas en las tablas se muestra de forma explícita el árbitro de bus (Arb) y las señales de petición (PtBus) y concesión (CnBus) de bus.

En la Tabla 5.7 se describen las peticiones del procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). En una lectura, si es acierto en cache sólo se accede a cache. En una escritura si es acierto y el estado es M se actualiza el bloque almacenado en cache. En los otros casos se solicita el bloque a memoria y si se pretende actualizar el bloque, la solicitud es con intención de modificación.

Procesador	Controlador de cache (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato	dato.	Si es un acierto en cache se lee el dato de cache. En caso contrario, antes de suministrar el dato, se inicia una transacción de lectura de bloque (Pt) y se asigna un contenedor para almacenar el bloque.
EPr (store): escritura de un dato	confirmación de la escritura.	Si es un acierto y el estado del bloque es M se escribe el dato en cache. En caso contrario, en primer lugar, se solicita el bloque con intención de modificación (PtIm).

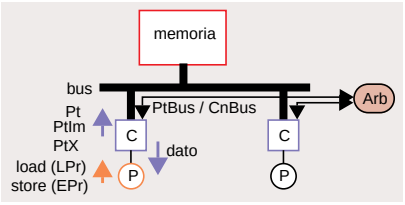


Tabla 5.7 Protocolo MLI. Peticiones del procesador y respuestas.

En la Tabla 5.8 se describen las peticiones y respuestas del CC y acciones del mismo para gestionar la ubicación de bloques en cache.

Controlador de coherencia	Memoria / otros CC	Comentario
Peticiones a memoria y acciones del CC y memoria	**Respuestas**	
Pt: petición de lectura de un bloque	bloque de datos.	Se obtiene el bloque de datos.
PtIm: petición de lectura de un bloque con intención de modificación	bloque de datos y confirmación de las invalidaciones (implícito al finalizar la transacción).	Se obtiene el bloque de datos en exclusividad.
	CaC: suministro del bloque.	Bloque en estado M. La cache suministra el bloque solicitado en una transacción.
	MOD: bloque en estado M.	Bloque en estado M. Inhibe el suministro desde memoria.
CcRe: expulsión de un bloque		Se invalida la información del contenedor.
PtX: petición de actualización de memoria		Actualización de memoria con un bloque en estado M.
Dev: actualización de memoria		Actualización de memoria en CaC, si es el caso.

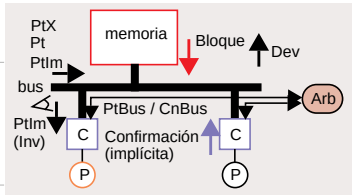


Tabla 5.8 Protocolo MLI. Peticiones del CC a memoria, respuestas de memoria y de los otros CC y acciones del CC.

En la Tabla 5.8 se distinguen las peticiones: a) lectura de bloque y b) lectura de bloque con intención de modificación. Las respuestas son: a) suministro del bloque y b) señal para inhibir el suministro de memoria. Finalmente se describe la petición asociada con la expulsión de un bloque modificado y la acción inducida de actualización de memoria, si es el caso.

Estados y transiciones

La memoria puede no estar actualizada y los posibles estados estables de un bloque en un contenedor de cache son: a) Inválido (I), b) Lectura (L) y c) Modificado (M). El número de estados transitorios es cuatro, que denominamos: IL, LM, IM y MI.

La descripción de las transiciones entre estados la efectuaremos en tres partes. En primer lugar tendremos en cuenta las transiciones entre estados estables debidas a las peticiones del procesador. En segundo lugar las transiciones entre estados estables inducidas por el observador[17]. Seguidamente analizaremos las transiciones inducidas por el observador en estados transitorios. Finalmente, como existe relación entre estas últimas y las primeras transiciones descritas se presenta el diagrama completo de estados y transiciones.

Eventos del procesador y reemplazo

El procesador se bloquea cuando hay que acceder a memoria y hay un acceso a memoria pendiente. Por otro lado, no hay buffer de escrituras. Por tanto, un CC no gestiona peticiones del procesador cuando un bloque está en un estado transitorio.

En un estado estable, después de determinar que hay que acceder a memoria (I, L) y a la vez que efectúa la petición de bus (PtBus), el agente procesador cambia el estado estable por un estado transitorio (IM, IL, LM, Figura 5.16).

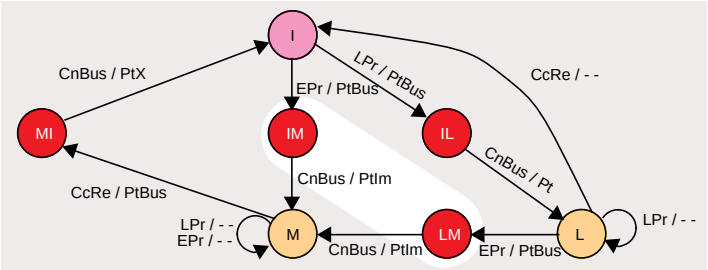


Figura 5.16 Protocolo MLI. Transiciones desde estados estables iniciadas por el agente procesador y acción de reemplazo.

17. Estos dos conjuntos de transiciones son los descritos en el Capítulo 4 utilizando los estados transitorios para identificar las acciones de petición y concesión de bus.

Al obtener la concesión de bus (CnBus), y por tanto emitir la petición por el bus, se efectúa la transición del estado transitorio al estado estable final. En una acción de reemplazo se invalida el bloque. Si el bloque está en estado M previamente hay que actualizar memoria. Para ello, se solicita acceso al bus (estado MI) y una vez concedido, se solicita a memoria que proceda a la actualización (PtX).

Eventos externos: acciones inducidas por observaciones

En la Figura 5.17 se muestra el diagrama de transiciones entre estados. Hay tres transiciones entre estados estables distintos.

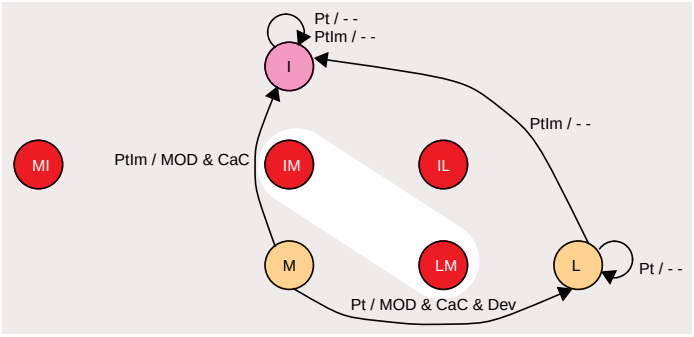


Figura 5.17 Protocolo MLI. Transiciones desde estados estables iniciadas por el agente observador.

Las transiciones desde el estado M requieren suministrar el bloque (CaC), además de indicar que la memoria debe inhibirse del suministro (señal MOD activada). En la transición al estado L se realizan las mismas acciones y además, se actualiza memoria (Dev)[18].

Eventos externos en estados transitorios

Mientras se está esperando obtener el bus (ventana de vulnerabilidad) se pueden observar transacciones al mismo bloque (Figura 5.18).

En los estados transitorios IL, IM y LM estas observaciones no determinan ningún cambio de estado. A los dos primeros estados se llega después de detectar un fallo de cache. Por tanto, no se tiene copia del bloque en cache. Al estado transitorio LM se llega debido a que se necesita la exclusividad para actualizar una palabra del bloque. Aunque el estado inicial era L y se tiene copia del bloque, se solicita el bloque con intención de modificarlo. Por tanto, la petición de exclusividad se gestiona igual que un fallo de cache.

18. En una transición al estado I también se puede actualizar memoria, aunque la actualización es superfula. La transición ente estados está inducida por una petición de lectura con intención de modificación.

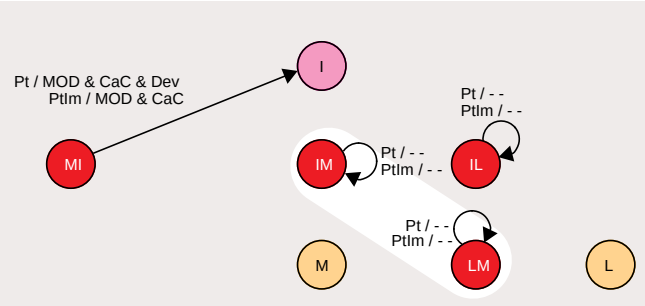


Figura 5.18 Protocolo MLI. Transiciones desde estados transitorios iniciadas por el agente observador.

En el estado MI se tiene copia del bloque en cache y la memoria no está actualizada. De hecho, se está esperando la concesión del bus para actualizar memoria, ya que el CC expulsa el bloque. La cache que tiene el bloque en el estado MI es la encargada de suministrar el bloque, cuando observa una petición que accede al bloque. Entonces, al observar una transacción, donde se referencia el bloque que se quiere expulsar, se suministra (CaC) y se indica a memoria que inhiba el suministro (MOD). Cuando la petición es Pt adicionalmente se actualiza memoria, aprovechando el suministro de la cache (operación Dev).

Conceptualmente la acción es la invalidación del bloque en cache. Entonces, el CC invalida el bloque y desactiva la solicitud de petición de acceso al bus[19].

Diagrama completo de estados y transiciones

En la Figura 5.19 se muestran los tres estados estables y los cuatro estados transitorios con todas las transiciones debidas a eventos del agente procesador, eventos del agente observador y a acciones de reemplazo.

Notemos que todas las acciones y respuestas de salida en los estados IM y LM son las mismas (Pt / - -, PtIm / - -, CnBus / PtIm, Figura 5.19). Por tanto, los dos estados transitorios pueden agruparse en un único estado transitorio, lo cual no realizaremos por razones de claridad en la descripción.

19. Esta alternativa no es válida en algunas redes ordenadas donde, por ejemplo, se encolan las peticiones de arbitraje.

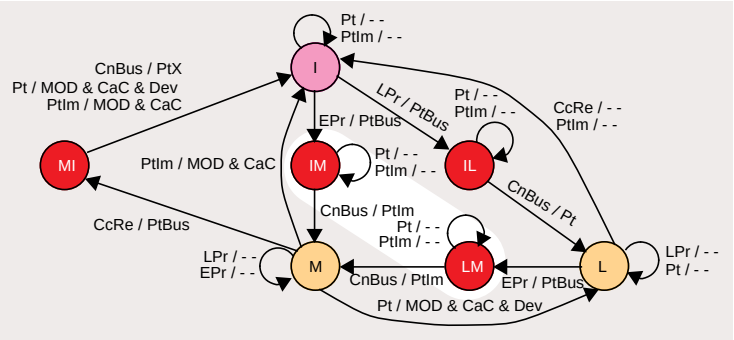


Figura 5.19 Bus atómico. Protocolo MLI. Diagrama de estado y transiciones.

Tabla de estados y transiciones

En la Tabla 5.9 se muestra la descripción de los estados estables y transitorios, eventos y transiciones entre estados en formato tabla. Las casillas que no contienen información indican un error: en un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

			Eventos del procesador y reemplazo			Arbitro	Eventos externos (transacciones de bus)	
			LPr	EPr	CcRe	CnBus	Pt	PtIm
Estado	Estables	I	PtBus ; IL	PtBus; IM			- - ; I	- - ; I
		L	- - ; L	- - , PtBus; LM	- -; I		- - ; L	- - ; I
		M	- - ; M	- - ; M	PtBus; MI		MOD&CaC&Dev; L	MOD&CaC; I
	Ttransitorios	IL				Pt; L	- - ; IL	- -; IL
		LM				PtIm; M	- - ; LM	- -; LM
		IM				PtIm; M	- - ; IM	- - ; IM
		MI				PtX; I	MOD&CaC&Dev; I	MOD&CaC; I

Tabla 5.9 Bus atómico y protocolo MLI: Tabla de estados y transiciones entre estados.

Representación de transacciones y transiciones entre estados

En este apartado se muestran dos formas de representar las transacciones de bus y transiciones entre estados al ejecutar una secuencia de accesos a memoria: a) en formato tabla y b) mediante un diagrama temporal. Finalmente se muestra, mediante varios gráficos, una animación.

Representación en formato tabla

Utilizaremos una tabla, donde cada fila representa un acceso a memoria y no se gestiona el siguiente acceso hasta que ha finalizado el anterior. En una fila, de izquierda a derecha, después de la instrucción de acceso a memoria, se especifica:

1. El estado transitorio, si es el caso.
2. La transacción de bus y la señal MOD, si es el caso.
3. El nombre de la variable y el valor en memoria.
4. Quién suministra el bloque (cache o memoria).
5. Para las cache donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable del bloque.

Cuando es necesario mostrar un reemplazo se utilizan dos filas consecutivas. En la primera fila se indica el bloque que se expulsa y en la segunda fila el acceso a memoria que determina la expulsión del bloque.

Ejemplo. En la Tabla 5.10 se muestra una secuencia de accesos a memoria realizada por tres procesadores. Para rellenar la tabla suponemos que la variable t no está almacenada en ninguna cache y el valor en memoria es dos.

	C 1	C 2	C 3	Bus		mem.			C 1			C 2			C 3		
acceso	est.	est.	est.	trans.	señal	var.	val.	sum.	var.	val.	est.	var.	val.	est.	var.	val.	est.
1. P1 load t	IL			Pt		t	2	mem.	t	2	L						
2. P3 load t			IL	Pt		t	2	mem							t	2	L
3. P3 store t (21)			LM	PtIm		t	2	mem	t	2	I				t	21	M
4. P1 load t	IL			Pt	MOD	t	21	C3	t	21	L				t	21	L
5. P2 store t (8)		IM		PtIm		t	21	mem	t	21	I	t	8	M	t	21	I

Tabla 5.10 Bus atómico. Protocolo MLI. Secuencia de accesos a memoria: transacciones y cambios de estado en las caches.

Los dos primeros accesos son instrucciones load y producen fallos en cache. El estado transitorio del bloque en las caches C1 y C3 es IL y el estado final es L. El siguiente acceso es un acierto, pero es una escritura y hay que obtener el acceso exclusivo al bloque. Se solicita el bloque con intención de modificación y la copia en la cache C1 será invalidada. El estado transitorio del bloque en la cache del procesador P3 es LM y el estado final es M.

Seguidamente, en el procesador P1 se produce un fallo de cache debido a una instrucción load. El bloque lo tiene en exclusividad la cache del procesador P3. Por tanto, la cache C3 es la encargada de suministrar el bloque en la transacción de bus iniciada por el CC1. Por tanto, el CC3 activa la señal MOD. La memoria se actualiza utilizando el bloque suministrado en la transacción. El estado transitorio del bloque en la cache del procesador P1 es IL y el estado final L. El estado final del bloque en la cache del procesador P3 al finalizar la transacción es L.

Finalmente el procesador P2, que no tiene almacenado el bloque en cache, ejecuta una instrucción store. Ello da lugar a que se invaliden las copias en las caches de los otros procesadores. En este caso el bloque lo suministra memoria. El estado transitorio del bloque en la cache del procesador P2 es IM y el final estado final es M.

Diagrama temporal

El diagrama temporal y la representación de una transacción son como en el Capítulo 4.

En el segundo grupo de filas del final de la tabla también se representa el estado transitorio al solicitar acceso al bus. En la fase arb de una transacción se especifica el estado transitorio del bloque. Al finalizar la transacción se especifica el estado estable del bloque.

Cuando en una transacción se produce una transferencia entre caches (CaC), se indica en la fase posterior a ROb mediante el acrónimo CX, donde X es el indentificador numérico del CC que suministra el bloque.

El cambio de estado de un bloque que se expulsa y que no determina una transacción se indica en la fase arb de la petición que produce el conflicto.

Una expulsión, que requiere una transacción, se especifica en la columna etiquetada como accesos. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtX) y en la segunda fila la petición que determina la expulsión.

Ejemplo. Para la secuencia de accesos a memoria de la Tabla 5.10, en la Figura 5.20 se muestra un diagrama donde se observa la secuencia de transacciones de bus con las fases.

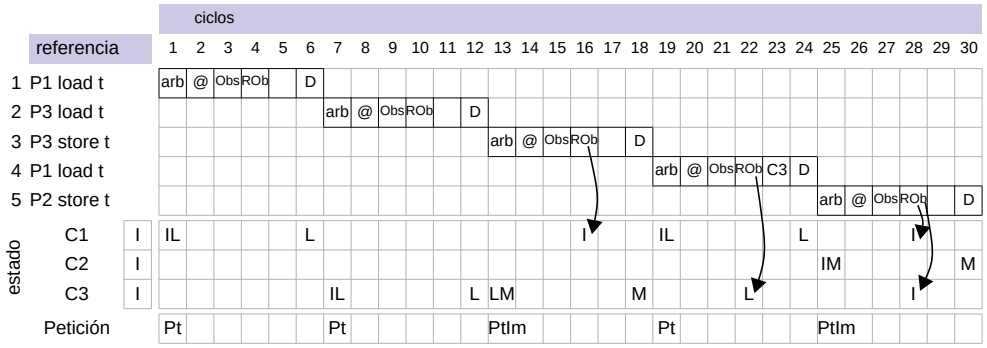


Figura 5.20 Bus atómico. Protocolo MLI. Secuencia de accesos a memoria y diagrama temporal y cambios de estado.

Animación

En la Figura 5.21 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 5.10. Además de las peticiones y respuestas se muestran los cambios de estado utilizando diagramas de transición entre estados. Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido. Para reducir la información representada en la figura, no se muestran los casos en que un agente observador no modifica el estado al observar una transacción. Tampoco se muestra el estado transitorio MI, ya que no se utiliza en esta secuencia de accesos a memoria.

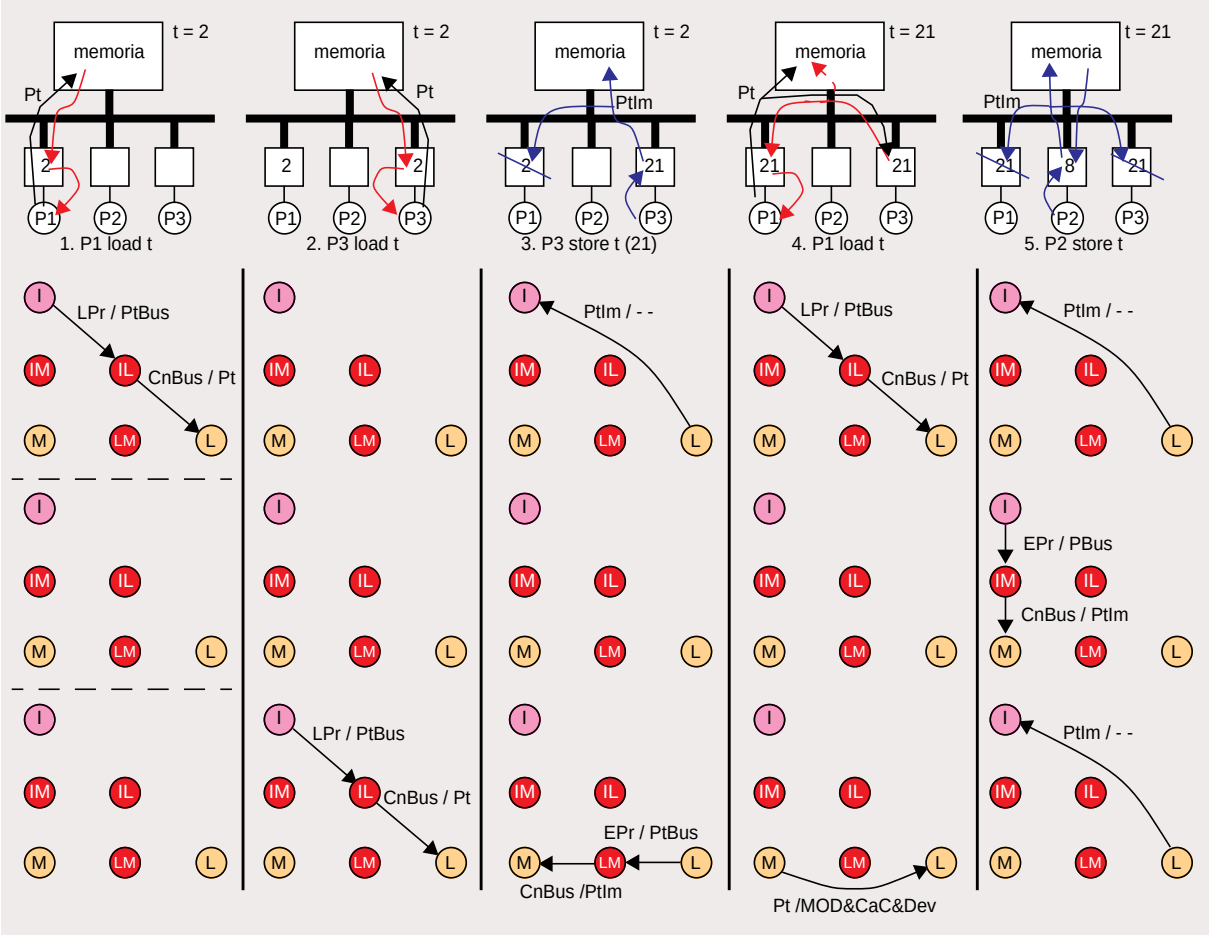


Figura 5.21 Bus atómico. Protocolo MLI. Secuencia de accesos a memoria. Fotogramas de las peticiones y respuestas y cambios de estado.

Representación en formato tabla de peticiones concurrentes

En una secuencia se agrupan los accesos a memoria y la agrupación se muestra mediante líneas horizontales continuas. Usualmente se agrupan de dos en dos.

Cada grupo de peticiones se gestiona independientemente. No se empieza a gestionar el siguiente grupo de peticiones hasta que ha finalizado el anterior. Cuando los accesos a memoria de un grupo necesitan acceder al bus se supone que las acciones de petición se realizan concurrentemente en el mismo ciclo.

En una fila de la tabla, de izquierda a derecha, después de la instrucción de acceso a memoria se especifica:

1. El orden de concesión del bus a la petición (1ª, 2ª, . . .). Supondremos usualmente que en un grupo de peticiones, el bus se concede en el orden en el cual se especifican los accesos a memoria en la secuencia de accesos.
2. El estado transitorio al efectuar la petición de bus.
3. La petición que ocupa el bus en la columna correspondiente al orden de concesión.
4. El nombre de la variable y el valor en memoria.
5. Quién suministra el dato o bloque (cache o memoria).
6. Para las caches donde se modifica la información almacenada, el nombre de la variable, el valor y el estado estable o transitorio del bloque.

Cuando es necesario mostrar un reemplazo se utilizan dos filas consecutivas. En la primera fila se indica el bloque que se expulsa y en la segunda fila el acceso a memoria que determina la expulsión del bloque.

Una expulsión que no se efectúa, ya que se ha suministrado el bloque, se representa indicando el estado I en la columna correspondiente al estado transitorio. Esta representación es similar a la expulsión de un bloque en el estado L.

Ejemplo. Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 7 y t = 2.

Los dos primeros accesos concurrentes son fallos de lectura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es IL y el estado estable final es L.

El segundo grupo de accesos concurrentes son fallos de escritura y existe un conflictos en las dos caches. Como los bloques que se expulsan están en estado L no es necesario efectuar ninguna transacción, memoria tiene una copia válida de los bloques. La acción de expulsión la realizan los dos CC antes de efectuar la petición de bus. En la tabla se observa una fila dedicada a este hecho en los dos accesos a memoria (primera fila en cada acceso a memoria, CcRe). El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio del bloque en los dos accesos a memoria (P1 y P2) es IM. Cada acceso actualiza una variable distinta y la petición es con intención de modificación. Ninguna de las dos transacciones invalida ningún bloque.

	arb.	C 1	C 2	C 3	Bus (trans.)			mem.			C 1			C 2		
acceso	ord.	est.	est.	est.	1º	2º	señal	var.	val.	sum.	var.	val.	est.	var.	val.	est.
1. P1 load t	1º	IL			Pt			t	2	mem.	t	2	L			
1. P2 load u	2º		IL			Pt		u	7	mem				u	7	L
2. P1 CcRe t		I									t	2	I			
store u (9)	1º	IM			PtIm			u	7	mem	u	9	M			
2. P2 CcRe u		I												u	7	I
store t (1)	2º		IM			PtIm		t	2	mem				t	1	M
3. P2 PtX t	1º	MI				PtX		t	1	C2				t	1	I
load u	2º	IL			Pt		MOD	u	9	C1	u	9	L	u	9	L
3. P1 load u																
4. P1 store u (12)	1º	LM			PtIm			u	9	mem	u	12	M	u	9	LM
4. P2 store u (3)	2º		LM			PtIm	MOD	u	12	C1	u	12	I	u	3	M

Tabla 5.11 Bus atómico. Protocolo MLI. Formato tabla: transacciones y cambios de estado del bloque en las caches en una secuencia de accesos a memoria concurrentes.

En el tercer grupo de accesos concurrentes hay un fallo de lectura y un acierto de lectura. El CC2 debe expulsar un bloque actualizado, debido a un conflicto en cache. Suponemos que las dos peticiones de bus (PtX y Pt) obtienen de forma consecutiva el bus. La expulsión de bloque y la petición de bloque por parte del CC2 se muestran en dos filas consecutivas. En la transacción iniciada por el CC2, donde se solicita un bloque, la memoria no está actualizada y el bloque lo suministra la cache C1. Al efectuar el suministro, el estado del bloque en la cache C1 pasa del estado M al estado L. Respecto al acierto de lectura no se indica ninguna información en la fila ya que no se efectúa ninguna transacción.

El cuarto grupo de accesos concurrentes son aciertos de escritura, siendo el estado del bloque L en las dos caches. La petición de los dos controladores de coherencia (CC1 y CC2) es lectura de bloque con intención de modificación y el estado transitorio es LM. La concesión de bus la obtiene en primer lugar CC1. El estado estable en la cache C1 al finalizar la transacción es M y el estado transitorio en la cache C2 es LM. Notemos que la transacción y la petición pendiente referencian la misma variable. Por tanto, el CC2 debe tener en cuenta la ordenación determinada por el bus. Como la petición pendiente obtendrá como respuesta el bloque, no se modifica el estado transitorio. En la transacción iniciada por CC2 la cache C1 suministra el bloque y activa la señal

correspondiente para que la memoria se inhiba (MOD). La transacción del CC2 invalida la copia en la cache de C1. Al finalizar la transacción iniciada por el CC2 el estado estable del bloque en la cache C1 es I y en la cache C2 es M.

Diagrama temporal de peticiones concurrentes

En un grupo de peticiones suponemos que se solapa la fase de arbitraje de la próxima transacción con la finalización de la transacción previa.

En un diagrama temporal, en la fase arb se especifica el estado transitorio y en la fase D se especifica el estado final en una transacción que ocupa el bus.

El cambio de estado de un bloque que se expulsa y que no determina una transacción se indica en la fase arb de la petición que produce el conflicto.

Una expulsión que requiere una transacción se especifica en la columna etiquetada como accesos. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtX) y en la segunda fila la petición que determina la expulsión.

En el diagrama temporal, la petición de expulsión es la primera que solicita el bus. Una vez ha obtenido el bus se indica la solicitud de la petición que ha determinado la expulsión. Una expulsión que no se efectúa, ya que se ha suministrado el bloque, deja de solicitar el arbitraje. La desactivación de la señal se efectúa en el ciclo siguiente a la fase ROb de la transacción que ha requerido el suministro. En este mismo ciclo se empieza a indicar el arbitraje de la petición que ha determinado la expulsión.

En un diagrama temporal un acierto en cache se indica con la letra A.

Ejemplo. En las Figura 5.15 se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado en una transacción. Las expulsiones que determinan el segundo grupo de acceso a memoria no requieren efectuar una transacción. Por tanto, no se utilizan filas para mostrarlo. El cambio de estado del bloque expulsado se indica en la fase arb de la petición que determina la expulsión. En cambio, en el tercer grupo de accesos a memoria se efectúa una expulsión de un bloque que ha sido actualizado en cache. La petición que determina el reemplazo empieza a solicitar el arbitraje después de que la petición de expulsión haya obtenido el bus. El acceso del procesador P1 es un acierto en cache.

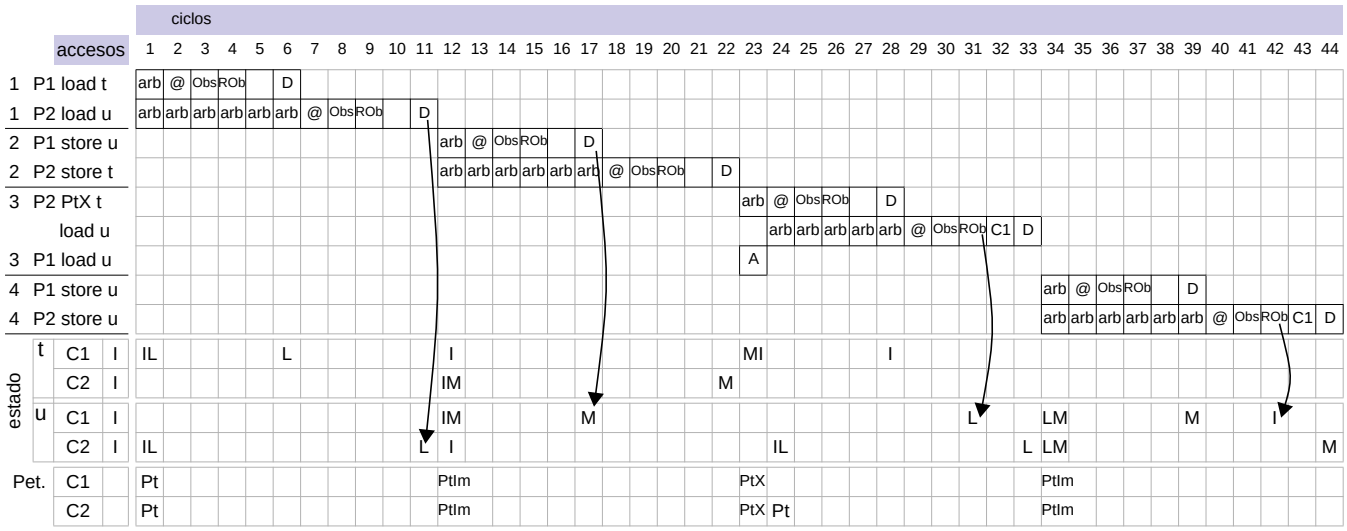


Figura 5.22 Bus atómico. Protocolo MLI. Diagrama temporal de una secuencia de accesos concurrentes.

Verificación no formal de coherencia y consistencia

La base es la descripción efectuada en el Capítulo 4. La diferencia se centra en los estados transitorios de un bloque en cache. Durante la ventana temporal en el CC se memoriza, localmente y por bloque, la observación de una escritura que haya ordenado previamente el árbitro del bus. En particular, en el protocolo MLI no es necesario memorizar ninguna ordenación de peticiones, ya que la petición PtX, correspondiente a una expulsión de bloque no requiere ser ordenada. Es para actualizar memoria. Esto es, mantener coherencia en la jerarquía observada por el procesador.

Por otro lado, durante la ventana de vulnerabilidad de una expulsión es necesario suministrar el bloque, para mantener la coherencia de cache. Una lectura obtiene el valor de la última escritura consolidada (atomicidad de las escrituras). Las escrituras a un bloque en el estado M están consolidas. Además, el bloque puede haber sido leído por el procesador cuya cache expulsa el bloque. Hay que garantizar que el orden de programa del hilo que ejecuta el procesador es observado por otro hilos.

APENDICE A: REDUCCIÓN DE LA LATENCIA DE UN FALLO

Cuando se produce una acción de reemplazo y el bloque que se expulsa está en el estado M hay que actualizar memoria.

Hasta ahora hemos supuesto que la actualización de memoria se efectúa antes de iniciar el servicio del fallo de cache (Figura 5.23). Esta forma de actuar determina que la latencia de servir el fallo de cache sea, como mínimo, el doble.

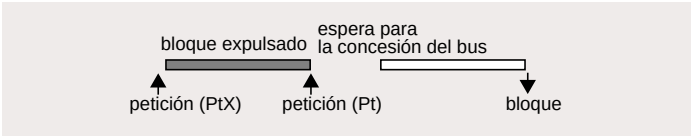


Figura 5.23 Reemplazo: actualización de memoria antes de servir el fallo.

Una forma de reducir la latencia de servicio de un fallo de cache es almacenar en un buffer el bloque expulsado e iniciar inmediatamente el servicio del fallo (Figura 5.24). Posteriormente se actualiza memoria con el bloque reemplazado.

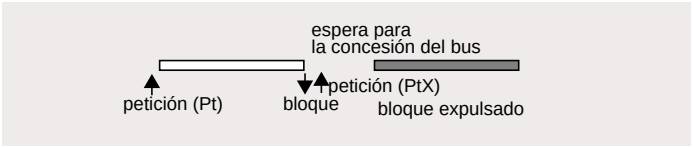


Figura 5.24 Reemplazo: actualización de memoria posterior al servicio del fallo.

El bloque expulsado se almacena en un buffer, que se denomina buffer de expulsiones (BEX). En el buffer se distinguen los mismos campos que en la cache: dirección, estado y datos (Figura 5.25).

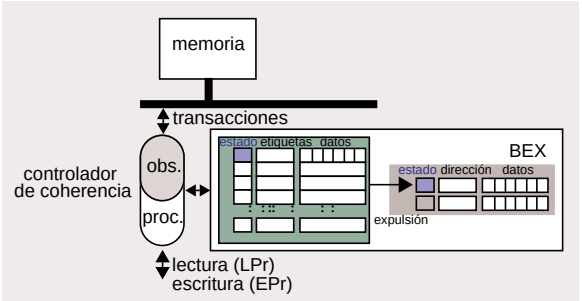


Figura 5.25 Cache con buffer de expulsiones (BEX).

Notemos que se almacena la dirección del bloque, la cual se obtiene concatenando los bits leídos del campo etiqueta con los bits utilizados para determinar el conjunto de cache. El BEX, desde el punto de vista de coherencia, debe observarse como una extensión de la cache. Esto es, los dos agentes, procesador y observador, tienen que comprobar si el bloque al que están accediendo está almacenado en el BEX.

El agente observador debe suministrar el bloque en una transacción de bus que acceda al bloque. En una acción de suministro del bloque, si no se ha solicitado acceso al bus (PtBus) no es necesario actualizar memoria explícitamente y la entrada del BEX queda libre. Si se ha solicitado acceso al bus, después de efectuar el suministro, el CC desactiva la solicitud de acceso al bus. En la Figura 5.26 se muestra el diagrama de transición entre estados de un bloque almacenado en el BEX[20].

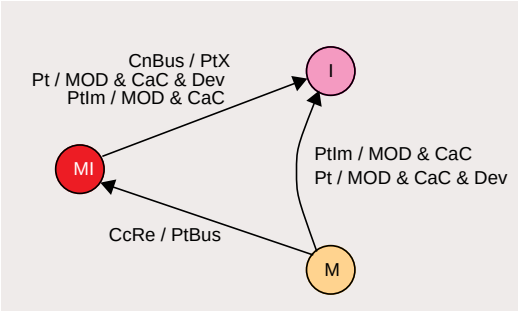


Figura 5.26 Diagrama de transiciones entre estados del buffer de expulsiones.

Usualmente el tamaño del BEX permite almacenar más de un bloque y la organización es totalmente asociativa. El BEX se intenta vaciar cuando no hay actividad de peticiones de bus en el procesador, cuando está lleno y es necesario otro reemplazo o cuando el número de entradas ocupadas llega a una marca, que usualmente se denomina marca de agua.

La posición de memoria de un acceso del procesador puede estar contenida en un bloque almacenado en el BEX[21]. Una alternativa es bloquear el procesador y actualizar memoria con el bloque referenciado. Después de actualizar memoria, se gestionará un fallo de cache[22].

Otra alternativa es disponer de circuitería en el BEX para suministrar el valor al procesador (load) y circuitería para poder actualizar una entrada en el BEX (store).

20. En los estados M y MI la copia del bloque en el BEX es válida. Además, es la única copia válida en el multiprocesador.
21. Hay que respetar las dependencias de datos del hilo que se está ejecutando en el procesador.
22. Esta serialización la gestiona el controlador de coherencia.

EJEMPLOS

Protocolo MLI

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 9 y t = 12.

Pregunta 1: *Utilice una tabla similar a la Tabla 5.11 para mostrar las transacciones de bus y estados de los bloques en las caches. La secuencia de accesos a memoria se indica en la tabla mostrada en la respuesta. En esta secuencia se agrupan los accesos a memoria de dos en dos y la agrupación se muestra mediante líneas horizontales continuas.*

Respuesta: Los dos primeros accesos concurrentes son fallos de escritura. El CC1 obtiene la concesión del bus en primer lugar. El estado transitorio de los bloques en los dos casos es IM y el estado estable final es M.

	arb.	C 1	C 2	C 3	Bus (trans.)					mem.			C 1			C 2		
accesos	ord.	est.	est.	est.	1º	2º	3º	4º	señal	var.	val.	sum.	var.	val.	est.	var.	val.	est.
1. P1 store t (2)	1º	IM			PtIm					t	9	mem.	t	2	M			
1. P2 store u (7)	2º		IM			PtIm				u	12	mem				u	7	M
2. P1 PtX t	1º	MI			PtX					t	2	C1	t	2	I			
load u	2º	IL				Pt			MOD	u	7	C2	u	7	L	u	7	I
2. P2 PtX u			I													u	7	I
load t	3º		IL					Pt	MOD	t	2	mem				t	2	L
3. P2 CcRe t			I													t	2	I
load u	1º		IL		Pt					u	7	mem				u	7	L
3. P1 load u	2º																	
4. P1 store u (12)	1º	LM			PtIm					u	7	mem	u	12	M	u	7	LM
4. P2 store u (3)	2º		LM			PtIm			MOD	u	12	C1	u	12	I	u	3	M

El segundo grupo de accesos concurrentes son fallos de lectura y existe un conflictos en las dos caches. Como los bloques que se expulsan están en el estado M, es necesario efectuar una transacción para actualizar memoria. En la tabla se observa una fila dedicada a este hecho en los dos accesos a memoria (primera fila en cada acceso a memoria). El CC1 obtiene la concesión del bus en primer lugar. Al finalizar la transacción de expulsión del bloque en la cache C1 el estado es I. Suponemos que la segunda petición en obtener el bus

también es del procesador P1. Durante la transacción el CC2 observa que el bloque referenciado es el que quiere expulsar, lo suministra, inhibe el suministro de memoria y cambia el estado del bloque al estado I (cuarta fila en la tabla, columna est.). Además, memoria se actualiza con el bloque que se transfiere por el bus. También, el CC2 desactiva la petición de bus cuyo objetivo era actualizar memoria. En la columna etiquetada como estado transitorio se indica que el estado del bloque es I (quinta fila). Seguidamente el CC2 obtiene el bus para leer el bloque que almacena la variable t. Al finalizar la transacción el estado del bloque es L.

En el tercer grupo de accesos concurrentes hay un fallo de lectura y un acierto de lectura. El CC2, debido a un conflicto en cache, debe expulsar un bloque que no ha sido actualizado. Como memoria está actualizada la expulsión no determina una transacción. La expulsión del bloque y la petición de bloque por parte del CC de P2 se muestran en dos filas consecutivas. En la transacción emitida por el CC2 responde memoria. Respecto al acierto de lectura no se indica ninguna información en la fila, ya que no se efectúa ninguna transacción.

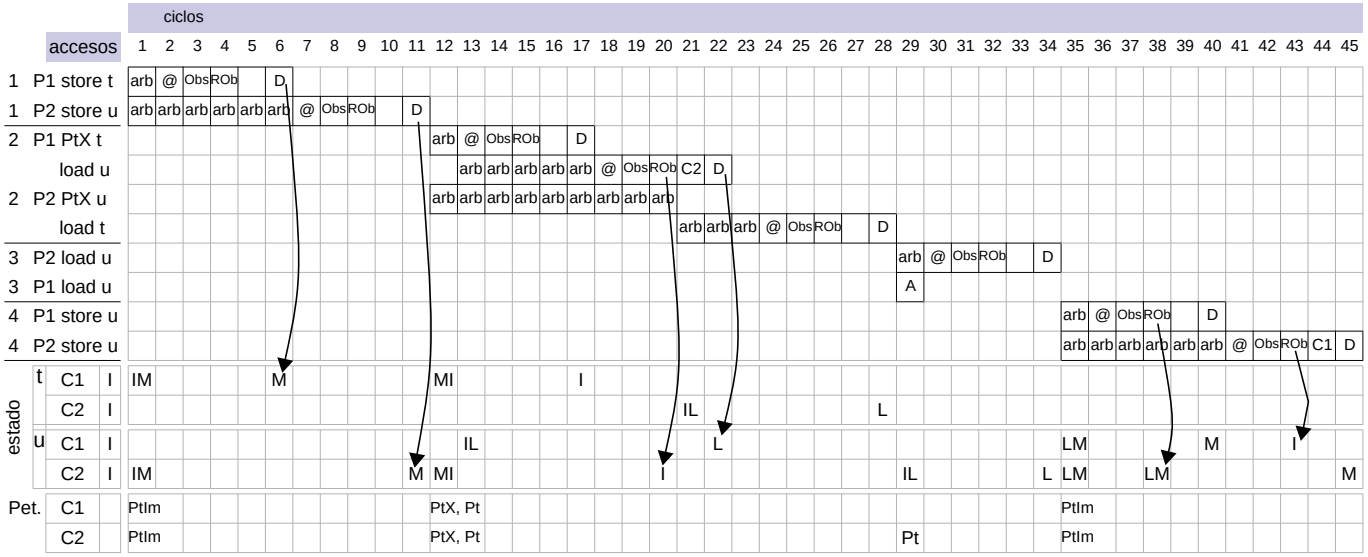
El cuarto grupo de accesos concurrentes son aciertos de escritura, siendo el estado del bloque L en las dos caches. La petición de los dos controladores de coherencia (CC1 y CC2) es lectura de bloque con intención de modificación y el estado transitorio es LM. La concesión de bus la obtiene en primer lugar el CC1. El estado estable en la cache C1 al finalizar la transacción es M y el estado transitorio en la cache C2 es LM. Notemos que la transacción y la petición pendiente referencian la misma variable. Por tanto, el CC2 debe tener en cuenta la ordenación determinada por el bus. Como la petición pendiente obtendrá como respuesta el bloque, no se modifica el estado transitorio. En la transacción iniciada por el CC2, la cache C1 suministra el bloque y activa la señal correspondiente para que la memoria se inhiba. La transacción del CC2 invalida la copia en la cache de C1. Al finalizar la transacción iniciada por el CC2, el estado estable del bloque en la cache C1 es I y en la cache C2 es M.

Pregunta 2: *Muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches.*

Respuesta: En las siguiente figura se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado.

En el segundo grupo, la petición de P1, que determina la expulsión de un bloque, solicita el bus después de que haya obtenido el bus la petición de expulsión (PtX). La segunda expulsión, en el segundo grupo de accesos a memoria, no requiere una transacción de bus, ya que el bloque ha sido

suministrado en la transacción previa del mismo conjunto de accesos a memoria. Por tanto, el CC2 desactiva la petición de acceso al bus. En el siguiente ciclo el CC2 activa una solicitud de acceso a bus para obtener el bloque que contiene la variable t.



Buffer de expulsiones

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Los bloques no están inicialmente almacenadas en cache y los valores de las variables en memoria son u = 9 y t = 12.

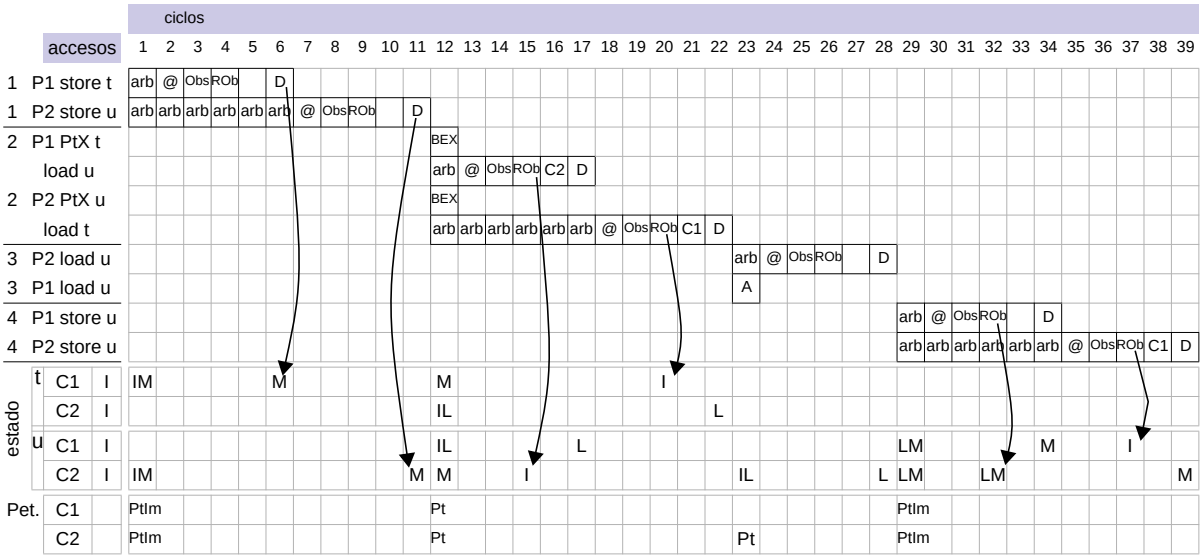
Utilice el protocolo de invalidación con escritura retardada (MLI) para mostrar las transacciones de bus y estados de los bloques en las caches. Cada cache dispone de un BEX para reducir la latencia de fallo cuando el contenedor asignado al bloque contiene un bloque en el estado M.

La secuencia de accesos a memoria se indica en el diagrama temporal mostrado en la respuesta. En esta secuencia se agrupan los accesos a memoria de dos en dos y la agrupación se muestra mediante líneas horizontales continuas.

El almacenamiento del bloque en el BEX se especifica mediante el acrónimo BEX. Cuando se almacena un bloque en el BEX no se solicita acceso al bus.

Pregunta 1: *Muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches.*

Respuesta: En las siguiente figura se muestra un diagrama temporal de la ocupación del bus y de los cambios de estado. Notemos que los bloques almacenados en el BEX se invalidan antes de efectuar una petición de expulsión. Al almacenar los bloques en el BEX no ha sido solicitado el bus. En el segundo grupo de accesos, el suministro de los bloques se efectúa desde los BEX.



EJERCICIOS

Descripción de un protocolo de observación VI denominado A

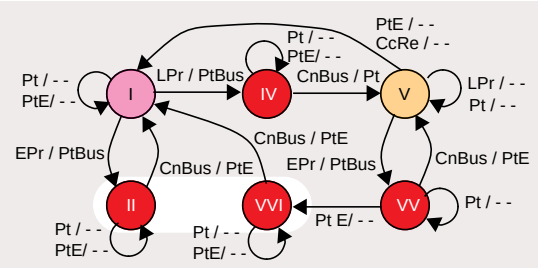
Un multiprocesador utiliza un bus como red de interconexión. Las caches privadas utilizan escritura inmediata, sin asignación de contenedor en fallo de escritura. El multiprocesador utiliza la técnica de invalidación para mantener la coherencia. En cada contenedor de cache se identifican dos posibles estados estables: inválido (I) y válido (V) y cuatro estado transitorios.

Las peticiones de procesador y las transacciones de bus son las siguientes.

Procesador	Controlador de cache	
Peticiones	**Transacciones**	**Acciones**
LPr: lectura del proc.	Pt: petición de bloque	CcRe: reemplazo de un bloque
EPr: escritura del proc.	PtE: petición de escritura de un dato	

En este multiprocesador no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Un CC compite con otros CC para obtener el bus. Mientras un CC está pendiente de que se le conceda el bus, el CC debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus.

Por ello, hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). En el siguiente diagrama de estados se muestran todas las transiciones entre estados estables y estados transitorios.



Un procesador inicia los accesos a memoria en orden de programa y se bloquea en un acceso a memoria cuando el procesador tiene una petición pendiente.

Descripción de un protocolo de observación MLI denominado B

Un multiprocesador utiliza un bus como red de interconexión. Las caches privadas utilizan escritura retardada, con asignación de contenedor en fallo. El multiprocesador utiliza la técnica de invalidación para mantener la coherencia.

En cada contenedor de cache se identifican tres posibles estados estables: inválido (I), lectura (L) y modificado (M). Además, se utilizan cuatro estados transitorios.

Las peticiones de procesador y las transacciones de bus son las siguientes.

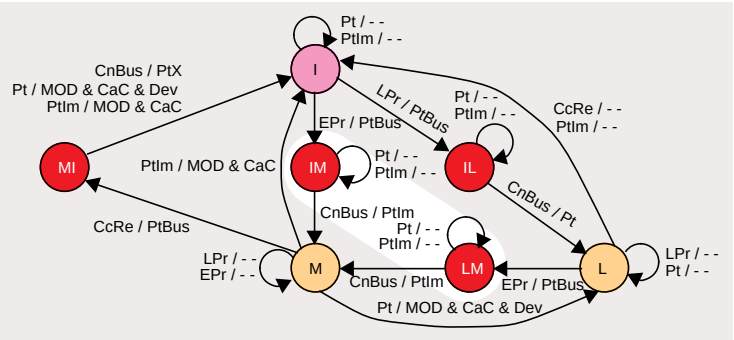
Procesador	Controlador de cache		Memoria
Peticiones	**Transacciones**	**Acciones**	**Acciones**
LPr : lectura del proc.	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: almacenar en memoria
EPr : escritura del proc.	PtIm: petición de bloque con intención de modificarlo	MOD: señal que indica bloque en estado M en una cache	
	PtX: actualización de memoria	CaC: suministro del bloque	

Una cache puede suministrar directamente el dato dentro de una transacción de bus iniciada por otro procesador (CaC). Además, en este caso se actualiza memoria, si lo determina el protocolo. Cuando una cache tiene un bloque en estado M activa la señal denominada MOD. En el bus se dispone de un cable que es la función OR de las señales MOD de cada una de las caches. Esta señal inhibe el suministro de memoria.

En este multiprocesador no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Un CC compite con otros CC para obtener el bus. Mientras un CC está pendiente de que se le conceda el bus, el CC debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus, y en su caso suministrar un bloque de cache.

Por ello, hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). El el estado MI, cuando se suministra el bloque, al observar una transacción, se desactiva la solicitud de arbitraje.

En el siguiente diagrama de estados se muestran todas las transiciones entre estados estables y estados transitorios.



Cuando es necesario efectuar una acción de reemplazo, primero se actualiza memoria, si es el caso. Posteriormente se efectuan las acciones relacionadas con el acceso que determina el reemplazo.

Un procesador inicia los accesos a memoria en orden de programa y se bloquea en un acceso a memoria cuando el procesador tiene una petición pendiente.

Descripción de un protocolo de observación MLI denominado C

Un multiprocesador utiliza un bus como red de interconexión y el protocolo de coherencia es del tipo invalidación, con escritura retardada e intervención directa con actualización de memoria. Esto es, cuando es necesario, un CC suministra el bloque (CaC) en los ciclos correspondientes de la transacción y en paralelo se actualiza la memoria, si es el caso.

Las cache privadas utilizan escritura retardada con asignación de contenedor en fallo. En cada contenedor de cache se dispone de dos bits para identificar los tres posibles estados: Inválido (I), Lectura (L) y Modificado (M).

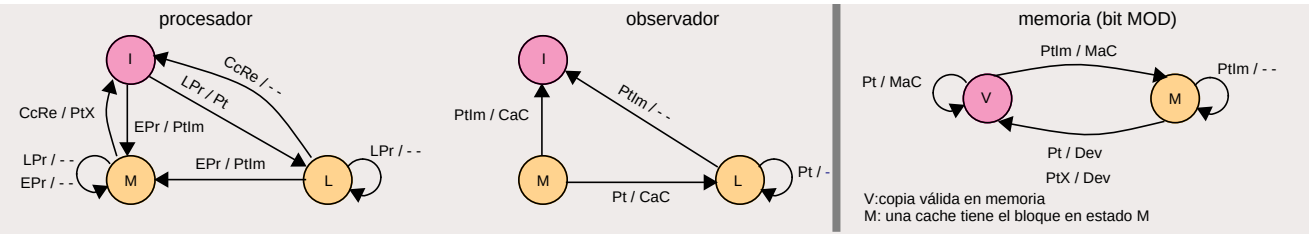
Cada bloque de memoria tiene asociado un bit que indica si una cache tiene el bloque en estado modificado (bit MOD). Este bit se utiliza para conocer si una cache suministrará el bloque. En este caso, memoria se inhibe del suministro.

Cuando un CC detecta que tiene una copia actualizada (estado M) suministra el bloque en los ciclos correspondientes de la transacción (CaC). En paralelo se actualiza la memoria (Dev), si es el caso. En los otros casos el bloque lo suministra la memoria (MaC).

Las peticiones del procesador, las transacciones de bus y las acciones son las siguientes.

Procesador	Controlador de coherencia (CC)		Memoria
Peticiones	**Transacciones**	**Acciones**	**Información / acciones**
LPr: lectura del proc.	Pt: petición de bloque	CcRe: reemplazo de un bloque	Dev: actualización de memoria
EPr: escritura del proc.	PtIm: petición de bloque con intención de modificación	CaC: suministro del bloque	bit MOD: bloque en estado M en una cache
	PtX: petición para actualizar memoria		MaC: suministro desde memoria

Los diagramas de transiciones entre estados en la cache y en memoria son los siguientes:



Cuando es necesario efectuar una acción de reemplazo, primero se actualiza memoria, si es el caso. Posteriormente se efectuan las acciones relacionadas con el acceso que determina el reemplazo. Un procesador inicia los accesos a memoria en orden de programa.

Ejercicio 5.1

El protocolo de coherencia que se utiliza es el denominado B en este capítulo.

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en el mismo contenedor. Las variables no están inicialmente almacenadas en cache y los valores en memoria son u = 9 y t = 12.

Suponga la siguiente secuencia de accesos. Las líneas más oscuras se utilizan para identificar agrupaciones de accesos concurrentes.

accesos	accesos	accesos	accesos
1. P1 store u	2. P1 load t	3. P1 store t	4. P2 store t
1. P2 store t	2. P2 store t	3. P2 load u	4. P1 store u

Pregunta 1: *Muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches al ejecutarse la anterior secuencia de accesos a memoria.*

Pregunta 2: *Indique el número total de reemplazos y el número de ellos que requieren actualizar memoria. De estos últimos, indique en cuántos el CC desactiva la petición de bus, ya que el CC ha suministrado el bloque para servir una petición en el bus.*

En una versión posterior de los procesadores, utilizados en el multiprocesador, se añade un buffer de expulsiones (BEX). El diagrama de estados de un bloque en el BEX es un subconjunto del diagrama de estados (estados M, I y MI), añadiendo una transición del estado M al estado I cuando se observa una petición Pt o PtIm.

El almacenamiento de un bloque en el BEX se especifica mediante el acrónimo BEX (su duración es una fase de una transacción de bus) y el CC no solicita acceso al bus al efectuarse el almacenamiento en el BEX.

Pregunta 3: *Utilice la secuencia de accesos previa. Muestre en un diagrama temporal la ocupación del buffer de expulsiones, del bus y los cambios de estado de los bloques en las caches. Además, suponga que el tamaño del BEX es ilimitado y por tanto no es necesario vaciarlo antes de que finalice la secuencia.*

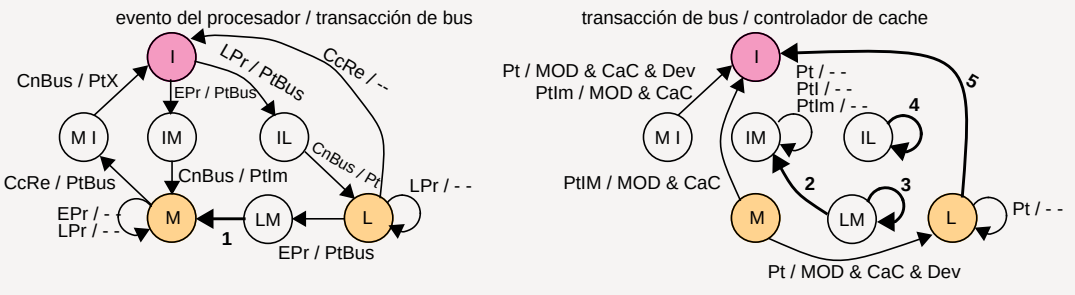
Pregunta 4: *Indique el número de bloques que han sido almacenados en el BEX y que permanecen en el BEX cuando finaliza la secuencia de accesos a memoria.*

Ejercicio 5.2

El protocolo de coherencia que se utiliza es el denominado B del Capítulo 4. Este protocolo se modifica con el mecanismo de transacción de exclusividad (PtI) descrito en el Capítulo 4.

En el protocolo descrito no se han especificado los estados transitorios que son necesarios para su implementación. Estos estados son necesarios debido a que no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Un controlador de coherencia (CC) compite con otros CC para obtener el bus. Un CC, mientras está pendiente de que se le conceda el bus, debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus, y en su caso suministrar un bloque de cache.

En el diseño que analizamos se considera atómica una transacción de bus. Por ello hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). En el siguiente diagrama de estados se muestran todas las transiciones entre estados estables y estados transitorios (IL, IM, LM, MI) y se etiquetan la mayoría de ellas con los eventos y acciones.



Pregunta 1: *Indique en las transiciones entre estados que se han etiquetado con números los eventos y acciones.*

Pregunta 2: *Suponga que se decide no utilizar la petición PtI y en su lugar se utiliza la petición PtIm. Indique en las transiciones entre estados que se han etiquetado con números los eventos y acciones. En el caso de que no se pueda producir una transición indique "no se puede producir". Dada una petición en un CC, al tener en cuenta posibles observaciones, el número de cambios de estado debe ser el mínimo posible.*

Suponga que no se distingue entre tipos de petición al efectuar una transacción. Las fases de una transacción de bus son:

referencia	ciclos 1	2	3	4	5	6
Pt, PtIm, PtX	arb	@	Obs	ROb		D
PtI	arb	@	Obs	ROb		D

Una secuencia de accesos a memoria referencia las variables u y t que están contenidas en bloques distintos. Estos bloques al almacenar en cache se ubican en contenedores distintos. Las variables no están inicialmente almacenadas en cache y los valores en memoria son u = 9 y t = 12.

Suponga la siguiente secuencia de accesos. Las líneas más oscuras se utilizan para identificar agrupaciones de accesos concurrentes.

accesos
1. P1 load u
1. P2 load u
2. P1 store u
2. P2 store u

accesos
3. P1 load t
3. P2 store t
4. P1 load t
4. P2 load t

Pregunta 3: *Cuando se utiliza la petición PtI para obtener la exclusividad de acceso a un bloque en el estado L, muestre en un diagrama temporal la ocupación del bus y los cambios de estado de los bloques en las caches al ejecutarse la anterior secuencia de accesos a memoria. Así mismo, muestre las peticiones que un CC espera emitir y la petición que finalmente emite.*

Suponga que las fases de una transacción de bus son:

referencia	ciclos 1	2	3	4	5	6	7	8	9
Pt, PtIm, PtX	arb	@	Obs	ROb		D	D	D	D
PtI	arb	@	Obs	ROb					

Pregunta 4: *Utilice la duración de las transacciones en función de la petición que ocupa el bus. Para la secuencia de accesos mostrada en el enunciado, calcule la duración en ciclos en los siguientes dos casos. En una escritura del procesador, estando el bloque en el estado L, se utiliza: a) la petición PtI y b) la petición PtIm.*

Ejercicio 5.3

En algunas implementaciones de las operaciones adquirir y liberar, utilizadas para serializar el acceso a variables compartidas, se produce bastante tráfico cuando se libera la zona de exclusión y existen hilos que están esperando, utilizando espera activa, para acceder a las variables compartidas. Todos estos hilos están observando el valor de una única variable.

Supongamos un multiprocesador donde se utiliza una política de invalidación para mantener la coherencia de cache. En las implementaciones comentadas previamente, cuando un hilo libera la zona de exclusión se invalidarán todas las copias del bloque en las otras cache y posteriormente se producirá una ráfaga de peticiones del bloque invalidado. Sin embargo, sólo uno de los hilos obtendrá el acceso exclusivo y los otros hilos volverán a la fase de espera activa. Esta fase se denomina cambio de mano.

Una forma de reducir el tráfico en un cambio de mano es utilizar una variable y dos vectores cuyo número de entradas es igual al número de hilos. Los vectores se denominan índice y espera y la variable se denomina cola. El algoritmo que se describe seguidamente se denomina vector de llaves.

Cada hilo tiene asignada estáticamente una entrada en el vector índice a la que se accede utilizando un identificador (hid). Las entradas en el vector espera se asignan dinámicamente, a medida que los hilos pretenden acceder a la zona de exclusión. La entrada asignada en una acción adquirir es la indicada por la variable cola.

Seguidamente se muestra el código de las primitivas adquirir y liberar, donde se utiliza la operación atómica fetch&add en la primitiva adquirir.

fetch&add (dir, valor)	comentario
tmp = dir	dir es una variable global
dir = tmp + valor	valor: cantidad en la que se incrementa el valor almacenado en dir
return tmp	

En el diseño del código se ha efectuado la hipótesis de que posteriormente no se va a reutilizar la estructura de datos y que el vector espera tiene N+1 elementos. Los valores de la variable hid están dentro del rango [0 . . . N - 1], siendo N el número de hilos.

adquirir ( )	liberar ( )	Inicialización	hipótesis
indice [hid] = fetch&add (cola, 1)	espera [indice [hid]] = 0	espera [0 . . . N] = [1, 0, . . . 0, 0]	N es multiplo de 4
while (espera [indice [hid]] = 0) { }	espera [indice [hid] +1] = 1	indice [0 . . . N-1] = [0, 0, . . . 0, 0]	
		cola = 0	

Las entradas en el vector espera se asignan de forma contigua. Por tanto, todas las entradas previas a la indicada por la variable cola han sido asignadas a hilos que han solicitado el acceso exclusivo previamente.

Cuando un hilo pretende acceder a la zona de exclusión, el valor de la variable cola se almacena, en el vector índice, en la entrada asignada estáticamente al hilo.

Un hilo efectúa espera activa consultando el valor de la entrada del vector espera que le ha sido asignada dinámicamente.

Los códigos previos se ejecutan en un sistema multiprocesador que no garantiza consistencia secuencial. Para garantizarla utilizaremos la directiva LNbarrera en el código. Al insertar esta directiva entre dos sentencias se inhibe en el compilador la posibilidad de reordenar operaciones de acceso a memoria. El compilador no puede ubicar después de la primitiva LNbarrera operaciones de acceso a memoria que el programador ha especificado antes de la primitiva LNbarrera y viceversa. Desde el punto de vista hardware la directiva se traduce en una instrucción INbarrera, que garantiza que las operaciones de acceso a memoria, previas a la instrucción INbarrera, han sido consolidadas antes de iniciar la ejecución de las operaciones de acceso a memoria, especificadas después de la instrucción INbarrera.

Pregunta 1: *Indique las posiciones en las cuales hay que insertar la directiva LNbarrera para garantizar consistencia secuencial. Justifique la respuesta. El número de inserciones de la directiva LNbarrera debe ser el mínimo posible.*

El protocolo de coherencia que se utiliza es el denominado C en este capítulo.

El tamaño de un bloque es 16 bytes y un elemento de los vectores y la variable cola ocupan 8 bytes. Suponga que cuando se declara un vector o una variable el compilador ubica el vector o variable, en el espacio lógico, alineada a tamaño de bloque.

En las siguientes preguntas no se tienen en cuenta los bloques accedidos mientras se ocupa la zona de exclusión.

Suponga que el multiprocesador dispone de cuatro procesadores y hay cuatro hilos. El procesador P0 ocupa la zona de exclusión y el resto de procesadores (P1, P2, P3) están en espera activa. Los hilos han obtenido las entradas del vector espera en el mismo orden que tiene asignadas las entradas del vector índice. El número de entrada en el vector índice se corresponde con el ordinal que identifica al procesador.

Pregunta 2: *El primer bloque del vector espera se almacena en el conjunto cero de una cache. En las condiciones descritas previamente, indique los bloques del vector espera que están almacenados en la cache de cada procesador, los elementos del vector espera que contiene cada bloque, el conjunto de cache que ocupa y el estado del bloque en cada cache.*

Cuando se libera la zona de exclusión por el procesador P0 se observa la siguiente secuencia de accesos a memoria.

acceso	descripción
1. P0 store R0, 0(R1); espera [0]	Liberación y despertar. P0 libera el acceso exclusivo
2. P0 store R1, 8(R1); espera [1]	
3. P1 load R3, 0(R2); espera [1]	Comprobación, entrar y esperar. P1 obtiene el acceso exclusivo
4. P2 load R3, 0(R2); espera [2]	
5. P3 load R3, 0(R2); espera [3]	
6. P1 store R0, 0(R1); espera [1]	Liberación y despertar. P1 libera el acceso exclusivo
7. P1 store R1, 8(R1); espera [2]	
8. P2 load R3, 0(R2); espera [2]	Comprobación, entrar y esperar. P2 obtiene el acceso exclusivo
9. P3 load R3, 0(R2); espera [3]	
10. P2 store R0, 0(R1); espera [2]	Liberación y despertar. P2 libera el acceso exclusivo
11. P2 store R1, 8(R1); espera [3]	
12. P3 load R3, 0(R2); espera [3]	Comprobacióny entrar. P3 obtiene el acceso exclusivo
13. P3 store R0, 0(R1); espera [3]	Liberación. P3 libera el acceso exclusivo
14. P3 store R1, 8(R1); espera [4]	

Pregunta 3: *Muestre en una tabla el estado en cache de los bloques referenciados en la anterior secuencia de accesos a memoria. Así mismo, muestre el valor de las variables en cache y en memoria y las transacciones de bus en cada acceso de los procesadores, si es el caso. En la columna variable indique el índice del vector espera. Suponga que el estado de los bloques del vector espera que están almacenados en las caches, en las condiciones descritas antes de la pregunta anterior, es L.*

	bus	mem.				C 0				C 1				C 2				C 3			
acceso	trans.	bit	var.	val.	sum.	cjto	var.	val.	est.	cjto	var.	val.	est.	cjto	var.	val.	est.	cjto	var.	val.	est.
1. P0 store R0, 0(R1); espera [0]																					

Pregunta 4: *A partir de la secuencia de accesos mostrada previamente, calcule el número de bloques invalidados en las caches, el número de peticiones de exclusividad (el bloque está en estado L) y el número de fallos (excluyendo peticiones de exclusividad). Recuerde que todos los hilos menos uno están en espera activa. Además, muestre expresiones algebraicas cuando el número de procesadores e hilos es N en lugar de cuatro. Para ello identifique un patrón y efectúe el cálculo con este patrón. Suponga que N es par.*

En las siguiente preguntas y hasta que se indique lo contrario suponga que el multiprocesador mantiene consistencia secuencial. Después de que todos los hilos hayan accedido a la zona de exclusión los vectores y la variable deben quedar inicializados para un uso posterior. El número de elementos de los vectores es igual a N.

Pregunta 5: *Modifique el algoritmo descrito al principio del enunciado, código y tamaño de las estructuras de datos, para que los vectores y variables queden inicializados para un uso posterior. Nota: el desbordamiento del valor que almacena la variable cola no es un problema, ya que el número de procesador e hilos es menor que el máximo número natural que puede almacenar la variable cola.*

Pregunta 6: *Modifique el algoritmo descrito al principio del enunciado, código y tamaño de las estructuras de datos, para que no se produzca compartición falsa. Los valores de hid siguen siendo los mismos, pero no los valores para acceder a los elementos del vector indice. Calcule el número de bloques invalidados y el número de fallos (excluyendo peticiones de exclusividad) con esta modificación. Para el resto de hipótesis utilice las de la pregunta 4.*

El lenguaje máquina de los procesador dispone de las primitivas Load Linked (LL) y Store conditional (SC).

instrucción	descripción
LL Rd, dir	la palabra leída en la posición de memoria dir se almacena en el registro Rd y se activa el bit de enlace (bit=1).
SC Rfd, dir	si el bit de enlace está activo (bit=1), almacena el contenido del registro Rfd en la posición de memoria dir y devuelve un 1 en el registro Rfd. En caso contrario no se ejecuta el store y devuelve un 0 en el registro Rfd.

Pregunta 7: *Especifique el código de una implementación de la operación atómica fetch&add con el menor número de instrucciones. La dirección de la variable está almacenada en el registro R1 y valor está almacenado en el registro R2. La función devuelve el resultado en el registro R4. Proponga una implementación para el caso de que se cumpla consistencia secuencial. Posteriormente suponga que el multiprocesador no mantiene consistencia secuencial. Utilice la instrucción INbarrera para garantizar consistencia secuencial. El número de inserciones de la instrucciones INbarrera debe ser el mínimo posible.*

Ejercicio 5.4

Suponga un multiprocesador con P procesadores, donde la organización de las caches privadas es mapeo directo y utilizan escritura retardada. El tamaño de bloque es 32 bytes y el número de contenedores es 16. Los estados de un bloque pueden ser I (inválido) y V (válido). La red de interconexión utilizada en el multiprocesador es un bus y el protocolo de coherencia es del tipo invalidación.

Las caches privadas de los procesadores son bloqueantes. En un fallo de cache se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

Una cache puede suministrar directamente el dato dentro de una transacción de bus iniciada por otro procesador. Además, en este caso se actualiza memoria. Cuando una cache tiene un bloque en estado V activa la señal denominada MOD. En la red de interconexión se dispone de un cable que es la función OR de las señales MOD de cada una de las caches. Las peticiones de procesador, las transacciones de bus y el diagrama de transiciones entre estados son las siguientes.

Procesador	Controlador de cache (CC)		Memoria	Diagrama de transiciones entre estados utilizado en los CC
Peticiones	**Transacciones**	**Acciones**	**Acciones**	
LPr : lectura	Pt : petición de bloque	CcRe: reemplazo de un bloque	Dev: almacenar en memoria	Agente procesador LPr / Pt EPr / PtIm I V LPr / - - EPr / - - CcRe / PtX & Dev Agente observador I Pt / CaC & Dev PtIm / CaC & Dev V
EPr : escritura	PtIm: petición de bloque con intención de modificarlo	MOD: señal que indica bloque en estado V en una cache		
	PtX: petición de expulsión	CaC: suministro del bloque		

Pregunta 1: *Indique cuántas copias de un bloque puede haber en las cache del multiprocesador. Justifique la respuesta.*

Dos diseñadores discuten sobre el soporte de la arquitectura para implementar operaciones de acceso exclusivo. Para ello estudian la implementación de la instrucción test&set y del par de instrucciones load linked y store conditional.

Test&set	load linked	store conditional		
TS R_d, X(R_f)	LL R_d, X(R_f)	El que ejecuta: SC R_{fd}, X(R_f)	Otros: dirección de la transacción (dir)	Comentarios
Rd = M [X + R_f^v]	R_d^v = M [X + Rf^v]	if (RLD^v = X + Rf^v and RLR^v = 1) then	if (RLD^v = $dir_{transacción}$ and RLR^v = 1) then	El superíndice v indica valor
M [X + R_f^v] = 1	RLD^v = X + Rf^v	M [X + R_f^v] = R_{fd}^v	RLR^v = 0	RLD: registro que almacena la dirección a la que accede un load linked
	RLR^v = 1	R_{fd}^v = 1		RLR: bit de enlace o reserva
		else R_{fd}^v = 0; la instrucción se convierte en una NOP		Los registros RLD y RLR los gestiona el CC
		RLR^v = 0		

El bit de enlace (RLR) se desactiva si el bloque al que hace referencia el contenido del RLD se invalida por alguna circunstancia. El CC garantiza que cualquiera de estas instrucciones se ejecuta de forma atómica. Si se produce un fallo de cache en una instrucción TS la transacción es del tipo PtIm.

Para comprobar la utilidad de las instrucciones se analiza la siguiente secuencia de accesos a memoria, observada al competir varios procesadores para acceder de forma exclusiva a la variable A.

secuencia de accesos (A)		secuencia de accesos (B)
1. P1 TS llave (1)	El valor inicial de las variables llave y A en memoria es 0 y 24 respectivamente. Inicialmente los bloque que contienen las variables no están almacenados en cache. Los bloques al mapearse en cache ocupan contenedores distintos	1. P1 LL llave
2. P1 store A (3)		2. P1 SC llave(1)
3. P3 TS llave (1)		3. P1 store A (3)
4. P2 TS llave (1)		4. P1 store llave (0)
5. P3 TS llave (1)		5. P3 LL llave
6. P1 store llave (0)		6. P2 LL llave
7. P2 TS llave (1)		7. P3 SC llave (1)
		8. P3 LL llave
		9. P2 SC llave (1)

Pregunta 2: *Para la secuencia A), muestre, mediante una tabla, el estado de los bloques que contienen las variables llave y A en las caches de cada procesador. Así mismo, muestre las transacciones de bus y quién suministra el bloque.*

Pregunta 3: *Para la secuencia B), muestre, mediante una tabla, el estado de los bloques que contienen las variables llave y A en las caches de cada procesador y el valor del bit de enlace o reserva (RLR). Así mismo, muestre las transacciones de bus y quién suministra el bloque. Si una instrucción no se ejecuta especifique el acrónimo NOP en el campo estado.*

Pregunta 4: *Justifique si la implementación de las instrucciones descritas previamente son de utilidad para obtener acceso exclusivo en cualquier circunstancia. Para ello analice las secuencias previas de acceso a memoria.*

En una implementación donde varios CC pueden competir para obtener el bus, no se puede considerar una acción atómica la detección de necesitar efectuar una transacción de bus y la realización de la misma. Por tanto, mientras un CC está pendiente de que se le conceda el bus, debe poder efectuar las acciones de observación inducidas por la transacción en curso en el bus, y en su caso suministrar un bloque de cache.

En el diseño que analizamos se considera atómica una transacción de bus. En consecuencia hay que distinguir la acción de solicitar el bus (PtBus) y la acción de concesión del bus (CnBus). Para ello se utilizan estados transitorios.

Pregunta 5: *Proponga el diagrama de transiciones entre estados cuando se considera que el bus es atómico. Utilice dos diagramas. En uno de ellos muestre las transiciones del agente procesador y en el otro las transiciones del agente observador. Un CC desactiva la solicitud de arbitraje cuando efectúa el suministro y ha soliciado el bus para expulsar el bloque.*

Pregunta 6: *Dada la siguiente secuencia de accesos indique el número de contenedor, si es acierto o fallo, si se produce una expulsión que requiere actualizar memoria y la transacciones de bus. El valor numérico es la dirección accedida.*

accesos	accesos
1. P1 load 2048	5. P2 load 3113
2. P2 store 1061	6. P1 store 3113
3. P3 load 1061	7. P1 load 2056
4. P3 store 1061	Las caches están vacías

Ejercicio 5.5

El protocolo de coherencia que se utiliza es el denominado B del Capítulo 4.

En este multiprocesador se ejecutan los siguientes dos trozos de código que implementan una sincronización mediante evento.

P0	P1	valores iniciales
A = 2. 46	while (flag =0) { };	A = flag = 0
flag = 1	y = A	
. . .	. . .	

Una simplificación del código para efectuar un análisis de consistencia secuencial es la siguiente.

P0	P1	valores iniciales	comentarios
1. store # 2.46, A	3. load R8, flag	A = flag = 0	la sentencia 4 en P1 no se ejecuta hasta que R8 almacena el valor 1
2. store R4, flag	4. load R6, A		

Las variables A y flag están ubicadas en bloques distintos de memoria y su valor inicial es cero. Al almacenarse en cache los bloques ocupan contenedores distintos.

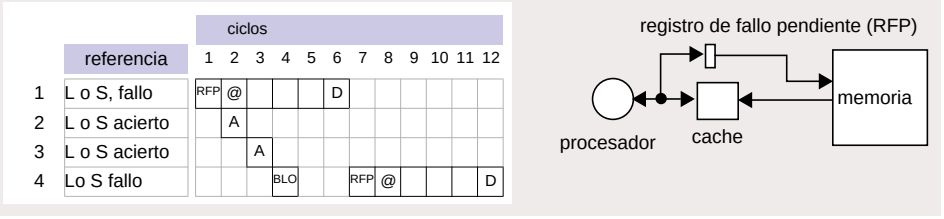
Suponga el siguiente entrelazado de acceso a memoria.

	instrucción	estado inicial
1.	P0 store A	A: estado L en C0 y C1
2.	P0 store flag	flag: estado M en C0 e I en C1
3.	P1 load flag	
4.	P1 load A	

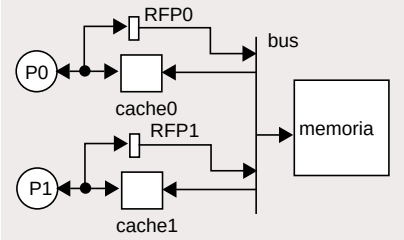
Pregunta 1: *Muestre, mediante un diagrama temporal, las transacciones de bus y los estados de los bloques en las caches.*

En una variante del procesador se permiten aciertos en cache después de un fallo de cache. Esto es, mientras que la cache está esperando el servicio de un fallo, accesos posteriores (más jóvenes) que aciertan en cache se ejecutan. Para ello el fallo se almacena en un registro denominado registro de fallo pendiente (RFP). La cache no soporta un fallo después de un fallo previo. Esto es, el procesador se bloquea cuando se detecta el segundo fallo.

En la parte izquierda de la siguiente figura se muestra un ejemplo donde se observa como dos instrucciones de acceso a memoria, que aciertan al acceder a cache (2, y 3), adelantan a una instrucción de acceso a memoria más vieja (1) que ha fallado al acceder a cache.



El acrónimo A indica acierto en cache. También se muestra una instrucción de acceso a memoria más joven (4) que falla en cache antes de que finalice el servicio del fallo previo. El procesador se bloquea hasta que finaliza el servicio previo (BLO).



Este procesador se utiliza en el diseño de un multiprocesador cuya red de interconexión es un bus y las caches son privadas. En la figura adjunta se muestra un esquema de un multiprocesador con dos procesadores.

En las siguientes preguntan debe considerarse que las caches son no bloqueantes en el primer fallo.

Cuando se especifique una secuencia de accesos a memoria, la ejecución de una instrucción de acceso a memoria, que requiere acceder al bus y no lo obtiene, se especificará mediante dos filas. En la primera fila se especifica la instrucción de acceso a memoria, la cual se almacena en el RFP. Posteriormente, después de otros accesos a memoria se especificará un acceso a memoria desde el RFP para indicar la petición de bloque.

	ciclos											
	1	2	3	4	5	6	7	8	9	10	11	12
L o S	RFP											
. . .												
RFP L o S							arb	@	Obs	ROb		D

En estas condiciones suponga el siguiente entrelazado de accesos a memoria y peticiones desde el RFP.

	instrucción	comentario	estado inicial
1.	P0 store A	inserción en RFP	A: estado L en C0 y C1
2.	P0 store flag		flag: estado M en C0 e I en C1
3.	P1 load flag		
4.	P1 load A		
5.	P0 RFP store A	petición desde RFP	

Pregunta 2: *Muestre, mediante un diagrama temporal, que el multiprocesador no cumple las condiciones de consistencia secuencial, al ejecutarse el entrelazado previo de accesos a memoria y peticiones desde el RFP. Esto es, hay que mostrar que P1 observa los accesos de P0 en un orden distinto al orden de programa.*

El bus dispone de una señal que permite que los procesadores intervengan en una transacción activando esta señal. Por intervención entendemos que se anula la transacción en curso. Esto es, los CC actúan como si no se hubiera efectuado la transacción. El CC al cual se le ha anulado una transacción debe, posteriormente, volver a solicitar el bus para que se repita la transacción.

Pregunta 3: *Indique como se puede aprovechar la señal de intervención descrita para mantener consistencia secuencial en el código anterior (en general no es factible). En una cache que está observando una transacción, especifique exclusivamente las acciones que se efectúa para activar la señal de intervención.*

Suponga que cuando una transacción es anulada pasa a la cola de peticiones pendientes del árbitro. En un diagrama temporal indique el periodo en el que actuaría la acción de anulación. Para ello, utilice filas adicionales después de las filas utilizadas para indicar el estado de los bloques.

En estas condiciones suponga el siguiente entrelazado de accesos a memoria y peticiones desde el RFP.

	instrucción	comentario	estado inicial
1.	P0 store A	inserción en RFP	A: estado L en C0 y C1
2.	P0 store flag		flag: estado M en C0 e I en C1
3.	P1 load flag		
4.	P0 RFP store A	petición desde RFP	
5.	P1 load flag		
6.	P1 load A		

Pregunta 4: *Muestre, mediante un diagrama temporal, que utilizando el mecanismo de intervención se cumplen las condiciones de consistencia secuencial, al ejecutarse el entrelazado previo de accesos a memoria.*

El mecanismo descrito requiere, para que funciones correctamente, que únicamente exista una petición pendiente en el multiprocesador. Esto es, si ya existe una petición pendiente, un fallo en otro procesador requiere que el procesador se bloquee.

Para responder a la siguiente pregunta suponga las siguientes secuencias de instrucciones correspondientes a dos hilos.

	hilo 1	
1	store A	fallo
2	store flag1	acierto en estado M
3	load flag2	fallo
4	load B	

	hilo 2	
5	store B	fallo
6	store flag2	acierto en estado M
7	load flag 1	fallo
8	load A	

Pregunta 5: *Utilice las secuencias de código previo para razonar que se produce abrazo mortal si se permiten en el multiprocesador varios fallos pendientes.*

Otra posibilidad para garantizar consistencia secuencial es mediante la inserción de la instrucción INbarrera, la cual al interpretarse bloquea al procesador si hay fallos pendientes que no han obtenido el bus. Esto es, no se ejecuta ninguna instrucción posterior hasta que la petición en RFP obtiene el bus. En estas condiciones el sistema no utiliza el mecanismo de anulación de transacciones.

Pregunta 6: *Indique en el código simplificado las posiciones en las cuales debe incluirse la instrucción INbarrera. Justifique la respuesta.*

Ejercicio 5.6

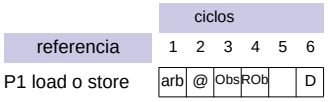
El protocolo de coherencia que se utiliza es el denominado C en este capítulo. Un ingeniero decide añadir al protocolo la funcionalidad denominada transacción de exclusividad (petición PtI), descrita en el Capítulo 4 junto con la descripción suscinta de los protocolos.

Pregunta 1: *Muestre las transiciones entre estados en el agente procesador, el agente observador y en el autómata de memoria cuando se utiliza en el protocolo la petición PtI.*

Como programa de prueba utilizaremos el que se muestra en la parte izquierda de la siguiente tabla. Las variables a y b se ubican en bloques distintos y al almacenarse en cache utilizan contenedores distintos. En la parte derecha de la misma tabla se muestra el entrelazado de accesos a memoria que utilizaremos, los valores de las variables en memoria y en cache, si es el caso, y el estado de los bloques en las caches.

hilo 1	hilo 2	accesos a memoria	valores en memoria	estado en las caches	
a = 3.2	while (b = 0) { }	1. P1 store a (3.2)	a = 8	C1	C2
b = 1	T = a	2. P1 store b (1)	b = 10	a: L	a: L
		3. P2 load b		b: M	b: I
		4. P2 load a		b = 0	

En los diagramas temporales que se soliciten una transacción de bus se representa de la siguiente forma.



En los diagramas temporales añada filas adicionales al final para representar el valor de las variables en las caches.

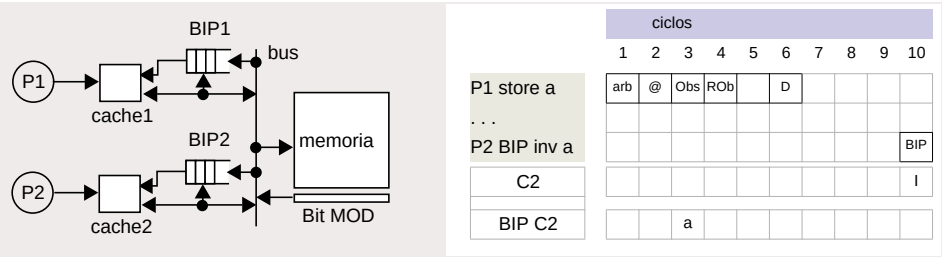
Pregunta 2: *Muestre en un diagrama temporal la secuencia de accesos especificada previamente. Un acceso a memoria que no genera transacción se efectúa en tiempo cero y la fila correspondiente en el diagrama temporal se deja vacía (si se modifica el valor se indica en la fase arb de la siguiente transacción). Indique el número de accesos a memoria que no generan transacción e identifíquelos con el número de orden.*

Una petición PtI requiere que, durante la transacción de bus, las caches que tienen copia del bloque deban invalidarlo. En ocasiones, la actividad en algunos procesadores puede requerir extender la duración de la transacción, para que se realice explícitamente la invalidación del bloque. Esta extensión de la duración de una transacción retrasa la concesión del bus a otros procesadores que quieran utilizarlo y en consecuencia se reduce el rendimiento.

Para no tener que extender la duración de una transacción, una posible solución es añadir en los CC un buffer que almacene invalidaciones pendientes (BIP) en la cache asociada. En estas condiciones, la respuesta a la petición de invalidación, implícita en la transacción de bus, es la inclusión de la petición de invalidación en el BIP (respuesta temprana). Además, para respetar las dependencias de datos que determina el orden de bus, esta petición será procesada por el CC antes de que éste emita cualquier petición relacionada con el bloque al que hace referencia.

Los accesos a memoria, que no referencian bloques con peticiones de invalidación almacenadas en el BIP, pueden generar una transacción antes de procesar las peticiones de invalidación almacenadas en el BIP y recibir la respuesta.

En la parte izquierda de la siguiente figura se muestra un esquema de un multiprocesador con dos procesadores que utilizan un BIP.



Cuando se especifique una secuencia de accesos a memoria, el almacenamiento en BIP se especificará en una fila utilizada para este propósito, ubicada después de las filas dedicadas a la representación del estado de las cache (parte derecha de la figura previa). El instante de inserción en el diagrama temporal se corresponde con la columna que determina la fase Obs de la

transacción. Posteriormente, usualmente después de otros accesos a memoria, se especificará el procesado de la invalidación almacenada en el BIP mediante “Px BIP inv variable”. El procesado de una entrada en el BIP es un ciclo.

En la secuencia anterior de accesos a memoria, las peticiones de invalidación se almacenan en el BIP de los procesadores. Añada al final de la secuencia la siguiente transacción pendiente de finalizar en P2, la cual se realiza desde el BIP, “P2 BIP inv a”, que indica una invalidación pendiente que referencia el bloque que almacena la variable a. Recordemos que cuando una invalidación se almacena en el BIP no se invalida el bloque en la cache. La invalidación se produce cuando se especifica “BIP inv”.

En una transacción Pt o PtIm todos los CC comprueban si tienen el bloque en el estado M. El procesado de esta transacción adelanta al procesado de cualquier entrada en el BIP, que son peticiones PtI.

Pregunta 3: *Al añadir un BIP a cada procesador muestre, mediante un diagrama temporal, que la secuencia de accesos produce un resultado incorrecto. Indique el número de accesos a memoria que no generan transacción e identifíquelos con el número de orden. Indique el valor que lee P2 al acceder a la variable a.*

Pregunta 4: *En una frase, exponga una razón por la cual el resultado es incorrecto.*

Un mecanismo para que el resultado sea correcto es utilizar una instrucción de lenguaje máquina denominada INbarrera. Cuando esta instrucción se ejecuta, todas las entradas ocupadas en el BIP son procesadas antes de que el procesador ejecute una instrucción load más joven. En las condiciones que se acaban de describir, se inserta una instrucción INbarrera entre las dos instrucciones del hilo 2 (mostrado en la parte izquierda de la siguiente tabla). En la parte derecha de la misma tabla se muestra el entrelazado de accesos a memoria que utilizaremos, los valores en memoria de las variables, en cache, si es el caso y el estado de los bloques en las caches.

hilo 1	hilo 2	accesos a memoria	valores en memoria	estado en las caches	
a = 3.2	while (b = 0) { }	1. P1 store a (3.2)	a = 8	C1	C2
b = 1	INbarrera	2. P1 store b (1)	b = 10	a: L	a: L
	T = a	3. P2 load b		b: M	b: I
		4. P2 INbarrera		b= 0	
		1.a. P2 BIP inv a			
		5. P2 load a			

Suponga que el tiempo de ejecución de la instrucción INbarrera es cero. El procesado de las entradas del BIP, que determina la instrucción INbarrera, se especifica en la siguiente fila de la secuencia que se ha mostrado ("1.a P2 BIP inv a").

Pregunta 5: *Muestre, mediante un diagrama temporal, que la secuencia de accesos mostrada previamente produce resultados correctos. Indique el número de accesos a memoria que no generan transacción e identifíquelos con el número de orden. Indique el valor que lee P2 al acceder a la variable a.*

Ejercicio 5.7

Un CC dispone de circuitería en el BEX para suministrar valores al procesador y actualizar un bloque.

Pregunta 1: *Diseñe el diagrama de transiciones entre estados de un bloque en el BEX.*

Ejercicio 5.8

En un multiprocesador las caches privadas son de mapeo directo, utilizan escritura retardada y son bloqueantes. En un fallo de cache o en una solicitud de exclusividad se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción. El multiprocesador dispone de 8 módulos de memoria. En estos módulos de memoria, los bloques están entrelazados utilizando los bits menos significativos de la dirección. La red de interconexión utilizada en el multiprocesador es un bus segmentado y se utiliza el protocolo de invalidación MLI, denominado B.

El arbitraje del bus es cada 3 ciclos y en el siguiente diagrama se muestran las distintas fases de una transacción de bus y el solapamiento de las fases. El bus lógico son dos buses físicos. El bus por el que se transmiten peticiones y el bus por el que se transmiten respuestas de observación y el bloque.

El número de transacciones concurrentes es 3 y cada una de ellas es a un módulo de memoria distinto. El árbitro no concede el bus a una petición que referencia a un módulo de memoria que está siendo accedido por una transacción en curso. La respuesta de observación (ej. señal MOD) se efectúa a una latencia fija desde el inicio de una transacción. El orden de las peticiones, que observan los CC, es el que determina el arbitraje.

		ciclos																								
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
peticiones	Módulo M1	Ar	d0																							
	Módulo M2 ≠ M1				Ar	d1																				
	Módulo M3 ≠ M2 ≠ M1							Ar	d2																	
	Cualquier módulo										Ar	d3														
														Ar	d4											
respuestas						d0		Ro			D	D														
									d1		Ro			D	D											
												d2		Ro			D	D								
															d3		Ro			D	D					
																		d4		Ro			D	D		

Ar: arbitraje
d0 en peticiones: emisión de la dirección
d0 en respuestas: indicación de transferencia (etiqueta de la petición)

Ro: respuesta de observación
D: transmisión de datos

Tiempo de ciclo: 13.3 ns

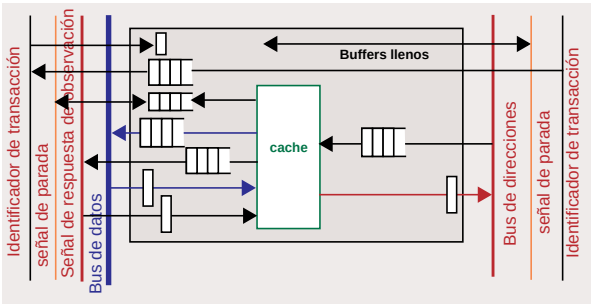
La fase Obs no se muestra en la figura. Se inicia en el ciclo siguiente al de la transmisión de la dirección

El acceso a los campos de la cache, por parte del agente observador, se efectúa en serie. El acceso al campo etiquetas requiere de 3 ciclos y el acceso al campo de datos requiere de 4 ciclos. La latencia de acceso a memoria son 7 ciclos. El agente observador utiliza un duplicado de etiquetas para realizar la acción de observación.

Pregunta 1: *Razone la causa de que el acceso al campo etiquetas no tenga una duración mayor de 3 ciclos. Por ejemplo, 4 ciclos.*

En este diseño el agente procesador tiene prioridad respecto del agente observador para acceder a la cache. En estas condiciones, se puede disponer de una señal de parada para extender la duración de la transacción. Ahora bien, como el bus es síncrono se extiende la duración de todas las transacciones en curso.

Pregunta 2: *¿Puede ser necesario la señal de parada por alguna otra circunstancia?. Analice una ráfaga de transacciones.*

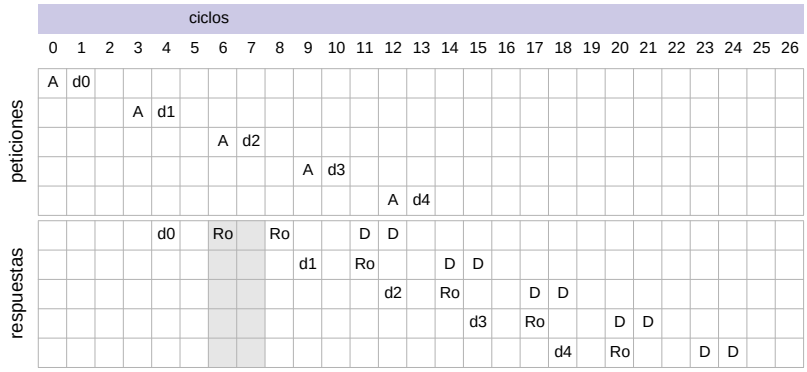


Parar todas las transacciones al unísono representa una perdida de rendimiento. Por ello, se utilizan dos buses. Uno de ellos se utiliza para transmitir la petición y el otro para transmitir la respuesta, la cual incluye la fase de respuesta de observación.

En estas condiciones, las peticiones tienen una etiqueta que se asignar al conceder el bus para emitir la petición. Las respuestas utilizan esta etiqueta para identificarse.

Las respuestas utilizan el bus en el mismo orden que las peticiones y no hay arbitraje. El módulo de memoria accedido indica el inicio de la respuesta 3 ciclos después de haberse emitido la petición, si el bus de repuestas no está ocupado. El envio de datos es 5 ciclos después de esta indicación. La respuesta de observación (e.j. señal MOD) se transmite 2 ciclos después de la indicación de inicio de la respuesta. En este ciclo se puede indicar que se pare el bus de respuestas (parada). Todas las respuestas en curso se paran al unísono.

Cada vez que se activa la señal de parada, el bus de respuestas se para durante 2 ciclos (actual y siguiente). La señal de parada se puede activar las veces que sea necesaria, cada 2 ciclos. En la figura se muestra la activación, una vez, de la señal de parada del bus de respuestas (ciclo 6), debido a que un CC no puede suministrar el dato y activa la señal de parada en el ciclo de respuesta de observación.



Las peticiones, emitidas por el bus de direcciones, se almacenan en cada CC en una cola FIFO junto con el identificador de la transacción. El CC analiza las peticiones en orden. Por tanto, el bus determina el orden en el cual los CC observan las peticiones. Si la cola de algún CC está llena se para el bus de peticiones. Mediante esta cola cada CC conoce la respuesta que se está transmitiendo por el bus.

La parada de cada uno de los buses (direcciones, respuestas) es independiente.

Dada una ráfaga de peticiones, se puede incrementar el rango de ciclos de que dispone un agente observador para acceder al campo de datos de la cache. Para ello, se reduce la latencia de inicio en el bus de direcciones. Esto

es, en lugar de emitir una petición cada 3 ciclos se puede emitir una petición cada 2 ciclos. La latencia de inicio en el bus de respuestas sigue siendo 3 ciclos.

Pregunta 3: *¿Cuál debe ser la latencia de acceso al duplicado de etiquetas para que un CC pueda suministrar el bloque en dos transacciones consecutivas de una ráfaga?.*

Pregunta 4: *Dada una ráfaga de 4 peticiones, muestre en un diagrama temporal, para cada transacción en curso, el rango de ciclos que tiene un agente observador para acceder al campo de datos de la cache. Suponga que el acceso al campo etiquetas es 1 ciclo, después de que se transmita la dirección por el bus y que el acceso al campo de datos requiere 4 ciclos. Suponga que, en peticiones consecutivas de la ráfaga, el agente observador que debe suministrar el bloque es distinto. Indique el número de ciclos de holgura que tienen la 2ª, 3ª y 4ª petición.*

Pregunta 5: *Dada una ráfaga de 4 peticiones, suponga que el primer acceso para el bus 2 veces. Muestre en un diagrama temporal, para cada transacción en curso, el rango de ciclos que tiene un agente observador para acceder al campo de datos de la cache. Suponga que el acceso al campo etiquetas es 1 ciclo, después de que se transmita la dirección por el bus y que el acceso al campo de datos requiere 4 ciclos. Suponga que, en peticiones consecutivas de la ráfaga, el agente observador que debe suministrar el bloque es distinto. Indique el número de ciclos de holgura que tienen la 2ª, 3ª y 4ª petición.*

Un procesador inicia las instrucciones en orden de programa. Si en un mismo ciclo hay más de una petición de acceso, obtiene acceso al bus, si es factible, la petición que ha sido especificada previamente en la secuencia de acceso.

En el multiprocesador se ejecutan los siguientes dos hilos. En la parte derecha se muestra un entrelazado de las peticiones que deben considerarse. El orden de la secuencia de accesos no indica el orden de acceso al bus. El acceso “P2 load aviso” representa el bucle. Entonces, en la fila correspondiente indique todos los accesos que son acierto (A), fallo (F) y la transición de bus, si es el caso.

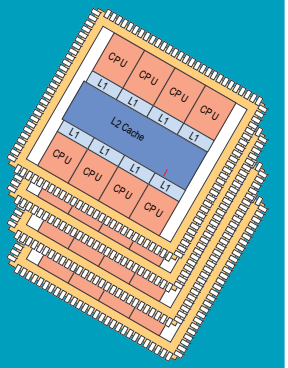
H1	H2	Comentarios	Accesos
A = 34.23	while (aviso <> 1) { }	Las variables A y aviso están ubicadas en los módulos de memoria M1 y M2 respectivamente.	P1 store A P1 store aviso P2 load aviso P2 load A
aviso = 1	T = A	El valor de las variables A y aviso inicialmente es 1.34 y 0 respectivamente. Las variables A y aviso están almacenada en la cache C2 en el estado L.	

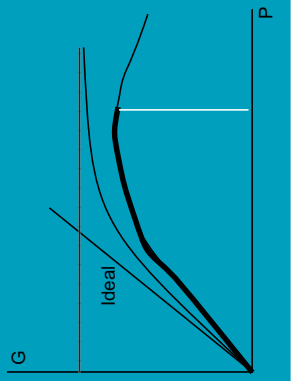
Pregunta 6: *Muestre un diagram temporal de la secuencia de accesos previa.*

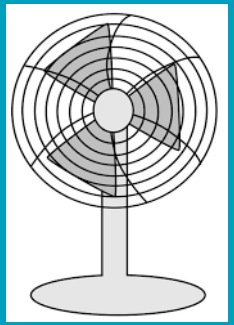
En el multiprocesador se ejecutan los siguientes dos hilos. En la parte derecha se muestra un entrelazado de las peticiones que deben considerarse.

H1	H2	Comentarios	Accesos
aviso1 = 1	aviso2 = 1	Las variables aviso1 y aviso2 están ubicadas en los módulos de memoria M1 y M2 respectivamente.	P1 store aviso1 P2 store aviso2 P1 load aviso2 P2 load aviso1
H = aviso2	T = aviso1	El valor de las variables aviso1 y aviso2 inicialmente es 0. Las variables aviso1 y aviso2 están almacenada en la cache C2 en el estado L.	

Pregunta 7: *Muestre un diagram temporal de la secuencia de accesos previa.*







Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 6
Protocolo de directorio con una red ordenada

La utilización de un bus como red de interconexión tiene debilidades desde el punto de vista de implementación. Un bus es un elemento al que se conectan directamente los nodos[1] y la memoria. La frecuencia de funcionamiento del bus está determinada por la carga eléctrica que representan los nodos y la memoria y la longitud de los cables que permite su conexión.

Las redes de interconexión de alto rendimiento se implementan mediante interconexiones punto a punto en lugar de buses. Una conexión punto a punto permite transmitir información a mayor frecuencia que un bus. Además, varias interconexiones punto a punto ofrecen concurrencia.

Un bus facilita la implementación de protocolos de coherencia debido, a su capacidad de difusión a todos los nodos y a la serialización de todas las transacciones. Entonces, los diseñadores emulan el comportamiento de un bus mediante redes punto a punto. Sin embargo, características tales como difusión y serialización, disponibles de forma natural en un bus, son un reto en la emulación.

Por otro lado, probablemente es mejor adaptar el protocolo de coherencia a las características de la red, en lugar de emular un protocolo diseñado para otro tipo de red con características particulares.

Un protocolo de coherencia rastrea u observa el estado de los bloques. Por tanto, tiene constancia del estado de compartición. En un protocolo de observación, debido a que la información está distribuida, se exploran todos los nodos. Cada cache que tiene copia de un bloque tiene copia del estado de compartición.

1. Un nodo es el par formado por un procesador y su cache privada.

Por tanto, una deficiencia de un protocolo de observación es que todos los nodos están involucrados en una acción de coherencia aunque no tengan el bloque. Una petición se difunde a todos los nodos. El caso extremo de ineficiencia es cuando no hay copia del bloque referenciado en la transacción en ninguna cache. El conjunto de agentes observador, con un bus como red de interconexión, efectúa tantas comparaciones como nodos multiplicado por la asociatividad de la cache para comprobar que no hay copias del bloque. Por otro lado, el mecanismo de difusión no es escalable, si aumentamos el número de buses con el objetivo de incrementar el ancho de banda. Cada agente observador debe observar todas las transacciones, que son transportadas por todos los buses, aunque no tenga el bloque almacenado en cache.

Para eliminar la difusión, que representa la exploración de todos los nodos, la idea es centralizar la información de presencia de copias en los nodos en una estructura denominada directorio. Entonces, mediante la indirección que representa enviar la petición al directorio, se filtra la observación de la petición en nodos que no tengan copias del bloque. Desde el directorio sólo se envían acciones de coherencia a los nodos que tiene constancia de que tienen copia.

El diseño con directorio se adecua en mayor media a una red con conexiones punto a punto. Sin embargo, desde el punto de vista hardware, respecto a un protocolo de observación, un protocolo de directorio representa un coste adicional. Es necesaria una estructura que almacene el estado de los bloques e identifique los procesadores que tienen copia.

Los mecanismos de coherencia que se sustentan en directorio pueden utilizar los mismos protocolos VI y MLI descritos de forma funcional en el Capítulo 2.

En este capítulo nos centraremos en utilizar una red tipo crossbar como red de interconexión. Esta red permite implementar de forma sencilla la ordenación, cuando es necesaria, de las peticiones y respuestas que realiza el directorio a los controladores de coherencia. Esta característica permite, de igual forma que en el bus, que no sea necesario que las caches respondan explícitamente las peticiones de invalidación. La idea en este capítulo es centrarse en la descripción de protocolos sencillos de directorio. La acción que denominamos transacción, que en el caso del bus incluye la petición y la respuesta, en un protocolo de directorio será una secuencia de mensajes desde un nodo al directorio y desde el directorio a los nodos, que pueden ser tanto mensajes de petición como de respuesta (Figura 6.1).

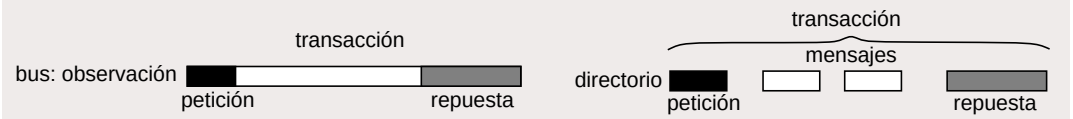


Figura 6.1 Transacción en un bus y en un esquema de directorio.

En la descripción que se efectúa en los apartados genéricos no se tiene en cuenta un único protocolo (VI, MLI). Los comentarios y mecanismo descritos pueden ser aplicables a uno de los protocolos o a ambos. La aplicación concreta a cada protocolo se efectúa en el apartado dedicado al protocolo.

DESCRIPCIÓN FUNCIONAL

Cada bloque de memoria tiene asociada una entrada en el directorio y supondremos que su gestión la realiza el controlador de memoria, el cual se encarga de mantener la coherencia en colaboración con los controladores de coherencia (Figura 6.2).

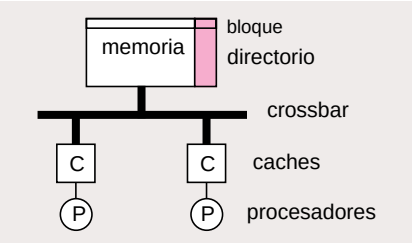


Figura 6.2 Esquema de multiprocesador con directorio.

Un protocolo de coherencia con directorio se convierte en una secuencia de mensajes punto a punto en los que sólo están involucrados los nodos "relevantes"[2]. Podemos decir que el directorio es un filtro para reducir el número de observaciones, en las caches privadas, que se efectúan por transacción. En la parte izquierda de la Figura 6.3 se muestra un esquema linealizado del flujo de información y en la parte derecha un esquema sin linealizar.

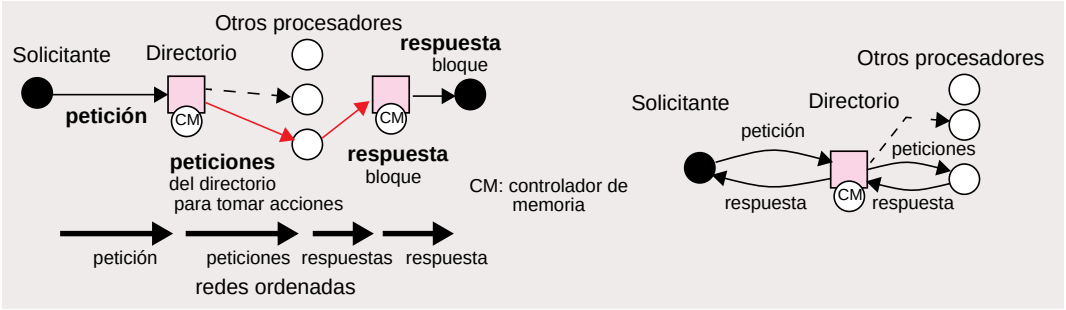


Figura 6.3 Esquema funcional de un protocolo de directorio.

- Una petición de un nodo se encamina al directorio, que es donde está la información de coherencia, lo cual representa una indirección.

2. Por relevante se entiende un nodo que tiene una copia del bloque.

- El controlador de memoria (CM), utilizando la información almacenada en el directorio, envía a los controladores de coherencia (CC) relevantes, que gestionan las caches, las ordenes oportunas para mantener la coherencia.
- Los CC atenderán las solicitudes de cambio de estado solicitadas desde el CM y en su caso, el suministro del bloque. Este suministro conlleva enviar el bloque al CM y posteriormente desde éste al nodo que lo ha solicitado.

A la vista de la descripción podemos notar que un protocolo de directorio puede representar un incremento de latencia respecto de un protocolo de observación. El CM tiene que solicitar el bloque a una cache cuando la cache lo tiene en exclusividad.

Dado un protocolo de observación, en una transacción se distingue la petición y la respuesta, pero notemos que si la petición del bloque es con intención de modificación o es una petición de escritura, hay implícitamente peticiones de invalidación en las caches que observan la transacción (Figura 2.4). En cambio, estas peticiones de invalidación en un protocolo de directorio son explícitas (desde el directorio a las caches), de la misma forma que lo es la petición de suministro, si una cache tiene el bloque en exclusividad. En un protocolo de observación las peticiones implícitas se efectúan en paralelo con la transacción. En un protocolo de directorio se efectúan de forma causal. Esto es, hay serialización, ya que hay que acceder al directorio para conocer el estado de compartición del bloque.

Información en el directorio

El directorio dispone de una entrada por cada bloque de memoria. Esta entrada se denomina vector de presencia y cada bit representa un nodo. El objetivo es identificar de forma explícita los nodos que tienen copia del bloque. La posición de los bits en el vector de presencia (VP) indica el nodo al que representan (Figura 6.4).

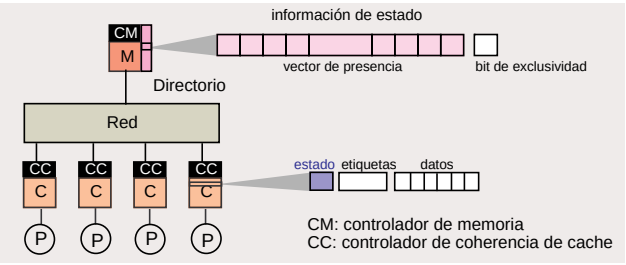


Figura 6.4 Información en el directorio.

Cuando el protocolo de coherencia lo requiera hay un bit adicional denominado de exclusividad (BE). Este bit indica que la copia, que tiene el único nodo identificado en el VP, puede estar actualizada.

Para la descripción del protocolo utilizaremos una organización centralizada del directorio. En un capítulo posterior, se muestra como incrementar el ancho de banda utilizando una organización distribuida.

Punto de ordenación de los accesos a un bloque

En un protocolo de directorio, el directorio mantiene información de las copias de los bloques en las caches y su estado. Los CC emiten los accesos a memoria relevantes al directorio (Figura 6.3)[3]. En estas condiciones, el directorio puede ser elegido, de forma natural, como el punto de ordenación de los accesos a un bloque.

Propagación de escrituras. Cuando el directorio procesa una escritura, desde el CM se emiten las peticiones oportunas a los CC involucrados en la acción de coherencia para que invaliden la copia del bloque.

Serialización de escrituras. El directorio espera a que finalicen (y se confirmen, si es el caso) todas las acciones de coherencia[4], inducidas por la petición que está procesando, antes de iniciar el procesado de otra petición al mismo bloque. Cuando sea necesaria la participación de terceros (otros CC), para servir una petición de coherencia, es necesario identificar que el procesado está en curso. Una forma de identificar un procesado en curso es utilizar estados transitorios en el directorio. Mientras un bloque en el directorio está en un estado transitorio, el CM no procesa otras peticiones que accedan el mismo bloque[5].

HIPÓTESIS EN LA DESCRIPCIÓN DE UN PROTOCOLO

En este capítulo efectuamos hipótesis similares a las utilizadas en el Capítulo 4.

Un procesador ejecuta las instrucciones en orden de programa y las caches son bloqueante.

3. En función del protocolo de coherencia: fallo de cache, escritura, exclusividad en el acceso al bloque.

4. La petición es serializada con respecto a todas las caches que tienen copia del bloque. La red de comunicación entre el CM y los CC, utilizada en este capítulo, mantiene el orden de los mensajes emitidos. En consecuencia no es necesario esperar la respuesta a peticiones de invalidación. Esta respuesta está implícita cuando el CM emite el mensaje. Por tanto, al emitir los mensajes de invalidación la escritura está consolidada.

5. En este capítulo no se observa este hecho debido a que se supone que sólo hay una transacción en curso.

El multiprocesador dispone de un directorio centralizado al que acceden los CC utilizando enlaces punto a punto. El acceso al directorio está determinado por un árbitro.

En el multiprocesador sólo existe un acceso a memoria en curso [6] [7].

En este capítulo los estados transitorios en los CC y en el CM se utilizan para identificar la emisión de una petición y la recepción de una respuesta.

El camino de comunicación del CM a los CC mantiene el orden de emisión de los mensajes desde el CM a los CC. Por tanto, todos los CC, involucrados en una acción de coherencia, observan los mensajes emitidos desde el directorio en el mismo orden lógico. Esta característica permite, de igual forma que en el bus, que no sea necesario que los CC respondan explícitamente las peticiones de invalidación. Posteriormente, en cada protocolo que se describe, se detalla la red utilizada.

PROTOCOLO DE DIRECTORIO CON INVALIDACION Y ESCRITURA INMEDIATA

En este apartado se describe en primer lugar la secuencia de mensajes utilizada en una transacción de coherencia. Posteriormente se describe la organización del multiprocesador y los agentes de coherencia en las caches y el CM. Seguidamente se detallan los tipos de mensajes utilizados y finalmente se describen los estados y transiciones entre estados.

A este protocolo lo denominamos protocolo de directorio VI.

Descripción funcional de la secuencia de mensajes en un transacción

La descripción funcional del protocolo de coherencia VI se ha efectuado en el Capítulo 4. En este apartado nos centraremos en la secuencia de mensajes de una transacción. Es interesante comparar la secuencia de mensajes con las acciones que se efectúan en una transacción de bus (Capítulo 4) para observar la similitud.

6. Un procesador puede utilizar, como en el caso de un bus (Capítulo 4), la señal de arbitraje para acceder al directorio, para gestionar el inicio del proceso de reinterpretación de instrucciones cuando sea necesario.

7. En un capítulo posterior analizaremos el caso de que todos los procesadores puedan tener una petición en curso. Además, el CM procesa peticiones concurrentemente o de forma paralela. En estas condiciones, un CC puede recibir una petición del CM y tener pendiente una petición al mismo bloque. De forma similar, en función del protocolo de coherencia, el CM al recibir una petición de un CC puede estar procesando una petición previa de otro CC al mismo bloque.

En la Figura 6.5 se muestra la secuencia de mensajes punto a punto en un protocolo de directorio VI. Se distinguen dos casos en función de si el CM, además de la respuesta al CC que efectúa la petición, realiza peticiones a otros CC. En cualquiera de los dos casos, una transacción es una secuencia de dos pasos: 1) petición del CC al CM y 2) respuesta del CM a este CC y en su caso, peticiones del CM a otros CC.

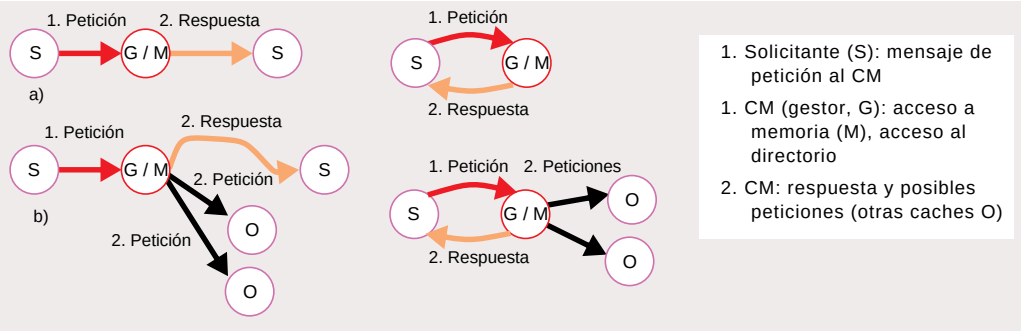


Figura 6.5 Flujo de mensajes en un protocolo de directorio VI entre el CC y el CM y el CM y otros CC. Izquierda: representación lineal. Derecha: representación no lineal (los nodos no se replican).

En una transacción de lectura se utiliza la secuencia de mensajes a) de la Figura 6.5. Cuando no hay copias del bloque en otras caches, en una transacción de escritura se utiliza la secuencia de mensajes a). En caso contrario se utiliza la secuencia de mensajes b)

Organización del multiprocesador

En la Figura 6.6 se muestra la organización del multiprocesador. En el camino de las caches a memoria distinguimos una cola de peticiones (CP). En esta cola se almacenan las peticiones de los CC, los cuales utilizan caminos independientes punto a punto para transmitir los mensajes hacia la CP. La salida de esta cola es un camino a memoria. El CM procesa en serie los mensajes almacenados en la CP[8].

De forma funcional, desde el CM a los CC se distinguen dos caminos independientes a cada uno de ellos (no se muestran en el dibujo). Uno de ellos, denominado de respuestas, lo utiliza el CM para responder a una petición de un CC. El otro camino, denominado de peticiones, lo utiliza el CM para efectuar peticiones de coherencia a uno o varios CC al procesar una petición de otro CC.

8. Mientras se suponga que sólo existe un acceso a memoria en el multiprocesador la CP no tiene utilidad.

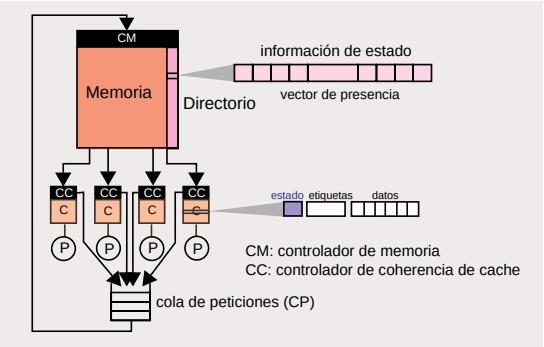


Figura 6.6 Detalle de las redes de interconexión.

En resumen, hay una red desde los CC al CM, a la que denominaremos red de ida (RI). De forma funcional, hay dos redes desde el CM a los CC, una de ellas transporta mensajes de respuesta y la otra transporta mensajes de petición. En un protocolo de directorio VI estas dos últimas redes se pueden implementar, de forma sencilla, mediante una única red física, a la que denominaremos red de vuelta (RV, Figura 6.9). Posteriormente, al detallar el protocolo se observa que, al procesar una petición, el CM puede emitir un mensaje de respuesta y varias peticiones y la intersección de los destinos es el conjunto vacío. Esto es, el CC que recibe el mensaje de respuesta no recibe un mensaje de petición.

A partir de la información ubicada en el directorio, el CM sólo envía peticiones de coherencia, para responder a la petición que está procesando, a los CC que tengan copia del bloque y necesiten efectuar alguna acción.

En los CC de las caches hay dos agentes: a) procesador y b) observador. Los seguimos identificando de la misma forma que en un protocolo de difusión. El agente observador procesa (observa) peticiones del CM.

En un protocolo VI la memoria siempre está actualizada. Por tanto, en una petición de lectura, efectuada por un CC, siempre se suministra el bloque que está almacenado en memoria. Además, con el objetivo de identificar a los nodos que tienen copia del bloque, es suficiente que el directorio tenga un VP por bloque.

Notemos que en el protocolo de directorio VI no se utiliza la fase de respuesta de los CC identificada en la Figura 6.3.

Camino de datos en el controlador de coherencia

En la Figura 6.7 se muestra un esquema del camino de datos de un CC. Se identifican los agentes procesador y observador. El agente observador utiliza una copia del campo etiqueta para eliminar riesgos estructurales con accesos del procesador al campo etiquetas. Por la misma razón, el campo estado tiene dos caminos de acceso, uno para cada agente.

El agente procesador atiende las peticiones del procesador y las respuestas del CM, que llegan por la RV. Cuando el agente procesador no puede servir una petición del procesador emite una petición al CM utilizando la RI. Al recibir la respuesta a una petición, inducida por un fallo de lectura de cache, actualiza la cache, el estado y el duplicado de etiquetas. Al recibir una respuesta a una petición de escritura actualiza la cache con el dato (no se muestra en la Figura 6.7), si el bloque está almacenado en cache.

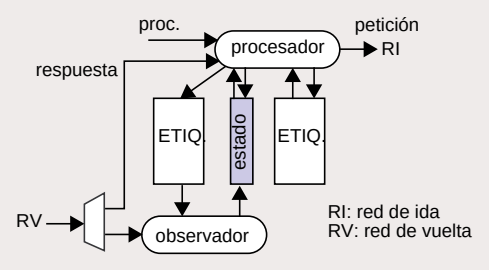


Figura 6.7 Esquema del camino de datos en el CC.

El agente observador procesa las peticiones del CM. Para ello, accede al duplicado de etiquetas con el objetivo de conocer el contenedor de cache, en el cual se almacena el bloque referenciado en la petición. La única petición del CM es observación de escritura, la cual requiere invalidar el bloque. Por tanto, es suficiente que el agente observador actualice el campo estado.

Incluido en el agente procesador está el autómata de reemplazo. Su función es seleccionar un bloque, almacenado en cache, para liberar un contenedor, en el cual se ubicará el bloque accedido en una lectura, que ha detectado un fallo al acceder a la cache. Cuando se conoce el bloque que se expulsa, el autómata procesador emite una petición de expulsión al CM. Esta petición es para mantener actualizada la información en el directorio.

En un acceso a memoria que es fallo en cache y que determina la expulsión de un bloque, en primer lugar se realizan las acciones encaminadas a expulsar el bloque. Posteriormente se inician las acciones para servir el fallo de cache.

Camino de datos en el controlador de memoria

En la Figura 6.8 se muestra un esquema del camino de datos del CM. El CM procesa una petición accediendo al directorio y a memoria y emite una respuesta, y en su caso peticiones a los CC, utilizando conexiones punto a punto dedicadas. La red utilizada para ello es la RV. Recordemos que el destinatario de la respuesta no es destinatario de una petición. Además, el CM actualiza el directorio.

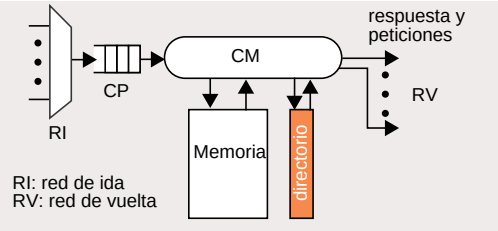


Figura 6.8 Esquema del camino de datos en el CM.

Mensajes del protocolo

Para mantener la coherencia se transmiten mensajes de los CC al CM, que son peticiones. Desde el CM se transmiten mensajes de petición y respuesta a los CC.

En la Figura 6.9 se muestra un esquema genérico de la transmisión de mensajes de peticiones y respuestas entre los CC y el CM. Notemos que el esquema se ha linealizado para que todos los mensajes fluyan de izquierda a derecha. Como los CC emiten peticiones y reciben peticiones y respuestas, en el esquema se dibujan más de una vez.

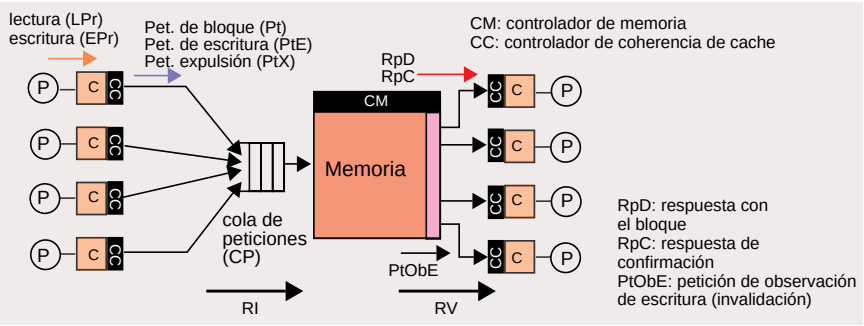


Figura 6.9 Esquema de la transmisión de mensajes en un protocolo de directorio VI.

En las siguientes tablas, al describir los mensajes de petición y respuesta, se indica la información básica que se transmite en el mensaje. El acrónimo “op” indica operación, el acrónimo “Id” indica identificador de la petición. En la respuesta se utiliza el mismo identificador.

En la Tabla 6.1 se describen las peticiones del procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). En un fallo de lectura y en una escritura se accede a memoria.

Procesador	Controlador de coherencia (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato op | dirección	dato dato	Si es un acierto en cache se lee el dato de cache. En caso contrario, se asigna un contenedor para almacenar el bloque y se envía un mensaje de petición de lectura del bloque (Pt) al CM. Finalmente se suministra el dato
EPr (store): escritura de un dato op | dirección | dato	confirmación de la escritura confirmación	En cualquier caso se envía un mensaje de actualización de memoria (PtE). Si es un acierto y el estado del bloque es válido, se escribe el dato en cache al recibir la respuesta del CM. No se asigna un contenedor en caso de fallo.

Tabla 6.1 Peticiones del procesador, respuestas y acciones del CC.

En la Tabla 6.2 se describen las peticiones de un CC al CM y las acciones del CC para gestionar la ubicación de bloques en la cache (conflictos, CcRe). Se distinguen los mensajes de petición de bloque (Pt) y petición de escritura de dato (PtE). También se describe la acción de expulsión de un bloque debido a un conflicto (PtX).

Controlador de coherencia (CC)	Controlador de memoria (CM)	Comentario
Mensajes de petición al CM y acciones	**Mensajes de respuesta**	
Pt: petición de bloque op | dirección | Id	RpD: respuesta con el bloque Id | bloque	Se lee el bloque de datos de memoria, se actualiza el directorio y se suministra el bloque.
PtE: petición de escritura de un dato op | dirección | dato | Id	RpC: respuesta de confirmación Id | confirmación	Se actualiza memoria con el dato, el CM envía un mensaje de respuesta y si es el caso, peticiones de observación de escritura a otros CC. También se actualiza el directorio.
CcRe: expulsión de un bloque.		Se invalida la información del contenedor.
PtX: petición (notificación) de expulsión op | dirección | Id	RpX: respuesta de confirmación Id | confirmación	El CM, después de actualizar el vector de presencia, emite una respuesta de confirmación.

Tabla 6.2 Peticiones de un CC al CM, respuestas del CM.

En la Tabla 6.3 se describen las peticiones del CM a los CC y la acción del CM para tener actualizada la información de presencia de los bloques. El CM envía mensajes de petición de observación de escritura, si es el caso, cuando procesa una petición de escritura.

Controlador de memoria	Controlador de coherencia (CC)	Comentario
Mensajes de petición a CC y acciones	**Respuesta**	
PtObE: petición de observación de escritura op | dirección	La red de transmisión de mensajes del CM a los CC mantiene el orden. Por tanto, no es necesario la respuesta	Petición para que un CC concreto invalide un bloque.
Actualización del directorio		Mantiene actualizada la información de compartición de los bloques.

Tabla 6.3 Petición del CM a los CC.

Estados y transiciones

En un protocolo de directorio hay que describir los estados de un bloque en las caches y los estados en el directorio.

Directorio. Los estados de un bloque en el directorio son: no presente (NP) y presente (Pr). El primero indica que no existe copia del bloque en las caches, mientras que el segundo indica que existe al menos una copia. Estos estados están codificados en el VP asociado al bloque. Si todos los bits del VP están desactivados el estado es NP. En caso contrario el estado es Pr.

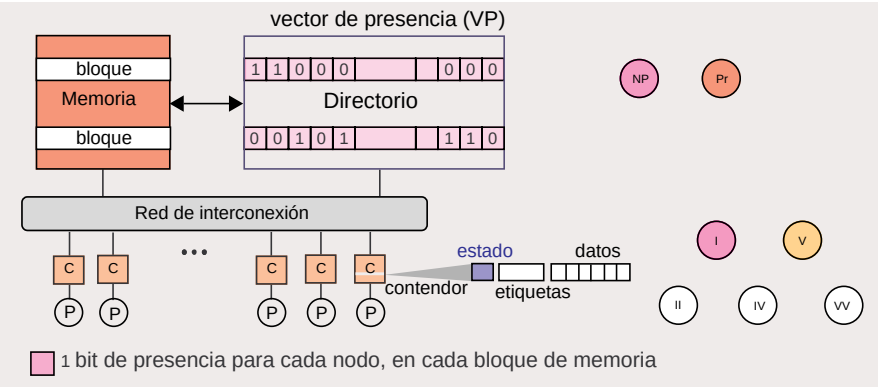


Figura 6.10 Protocolo de directorio VI. Estados de un bloque en el directorio y en un contenedor de caches.

Cache. En una cache, un bloque puede tener dos estados estables y tres estados transitorios. Los estados estables son: Inválido (I) y Válido (V). Los estados transitorios son: II, IV y VV. En este apartado, los estados transitorios se utilizan para distinguir entre la emisión de la petición (mensaje de petición)

y la recepción de la respuesta (mensaje de respuesta). El etiquetado de los estados transitorios sigue las reglas utilizadas al describir los protocolos de observación en el Capítulo 4.

Para describir el protocolo se utilizan las dos peticiones del procesador que requieren acceder al directorio, las cuales son fallo de lectura (load, Pt) y cualquier escritura (store, PtE). En un acierto de lectura el estado en la cache no se modifica. Posteriormente se detallan las transiciones entre estados en una expulsión de un bloque (PtX).

Al describir cada una de las peticiones indicadas se mostrarán las transiciones entre estados efectuadas por el CC que efectúa la petición (agente procesador), en el directorio gestionado por el CM y en otros CC (agente observador).

Fallo de lectura

En la Figura 6.11 se muestra el flujo de mensajes en un fallo de lectura. El CC, después de detectar el fallo, emite un mensaje con una petición de bloque, que se almacena en la CP. Entonces, el CM extrae el mensaje de la CP, lee del directorio el VP asociado al bloque, accede a memoria para leer el bloque y actualiza el VP. Finalmente el CM emite un mensaje de respuesta con el bloque al CC que ha efectuado la petición.

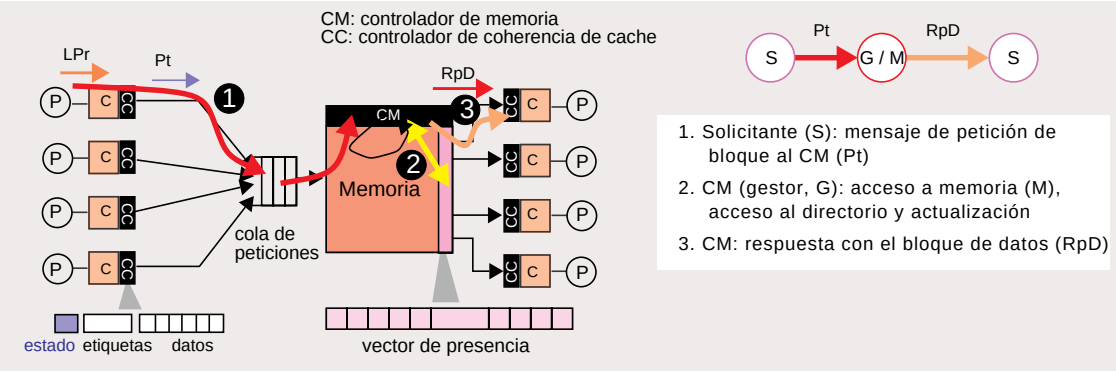


Figura 6.11 Fallo de lectura. Esquema de transmisión de mensajes.

En la Figura 6.12 se muestran las transiciones entre estados de un bloque en un fallo de lectura: a) el CC que efectúa la petición (solicitante), b) el directorio (CM) y c) otros CC. Además, se muestra la actualización del VP, asociado al bloque, que está almacenado en el directorio.

Después de emitir el mensaje de petición de lectura, el CC (agente procesador) establece que el estado del bloque es IV (estado transitorio). El bloque permanece en este estado hasta que se recibe la respuesta del CM, la cual es procesada por el agente procesador del CC, que modifica el estado del bloque a válido.

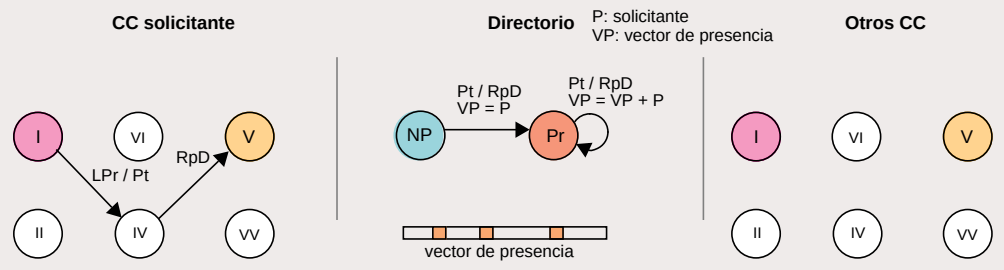


Figura 6.12 Fallo de lectura. Transiciones entre estados.

El CM, al recibir el mensaje de petición de lectura, accede al directorio para leer el VP, a memoria para leer el bloque y después emite la respuesta. Además, actualiza el VP. Al conjunto de bits activados, que indican copia del bloque en otras caches (VP), se añade el bit asociado al procesador solicitante (P). En la Figura 6.12 se indica como VP = VP + P o VP = P, siendo P el procesador cuyo CC efectúa la petición. Los otros CC no reciben ninguna petición de coherencia desde el CM, ya que la petición es de lectura y la memoria, en el protocolo de directorio VI, siempre está actualizada.

Escritura

En la Figura 6.13 se muestra el flujo de mensajes en una escritura. El CC emite un mensaje con una petición de escritura, que se almacena en la CP. Seguidamente, el CM extrae el mensaje de la CP y accede a memoria para actualizar el dato referenciado, lee el VP correspondiente al bloque, para conocer las caches que tienen copia del bloque, y lo actualiza. Finalmente el CM emite un mensaje de respuesta (RpC) al CC que ha efectuado la petición y mensajes de petición de observación de escritura (PtObE) a los CC que tienen almacenada una copia del bloque en su cache, excluyendo, si es el caso, al CC que ha efectuado la petición de escritura.

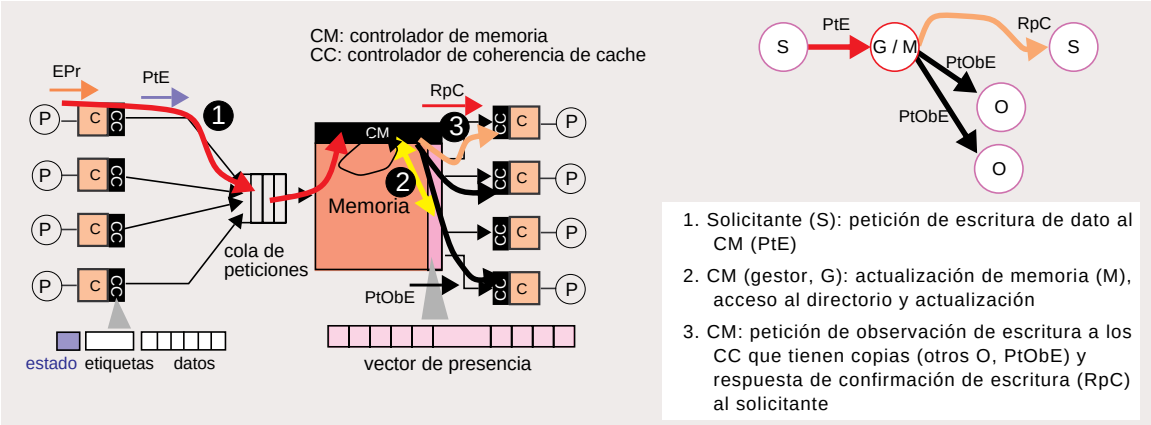


Figura 6.13 Escritura. Esquema de transmisión de mensajes.

Como se utiliza escritura inmediata todas las escritura se propagan a memoria. Al emitir una petición de escritura (PtE) el CC modifica el estado del bloque. El nuevo estado es transitorio y se identifica como II o VV, en función, respectivamente, de si se ha detectado fallo o acierto (Figura 6.14).

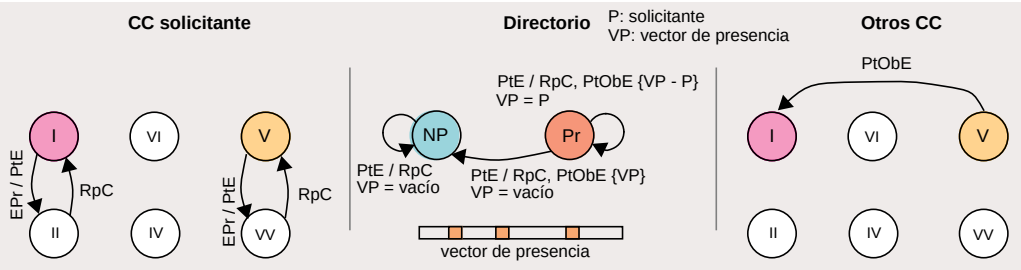


Figura 6.14 Escritura. Transiciones entre estados.

El CM al recibir la petición lee el VP para conocer las caches que tienen copia del bloque. Si es el caso, emite peticiones de observación de escritura a los CC que gestionan las caches, excluyendo al CC que ha efectuado la petición. En la Figura 6.14 las peticiones de observación de escritura se indican como PtObE{VP - P}, siendo P el procesador cuyo CC efectúa la petición. En cualquier caso, el CM emite un mensaje de confirmación al CC que ha efectuado la petición. El estado final del bloque en el directorio es no presente (NP), si el CC que solicita la escritura no tiene copia del bloque y presente (Pr) en caso contrario (en el VP está activado el bit correspondiente al CC). Además, en cualquier caso, el CM actualiza memoria con el dato recibido (no se indica explícitamente en la transición entre estados).

Los CC (agente observador) que reciben una petición de observación de escritura cambian el estado del bloque de válido a inválido.

Expulsión de un bloque de cache

Las expulsiones se notifican al directorio para mantener actualizado el VP. Esta notificación mantiene en el directorio, de forma precisa, las copias existentes del bloque en las caches.

En la Figura 6.15 se muestra el flujo de mensajes en la expulsión de un bloque de una cache. El CC emite un mensaje de petición de expulsión (PtX). El CM procesa el mensaje desactivando el bit correspondiente al CC en el VP y envía un mensaje de confirmación (RpX).

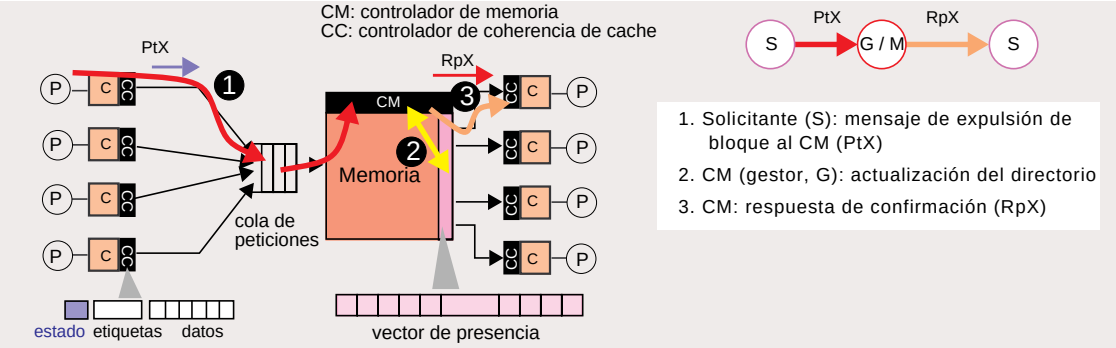


Figura 6.15 Expulsión de un bloque. Esquema de transmisión de mensajes.

En la Figura 6.16 se muestran las transiciones entre estados de un bloque al efectuarse una expulsión en: a) el CC que efectúa la petición (solicitante), b) el directorio y c) otros CC. Además, se muestra la actualización del VP almacenado en el directorio. Un CC después de enviar el mensaje de petición de expulsión (PtX) cambia el estado a VI. El CM procesa el mensaje y cambia el estado a NP o permanece en el estado Pr, en función de si el mensaje provenía del único CC cuya cache almacenaba el bloque o no. Los otros CC no se ven afectados por la actualización del directorio.

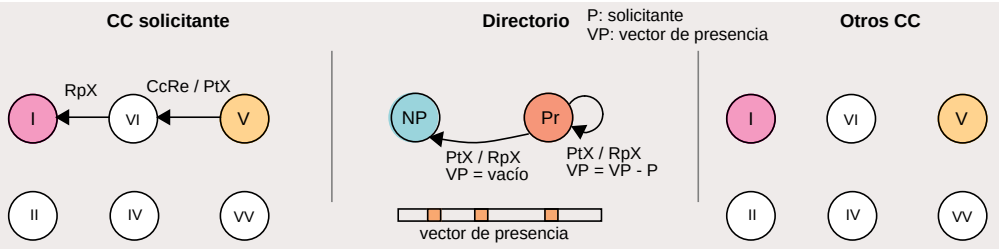


Figura 6.16 Expulsión de un bloque de cache. Transiciones entre estados.

Diagrama completo de estados y transiciones

En las Figura 6.17 y Figura 6.18 se muestran, respectivamente, los diagramas de transiciones entre estados de un bloque en cache y en el directorio. En el CC se distinguen, en diagramas separados, las transiciones iniciadas por el agente procesador y el agente observador (Figura 6.17). En el primer diagrama están incluidas las peticiones del procesador que no requieren iniciar transacciones explícitas de coherencia.

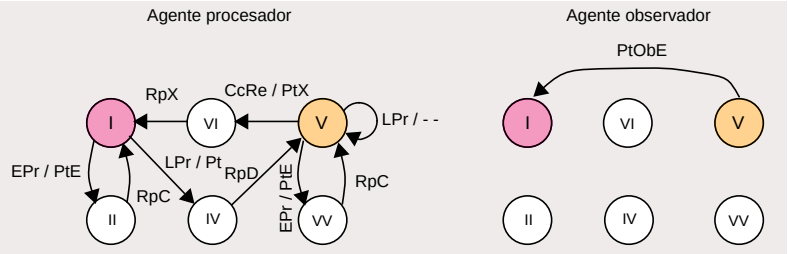


Figura 6.17 Protocolo de directorio VI. Estados y transiciones entre estados de un bloque en una cache.

En el diagrama del CM (Figura 6.18) hay que distinguir dos casos al procesar las peticiones PtE y PtX de un CC. En una petición PtE, el estado final depende de si el CC, que efectúa la petición, está en el VP o no está. En una petición PtX, la condición es si el CC es el único que está en el VP o hay más CC.

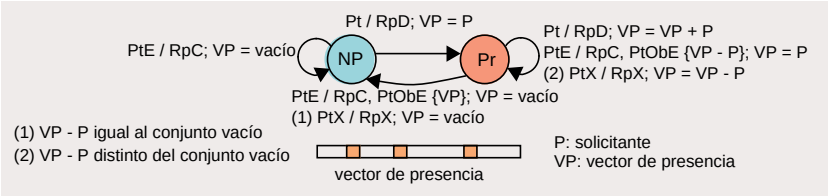


Figura 6.18 Protocolo de directorio VI. Estados y transiciones entre estados de un bloque en el directorio.

Tablas de estados y transiciones

En la Tabla 6.4 se muestran en formato tabla los estados y las transiciones entre estados de un bloque en una cache. Las casillas que no contienen información indican un error. En un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

			Eventos del procesador y reemplazo			Eventos externos (respuestas y petición)			
			LPr	EPr	CcRe	RpD	RpC	RpX	PtObE
Estados	Estables	I	Pt; IV	PtE; II					
		V	- - ; V	PtE; VV	PtX; VI				- - ; I
	transitorios	IV				- -; V			
		VV					- - ; V		
		VI						- - ; I	
		II					- -; I		

Tabla 6.4 Protocolo de directorio VI. Tabla de estados y transiciones de un bloque de cache.

En la Tabla 6.5 se muestran en formato tabla los estados y transiciones entre estados de un bloque en el CM. En el evento PtE se distinguen dos casos, en función de si el CC está o no está presente en el vector de presencia. En el evento PtX se distingue el caso de que el CC sea el único que está en el VP o haya más CC.

			Eventos del controlador de coherencia					
				PtE			PtX	
			Pt	VP = vacío	P ∈ VP	P ∉ VP	VP = P	P ∈ VP VP ≠ P
Estados	Estables	NP	RpD; Pr, VP = P	RpC; NP, VP = vacío				
		Pr	RpD; Pr, VP = VP + P		RpC, PtObE {VP - P}; Pr, VP = P	RpC, PtObE {VP}; NP, VP = vacío	RpX; NP, VP = vacío	RpX; Pr, VP = VP - P

Tabla 6.5 Protocolo de directorio VI. Tabla de estados y transiciones de un bloque en el directorio.

Representación de transacciones y transiciones entre estados

En este apartado se muestran tres formas de representar las transacciones y transiciones al ejecutar una secuencia de accesos a memoria: a) en formato tabla, b) mediante un diagrama temporal y c) mediante un diagrama temporal simplificado. Finalmente se muestra, mediante varios gráficos, una animación.

Representación en formato tabla

Cada fila representa un acceso a memoria y no se gestiona el siguiente acceso hasta que finaliza el anterior. En una fila de izquierda a derecha, después de la instrucción de acceso a memoria, se especifica:

1 El estado transitorio del bloque en la cache del procesador cuyo acceso a memoria requiere una transacción.

2 La petición de acceso a memoria en la red que interconecta los CC y el CM (red de ida, RI).

3 El nombre de la variable accedida y el vector de presencia (VP) en la memoria. La posición más a la izquierda representa la cache con el menor ordinal y sucesivas posiciones, contiguas en el VP, representan caches con ordinales crecientes y consecutivos.

4 Quién suministra el dato o bloque (cache o memoria).

5 La respuesta del CM en la red que interconecta el CM y los CC (red de vuelta, RV). En la siguiente fila, y la misma columna, se especifican las peticiones del CM, si es el caso, a los CC. En este caso, también se enumeran los CC que reciben la petición.

6 Para las caches donde se modifica la información almacenada, el nombre de la variable y el estado estable del bloque al finalizar el acceso a memoria.

Ejemplo. En la Tabla 6.6 se muestra una secuencia de accesos a memoria realiza por tres procesadores. En este ejemplo suponemos que las variables t y u no están almacenadas en cache y los bloques que las contienen se almacenan en contenedores distintos de cache.

	C 1	C 2	C 3	Red	mem.			Red	C 1		C 2		C 3	
acceso	est.	est.	est.	RI	var.	VP	sum.	RV	var.	est.	var.	est.	var.	est.
1. P1 load t	IV			Pt	t	1, 0, 0	mem.	RpD	t	V				
2. P2 store u		II		PtE	u	0, 0, 0	C2	RpC			u	I		
3. P1 load u	IV			Pt	u	1, 0, 0	mem.	RpD	u	V				
4. P3 load u			IV	Pt	u	1, 0, 1	mem.	RpD					u	V
5. P1 store u	VV			PtE	u	1, 0, 0	C1	RpC	u	V				
								PtObE 3					u	I

Tabla 6.6 Protocolo de directorio VI. Formato tabla. Secuencia de accesos a memoria.

El primer acceso a memoria es una instrucción load y se produce un fallo en la cache C1. Al emitir el mensaje de petición de lectura de bloque, el estado transitorio del bloque es IV. La variable accedida es t y el VP en el directorio sólo indica la presencia del bloque en la cache C1. La RV transporta un mensaje de respuesta de datos y el estado estable del bloque en la cache C1 al finalizar el servicio del fallo es V.

El segundo acceso a memoria es una instrucción store. El bloque no está almacenado en cache. Por tanto, el estado transitorio es II y el mensaje que se transmite por la red RI es PtE. No hay copia del bloque referenciado en ninguna cache. El dato se transmite del CC a memoria. La respuesta que se transmite por la red RV es una confirmación de escritura (RpC) y el estado del bloque en la cache, al recibir la respuesta, es I.

El tercer acceso a memoria determina que el CC1 emita una petición Pt. La variable accedida es u. El comportamiento es mimético al primer acceso a memoria del mismo procesador. El cuarto acceso a memoria, efectuado por el procesador P3, es mimético al anterior con la salvedad de que el VP indica copias del bloque en las cache C1 y C3.

El quinto acceso a memoria es una instrucción store en el procesador P1. Al acceder a cache se determina que es acierto. Por tanto, el estado transitorio es VV. El mensaje que transmite el CC al CM es PtE. En el directorio el VP se actualiza para indicar que hay copia del bloque sólo en la cache C1. Las otras copias, que había del bloque en otras caches, se invalidan mediante la emisión, por parte del CM, de un mensaje PtObE a los respectivos CC. En la tabla se representa este hecho utilizando la siguiente fila e identificando la petición del CM y el ordinal de los procesadores en la columna de la RV. El estado estable del bloque en la cache C1 es V y en la cache C3 es I.

Diagrama temporal

En la Figura 6.19 se muestran varios ejemplos de diagramas temporales. Aunque se asocian fases de una acción de coherencia (transacción) a ciclos estos no son representativos.

En un diagrama temporal, los mensajes correspondientes a una transacción se representan en la fila asociada al acceso a memoria. Antes de utilizar la RI para transmitir el mensaje se representa el arbitraje. Esta fase se utiliza para mostrar la ordenación de los mensajes. Tengamos en cuenta que, en un instante determinado, sólo se puede encolar un mensaje en la CP. La espera para acceder a la red se indica representando en ciclos consecutivos la fase de arbitraje (arb)[9].

Después de utilizar la RI se representa el acceso a memoria y al directorio (M). Seguidamente se representa la utilización de la RV y finalmente se representa la recepción de los mensajes de respuesta y de petición en los CC.

9. Cuando en el modelo de multiprocesador sólo existe una petición en curso esta fase no es necesaria. Sin embargo, para utilizar la misma representación en este caso y en un próximo capítulo, donde se analizan peticiones concurrente, la mantendremos.

Cuando el CM emite una respuesta y peticiones se utilizan dos filas. En la primera fila se indica la respuesta y en la segunda fila la secuencia de ordinales de los CC que reciben la petición (PtObE). Notemos que antes de utilizar la RV también se representa una fase de arbitraje[10].

En la parte central de un diagrama temporal se muestran, en las distintas fases de una transacción, los estados de un bloque en las caches y en el directorio. En la fase de arbitraje de la RI (arb), que se corresponde con el arbitraje para enviar un mensaje de un CC al CM, se indica el estado transitorio del bloque en el CC al emitirse la petición.

En la fase correspondiente al acceso al directorio (M) se indica la actualización del VP, si es el caso. El VP se representa utilizando los ordinales de los procesadores o CC cuyos bits están activados (fila M).

En la fase de respuesta (D, C, X) y de recepción de peticiones del CM en un CC se indica el estado estable de los bloques en las caches. En fases donde no se indica explícitamente el estado, éste es el último que se ha indicado explícitamente en la misma fila. Los acrónimos D, C y X indican respectivamente bloque, confirmación de escritura y respuesta a una petición de expulsión.

En la parte inferior del diagrama temporal se muestra el acrónimo de los mensajes emitidos por un CC o por el CM. El mensaje se indica en la fase arb.

En las fases M y recepción de respuesta se utilizan líneas continuas finalizadas con una flecha para indicar la relación entre la fase y la actualización del estado en el directorio y en otros CC respectivamente (peticiones PtObE).

En el ejemplo de la izquierda de la Figura 6.19 se muestra un fallo de lectura. Antes de ejecutar la secuencia de accesos hay copia del bloque en la cache C2. En la fase arb se indica el estado transitorio en la cache del CC que envía el mensaje. En la fase M se indica que en el VP están activados los bits correspondientes a las caches C1 y C2. En la fase de respuesta y peticiones del CM se indica que el estado del bloque en la cache C1 es V.

En el ejemplo del centro de la Figura 6.19 se muestra la ejecución de una instrucción store cuando se produce acierto en cache. Suponemos que no hay copias del bloque en otras caches. El estado transitorio del bloque, al emitir el CC la petición de escritura, es VV y el estado final es V. El contenido del VP del bloque en el directorio no se modifica.

10. Esta fase no es de utilidad cuando sólo existe un acceso memoria en el multiprocesador. Sin embargo, para utilizar la misma representación que cuando se utiliza un multiprocesador con varios módulos de memoria, la cual se efectúa en un próximo capítulo, la mantendremos.

En el ejemplo de la derecha de la Figura 6.19 se muestra la ejecución de una instrucción store, cuando hay copias del bloque en otras cache y no hay copia en la cache cuyo procesador ejecuta la instrucción. Para representar, en el mismo ciclo, la emisión de mensajes de respuesta y de petición desde el CM se utilizan dos filas. En la segunda fila se explicita la secuencia de ordinales de los CC que reciben la petición PtObE. Notemos que el VP se actualiza en la fase M y el estado de las copias del bloque en la fase de recepción de respuesta y petición.

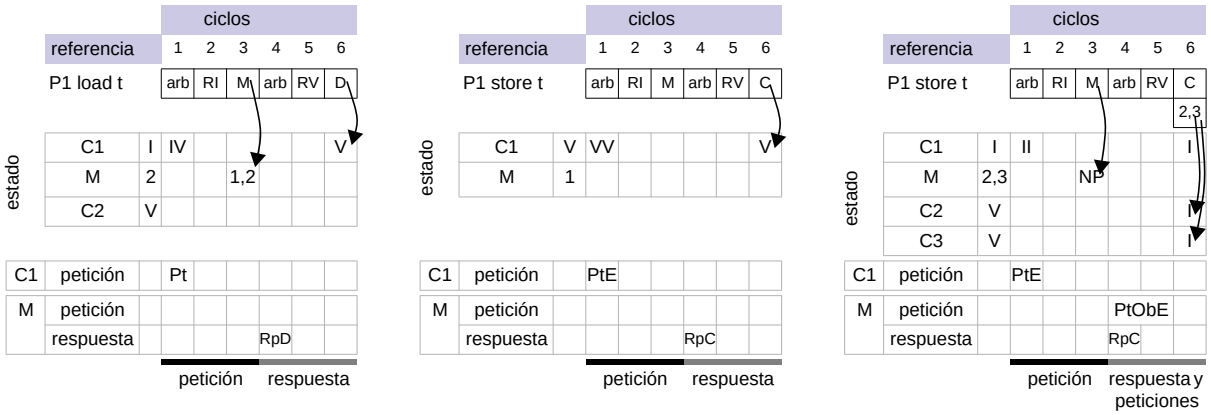


Figura 6.19 Protocolo de directorio VI: diagramas temporales de acciones de coherencia.

Ejemplo. En la Figura 6.20 se muestra el diagrama temporal de la secuencia de accesos mostrada en la Tabla 6.6. En la parte central de la Figura 6.20 se especifican las caches de los procesadores, el directorio (M) y las variables. En estas filas se representa el estado, en cada cache y en el directorio, de la forma descrita en los ejemplos de la Figura 6.19. En la parte inferior se representan las peticiones de los CC y las peticiones y respuestas del CM.

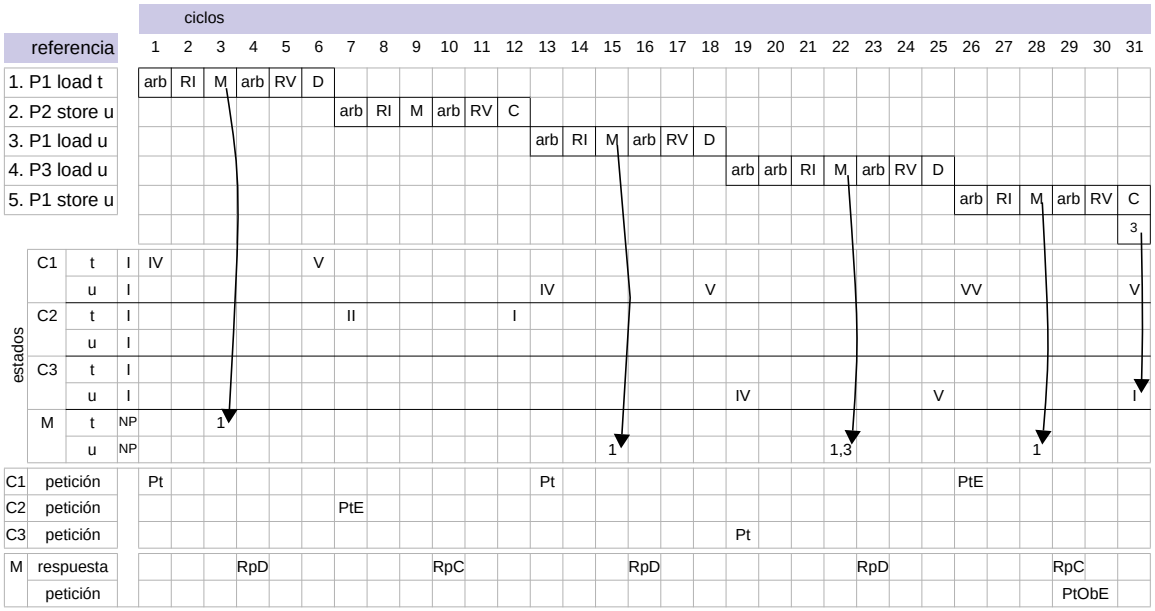


Figura 6.20 Protocolo de directorio VI. Diagrama temporal de una secuencia de accesos a memoria.

Diagrama temporal simplificado

La idea de este diagrama es representar de forma más esquemática los mensajes y la temporalidad de los mismos[11].

Cada CC y el CM se representan por una línea vertical[12]. El tiempo avanza de arriba hacia abajo.

Un trazo continuo entre un CC (fuente) y el CM (destino), finalizado con una flecha al llegar al CM, indica un mensaje del CC al CM. Además, el origen del trazo es temporalmente previo a la finalización del mismo. Un mensaje del CM a un CC se representa de forma similar.

El acrónimo de los mensajes de petición de un CC se representan en el CC emisor. El acrónimo de las peticiones y respuestas del CM se representan en el CC receptor. El estado se representa, en los CC y en el CM, al emitir una petición, recibir una petición o recibir una respuesta. Los estados en el CM se representan de la misma forma que en un diagrama temporal.

11. Esta representación permite visualizar en menos espacio más mensajes.
12. En ocasiones puede girarse la representación 90 grados.

Como sólo hay una transacción en curso no se pueden solapar temporalmente transacciones.

Cuando un CM al procesar una petición emita varios mensajes supondremos que se emiten simultáneamente y llegan simultáneamente a los destinos.

En la Figura 6.21 se muestra un diagrama temporal donde se identifican los mensajes entre los CC y el CM y viceversa para la secuencia de accesos mostrada en la Tabla 6.6. También se indica el estado de los bloques en el directorio y en los CC.

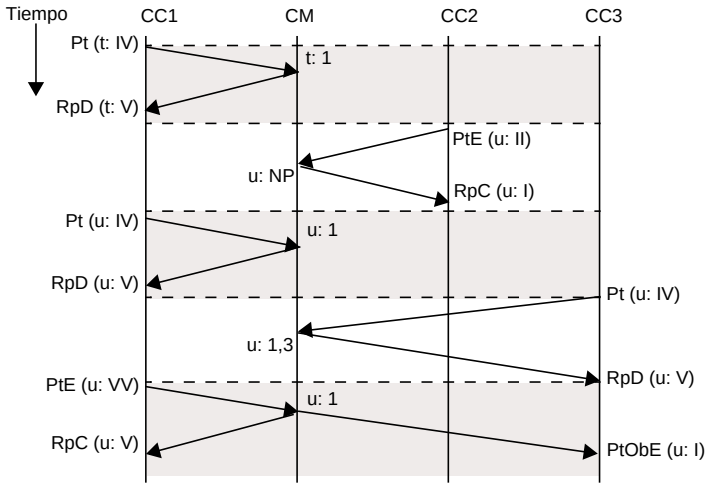


Figura 6.21 Protocolo de directorio VI. Diagrama temporal simplificado de una secuencia de accesos a memoria concurrentes.

Animación

En la Figura 6.22 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 6.6. Además de las peticiones y respuestas, se muestran los cambios de estado utilizando diagramas de transición entre estados. Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido. En el directorio, el VP del bloque, que contiene una variable, se identifica con el nombre de la variable seguido de los ordinales de las caches que tienen copia del bloque.

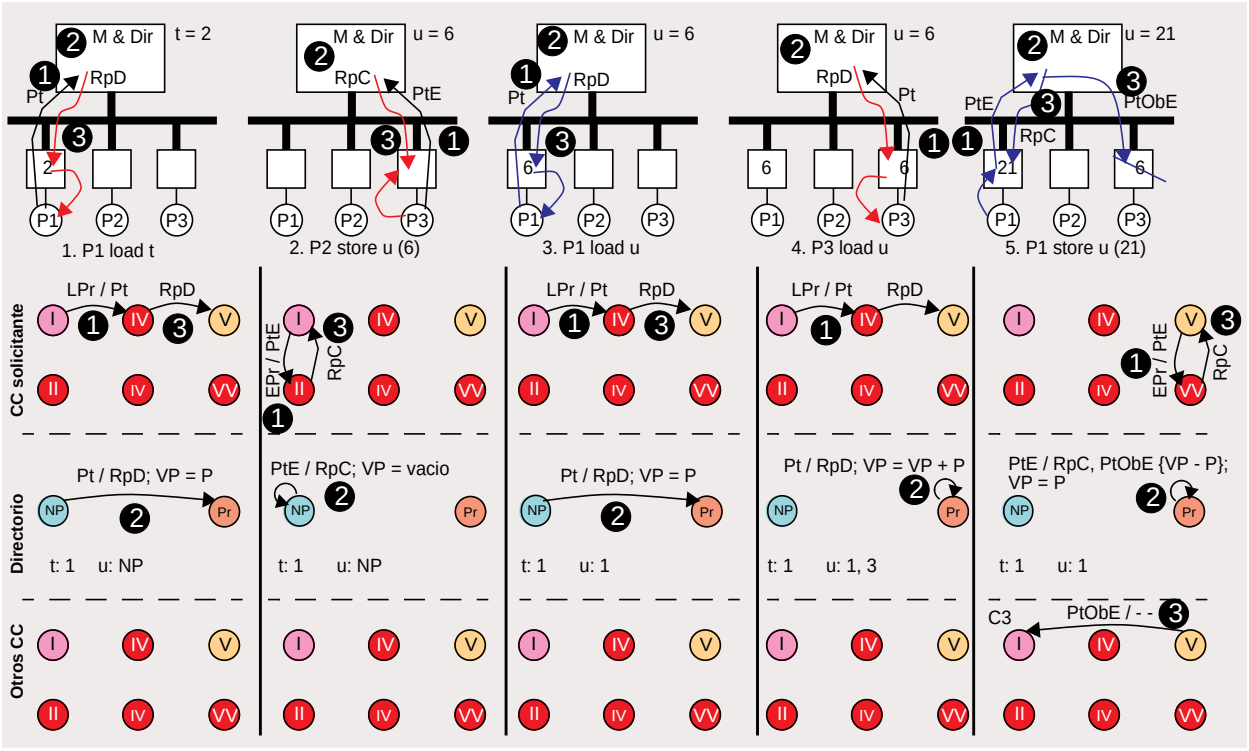


Figura 6.22 Protocolo de directorio VI. Secuencia de accesos a memoria y fotogramas de las peticiones y respuestas y cambios de estado.

Verificación no formal de coherencia y consistencia

En este capítulo el CM actúa de forma similar al bus en el Capítulo 4. En el multiprocesador existe un CM centralizado. Para acceder al CM se utiliza un árbitro. El CM no empieza a procesar una petición a un bloque hasta que ha finalizado el procesado de la petición previa, ya sea al mismo o distinto bloque.

La red de comunicación del CM a los CC mantiene el orden de emisión de los mensajes desde el CM. Por tanto, el CM establece un orden global entre las transacciones que gestiona.

Coherencia de cache

Orden de programa en accesos a la misma posición de memoria. El procesador efectúa los accesos en el orden determinado por el L.M. El procesador se bloquea en un fallo de lectura o en una escritura, hasta que finaliza la transacción correspondiente[13].

Propagación de escrituras. Al utilizar escritura inmediata todas las escrituras se emiten al directorio. Un CC emite una petición de escritura al CM y éste, utilizando la información disponible en el directorio, la propaga a los CC que tienen copia del bloque. Las peticiones de invalidación (PtObE) se transmiten concurrentemente con la respuesta al CC solicitante (RpDC). La red del CM a los CC mantiene el orden de emisión de los mensajes. Entonces, no se espera respuesta de las peticiones de invalidación (PtObE). La respuesta está implícita en la emisión del mensaje.

Serialización de escrituras (atomicidad de una escritura).

- Punto de serialización: El CM es el punto de serialización de las escrituras a la misma posición de memoria[14]. El arbitraje para acceder al CM determina el orden de serialización de las transacciones de escritura a una posición de memoria. El CM procesa las peticiones en serie. Sólo hay una transacción en curso.
- Consolidación de una escritura: Una escritura está consolidada al finalizar la transacción. El CM finaliza la transacción al emitir el mensaje de respuesta al CC que ha efectuado la petición y las peticiones de invalidación a los CC que tiene copia. Por otro lado, una escritura es atómica, ya que, una vez finalizada la transacción correspondiente, una instrucción load posterior lee el valor establecido por la escritura previa, en el orden de procesado en el CM[15].

Consistencia secuencial de memoria

Orden de programa. El procesador efectúa los accesos a cualquier posición de memoria en el orden determinado por el L.M.

- Lectura: Espera hasta que se obtiene el dato.
- Escritura: El CM es el punto de ordenación de todas las escrituras. Todas las escrituras requieren emitir una petición al CM. La recepción de la respuesta de confirmación del CM es una indicación de que la escritura

13. Hay que garantizar las dependencias de datos en el hilo que se ejecuta.

14. Aún más, es el punto de serialización de las escrituras y los fallos de lectura a cualquier posición de memoria, ya que se utiliza un único CM.

15. Las copias del bloque son invalidadas en cada transacción de escritura. Un fallo de lectura lee el valor establecido por la escritura previa en el orden de procesado en el CM. Un acierto de lectura lee el valor obtenido en el fallo de lectura previo por la escritura previa, que acierta en cache, del mismo procesador.

está consolidada. Durante el procesado de la transacción, el CM emite peticiones de invalidación a las caches que tienen copia del bloque. Debido a la propiedad de la RV, que conecta el CM a los CC, en la emisión de la petición se considera implícita la respuesta a la misma.

Atomicidad de las escrituras. Una escritura siempre requiere emitir una petición al CM.

- Consolidación de una escritura: La finalización de la transacción (recepción de la respuesta del CM) es una indicación de que la escritura está consolidada.
- Suministro del valor en una lectura: Un fallo de lectura requiere emitir una petición al CM. El valor devuelto en la transacción es el valor establecido en la última escritura consolidada, a la misma posición de memoria. Un acierto de lectura lee el valor de la copia en cache. Este valor ha sido establecido en el fallo de lectura previo o en la escritura previa, que acierta en cache, del mismo procesador.

PROTOCOLO DE DIRECTORIO CON INVALIDACION Y ESCRITURA RETARDADA

En este apartado se describe en primer lugar la secuencia de mensajes utilizada en una transacción de coherencia. Posteriormente se describe la organización del multiprocesador y los agentes de coherencia en las caches y memoria. Seguidamente se detallan los tipos de mensajes utilizados y finalmente se describen los estados y transiciones entre estados.

A este protocolo lo denominaremos protocolo de directorio MLI.

Descripción funcional de la secuencia de mensajes en un transacción

La descripción funcional de un protocolo MLI se ha efectuado en el Capítulo 4. En este apartado nos centraremos en la secuencia de mensajes de una transacción en un protocolo de directorio. Es interesante comparar la secuencia de mensajes con las acciones que se efectúan en una transacción de bus para observar la similitud (Capítulo 4).

En la Figura 6.23 se muestra la secuencia de mensajes punto a punto en un protocolo de directorio MLI. El caso a) representa una transacción de lectura y la memoria tiene el bloque actualizado. El caso b) representa una transacción

de escritura, la memoria tiene el bloque actualizado y hay copias del bloque en otras caches. El caso c) representa una transacción de lectura o escritura y una cache tiene el bloque en exclusividad.

Se distinguen dos grupos en función del número de pasos[16] serie en una transacción. En un grupo (a y b) la secuencia es de dos pasos y en el otro grupo (c) es de cuatro pasos y requiere la colaboración de un CC distinto del que hace la petición.

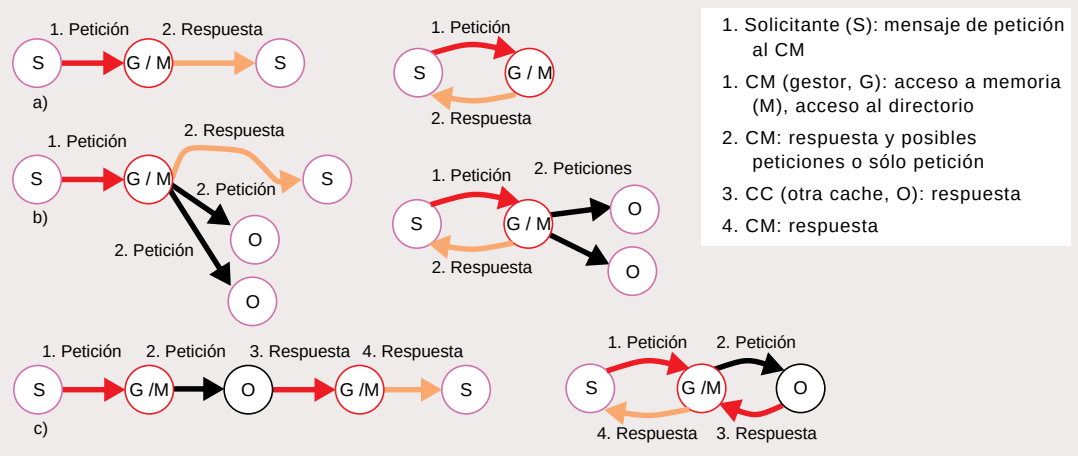


Figura 6.23 Flujo de mensajes en un protocolo de directorio MLI entre el CC y el CM y entre el CM y otros CC. Izquierda: representación lineal. Derecha: representación no lineal (los nodos no se replican).

Cuando la secuencia es de dos pasos se distinguen dos casos en función de si el CM, además de la respuesta, también emite peticiones a algún CC que no es el receptor de la respuesta. En cualquiera de los dos casos los dos pasos de una transacción son: 1) petición de un CC y 2) respuesta y en su caso, peticiones del CM.

En una transacción con una secuencia de cuatro pasos distinguimos: 1) petición de un CC, 2) petición del CM a un CC distinto del que ha efectuado la petición, 3) respuesta de este CC al CM y 4) respuesta del CM al CC que ha efectuado la petición.

Organización del multiprocesador

En la Figura 6.24 se muestra un esquema linealizado del multiprocesador y se muestra una transacción de 4 pasos. En el camino de las caches a memoria distinguimos una CP donde se almacenan las peticiones de los CC. Cada CC

16. Por pasos se entiende el número de mensajes en secuencia. En un paso puede haber varios mensajes concurrentes.

tiene un camino independiente, punto a punto, para transmitir los mensajes a la CP. A esta red la denominamos red de ida (RI). El CM procesa en serie los mensajes almacenados en la CP[17].

Del CM a las caches hay, para cada una de ellas, un camino punto a punto que transporta peticiones de coherencia emitidas por el CM. El CM al procesar una petición, utilizando la información disponible en el directorio, sólo envía peticiones de coherencia a los CC que tiene copia del bloque y que además, es necesario que efectúen una acción para mantener la coherencia.

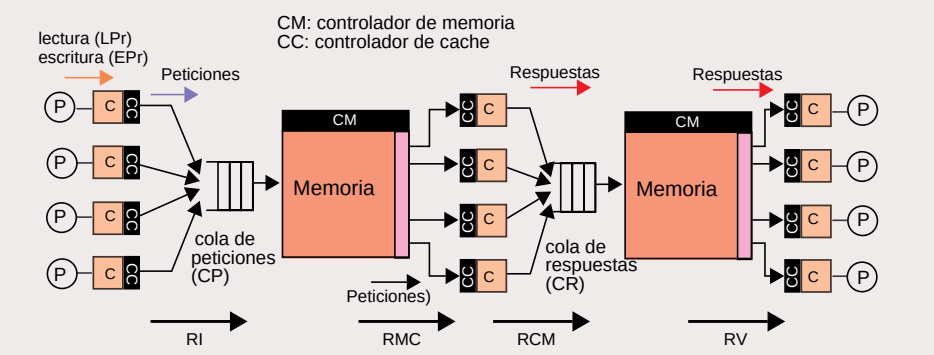


Figura 6.24 Detalle de las redes de interconexión.

En la Figura 6.24, se distingue una cola de respuestas (CR). Los CC almacenan en esta cola la respuesta a la petición de coherencia del CM. Recordemos que, en un protocolo MLI, en ocasiones es necesario que una cache suministre el bloque[18].

Cada respuesta almacenada en la CR es procesada por el CM para finalizar la transacción pendiente en el directorio, la cual ha inducido la necesidad de esta respuesta.

El CM procesa la respuesta y finaliza la transacción, enviando un mensaje de respuesta al CC que inició la transacción. La transmisión de las respuestas se efectúa utilizando conexiones punto a punto entre el CM y los CC.

En resumen, en el multiprocesador distinguimos de forma lógica dos redes desde los CC al CM. Una de ellas es para transmitir peticiones (red de ida, RI) y la otra para transmitir respuestas a una petición del CM (red RCM).

17. Mientras se suponga que sólo existe un acceso a memoria en el multiprocesador la cola de peticiones no tiene utilidad.

18. Mientras se suponga que sólo existe un acceso a memoria en el multiprocesador la cola de respuestas no tiene utilidad.

En el camino del CM a los CC también se distinguen dos redes lógicas. Una de ellas se utiliza para transmitir peticiones del CM a los CC, como paso necesario en el procesado de una transacción (red RMC). La otra red se utiliza para transmitir el mensaje de respuesta, correspondiente a la finalización de una transacción (red de vuelta, RV).

Las redes lógicas RV y RMC se pueden implementar mediante una única red (Figura 6.25). La razón es que cuando el CM requiere emitir en paralelo una respuesta a un CC y varias peticiones a otros CC, para efectuar una acción de coherencia, el destinatario de la respuesta no es destinatario de una petición (transacción de dos pasos, Figura 6.23). Entonces, en este caso sólo indicaremos que se utiliza la red de vuelta. Sin embargo, notemos que las dos redes lógicas se utilizan en paralelo.

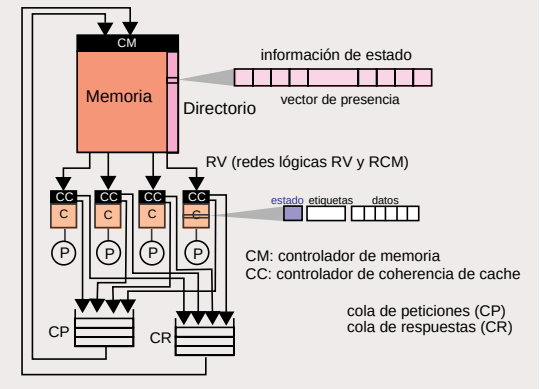


Figura 6.25 Redes y colas de peticiones y respuestas.

Otro caso donde el CM efectúa una petición a un CC es en una transacción de cuatro pasos (Figura 6.23). En este caso la petición a un CC es previa al envio de la respuesta del CM al CC que ha iniciado la transacción. Entonces, para identificar la petición del CM, de forma explícita, indicaremos la red RMC cuando el CM envíe la petición y la red RV cuando el CM envíe la respuesta.

Camino de datos en el controlador de coherencia

En las caches distinguimos dos agentes (Figura 6.26): a) agente procesador y b) agente observador. El primero sirve las peticiones de acceso a la cache del procesador y procesa la respuesta del CM a la petición efectuada previamente por el CC. El segundo se encarga de: a) atender las peticiones del CM al CC y b) emitir una respuesta a una petición del CM, si es necesario. El agente observador utiliza una copia del campo etiqueta para eliminar riesgos estructurales con accesos del procesador al campo etiquetas. Por la misma razón, el campo estado tiene dos caminos de acceso, uno para cada agente.

El agente procesador atiende las peticiones del procesador y las respuestas del CM, que llegan por la RV. Cuando el agente procesador no puede servir una petición del procesador, emite una petición al CM utilizando la RI. Al recibir un CC la respuesta a una petición actualiza la cache, el estado y el duplicado de etiquetas si es necesario.

Incluido en el agente procesador está el autómata de reemplazo. Su función es seleccionar un bloque, almacenado en cache, para liberar un contenedor. En este contenedor se ubicará el bloque accedido en un fallo de cache. Cuando se conoce el bloque que se expulsa, el autómata procesador emite una petición de expulsión al CM. Esta petición es para mantener actualizada la información en el directorio y si es el caso, en memoria.

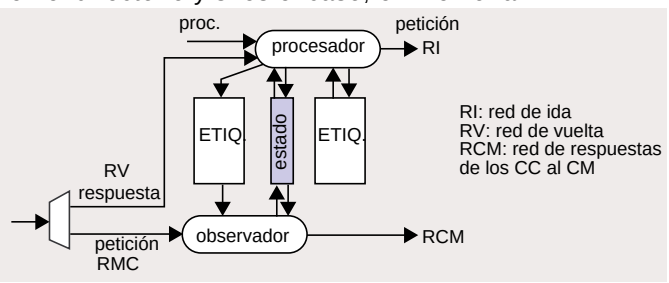


Figura 6.26 Esquema del camino de datos en un CC.

En un acceso a memoria que es fallo en cache y determina la expulsión de un bloque, en primer lugar se realizan las acciones encaminadas a expulsar el bloque. Posteriormente se inician las acciones para servir el fallo de cache.

El agente observador procesa peticiones del CM. Para ello, accede al duplicado de etiquetas, con el objetivo de conocer el contenedor de cache, en el cual se almacena el bloque referenciado en la petición. En un protocolo MLI es posible que memoria no almacene el bloque actualizado. En estas condiciones, el CM, para responder a una petición de un CC, necesita efectuar una petición a otro CC para que suministre el bloque. Por otro lado, el CM al procesar un mensaje de petición de bloque con intención de modificación, utiliza la información disponible en el directorio, para efectuar las peticiones de invalidación necesarias. La red que transporta los mensajes del CM a los CC mantiene el orden en el cual son emitidos los mensajes por el CM (redes lógicas RMC y RV). Por tanto, no es necesario que las caches respondan a peticiones de invalidación. Sólo es necesario que respondan a las peticiones que requieren suministrar el bloque.

Camino de datos en el controlador de memoria

Posteriormente al detallar el protocolo MLI se observa que el CM, al procesar una petición de un CC, puede emitir varias peticiones y una respuesta para finalizar una transacción. El conjunto de CC que reciben una petición no incluye al CC que recibe la respuesta. También, cuando un bloque no está actualizado en memoria, el CM emite un mensaje de petición a un CC, el cual responderá con el bloque de datos. Finalmente, el CM emitirá una respuesta al CC que inició la transacción[19].

En la Figura 6.27 se muestra un esquema del camino de datos del CM en el que se distinguen dos agentes. El agente que procesa las peticiones (P) y el agente que procesa las respuestas (R).

El agente P accede al directorio y a memoria y emite peticiones, si es necesario, a los CC utilizando conexiones punto a punto. Cuando debe esperar una respuesta, a una petición que ha emitido, actualiza el estado del bloque en el directorio y almacena la petición que está procesando, que denominamos petición pendiente, en el buffer de transacciones pendientes (BTP). En este buffer se almacena la dirección del bloque y un identificador de la petición al CC. Este identificador lo utiliza un CC en la respuesta que efectúa al CM.

El agente R empareja una respuesta con la petición pendiente almacenada en el BTP y después actualiza el directorio y la memoria. La recepción y procesado de la respuesta se indica en la entrada asociada en BTP. El agente P está monitorizando BTP por la llegada de respuestas. Cuando el agente R indica que se ha procesado una respuesta, el agente P emite la respuesta al CC que ha iniciado la transacción.

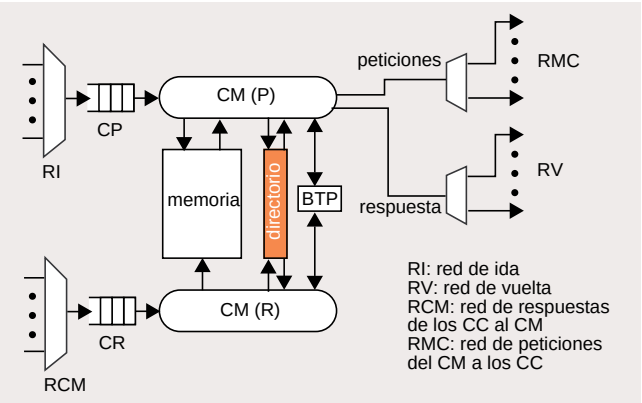


Figura 6.27 Esquema del camino de datos en el CM.

19. En este capítulo la red RMC y la red RV son una única red.

Mensajes del protocolo

En la Figura 6.28 se muestra un esquema genérico de la transmisión de mensajes con peticiones y respuestas entre los CC y el CM y viceversa. Notemos que el esquema se ha linealizado para que todos los mensajes fluyan de izquierda a derecha. Como los CC emiten peticiones y reciben peticiones y respuestas y el CM emite peticiones y recibe respuestas, estos elementos se dibujan más de una vez en el esquema.

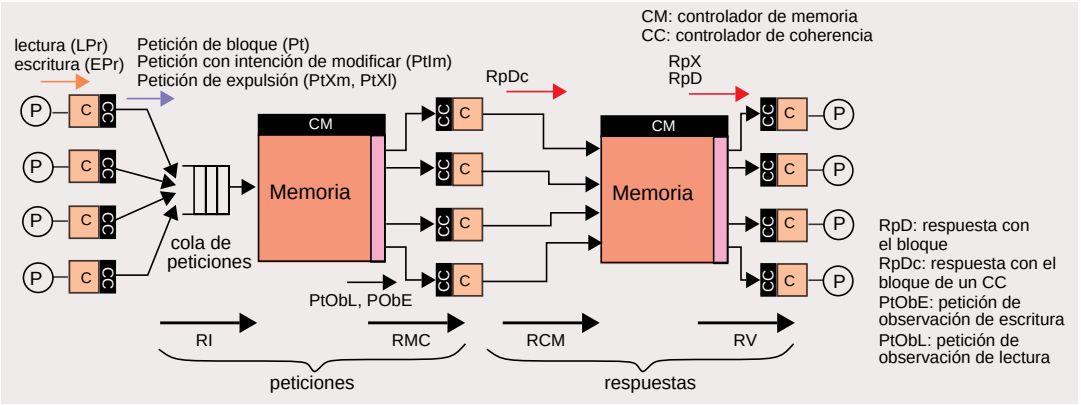


Figura 6.28 Esquema de la transmisión de mensajes en un protocolo de directorio MSI.

En la descripción del protocolo se distinguen los siguientes tipos de mensajes:

- Peticiones de un CC al CM (Pt, PtIm, PtXm, PtXl)
- Peticiones del CM a los CC (PtObL, PtObE)
- Respuestas del CM a un CC (RpD, RpX)
- Respuestas de un CC al CM (RpDc)

En las siguientes tablas al describir los mensajes de petición y respuesta se indica la información básica que se transmite en el mensaje. El acrónimo “op” indica operación, el acrónimo “Id” indica identificador de la petición. En la respuesta se utiliza el mismo identificador.

En la Tabla 6.7 se describen las peticiones de un procesador al CC. El procesador efectúa dos operaciones: a) lectura (load) y b) escritura (store). En caso de acierto en cache sólo se accede a la cache. En un fallo de lectura o de escritura se accede a memoria. En una escritura, si no se dispone de exclusividad en el acceso al bloque, también se accede a memoria.

Procesador	Controlador de coherencia (CC)	Comentario
Peticiones al CC	**Respuestas del CC**	
LPr (load): lectura de un dato op | dirección	dato dato	Si es un acierto en cache se lee el dato de cache. En caso contrario se inicia una transacción de lectura de bloque (Pt) y se asigna un contenedor para almacenar el bloque. Posteriormente se suministra el bloque.
EPr (store): escritura de un dato op | dirección | dato	confirmación de la escritura confirmación	Si es un fallo o no se dispone de exclusividad en el acceso al bloque se solicita el bloque con intención de modificarlo (PtIm). Cuando se dispone de los derechos de acceso se actualiza el bloque en cache.

Tabla 6.7 Peticiones del procesador y respuestas.

En la Tabla 6.8 se describen las peticiones de un CC al CM y las acciones del CC para gestionar la ubicación de bloques en la cache (conflictos, CcRe). Se distinguen los mensajes de petición de bloque (Pt) y petición de bloque con intención de modificación (PtIm). También se describe la acción de expulsión de un bloque debido a un conflicto.

Controlador de coherencia (CC)	Controlador de memoria (CM)	Comentario
Mensajes de petición al CM y acciones	**Mensajes de respuestas**	
Pt: petición de bloque op | dirección | Id	RpD: respuesta con el bloque Id | bloque	Se lee el bloque de la memoria, se actualiza el directorio y se suministra el bloque.
PtIm: petición de bloque con intención de modificación op | dirección | Id	RpD: respuesta con el bloque Id | bloque	El CM suministra el bloque después de leerlo de memoria u obtenerlo de otra cache. Se actualiza el directorio en consecuencia.
CcRe: expulsión de un bloque.		Se invalida la información del contenedor.
PtXm: petición de expulsión de un bloque en estado M op | dirección | bloque | Id PtXl: petición (notificación) de expulsión de un bloque en estado L op | dirección | Id	RpX: respuesta de confirmación Id | confirmación	Se notifica al directorio de la expulsión de un bloque, para que actualice la información de estado del bloque en el directorio y actualice memoria, si es el caso.

Tabla 6.8 Peticiones del CC al CM y respuestas del CM.

En la Tabla 6.9 se describen las peticiones del CM a los CC y la acción del CM para tener actualizada la información de presencia de los bloques. El CM envía mensajes de petición de observación de lectura o de escritura para mantener la coherencia. Un CC responde incluyendo el bloque en un mensaje de respuesta cuando tiene el bloque en exclusividad.

Notemos que un CC siempre responde a una petición PtObL. El CC suministra el bloque. En cambio, un CC sólo responde a una petición PtObE si tiene el bloque en exclusividad. En cualquier caso un CC invalida el bloque referenciado en una petición PtObE.

Controlador de memoria	Controlador de coherencia	Comentario
Mensajes de petición a CC y acciones	**Respuestas**	
PtObE: petición de observación de escritura op | dirección | Id	RpDc: respuesta con el bloque, si es el caso Id | bloque	Si el CC que recibe la petición tiene el bloque en exclusividad emite una respuesta con el bloque. En cualquier caso se invalida el bloque.
PtObL: petición de observación de lectura op | dirección | Id	RpDc: respuesta con el bloque Id | bloque	El CC emite una respuesta con el bloque y cambia el estado del bloque para indicar que no hay exclusividad.
Actualización del directorio. Dev: actualización de memoria		Mantiene actualizada la información de compartición de los bloques. Actualiza la memoria con el bloque de datos.

Tabla 6.9 Peticiones del CM a los CC.

Estados y transiciones

En un protocolo de coherencia de cache, donde se utiliza escritura retardada, es necesario en ocasiones obtener el bloque de datos de una cache para suministrar el bloque a otra cache. En estas condiciones, el CM deberá solicitar el bloque a un CC y esperar a que responda con el bloque. Para identificar la espera de una respuesta de un CC se utilizan estados transitorios en el CM[20].

Directorio. Los estados estables de un bloque en el directorio son tres (Figura 6.29): a) no presente (NP), b) copias del bloque en algunas caches para accesos de lectura (L) y c) copia del bloque en exclusividad en una cache (M).

Estos tres estados están codificados en el VP y el bit de exclusividad (BE) asociado a cada bloque. Si todos los bits del VP están desactivados el estado es NP. Para distinguir entre el estado L y el estado M se utiliza el BE. Si este bit está desactivado, los bits del VP que están activados identifican los procesadores que tienen copia del bloque. En caso contrario, sólo un bit del VP puede estar activado e identifica el procesador que tiene copia del bloque en exclusividad.

20. Notemos la diferencia entre el protocolo de directorio MLI y el protocolo de directorio VI. En este último las cache utilizan escritura inmediata para actualizar memoria. Entonces, como la memoria está actualizada, no es necesario solicitar el bloque a un CC para que el CM pueda responder a una petición.

El etiquetado de los estados transitorios sigue las reglas descritas al describir los protocolos de observación en Capítulo 4.

El número de estados transitorios en el directorio es dos (ML, MM). Se utilizan para identificar la espera de la respuesta de un CC que suministra un bloque solicitado por el CM.

Cache. En una cache un bloque puede tener tres estados estables (Figura 6.29): a) inválido (I), b) copia del bloque para operaciones de lectura (L) y c) copia del bloque en exclusividad, lo cual permite modificarlo (M). Para identificar la espera de la respuesta del CM a una petición del CC se utilizan cinco estados transitorios (IM, IL, LM, MI, LI).

Para describir el protocolo se utilizan las dos peticiones del procesador que requieren acceder al directorio (fallo en una instrucción load, store y una instrucción store que accede a un bloque sin permiso de exclusividad) y las dos posibilidades de ubicación del bloque solicitado, en memoria o en una cache. Posteriormente se detallan las transiciones entre estados en una expulsión de un bloque de cache.

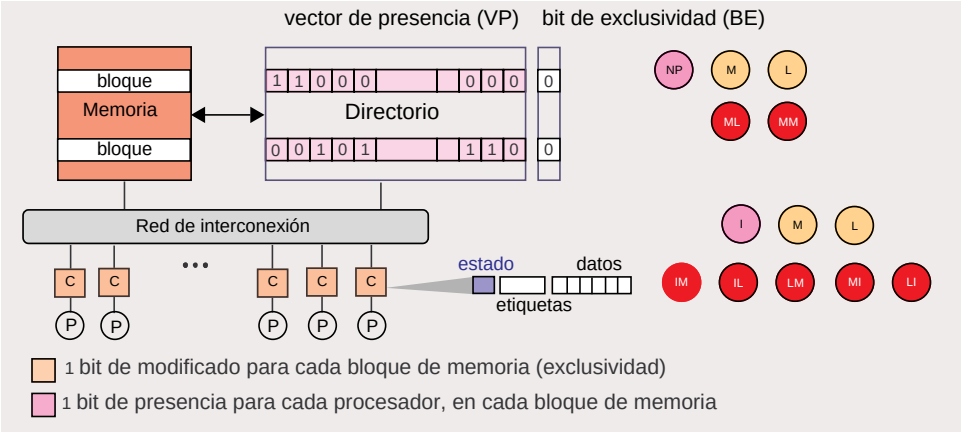


Figura 6.29 Protocolo de directorio MLI. Estados de un bloque en el directorio y en un contenedor de cache.

En la descripción se mostrarán las transiciones entre estados de un bloque en el CC que efectúa la petición (agente procesador), en el CM y en otros CC (agente observador).

Fallo en lectura

En la Figura 6.30 se muestra el flujo de mensajes en un fallo de lectura. El CC emite un mensaje con una petición de lectura de bloque (Pt). El CM accede al directorio para leer el VP y el BE (estado del bloque) y especulativamente accede a memoria para leer el bloque[21]. El CM después de analizar el estado

del bloque, el cual es NP o L, emite un mensaje de respuesta (RpD), que incluye el bloque, al CC que ha efectuado la petición. Además, el CM actualiza el VP añadiendo al CC que ha efectuado la petición.

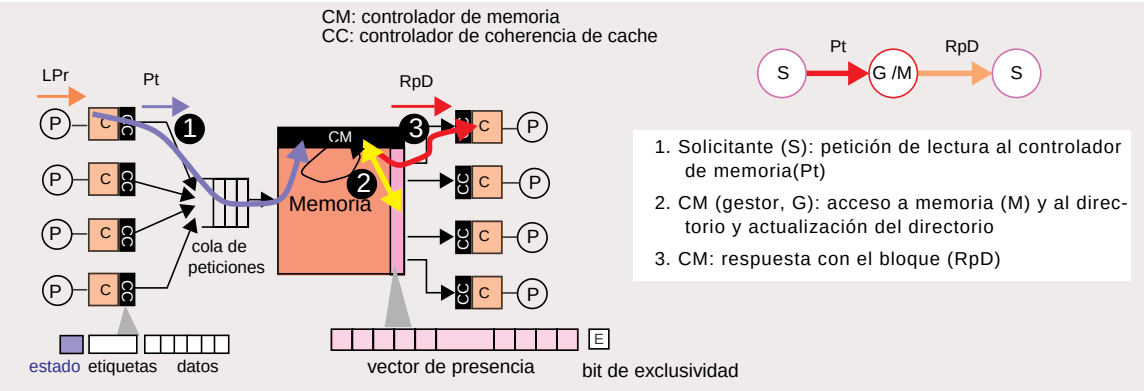


Figura 6.30 Protocolo de directorio MLI. Flujo de mensajes en un fallo de lectura.

En la Figura 6.31 se muestran las transiciones entre estados en un fallo de lectura en: a) la cache del CC que efectúa la petición (solicitante), b) el directorio (CM) y c) otros CC. El CC envía el mensaje de petición de lectura y establece el estado transitorio del bloque como IL. El bloque permanece en este estado hasta que el CC recibe la respuesta del CM, la cual es procesada por el agente procesador, que establece como estado estable del bloque el estado L.

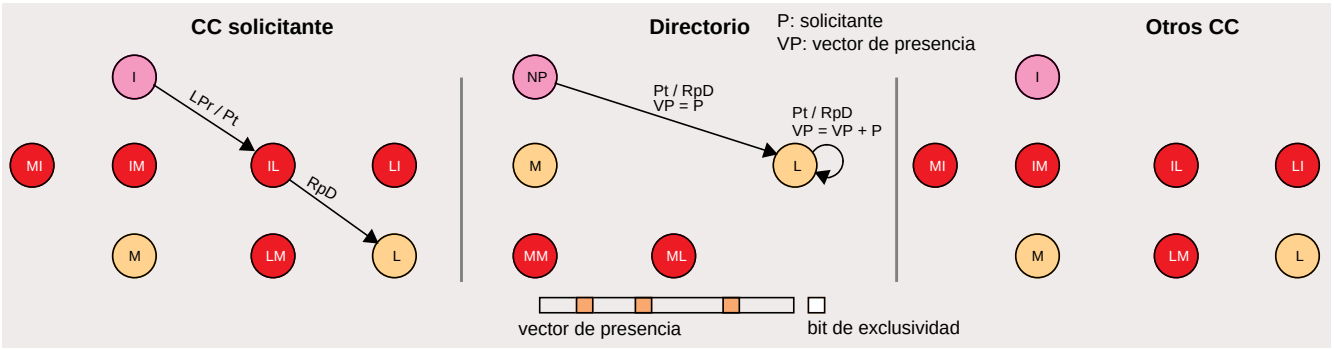


Figura 6.31 Protocolo de directorio MLI. Transiciones entre estados en un fallo de lectura.

21. Decimos que es especulativo debido a que el acceso se efectúa en paralelo con el acceso al directorio para leer el VP del bloque. Por tanto, no se conoce si memoria almacena un bloque válido.

El CM al recibir el mensaje de petición de lectura accede al directorio y especulativamente a memoria. A partir del estado del bloque en el directorio, el CM determina que puede responder a la petición utilizando el bloque leído de memoria. El CM, además de enviar la respuesta, actualiza el VP añadiendo al CC, que ha efectuado la petición, en el VP. Los otros CC no reciben peticiones de coherencia desde el CM.

Fallo en lectura y bloque en estado M en otra cache

En la Figura 6.32 se muestra el flujo de mensajes entre el CC que efectúa la petición y el CM y entre este último y el CC que tiene el bloque en exclusividad.

Respecto de la descripción de un “fallo de lectura”, la diferencia es que el CM debe obtener el bloque de una cache que lo tiene en exclusividad. Este hecho lo determina el CM después de analizar el estado del bloque en el directorio. Para obtener el bloque, el CM efectúa una petición de observación de lectura de bloque (PtObL) al CC correspondiente y espera que le responda con un mensaje que incluya el bloque (RpDc). Al recibir el CM esta respuesta, responde al CC que ha efectuado la petición (RpD) y actualiza memoria y el estado del bloque en el directorio.

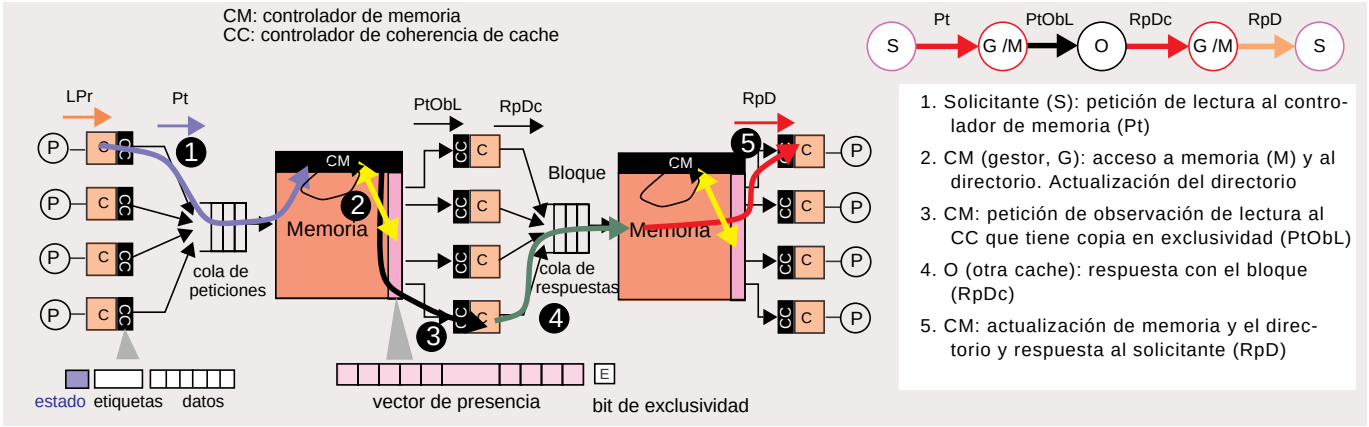


Figura 6.32 Protocolo de directorio MLI. Flujo de mensajes en un fallo de lectura y una cache tiene el bloque en exclusividad.

En la Figura 6.33 se muestran las transiciones entre estados en la cache del CC solicitante, el directorio y otros CC. La diferencia con la Figura 6.30 es que el CM parte de un bloque en estado M. Entonces, el CM establece el estado transitorio ML, al emitir la petición al CC que tiene el bloque en exclusividad. Este CC, al recibir esta petición (PtObL), responde con un mensaje que incluye el bloque y establece como nuevo estado del bloque el estado L. El CM al

recibir la respuesta (RpDc) y procesarla envía un mensaje de respuesta al CC del solicitante (RpD). Además, actualiza memoria con el bloque y modifica el VP y el BE, estableciendo en el directorio como estado del bloque el estado L.

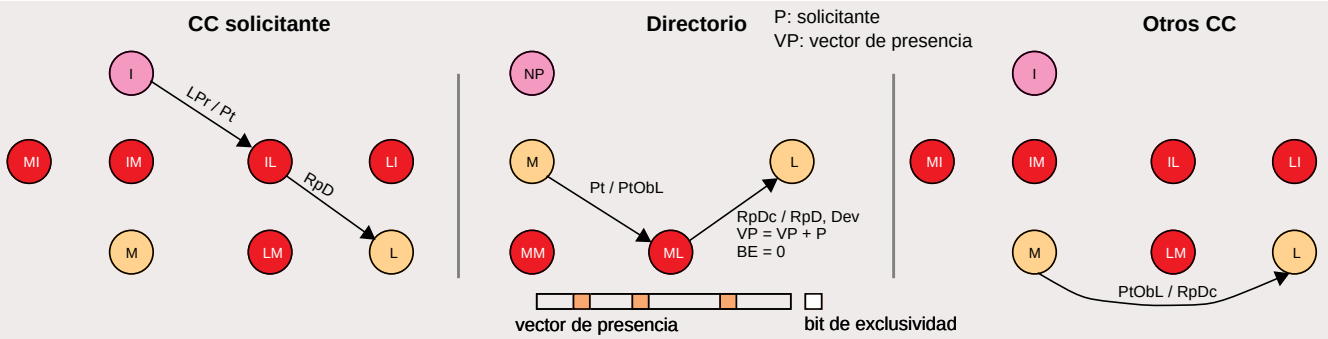


Figura 6.33 Protocolo de directorio MLI. Transiciones entre estados en un fallo de lectura y una cache tiene el bloque en exclusividad.

Fallo de escritura o petición de exclusividad

En la Figura 6.34 se muestra el flujo de mensajes en un fallo de escritura entre un CC y el CM y entre éste y posiblemente varios CC. El CC del solicitante emite un mensaje con una petición de bloque con intención de modificación (PtIm). El CM, al recibir el mensaje, lee el estado del bloque en el directorio y accede especulativamente a memoria. A partir de la información de estado del bloque en el directorio, el CM determina las peticiones de observación de escritura que debe emitir (PtObE). A la vez que envía los mensajes PtObE también envía la respuesta al solicitante (RpD).

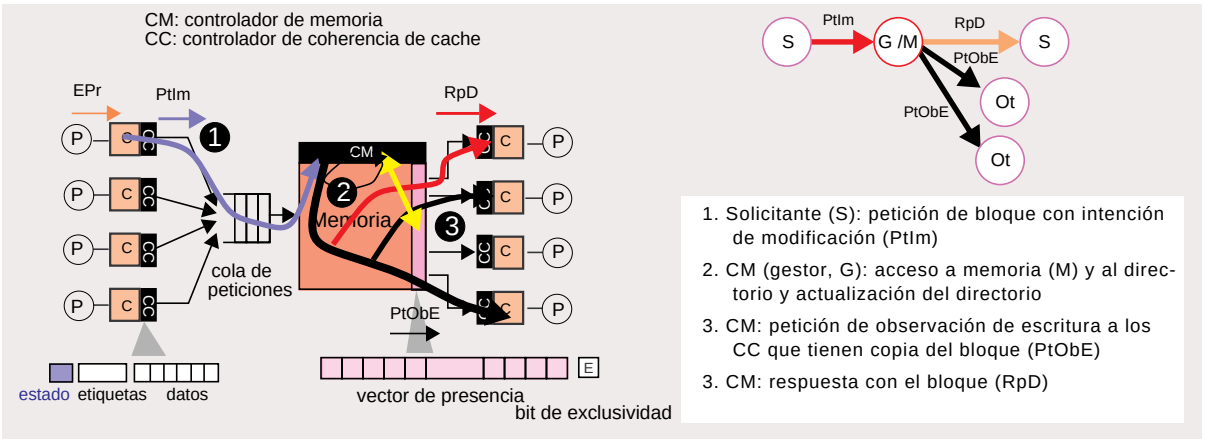


Figura 6.34 Protocolo de directorio MLI. Flujo de mensajes en un fallo de escritura.

Como se utiliza una red ordenada (RV), para transmitir los mensajes entre el CM y los CC, no es necesario esperar una respuesta de los CC que reciben una petición PtObE. Por tanto, estos CC no emiten una respuesta al recibir dicha petición.

En la Figura 6.35 se muestran las transiciones entre estados en la cache del CC que efectúa la petición, el directorio y otros CC. El CC del solicitante establece un estado transitorio (IM o LM), en función del estado estable inicial, esperando la respuesta. Al recibir la respuesta el estado estable del bloque será M.

En el directorio el estado del bloque pasa de L a M o de NP a M, el VP se actualiza en consecuencia y el BE se activa. Además, el CM emite mensajes de respuesta (RpD) y de petición de observación de escritura (RpObE).

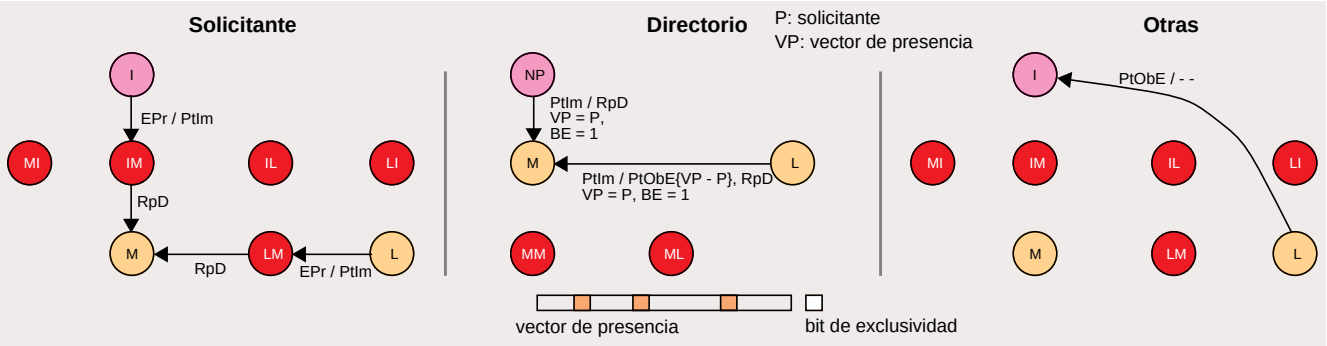


Figura 6.35 Protocolo de directorio MLI. Transiciones entre estados en un fallo de escritura.

Cuando un CC emite una petición PtIm estando el bloque en estado L, el bit de presencia del CC está activado en el VP. Por tanto, hay que excluirlo de la lista de CC que reciben una petición PtObE (VP - P). El CC, al recibir la respuesta RpD, siempre almacena el bloque recibido en el contenedor correspondiente, tanto si el estado estable inicial es L como I.

Los CC que tienen copia del bloque, al recibir la petición PtObE invalidan la copia del bloque.

Fallo en escritura o petición de exclusividad y bloque en estado M en otra cache

En la Figura 6.36 se muestra el flujo de mensajes en un fallo de escritura o petición de exclusividad y una cache tiene el bloque solicitado en exclusividad. El CC solicitante envía un mensaje de petición de bloque con intención de modificación (PtIm). El CM al leer el estado del directorio determina que un CC tiene el bloque en exclusividad. Por tanto, envía una petición de observación

de escritura (PtObE) a este CC. Como respuesta a esta petición, el CC emite un mensaje de respuesta al CM que incluye el bloque (RpDc). Entonces, el CM responde al CC solicitante suministrando el bloque (RpD). Además, actualiza el estado del bloque en el directorio.

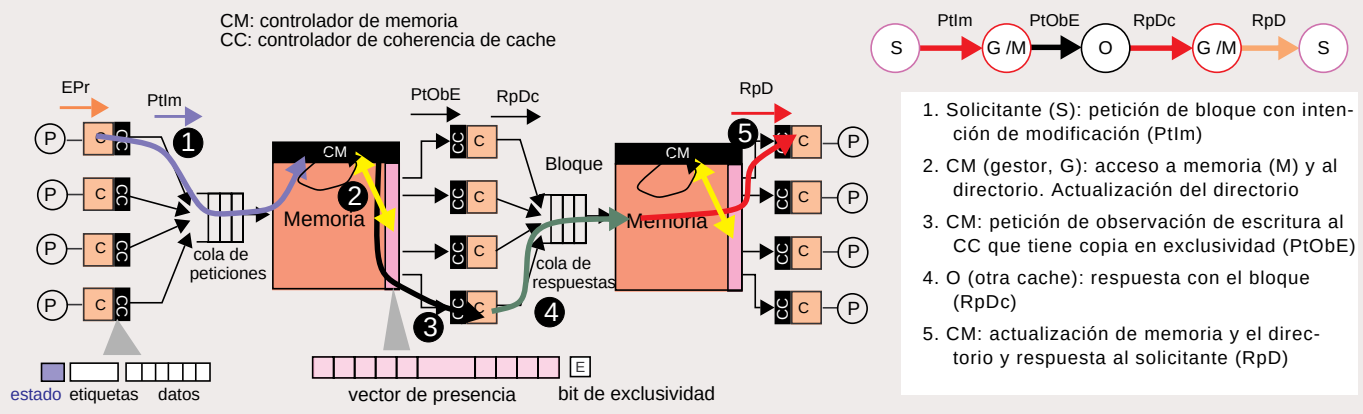


Figura 6.36 Protocolo de directorio MLI. Flujo de mensajes en un fallo de escritura y otra cache tiene el bloque en exclusividad.

En la Figura 6.37 se muestran las transiciones entre estados en el CC del solicitante, el directorio y otros CC. La transición en el solicitante entre los estados I y M es la misma que en la Figura 6.35. El CM, al emitir la petición PtObE, establece como estado transitorio del bloque MM. Cuando se recibe la respuesta del CC, que tiene el bloque en exclusividad, el CM cambia el estado del bloque al estado M. Entonces, el CM envía el mensaje de respuesta al CC solicitante. El CM actualiza el estado identificando al nuevo procesador que tiene el bloque en exclusividad. Además, el CC cuya cache tenía el bloque en exclusividad, al emitir la respuesta cambia el estado del bloque del estado M al estado I.

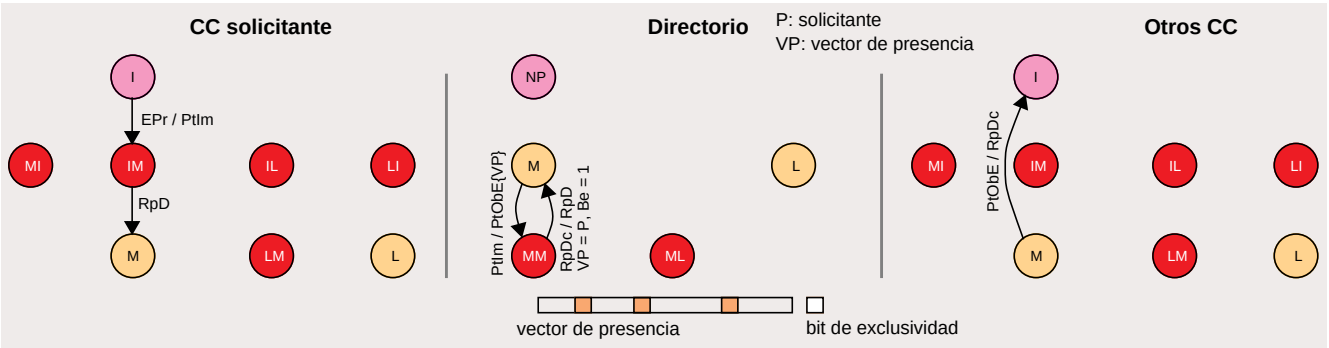


Figura 6.37 Protocolo de directorio MLI. Transiciones entre estados en un fallo de escritura y una cache tiene el bloque en exclusividad.

Expulsión de un bloque de cache

En la Figura 6.38 se muestra el flujo de mensajes cuando se expulsa un bloque de cache. El CC efectúa una petición de expulsión (PtXm, PtXl) que va acompañada del bloque cuando el estado del bloque es M (PtXm). El CM en cualquier caso actualizada la entrada correspondiente en el directorio. Además, se actualiza memoria si el bloque expulsado estaba en el estado M en la cache. El CM responde al CC confirmando que se ha realizado la acción solicitada.

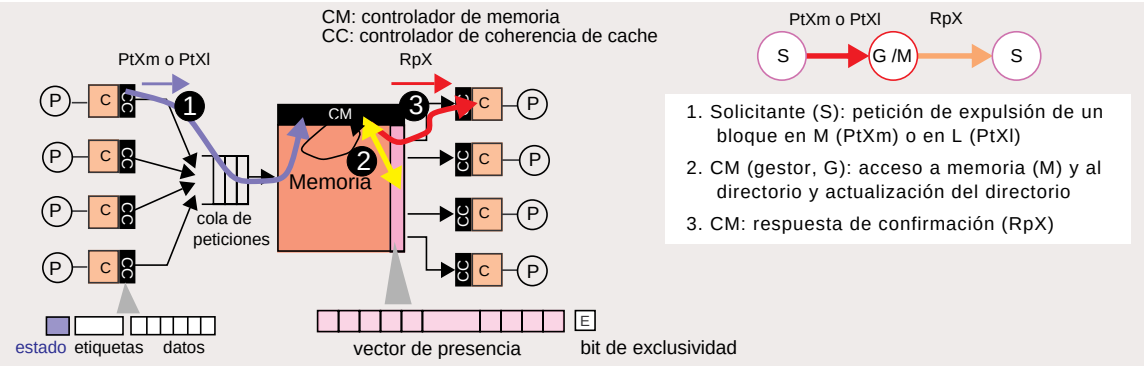


Figura 6.38 Expulsión de un bloque de cache.

En la Figura 6.39 se muestran las transiciones entre estados en: a) la cache del CC que expulsa el bloque, b) el directorio y c) otros CC. En otros CC no se producen transiciones entre estados. Para distinguir entre la expulsión de un bloque en el estado M o en el estado L, se utilizan distintos tipos de peticiones y también estados transitorios distintos, MI y LI respectivamente. Cuando el CM procesa la petición actualiza la memoria si la petición es PtXm y en cualquiera de los dos casos, se actualiza la entrada correspondiente del bloque en el directorio. Finalmente el CM envía una respuesta de confirmación. El CC al recibir la respuesta cambia el estado del bloque al estado I.

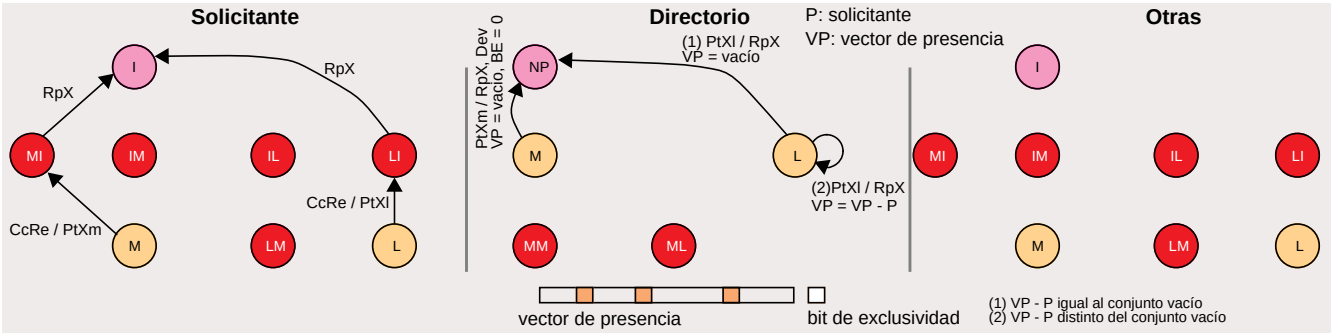


Figura 6.39 Protocolo de directorio MLI. Transiciones entre estados cuando se expulsa un bloque.

Diagrama completo de estados y transiciones

En la Figura 6.40 y la Figura 6.41 se muestran, respectivamente, los diagramas de transiciones entre estados de un bloque en cache y en el directorio. En el primer diagrama están incluidas las peticiones del procesador que no requieren iniciar transacciones explícitas de coherencia. En el CC se distinguen, en diagramas separados, las transiciones iniciadas por el agente procesador y el agente observador (Figura 6.40).

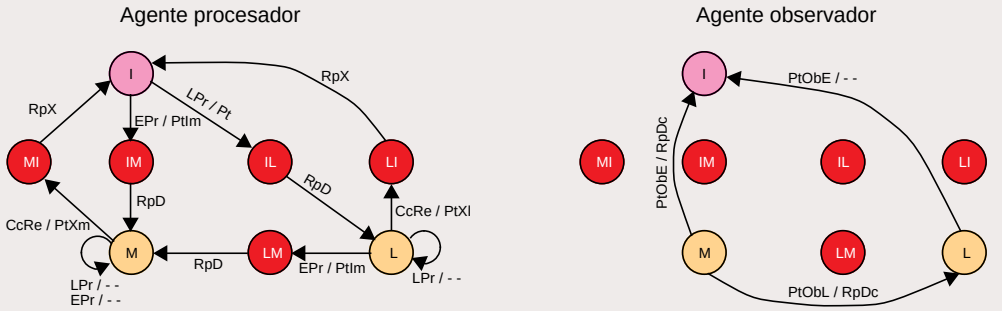


Figura 6.40 Protocolo de directorio MLI. Estados y transiciones entre estados de un bloque en una cache.

En el diagrama del CM (Figura 6.41) hay que distinguir dos casos al procesar la petición PtXl de un CC. El estado final depende de si el CC, que efectúa la petición, es el único que está en el vector de presencia.

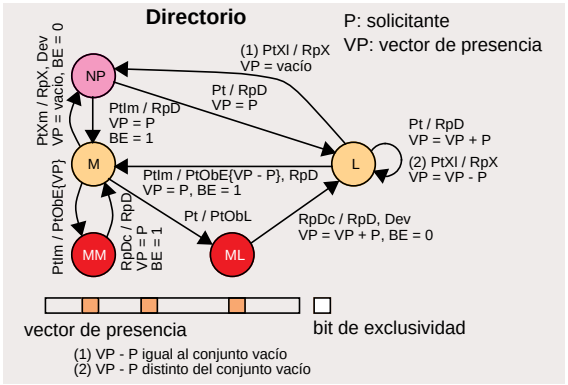


Figura 6.41 Protocolo de directorio MLI. Estados y transiciones entre estados de un bloque en el directorio.

Tablas de estados y transiciones

En la Tabla 6.10 se muestran en formato tabla los estados y las transiciones entre estados de un bloque en una cache. Las casillas que no contienen información indican un error. En un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

Estados			Eventos del procesador y reemplazo			Eventos externos (respuestas y petición)			
			LPr	EPr	CcRe	RpD	RpX	PtObL	PtObE
	Estables	I	Pt; IL	PtIm; IM					
		L	- - ; L	PtIm; LM	PtXl; LI				- -; I
		M	- - ; M	- - ; M	PtXm; MI			RpDc: L	RpDc; I
	transitorios	IL				- -; L			
		IM				- -; M			
		LM				- -; M			
		LI					- -; I		
		MI					- - ; I		

Tabla 6.10 Protocolo de directorio MLI. Tabla de estados y transiciones de un bloque en cache.

En la Tabla 6.11 se muestran en formato tabla los estados y transiciones entre estados de un bloque en el CM. En el evento PtXl se distingue el caso de que el CC sea el único que está en el vector de presencia o haya más CC.

Estados			Eventos del controlador de coherencia					
					PtXl			
			Pt	PtIm	VP = P	P ∈ VP P ≠ VP	PtXm	RpDc
	Estables	NP	RpD; L, VP = P	RpD; M VP = P, BE = 1				
		L	RpD; L, VP = VP + P	PtObE {VP - P}, RpD; M, VP = P, BE = 1	RpX; NP, VP = vacío	RpX; L VP = VP - P		
		M	PtObL; ML	PtObE {VP}; MM			RpX, Dev; NP, VP = vacío, BE = 0	
	transitorios	ML						RpD, Dev; L VP = VP + P, BE = 0
		MM						RpD; M VP = P, BE = 1

Tabla 6.11 Protocolo de directorio MLI. Tabla de estados y transiciones de un bloque en el directorio.

Representación de transacciones y transiciones entre estados

En este apartado se muestran tres formas de representar las transacciones y transiciones al ejecutar una secuencia de accesos a memoria: a) en formato tabla, b) mediante un diagrama temporal y c) mediante un diagrama temporal simplificado. Finalmente se muestra, mediante varios gráficos, una animación.

Representación en formato tabla

Utilizaremos una tabla, donde cada fila representa un acceso a memoria y no se gestiona el siguiente acceso hasta que ha finalizado el anterior. El número de columnas se incrementa, respecto del protocolo de directorio VI, con el objetivo de representar las transacciones que requieren cuatro pasos. Entre las columnas de memoria y suministro se intercalan dos nuevas columnas. Estas columnas se utilizan para representar las peticiones del CM a un CC (red RMC) y la respuesta del CC (red RCM). En la columna correspondiente a RMC, además de la petición se indica la secuencia de ordinales de los procesadores cuyos CC reciben la petición.

Después de la columna correspondiente a VP se añade una columna que se corresponde con el BE de la entrada del directorio (E).

En el caso de una transacción que requiere dos pasos y el CM efectúa peticiones, además de emitir una respuesta, las peticiones se representan en la columna correspondiente a RV. Recordemos que las redes lógicas RV y RMC se pueden implementar utilizando sólo una red física. Para representar las respuestas y las peticiones se utilizan dos filas.

Ejemplo. En la Tabla 6.12 se muestra una secuencia de accesos a memoria realiza por tres procesadores. Suponemos que las variables t y u no están almacenadas en cache y los bloques que las contienen se almacenan en contenedores distintos de cache.

El primer acceso a memoria es una instrucción load y se produce un fallo en la cache C1. Al emitir el mensaje de petición de bloque, el estado transitorio del bloque es IL. La variable accedida es t y el VP en el directorio sólo indica la presencia del bloque en la cache C1. El BE está desactivado. La RV transporta un mensaje de respuesta de datos y el estado estable del bloque en la cache C1, al finalizar la transacción, es L.

	C 1	C 2	C 3	Red	mem.			Red	Red		Red	C 1		C 2		C 3	
acceso	est.	est.	est.	RI	var.	VP	E	RMC	RCM	sum.	RV	var.	est.	var.	est.	var.	est.
1. P1 load t	IL			Pt	t	1, 0, 0	0			mem.	RpD	t	L				
2. P2 store u		IM		PtIm	u	0, 1, 0	1			mem	RpD			u	M		
3. P1 load u	IL			Pt	u	1, 1, 0	0	PtObL, 2	RpDc	C2	RpD	u	L	u	L		
4. P3 load u			IL	Pt	u	1, 1, 1	0			mem.	RpD					u	L
5. P1 store u	LM			PtIm	u	1, 0, 0	1			mem	RpD	u	M				
											PtObE,2,3			u	I	u	I

Tabla 6.12 Protocolo de directorio MLI. Secuencia de accesos a memoria.

El segundo acceso a memoria es una instrucción store. El bloque no está almacenado en cache. Por tanto, el estado transitorio es IM y el mensaje que se transmite por la RI es PtIm. No hay copia del bloque referenciado en ninguna cache. El CM actualiza el VP y activa el BE. Además, suministra el bloque utilizando la RV. El estado del bloque en cache al recibir la respuesta es M.

El tercer acceso a memoria lo efectúa el CC de la cache C1 y es a la variable u. El CM al procesar el mensaje Pt determina, utilizando la información almacenada en el directorio, que otra cache almacena el bloque en exclusividad. Entonces solicita el bloque mediante una petición PtObL al CC de la cache C2 y espera la respuesta. Cuando llega la respuesta, el CM actualiza la memoria y el directorio y transmite la respuesta por la RV, al CC que efectuó la petición. El estado final del bloque, tanto en la cache que ha suministrado el bloque como en la cache donde se ha almacenado, es L.

El cuarto acceso a memoria, efectuado por el CC de la cache C3, es mimético al primer acceso a memoria, con la salvedad de que el VP indicará copias del bloque en las caches C1, C2 y C3.

El quinto acceso a memoria es una instrucción store en el procesador P1. Al acceder a cache se determina que es acierto, pero no se dispone de exclusividad en el acceso al bloque. Por tanto, el estado transitorio es LM. El mensaje que transmite el CC al CM es PtIm. En el directorio el VP se actualiza para indicar que hay copia del bloque sólo en la cache C1 y además en exclusividad (BE = 1). Las otras copias del bloque, que había en otras caches, se invalidan mediante la emisión, por parte del CM, de un mensaje PtObE a los respectivos CC. En la tabla se representa este hecho utilizando una segunda fila en la columna RV, donde se indica la petición del CM y el ordinal de los CC. El estado estable del bloque en la cache C1 es M y en las otras caches es I.

Diagrama temporal

Para representar las transacciones de dos pasos utilizaremos la misma representación que la descrita en el protocolo de directorio VI. Esto es, en el diagrama sólo representaremos la RV. La red RMC no se representa.

En la Figura 6.42 se muestra una transacción de dos pasos, donde se observa la utilización de las redes lógicas RV y RMC. La primera red lógica se utiliza para transmitir la respuesta y la segunda para transmitir las peticiones. Previamente hemos comentado que el conjunto de destinatarios de las peticiones no incluye el destinatario de la respuesta. Por tanto, se pueden implementar como una única red, que se identifica como RV. Entonces, en las transacciones de dos pasos sólo se indica la red RV como se muestra en la parte derecha de la Figura 6.42.

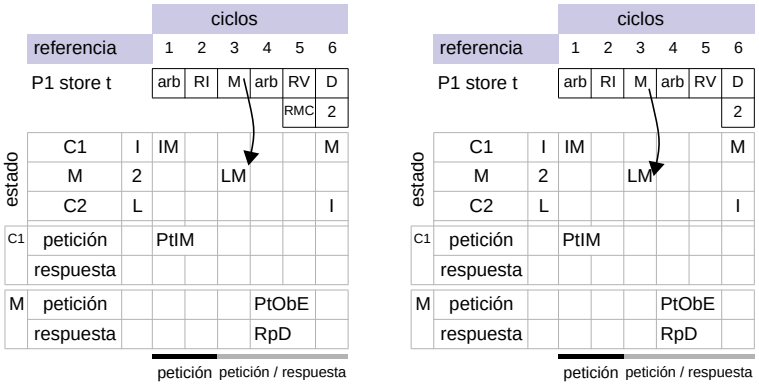


Figura 6.42 Protocolo de directorio MLI: diagrama temporal de acciones de coherencia de 2 pasos

Sin embargo, para identificar mejor peticiones y respuestas, en un diagrama temporal donde se muestra una transacción de cuatro pasos, utilizaremos el acrónimo RMC para indicar una petición del CM (Figura 6.43).

En una transacción que requiere cuatro pasos está involucrado un CC que es distinto del que efectúa la petición. Las fases por las que transcurre el mensaje de petición hasta llegar al CM (M) son idénticas al caso de un transacción de dos pasos. Después de esta fase, el CM solicita la colaboración de un CC para que suministre el bloque. Para ello, es necesario una fase de transmisión en la que se identifica la red que transporta peticiones del CM a los CC (RMC). En la siguiente fase el CC, de la cache que recibe la petición, accede al campo de datos (C). Posteriormente se especifica la red que transmite la respuesta del CC al CM (RCM). La siguiente fase es procesar la respuesta en el directorio (M) y finalmente, las tres fases que se utilizan para representar la transmisión y recepción de un mensaje de respuesta del CM a un CC.

Cuando una cache tiene el bloque en exclusividad, el estado de un bloque en el directorio se representa mediante el ordinal del procesador y la letra E.

En el ejemplo de la parte izquierda de la Figura 6.43 se muestra un fallo de lectura y el bloque está en estado M en otra cache. En la fase arb se indica el estado transitorio en la cache del CC que envía el mensaje de petición. Hay una copia del bloque en el estado M en la cache C2. En la fase M el CM determina que hay que solicitar el bloque al CC que gestiona la cache C2 y establece como estado transitorio del bloque ML.

En la fase RMC se envía una petición de observación de lectura al CC de la cache C2 (PtObL). En la fase Cx, donde x es el ordinal del procesador, el CC de la cache C2 accede al campo de datos de la cache para leer el bloque, actualiza el estado del bloque y en la siguiente fase transmite la respuesta al CM.

El CM, utilizando la respuesta del CC2, actualiza el directorio y la memoria. Posteriormente envía la respuesta, que incluye el bloque, al CC que ha efectuado la petición.

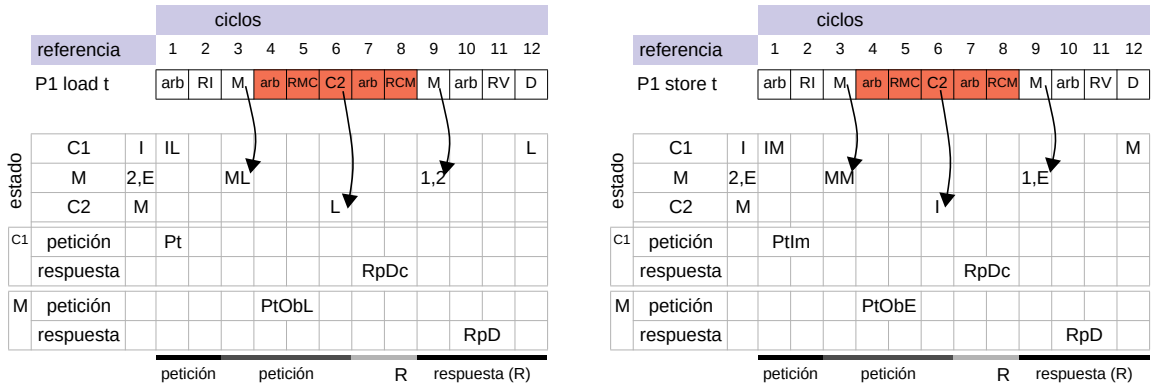


Figura 6.43 Protocolo de directorio MLI: diagrama temporal de acciones de coherencia de 4 pasos.

En el ejemplo de la parte derecha de la Figura 6.43 se muestra un fallo de escritura y el bloque está en estado M en otra cache. La secuencia de mensajes es similar a la secuencia que se muestra en la parte izquierda. Se diferencian en el tipo de mensaje y en consecuencia en los estados de las copias del bloque en las caches y el estado en el directorio.

Ejemplo. En la Figura 6.44 se muestra el diagrama temporal de la secuencia de accesos mostrada en la Tabla 6.12. En la parte central de la Figura 6.44 se especifican las caches de los procesadores, el directorio (M) y

las variables. En estas filas se representa el estado del bloque en cada cache y en el directorio, de la forma descrita en los ejemplos de la Figura 6.43. En la parte inferior se representan los mensajes transmitidos.

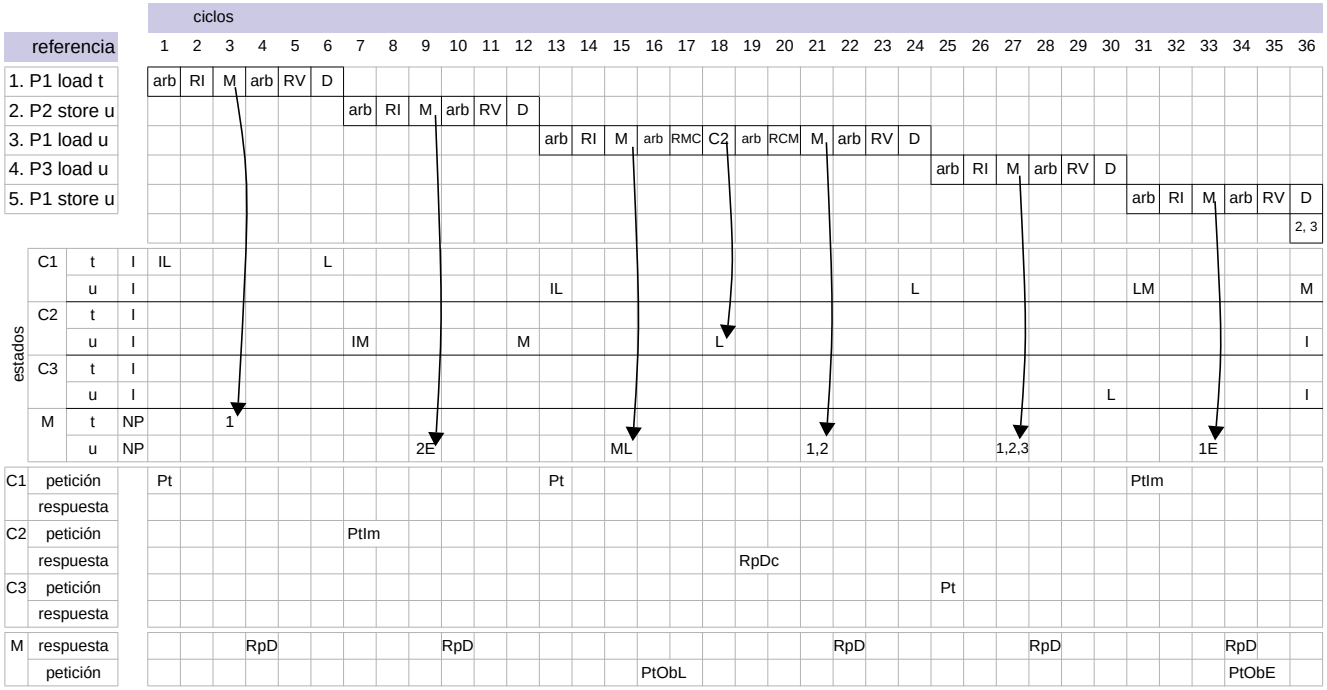


Figura 6.44 Protocolo de directorio MLI. Diagrama temporal de una secuencia de accesos a memoria.

Diagrama temporal simplificado

La idea de este diagrama es representar de forma más esquemática los mensajes y la temporalidad de los mismos. La representación es similar a la utilizada para el protocolo de directorio VI.

El acrónimo de una respuesta de un CC al CM se representan en el CM.

En la Figura 6.45 se muestra un diagrama temporal simplificado, donde se identifican los mensajes entre los CC y el CM y viceversa, correspondientes a la secuencia de accesos mostrada en la Tabla 6.12. También se indica el estado de los bloques en el CM y en los CC.

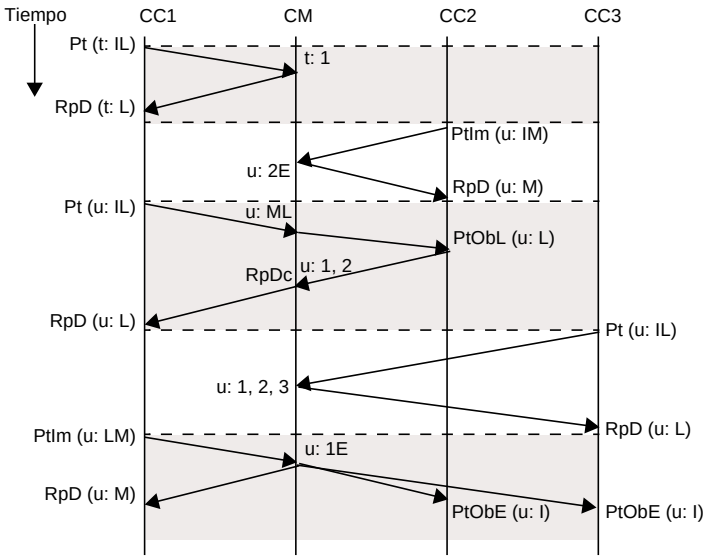


Figura 6.45 Protocolo de directorio MLI. Diagrama temporal simplificado de una secuencia de accesos a memoria concurrentes.

Animación

En la Figura 6.46 se muestra, mediante fotogramas, una animación de la secuencia de accesos a memoria de la Tabla 6.12. Además de las peticiones y respuestas, se muestran los cambios de estado de un bloque, utilizando diagramas de transición entre estados. Cuando un bloque no está almacenado en un contenedor de cache se supone que está en estado inválido. Para reducir la información representada en la figura, se distinguen solicitante, directorio y otros. Tampoco se muestra el estado transitorio MI en un CC, ya que no se utiliza en esta secuencia de accesos a memoria. Los mensajes y el acceso al directorio y memoria se identifican con una secuencia numérica.

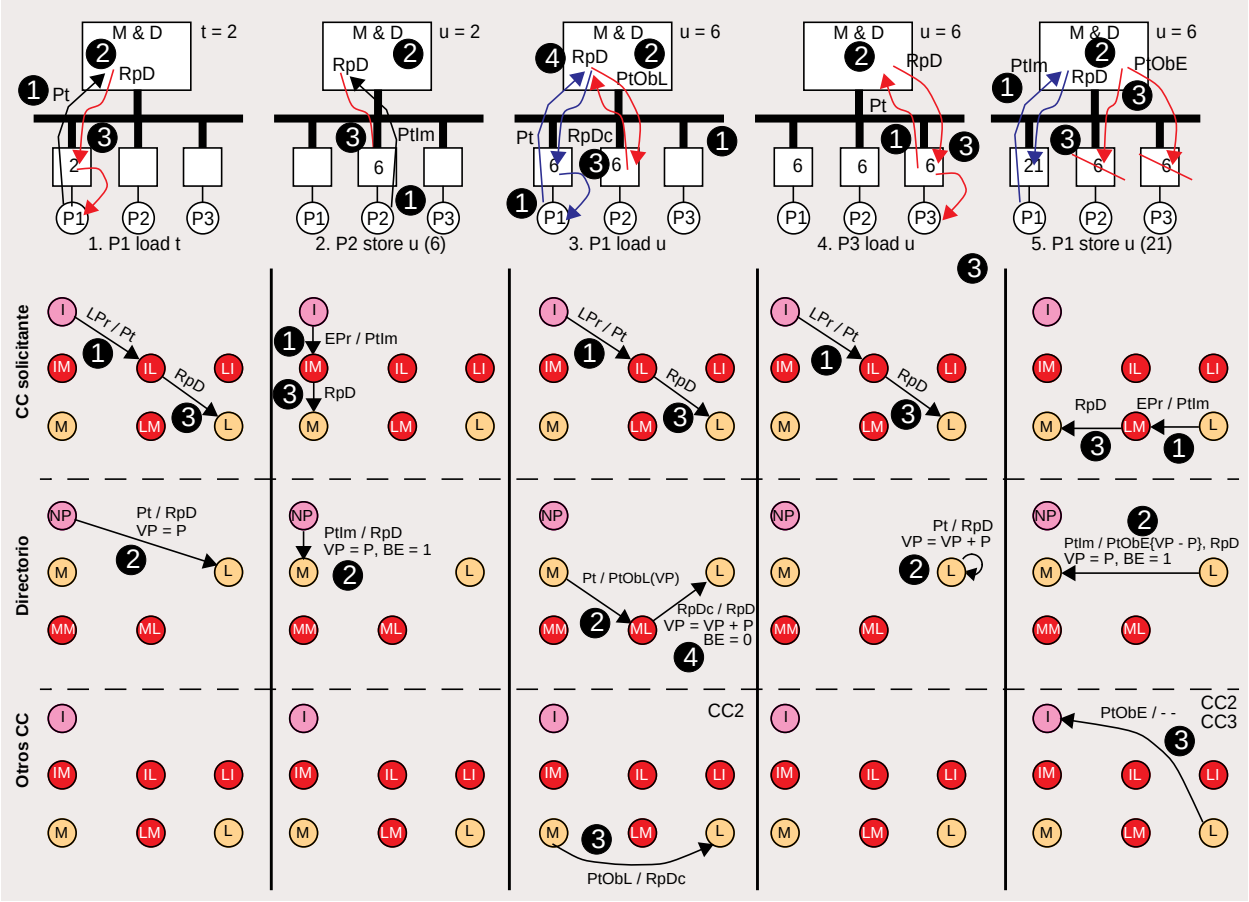


Figura 6.46 Protocolo de directorio MLI. Secuencia de accesos a memoria. Fotogramas de las peticiones y respuestas y cambios de estado.

Verificación no formal de coherencia y consistencia

Coherencia de cache

Orden de programa en accesos a la misma posición de memoria. El procesador efectúa los accesos en el orden determinado por el L.M. El procesador se bloquea en un fallo de lectura o cuando requiere obtener la exclusividad de acceso al bloque, hasta que finaliza la transacción correspondiente.

Propagación de escrituras. Para efectuar una escritura en un bloque es necesario tener acceso al bloque en exclusividad (estado M). Ello garantiza que no hay copias del bloque. La exclusividad de acceso a un bloque se

obtiene al recibir la respuesta a una petición PtIm. Las copias del bloque en otras caches se invalidan durante el procesado de la transacción PtIm en el CM. La respuesta de cada petición de invalidación (PtObE) del CM, a los CC que tienen copia, está implícita al efectuar la emisión de la petición, ya que la red del CM a los CC mantiene el orden de los mensajes.

Serialización de escrituras (atomicidad de escritura).

- Punto de serialización. El árbitro del CM es el punto de serialización de las transacciones PtIm. Las escrituras de un procesador, que tiene un bloque en exclusividad, quedan serializadas por el orden de programa. Se tiene garantía de que no existen copias del bloque.
- Consolidación de una escritura. Una transacción PtIm está consolidada al finalizar la transacción (emitir respuesta al solicitante). Cuando una cache no tiene el bloque en exclusividad, el CM finaliza la transacción PtIm una vez emite las peticiones de invalidación a las caches que tienen copia y la respuesta al CC solicitante. Debido a la propiedad de la red entre el CM y los CC, la emisión de las peticiones de invalidación por parte del CM tienen implícita la respuesta del CC correspondiente. Cuando una cache tiene el bloque en exclusividad, el CM finaliza la transacción PtIm al recibir la respuesta del CC que tiene el bloque en exclusividad y responder al CC solicitante, además de actualizar la memoria. Una escritura a un bloque, al que se tiene acceso en exclusividad, está consolidada después de actualizar la cache. Por otro lado, una escritura es atómica. Una vez finalizada una transacción PtIm y actualizada la cache, una lectura posterior del bloque (con o sin intención de modificar) por parte de otro procesador requiere emitir una petición al CM. El CM responde con el valor establecido por la escritura previa. En el lapso de tiempo entre obtener la exclusividad de acceso al bloque y el suministro del bloque, un procesador puede haber actualizado el bloque varias veces. Una lectura del procesador que tiene el bloque en exclusividad lee un valor consolidado.

Consistencia secuencial de memoria

Orden de programa. El procesador efectúa los accesos a cualquier posición de memoria en el orden determinado por el L.M.

- Lectura: Espera hasta que se obtiene el dato.
- Escritura: El árbitro del CM es el punto de ordenación de todas las transacciones PtIm. La finalización de la transacción es una indicación de que la escritura está consolidada. Si no existe una copia en exclusividad, el CM, durante el procesado de una transacción de escritura, emite peticiones de observación de escritura o invalidación, si es el caso. La emisión de las peticiones de invalidación por parte del CM tienen implícita

la respuesta del CC correspondiente. Si un CC tiene el bloque en exclusividad, el CM espera la respuesta de este CC, la cual contiene el bloque, para emitir la respuesta al CC solicitante. En caso de acierto a un bloque en estado M, la escritura consolida al ser actualizada la cache (existe garantía de que no hay copias del bloque).

Atomicidad de las escrituras.

- Consolidación de una escritura: Para efectuar una escritura es necesario disponer de acceso exclusivo al bloque. El acceso exclusivo garantiza que no hay copias del bloque y se obtiene mediante una transacción PtIm. El CM emite peticiones PtObE a las caches que tienen copia del bloque. La respuesta de estas cache está implícita en la emisión de la petición PtObE, ya que la RV es ordenada. Cuando una cache tiene el bloque en exclusividad, el CM finaliza la transacción PtIm al recibir la respuesta del CC que tiene el bloque en exclusividad y responder al CC solicitante además de actualizar la memoria. Una vez obtenida la exclusividad se actualiza la cache y la escritura está consolidada.
- Suministro del valor en una lectura: Un fallo de lectura de un bloque (con o sin intención de modificarlo) requiere emitir una petición al CM. El valor devuelto en la transacción es el valor establecido en la última escritura consolidada, a la misma posición de memoria. El CM obtiene este valor de la memoria o de otra cache. Un acierto de lectura lee el valor de la copia en cache. Este valor ha sido establecido en un fallo de lectura previo o en una escritura previa del mismo procesador.

ORGANIZACIÓN DEL DIRECTORIO

Como hemos comentado interesa que las caches tengan un duplicado de las etiquetas para que el agente observador no tenga que competir con el agente procesador. El efecto se nota en mayor medida en una petición de observación de escritura cuando sólo requiere invalidación[22]. Notemos que cuando es necesario suministrar el bloque hay que acceder al campo de datos de la cache.

Este duplicado de etiquetas se puede utilizar como directorio, si en lugar de estar en las caches se ubica en memoria (Figura 6.47). En estas condiciones el mensaje de petición del CM a los CC identifica el contenedor en cache (conjunto y elemento dentro del conjunto). Por tanto, el agente observador puede acceder directamente al estado. Sin embargo, como hay que mantener coherente el duplicado de etiquetas, es necesario mantener el directorio

22. Por ejemplo, en el protocolo VI.

preciso notificando todas las expulsiones. Como el algoritmo de reemplazo está en las caches privadas, se notifica al directorio el contenedor que queda libre (elemento del conjunto).

Un beneficio es que el tamaño de un duplicado de etiquetas es mucho menor que un directorio con VP, ya que el número total de contenedores de las caches es mucho menor que el número de bloques de memoria. Una deficiencia, o desventaja de un duplicado de etiquetas, es que para cada petición se efectúa una búsqueda asociativa. Hay que acceder al duplicado de etiquetas de todas las caches. Es similar a la búsqueda que efectúan todos los agentes observadores cuando se utiliza un bus como red de interconexión.

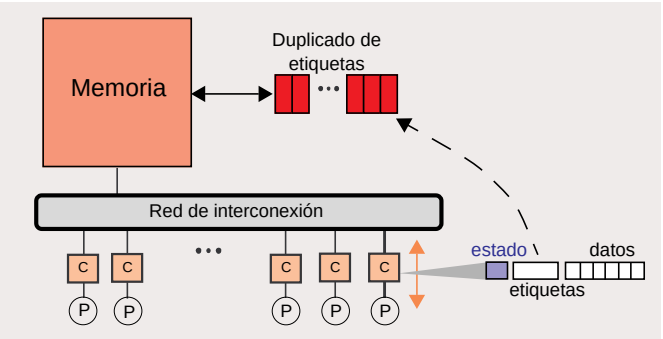


Figura 6.47 Duplicado de etiquetas de cache como directorio.

Ejercicio

Indique el número de comparaciones que se efectúan en un directorio que está organizado como un duplicado de etiquetas al procesar una petición de bloque.

Respuesta

Suponemos que todas las caches tiene la misma asociatividad. El número de comparaciones es igual al número de cache privadas por la asociatividad de las caches.

Ultimo nivel de cache con inclusión: directorio

Una alternativa para reducir la latencia de acceso a memoria es utilizar una cache compartida inclusiva. Esto es, el contenido de todas las caches privadas está almacenado en la cache compartida.

En una organización de este tipo, el directorio está asociado a la cache compartida. Al ser inclusiva se tienen identificados todos los bloques con copias en las caches privadas.

En la Figura 6.48 se muestra una organización del directorio con vectores de presencia. También se puede utilizar una organización con un duplicado de las etiquetas de las caches privadas.

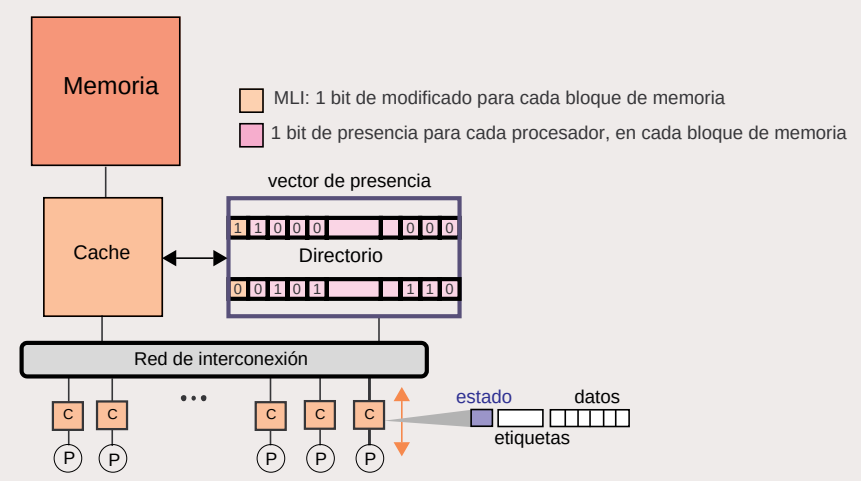


Figura 6.48 Multiprocesador con cache compartida inclusiva

Optimización. En lugar de almacenar en el duplicado de etiquetas las etiquetas de las caches privadas se puede almacenar el identificador del contenedor en la cache compartida. Con ello se reduce el número de bits que debe almacenarse y el tamaño de las comparaciones. Para ello, hay que acceder a las etiquetas de la cache compartida antes de acceder al directorio. El acceso se puede efectuar en paralelo con el acceso al campo de datos de la cache compartida.

EJEMPLOS

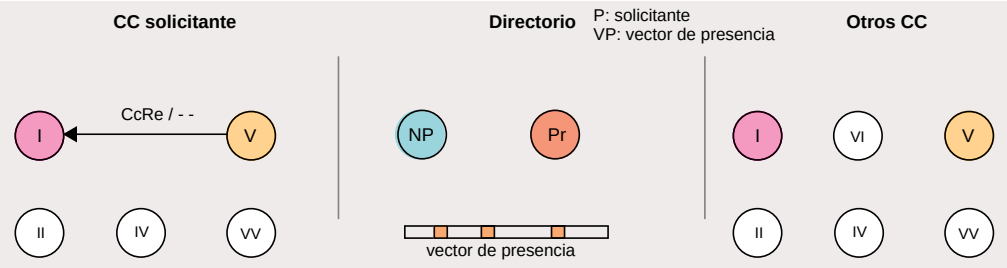
Protocolo de directorio VI. Expulsión silenciosa

Respecto de las expulsiones, debido a acciones de reemplazo en cache, podemos distinguir dos posibilidades: a) no silenciosa y b) silenciosa. Cuando se utiliza expulsión no silenciosa, el directorio recibe una notificación de la expulsión y un CC sólo recibe peticiones de observación de escritura de bloques almacenados en contenedores de la cache. Esto es, el directorio es preciso.

En una expulsión silenciosa el directorio no recibe notificación de la expulsión de un bloque. Esta característica determina que un CC pueda recibir una petición de observación de escritura (PtObE) de un bloque que no tiene almacenado en cache. Esto es, el directorio es impreciso. Por tanto, el agente observador, antes de invalidar el contenido de un contenedor, debe comprobar que el contenedor almacena el bloque al que hace referencia la petición de invalidación[23].

Pregunta 1: *En un protocolo de directorio VI, diseñe los autómatas de cambio de estado de un bloque en un CC y en el CM cuando se utiliza expulsión silenciosa.*

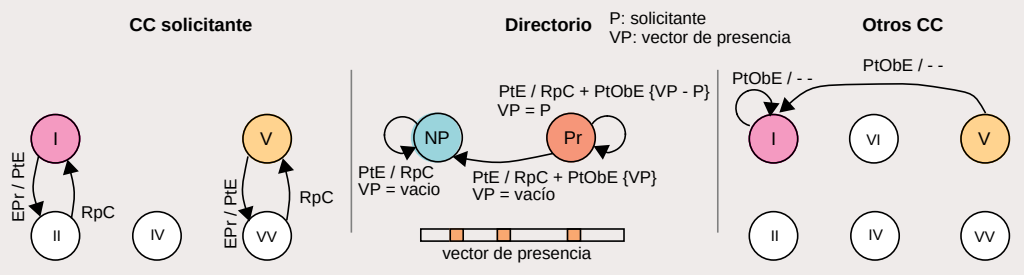
Respuesta: En la siguiente figura se muestran los diagramas de transiciones entre estados al efectuar una expulsión en un CC. Como la expulsión es silenciosa no se notifica al CM, sólo se produce un cambio de estado en el CC del cual se expulsa el bloque. Por tanto, el VP no se actualiza.



En estas condiciones un CC puede recibir una petición de observación de escritura cuando no almacena el bloque en cache. En los siguientes diagramas de transiciones entre estados de un bloque se representan las transiciones en una petición de escritura, que es la que determina que el CM efectúe una petición PtObE.

23. En este protocolo, si se invalida el bloque sin efectuar la comprobación se sigue manteniendo la coherencia. Recordemos que memoria siempre está actualizada.

Los diagramas de transiciones entre estados en el CC del solicitante y en el CM son los mismos que en la Figura 6.17 y la Figura 6.18. La diferencia está en el diagrama de otros CC.



Un bloque no presente en una cache se identifica que está en el estado I. Al utilizar expulsión silenciosa, un CC puede recibir una petición PtObE en el estado V y en el estado I. En cualquiera de los dos casos el estado final es I.

Por otro lado, el CM puede recibir una petición Pt de un CC que está identificado en el VP. En este caso el CM suministra el bloque. También, el CM puede recibir una petición PtE de un CC que está identificado en el VP, pero la cache no tiene copia del bloque. El CM actualiza la memoria e invalida las copias en otras caches. En el VP se sigue identificando al CC que ha efectuado la petición PtE, aunque no tiene copia del bloque.

Protocolo de directorio MLI

Una secuencia de accesos a memoria referencia las variables u y t que se ubican en bloques distintos. Todas las cache utilizan la misma organización y tienen el mismo tamaño. Los bloques que contienen las variable u y t al almacenarse en las caches se ubican en el mismo contenedor de cache.

El bloque que contiene la variable t está almacenado en la cache C1 y el bloque que contiene la variable u está almacenado en la cache C2. El estado de estos dos bloques en las caches es L.

Pregunta 1: *Utilice el protocolo de directorio con invalidación y escritura retardada y una tabla similar a la Tabla 6.12 para mostrar los mensajes de una transacción y estados de los bloques en las caches. La secuencia de accesos a memoria se indica en la tabla mostrada en la respuesta. Indique el número de expulsiones que se producen.*

Respuesta: Para realizar el primer acceso a memoria el CC del procesador P1 debe obtener la exclusividad en el acceso al bloque. Para ello el CC genera una petición de bloque con intención de modificación (PtIm). El segundo acceso a memoria es mimético al primer acceso, estando la diferencia en el procesador que efectúa el acceso y la variable accedida.

	C 1	C 2	Red	mem.			Red	Red		Red	C 1		C 2	
acceso	est.	est.	ida	var.	VP	E	RMC	RCM	sum.	vuelta	var.	est.	var.	est.
1. P1 store t	LM		PtIm	t	1, 0	1			mem.	RpD	t	M		
2. P2 store u		LM	PtIm	u	0, 1	1			mem.	RpD			u	M
3. P2 load t		MI	PtXm	u	0, 0	0			C2	RpX			u	I
		IL	Pt	t	1, 1	0	RpObL, 1	RpDc	C1	RpD	t	L	t	L
4. P2 store u		LI	PtXI	t	1, 0	0				RpX			t	I
		LM	PtIm	u	0, 1	1			mem.	RpD			u	M
5. P1 load t											t	L		
6. P2 load u													u	M

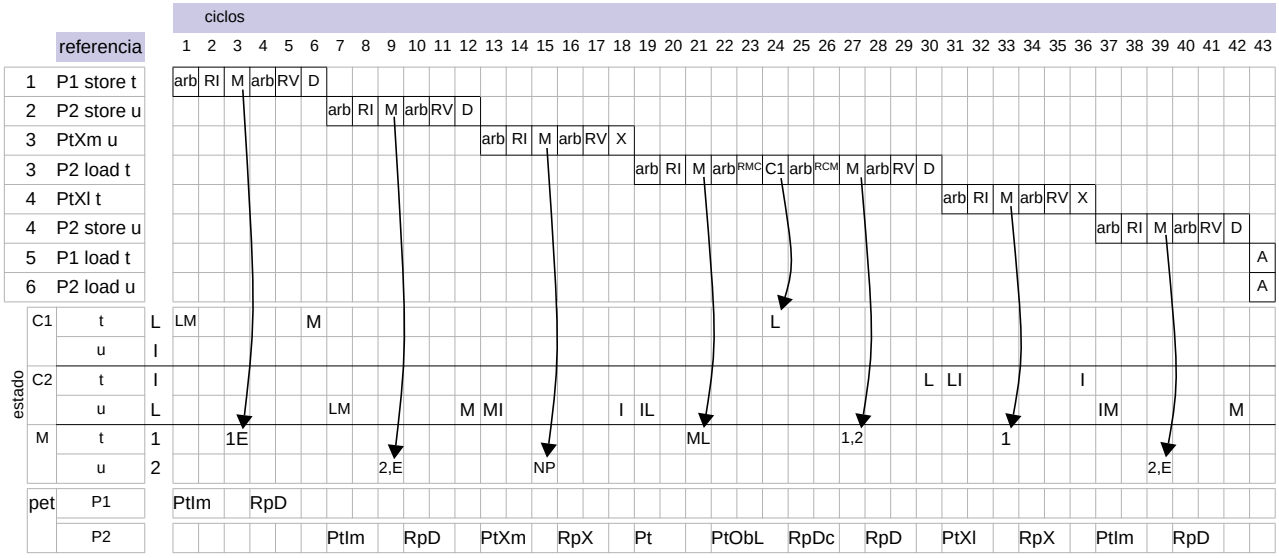
En el tercer acceso el CC2 detecta un fallo y es necesaria una acción de reemplazo. El bloque que se expulsa está en el estado M. Por tanto, es necesario actualizar la memoria (PtXm). Después de la transacción de actualización de memoria se efectúa la petición de bloque. El bloque solicitado está en estado M en otra cache. El CM solicita el bloque a la cache correspondiente y espera la recepción de una respuesta. Después de recibir la respuesta, el CM actualiza el directorio y emite una respuesta al CC que efectuó la petición.

El cuarto acceso es una escritura y el CM determina que es un fallo. La acción de reemplazo requiere expulsar un bloque en estado L. Después de expulsar el bloque, se inicia una transacción con una petición de bloque con intención de modificación. Al finalizar la transacción el estado del bloque es M. Los dos últimos accesos a memoria son operaciones de lectura que son aciertos en cache. En el primer caso el bloque está en estado L y en el segundo el bloque está en estado M.

En total se efectúan dos expulsiones. Una corresponde a un bloque en estado M (tercer acceso) y otra corresponde a un bloque en estado L (cuarto acceso). Las dos expulsiones se efectúan en la cache del procesador P2.

Pregunta 2: *Muestre un diagrama temporal de la secuencia de accesos. Utilice una fila para representar sólo la expulsión y en la siguiente fila represente la transacción correspondiente a la petición que ha inducido la expulsión.*

Respuesta: La siguiente figura muestra el diagrama temporal que se solicita. Recordemos que el acrónimo X en la última fase de una transacción indica la respuesta a una expulsión.



EJERCICIOS

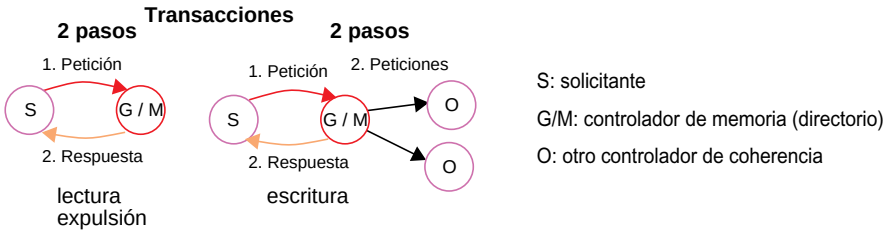
Descripción de un protocolo de directorio VI denominado A

Suponga un multiprocesador donde las caches privadas son de mapeo directo y utilizan escritura inmediata. Las redes de interconexión entre las caches y el módulo de memoria son de tipo crossbar y mantienen el orden de los mensajes emitidos. El multiprocesador utiliza un directorio para mantener la coherencia y el protocolo de coherencia es de invalidación (VI).

Las caches privadas de los procesadores son bloqueantes. En un fallo de cache o en una escritura se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

El directorio utiliza un vector de presencia (VP) por bloque. El vector de presencia es un vector de bits, con tantos bits como procesadores y cada bit está asociado a un procesador.

Las secuencias de mensajes de las transacciones son las siguientes:



Las peticiones de procesador y los mensajes utilizados en la transacciones para mantener la coherencia son:

Procesador	Controlador de cache (CC)		Controlador de memoria (CM)	
Peticiones	**Peticiones del CC al CM**	**Respuestas del CM al CC**	**Peticiones del CM a los CC**	**Acciones**
LPr : lectura	Pt : petición de bloque	RpD: respuesta con el bloque	PtObE: petición de observación de escritura	Actualización del directorio
EPr : escritura	PtE: petición de escritura de un dato	RpC: respuesta de confirmación		Dev: actualización de memoria
	PtX: petición de expulsión	RpX: respuesta de confirmación a una petición PtX		

El controlador de cache también efectúa acciones de reemplazo cuando es necesario (CcRe). En una acción de reemplazo se distingue la acción de notificación al directorio, ya que éste es preciso. En una petición PtX se actualiza el directorio.

Cuando el servicio de un acceso a memoria requiere un reemplazo, éste se efectúa antes de gestionar el acceso a memoria que produce la acción de reemplazo.

Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtE, PtX	arb	RI	M
RpD, RpC, RpX	arb	RV	D ó X
PtObE	arb	RMC	Cx

arb: arbitraje en la red correspondiente

RI: red de peticiones desde los CC al CM

RV: red de respuestas desde el CM a los CC

RMC: red de peticiones desde el CM a los CC

M: memoria (directorio)

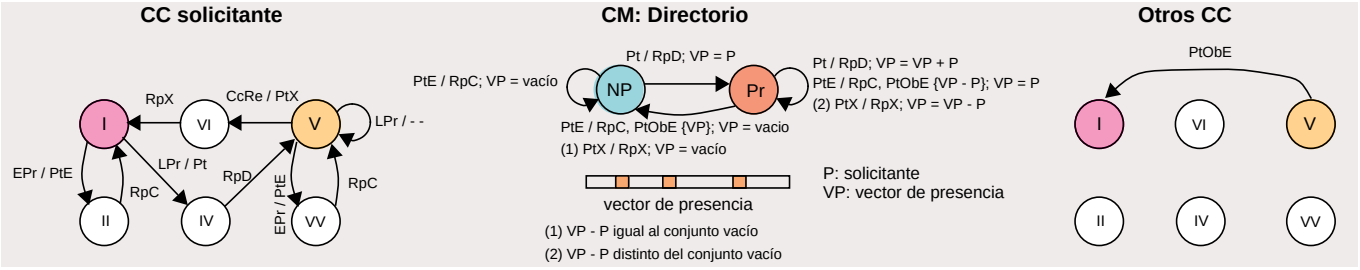
D: dato (RpD)

X: confirmación (RpX)

Cx: cache, donde x es el ordinal de la cache que recibe PtObE

En un CC, para distinguir, en una transacción, entre la emisión de un mensaje de petición y la recepción de una respuesta, se utilizan estados transitorios (II, IV, VV, VI). En el CM no es necesario ya que no espera respuestas.

En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio.



En este multiprocesador supondremos que sólo existe un acceso a memoria en un instante determinado.

En un diagrama temporal se muestran en la parte superior las fases de los mensajes de una transacción, en la parte central se especifica el estado de los bloques en las cache y en el directorio. En la parte inferior se etiqueta el mensaje o la respuesta que se representa en la parte superior. El estado de los bloques en cache o en el directorio se indica sólo cuando hay un cambio de estado.

Fases y eventos	Especificaciones
arb	Se especifica el estado transitorio del bloque.
M	Se especifica el VP utilizando el ordinal de los procesadores cuyas caches tienen copia del bloque.
D ó X	Se especifica el estado estable del bloque en la cache cuyo CC ha efectuado la petición.
x	Se especifica la modificación de estado determinada por la petición del CM
Transacción de 2 pasos con petiiciones del CM	Se utiliza una fila para indicar la respuesta del CM y otra fila para indicar todas las peticiones del CM.
Reemplazo	Determina una expulsión: se especifica en la columna etiquetada como referencia. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtX) y en la segunda fila la petición que determina la expulsión.
Mensaje	Se indica en la columna correspondiente a arb.

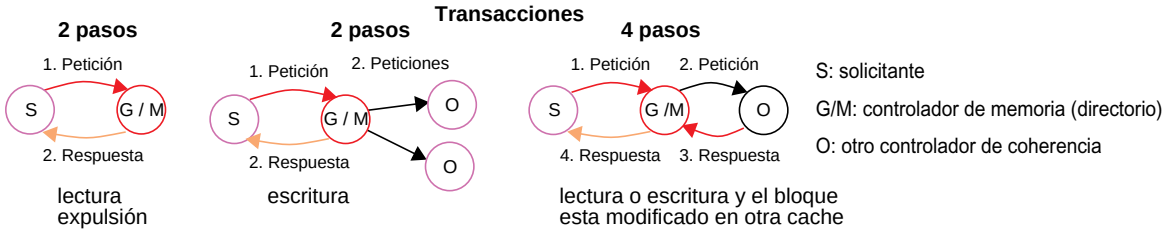
Descripción de un protocolo de directorio MLI denominado B

Suponga un multiprocesador donde las caches privadas son de mapeo directo y utilizan escritura retardada. Las redes de interconexión entre las caches y el módulo de memoria son de tipo crossbar y mantienen el orden de los mensajes emitidos. El multiprocesador utiliza un directorio para mantener la coherencia y el protocolo de coherencia es de invalidación (MLI).

Las caches privadas de los procesadores son bloqueantes. En un fallo de cache o en una solicitud de exclusividad se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

El directorio utiliza un vector de presencia (VP) y un bit de exclusividad (BE) por bloque. El vector de presencia es un vector de bits, con tantos bits como procesadores y cada bit está asociado a un procesador. El bit de exclusividad se utiliza para indicar que sólo existe una copia del bloque en una cache privada, la cual está identificada en el vector de presencia.

Las secuencias de mensajes de las transacciones son las siguientes:



Las peticiones de procesador y los mensajes utilizados en la transacciones para mantener la coherencia son:

Procesador	Controlador de cache (CC)		Controlador de memoria (CM)		
Peticiones	**Peticiones del CC al CM**	**Respuestas del CM al CC**	**Peticiones del CM a los CC**	**Respuestas del CC al CM**	**Acciones**
LPr : lectura	Pt : petición de bloque	RpD: respuesta con el bloque a una petición Pt o PtIm	PtObE: petición de observación de escritura, inducida por una petición PtIm	RpDc: respuesta con el boque a una petición PtObL o PtObE y el estado del bloque en cache es M	Actualización del directorio
EPr : escritura	PtIm: petición de bloque con intención de modificarlo	RpX: respuesta de confirmación a una petición PtXm o PtXl	PtObL: petición de observación de lectura, inducida por una petición Pt y el estado del bloque en el directorio es M		Dev: actualización de memoria
	PtXm: petición de expulsión de un bloque en estado M				
	PtXl: petición (notificación) de expulsión de un bloque en estado L				

El controlador de cache también efectúa acciones de reemplazo cuando es necesario (CcRe). En una acción de reemplazo se distingue la acción de notificación al directorio, ya que éste es preciso y si es el caso, una actualización de memoria con el bloque expulsado, si éste ha sido modificado durante su estancia en la cache. En una petición PtXm se actualiza el directorio y memoria, mientras que en una petición PtXl sólo se actualiza el directorio.

Cuando el servicio de un acceso a memoria requiere un reemplazo, éste se efectúa antes de gestionar el acceso a memoria que produce la acción de reemplazo.

Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtIm, PtXm, PXl	arb	RI	M
RpD, RpX	arb	RV	D ó X
PtObE, PtObL	arb	RMC	Cx
RpDc	arb	RCM	M

arb: arbitraje en la red correspondiente

RI: red de peticiones desde los CC al CM

RV: red de respuestas desde el CM a los CC

RMC: red de peticiones desde el CM a los CC

RCM: red de respuestas de los CC al CM

M: memoria (directorio)

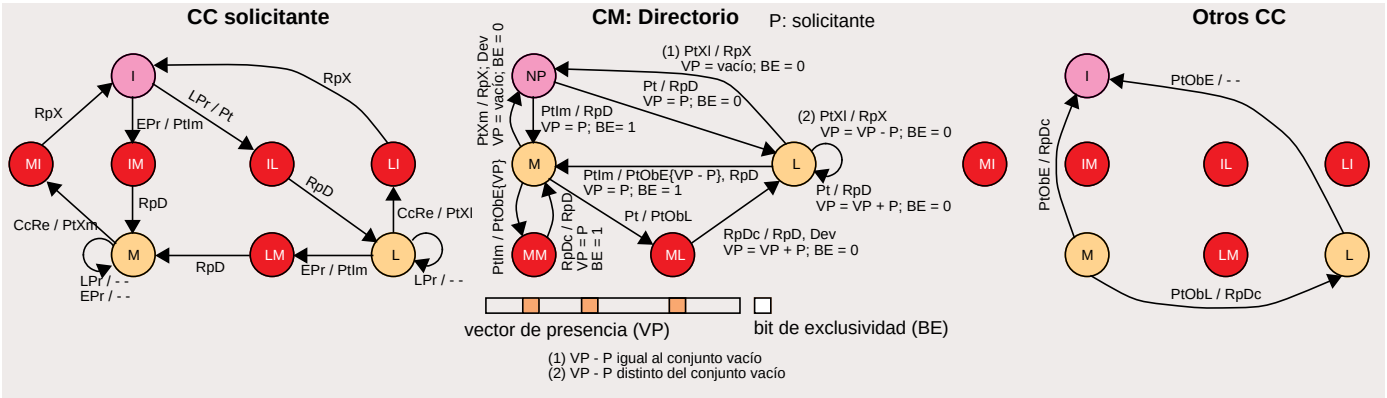
D: dato (RpD)

X: confirmación (RpX)

Cx: cache, donde x es el ordinal de la cache

En un CC, para distinguir, en una transacción, entre la emisión de un mensaje de petición y la recepción de una respuesta, se utilizan estados transitorios (IL, LM, IM, LI, MI). En el CM para distinguir entre la emisión de un mensaje de petición, correspondiente a una transacción que está procesando el CM, y la respuesta de un CC se utilizan estados transitorios (ML, MM).

En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio. En el protocolo que se describe, el bit de exclusividad del directorio se activa cuando una cache solicita el bloque para actualizarlo.



En este multiprocesador supondremos que sólo existe un acceso a memoria en un instante determinado.

En un diagrama temporal se muestran en la parte superior las fases de los mensajes de una transacción, en la parte central se especifica el estado de los bloques en las caches y en el directorio. En la parte inferior se etiqueta el mensaje o la respuesta que se representa en la parte superior. El estado de los bloques en cache o en el directorio se indica sólo cuando hay un cambio de estado.

Fases y eventos	Especificaciones
arb	Se especifica el estado transitorio del bloque.
M	Se especifica el VP utilizando el ordinal de los procesadores cuyas caches tienen copia del bloque y si el bloque lo tiene una cache en exclusividad se añade la letra E. La especificación se efectúa la última vez que se visita el directorio en una transacción.
D ó X	Se especifica el estado estable del bloque en la cache cuyo CC ha efectuado la petición.
Cx	Se especifica la modificación de estado determinada por la petición del CM
Transacción de 2 pasos	Se utiliza una fila para indicar la respuesta del CM y otra fila para indicar todas las peticiones del CM.
Reemplazo	Determina una expulsión: se especifica en la columna etiquetada como referencia. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtXm o PtXl) y en la segunda fila la petición que determina la expulsión.
Mensaje	Se indica en la columna correspondiente a arb.

Ejercicio 6.1

Utilice el protocolo de directorio MLI denominado B descrito en este capítulo. Suponga la siguiente secuencia de accesos.

accesos	
1. P1 load t	5. P2 store u
2. P1 store t	6. P1 load t
3. P2 store u	7. P2 load u
4. P2 load t	

Las variables u y t se ubican en bloques distintos.

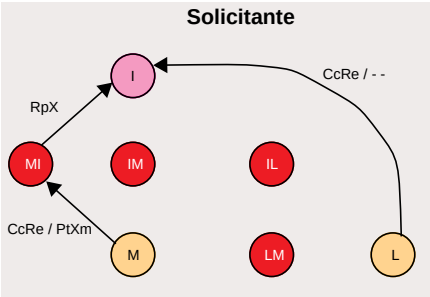
Estos bloques al almacenarse en cache se ubican en el mismo contenedor.

La variable t no está inicialmente almacenada en ninguna cache y el valor en memoria es t = 12.

La variable u está almacenada inicialmente en la cache de P2, en el estado L y el valor es u = 9.

Pregunta 1: *Muestre en un diagrama temporal la secuencia de mensajes que genera cada transacción y los cambios de estado de los bloques en las caches y en el directorio al ejecutarse la anterior secuencia de accesos a memoria.*

Un diseñador observa que el número de expulsiones de bloques en estado L es mucho mayor que el número de bloques expulsados en estado M. En consecuencia, para reducir el tráfico en las redes de interconexión decide no notificar al directorio la expulsión de un bloque en estado L. Esta característica se denomina expulsión silenciosa y el directorio se denomina impreciso. Esto es, el directorio no indica de forma precisa las copias del bloque en las cache privadas y siempre indica un conjunto mayor de CC que tienen copia que el que realmente existe. En contraposición, el directorio denominado B dice que es preciso.



Pregunta 2: *En un directorio impreciso, un CC puede recibir una petición de observación de escritura (PtObE) de un bloque que no tiene almacenado en cache. Enumere los estados en los cuales un CC puede recibir una petición PtObE del CM e indique la respuesta en cada caso.*

Pregunta 3: *Para caches privadas de mapeo directo o de mapeo asociativo por conjuntos y los dos posibles tipos de directorio (preciso o impreciso) indique si es necesario comprobar el contenido de la cache (comparar etiquetas) cuando se recibe una petición del CM.*

Pregunta 4: *Justifique si el CM puede recibir por parte de un CC, que tiene un bloque en el estado I, una petición de este bloque y el directorio no tiene identificado al CC en el VP.*

Ejercicio 6.2

Utilice el protocolo de directorio MLI denominado B descrito en este capítulo. Suponga la siguiente secuencia de accesos.

accesos	
1. P1 load t	5. P2 store u
2. P1 store v	6. P1 load t
3. P2 store u	7. P2 load u
4. P2 load t	

Las variables u y t se ubican en bloques distintos. Las variables t y v se ubican en el mismo bloque

Estos bloques al almacenarse en cache se ubican en el mismo contenedor.

La variable u no está inicialmente almacenada en ninguna cache y el valor en memoria es u = 12.

El bloque que contiene las variables t y v está almacenado inicialmente en la cache de P2, en el estado L y los valores son t = 9 y v =125.

Pregunta 1: *Muestre en un diagrama temporal la secuencia de mensajes que genera cada transacción y los cambios de estado de los bloques en las caches y en el directorio al ejecutarse la anterior secuencia de accesos a memoria.*

Un diseñador observa que en una secuencia load A - store A, efectuada por un procesador, no siendo en este intervalo de tiempo una variable compartida, se requieren dos transacciones. La primera transacción para obtener el bloque y ejecutar la instrucción load y la segunda transacción para obtener la exclusividad y ejecutar la instrucción store.

Protocolo modificado. El diseñador propone que, si no existen copias del bloque en las caches privadas, el controlador de memoria (CM), al recibir una petición Pt, suministre el bloque en exclusividad. Para ello, añade en el protocolo una nueva respuesta en el CM, denominada RpDe. El CM al emitir una respuesta RpDe, activa el bit correspondiente del procesador en el VP y también activa el BE.

Un controlador de coherencia (CC) al recibir la respuesta RpDe, a una petición Pt, determina que el bloque ha sido suministrado en exclusividad y establece el estado en consecuencia.

Pregunta 2: *¿Cuál es la transición entre estados que se efectúa en el directorio al responder con RpDe a una petición Pt?. ¿Cuál es la transición entre estados que se efectúa en el CC al emitir la petición Pt y recibir una respuesta RpDe?.*

Para analizar el rendimiento se utiliza la siguiente secuencia de accesos a memoria.

accesos
1. P1 load t
2. P1 load u
3. P1 store u
4. P1 load t

Las variables u y t se ubican en bloques distintos.

Estos bloques al almacenarse en cache se ubican en el mismo contenedor.

Las variables u y t no están inicialmente almacenada en ninguna cache

Pregunta 3: *Indique la secuencia de mensajes de peticiones y respuestas necesaria para servir cada petición en el protocolo original y en el protocolo modificado. Cuando, para servir un acceso a memoria, sea necesario expulsar un bloque, utilice dos filas consecutivas. En la primera fila especifique la expulsión y en la segunda fila especifique el acceso a memoria.*

Supongamos que la transmisión del bloque desde la cache al CM ocupa durante varios ciclos la red de ida (RI). Entonces, en el protocolo original y en el modificado, considere las siguientes latencias en las peticiones PtXm y PtXl.

	ciclos					
mensajes	1	2	3	4	5	6
PtXm	arb	RI	RI	RI	RI	M
PXl	arb	RI	M			

Pregunta 4: *Para la secuencia de accesos a memoria anterior, calcule la duración en ciclos utilizando el protocolo original y utilizando el protocolo modificado.*

Ejercicio 6.3

En el protocolo de directorio MLI denominado B descrito en este capítulo, un diseñador analiza una secuencia load A - load B, efectuada por un procesador. Las variables A y B no están siendo compartidas y están ubicadas en bloques distintos. Sin embargo, los bloques que contienen estas variables, cuando se mapean en cache, se almacenan en el mismo contenedor. En una modificación previa del protocolo A, al expulsar el bloque que contiene la variable A se actualiza memoria (ejercicio 6.2). En este nuevo diseño, el ingeniero pretende que al expulsar el bloque, que contiene la variable A, sea suficiente con actualizar el directorio. Esto es, la memoria no se actualiza. Otro ejemplo es la expulsión de bloques que contienen instrucciones no compartidas. Instrucciones que son sólo accedidas desde un procesador.

Protocolo modificado. El diseñador añade el estado E al conjunto de estados de un bloque en las caches privadas. El estado E identifica que es la única copia del bloque en una cache privada y que se tiene acceso al bloque en exclusividad.

Si no existen copias del bloque en las caches privadas, el controlador de memoria (CM), al recibir una petición Pt, suministra el bloque en exclusividad. Para este caso, el protocolo utiliza una nueva respuesta en el CM, denominada RpDe. El CM al emitir una respuesta RpDe, activa el bit correspondiente del procesador en el VP y también activa el BE.

Un controlador de coherencia (CC) al recibir la respuesta RpDe, a una petición Pt, determina que el bloque ha sido suministrado en exclusividad y establece el estado en consecuencia.

Además, el diseñador modifica el autómata en el directorio para procesar una petición PtXl cuando un bloque, en el directorio, está en estado M.

Pregunta 1: *¿Cuál es la transición entre estados que se efectúa en el directorio al responder con RpDe a una petición Pt?. ¿Cuál es la transición entre estados que se efectúa en el CC al emitir la petición Pt y recibir una respuesta RpDe?.*

Pregunta 2: *Suponga que no existe ningún acceso a la misma posición de memoria por parte de otro procesador. Entonces, el mismo procesador que ha emitido una petición Pt efectúa una escritura a una posición del bloque que ha*

recibido con la respuesta RpDe. ¿Cuál es la transición entre estados, y, si es el caso, la petición al directorio, que efectúa el CC cuando recibe una escritura del procesador (EPr)?.

En las siguientes dos preguntas especifique sólo las transiciones adicionales, si existen, que determinan la modificación del protocolo.

Pregunta 3: *En un grafo de estados muestre las transiciones en un CC al recibir del CM una petición PtObE o una petición PtObL.*

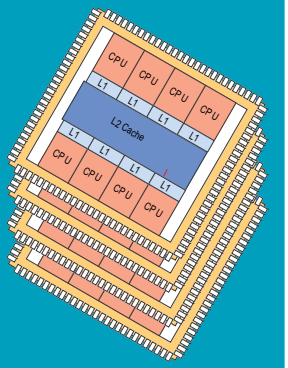
Pregunta 4: *Especifique las transiciones entre los estados de un bloque en el directorio y respuestas del CM al recibir una petición de expulsión. Especifique también las transiciones en el CC.*

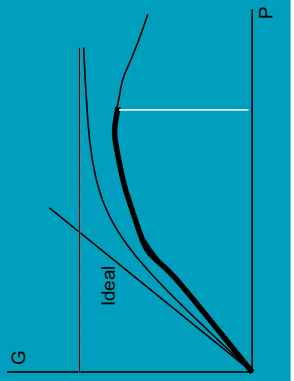
Suponga la siguiente secuencia de accesos a memoria. Las variables t, u y s están ubicadas en bloques distintos. Las variables t y w están ubicadas en el mismo bloque. Al mapearse en cache los tres bloque utilizan el mismo contenedor e inicialmente no están almacenados en ninguna cache.

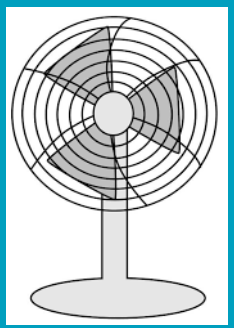
accesos	1. P1 load t	2. P2 load u	3. P1 store w	4. P2 load u	5. P3 store u	6. P4 load s	7. P4 load u

Pregunta 5: *Indique la secuencia de mensajes de peticiones y respuestas necesaria para servir cada petición en el protocolo original (A) y en el protocolo modificado. Cuando, para servir un acceso a memoria, sea necesario expulsar un bloque, utilice dos filas consecutivas. En la primera fila especifique la expulsión y en la segunda fila especifique el acceso a memoria.*

Pregunta 6: *Para la secuencia de accesos a memoria anterior, calcule la duración en ciclos utilizando el protocolo original y el protocolo modificado. En las peticiones PtXm y Ptxl considere las latencias indicadas en el ejercicio 6.2.*







Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 7
Concurrencia y paralelismo en un protocolo de directorio con una red ordenada

El diseño descrito en el Capítulo 6 utiliza como hipótesis que, en un ciclo determinado, sólo existe un acceso a memoria en el multiprocesador. En este capítulo se incrementa la concurrencia permitiendo que todos los procesadores efectúen accesos a memoria en paralelo.

Como existe concurrencia un CC puede recibir una petición del CM y tener pendiente una petición al mismo bloque u otro bloque. De forma similar, el CM al recibir una petición puede estar procesando una petición previa, de otro CC, al mismo bloque u otro bloque. Por otro lado, el CM puede necesitar procesar peticiones absoletas teniendo en cuenta el vector de presencia o el estado actual del bloque.

Finalmente, para incrementar el número de transacciones por unidad de tiempo se utiliza la técnica de paralelismo. Para ello se dispone de varios módulos de memoria, cada uno con su CM. Cada módulo de memoria puede ser accedido de forma independiente y el directorio está distribuido entre los CM. En consecuencia, se pueden estar sirviendo tantas peticiones de los CC como módulos de memoria.

CONCURRENCIA

Suponemos un único módulo de memoria con el CM correspondiente. El CM procesa concurrentemente peticiones a bloques distintos y serializa peticiones al mismo bloque.

Hipótesis en la descripción del protocolo

Un procesador interpreta las instrucciones en orden de programa y se bloquea al detectar un riesgo y en un acceso a memoria que requiere acceder al directorio.

Los procesadores realizan peticiones concurrentes al sistema de memoria.

Cuando una transacción requiere un reemplazo, en primer lugar se efectúa la expulsión del bloque. Una vez finalizada, se efectúa la petición del bloque que determina la expulsión[1].

El CM puede estar procesando peticiones concurrentemente a bloques distintos.

Redes de interconexión

Hay un árbitro en la RI. La red se construye mediante conexiones punto a punto y se encola una petición por ciclo en la CP. El CM procesa las peticiones de la CP en orden de llegada (FIFO).

En la Figura 7.1 se muestra un esquema del multiplexor que hay en la entrada de la CP y un diagrama temporal con la acción de arbitraje. Dos CC solicitan encolar una petición en el mismo ciclo. El árbitro (arb) ordena las peticiones. La espera de una de las peticiones para acceder a la red se indica representando en ciclo consecutivos la fase de arbitraje (arb, parte derecha de la Figura 7.1). En este ejemplo, se supone que el acceso a memoria ocupa dos ciclos. Por tanto, la latencia de iniciación es dos ciclos. En consecuencia, peticiones concurrentes de acceso a memoria se encolan en la CP.

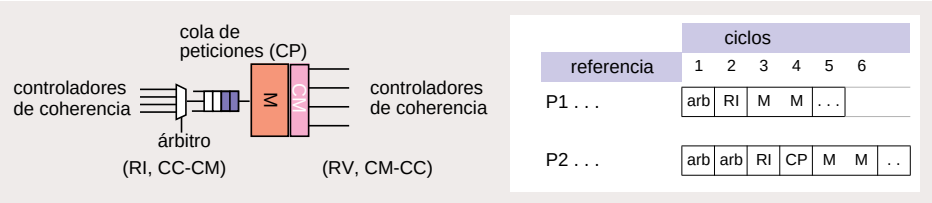


Figura 7.1 Red de peticiones de los CC al CM (RI) y red de peticiones y respuesta del CM a los CC (RV).

1. Si se utiliza un buffer de expulsiones (BEX), el contenedor de cache queda libre y se pueden emitir las dos peticiones en secuencia, pero en orden inverso. Si sólo hay un CM se procesan en serie, aunque concurrentemente. Si hay varios CM se pueden procesar en paralelo cuando el bloque expulsado y el solicitado no se almacenan en el mismo módulo de memoria. El procesador reanuda la ejecución al llegar la respuesta a la petición del bloque. La petición debida a la expulsión queda en la sombra. Si una nueva referencia del procesador accede al bloque que se expulsa, al considerarse que está en cache (BEX), puede bloquearse o servirse como acierto, en función de la circuitería disponible, del estado y de las peticiones que se hayan procesado mientras se espera la respuesta del CM.

En ocasiones el CM, al procesar una petición de un CC, necesita emitir peticiones y una respuesta a los CC. El conjunto de CC destinatarios de las peticiones y el CC destinatario de la respuesta es disjunto. Entonces, las peticiones y la respuesta se transmiten utilizando la red de vuelta (RV) y en paralelo[2].

La RV se implementa mediante conexiones punto a punto entre el CM y los CC[3]. Esta red garantiza que los mensajes (peticiones y respuestas) emitidos por el CM llegan a un CC en el mismo orden que son emitidos. Esto es, además de que se mantiene el orden entre peticiones y se mantiene el orden entre respuestas, también se mantienen el orden entre una petición y una respuesta y viceversa.

Coherencia. El CM es el punto de ordenación de las transacciones iniciadas por los CC. El CM propaga una transacción de escritura de un bloque a los CC. La red RV mantiene el orden de emisión de los mensajes del CM a los CC. Por tanto, todos los CC observan las peticiones de observación de escritura del CM en el mismo orden lógico y no es necesario que los CC respondan a las peticiones de invalidación. Una escritura está consolidada cuando el CM emite la respuesta al CC que ha iniciado la transacción. El CM serializa las transacciones a un mismo bloque[4].

Consistencia. El CM y la transmisión en orden de los mensajes en la RV determinan un orden lógico global de las transacciones. Una escritura está consolidada cuando el CC recibe la respuesta del CM. Una petición de lectura lee el valor establecido por la instrucción store previa a la misma dirección.

Concurrencia en el controlador de memoria

El CM siempre responde a una petición de un CC. Por otro lado, un CC también responde a las peticiones del CM, si es el caso. En particular, un CC dispone del bloque que quiere expulsar hasta que se recibe la respuesta del CM a la petición de expulsión[5].

La llegada de una nueva petición en el CM, antes de haber inyectado en la RV los mensajes de coherencia, correspondientes a una petición anterior, sólo requiere capacidad de almacenamiento en la entrada del controlador de memoria.

2. Por ejemplo, en el protocolo de directorio VI.
3. En el protocolo MLI la RV implementa las redes lógicas RMC y RV (Capítulo 6).
4. El CM no procesa otra petición a un bloque hasta que ha emitido la respuesta a la petición previa que accede al mismo bloque.
5. Esta característica facilita la gestión de un cruce de peticiones, denominado tardío, en el CC.

La CP se utiliza para almacenar peticiones mientras el CM está procesando una petición. Supondremos que se dispone de la capacidad de almacenamiento suficiente para el peor caso[6]. De esta forma no es necesario entrar en detalles de control de flujo de mensajes desde los CC a un CM.

En la Figura 7.2 se muestra la concurrencia en el procesado de peticiones en el CM. Las dos primera peticiones de los CC se efectúan en el mismo ciclo y hay que arbitrar. La segunda petición necesita esperar en la CP para ser procesada. Esta petición se procesa cuando el CM finaliza el procesado de la petición anterior. El procesado de la tercera petición también debe esperar en la CP.

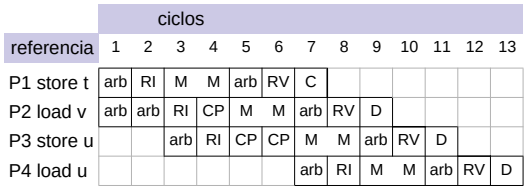


Figura 7.2 Concurrencia en el directorio. El acrónimo CP indica cola de peticiones pendientes.

Típicamente, en los diagramas temporales, para mostrar de forma simple las transacciones, supondremos que la latencia de cada fase es un ciclo. Esto nos permite suponer una latencia de iniciación igual a uno a partir de la cola de peticiones.

El CM, en función del protocolo, tiene una cola de respuestas de los CC[7]. El procesado de mensajes en esta cola es prioritario respecto de la cola CP.

Transacciones no atómicas

Cuando transacciones concurrentes referencian bloques distintos, no existe interacción entre ellas[8]. Sin embargo, si las transacciones referencian el mismo bloque, es posible que tanto el CM como los CC tengan que procesar una petición al bloque, de un CC o del CM respectivamente, cuando tienen pendiente de finalizar o completar una transacción al mismo bloque. En estas condiciones una transacción no es atómica.

6. Los procesadores ejecutan las instrucciones en orden de programa y un procesador se bloquea en un acceso al CM. Entonces, el número de entradas necesarias en la CP es el número de procesadores. Si los CC utilizan un BEX hay que tener en cuenta el número de entradas en el BEX.
7. Protocolos donde una cache puede tener el bloque en exclusividad o ser la encargada de suministrar el bloque aunque no tenga la exclusividad.
8. La interacción puede ser por ocupación de recursos hardware.

Cruce de peticiones al mismo bloque. Entre el CM y un CC se produce un cruce de peticiones al mismo bloque cuando un CC, que está incluido en el VP, emite una petición al CM y a partir de este instante, el CM efectúa una o varias peticiones al CC, inducidas por una o varias peticiones de otros CC, antes de procesar la petición del CC.

En la Figura 7.3 se muestran dos ejemplos de cruce de peticiones. En la parte izquierda de la figura, protocolo VI, el CC1 debe procesar una petición del CM en un estado transitorio[9]. También, el CM debe procesar una petición no esperada (PtX), ya que el VP no contiene el identificador del CC1 que la ha emitido. Observemos que la petición que ha procesado previamente el CM, al mismo bloque (PtE), desactiva el bit del CC1 en el VP.

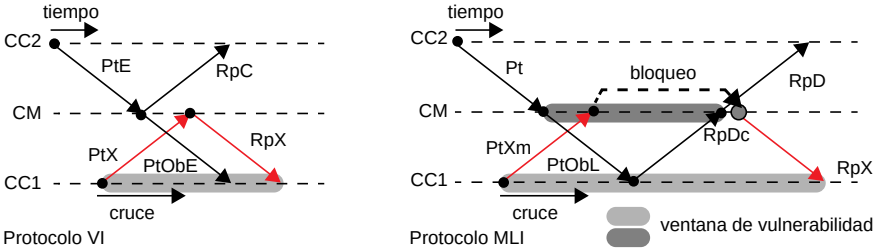


Figura 7.3 Ejemplos de cruce de peticiones.

En la parte derecha de la figura, protocolo MLI, el CC1 debe procesar una petición del CM en un estado transitorio y responder al CM para que éste finalice la transacción. Por otro lado, el CM puede necesitar procesar una petición de un CC en un estado transitorio. En el caso mostrado se pospone el procesado de la petición del CC1 en el CM hasta que el bloque en el CM esté en un estado estable[10].

Gestión de un cruce en un CC. Entre la emisión de una petición por un CC y la recepción de la respuesta del CM puede ser necesario procesar alguna petición del CM al mismo bloque en el CC (ventana de vulnerabilidad).

La RV mantiene el orden de los mensajes emitidos desde el CM a los CC. Por ejemplo, el CM ordena en primer lugar una petición del CC2 respecto de una petición del CC1. La petición del CM al CC1, inducida por la petición del CC2, es recibida por el CC1 antes que la respuesta del CM a su petición. En particular, la petición y la respuesta pueden referenciar el mismo bloque. Esto es, la RV garantiza que un CC recibe las peticiones y respuestas del CM en el orden en que han sido emitidas desde el CM. Por tanto, un CC al recibir una petición

9. En el CC debe ser procesada, ya que el CM ha ordenado esta petición antes que la del CC1.
10. En el desarrollo del capítulo se pospone el procesado en el CM de la petición del CC que está en la cabeza de la CP y de las siguientes peticiones almacenas en la CP. El procesado se reanuda cuando finaliza la transacción en curso. En los ejercicios se analizan casos donde es factible que el CM siga procesando peticiones de la CP.

del CM, a un bloque en un estado transitorio, infiere que el CM ha procesado la petición de otro CC, al mismo bloque, antes que su petición y responde, si es el caso (Figura 7.4).

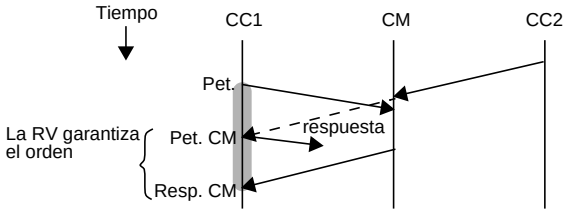


Figura 7.4 Cruce de peticiones.

En general, un CC pueden recibir varias peticiones mientras un bloque está en un estado transitorio. En consecuencia, usualmente, es necesario identificar el orden de recepción, para tenerlo en cuenta en las respuestas de las sucesivas peticiones (Figura 7.5).

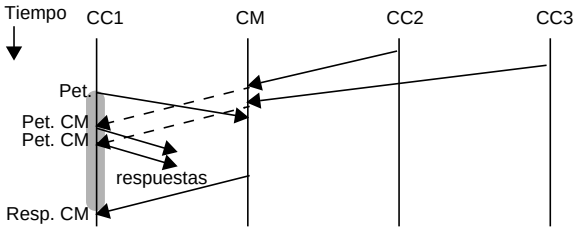


Figura 7.5 Cruce de varias peticiones.

En el diagrama de estados de un CC hay que identificar los casos en que el CM ha ordenado previamente peticiones de otros CC. Para ello se utilizan estados transitorios adicionales. Por otro lado, el CC debe responder teniendo en cuenta que el CM ha ordenado previamente la petición de otro CC, la cual ha inducido la petición que recibe del CM.

En la Figura 7.6 se muestra un ejemplo donde un CC puede recibir dos peticiones distintas en el estado transitorio INF. Las peticiones son P1 y P2. El CC puede recibir la petición P2 o puede recibir la secuencia P1, P2. En el ejemplo, si el CC recibe una petición P2, en el estado transitorio INF, cambia al estado INF1. Si recibe una petición P1 cambia al estado INF2 para identificar la secuencia. Una posibilidad menos ortodoxa, para reconocer las secuencias de peticiones, es la mostrada en la parte derecha de la figura[11]. En ella se está suponiendo implícitamente que no se pueden recibir peticiones de tipo P2 antes de P1 y que después de P1 sólo puede recibirse una petición de tipo P2. Además, se supone que el estado final es el mismo.

11. Observemos que reconoce otras secuencias de peticiones del CM, además de las esperadas.

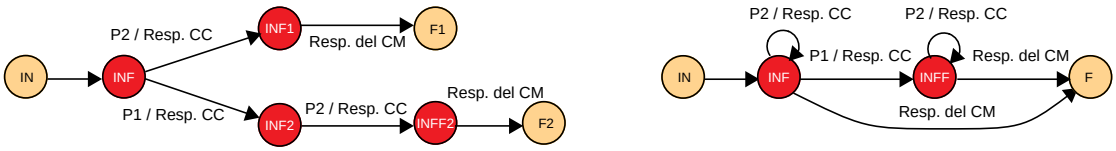


Figura 7.6 Estados transitorios en un CC para identificar peticiones del CM mientras un bloque está en un estado transitorio.

La respuesta del CC al CM en un estado transitorio tiene que estar en consonancia con el estado estable del que se proviene y con la secuencia de peticiones previas del CM que han sido respondidas[12].

Gestión de un cruce en el CM. Cuando el CM gestiona una acción de coherencia, puede emitir una petición a un CC que requiere una respuesta de este (por ejemplo, en el protocolo de directorio MLI). En el lapso de tiempo entre la petición del CM y la respuesta del CC, el CM analiza la cabeza de la CP. La petición en la cabeza de la CP puede referenciar el mismo bloque (ventana de vulnerabilidad). En este caso se bloquea el procesado de la petición y el análisis de las peticiones que hay encoladas en la CP (Figura 7.7).

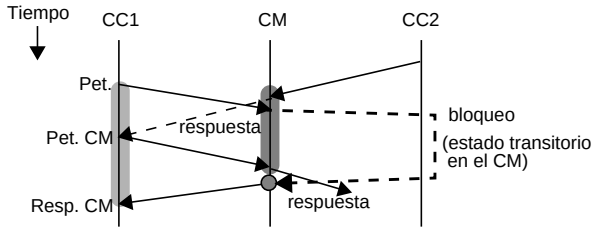


Figura 7.7 Bloqueo del procesado de peticiones de los CC en el CM cuando el bloque está en un estado transitorio.

En estas condiciones, en el CM sólo se procesan peticiones en estados estables. Un CC siempre responde a una petición del CM. Entonces, el CM reanuda el procesado de peticiones al recibir la respuesta del CC.

La petición de la cabeza de la CP, que ha bloqueado el procesado en el CM, es factible que al procesarse en un estado estable: a) sea un tipo de petición en la que se espera que el CC que la emite esté identificado en el VP y no sea así o b) sea necesario procesar la petición del CC en un estado que no es el esperado[13]. En el ejemplo de la parte derecha de la Figura 7.3, el CM debe procesar la petición PtXm en un estado estable, en la que no es esperada[14].

12. Recordemos que la RV mantiene el orden de los mensajes emitidos desde el CM a los CC.
13. Esto es, en el estado actual del bloque en el CM no se espera este tipo de petición de un CC. Recordemos que un CC mantiene el bloque que expulsa hasta que no recibe la respuesta del CM.

PROTOCOLO DE DIRECTORIO VI

La descripción del protocolo de directorio VI se ha efectuado en el Capítulo 4. En cuanto al camino de datos es el descrito en el mismo capítulo.

En la Figura 7.8 se muestran por completitud (Figura 6.5) los mensajes en los dos tipos de transacciones del protocolo de coherencia.

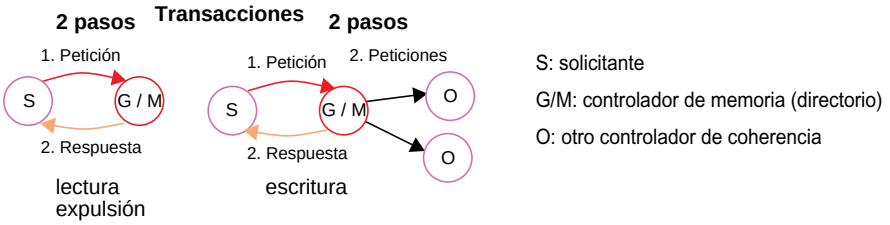


Figura 7.8 Protocolo de directorio VI. Mensajes en los dos tipos de transacciones.

Organización del multiprocesador

En la Figura 7.9 se muestra la organización del multiprocesador. En ella se distingue la cola de peticiones de los CC al CM. También se observa el encaminamiento de los mensajes de los CC al CM y del CM a los CC.

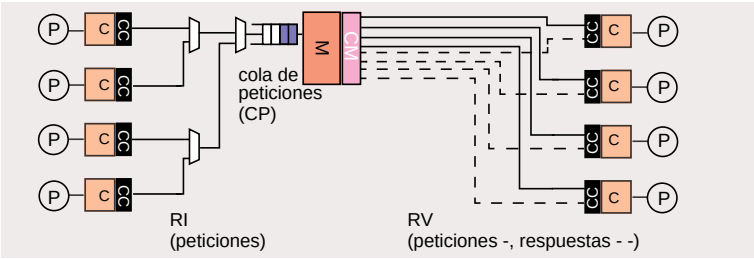


Figura 7.9 Protocolo de directorio VI. Organización del multiprocesador.

En la Figura 7.9, aunque la RV es única, se distinguen conexiones punto a punto para identificar que se transmiten peticiones y respuestas.

14. Estado L en lugar de estado M en el CM.

Transiciones entre estados estables de los controladores

En la Figura 7.10 se muestran por completitud los diagramas de transiciones entre estados de un bloque en el CM y en un CC (Figura 6.17 y Figura 6.18).

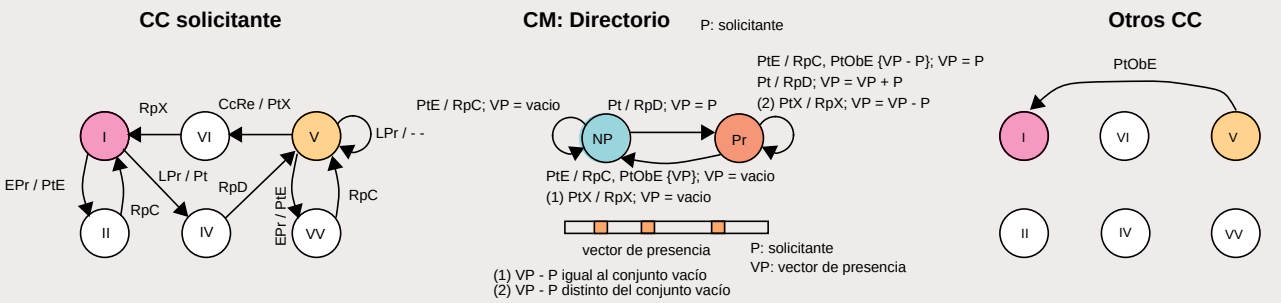


Figura 7.10 Protocolo VI. Transiciones entre estados en el CC y el CM.

Concurrencia en el controlador de memoria

El CM para procesar una petición sólo requiere acceder una vez al directorio. Con la información leída del directorio emite la respuesta y peticiones necesarias, si es el caso (Figura 7.10).

Peticiones concurrente a bloques distintos

En la Figura 7.11 se muestra la concurrencia en el procesado de peticiones en el CM. Las tres primeras peticiones se efectúan en el mismo ciclo y hay que arbitrar. El arbitraje determina el secuenciamiento en el CM.

	ciclos												
referencia	1	2	3	4	5	6	7	8	9	10	11	12	13
P1 store t	arb	RI	M	arb	RV	C							
P2 load v	arb	arb	RI	M	arb	RV	D						
P3 store u	arb	arb	arb	RI	M	arb	RV	C					
P4 load r							arb	RI	CP	M	arb	RV	D

Figura 7.11 Protocolo VI. Concurrencia en el CM. Las peticiones referencian bloques distintos. El acrónimo CP indica cola de pendientes.

Identificación de posibles cruces de peticiones

Seguidamente analizamos en cada estado estable de un CC la emisión, por parte de este CC, de una petición al CM, compatible con el estado del bloque, y el procesado en el CM de peticiones que inducen peticiones al CC[15]. El estado del bloque en el CC determina un posible conjunto de estados del bloque en el directorio. Las peticiones de los otros CC deben ser compatibles con el estado del bloque en el directorio.

Las peticiones emitidas por el CM se reciben en el CC en la ventana de vulnerabilidad del CC (Figura 7.12).

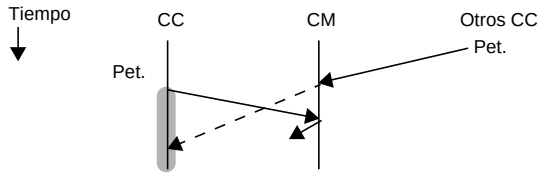


Figura 7.12 Protocolo VI. Cruce de peticiones de un CC y un CM.

Como la RV es ordenada, mantiene el orden de emisión de mensajes del CM a los CC. Entonces, un CC al recibir una petición en un estado transitorio debe procesarla, y si es el caso responder.

Un CC sólo tiene pendiente una petición a un mismo bloque. Recordemos que en una expulsión se espera una respuesta del CM. Entonces, mientras no se reciba la respuesta, un CC no emite otra petición al mismo bloque.

La RI utiliza un arbitraje por edad o antigüedad[16] y la CP del CM se gestiona de forma FIFO.

En estas condiciones, la petición de un CC, denominado CC1, se cruza con una petición de otro CC (CC2) o con la petición de expulsión del bloque por parte de este CC (CC2), pero no con ambas peticiones[17].

Una petición de expulsión sólo afecta al estado del bloque en el CM que gestiona el bloque. Entonces, una petición de expulsión puede determinar un cruce en el CC que la emite. También puede determinar un cruce en el CM, cuando la petición de otro CC es distinta de PtX.

Bloque en el estado V. Las peticiones de un CC1 al CM, cuando un bloque está en el estado V, son: PtE y PtX. El estado del bloque en el directorio puede ser NP o Pr (Figura 7.13).

15. Para ello el CC debe estar identificado en el VP.
16. El árbitro selecciona la petición más antigua.
17. Un CC no tiene pendientes dos peticiones al mismo bloque. La petición del CC1 ha sido encolada en la CP antes que la segunda petición de otro CC (CC2).

El CM puede procesar las siguientes peticiones de otros CC antes de la petición del CC1: PtE, Pt. La petición PtX por parte de otro CC no se considera en el análisis, ya que no determina un cruce con la petición del CC1 en el CM. En el CC1 la petición PtX de otro CC no puede determinar un cruce, ya que el CM al procesarla no emite ninguna petición al CC1.

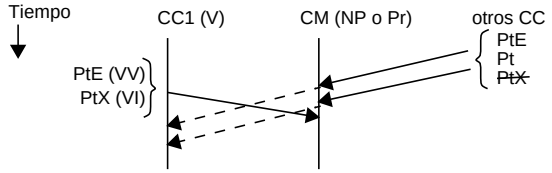


Figura 7.13 Bloque en estado V en la cache C1. Peticiones del CC1 y de otros CC.

Para que se produzca un cruce, la petición de un CC2, que procesa el CM, debe modificar el estado del bloque en la cache del CC1, cuya petición está esperando ser procesada en el CM (estado transitorio en el CC1).

Por tanto, todas las posibles secuencias, que se analicen, tienen que tener en primer lugar una instrucción que efectúe una escritura.

Una petición Pt sólo modifica el estado del bloque en el directorio. Recordemos que no se asigna contenedor en un fallo de escritura. Por tanto, es suficiente con analizar el caso en el cual el CC, que efectúa la primera petición de una secuencia de accesos a memoria, no tiene almacenado el bloque.

En la Figura 7.14 se muestra el orden de procesado en el CM de varias secuencia de accesos.

Orden de procesado en el CM	
A	**B**
P2 store t	P2 store t
P1 PtX t	P1 store t

Figura 7.14 Bloque en el estado V en la cache C1 y en el estado NP o Pr en el directorio. Orden de procesado en el CM de cada secuencia de accesos.

Estado I. Las peticiones de un CC1 al CM cuando un bloque está en el estado I son: PtE y Pt. El estado del bloque en el directorio puede ser NP o Pr (Figura 7.15).

El CM puede procesar las siguientes peticiones antes de la petición del CC1: PtE, Pt.

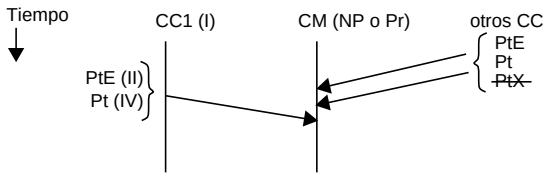


Figura 7.15 Bloque en estado I en la cache C1. Peticiones del CC1 y de otros CC.

El CC1 no recibe ninguna petición del CM, ya que no está identificado en el VP. Por tanto, en el CM no se infiere ningún cruce de peticiones.

Estados transitorios y transiciones en el controlador de memoria

Peticiones concurrentes al mismo bloque

En el diagrama de estados del CM no existen estados transitorios (Figura 7.10). La memoria siempre está actualizada y las peticiones del CM a los CC no requieren respuesta, ya que la RV es ordenada. Por tanto, en el CM no existen ventanas de vulnerabilidad.

Sin embargo, como pueden existir peticiones concurrentes de varios CC al mismo bloque: a) es posible que el CM tenga que procesar peticiones en las que se espera que el CC esté identificado en el VP del bloque, y no sea así o b) la petición no es esperada en el estado actual del bloque en el directorio. En estos casos, el CM infiere un cruce en un estado estable.

Identificación de cruces de peticiones

Estado V. La secuencia de accesos (B) donde el CC1 emite una petición PtE no determina la detección de un cruce de peticiones en el CM. En el protocolo VI, el CM procesa una petición PtE independientemente del estado del bloque en el directorio.

Dado el orden A de accesos a memoria, el CM al procesar la petición del CC2 emite una petición al CC1 y modifica el VP. El procesado de la petición PtX del CC1se puede efectuar en el CM con el bloque en el estado NP o Pr, en función de si el CC2 está en el VP. En cualquiera de los dos casos, al procesar el CM la petición del CC1, este CC no está incluido en el VP. Además, si la petición de CC2 se procesa en el estado NP no es esperada[18].

18. La petición del CC1 puede procesarse en el estado Pr aunque la cache C2 no contenga el bloque. Es suficiente que entre las dos peticiones el CM procese una petición Pt de un tercer CC.

Resumen. En la Figura 7.16 se indica el estado en el cual se infiere el cruce y la condición de detección del mismo.

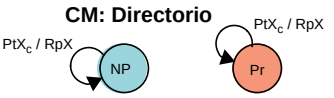
		Condición			
Secuencia	**Estado**	**P ∉ VP**	**P ∈ VP**	**no esperada**	**Diagrama de transiciones**
A	NP	X		X	
A	Pr	X			

Figura 7.16 Protocolo VI. Condiciones de inferencia de cruces en el directorio. El subíndice C indica que el CC que emite la petición no está en el VP o que es una petición no esperada en el estado actual del bloque en el directorio.

En resumen, en una petición PtX de un CC, el VP del directorio debería contener al CC que ha emitido la petición. Si este no es el caso, el CM infiere un cruce.

Estados y transiciones en los controladores de coherencia

El directorio es preciso. Por tanto, las expulsiones no son silenciosas. En los estados transitorios II e IV de un bloque en cache no existe ventana de vulnerabilidad (Figura 7.10)[19]. En estos estados, el VP del bloque en el directorio no incluye al CC que solicita el bloque.

En la Figura 7.17 se observa que en el estado V un CC puede recibir una petición PtObE del CM. El estado V es el estado estable a partir del cual se puede efectuar una transición a los estados transitorios VI y VV.

Ventana de vulnerabilidad. En los estados transitorios VI y VV un CC puede recibir una petición de observación de escritura (PtObE) del CM.

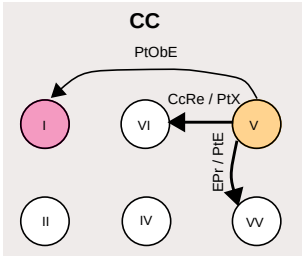


Figura 7.17 Protocolo de directorio VI. CC: peticiones que puede recibir un bloque estando en un estado transitorio.

19. Un CC sólo tiene pendiente una transacción a un bloque. Una segunda petición del CC al bloque se bloquea hasta que el CC recibe la respuesta de la transacción pendiente. Es posible que un CC tenga pendiente una expulsión de un bloque y una petición a un bloque distinto.

Orden de procesado en el CM	
A	B
P2 store t P1 PtX t	P2 store t P1 store t

Para facilitar la comprensión en el siguiente desarrollo, en el margen izquierdo se replica la Figura 7.14.

Dado el orden A de accesos a memoria, el CC1, con el bloque en el estado VI, recibe una petición PtObE del CM que requiere invalidar el bloque. El CC1 no cambia el estado del bloque, ya que al recibir la respuesta del CM establece como estado del bloque I.

Dado el orden B de accesos a memoria, cuando el CC1 procesa la petición PtObE, emitida por el CM al procesar la petición PtE del CC2, el bloque en la cache C1 está en el estado VV. El CC1 está esperando la notificación de la escritura por parte del CM para pasar el bloque a estado V. Sin embargo, el CM ha ordenado una petición PtE de otro CC antes que la petición del CC1. Por tanto, el bloque debe quedar en estado inválido al recibir la confirmación de la escritura. Para recordar la escritura, de otro CC, que ha sido ordenada previamente por el CM, se utiliza un estado transitorio denominado VVI. Entonces, el CC1 al procesar la petición PtObE del CM cambia el estado del bloque a VVI y posteriormente, cambia el estado del bloque a I, cuando recibe la respuesta del CM a su petición.

Seguidamente se presenta un análisis detallado de los cruces de peticiones.

Estado VV.

En el estado VV un CC puede recibir una petición PtObE, lo cual indica que el CM ha procesado previamente una petición PtE de otro CC.

En la Figura 7.18 se muestra un diagrama temporal y las transiciones entre estados. En la cache C1, la transición del estado estable del bloque a un estado transitorio está determinado por la emisión de una petición del CC1. Es suficiente que la emisión de la petición PtE se efectúe antes que la recepción de la petición PtObE del CM. El CC1 debe invalidar el bloque, ya que la escritura de otro procesador (PtE) ha sido ordenada por el CM antes que la petición de escritura del CC1. En consecuencia, en el CM se ha modificado el VP y el bit correspondiente al CC1 no está activado. Por tanto, la información almacenada en el campo de datos de la cache del CC1, cuya petición está pendiente, no es válida[20].

El CM al procesar una petición PtE no suministra el bloque, sólo actualiza la memoria. Por tanto, cuando el CC1 reciba la respuesta a su petición PtE debe establecer I como estado del bloque. En consecuencia hay que identificar el cruce en el CC1. Para ello, se utiliza un nuevo estado transitorio denominado VVI. El CC1, cuando recibe una petición PtObE del CM en el estado VV, cambia el estado del bloque a VVI y posteriormente, al recibir la respuesta del CM, a su petición PtE, establece I como estado del bloque.

20. Recordemos también que no se asigna contenedor en un fallo de escritura.

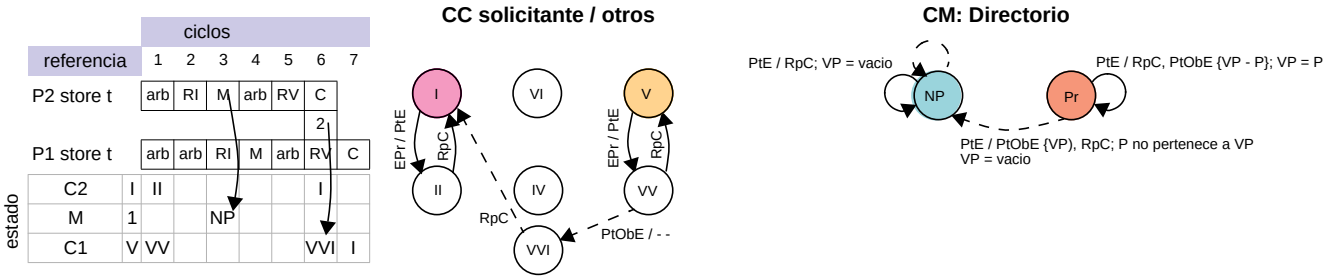


Figura 7.18 Cruce entre una petición PtObE del CM y una petición PtE de un CC. En trazo continuo las transiciones debidas al CC de C2. En trazo discontinuo las transiciones debidas al CC de C1.

En el ejemplo del diagrama temporal de la Figura 7.18, el CM no infiere un cruce, ya que el estado del bloque, al procesar la petición PtE del CC1, es NP. El CM tampoco infiere un cruce cuando procesa la petición PtE en el estado Pr y el CC1 no está en el VP. Este último caso se produce cuando en la cache C2 el bloque está en estado V al emitir la petición PtE (se muestra en el diagrama de transiciones entre estados de la Figura 7.18). Recordemos que un CC puede emitir una petición PtE a un bloque en el estado I.

El funcionamiento de la RV garantiza que el CC1 recibe primero la petición del CM y después la respuesta del CM a su petición. De forma similar, el CC2 recibe primero la respuesta a su petición y después la petición del CM, si es el caso, inducida al procesar la petición del CC1 (bloque en estado V en C2).

Estado VI.

En el estado VI el CC puede recibir una petición PtObE, lo cual indica que el CM ha procesado previamente una petición PtE de otro CC[21].

En la Figura 7.19 se muestra un diagrama temporal y las transiciones entre estados. En la cache C1, la transición del estado estable del bloque a un estado transitorio está determinado por la emisión de una petición del CC1. Es suficiente que la emisión se efectúe antes que la recepción de la petición del CM[22] (PtObE). El CM, al procesar la petición del CC2, ha eliminado al CC1 del VP del bloque en el directorio. Entonces, el CM procesa la petición PtX en el estado NP o en el estado Pr.

21. A este cruce se le denomina usualmente petición tardía. La petición del CM llega después de iniciar la expulsión.
22. Notemos que temporalmente el procesado de la petición del CC1 en el CM puede ser bastante después si hay entrelazadas peticiones a otros bloques. También pueden ser peticiones de lectura de otros CC al mismo bloque. Cualquiera de estas peticiones al ser procesada por el CM no induce una petición al CC1.

En el diagrama de la Figura 7.19 se muestra el caso de que el estado sea NP, que se corresponde con el estado I del bloque en la cache C2[23].

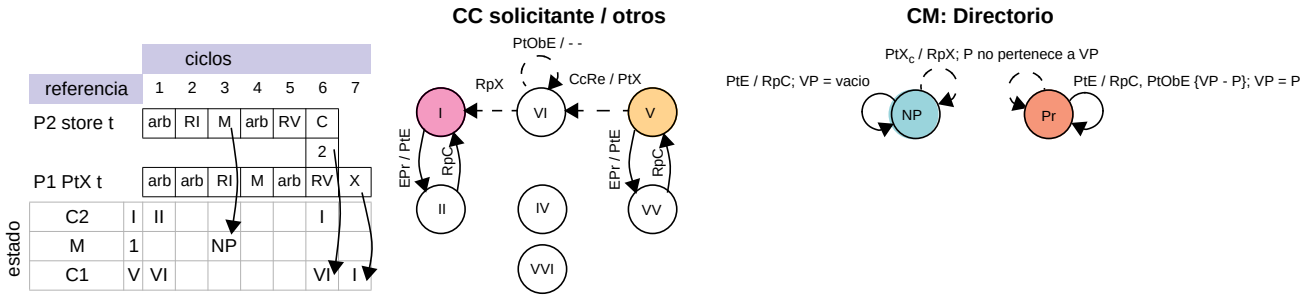


Figura 7.19 Cruce entre una petición PtObE del CM y una petición PtX de un CC. En trazo continuo las transiciones debidas al CC de C2. En trazo discontinuo las transiciones debidas al CC de C1.

El CM infiere un cruce en los dos estados, ya que al procesar la petición PtX espera que el CC esté identificado en el VP y no es así. El CM responde a la petición PtX y no modifica el estado del bloque en el directorio. El CC al procesar la petición PtObE en el estado VI tampoco cambia de estado.

Resumen. En la Figura 7.20 se muestran los estados en los cuales se infiere un cruce y la respuesta del CC.

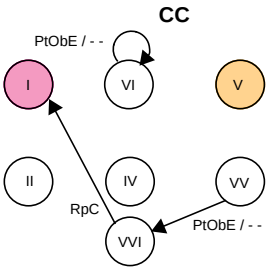


Figura 7.20 Protocolo VI. Inferencia de cruces en un CC.

Diagramas completos de cruce de peticiones

En la Figura 7.21 se muestran las transiciones en un CC y en el CM cuando se produce un cruce de peticiones. En el CM se infiere un cruce cuando el CC no está en el vector de presencia y se recibe una petición PtX en cualquiera de los dos estados.

23. Si el estado en la cache C2 fuera V, la petición PtX del CC1 se procesaría en el CM en el estado Pr.

Al procesar una petición, la primera acción en el CM es inferir si se ha producido un cruce de peticiones con un CC. Para ello el CM utiliza el VP. En el diagrama de estados de un bloque en el directorio, se utiliza el subíndice c (cruce) para identificar una petición de un CC que se ha cruzado con una petición del CM.

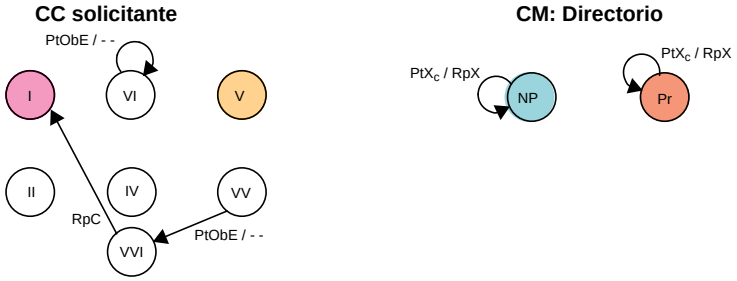


Figura 7.21 Diagramas de transiciones en un CC y en el directorio cuando se produce un cruce de peticiones a un bloque entre un CC y el CM.

Notemos que al recibir un CC una petición PtObE del CM, a un bloque en estado VV, el CC infiere que el CM ha extraído al CC del VP. Es como si el CC efectuara la petición PtE desde el estado I. Por ello, al recibir la respuesta a la petición PtE el estado del bloque debe ser I.

De forma similar, un CC al recibir una petición PtObE del CM, a un bloque en estado VI, infiere que el CM ha extraído al CC del VP y espera la respuesta a su petición PtX.

Tablas de estados y transiciones

En la Tabla 7.1 se muestra la descripción de los estados, estables y transitorios, eventos y transiciones entre estados en un CC en formato tabla. Las casillas que no contienen información indican un error. En un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

			Eventos del procesador y reemplazo			Eventos externos (respuestas y petición)			
			LPr	EPr	CcRe	RpD	RpC	RpX	PtObE
Estados	Estables	I	Pt; IV	PtE; II					
		V	- - ; V	PtE; VV	PtX; VI				- - ; I
	transitorios	IV				- -; V			
		VV					- - ; V		- - ; VVI
		VI						- - ; I	- - ; VI
		II					- -; I		
		VVI					--; I		

Tabla 7.1 Peticiones concurrentes. Protocolo de directorio VI. Tabla de estados y transiciones de un bloque en cache. Las casillas con fondo blanco indican cruces de peticiones.

En la Tabla 7.2 se muestra en formato tabla los estados y transiciones entre estados de un bloque en el directorio. En el evento PtE se distinguen dos casos en función de si el procesador, cuyo CC efectúa la petición, está o no está en el vector de presencia. En el evento PtX se distingue el caso de que el CC sea el único que está en el vector de presencia o haya más CC.

			Eventos del controlador de coherencia					
			Pt	PtE		PtX		
				VP = vacio	P ∈ VP	P ∉ VP	VP = P	VP ≠ P P ∈ VP	P ∉ VP
Estados	Estables	NP	RpD; Pr, VP = P	RpC; NP, VP = vacío					RpX; NP, VP = vacío
		Pr	RpD; Pr, VP = VP + P		RpC, PtObE {VP - P}; Pr, VP = P	RpC, PtObE {VP}; NP, VP = vacío	RpX; NP, VP = vacío	RpX; Pr, VP = VP - P	RpX; Pr, VP = VP

Tabla 7.2 Peticiones concurrentes. Protocolo de directorio VI. Tabla de estados y transiciones de un bloque en el directorio. Las casillas con fondo blanco indican cruces de peticiones.

Representación de transacciones y transiciones entre estados

En este apartado se muestran tres formas de representar una secuencia de accesos a memoria: a) en formato tabla y b) mediante un diagrama temporal y c) mediante un diagrama temporal simplificado.

Representación en formato tabla

Los accesos a memoria se muestran en grupos separados mediante líneas horizontales continuas. Cada grupo de accesos a memoria se gestiona independientemente y no se empieza a gestionar el siguiente grupo hasta que ha finalizado el anterior. Los accesos a memoria de un grupo se realizan concurrentemente en el mismo ciclo. En una fila de la tabla, de izquierda a derecha, después de la instrucción de acceso a memoria se especifica:

1 La primera columna se utiliza para representar el arbitraje de peticiones. Esto es, se indica el orden de las transacciones, si es el caso. Supondremos usualmente que en un grupo de peticiones el árbitro las selecciona en el orden en que se especifica la secuencia de accesos a memoria.

2 El siguiente grupo de columnas se utiliza para representar el estado transitorio del bloque, en la cache correspondiente, cuando se emite la petición.

3 El tercer grupo de columnas se utiliza para representar la petición del CC en la red de ida (RI). La petición se indica en la casilla correspondiente y en el orden en el que el árbitro concede el acceso al módulo de memoria.

4 El cuarto grupo de columnas se utiliza para identificar la variable o bloque y el VP en el directorio. Los bits en el VP identifican a las caches; de izquierda a derecha en ordinal creciente. Un valor de uno en el VP indica que hay una copia del bloque en la cache correspondiente.

5 La siguiente columna se utiliza para indicar qué elemento (memoria o cache) suministra el bloque o dato.

6 La siguiente columna representa el arbitraje en la red de vuelta (RV). Como las peticiones han ocupado la RI en ciclos distintos, también ocupan la RV en ciclos distintos. En esta columna se indica el ordinal de los procesadores que reciben la respuesta o peticiones (destinatario). Para la respuesta se utiliza una fila y para las peticiones la siguiente fila y la misma columna.

7 El séptimo grupo de columnas se refiere a la RV. La representación es idéntica a la RI exceptuando que los mensajes que se indican son de respuesta y petición. Cuando haya que indicar una petición, se utiliza la siguiente fila y además se indica el ordinal de los controladores de coherencia que reciben la petición.

8 El octavo grupo de columnas se utiliza para identificar la variable accedida en cache y el estado del bloque al finalizar la acción de coherencia. Cuando un bloque está en una ventana de vulnerabilidad y recibe una petición del CM se indica el estado transitorio del bloque. En estas

condiciones, el estado transitorio, debido a que se ha atendido una petición del CM, se observa en las columnas de la derecha. El estado se indica en las filas asociadas a las peticiones del CM.

9 Si no se accede al CM la fila correspondiente a la petición se deja en blanco.

Ejemplo. En la Tabla 7.3 se muestra una secuencia de accesos a memoria realizada por tres procesadores. Los accesos a memoria están agrupados de dos en dos. Cuando se inicia la secuencia de accesos a memoria, las caches no almacenan los bloques en los que se ubican las variables accedidas. Tampoco se producen conflictos de contenedor al almacenar los bloques en los contenedores de cache.

Los primeros dos accesos concurrentes requieren iniciar una transacción cada uno. La primera transacción es debida a un fallo de lectura y la segunda es debida a una escritura y los bloques referenciados son distintos. Los mensajes de petición emitidos por CC1 y CC2 son Pt y PtE respectivamente. El mensaje de petición de CC1 es elegido por el árbitro en primer lugar. El estado transitorio del bloque en la cache C1 es IV y en la cache C2 es II. El estado final del bloque en la cache C1 es V y en la cache C2 es I. En el directorio se indica que el CC1 tiene copia del bloque que contiene la variable t.

El segundo par de accesos a memoria son fallos de lectura al mismo bloque. El estado transitorio en la cache C1 y en la cache C3 es IV, ya que la transacción, que inicia cada uno de ellos, es debida a un fallo de lectura. El mensaje emitido por CC1 y por CC3 es Pt. La respuesta del CM a cada uno de ellos es RpD. Después de finalizar las dos transacciones, el VP del directorio identifica que CC1 y CC3 tienen copia del bloque que contiene la variable u.

El tercer par de accesos a memoria requiere una transacción para servir un fallo de lectura (CC1) y una transacción para servir una escritura (CC2). La primera transacción es mimética a los fallos de lectura previos. La segunda transacción requiere que el CM emita mensajes de observación de escritura (PtObE) a CC1 y CC3 que tienen copia del bloque. Al finalizar la segunda transacción el VP del bloque, que contiene la variable u, en el directorio indica que no hay copia del bloque en ninguna cache.

En el siguiente par de accesos a memoria tenemos que el primero de ellos es un acierto en cache. El segundo acceso requiere una transacción para servir un fallo de lectura. El VP del bloque, que contiene la variable w, en el directorio indica que hay copia del bloque en las caches C1 y C3.

	arb.	C 1	C 2	C 3	RI		mem.			arb.	RV		C 1		C 2		C 3	
acceso	ord.	est.	est.	est.	1º	2º	var.	VP	sum.	dest.	1º	2º	var.	est.	var.	est.	var.	est.
1. P1 load t	1º	IV			Pt		t	1, 0, 0	mem.	CC1	RpD		t	V				
1. P2 store u	2º		II			PtE	u	0, 0, 0	C2	CC2		RpC			u	I		
2. P1 load u	1º	IV			Pt		u	1, 0, 0	mem.	CC1	RpD		u	V				
2. P3 load u	2º			IV		Pt	u	1, 0, 1	mem.	CC3		RpD					u	V
3. P1 load w	1º	IV			Pt		w	1, 0, 0	mem	CC1	RpD		w	V				
3. P2 store u	2º		II			PtE	u	0, 0, 0	C 2	CC2		RpC			u	I		
										CC1,3		PtObE 1,3	u	I			u	I
4. P1 load w																		
4. P3 load w	1º			IV	Pt		w	1, 0 ,1	mem	CC3	RpD						w	V
5. P2 store w	1º		II		PtE		w	0, 0, 0	C 2	CC2	RpC							
										CC1,3	PtObE 1,3		w	VVI			w	I
5. P1 store w	2º	V V				PtE	w	0, 0, 0	C1	CC1		RpC	w	I				

Tabla 7.3 Peticiones concurrentes. Protocolo de directorio VI. Secuencia de accesos a memoria concurrentes.

El último par de transacciones son debidas a dos escrituras a la misma variable (PtE). La cache C2 no almacena el bloque que contiene la variable y la cache C1 almacena el bloque. El estado transitorio del bloque en la cache C2 es II y en la cache C1 es VV. El CM al procesar la petición PtE de CC2 emite mensajes de observación de escritura (PtObE) a CC1 y CC3. Las caches correspondientes tienen copia del bloque. Al recibir el mensaje PtObE, CC3 invalida el bloque y CC1 establece el estado transitorio VVI. Al finalizar la primera transacción el VP del bloque en el directorio indica que no hay copia del bloque en las caches. El CC1 al recibir la respuesta del CM invalida el bloque.

Diagrama temporal

La fase arb se utiliza para mostrar la ordenación de los mensajes. La espera para acceder a una red se indica representando en ciclos consecutivos la fase de arbitraje (arb). El resto se representa de la forma descrita en el Capítulo 5.

Ejemplo. En la Figura 7.22 se muestra el diagrama temporal de la secuencia de accesos a memoria mostrada en la Tabla 7.3.

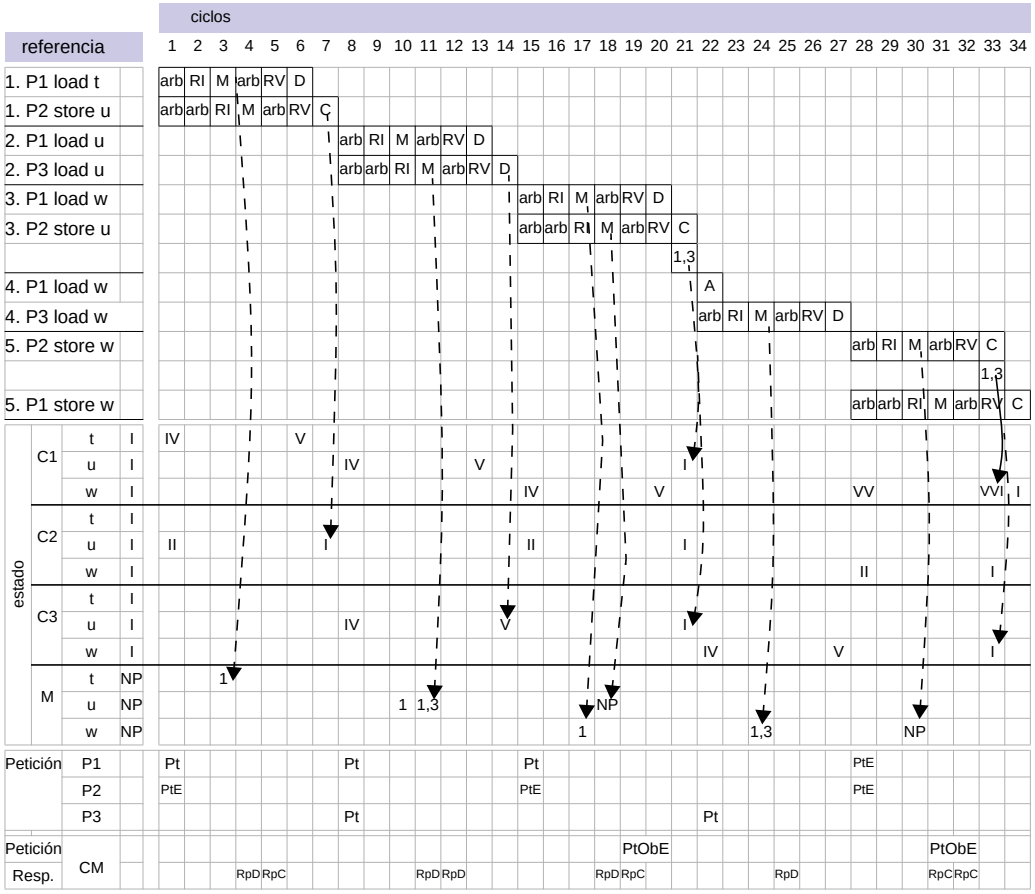


Figura 7.22 Peticiones concurrentes. Protocolo de directorio VI. Diagrama temporal de una secuencia de accesos a memoria concurrentes.

Diagrama temporal simplificado

En el diagrama temporal simplificado de la Figura 7.23 se utiliza la secuencia de accesos a memoria mostrada en la Tabla 7.3. Las líneas horizontales a trazos separan grupos de peticiones que se inician concurrentemente. El siguiente grupo de peticiones no se inicia hasta que ha finalizado el grupo previo. El orden de arbitraje se observa en el orden de llegada de las peticiones concurrentes al CM. Un círculo negro indica que el acceso a memoria no genera una transacción.

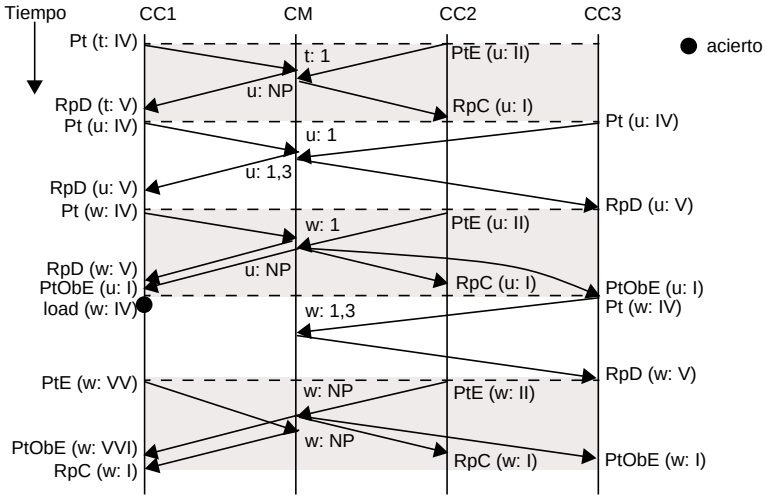


Figura 7.23 Peticiones concurrentes. Protocolo de directorio VI. Diagrama temporal simplificado de una secuencia de accesos a memoria concurrentes.

PROTOCOLO DE DIRECTORIO MLI

La descripción del protocolo de directorio MLI se ha efectuado en el Capítulo 6. En cuanto al camino de datos es el descrito en el mismo capítulo.

En la Figura 7.24 por completitud se muestran los mensajes en los tres tipos de transacciones del protocolo de coherencia (Figura 6.23).

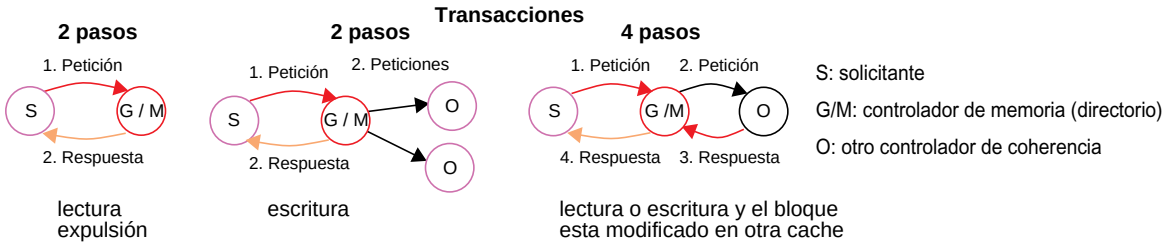


Figura 7.24 Protocolo MLI. Escritura retardada. Mensajes en los tres tipos de transacciones.

Organización del multiprocesador

En la Figura 7.25 se muestra la organización del multiprocesador. En ella se distingue la cola de peticiones y la cola de respuestas de los CC al CM, juntos con las redes asociadas (RI y RCM). También se observa el encaminamiento de los mensajes del CM a los CC.

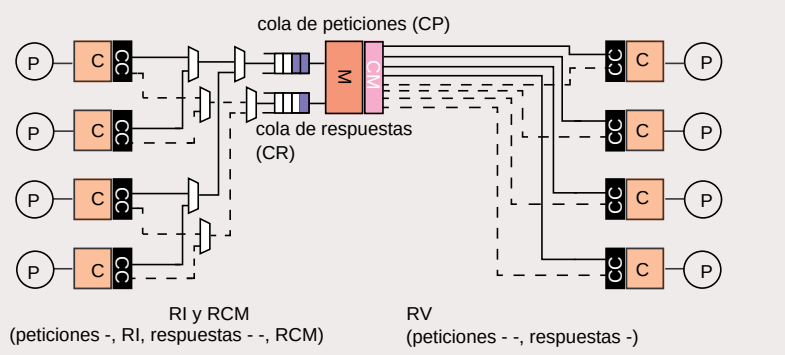


Figura 7.25 Protocolo de directorio MLI. Organización del multiprocesador. Caches privadas con escritura retardada.

En la Figura 7.25 se distinguen conexiones punto a punto para identificar peticiones y respuestas del CM a los CC, aunque la RV es única.

Al procesar el CM una petición, que requiere la colaboración de varios CC, los destinos del conjunto de peticiones PtObE y la respuesta (RpD) son disjuntos. Por tanto, puede utilizarse la misma red para peticiones y respuestas del CM (2 pasos en la Figura 7.24). Por tanto, las redes RV y RMC se pueden implementar mediante una única red, a la que denominamos RV. En estas condiciones, en una transacción de dos pasos sólo indicaremos la red RV.

Al procesar el CM una petición, que requiere que un CC suministre el bloque, sólo hay una petición a este CC. Al recibir la respuesta del CC, que participa en la acción de coherencia, el CM emite una respuesta al CC solicitante (4 pasos en la Figura 7.24). En este caso, para la petición del CM indicaremos la red RMC y para la respuesta del CM indicaremos la red RV, aunque la red RMC y la red RV sean la misma.

Transiciones entre estados estables de los controladores

En la Figura 7.26 se muestra, por completitud, el diagrama de transiciones entre estados de un bloque en el directorio y en un CC (Figura 6.40 y Figura 6.41).

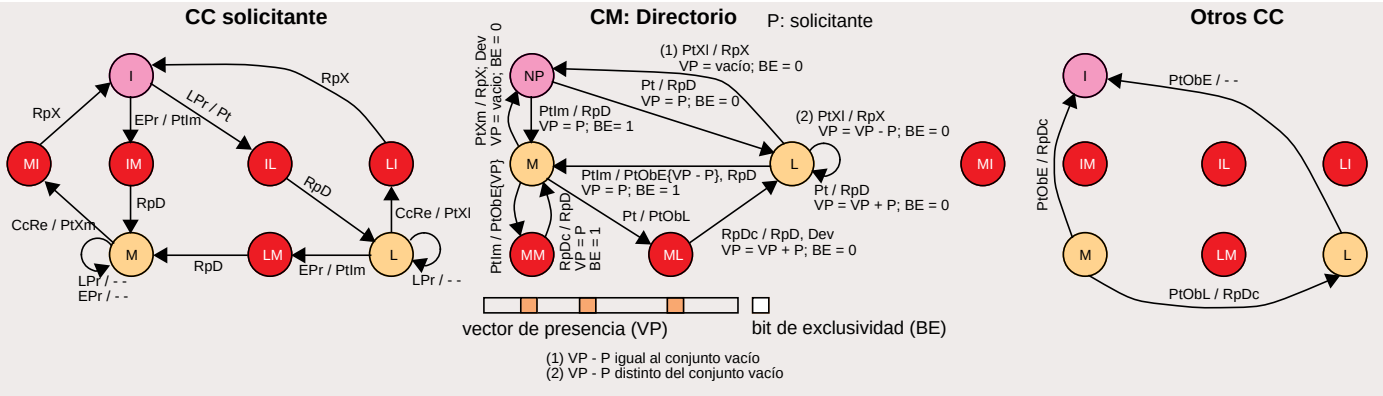


Figura 7.26 Transiciones entre estados en el CC y en el CM.

Concurrencia en el controlador de memoria

Peticiones concurrente a bloques distintos

El procesado de una petición en el CM puede necesitar solicitar el bloque a una cache y esperar la respuesta (petición pendiente). Durante este periodo de tiempo el estado del bloque es transitorio (MM y ML, Figura 7.26).

Mientras el CM está esperando la respuesta de un CC, puede extraer otra petición de la CP y procesarla durante este lapso de tiempo. Si el bloque de memoria, al que accede esta segunda petición, es distinto de los bloques que están en un estado transitorio, el CM puede iniciar el procesado de la petición y por tanto existe concurrencia.

En la Figura 7.27 se muestra la concurrencia en el procesado de peticiones en el CM. Las dos primera peticiones se efectúan en el mismo ciclo y hay que arbitrar. El procesado de la primera petición necesita que el directorio solicite el bloque a otro CC. En la parte inferior de la Figura 7.27 se indican los cuatros mensajes correspondientes a la primera petición.

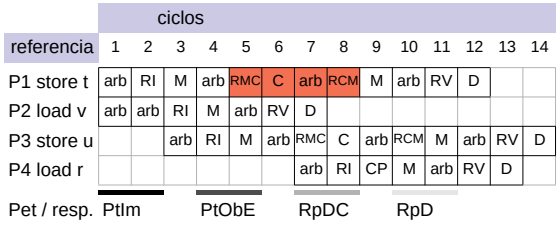


Figura 7.27 Protocolo MLI. Concurrencia en el CM. Las peticiones referencian bloques distintos. El acrónimo CP indica cola de pendientes.

La segunda y tercera petición se pueden procesar mientras se espera la respuesta del CC. La cuarta petición se efectúa dos ciclos antes de que el CM reanude el procesado de la primera petición, al recibir la respuesta de la cache[24]. Entonces, la cuarta petición está almacena en la CP durante un ciclo, para eliminar el riesgo estructural en la fase M entre ella y la primera petición.

Identificación de posibles cruces de peticiones

Seguidamente analizamos en cada estado estable de un CC la emisión, por parte de este CC, de una petición al CM, compatible con el estado del bloque, y el procesado en el CM de peticiones que inducen peticiones al CC[25]. El estado del bloque en un CC determina un posible conjunto de estados del bloque en el CM. Las peticiones de los otros CC deben ser compatibles con el estado del bloque en el directorio.

Las peticiones emitidas por el CM se reciben en el CC durante la ventana de vulnerabilidad (Figura 7.28).

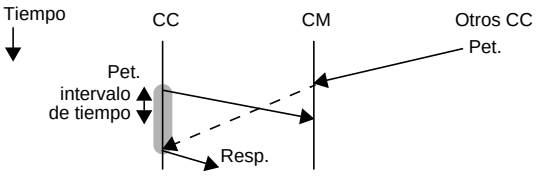


Figura 7.28 Cruce de peticiones de un CC y un CM.

Como la RV es ordenada, mantiene el orden de emisión de mensajes del CM a los CC. Entonces, un CC al recibir una petición en un estado transitorio debe procesarla, y si es el caso responder.

24. En el directorio el procesado de las respuestas es prioritario frente a las peticiones.
25. Para ello el CC debe estar identificado en el VP.

Un CC sólo tiene pendiente una petición a un mismo bloque. Recordemos que en una expulsión se espera una respuesta del CM. Entonces, mientras no se reciba la respuesta, un CC no emite otra petición al mismo bloque.

La RI utiliza un arbitraje por edad o antigüedad[26] y la CP del CM se gestiona de forma FIFO.

En estas condiciones, la petición de un CC, denominado CC1, se cruza con la petición de otro CC (CC2) o con la petición de expulsión del bloque por parte de este CC (CC2), pero no con ambas peticiones.

Una petición de expulsión sólo afecta al estado del bloque en el CM que gestiona el bloque. Entonces, una petición de expulsión puede determinar un cruce en el CC que la emite. También puede determinar un cruce en el CM, cuando la petición de otro CC es distinta de PtXl o PtXm.

Bloque en el estado L. Cuando un bloque está en el estado L en la cache, las peticiones de un CC1 al CM son: PtIm y PtXl. El estado del bloque en el directorio es L (Figura 7.29).

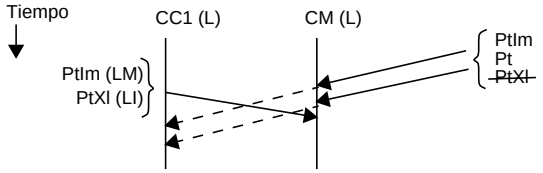


Figura 7.29 Bloque en estado L en la cache C1. Peticiones del CC1 y de otros CC.

El CM puede haber procesado (ordenado), antes de la petición del CC1, las siguientes peticiones de otros CC: PtIm y/o Pt. La petición PtXl por parte de otro CC no se considera en el análisis, ya que no determina un cruce con la petición del CC1 en el CM. En el CC1, la petición PtX de otro CC no puede determinar un cruce, ya que el CM no emite ninguna petición al CC1 al procesarla.

Para que se produzca un cruce, alguna de las peticiones que el CM procesa, antes de la petición del CC1, debe modificar el estado del bloque en la cache C1. Por tanto, como el estado del bloque en el CM es L, todas las posibles secuencia de accesos a memoria, que se analicen, tienen que tener como primer acceso a memoria una instrucción que genera una petición de exclusividad al CM (PtIm). En la Figura 7.30 se muestra el orden de procesado en el CM de varias secuencia de accesos.

26. El árbitro selecciona la petición más antigua.

Orden de procesado en el CM			
A	B	C	D
P2 store t	P2 store t	P2 store t	P2 store t
P1 PtXl t	P3 load t	P1 store t	P3 load t
	P1 PtXl t		P1 store t

Figura 7.30 Bloque en estado L en la cache C1 y en el directorio. Orden de procesado en el CM de cada secuencia de accesos.

Bloque en el estado M. Cuando un bloque está en el estado M en la cache, la petición de un CC1 al CM es PtXm. El estado del bloque en el directorio es M (Figura 7.31).

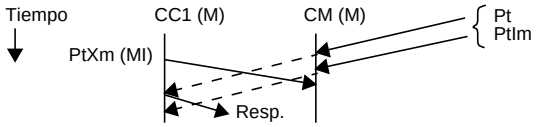


Figura 7.31 Bloque en estado M en la cache C1. Peticiones del CC1 y de otros CC.

El CM puede haber procesado (ordenado), antes de la petición del CC1, las siguientes peticiones: PtIm, Pt.

Para que se produzca un cruce de peticiones, alguna de las peticiones que el CM procesa, antes de la petición del CC1, debe modificar el estado del bloque en la cache C1. Cualquiera de las dos peticiones enumeradas (PtIm, Pt) modifica el estado del bloque en la cache C1. En la Figura 7.32 se muestra el orden de procesado en el CM de varias secuencia de accesos.

Orden de procesado en el CM			
E	F	G	H
P2 store t	P2 load t	P2 store t	P2 load t
P1 PtXm t	P1 PtXm t	P3 load t	P3 store t
		P1 PtXm t	P1 PtXm t

Figura 7.32 Bloque en estado M en la cache C1 y en el directorio. Orden de procesado en el CM de cada secuencia de accesos.

Bloque en el estado I. No se reciben peticiones, ya que el CC1 no está identificado en el VP.

Estados transitorios y transiciones en el controlador de memoria

Peticiones concurrentes al mismo bloque

Ventana de vulnerabilidad. Mientras el CM está esperando una respuesta puede analizar una petición al mismo bloque (estados transitorios MM y ML).

El directorio es el punto de ordenación de las peticiones al mismo bloque. En otras palabras es el encargado de serializar las peticiones al mismo bloque. Para efectuar la serialización, una alternativa es no iniciar en el CM el procesado de una petición hasta que ha finalizado el procesado de la petición previa al mismo bloque. Esto es, cuando el bloque está en un estado estable en el CM.

Un mecanismo en el CM, para gestionar que no se procesa una petición, es bloquear el análisis de la CP hasta que el CM recibe la respuesta relacionada con el bloque que determina el bloqueo[27] (Figura 7.33).

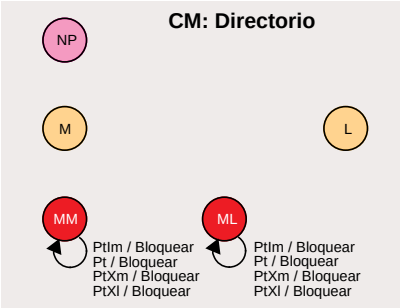


Figura 7.33 CM: bloqueo del procesado de peticiones en los estados transitorios del directorio.

En la Figura 7.34 se muestra la gestión de dos peticiones que referencian el mismo bloque. La primera petición (P1 store t) requiere una petición del CM a otro CC para mantener la coherencia. Para representar el bloqueo del análisis de peticiones utilizaremos el acrónimo B. El bloqueo se propaga a la peticiones más jóvenes, independientemente del bloque al que acceden. Estas peticiones más jóvenes se almacenan o están almacenadas en la CP. La petición que queda bloqueada se procesa en el CM después de que el CM emita la respuesta de la petición que ha determinado el bloqueo.

27. El directorio sigue procesando respuestas a medida que llegan. En consecuencia, el bloqueo del procesado de la CP desaparece al procesar la respuesta que está relacionada con el bloqueo.

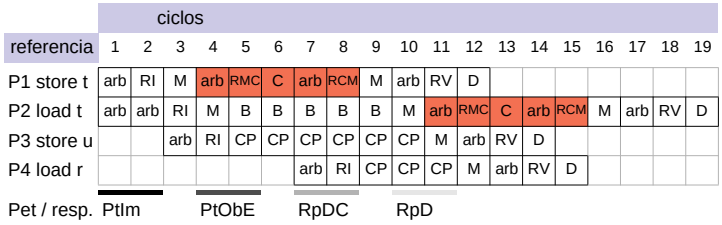


Figura 7.34 Bloqueo del procesado de peticiones en el CM. Dos peticiones referencian el mismo bloque y para servir la 1ª petición es necesario efectuar una petición a otro CC.

El CM no procesa peticiones de los CC a un bloque en un estado transitorio. Entonces, el resto de cruces entre peticiones del CM y un CC se observan (infieren) en estados estables.

Identificación de cruces de peticiones

Para facilitar la comprensión en el siguiente desarrollo, en el margen izquierdo se replica la Figura 7.30.

Orden de procesado en el CM	
A	**B**
P2 store t P1 PtXl t	P2 store t P3 load t P1 PtXl t

Orden de procesado en el CM	
C	**D**
P2 store t P1 store t	P2 store t P3 load t P1 store t

Bloque en el estado L. Los ordenes de acceso a memoria C y D, donde el CC1 emite una petición PtIm, no determinan la detección de un cruce en el CM. Recordemos que cuando el CM analiza una petición de un CC, a un bloque en un estado transitorio en el directorio, el procesado de las peticiones en la CP se bloquea (secuencia D). El procesado se reanuda cuando el bloque está en un estado estable. En el protocolo MLI, el CM procesa una petición de exclusividad independientemente del estado estable (Figura 7.26). Entonces, sólo tendremos en cuenta los ordenes A y B para analizar cruces de peticiones en el directorio.

Dado el orden de accesos a memoria A, el CM, al procesar la petición del CC2, establece el estado M en el directorio. Entonces, al procesar el CM la petición del CC1, el estado del bloque no es el esperado. La petición PtXl de CC1 se espera procesar en el estado L. Por otro lado, el VP tiene activado el identificador de un CC distinto al identificador de CC1 y el BE está activado. El CM responde a la petición del CC1 (RpX).

El CM, al procesar el orden B de accesos a memoria, establece la secuencia de estados estables M y L, al procesar respectivamente las peticiones del CC2 y del CC3. La petición del CC1 se procesa en el estado L. Sin embargo, el VP no tiene activado el identificador del CC1. El CM responde a la petición del CC1 (RpX).

Para facilitar la comprensión en el siguiente desarrollo, en el margen izquierdo se replica la Figura 7.32.

Orden de procesado en el CM	
E	**F**
P2 store t P1 PtXm t	P2 load t P1 PtXm t

Orden de procesado en el CM	
G	**H**
P2 store t P3 load t P1 PtXm t	P3 load t P2 store t P1 PtXm t

Bloque en el estado M. Dado el orden de accesos a memoria E, el CM establece el estado M en el directorio al procesar la petición del CC2. Cuando el CM procesa la petición del CC1 el contenido de VP no identifica al CC1. El CM responde al CC1 y no se actualiza memoria.

Dado el orden de accesos a memoria F, el CM establece el estado L en el directorio, al procesar la petición del CC2. Entonces, cuando el CM procesa la petición del CC1 el estado no es el esperado. Sin embargo, el VP tiene activado el identificador del CC1. El CM responde al CC1, desactiva el identificador del CC1 de VP y no actualiza la memoria.

Dado el orden G de accesos a memoria, el CM establece los estados M y L en el directorio al procesar respectivamente las peticiones del CC2 y del CC3. Cuando el CM procesa la petición del CC1 el estado del bloque es L y la petición PtXm no es esperada. Además, el VP no tiene activado el identificador del CC1. El CM responde y no actualiza memoria.

Dado el orden H de accesos a memoria, la petición del CC1 se procesa en el estado M y el VP no tiene activado el identificador del CC1. El CM responde y no actualiza memoria. La funcionalidad del CM en el orden H de accesos a memoria está cubierta por el orden E.

Directorio: bloque en el estado NP. Seguidamente razonamos que, cuando el estado del bloque es NP, no se producen cruces en el directorio. El CM procesa las peticiones almacenadas en la CP de forma FIFO.

Sean dos controladores CC1 y CC2 (Figura 7.35). Una petición de CC1 induce una petición del CM al CC2, que se cruza con una expulsión de CC2 (PtXm, PtXl). La petición del CC2 debe haberse emitido antes de que la petición del CM, inducida por la petición de CC1, se haya procesado en el CC2. La transacción del CC1, que ha determinado el cruce, finaliza como muy pronto a la par que llega la petición del CM al CC2. Cualquier transacción que expulse el bloque en el CC1 se encolará en la CP después de la petición PtXm o PtXl de CC2[28]. Entonces, en el estado NP no se detectan cruces.

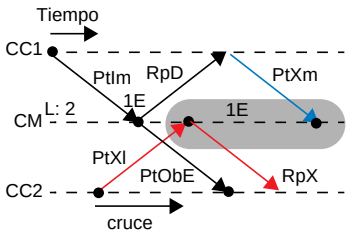


Figura 7.35 Protocolo LMI. Cruce de peticiones.

Resumen. En la Figura 7.36 se indica el estado en el cual se infiere el cruce y la condición de detección del mismo.

		Condición			
Secuencia	**Estado**	$P \notin VP$	$P \in VP$	**no esperada**	**Diagrama de transiciones**
A	M	X		X	
B	M	X			
E	M	X			
F	L		X	X	
G	L	X		X	
H	L	X			

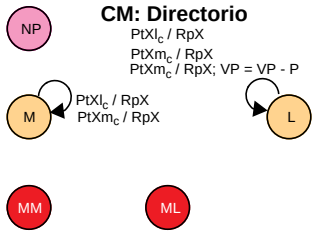


Figura 7.36 Protocolo MLI. Condiciones de inferencia de cruces en el directorio. El subíndice C indica que el CC que emite la petición no está en el VP o que es una petición no esperada en el estado del bloque en el CM.

En resumen, en peticiones PtXm y PtXl de un CC, el VP del directorio debería contener al CC que ha emitido la petición. Si este no es el caso, el CM infiere un cruce. En el CM, también se infiere un cruce cuando una petición PtXm de un CC se procesa en el estado L y el VP tiene identificado al CC.

Estados transitorios y transiciones en los controladores de coherencia

El directorio es preciso. Por tanto, las expulsiones no son silenciosas. En los estados transitorios IL e IM de un bloque en cache no existe ventana de vulnerabilidad, ya que el vector de presencia del bloque en el directorio indica que no existe copia del bloque en las caches de los procesadores (Figura 7.26).

En los estados transitorios LM, MI y LI existe una ventana de vulnerabilidad. Durante la espera de la respuesta a una petición, en uno de estos estados transitorios, se puede recibir del CM una petición de observación de escritura (PtObE) y en el caso particular del estado transitorio MI, también se puede recibir una petición de observación de lectura (PtObL). En la Figura 7.37 se observa que es factible recibir estas peticiones en los estados M y L, que son los estados estables desde los que se parte para llegar a los estados transitorios. Para que se produzca un cruce de peticiones, el CM ha procesado y

28. El árbitro que determina el orden, en el cual se encolan las peticiones en la CP, concede el permiso por antigüedad. Si este no es el caso, en el CM hay que procesar cruces de peticiones, siendo el estado del bloque NP.

ordenado una o algunas peticiones de otros CC, antes que la petición del CC que recibe la petición del CM. En particular, una de las peticiones ha inducido que el CM emita una petición al CC.

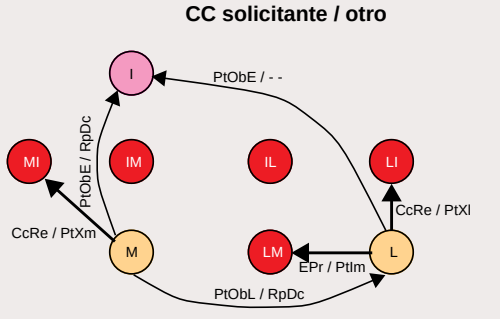


Figura 7.37 Protocolo de directorio MLI. CC: peticiones que puede recibir un bloque estando en un estado transitorio.

Identificación de cruces de peticiones

Para facilitar la comprensión en el siguiente desarrollo, en el margen izquierdo se replica la Figura 7.30.

Orden de procesado en el CM	
A	B
P2 store t P1 PtXl t	P2 store t P3 load t P1 PtXl t

Dado el orden A de accesos a memoria, el bloque en la cache C1 está en el estado LI cuando el CC1 procesa la petición PtObE, emitida por el CM al procesar la petición PtIm del CC2. El CC1 está esperando la notificación del CM para pasar el bloque al estado I. El CC1 no responde y el bloque permanece en el estado LI[29].

Orden de procesado en el CM	
C	D
P2 store t P1 store t	P2 store t P3 load t P1 store t

Dado el orden B de accesos a memoria el comportamiento en el CC1 es el mismo que en el orden A. El CM no emite ninguna petición al CC1 al procesar la petición del CC3.

Dado el orden C de accesos a memoria, el CC1 procesa la petición PtObE del CM, cuando el bloque está en el estado LM. Esta petición es para invalidar el bloque. Como la respuesta del CM a la petición del CC1 (PtIm) incluye el bloque, no hay cambio de estado del bloque en el CC1.

El comportamiento del CC1 en el orden D de accesos a memoria es el mismo que en el orden C. El CM al procesar la petición del CC3 no emite ningún mensaje al CC1. El bloque lo tiene en exclusividad el CC2.

Para facilitar la comprensión en el siguiente desarrollo, en el margen izquierdo se replica la Figura 7.32.

29. En el apartado de ejemplos se muestran los diagramas temporales para las distintas secuencias de acceso a memoria.

Orden de procesado en el CM	
E	F
P2 store t P1 PtXm t	P2 load t P1 PtXm t

Orden de procesado en el CM	
G	H
P2 store t P3 load t P1 PtXm t	P3 load t P2 store t P1 PtXm t

Dado el orden E de accesos a memoria, el CC1 procesa la petición PtObE del CM con el bloque en el estado MI. Esta petición es para suministrar e invalidar el bloque. El CC1 suministra el bloque y permanece en el estado MI, a la espera de la respuesta del CM a su petición.

Para identificar una posible secuencia de peticiones del CM se añade un nuevo estado transitorio denominado MII (Figura 7.38). Dado el orden F de accesos a memoria, el CC1 procesa la petición PtObL del CM con el bloque en el estado MI. Esta petición es para suministrar el bloque. El CC1 suministra el bloque y cambia a un nuevo estado transitorio, denominado MII, y espera la respuesta del CM a su petición. Este estado es para recordar que el CM ha ordenado una petición Pt antes que la petición PtXm del CC1.

Dado el orden G de accesos a memoria, el comportamiento del CC1 es el mismo que en el orden E. El CM al procesar la petición del CC3 no emite ningún mensaje al CC1. El bloque lo tiene en exclusividad el CC2.

Dado el orden H de accesos a memoria, el CC1 se comporta como en el orden F al procesar la petición PtObL del CM. Posteriormente, el CC1 recibe una petición PtObE. En el estado MII esta petición tiene como objetivo invalidar el bloque. Notemos que el CC1, al procesar la petición PtObL, ha suministrado el bloque. El CC1 al procesar la petición PtObE no modifica el estado del bloque y espera la respuesta del CM a su petición[30].

Resumen. En la Figura 7.38 se muestran los estados en los cuales se infiere un cruce y la respuesta del CC.

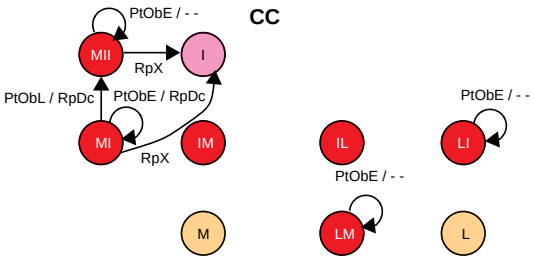


Figura 7.38 Protocolo MLI. Inferencia de cruces en un CC.

Diagramas completos de cruces de peticiones

En la Figura 7.39 se muestran las transiciones en un CC y el CM cuando se produce un cruce de peticiones a un bloque. En el CM se infiere un cruce cuando el CC, que ha emitido una petición PtXm o PtXl, no está en el VP del

30. En una sección previa de este capítulo se ha mostrado un diseño canónico y más ortodoxo. El CC en el estado MI puede recibir más de un tipo de petición del CM.

bloque. También, cuando se recibe una petición PtXm para un bloque que está en el estado L, la cual no es esperada. En ninguno de los casos hay que actualizar memoria.

Al procesar una petición, la primera acción en el CM es inferir si se ha producido un cruce de peticiones. Para ello, el CM analiza si el CC está en el VP. Además, en una petición PtXm, el CM analiza si el estado del bloque en el directorio se corresponde con la petición. En el diagrama de estados de un bloque en el directorio se utiliza el subíndice c para identificar una petición del CC que se ha cruzado con una petición del CM.

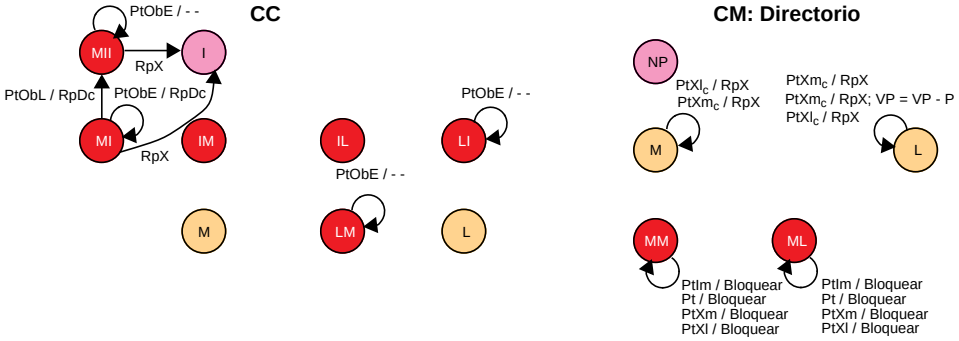


Figura 7.39 Diagramas de transiciones en un CC y en el directorio cuando se produce un cruce entre peticiones a un bloque entre un CC y el CM.

En un CC se infieren cruces en los estados transitorio LI, LM y MI. En cualquiera de ellos no se modifica el estado cuando se procesa una petición PtObE y sólo se responde con el bloque en el estado MI. En los estados LI y LM la petición PtObE sólo indica invalidación.

En el estado MI un CC también responde a una petición de bloque (PtObL) por parte del CM. Entonces, para identificar que no hay que suministrar el bloque en una petición PtObE posterior, se cambia al estado MII.

Observemos que el estado MI es similar al estado M cuando el CC procesa una petición PtObE. El siguiente estado estable debe ser I. En el caso del estado MI hay que esperar la respuesta del CM a la petición PtXm.

Cuando el CC procesa una petición PtObL, el estado MII es similar al estado LI. El siguiente estado estable debe ser I. Notemos que aunque la petición al CM sea PtXm, su única funcionalidad, al haber respondido a la petición PtObL del CM, una vez suministrado el bloque, es mantener el directorio preciso.

Tablas de estados y transiciones

En la Tabla 7.4 se muestran en formato tabla los estados y las transiciones entre estados de un bloque en una cache. Las casillas que no contienen información indican un error. En un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

			Eventos del procesador y reemplazo			Eventos externos (respuestas y peticiones			
			LPr	EPr	CcRe	RpD	RpX	PtObL	PtObE
Estados	Estables	I	Pt; IL	PtIm; IM					
		L	- - ; L	PtIm; LM	PtXI; LI				- - ; I
		M	- - ; M	- - ; M	PtXm; MI			RpDc: L	RpDc; I
	transitorios	IL				- - ; L			
		IM				- - ; M			
		LM				- - ; M			- - ; LM
		LI					- -; I		- - ; LI
		MI					- - ; I	RpDc; MII	RpDc; MI
		MII					- - ; I		- - ; MII

Tabla 7.4 Peticiones concurrentes. Protocolo de directorio MLI. Tabla de estados y transiciones en un bloque de cache. Las casillas con fondo blanco indican cruces de peticiones.

En la Tabla 7.5 se muestran en formato tabla los estados y transiciones entre estados de un bloque en el directorio. En el evento PtXI se distingue el caso de que el CC sea el único que está en el vector de presencia o haya más CC.

Para identificar un cruce, se comprueba si el CC solicitante están en el vector de presencia o no. Además se comprueba si la petición recibida es esperada en el estado actual.

Estados			Eventos del controlador de coherencia									
					PtXl			PtXm				
			Pt	PtIm	VP = P	VP ≠ P P ∈ VP	P ∉ VP	VP = P	VP ≠ P	P ∈ VP	RpDc
	Estables	NP	RpD; L, VP = P	RpD; M VP = P, BE = 1							
		L	RpD; L, VP = VP + P	PtObE {VP - P}, RpD; M, VP = P, BE = 1	RpX; NP, VP = vacio	RpX; L VP = VP - P	RpX; L	RpX; L		RpX; L VP = VP - P	
		M	PtObL; ML	PtObE {VP}; MM			RpX; M	RpX; NP, VP = vacío, BE = 0	RpX; M		
	transitorios	ML	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	RpD, Dev; L VP = VP + P, BE = 0
		MM	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	RpD; M VP = P, BE = 1

Tabla 7.5 Peticiones concurrentes. Protocolo de directorio MLI. Tabla de estados y transiciones de un bloque en el directorio. Las casillas con fondo blanco indican cruces de peticiones.

Representación de cambios de estado y transiciones en una secuencia de accesos a memoria

En este apartado se muestran tres formas de representar una secuencia de accesos a memoria: a) en formato tabla, b) mediante un diagrama temporal y c) mediante un diagrama temporal simplificado.

Representación en formato tabla

Los accesos a memoria se muestran en grupos separados mediante líneas horizontales continua. Cada grupo de accesos a memoria se gestiona independientemente y no se empieza a gestionar el siguiente grupo hasta que ha finalizado el anterior. Los accesos a memoria de un grupo se realizan concurrentemente en el mismo ciclo. En una fila de la tabla, de izquierda a derecha, después de la instrucción de acceso a memoria se especifica:

1 La primera columna se utiliza para representar el arbitraje de peticiones. Esto es, se indica el orden. Supondremos usualmente que en un grupo de peticiones el árbitro las selecciona en el orden en que se especifica la secuencia de accesos a memoria.

2 El siguiente grupo de columnas se utiliza para representar el estado transitorio del bloque, en la cache correspondiente, cuando se emite la petición.

3 El tercer grupo de columnas se utiliza para representar la petición de cada CC en la RI. La petición se indica en la casilla correspondiente y en el orden en el que el árbitro concede el acceso al módulo de memoria.

4 El siguiente grupo de columnas se utiliza para identificar la variable o bloque y el VP en el directorio. Los bits en el VP identifican a las caches; de izquierda a derecha en ordinal creciente. Un valor de uno en el VP indica que hay una copia del bloque en la cache correspondiente. El campo E se corresponde con el BE de la entrada en el directorio.

5 En el quinto grupo de columnas sólo se representan peticiones del CM que requieren una respuesta del CC. En la columna arb se indica el receptor del mensaje y en la columna RMC el mensaje de petición del CM.

6 El siguiente grupo de columnas se utiliza para representar la red que utilizan los CC para responder al CM, si es el caso (RCM). Se representa al receptor del mensaje (columna arb) y el mensaje (columna que identifica la red).

7 El séptimo grupo de columnas se utiliza para indicar qué elemento (memoria o cache) suministra el bloque o dato.

8 En el octavo grupo de columnas se representan las respuestas del CM y las peticiones que no requieren respuesta de los CC. En la columna arb se indica el destinatario. En la columna RV se indica la respuesta del CM. Si además hay mensajes de petición se utiliza la siguiente fila para indicarlos.

9 El noveno grupo de columnas se utiliza para identificar la variable accedida en cache y el estado del bloque al finalizar la acción de coherencia. Cuando un bloque está en una ventana de vulnerabilidad y recibe una petición del CM, se indica el estado transitorio del bloque. En estas condiciones, el estado transitorio, debido a que se ha atendido una petición del CM, se observa en las columnas de la derecha.

10 Si no se accede al CM la fila correspondiente a la petición se deja en blanco.

Ejemplo. En la Tabla 7.3 se muestra una secuencia de accesos a memoria realizada por tres procesadores. Los accesos a memoria están agrupados de dos en dos. Cuando se inicia la secuencia de accesos a memoria, las caches no almacenan los bloques en los que se ubican las variables accedidas. Tampoco se producen conflictos de contenedor al almacenar los bloques en los contenedores de cache.

El primer par de accesos a memoria son debidos a un fallo de lectura y a un fallo de escritura. El estado transitorio de los bloques al emitir las peticiones es IL en la cache C1 (Pt) e IM en la cache C2 (PtIm). En las dos transacciones el

bloque lo suministra memoria. El estados de los bloques al finalizar las transacciones es L en la cache C1 y M en la cache C2. En el directorio ha sido activado el BE del bloque que contiene la variable u.

	arb.	C 1	C 2	C 3	RI		mem.			arb.	Red	arb.	Red		arb.	RV		C 1		C 2		C 3	
acceso	ord.	est.	est.	est.	1º	2º	var.	VP	E	ord.	RMC	ord.	RCM	sum.	ord.	1º	2º	var.	est.	var.	est.	var.	est.
1. P1 load t	P1	IL			Pt		t	1, 0, 0	0					mem	CC1	RpD		t	L				
1. P2 store u	P2		IM			PtIm	u	0, 1, 0	1					mem	CC2		RpD			u	M		
2. P1 load u	P1	IL			Pt		u	1, 1, 0	0	C2	PtObL	CM	RpDc		CC1	RpD		u	L	u	L		
2. P3 load u	P3			IL		Pt	u	1, 1, 1	0					mem	CC3		RpD					u	L
3. P1 load w	P1	IL			Pt		w	1, 0, 0	0					mem	CC1	RpD		w	L				
3. P2 store u	P2		LM			PtIm	u	0, 1, 0	1					mem	CC2		RpD			u	M		
															CC1,3		PtObE	u	I			u	I
4. P1 load w																							
4. P3 load w	P3			IL	Pt		w	1, 0 ,1	0					mem	CC3	RpD						w	L
5. P2 store w	P2		IM		PtIm		w	0, 1, 0	1					mem	CC2	RpD				w	M		
															CC1,3	PtObE		w	LM			w	I
5. P1 store w	P1	LM				PtIm	w	1, 0, 0	1	C2	PtObE	CM	RpDc		CC1	RpD		w	M	w	I		

Tabla 7.6 Protocolo de directorio MLI. Secuencia de accesos a memoria concurrentes.

El segundo par de accesos a memoria son fallos de lectura. El CM procesa en primer lugar el mensaje emitido por el CC1. El procesado del mensaje requiere que el CM solicite el bloque a la cache C2, mediante una petición de observación de lectura (PtObL). El segundo fallo hace referencia al mismo bloque. Como memoria ha sido actualizada en la transacción previa, el CM suministra directamente el bloque. Al finalizar las dos transacciones el VP del bloque identifica que las caches C1, C2 y C3 tienen copia del bloque.

El tercer grupo de accesos a memoria requiere dos transacciones. La primera petición es de lectura (Pt) y el bloque lo suministra la memoria. La segunda petición es de lectura con intención de modificación (PtIm) y el bloque lo suministra la memoria. Adicionalmente, el CM emite peticiones de observación de escritura del bloque (PtObE) a las caches C1 y C3. El estado estable del bloque, al finalizar la transacción, es M en la cache C2 e I en las caches C1 y C3.

El cuarto par de accesos a memoria requiere sólo una transacción. El primer acceso a memoria es un acierto en cache. El segundo acceso a memoria, accede al mismo bloque, pero es un fallo. Al finalizar la transacción, el estado del bloque que contiene la variable w es L en las caches C1 y C3.

El último grupo de accesos a memoria son dos escrituras. Ninguna de las caches accedidas tiene el bloque en exclusividad. La cache C2 no tiene copia del bloque y la cache C1 tiene una copia en el estado L. La petición de CC2 requiere que el CM emita una petición de observación de escritura al CC1 y al CC3. Posteriormente, el CM, al procesar la petición de CC1, emite un mensaje de petición de observación de escritura al CC2, el cual suministra el bloque al CM. Finalmente el CM suministra el bloque a CC1. Al finalizar las dos transacciones, el VP del bloque en el directorio indica que la cache C1 tiene copia del bloque en exclusividad.

Diagrama temporal

Antes de utilizar una red para transmitir el mensaje se representa el arbitraje. Esta fase se utiliza para mostrar la ordenación de los mensajes. La espera para acceder a una red se indica representando en ciclos consecutivos la fase de arbitraje (arb). El resto se representa de la forma descrita en el Capítulo 5.

Ejemplo. En la Figura 7.40 se muestra el diagrama temporal de la secuencia de accesos mostrada en la Tabla 7.6. Observemos el bloqueo del procesado de peticiones en el CM en el segundo grupo de peticiones. La primera petición que procesa el CM requiere que la cache C2 suministre el bloque. Durante el intervalo de tiempo, entre la petición del CM y la respuesta del CC2, el CM no procesa la siguiente petición en la CP, ya que hace referencia al mismo bloque.

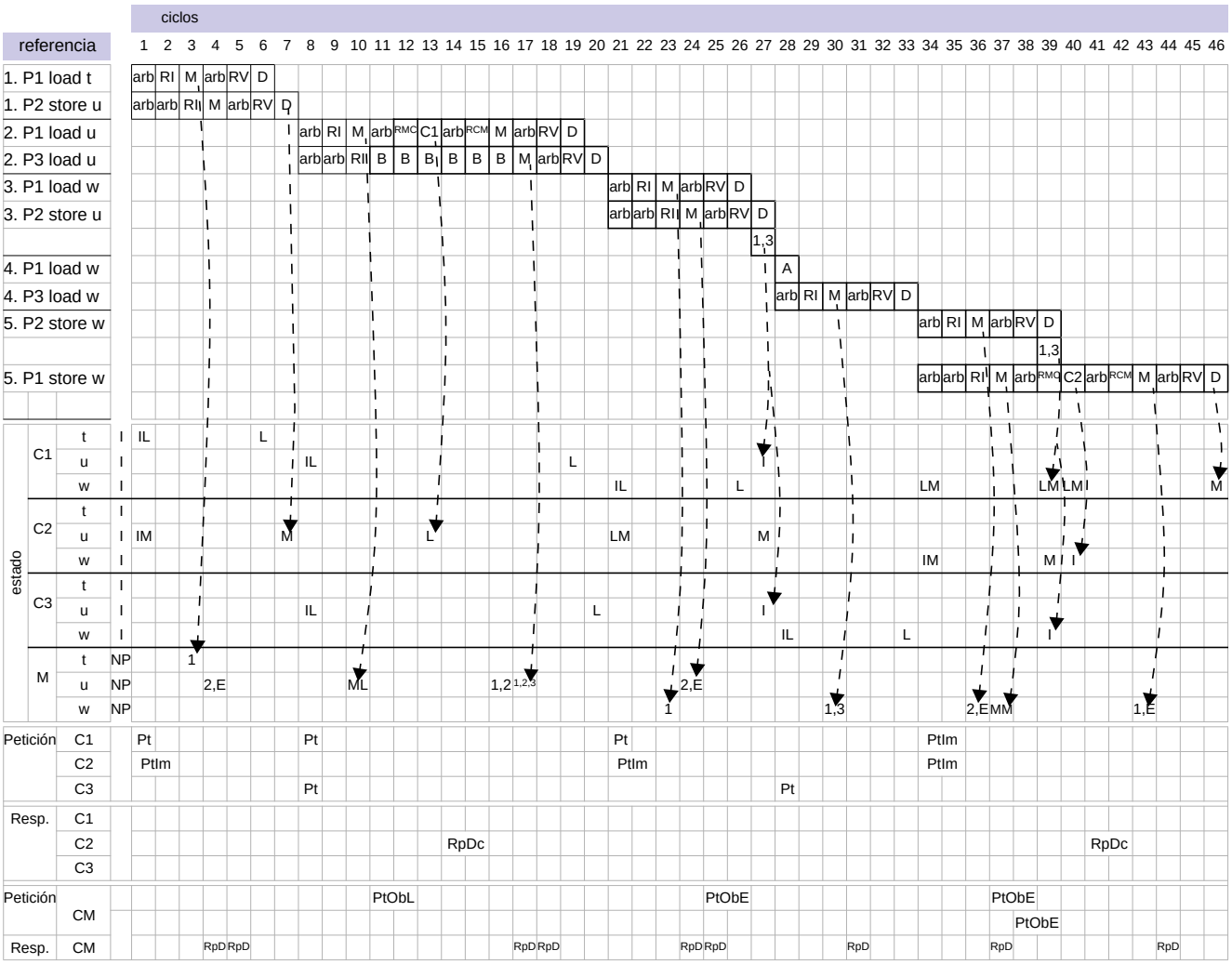


Figura 7.40 Protocolo de directorio MLI. Diagrama temporal de una secuencia de accesos a memoria concurrentes.

Diagrama temporal simplificado

En la Figura 7.41 se muestra el diagrama temporal simplificado de la secuencia de accesos mostrada en la Tabla 7.6. El bloqueo del procesado de una petición en el CM se indica mediante una línea curva a trazos.

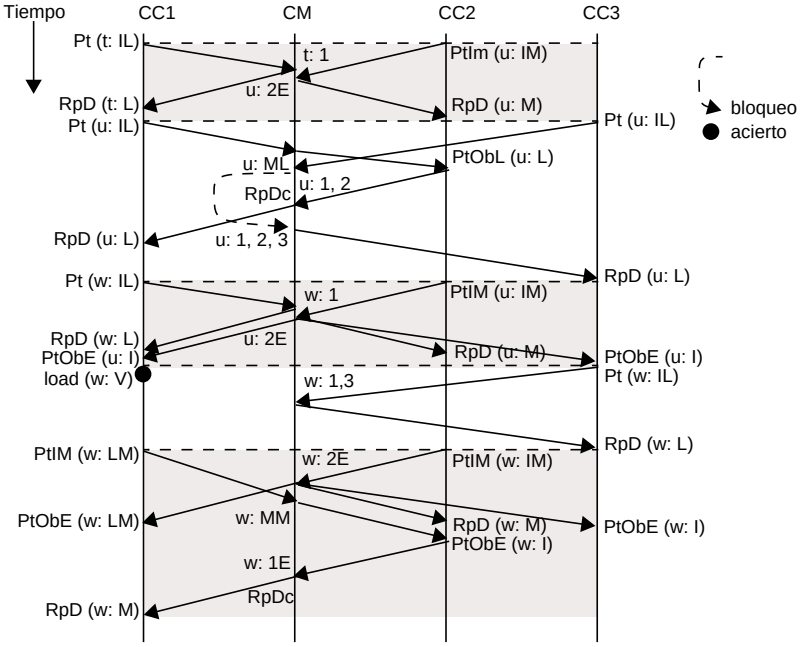


Figura 7.41 Protocolo de directorio MLI. Diagrama temporal simplificado de una secuencia de accesos a memoria concurrentes.

PARALELISMO

En un multiprocesador el tráfico de peticiones es proporcional al número de procesadores. Para incrementar el rendimiento es necesario que la memoria pueda servir varias peticiones en cada ciclo. Aunque el directorio conceptualmente es una estructura centralizada, puede distribuirse añadiendo más módulos de memoria con un CM y directorio asociado.

En este apartado se incrementa el paralelismo, utilizando varios módulos de memoria y distribuyendo el directorio, con el objetivo de incrementar el número de peticiones procesadas por unidad de tiempo o ancho de banda.

Organización del multiprocesador

El paralelismo en el procesado de peticiones se obtiene utilizando varios módulos de memoria, cada uno con un CM, que se entrelazan por bloque[31]. El directorio de cada CM gestiona los bloques ubicados en el módulo de memoria. Esta organización facilita que se distribuya el tráfico de peticiones entre los módulos de memoria y el sistema es escalable en cierta medida. En el mejor caso se podrán estar sirviendo tantas peticiones de los CC como módulos de memoria, si hay menos módulos de memoria que CC.

En la Figura 7.42 se muestra un esquema de un multiprocesador con varios módulos de memoria. La RI y la RV son redes crossbar[32]. La RV se utiliza para transmitir peticiones y respuestas de los CM (las redes lógicas RV y RMC son la misma red). En la misma figura se muestra el detalle del conexionado que permite que los procesadores envíen las peticiones hacia un módulo de memoria. Si en un mismo ciclo las peticiones de los procesadores son a módulos de memoria distintos, la red crossbar permite que se sirvan todas las peticiones. Cuando es necesario para el protocolo de coherencia, hay una red de respuestas de los CC a los CM (red RCM). Esta red también es un crossbar.

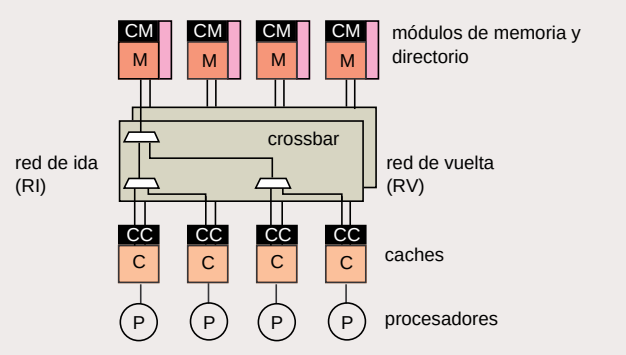


Figura 7.42 Multiprocesador con varios módulos de memoria.

En un ciclo determinado, un CM puede emitir una respuesta (RpC) y varias peticiones (PtObE)[33]. También, en el mismo ciclo, otro CM puede realizar la misma acción. Además, la intersección de los mensajes emitidos por los dos CM puede ser distinta del conjunto vacío. Por tanto, es necesario un algoritmo de arbitraje en la RV. Este algoritmo debe ser tal que permita que todos los CC

31. Bloques consecutivos del espacio físico se almacenan en módulos de memoria distintos. La granularidad del entrelazado puede ser mayor. Por ejemplo, una página del espacio físico de direcciones.
32. Podemos observar una red crossbar como un grupo de buses independientes. Cada uno de ellos permite la conexión a un módulo de memoria.
33. Por ejemplo, en el protocolo de directorio VI. En una petición de escritura, el CM emite una respuesta y puede emitir varias peticiones de observación de escritura si hay copias del bloque en otras caches.

observen el mismo orden lógico global de los mensajes emitidos en cada transacción desde los CM a los CC. Esto es, la RV debe ser ordenada. Esta característica permite eliminar la necesidad de que las peticiones de observación de escritura (PtObE) deban responderse.

En la Figura 7.43 se muestra, mediante un ejemplo, la necesidad de un algoritmo de arbitraje en la RV, que garantice que los CC observen el mismo orden lógico global. Los dos trozos de código son el esqueleto del algoritmo de Dekker para exclusión mutua. En el desarrollo utilizamos un protocolo de directorio VI.

Las variables aviso1 y aviso2 están contenidas en bloques que se ubican en los módulos de memoria M1 y M2 respectivamente. Antes de iniciarse la ejecución, la cache C1 tiene copia del bloque que contiene la variable aviso2 y la cache C2 tiene copia del bloque que contiene la variable aviso1. El valor de las dos variables es cero.

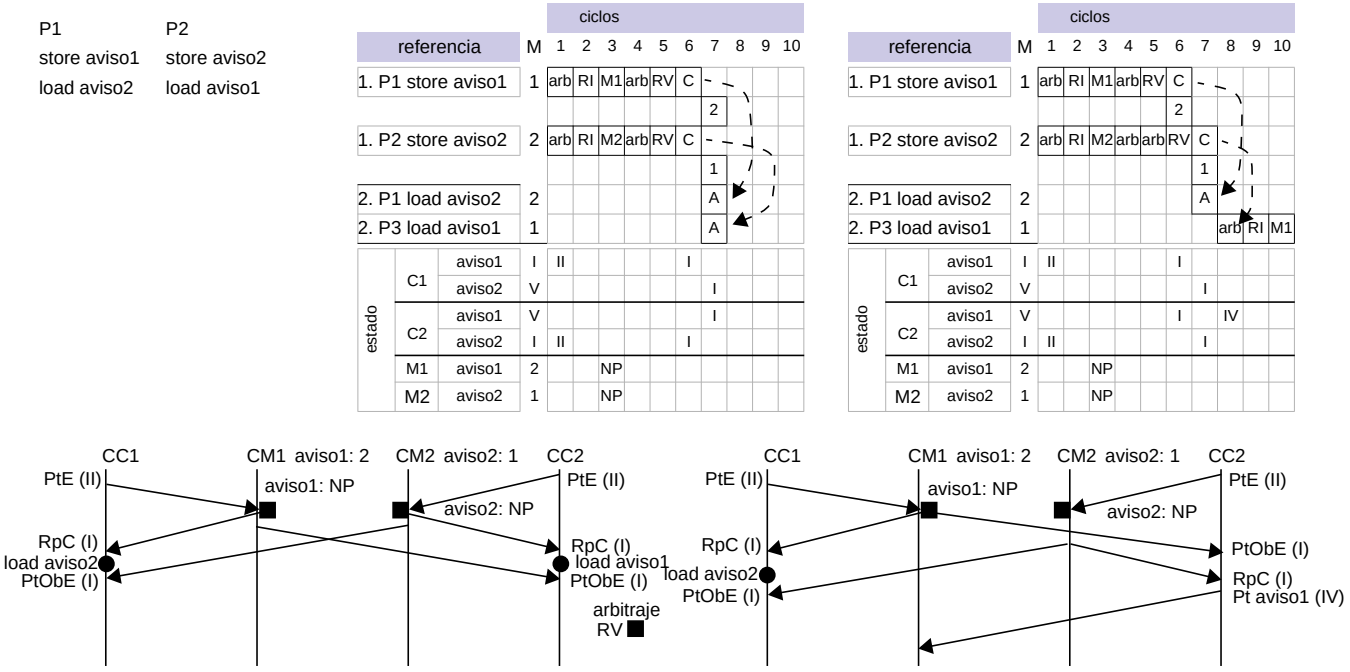


Figura 7.43 Protocolo de directorio VI. Ejemplos de arbitraje en la RV. El acrónimo M indica módulo de memoria.

Las instrucciones store de los procesadores P1 y P2 son fallos en cache. Las peticiones que emiten CC1 y CC2 acceden a CM distintos. Por tanto progresan de forma paralela hasta el CM correspondiente. El CM1 responde al CC1 y

efectúa una petición al CC2. El CM2 responde al CC2 y efectúa una petición al CC1. La intersección de los conjuntos de mensajes de los dos CM es distinta del conjunto vacío. Por tanto, es necesario un arbitraje que serialice la utilización de la RV.

En la parte izquierda de la Figura 7.43 se muestra el caso de que el arbitraje conceda las conexiones a las dos respuestas. Suponemos que mientras se reciben las peticiones de observación de escritura (PtObE) los procesadores ejecutan las instrucciones load, ya que para cada uno de ellos ha finalizado la ejecución de la instrucción store (flecha en la Figura 7.43)[34], aunque no han sido globalmente consolidadas. En los dos procesadores el acceso a memoria es un acierto en cache y el valor leído en los dos casos es cero. Este valor permite que los dos procesadores accedan a la sección crítica.

Los procesadores observan las escrituras en un orden distinto. El procesador P1 al recibir la respuesta ejecuta la siguiente instrucción. El procesador P2 también ejecuta la siguiente instrucción al recibir la respuesta, pero el procesador P2 no ha observado la escritura de P1. El razonamiento intercambiando los procesadores es el mismo.

En la parte derecha de la Figura 7.43 se muestra el caso de que el arbitraje conceda en primer lugar el acceso a la RV a los mensajes que emite M1. En este caso el procesador P1 accede a la sección crítica. Los dos procesadores observan las escrituras en el mismo orden.

Crossbar de vuelta y arbitraje. En la Figura 7.44 se muestra un posible diseño de la RV. Desde cada CM existe una conexión punto a punto con cada CC. En cada CC existe un multiplexor cuyas entradas son las conexiones con los CM. Notemos que un CM está conectado en todos los multiplexores en la entrada identificada con el mismo ordinal.

Una entrada determinada, de un multiplexor, tiene la misma prioridad relativa en todos los multiplexores respecto a las otras entradas del multiplexor. Esto es, la prioridad de la conexión de un CM es la misma en todos los CC.

Entonces, dado un ciclo en el que hay peticiones o respuestas estas se arbitran. No se tiene en cuenta ninguna nueva solicitud de transmisión de mensajes hasta que se han arbitrado todas las que había en el ciclo previo. En un ciclo se distinguen subciclos de arbitraje hasta que se han arbitrado todas las peticiones o respuestas[35].

34. Tengamos en cuenta que una fase de una transacción son varios ciclos de procesador. También, el acceso del agente procesador puede ser prioritario respecto del agente observador. Los mensajes recibidos se pueden almacenar en una cola.
35. En este contexto los CM se bloquean hasta que sus mensajes han sido transmitidos.

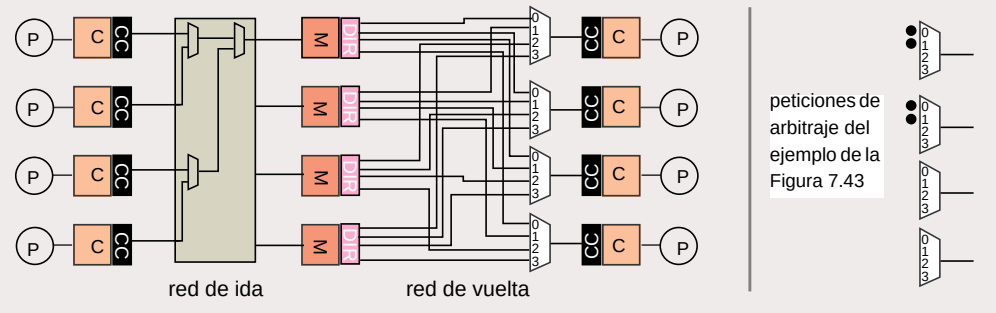


Figura 7.44 Esquema de la red de vuelta.

Supongamos que la entrada cero de los multiplexores es la más prioritaria. Entonces, en el primer subciclo la salida de los multiplexores es la entrada cero, en el segundo subciclo la entrada uno y así de forma sucesiva[36]. De esta forma se garantiza que todos los CC observan las peticiones o respuestas de los CM en el mismo orden (red ordenada). En la parte derecha de la Figura 7.44 se muestran las peticiones paralelas de la Figura 7.43.

Optimización. Notemos que con el funcionamiento descrito, en un subciclo puede haber peticiones o respuestas en las entradas de un multiplexor y no existir ninguna en la salida. Esto es debido a que el sincronismo en la selección de la entrada determina que la entrada elegida no tenga petición o respuesta. Una mejora en el rendimiento es que en cada subciclo se seleccione, partiendo de la entrada más prioritaria, la primera entrada con petición. Notemos que si un CM ha emitido varios peticiones hacia los CC y existen peticiones o respuestas de otros CM, la peticiones o respuestas de un CM no tienen porqué ser observadas en el mismo ciclo en los CC, pero si serán observadas en el mismo orden global. El tiempo físico en el cual una petición de un CM llega a un CC no es importante, mientras el orden global en el que todos los CC del multiprocesador observen una petición sea el mismo.

Arbitraje sin sesgo. Un arbitraje con prioridad fija como el descrito previamente favorece siempre a los mismos CM. Una posibilidad, para reducir el sesgo, es que en ciclos pares la entrada más prioritaria del multiplexor sea la que tiene el menor ordinal. En ciclos impares la entrada más prioritaria es la que tiene el mayor ordinal.

36. Este funcionamiento es similar al caso de un multiprocesador con varios buses y un protocolo de observación (Capítulo 4). Para disponer de consistencia secuencial los buses deben ser observados en el mismo orden por todos los CC. En el contexto de buses se dispone de un buffer donde se almacenan las observaciones inducidas por la transacciones que transportan los buses. En particular, en un protocolo VI sólo se inducen peticiones de invalidación. Ahora bien, para utilizar la respuesta de una petición del CC (propia) el buffer debe vaciarse (orden en el bus).

Incremento del rendimiento. Para incrementar el rendimiento se pueden utilizar colas en las entradas de los multiplexores, una para cada conexión con un CM. Ahora bien, para mimetizar el diseño sin colas, la misma entrada en todas las colas de todos los multiplexores representa un ciclo de peticiones. En un ciclo determinado, todos los CM que emiten una petición o respuesta la ubican en la misma entrada de la cola correspondiente. Entonces, el arbitraje descrito previamente se utiliza para arbitrar todas las entradas asociadas al mismo ciclo.

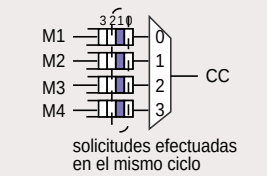


Figura 7.45 Cola de mensajes ordenados en las entradas de un multiplexor.

Una mejora es, en un multiplexor determinado, pasar a arbitrar el siguiente ciclo cuando no hay peticiones o respuestas pendiente de arbitrar en el ciclo actual. Entonces, en un multiplexor no se empieza a procesar el siguiente ciclo de peticiones o respuestas hasta que se han procesado todas las peticiones o respuestas del ciclo previo. En estas condiciones, un multiplexor puede estar en un ciclo par y otro multiplexor en un ciclo impar.

Ejemplo de diseño. El diseño previo se puede implementar cerca de los módulos de memoria y utilizarlo exclusivamente para garantizar un orden lógico global. Notemos que en un subciclo determinado sólo se selecciona una entrada en cada multiplexor. Por tanto, el diseño descrito se puede utilizar como algoritmo de arbitraje en una red crossbar que transporte la información desde los CM a los CC.

En la Figura 7.46 se muestra un esquema de diseño de la red crossbar desde los CM a un CC. El esquema para la conexión de los CM a cada uno de los otros CC es idéntico. Los CM efectúan peticiones al árbitro. El árbitro en cada ciclo almacena las peticiones efectuadas en una entrada de la cola de peticiones de arbitraje (CPA). Esto es, cada entrada almacena las peticiones efectuadas por los CM en ese ciclo.

La CPA se gestiona de forma FIFO. Dada una entrada en la CPA se conceden todas las peticiones antes de pasar a la siguiente entrada. El vector de concesiones (c0, c1, c2, c3) en cada ciclo sólo tiene un bit activo. Cada bit se corresponde con la concesión del arbitraje a un CM (el bit Ci se corresponde con la petición del CMi). Un bit de concesión se utiliza para controlar un multiplexor. Notemos que podemos considerar que cada multiplexor, menos el último, está

asociado a un CM. El último multiplexor se controla con el bit del CM asociado y la función OR de los grupos de bits de concesión de los CM con ordinal menor y los grupos de bits con ordinal mayor.

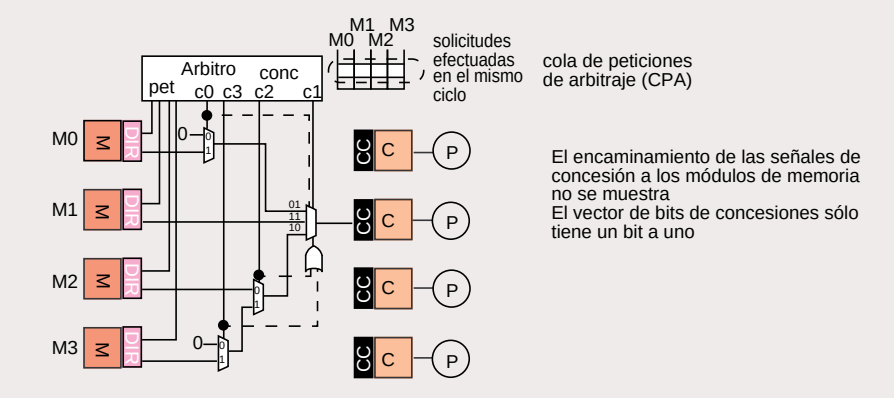


Figura 7.46 Crossbar: esquema de arbitraje y encaminamiento de los CM a un CC.

En los diagramas temporales que se utilicen, para simplificar el arbitraje, supondremos que los árbitros conceden el acceso a la red a todos los mensajes emitidos desde un CM o a ninguno. Este funcionamiento no prohíbe que accedan a la red mensajes emitidos por varios CM mientras la intersección sea nula. En la Figura 7.47 se muestra un ejemplo donde hay que arbitrar en la RV. Los arbitros conceden el acceso en primer lugar a M1 y después a M2.

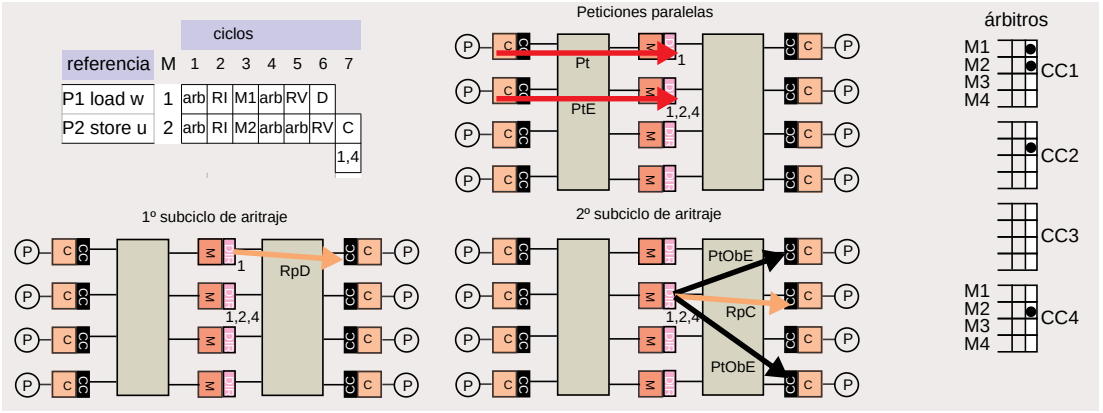


Figura 7.47 Protocolo de directorio VI. Arbitraje en la RV.

En la Figura 7.48 se muestra otro ejemplo de arbitraje. La primera transacción es de 4 pasos y la segunda transacción es de dos pasos. El destinatario de la petición de CM inducida por la primera transacción también es el destinatario de un mensaje de la segunda transacción servida por otro CM. Recor-

demos que las redes lógicas RMC y RV son la misma red. Por tanto, hay que arbitrar. El árbitro concede el arbitraje a la petición inducida por la primera transacción.

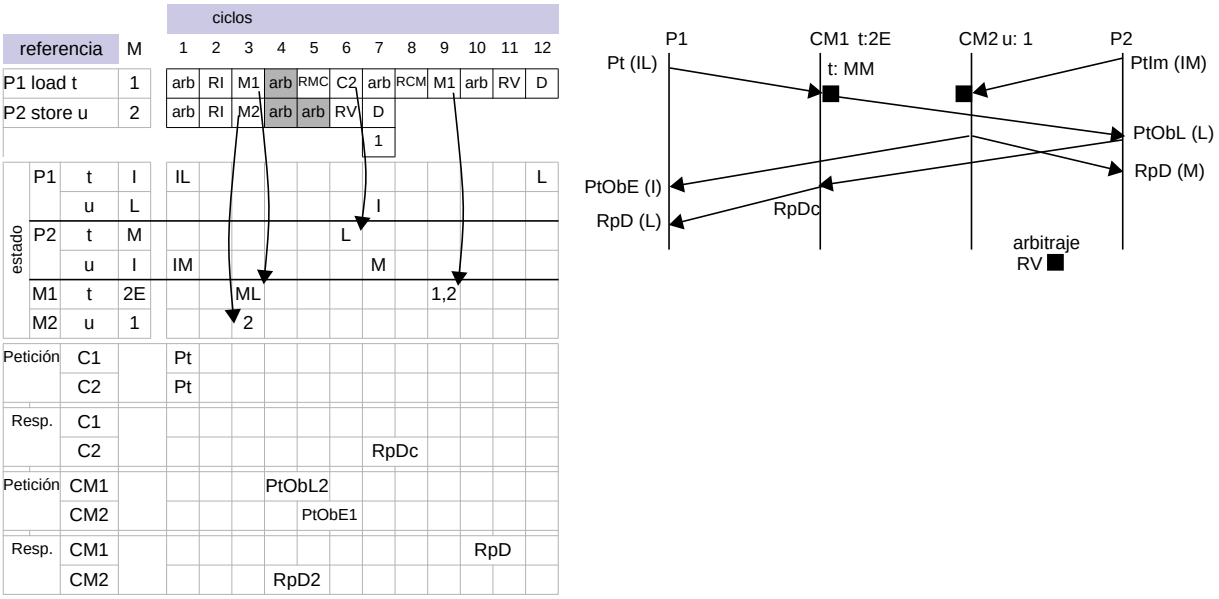


Figura 7.48 Protocolo de directorio MLI. Arbitraje en la RV.

Coherencia. El CM de cada módulo de memoria es el punto de ordenación de las transacciones que referencian los bloques que almacena. En cada CM se propagan y serializan las transacciones de escritura de cada uno de los bloques que gestiona. La RV mantiene, mediante el arbitraje, un orden lógico global de los mensajes emitidos por los CM a los CC. Una escritura es observada, por todos los procesadores involucrados, en el mismo orden global. Por tanto, todos los CC observan las peticiones de observación de escritura a un bloque, efectuadas por un CM, en el mismo orden y no es necesario que los CC respondan a las peticiones de invalidación.

Consistencia. Los CM y el arbitraje en la RV determinan un orden lógico global de todas las transacciones. Una escritura está consolidada cuando el CC recibe la respuesta del CM. Una petición de lectura lee el valor establecido por la instrucción store previa a la misma dirección.

Protocolo de directorio VI

Los diagramas de transiciones entre estados en el CM y en un CC son los descritos previamente en este capítulo.

Organización

En la Figura 7.49 se muestra la organización del multiprocesador.

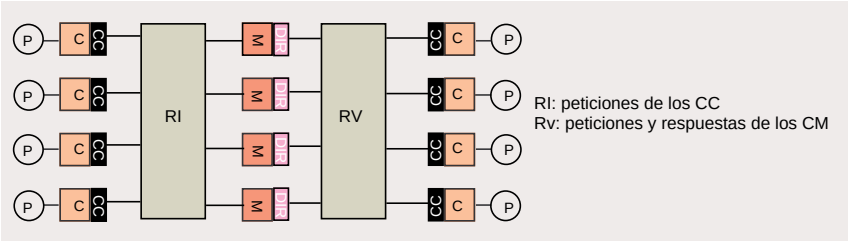


Figura 7.49 Protocolo VI. Redes de comunicación.

Representación de cambios de estado y transiciones en una secuencia de accesos a memoria

En este apartado se muestran tres formas de representar una secuencia de accesos a memoria: a) en formato tabla, b) mediante un diagrama temporal y c) mediante un diagrama temporal simplificado.

Representación en formato tabla. La representación es similar a la utilizada con un módulo de memoria. Seguidamente se indican las diferencias.

1 Se añade una primera columna para indicar el ordinal del módulo de memoria accedido en la transacción.

2 En arbitraje de la RI se distinguen varias columnas. El número de columnas en el peor caso es el número de accesos a memoria incluidos en el mismo grupo. En la casilla de la columna que indica el orden se especifica el módulo de memoria al que se accede. A peticiones de acceso realizadas al mismo módulo de memoria, el árbitro les concede el acceso en ciclos consecutivos. La concesión es en el orden en que se especifican en la secuencia de accesos a memoria.

3 En la red de vuelta (RV) hay que tener encuentra lo siguiente: si las peticiones se han servido en el mismo ciclo hay que comprobar que las respuestas y peticiones que generan los CM tengan intersección nula. Esto es, dos mensajes de distintos CM no se pueden enviar al mismo CC. Si la intersección es nula pueden ocupar la red de vuelta en el mismo ciclo. La concesión es en el orden en que se especifican en la secuencia de accesos

a memoria. En la columna correspondiente se indica el mensaje de respuesta. Si además hay mensajes de petición se utiliza la siguiente fila para indicarlos.

Ejemplo. En la Tabla 7.7 se muestra una secuencia de accesos a memoria realizada por tres procesadores. Los accesos a memoria están agrupados de dos en dos.

Las variables t, u, w están ubicadas en bloques distintos de memoria. Los bloques que contienen las variables t y w se almacenan en el módulo de memoria M1. El otro bloque se almacena en el módulo de memoria M2. Al almacenar los bloques en los contenedores de caches no se producen conflictos. Inicialmente no hay copia de los bloques en las caches.

En el primer grupo de accesos a memoria, las peticiones emitidas por los CC se encaminan a CM distintos y estos, al procesar la petición, sólo emiten un mensaje de respuesta al CC que ha emitido la petición. Por tanto, existe paralelismo.

		arb.		C 1	C 2	C 3	RI		mem.			arb.	RV		C 1		C 2		C 3	
acceso	M	1º	2º	est.	est.	est.	1º	2º	var.	VP	sum.	ord.	1º	2º	var.	est.	var.	est.	var.	est.
1. P1 load t	1	M1		IV			Pt		t	1, 0, 0	mem.	CC1	RpD		t	V				
1. P2 store u	2	M2			II		PtE		u	0, 0, 0	C2	CC2	RpC				u	I		
2. P1 load u	2	M2		IV			Pt		u	1, 0, 0	mem.	CC1	RpD		u	V				
2. P3 load u	2		M2			IV		Pt	u	1, 0, 1	mem.	CC3		RpD					u	V
3. P1 load w	1	M1		IV			Pt		w	1, 0, 0	mem	CC1	RpD		w	V				
3. P2 store u	2	M2			II		PtE		u	0, 0, 0	C 2	CC2		RpC			u	I		
												CC1,3		PtObE 1,3	u	I			u	I
4. P1 load w	1																			
4. P3 load w	1	M1				IV	Pt		w	1, 0 ,1	mem	CC3	RpD						w	V
5. P2 store w	1	M1			II		PtE		w	0, 0, 0	C 2	CC2	RpC							
												CC1,3	PtObE 1,3		w	VVI			w	I
5. P1 store w	1		M1	V V				PtE	w	0, 0, 0	C1	CC1		RpC	w	I				

Tabla 7.7 Protocolo de directorio VI. Paralelismo en la secuencia de accesos.

En el segundo grupo de accesos a memoria, las peticiones emitidas por los CC requieren acceder al mismo CM. Por tanto, son serializada por el árbitro en la RI.

La peticiones del tercer grupo de accesos a memorias acceden a CM distintos. Hasta el procesado en los CM existe paralelismo. Sin embargo, la intersección de los mensajes emitidos por los dos CM es distinta del conjunto vacío. En consecuencia deben serializarse. En primer lugar el árbitro concede la RV al CM1 y posteriormente al CM2.

En el cuarto grupo de accesos a memoria, el acceso del procesador P1 es acierto en cache.

En el quinto grupo de accesos a memoria, las peticiones emitidas por los CC acceden al mismo CM. Por tanto, el arbitraje serializa la utilización de la RI.

Diagrama temporal

Después de utilizar la RI se representa el acceso a memoria, indicando como sufijo el ordinal del módulo de memoria accedido. Antes de utilizar una red para transmitir el mensaje o los mensajes se representa el arbitraje. Esta fase se utiliza para mostrar la ordenación de los mensajes. La espera para acceder a una red se indica representando en ciclos consecutivos la fase de arbitraje (arb). La concesión es en el orden en que se especifican en la secuencia de accesos a memoria. El resto se representa de la forma descrita previamente en este Capítulo.

Ejemplo. En la Figura 7.50 se muestra el diagrama temporal de la secuencia de accesos mostrada en la Tabla 7.7. Observemos el arbitraje en la RV en el tercer grupo de accesos a memoria.

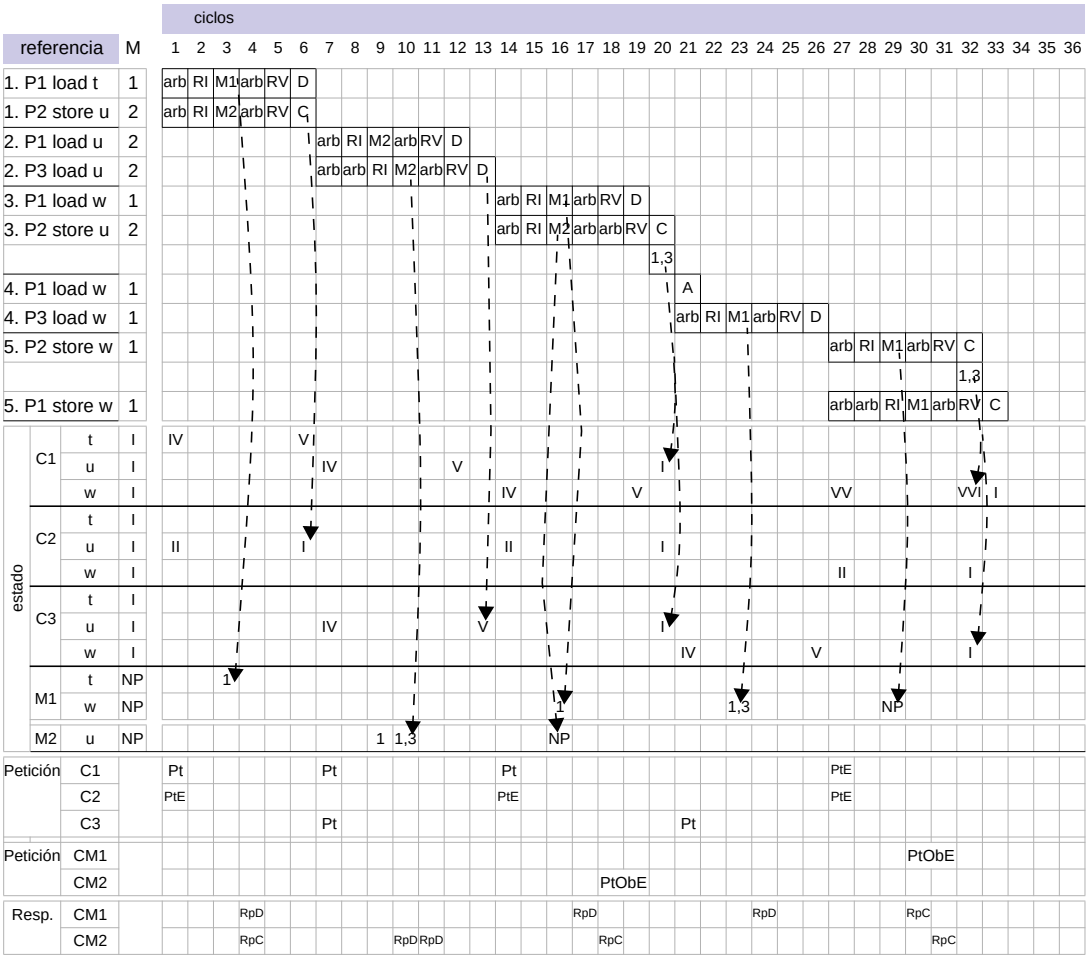


Figura 7.50 Protocolo de directorio VI. Diagrama temporal de una secuencia de accesos paralelos a memoria.

Diagrama temporal simplificado

En la Figura 7.51 se muestra el diagrama temporal simplificado del flujo de mensajes, en los dos sentidos, entre los CC y los CM. En particular se muestra el arbitraje en las distintas redes. La secuencia de accesos es la mostrada en la Tabla 7.7.

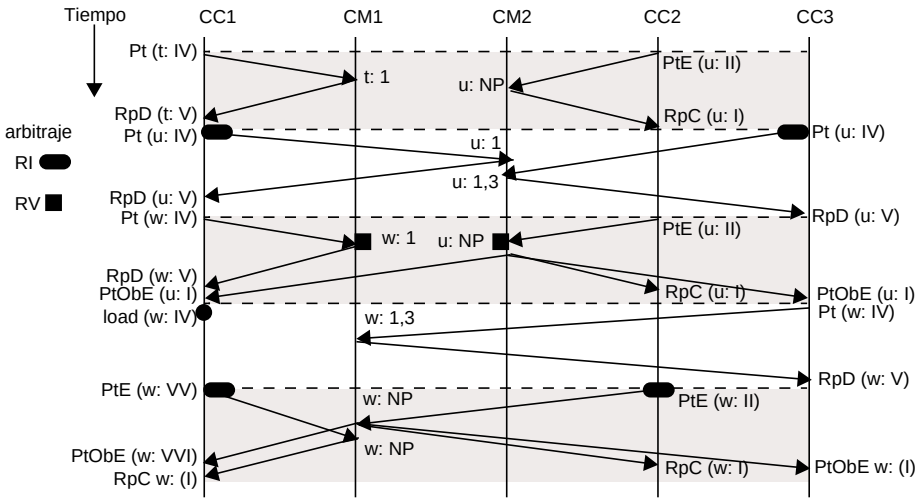


Figura 7.51 Protocolo de directorio VI. Diagrama temporal simplificado de una secuencia de accesos paralelos.

Protocolo de directorio MLI

Los diagramas de transiciones entre estados en el CM y en un CC son los descritos previamente en este capítulo.

Organización del multiprocesador

En la Figura 7.52 se muestra la organización del multiprocesador.

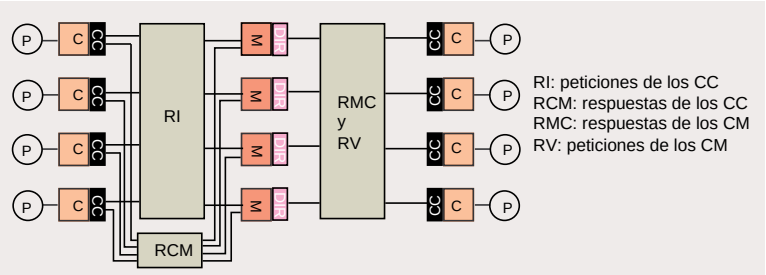


Figura 7.52 Protocolo MLI. Redes lógicas de comunicación.

Representación de cambios de estado y transiciones en una secuencia de accesos a memoria

Representación en formato tabla. La representación es similar a la utilizada con un módulo de memoria. Seguidamente se indican las diferencias.

1 Se añade una primera columna para indicar el ordinal del módulo de memoria accedido en la transacción.

2 En arbitraje de la RI se distinguen varias columnas. El número de columnas en el peor caso es el número de accesos a memoria incluidos en el mismo grupo. En la casilla de la columna que indica el orden se especifica el módulo de memoria al que se accede. A peticiones de acceso realizadas al mismo módulo de memoria, el árbitro les concede el acceso en ciclos consecutivos. La concesión es en el orden en que se especifican en la secuencia de accesos a memoria.

3 En la red RMC la etiqueta arb tiene varias columnas. En ellas se representa el destinatario de la petición de cada CM. En la columna etiquetada como RMC se indica el mensaje emitido por el CM. Si hay peticiones concurrentes con el mismo destinatario hay que arbitrar[37].

4 En la red RCM la etiqueta arb tiene varias columnas. En ellas se representa el destinatario de la petición del CC. En la columna etiquetada como RCM se indica el mensaje emitido por el CM. Si hay peticiones concurrentes con el mismo destinatario hay que arbitrar.

5 El siguiente grupo de columnas corresponde a la RV. En las columnas etiquetadas como arb hay que indicar el destinatario. En la columna correspondiente, etiquetada como RV, se indican el mensaje de respuesta. Si además hay mensajes de petición se utiliza la siguiente fila para indicarlos.

Ejemplo. En la Tabla 7.8 se muestra una secuencia de accesos a memoria realizada por tres procesadores. Los accesos a memoria están agrupados de dos en dos.

Las variables t, u, w están ubicadas en bloques distintos de memoria. Los bloques que contienen las variables t y w se almacenan en el módulo de memoria M1. El otro bloque se almacena en el módulo de memoria M2. Al almacenar los bloques en los contenedores de caches no se producen conflictos. Inicialmente no hay copia de los bloques en las caches.

Las dos primeras instrucciones emiten peticiones que acceden a módulos de memoria distintos. Por tanto, se procesan en paralelo.

37. Recordemos que las redes lógicas RMC y RV son la misma red.

		arb.		C 1	C 2	C 3	RI		mem.			arb		Red	arb		Red		arb.		RV		C 1		C 2		C 3	
acceso	M	1º	2º	est.	est.	est.	1º	2º	var.	VP	E	1º	2º	RMC	1º	2º	RCM	sum.	1º	2º	1º	2º	var.	est.	var.	est.	var.	est.
1. P1 load t	1	M1		IL			Pt		t	1, 0, 0	0							mem	CC1		RpD		t	L				
1. P2 store u	2	M2			IM		PtIm		u	0, 1, 0	1							mem	CC2			RpD			u	M		
2. P1 load u	2	M2		IL			Pt		u	1, 1, 0	0	P2		PtObL2	M2		RpDc	C2	CC1		RpD		u	L	u	L		
2. P3 load u	2		M2			IL		Pt	u	1, 1, 1	0							mem		CC3		RpD					u	L
3. P1 load w	1	M1		IL			Pt		w	1, 0, 0	0							mem	CC1		RpD		w	L				
3. P2 store u	2	M2			LM		PtIm		u	0, 1, 0	1							mem		CC2		RpD			u	M		
																				CC1,3		PtObE	u	I			u	I
4. P1 load w	1																											
4. P3 load w	1	M1				IL	Pt		w	1, 0 ,1	0							mem	CC3		RpD						w	L
5. P2 store w	1	M1			IM		PtIm		w	0, 1, 0	1							mem	CC2		RpD				w	M		
																			CC1,3		PtObE		w	IM			w	I
5. P1 store w	1		M1	LM				PtIm	w	1, 0, 0	1	P2		PtObE2	M1		RpDc	C2		CC1		RpD	w	M	w	I		

Tabla 7.8 Protocolo de directorio MLI. Paralelismo en la secuencia de accesos

El segundo grupo de peticiones accede a la misma variable y por tanto, al mismo módulo de memoria. El acceso al módulo de memoria es serializado por el árbitro de la RI. El segundo acceso, al acceder a la misma variable que el primero, es serializado por el CM. El procesado del segundo acceso queda bloqueado hasta que finaliza el procesado del primero.

El tercer grupo de peticiones accede a módulos de memoria distintos. La intersección de los conjunto de mensajes emitidos por el CM1 y el CM2 es distinto del conjunto vacío. En consecuencia hay que arbitrar en RV. En primer lugar se transmite el mensaje de CM1 y posteriormente los mensajes de CM2.

En el cuarto grupo de instrucciones, la primera de ellas es un acierto en cache. la segunda es un fallo.

El quinto grupo de peticiones accede al mismo módulo de memoria y en particular a la misma variable. Las dos transacciones son de escritura. El árbitro de la RI determina que en primer lugar se procese la petición del CC2. La primera transacción es de 2 pasos y segunda de 4 pasos.

Diagrama temporal

Después de utilizar la RI se representa el acceso a memoria, indicando como sufijo el ordinal del módulo de memoria accedido. Antes de utilizar una red para transmitir el mensaje o los mensajes se representa el arbitraje. Esta fase se utiliza para mostrar la ordenación de los mensajes. La espera para acceder a una red se indica representando en ciclos consecutivos la fase de arbitraje (arb). El resto se representa de la forma descrita previamente en este Capítulo.

Ejemplo. En la Figura 7.53 se muestra el diagrama temporal de la secuencia de accesos mostrada en la Tabla 7.8. Observemos el bloqueo del segundo acceso a memoria en el segundo grupo de accesos. En el tercer grupo de accesos hay que arbitrar en RV.

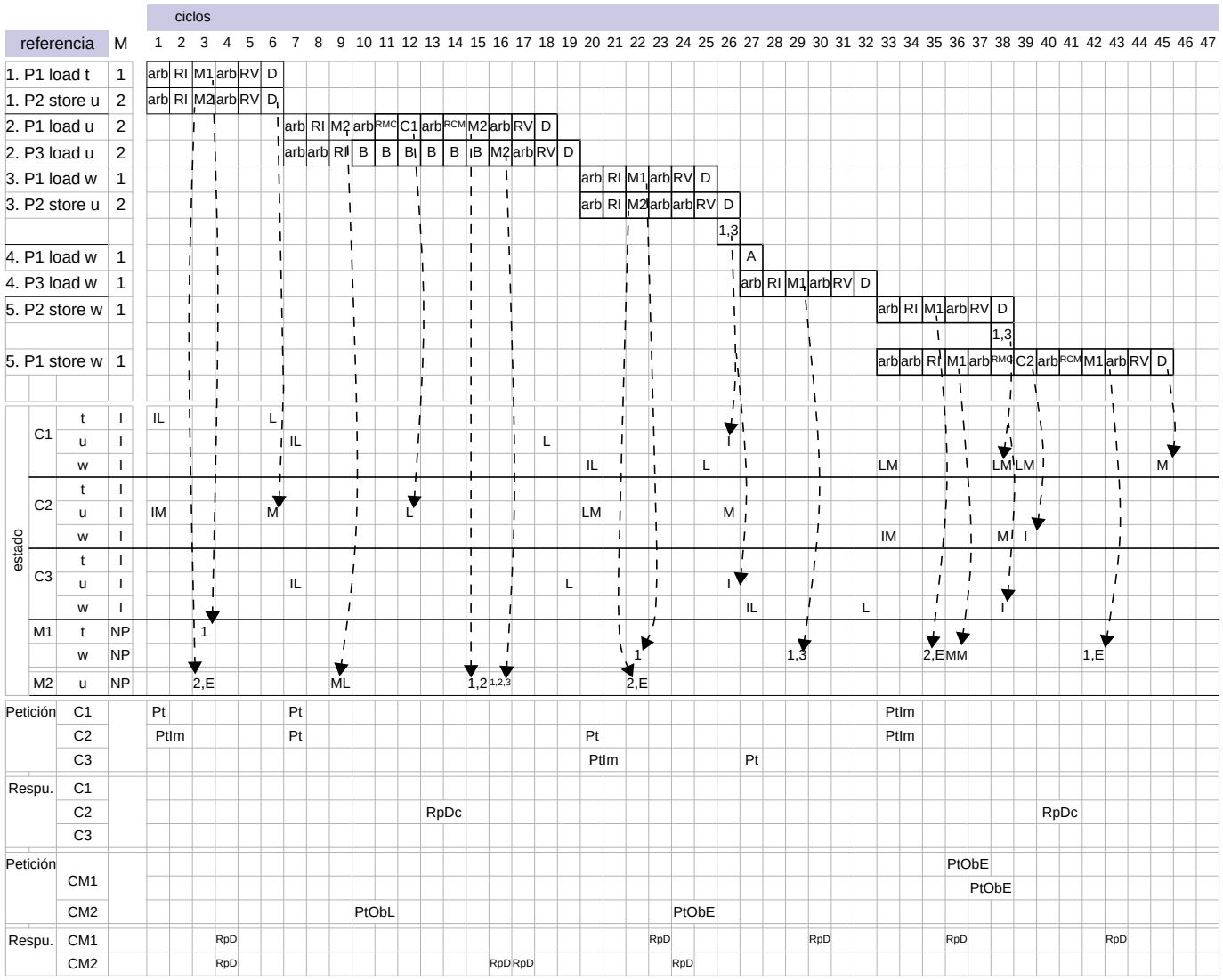


Figura 7.53 Protocolo de directorio MLI. Diagrama temporal de una secuencia de accesos paralelos a memoria.

Diagrama temporal simplificado

En la Figura 7.54 se muestra el diagrama temporal simplificado del flujo de mensajes, en los dos sentidos, entre los CC y los CM. En particular se muestra el arbitraje en las distintas redes y la espera en la CP de una petición para ser procesada por el CM (serialización de escrituras). La secuencia de accesos es la mostrada en la Tabla 7.8.

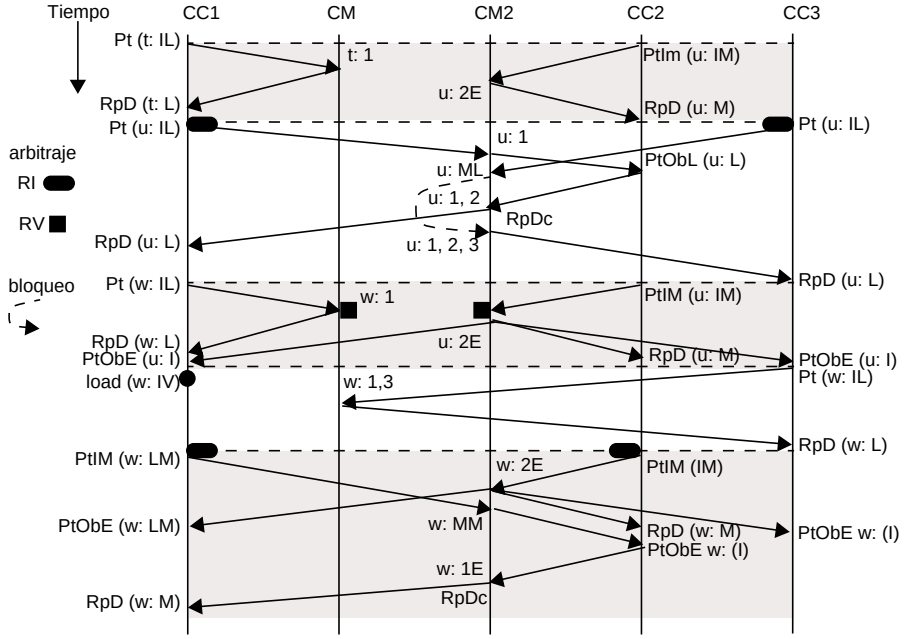


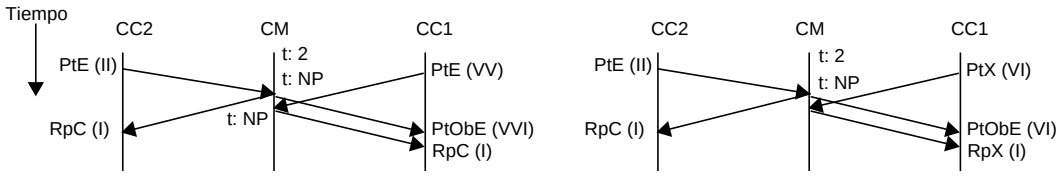
Figura 7.54 Protocolo de directorio MLI. Diagrama temporal simplificado en una secuencia de accesos paralela.

EJEMPLOS

Protocolo de directorio VI. Diagramas temporales simplificados

Pregunta 1: *Represente mediante un diagrama temporal simplificado los ejemplos mostrados en la Figura 7.18 y en la Figura 7.19 relativos a cruces de peticiones en un protocolo de directorio VI.*

Respuesta: En la parte izquierda de la figura se muestra el diagrama correspondiente a la Figura 7.18 y en la parte derecha el diagrama correspondiente a la Figura 7.19. El CC de la izquierda del diagrama es el que accede en primer lugar al CM.



Protocolo de directorio MLI. Cruces de peticiones

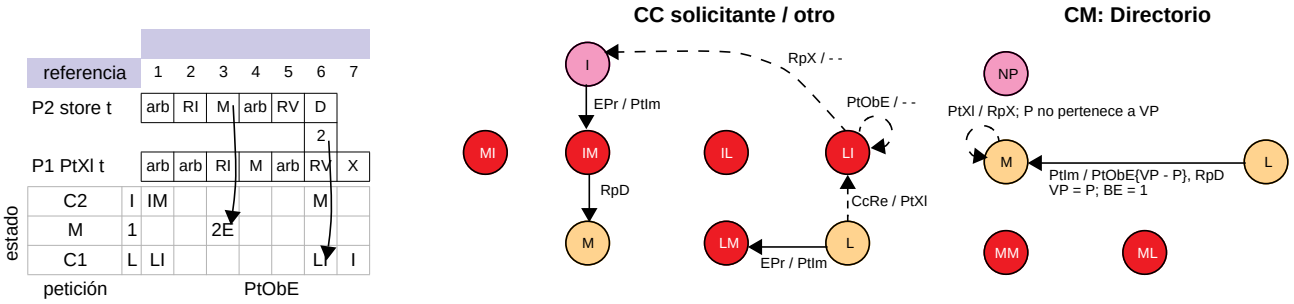
Pregunta 1: *Para las secuencias de accesos a memoria mostradas en la Figura 7.30 y en la Figura 7.32 muestre un diagrama temporal.*

Respuesta:

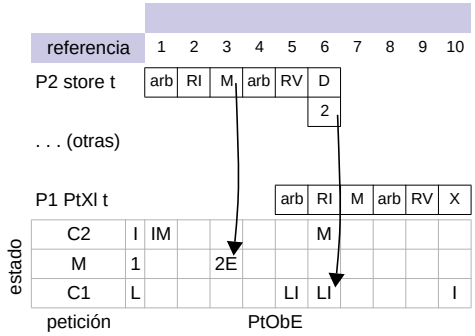
Orden A. En el estado LI un CC puede recibir una petición PtObE, lo cual indica que el CM ha procesado previamente una petición PtIm de otro CC. Cuando en el CM se procesa la petición PtXl el CC no está en el VP.

En la siguiente figura se muestra un diagrama temporal y las transiciones entre estados. En trazo continuo se muestran las transiciones debidas al CC de la cache C2. En trazo discontinuo las transiciones debidas al CC de la cache C1.

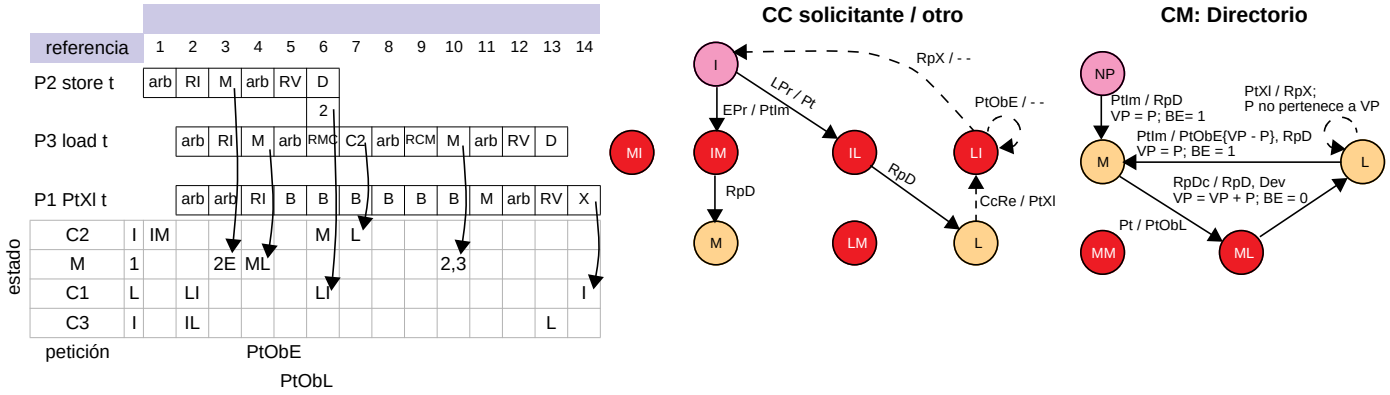
El cruce en el CM se observa en el estado M y el CC no cambia el estado y espera la respuesta.



En la cache C1, la transición del estado estable del bloque a un estado transitorio está determinado por la emisión de una petición de CC1. Como se muestra en la siguiente figura, es suficiente que la emisión se efectúe antes que la recepción de la petición del CM. Notemos que entre las dos transacciones, el CM puede procesar otras transacciones de otros CC que referencian el mismo o distinto bloque.



Orden B. Antes de procesar la petición de expulsión se puede procesar en el CM, además de la petición de exclusividad (PtIm), una o varias peticiones de lectura del bloque (Pt). En estas condiciones la petición de expulsión se procesa en el CM en el estado L y en el VP no está incluido el CC1 (siguiente figura). El cruce en el CM se observa en el estado L y el CC1, al recibir la petición del CM, se queda en el mismo estado esperando la respuesta.

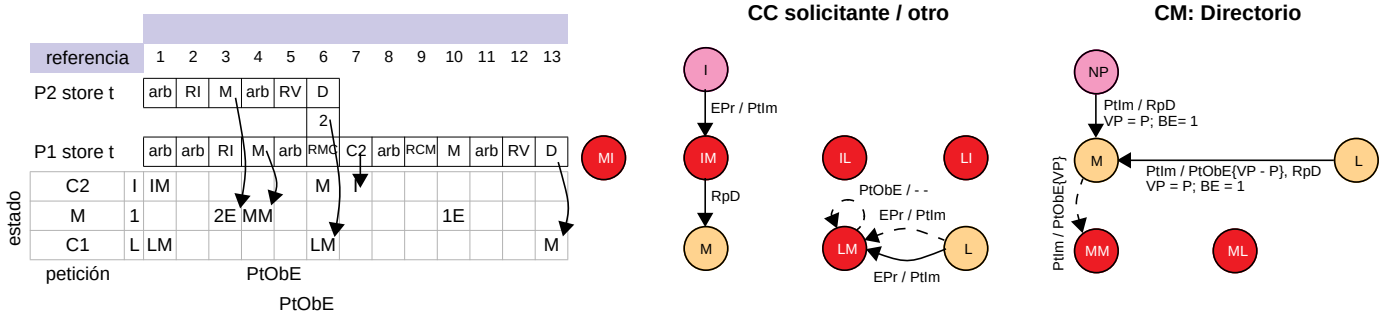


Si la petición de P3 en la figura previa es PtIm, debido a un store, el cruce en el CM se observa en el estado M, el cual es el orden A.

Orden C. En el estado LM un CC puede recibir una petición PtObE, lo cual indica que el CM ha procesado previamente una petición PtIm de otro CC.

En la siguiente figura se muestra un diagrama temporal y las transiciones entre estados. En trazo continuo se muestran las transiciones debidas al CC de C2. En trazo discontinuo las transiciones debidas al CC de C1.

En la cache C1, la transición del estado estable del bloque a un estado transitorio está determinado por la emisión de una petición de CC1. Es suficiente que la emisión se efectúe antes que la recepción de la petición del CM[38]. El CM no infiere ningún cruce, ya que la petición PtIm se puede recibir en el CM cuando el bloque en el CC está en los estados I o L.



38. Notemos que temporalmente el procesado de la petición de CC2 en el CM puede ser bastante después si hay entrelazadas peticiones a otros bloques.

La petición del CM es para invalidar el bloque. Sin embargo, el CC1 no cambia de estado, ya que su petición es PtIm y en la respuesta se incluye el bloque.

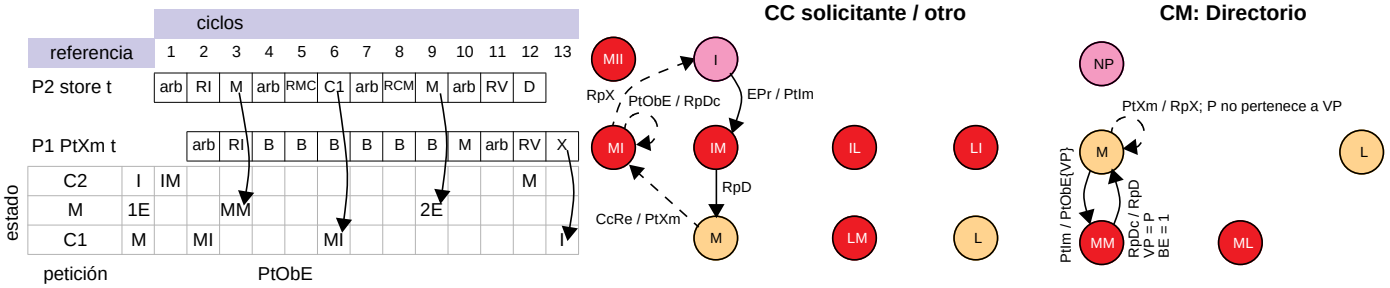
El funcionamiento de la RV garantiza que el CC1 recibe primero la petición y después la respuesta. De forma similar, el CC2 recibe primero la respuesta a su petición y después la petición del CM, inducida al procesar la petición del CC1.

Orden D. El comportamiento del CC1 en el orden D de accesos a memoria es el mismo que en el orden C. El CM al procesar la petición del CC3 no emite ningún mensaje al CC1. El bloque lo tiene en exclusividad el CC2.

Ordenes E, F y H. En el estado MI un CC puede recibir peticiones PtObE o PtObL y la respuesta es suministrar el bloque. Estos casos son los ordenes E y F de la Figura 7.32. También puede recibir una secuencia PtObL seguido de PtObE. En este caso, la respuesta a la petición PtObL es suministrar el bloque y la petición PtObE no tiene respuesta, ya que es una petición para invalidar el bloque. Este caso es el orden H de la Figura 7.32.

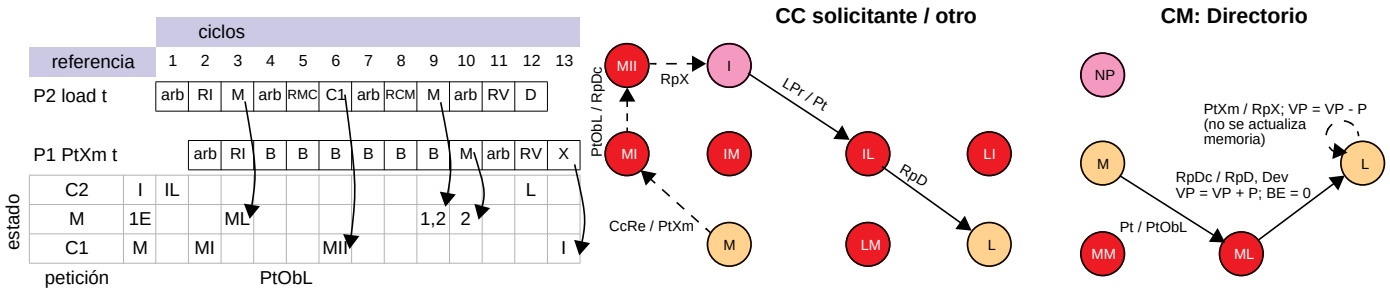
Para identificar la posible secuencia de peticiones desde el CM añadimos un estado transitorio denominado MII.

Orden E. El CM, antes de procesar una petición de expulsión PtXm, procesa una petición de exclusividad (PtIm). En estas condiciones la petición de expulsión se procesa en el estado M y en el VP no está incluido el CC que efectúa la petición de expulsión. El cruce en el CM se observa en el estado M. El CC1 suministra el bloque, no cambia el estado del bloque y espera la respuesta.

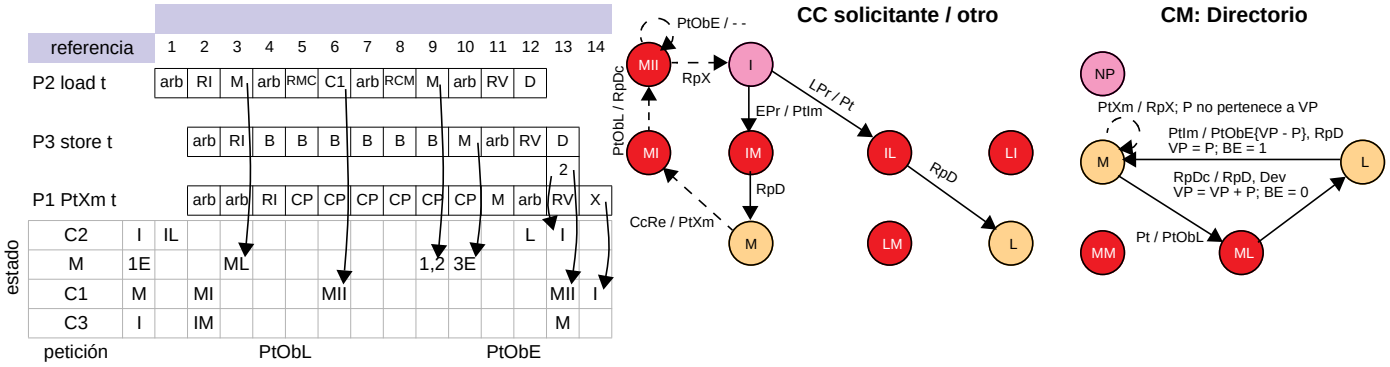


Orden F. Antes de procesar una petición de expulsión, el CM puede procesar una petición de bloque (Pt), la cual requiere que el CM emita una petición de observación de lectura (PtObL). En estas condiciones la petición de expulsión (PtXm) se procesa en el CM en el estado L y el CC1 está incluido en el VP. Sin embargo, no es una petición esperada en este estado. Por tanto, aunque sea una petición PtXm no se actualiza memoria[39]. El cruce en el CM se

observa en el estado L. El CC1, al recibir la petición PtObL, suministra el bloque, cambia el estado del bloque a MII y espera la respuesta a la petición de expulsión.



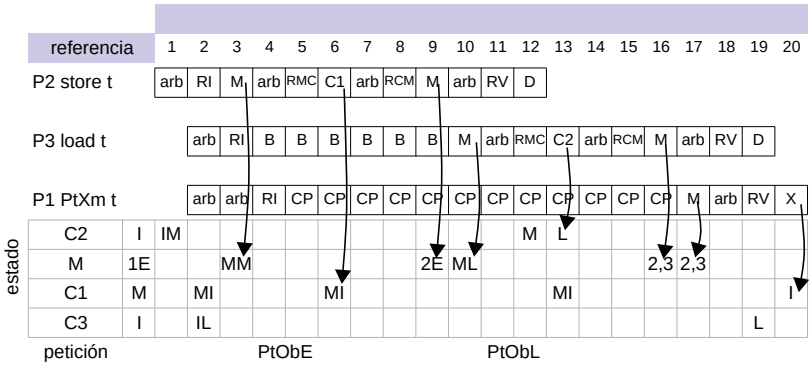
Orden H. Antes de procesar una petición de expulsión el CM puede procesar, además de la petición de bloque (Pt), la cual induce una petición de bloque (PtObL), una petición de exclusividad (PtIm), la cual induce una petición de observación de escritura (PtObE) desde el CM. En estas condiciones, la petición de expulsión se procesa en el CM en el estado M y el CC1 no está incluido en el VP. Esto es, el cruce en el CM se observa en el estado M. Cuando el CC1 recibe la petición PtObE el bloque está en estado MII y el CC1 no cambia el estado del bloque.



Orden G. El CM antes de procesar una petición de expulsión de un CC puede procesar una petición PtIm y después una petición Pt. La primera petición induce una petición PtObE desde el Cm al CC1. Al finalizar la transacción el CC1 no está en el VP. El CM al finalizar el procesado de la

39. Memoria ha sido actualizada al responder el CC a la petición PtObL.

petición Pt de CC3 establece como estado del bloque en el directorio el estado L. En consecuencia, cuando el CM procesa la petición de expulsión del CC1, el VP no contiene el identificador del CC1 y además, la petición no es esperada.



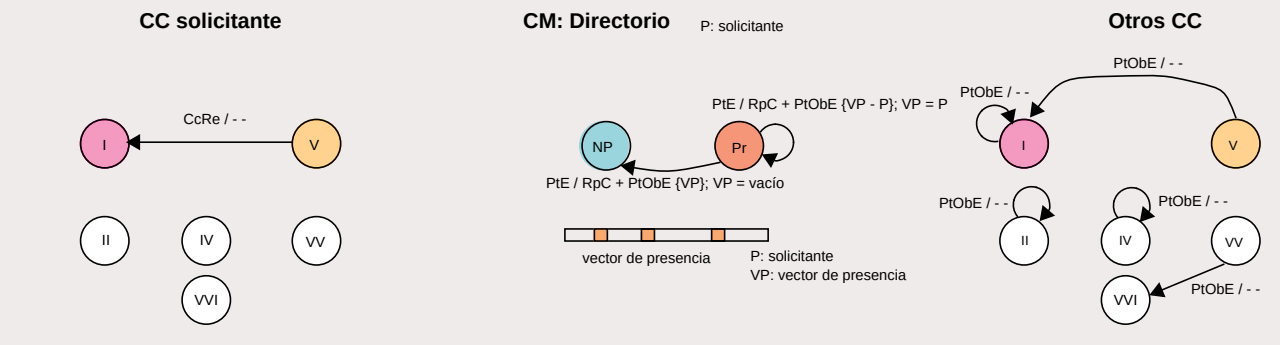
Protocolo de directorio VI. Expulsión silenciosa

En una expulsión silenciosa el directorio no recibe notificación de la expulsión de un bloque. Esta característica determina que un CC pueda recibir una petición de observación de escritura (PtObE) de un bloque que no tiene almacenado en cache. Por tanto, el agente observador, antes de invalidar el contenido de un contenedor, debe comprobar que el contenedor almacena el bloque al que hace referencia la petición de invalidación.

Pregunta 1: *Suponga que pueden existir accesos concurrentes a memoria. Para un protocolo de directorio VI, diseñe los autómatas de cambio de estado de un bloque en el CC y en el CM cuando se utiliza expulsión silenciosa.*

Respuesta: En la siguiente figura se muestra el cambio de estado de un bloque cuando una expulsión de un bloque de cache es silenciosa. Notemos que en el autómata de estados de un bloque en el CC no se utiliza el estado VI. La expulsión no se notifica al CM y en consecuencia el vector de presencia no se actualiza. Por tanto, no se detectan cruces en el CM. Recordemos que, en este protocolo, los cruces son debidos a expulsiones.

En la figura también se muestran las transiciones, de un bloque en el directorio, en las que se emite una petición PtObE.



Al ser la expulsión silenciosa, un CC puede recibir una petición de observación de escritura (PtObE) en cualquier estado, ya sea estable o transitorio. Entonces, hay que analizar estado por estado las consecuencias de una observación de escritura emitida por el CM.

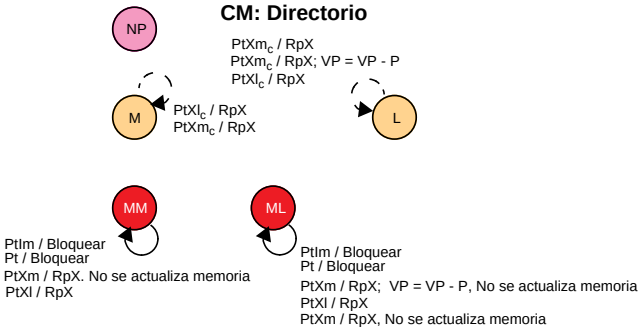
Las transiciones desde los estados estables V y VV han sido descritas al suponer el directorio preciso.

En el estado VVI un CC no puede recibir una petición PtObE, ya que el CM ha extraído al CC del VP al emitir la petición PtObE, que ha determinado la transición del estado VV al estado VVI.

Las transiciones en los estados I, II, IV al recibir una petición PtObE del CM son autotransiciones. La primera se puede producir después de expulsar el bloque. Las dos siguientes se pueden producir después de la expulsión del bloque y una posterior referencia al mismo bloque por parte del procesador. En el caso del estado II es una instrucción store y en el caso del estado IV es una instrucción load.

Protocolo MLI: diagramas temporales con cruces de peticiones

Utilice el protocolo de directorio MLI descrito en este capítulo. El multiprocesador dispone de un CM. Para los cruces de peticiones en el CM utilice el siguiente diagrama de transiciones entre estados. En los estados transitorios MM y ML se responde a las peticiones PtXm y PtXl.



En los siguientes diagramas temporales se muestra el estado estable en cache antes de emitirse la petición y el instante en que se obtiene el acceso a la RI. Cuando una petición está varios ciclos en la fase arb indica que hay otros CC que están encolando peticiones en la CP. Cuando se indica la fase CP, el CM está procesando peticiones de otros CC, a bloques distintos, que han sido emitidas previamente o concurrentemente.

Pregunta 1: *Complete el diagrama temporal en cada uno de los siguientes casos.*

1

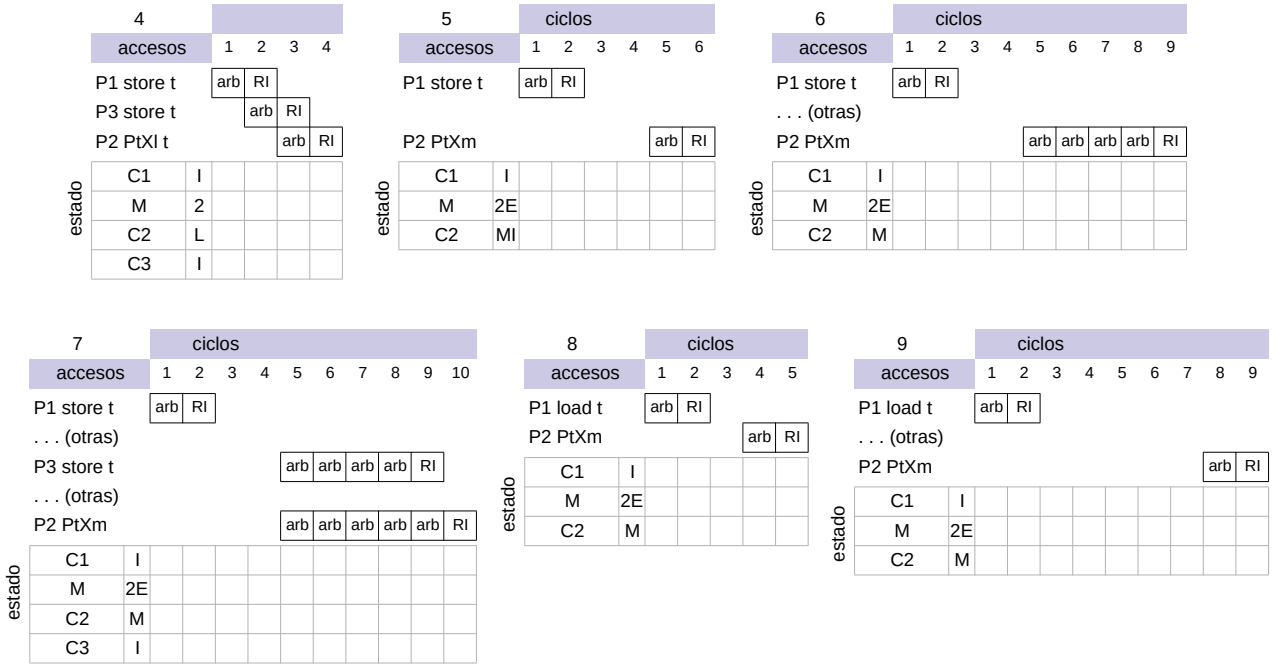
accesos		1	2	3	4	5	6
P1 store t		arb	RI				
P2 PtXl t						arb	RI
estado C1	I						
estado M	2						
estado C2	L						

2

accesos		1	2	3	4	5	6
P1 store t		arb	RI				
P3 load t					arb	RI	
P2 PtXl t						arb	RI
estado C1	I						
estado M	2						
estado C2	L						
estado C3	I						

3

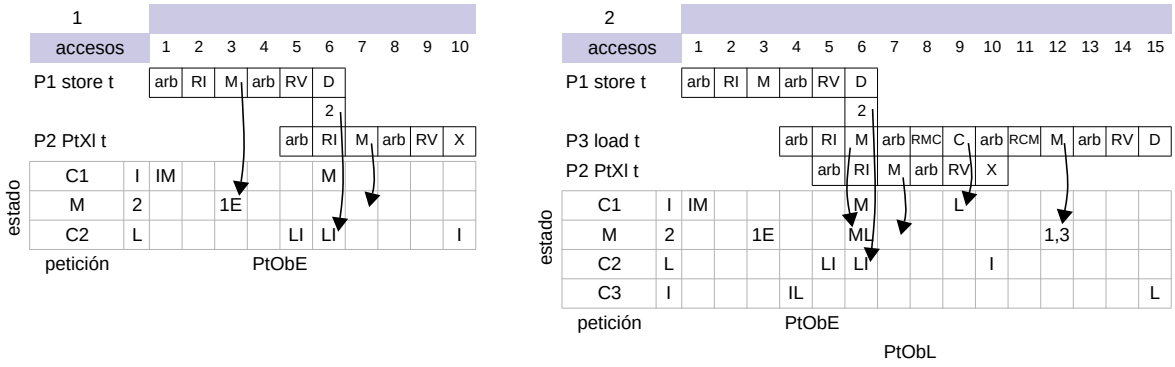
accesos		1	2	3	4	5	6	7	8	9	10	11	12
P1 store t		arb	RI										
P3 load t					arb	RI							
. . . (otras)													
P2 PtXl t								arb	arb	arb	RI	CP	CP	CP	CP
estado C1	I														
estado M	2														
estado C2	L														
estado C3	L														



Respuesta:

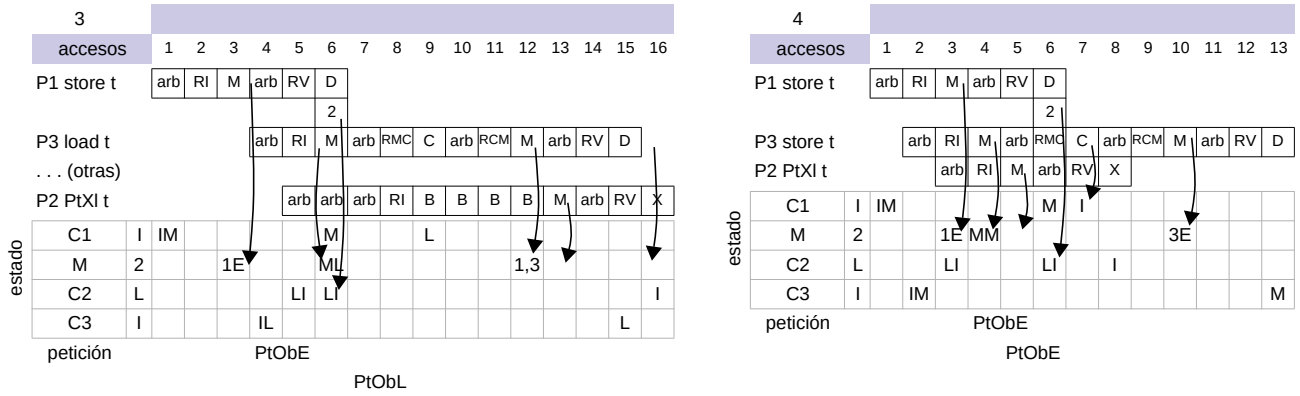
1º diagrama temporal. La petición PtXl de CC2 se procesa en el CM en el estado M. Es el instante más tardío, respecto a la petición de CC1, en la cual CC2 puede emitir la petición PtXl para que se produzca un cruce.

2º diagrama temporal. La petición de PtXl de CC2 se procesa en el CM en el estado ML.



3º diagrama temporal. La petición PtXl de CC2 se procesa en el CM en el estado L.

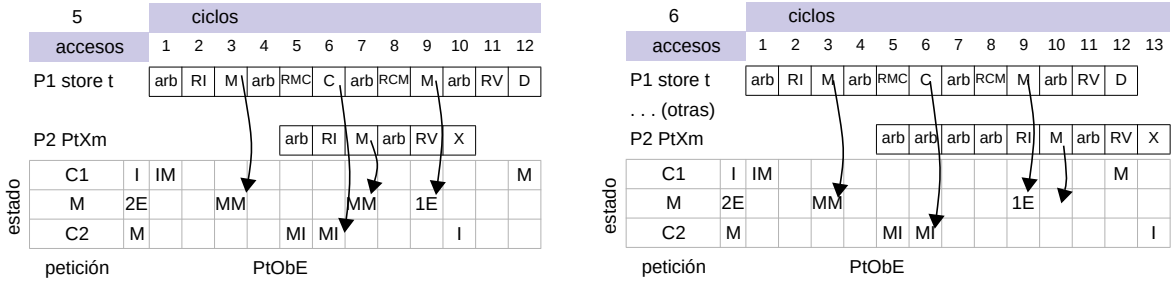
4º diagrama temporal. El procesado de la petición en el CM es en el estado MM.



En los cuatro diagrama temporales previos, cuando se infiere el cruce, el estado del bloque en el CC2 es LI.

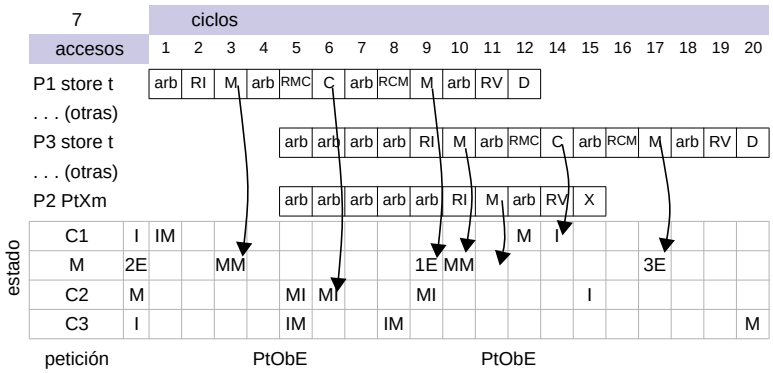
5º diagrama temporal. La petición PtXlm de CC2 se procesa en el CM en el estado MM.

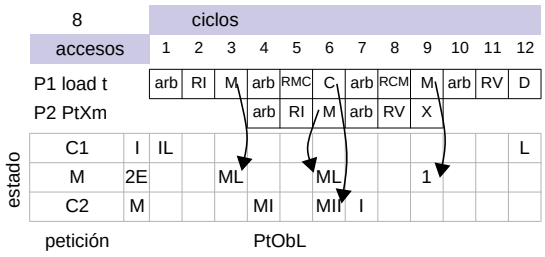
6º diagrama temporal. El procesado de la petición en el CM es en el estado M. En el CC2 el cruce se infiere en el estado MI.



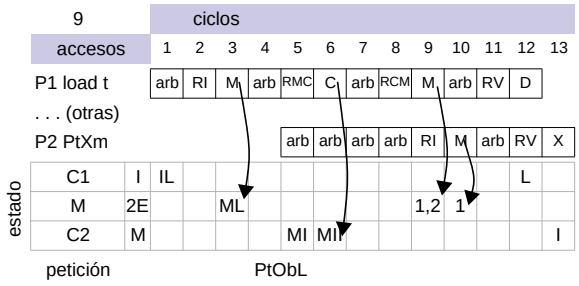
7º diagrama temporal. La petición PtXlm de CC2 se procesa en el CM en el estado MM.

8º diagrama temporal. El procesado de la petición en el CM es en el estado ML. En el CC2 el cruce se infiere en el estado MI.





9º diagrama temporal. La petición PtXlm de CC2 se procesa en el CM en el estado L. En el CC2 el cruce se infiere en el estado MI.

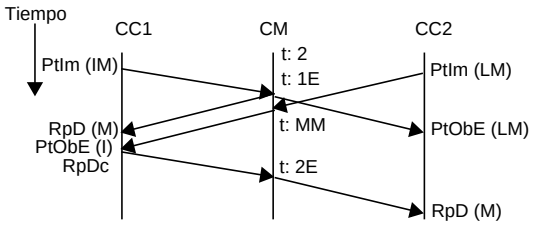


Protocolo de directorio MLI. Diagramas temporales simplificados

Pregunta 1: *Represente mediante un diagrama temporal simplificado la siguiente secuencia de accesos, cuyo orden de procesado en el CM se muestra.*

Orden de procesado en el CM
P2 store t
P1 store t

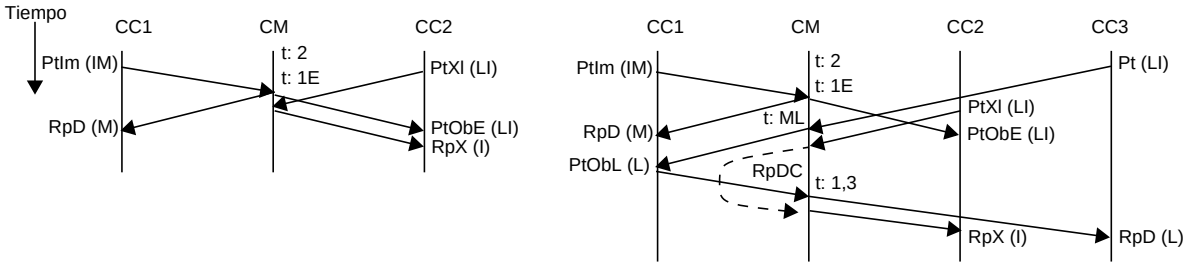
Respuesta: En la figura se muestra el diagrama correspondiente.



Pregunta 2: *Represente mediante un diagrama temporal simplificado las siguientes secuencias de accesos, cuyo orden de procesado en el CM se muestra.*

Orden de procesado en el CM	
A	**B**
P2 store t	P2 store t
P1 PtXl t	P3 load t
	P1 PtXl t

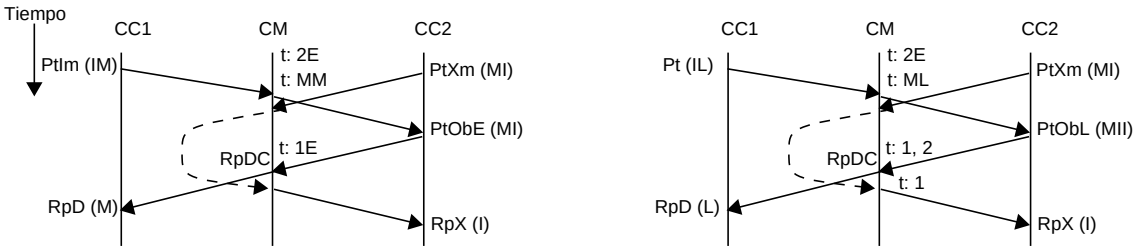
Respuesta: En la parte izquierda de la figura se muestra el diagrama correspondiente a la secuencia A. El diagrama de la parte derecha corresponde a la secuencia B.



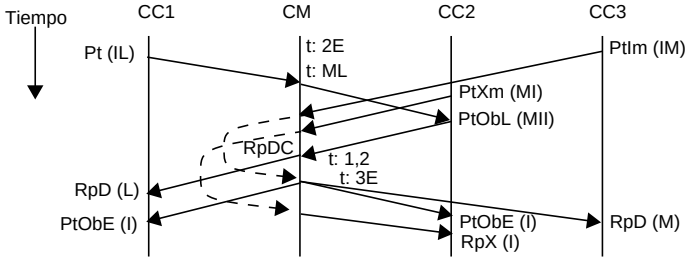
Pregunta 3: *Represente mediante un diagrama temporal simplificado las siguientes secuencias de accesos, cuyo orden de procesado en el CM se muestra.*

Orden de procesado en el CM		
C	**D**	**E**
P1 store t	P1 load t	P3 load t
P2 PtXm t	P2 PtXm t	P1 store t
		P2 PtXm t

Respuesta: En la parte izquierda de la figura se muestra el diagrama correspondiente a la secuencia C. El diagrama de la parte derecha corresponde a la secuencia D.



En la figura se muestra el diagrama correspondiente a la secuencia E.



EJERCICIOS

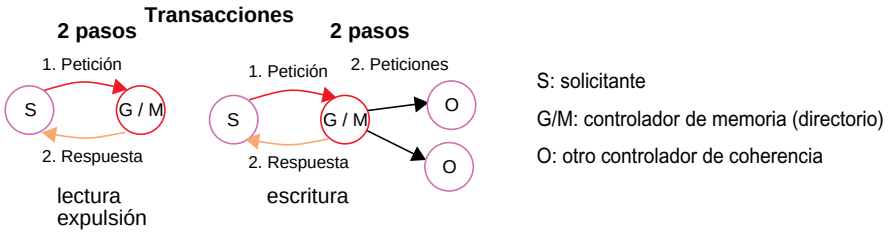
Descripción de un protocolo de directorio VI denominado A

Suponga un multiprocesador donde las caches privadas son de mapeo directo y utilizan escritura inmediata. Las redes de interconexión entre las caches y el módulo de memoria son de tipo crossbar y mantienen el orden de los mensajes emitidos. El multiprocesador utiliza un directorio para mantener la coherencia y el protocolo de coherencia es de invalidación (VI).

Las caches privadas de los procesadores son bloqueantes. En un fallo de cache o en una escritura se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

El directorio utiliza un vector de presencia (VP) por bloque. El vector de presencia es un vector de bits, con tantos bits como procesadores y cada bit está asociado a un procesador.

Las secuencias de mensajes de las transacciones son las siguientes:



Las peticiones de procesador y los mensajes utilizados en la transacciones para mantener la coherencia son:

Procesador	Controlador de cache (CC)		Controlador de memoria (CM)	
Peticiones	**Peticiones del CC al CM**	**Respuestas del CM al CC**	**Peticiones del CM a los CC**	**Acciones**
LPr : lectura	Pt : petición de bloque	RpD: respuesta con el bloque	PtObE: petición de observación de escritura	Actualización del directorio
EPr : escritura	PtE: petición de escritura de un dato	RpC: respuesta de confirmación		Dev: actualización de memoria
	PtX: petición de expulsión	RpX: respuesta de confirmación a una petición PtX		

El controlador de cache también efectúa acciones de reemplazo cuando es necesario (CcRe). En una acción de reemplazo se distingue la acción de notificación al directorio, ya que éste es preciso. En una petición PtX se actualiza el directorio.

Cuando el servicio de un acceso a memoria requiere un reemplazo, éste se efectúa antes de gestionar el acceso a memoria que produce la acción de reemplazo.

Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtE, PtX	arb	RI	M
RpD, RpC, RpX	arb	RV	D ó X
PtObE	arb	RMC	Cx

arb: arbitraje en la red correspondiente

RI: red de peticiones desde los CC al CM

RV: red de respuestas desde el CM a los CC

RMC: red de peticiones desde el CM a los CC

M: memoria (directorio)

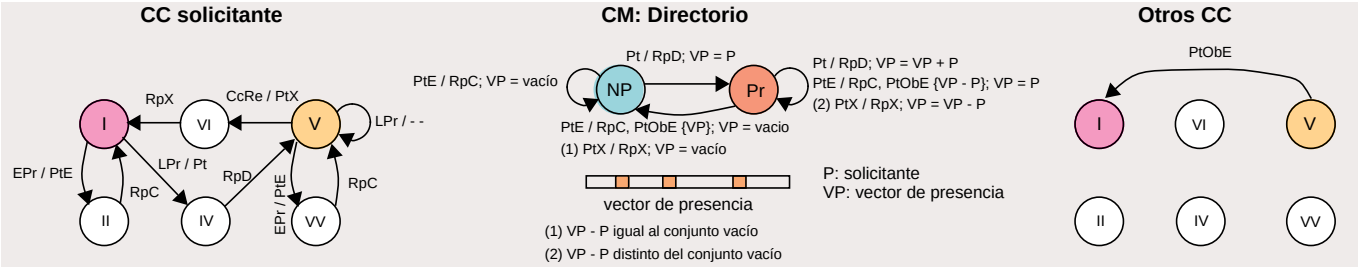
D: dato (RpD)

X: confirmación (RpX)

Cx: cache, donde x es el ordinal de la cache que recibe PtObE

En un CC, para distinguir, en una transacción, entre la emisión de un mensaje de petición y la recepción de una respuesta, se utilizan estados transitorios (II, IV, VV, VI). En el CM no es necesario ya que no espera respuestas.

En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio cuando no se consideran posibles cruces de peticiones.



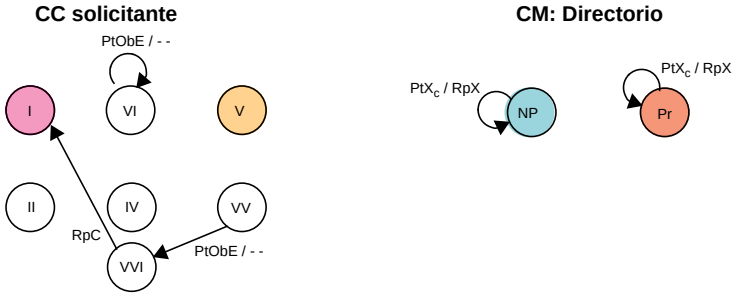
En este multiprocesador supondremos que sólo existe un acceso a memoria en un instante determinado.

En un diagrama temporal se muestran en la parte superior las fases de los mensajes de una transacción, en la parte central se especifica el estado de los bloques en las cache y en el directorio. En la parte inferior se etiqueta el mensaje o la respuesta que se representa en la parte superior. El estado de los bloques en cache o en el directorio se indica sólo cuando hay un cambio de estado.

Fases y eventos	Especificaciones
arb	Se especifica el estado transitorio del bloque.
M	Se especifica el vector de presencia utilizando el ordinal de los procesadores cuyas caches tienen copia del bloque.
D ó X	Se especifica el estado estable del bloque en la cache cuyo CC ha efectuado la petición.
x	Se especifica la modificación de estado determinada por la petición del CM
Transacción de 2 pasos con petiiciones del CM	Se utiliza una fila para indicar la respuesta del CM y otra fila para indicar todas las peticiones del CM.
Reemplazo	Determina una expulsión: se especifica en la columna etiquetada como referencia. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtX) y en la segunda fila la petición que determina la expulsión.
Mensaje	Se indica en la columna correspondiente a arb.

Al tener en cuenta la concurrencia de peticiones de los CC hay que considerar los posibles cruces de peticiones en el CM y en los CC. En el CM se procesan peticiones a bloques en estados estables.

En la siguiente figura se muestra la gestión de peticiones en los cruces. En el diagrama de estados de un bloque en el directorio, se utiliza el subíndice c para identificar una petición del CC que se ha cruzado con una petición del CM y la petición del CC se analiza en un estado estable.



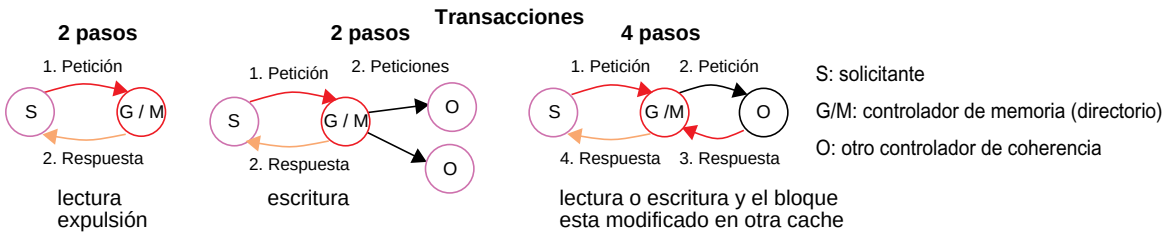
Descripción de un protocolo de directorio MLI denominado A

Suponga un multiprocesador donde las caches privadas son de mapeo directo y utilizan escritura retardada. Las redes de interconexión entre las caches y el módulo de memoria son de tipo crossbar y mantienen el orden de los mensajes emitidos. El multiprocesador utiliza un directorio para mantener la coherencia y el protocolo de coherencia es de invalidación (MLI).

Las caches privadas de los procesadores son bloqueantes. En un fallo de cache o en una solicitud de exclusividad se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

El directorio utiliza un vector de presencia y un bit de exclusividad por bloque. El vector de presencia (VP) es un vector de bits, con tantos bits como procesadores y cada bit está asociado a un procesador. El bit de exclusividad (BE) se utiliza para indicar que sólo existe una copia del bloque en una cache privada, la cual está identificada en el vector de presencia.

Las secuencias de mensajes de las transacciones son las siguientes:



Las peticiones de procesador y los mensajes utilizados en la transacciones para mantener la coherencia son:

Procesador	Controlador de cache (CC)		Controlador de memoria (CM)		
Peticiones	**Peticiones del CC al CM**	**Respuestas del CM al CC**	**Peticiones del CM a los CC**	**Respuestas del CC al CM**	**Acciones**
LPr : lectura	Pt : petición de bloque	RpD: respuesta con el bloque a una petición Pt o PtIm	PtObE: petición de observación de escritura, inducida por una petición PtIm	RpDc: respuesta con el boque a una petición PtObL o PtObE y el estado del bloque en cache es M	Actualización del directorio
EPr : escritura	PtIm: petición de bloque con intención de modificarlo	RpX: respuesta de confirmación a una petición PtXm o PtXl	PtObL: petición de observación de lectura, inducida por una petición Pt y el estado del bloque en el directorio es M		Dev: actualización de memoria
	PtXm: petición de expulsión de un bloque en estado M				
	PtXl: petición (notificación) de expulsión de un bloque en estado L				

El controlador de cache también efectúa acciones de reemplazo cuando es necesario (CcRe). En una acción de reemplazo se distingue la acción de notificación al directorio, ya que éste es preciso y si es el caso, una actualización de memoria con el bloque expulsado, si éste ha sido modificado durante su estancia en la cache. En una petición PtXm se actualiza el directorio y memoria, mientras que en una petición PtXl sólo se actualiza el directorio.

Cuando el servicio de un acceso a memoria requiere un reemplazo, éste se efectúa antes de gestionar el acceso a memoria que produce la acción de reemplazo.

Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtIm, PtXm, PXl	arb	RI	M
RpD, RpX	arb	RV	D ó X
PtObE, PtObL	arb	RMC	Cx
RpDc	arb	RCM	M

arb: arbitraje en la red correspondiente

RI: red de peticiones desde los CC al CM

RV: red de respuestas desde el CM a los CC

RMC: red de peticiones desde el CM a los CC

RCM: red de respuestas de los CC al CM

M: memoria (directorio)

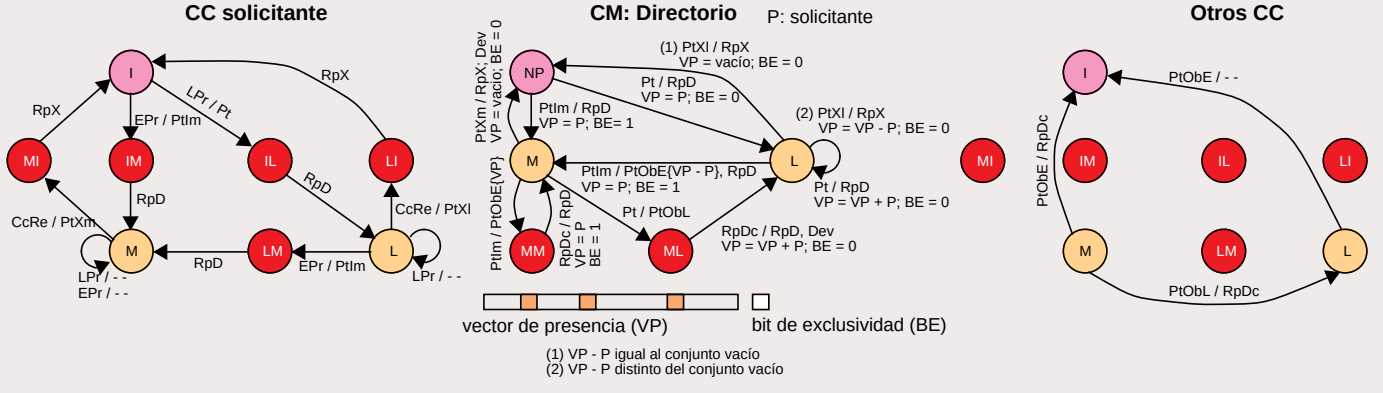
D: dato (RpD)

X: confirmación (RpX)

Cx: cache, donde x es el ordinal de la cache

En un CC, para distinguir, en una transacción, entre la emisión de un mensaje de petición y la recepción de una respuesta, se utilizan estados transitorios (IL, LM, IM, LI, MI). En el CM para distinguir entre la emisión de un mensaje de petición, correspondiente a una transacción que está procesando el CM, y la respuesta de un CC se utilizan estados transitorios (ML, MM).

En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio, cuando no se consideran posibles cruces de peticiones. En el protocolo que se describe, el bit de exclusividad del directorio se activa cuando una cache solicita el bloque para actualizarlo.

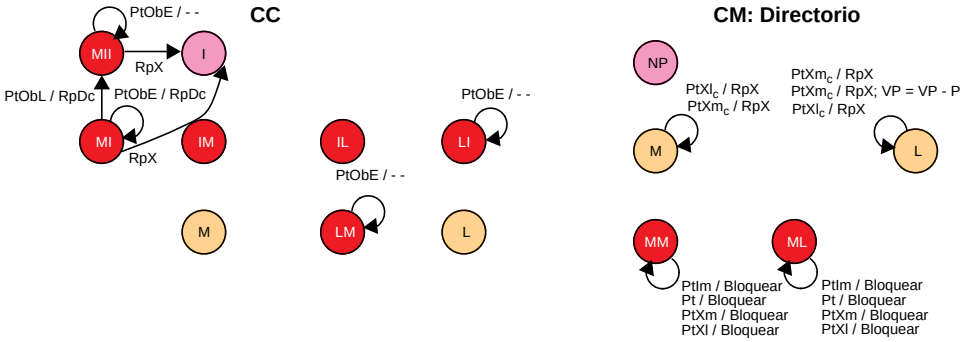


En un diagrama temporal se muestran en la parte superior las fases de los mensajes de una transacción, en la parte central se especifica el estado de los bloques en las cache y en el directorio. En la parte inferior se etiqueta el mensaje o la respuesta que se representa en la parte superior. El estado de los bloques en cache o en el directorio se indica sólo cuando hay un cambio de estado.

Fases y eventos	Especificaciones
arb	Se especifica el estado transitorio del bloque.
M	Se especifica el VP utilizando el ordinal de los procesadores cuyas caches tienen copia del bloque y si el bloque lo tiene una cache en exclusividad se añade la letra E. La especificación se efectúa la última vez que se visita el directorio en una transacción.
D ó X	Se especifica el estado estable del bloque en la cache cuyo CC ha efectuado la petición.
C	Se especifica la modificación de estado determinada por la petición del CM
Transacción de 2 pasos	Se utiliza una fila para indicar la respuesta del CM y otra fila para indicar todas las peticiones del CM.
Reemplazo	Determina una expulsión: se especifica en la columna etiquetada como referencia. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtXm o PtXl) y en la segunda fila la petición que determina la expulsión.
Mensaje	Se indica en la columna correspondiente a arb.

Al tener en cuenta la concurrencia de peticiones de los CC hay que considerar los posibles cruces de peticiones en el CM y en los CC. En el CM sólo se procesan peticiones a bloques en estados estables. Una petición en la cabeza de la cola de peticiones (CP) que accede a un bloque en un estado transitorio determina un bloqueo del análisis de esta petición y las que le siguen en al CP. El bloqueo en la CP se indica en un diagrama temporal mediante el acrónimo B. La espera en la CP se indica mediante el acrónimo CP.

En la siguiente figura se muestra la gestión de peticiones en los cruces. En el diagrama de estados de un bloque en el directorio, se utiliza el subíndice c para identificar una petición del CC que se ha cruzado con una petición del CM y la petición del CC se analiza en un estado estable. Notemos que en un cruce no se actualiza la memoria al procesar la petición PtXm.



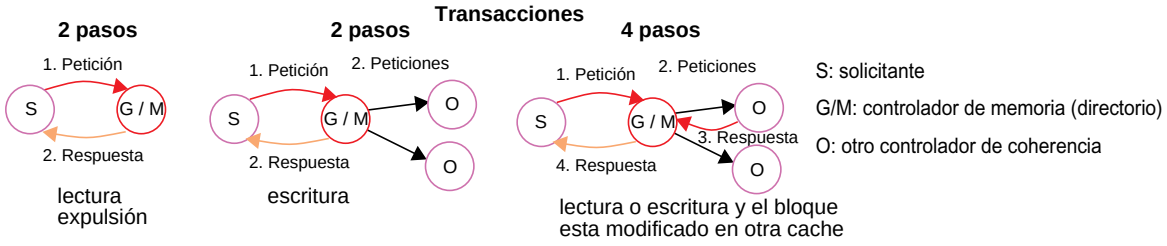
Descripción de un protocolo de directorio MLI denominado B

En un multiprocesador las caches privadas son de mapeo directo y utilizan escritura retardada. Las redes de interconexión entre las caches y los módulos de memoria (CM) son de tipo crossbar. Todos los mensaje generados por un CM, en un paso intermedio del procesado de una transacción o para finalizar una transacción, se emiten en el mismo ciclo a todos los CC involucrados en la acción de coherencia. El multiprocesador utiliza un directorio para mantener la coherencia y el protocolo de coherencia es de invalidación (MLI).

Las caches privadas de los procesadores son bloqueantes. En un fallo de cache o en una solicitud de exclusividad se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

El directorio utiliza 2 bits por bloque. Con estos bits se codifican 4 estados (NP, L1, L y M). El estado NP indica que no hay copias del bloque en las caches. El estado L1 indica que hay copia del bloque en sólo una cache y se ha solicitado para leer. El estado L indica que hay copia del bloque en varias caches y se ha solicitado para leer. El estado M indica que una cache tiene copia del bloque y el bloque ha sido solicitado para actualizarlo. Notemos que desde el estado L no se puede pasar al estado L1 ya que no se identifican explícitamente las caches que tienen copia y tampoco se dispone de un contador para conocer el número de copias. Como no se identifican las caches que tienen copia, las acciones del protocolo deben ser conservadoras.

Las secuencias de mensajes de las transacciones son las siguientes:



Las peticiones de procesador y los mensajes utilizados en las transacciones para mantener la coherencia son:

Procesador	Controlador de cache (CC)		Controlador de memoria (CM)		
Peticiones	**Peticiones del CC al CM**	**Respuestas del CM al CC**	**Peticiones del CM a los CC**	**Respuestas del CC al CM**	**Acciones**
LPr : lectura	Pt : petición de bloque	RpD: respuesta con el bloque a una petición Pt, PtIm o PtI	PtObE: petición de observación de escritura, inducida por una petición PtIm o PtI	RpDc: respuesta con el boque a una petición PtObL o PtObE y el estado del bloque en cache es M	Actualización del directorio
EPr : escritura	PtIm: petición de bloque con intención de modificarlo	RpX: respuesta de confirmación a una petición PtXm o PtXI	PtObL: petición de observación de lectura, inducida por una petición Pt y el estado del bloque en el directorio es M		Dev: actualización de memoria
	PtI: petición de exclusividad (bloque en estado L)				
	PtXm: petición de expulsión de un bloque en estado M				
	PtXI: petición (notificación) de expulsión de un bloque en estado L				

Un CC también efectúa acciones de reemplazo cuando es necesario (CcRe). En una acción de reemplazo se distingue la acción de notificación al directorio (bloque en estado L) y esta acción y la acción, adicional, de actualización de memoria (bloque en estado M). En una petición PtXm se actualiza el directorio y memoria, mientras que en una petición PtXI sólo se actualiza el directorio.

Cuando el servicio de un acceso a memoria requiere un reemplazo, éste se efectúa antes de gestionar el acceso a memoria que produce la acción de reemplazo.

Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtIm, PtI, PtXm, PXI	arb	RI	M
RpD, RpX	arb	RV	D ó X
PtObE, PtObL	arb	RMC	T
RpDc	arb	RCM	M

arb: arbitraje en la red correspondiente

RI: red de peticiones desde los CC a los CM

RV: red de respuestas desde los CM a los CC

RMC: red de peticiones desde los CM a los CC

RCM: red de respuestas de los CC a los CM

M: memoria (directorio)

D: dato (RpD)

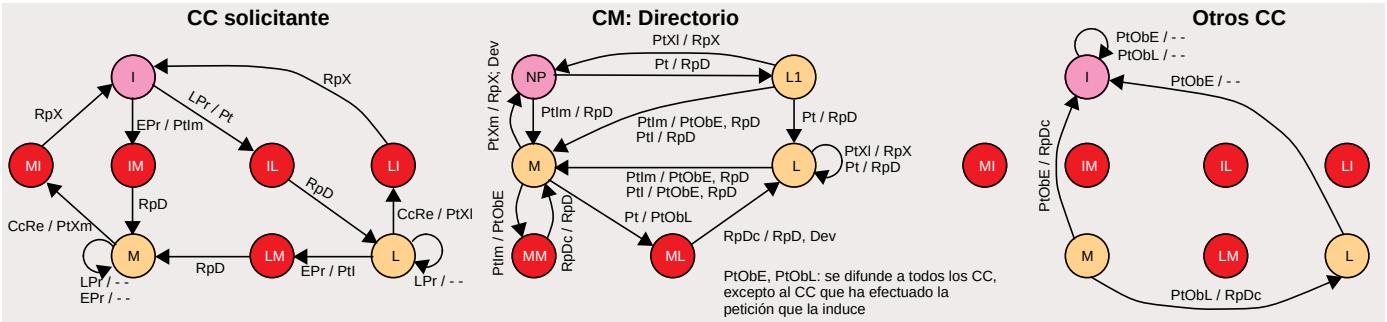
X: confirmación (RpX)

T: difusión de una petición del CM a todas las caches, excepto, si es el caso, a la cache que ha efectuado la petición

Las redes RMC (peticiones) y RV (respuestas) son la misma red física. Se utilizan acrónimos distintos para identificar el tipo de mensaje. Los enlaces puntos a punto mantienen el orden de los mensajes transmitidos. La gestión de las colas de mensajes de los CM y CC es FIFO.

En un CC se utilizan estados transitorios (IL, LM, IM, LI, MI) entre la emisión de un mensaje de petición y la recepción de una respuesta. En el CM para distinguir entre la emisión de un mensaje de petición, correspondiente a una transacción que está procesando el CM, y la respuesta de un CC se utilizan estados transitorios (ML, MM).

En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio cuando no se consideran posibles cruces de peticiones. Recordemos que no se identifican explícitamente las caches que tiene copia. Una petición PtObE o PtObL de un CM se difunde a todos los CC, excepto al CC que ha efectuado la petición, la cual ha inducido al CM a emitirla.



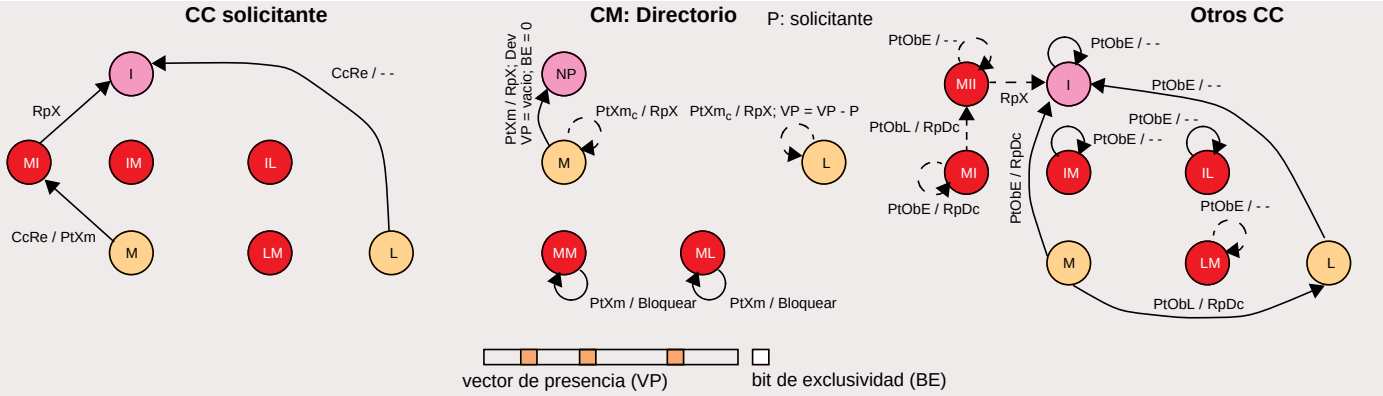
En un diagrama temporal se muestran en la parte superior las fases de los mensajes de una transacción, en la parte central se especifica el estado de los bloques en las cache y en el directorio. En la parte inferior se etiqueta el mensaje o la respuesta que se representa en la parte superior. El estado de los bloques en cache o en el directorio se indica sólo cuando hay un cambio de estado.

Fases y eventos	Especificaciones
arb	Se especifica el estado transitorio del bloque en la cache.
M	Se especifica el estado, ya sea transitorio o estable, del bloque en el directorio. Notemos que no se pueden indicar los ordinales de las caches que tienen copia, ya que no se conocen.
D ó X	Se especifica el estado estable del bloque en la cache cuyo CC ha efectuado la petición.
T	Se especifica la modificación de estado determinada por la petición del CM
Transacción de 2 pasos	Se utiliza una fila para indicar la respuesta del CM y otra fila para indicar la difusión de las peticiones del CM, si es el caso. La difusión de una petición PtObE o PtObL se indica mediante el acrónimo T (indica todas las caches excluida la que efectúa la petición que la induce).
Reemplazo	Determina una expulsión: se especifica en la columna etiquetada como referencia. Para ello, se utilizan dos filas contiguas. En la primera fila se especifica la expulsión (PtXm o PtXl) y en la segunda fila la petición que determina la expulsión.
Mensaje	Se indica en la columna correspondiente a la fase arb.

Ejercicio 7.1

Consideramos el protocolo de directorio MLI denominado A. Este protocolo se ha modificado para disponer de expulsión silenciosa. La expulsión de un bloque en el estado L no se notifica al CM, sólo existe un cambio de estado en el CC en el cual se expulsa el bloque. Por tanto, el VP no se actualiza. Esta característica determina que un CC pueda recibir una petición de observación de escritura (PtObE) de un bloque que no tiene almacenado en cache.

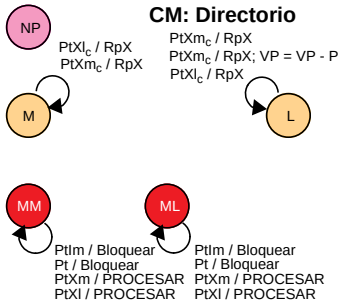
En la siguiente figura se muestran las transiciones en los diagramas de estados relacionadas con la expulsión de un bloque de cache, cuando el directorio no es preciso.



Pregunta 1: *Indique secuencias de peticiones concurrentes donde se observen las transiciones añadidas, respecto del caso de un directorio preciso, en el diagrama de estados de un CC. Muestre las transiciones entre estados utilizando diagramas temporales.*

Ejercicio 7.2

Consideramos el protocolo de directorio MLI denominado A. El objetivo es reducir el tiempo de ocupación del CM para incrementar el número de transacciones procesadas por unidad de tiempo. Para ello se analiza si el CM puede responder a peticiones de expulsión (PtXm, PtXl) en estados transitorios (MM, ML). En estas condiciones, el CM sólo se bloquearía al analizar peticiones Pt y PtIm al mismo bloque en un estado transitorio.



Para que el CM esté en un estado transitorio (MM, ML) es necesario que al procesar una petición haya solicitado el bloque a una cache que lo tiene en exclusividad.

En el caso de una petición PtXm, la cache que tiene actualmente el bloque en exclusividad puede ser la que emite la petición de expulsión del bloque o ser una tercera cache. Las siguientes secuencia de accesos muestran los distintos casos. Las dos primeras secuencia de accesos, empezando por la izquierda, corresponden al caso en el cual la cache, que tiene el bloque en exclusividad, es la que expulsa el bloque. En las dos siguientes secuencias de accesos, la cache que tiene actualmente el bloque en exclusividad es distinta de la cache que emite la petición de expulsión del bloque.

accesos A	accesos B	accesos C	accesos D
P1 store t	P1 load t	P1 store t	P1 store t
P2 PtXm t	P2 PtXm t	P3 store t	P3 load t
		P2 PtXm t	P2 PtXm t

estado							
C1	I	C1	I	C1	I	C1	I
M	2E	M	2E	M	2E	M	2E
C2	M	C2	M	C2	M	C2	M
				C3	I	C3	I

En el caso de una petición PtXl debe haber involucrada una tercera cache, la cual ha solicitado el bloque en exclusividad. En las siguientes secuencias de accesos hay una petición de una tercera cache que solicita el bloque en exclusividad.

accesos E	accesos F
P1 store t	P1 store t
P3 store t	P3 load t
P2 PtXl t	P2 PtXl t

estado			
C1	I	C1	I
M	2	M	2
C2	L	C2	L
C3	I	C3	I

El contenido del VP se actualiza al pasar de un estado transitorio al estado estable.

Pregunta 1: *Analice si es factible que el CM responda en el estado ML cuando procesa una petición PtXl o una petición PtXm. Para ello, seleccione, de las secuencias de accesos mostradas previamente, las secuencias que sean de utilidad. Muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Determine la información en el VP al procesarse la expulsión e indique si hay que actualizar la memoria, si es el caso. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.*

Pregunta 2: *Analice si es factible que el CM responda en el estado MM cuando procesa una petición PtXl o una petición PtXm. Para ello, seleccione, de las secuencias de accesos mostradas previamente, las secuencias que sean de utilidad. Muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Determine la información en el VP al procesarse la expulsión e indique si hay que actualizar la memoria, si es el caso. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.*

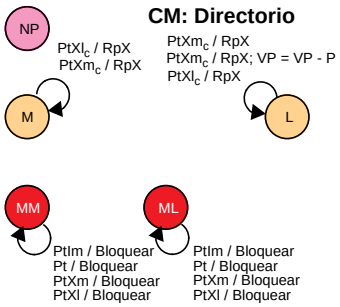
Pregunta 3: *Resuma, utilizando la siguiente tabla, las condiciones de inferencia de un cruce en el CM, cuando un bloque está en un estado transitorio. Indique la respuesta, el cambio de estado y la actualización del VP, si es el caso.*

		PtXl	PtXm
ML	$P \notin VP$		
	$P \in VP$		
MM	$P \notin VP$		
	$P \in VP$		

Ejercicio 7.3

Consideramos el protocolo de directorio MLI denominado A. La idea es aprovechar la inferencia de un cruce de peticiones para reducir el tráfico de mensajes. En concreto, se quiere aprovechar la petición PtObE del CM, cuando se cruza con una petición PtXm o PtXl de un CC, para utilizarla como respuesta en el CC. El CC responde a la petición PtObEsi es el caso. En estas condiciones el CM, al inferir el cruce, no responde a una petición PtXm o PtXl. Mediante esta actuación, el CC está menos tiempo bloqueado y además el CM

estará menos tiempo ocupado. Tenga en cuenta que en el multiprocesador utilizado la RV mantiene el orden de los mensajes emitidos desde el CM e implementa las redes lógicas RCM y RV.



El CM bloquea el análisis de la cola CP cuando la petición en la cabeza de la misma accede a un bloque en un estado transitorio.

El contenido del VP se actualiza al pasar de un estado transitorio al estado estable.

Un bloque en el estado LI en el CC2 puede recibir una petición PtObE. Esta petición ha sido emitida por el CM al procesar una petición PtIm de otro CC1, antes de procesar la petición PtXl del CC2. En el CC2 la petición PtObE se utiliza como respuesta del CM a la petición PtXl. El CM no responde a la petición PtXl cuando infiere el cruce de peticiones.

En el CM, la petición PtXl puede procesarse en el estado M o en el estado L, en función del entrelazado de accesos al bloque. Las siguientes secuencias de accesos permiten analizar los distintos casos.

accesos A	
P1 store t	
P2 PtXl t	

estado	
C1	I
M	2
C2	L

accesos B	
P1 store t	
P3 load t	
P2 PtXl t	

C1	I
M	2
C2	L
C3	I

Pregunta 1: *Analice si es factible que el CC utilice la petición PtObE del CM, para un bloque en el estado LI, como respuesta del CM a una petición PtXl. Para ello, utilice las secuencias de accesos previas. Muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Determine la información en el VP al procesarse la expulsión e indique si hay que actualizar la memoria, si es el caso. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.*

Un bloque en el estado MI en el CC2 puede recibir una petición PtObE. Esta petición ha sido emitida por el CM al procesar una petición PtIm de otro CC1, antes de procesar la petición PtXm del CC2. En el CC2 la petición PtObE se responde y se utiliza como respuesta del CM a la petición PtXm. El CM no responde a la petición PtXm cuando infiere el cruce de peticiones.

En el CM, la petición PtXm puede procesarse en el estado M o en el estado L, en función del entrelazado de accesos al bloque. Las siguientes secuencias de accesos permiten analizar los distintos casos.

accesos A	accesos B
P1 store t	P1 store t
P2 PtXm t	P3 load t
	P2 PtXl m

estado

C1	I	C1	I
M	2E	M	2E
C2	M	C2	M
		C3	I

Pregunta 2: *Analice si es factible que el CC utilice la petición PtObE del CM, para un bloque en el estado LI, como respuesta del CM a la petición PtXm. Para ello, utilice las secuencias de accesos previas. Muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Determine la información en el VP al procesarse la expulsión e indique si hay que actualizar la memoria, si es el caso. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.*

Un bloque en el estado MI en el CC2 puede recibir una petición PtObL. Esta petición ha sido emitida por el CM al procesar una petición Pt de otro CC1, antes de procesar la petición PtXm del CC2. En el CC2 la petición PtObL se responde y se utiliza como respuesta del CM a la petición PtXm. En el CM, la petición PtXm puede procesarse en el estado M o en el estado L, en función del entrelazado de accesos al bloque. El CM no responde a la petición PtXm cuando infiere el cruce de peticiones, Las siguientes secuencias de accesos permiten analizar los distintos casos.

accesos A	accesos B
P1 load t	P1 load t
P2 PtXm t	P3 store t
	P2 PtXm t

estado

C1	I	C1	I
M	2E	M	2E
C2	M	C2	M
		C3	I

Pregunta 3: *Analice si es factible que el CC utilice la petición PtObL del CM para un bloque en el estado MI como respuesta del CM a la petición PtXm. Para ello, utilice las secuencias de accesos previas. Muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Determine la información en el VP al procesarse la expulsión e indique si hay que actualizar la memoria, si es el caso. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.*

El análisis de las secuencias de accesos previas muestra que un CC puede recibir una petición PtObE estando el bloque en el estado I. Este comportamiento indica que el directorio es impreciso.

En estas condiciones hay que analizar si es posible que un CC reciba una petición PtObE estando el bloque en el estado IM o el estado IL. Las siguientes secuencias de accesos permiten analizar los distintos casos.

accesos A
P1 load t
P3 store t
P2 PtXm t
P2 store t

estado	
C1	I
M	2E
C2	M

accesos B
P1 load t
P3 store t
P2 PtXm t
P2 load t

C1	I
M	2E
C2	M
C3	I

En estas secuencias se supone que la expulsión del bloque en CC2 ha sido voluntaria. Esto es, no está determinada por un reemplazo. Si estuviera determinada por un reemplazo, se puede suponer que hay varios módulos de memoria y el bloque que determina el reemplazo está almacenado en otro módulo de memoria desocupado. La petición a este bloque es de lectura y se efectúa inmediatamente, ciclo siguiente al de la expulsión. Posteriormente se expulsa este bloque y se vuelve a referenciar el bloque expulsado previamente.

Mientras esté pendiente la expulsión, cualquier acceso al mismo bloque bloquea al procesador. El mensaje no se emite hasta que el bloque expulsado está en el estado I.

Pregunta 4: *Analice si es factible que un CC tenga que procesar una petición PtObE estando el bloque en el estado IM o el estado IL. Para ello, utilice las secuencias de accesos previas. Muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.*

Pregunta 5: *Resuma, utilizando la siguiente tabla, las condiciones de inferencia de un cruce en el CM. Indique la respuesta, el cambio de estado y la actualización del VP, si es el caso.*

		PtXl	PtXm
L	P ∉ VP		
	P ∈ VP		
M	P ∉ VP		
	P ∈ VP		

Un diseñador propone el siguiente mecanismo para mantener el directorio preciso. Cuando el CM emite una petición PtObL a un CC, se mantiene en un campo adicional del VP la identidad del CC que tiene la exclusividad.

Este campo adicional se invalida al procesarse una petición PtXm del CC que identifica.

Este campo adicional se utiliza, si es valido, al procesar una petición PtIm, para excluir al CC identificado de los CC que reciben una petición PtObE. Además se invalida el campo adicional.

Pregunta 6: *Analice si el mecanismo propuesto mantiene el directorio preciso. Para ello, seleccione, de las secuencias de accesos previas, las secuencias que sean de utilidad. Muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Determine la información en el VP, especificando el contenido del campo adicional (CA), al procesarse la expulsión e indique si hay que actualizar la memoria, si es el caso. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.*

Ejercicio 7.4

Consideramos el protocolo de directorio MLI denominado A. La idea es aprovechar la inferencia de un cruce de peticiones para reducir el tráfico de mensajes. En concreto, se quieren aprovechar la petición PtObE del CM, cuando se cruza con una petición PtXm o PtXl de un CC, para utilizarla como respuesta en el CC. Esto es, se reduce el tráfico de respuestas. Por otro lado, se quiere disponer de un directorio preciso.

Tenga en cuenta que, en el multiprocesador utilizado, la RV mantiene el orden de los mensajes emitidos desde el CM e implementa las redes lógicas RCM y RV.

El CM bloquea el análisis de la cola CP cuando la petición en la cabeza de la misma accede a un bloque en un estado transitorio.

El contenido del VP se actualiza al pasar de un estado transitorio al estado estable.

Un ingeniero, después de estudiar en detalle los cruces de peticiones propone lo siguiente:

- Cuando se procesa en el CM una petición PtXm, siendo L el estado del bloque en el directorio, y el CC, que ha emitido la petición, está activado en el VP se responde RpX. Además se desactiva al CC del VP.
- Una petición PtObL del CM no se puede utilizar en un CC como respuesta a una petición PtXm del CC. El CC debe esperar una respuesta explícita (RpX) o una petición PtObE del CM.
- El CM no responde a una petición PtXl si el CC no está en el VP.
- Un CC utiliza una petición PtObE del CM como respuesta a una petición PtXm o PtXl. Si es el caso, suministra el bloque.

El ingeniero ha observado que en el CM es necesario discernir si hay que responder o no a una petición PtXm. Para ello, utiliza el estado del bloque en el directorio.

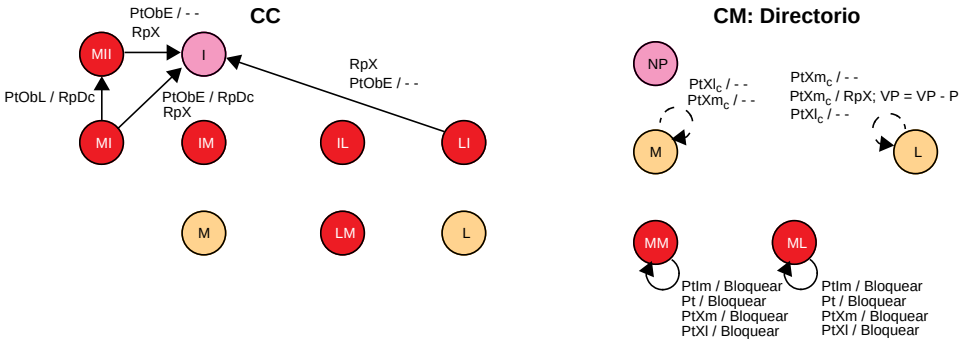
Sea el CC2 el que emite una petición PtXm. Si esta petición se procesa en el CM, siendo M el estado del bloque en el directorio, y el VP conjuntamente con el BE identifican a un CC3, el CM ha ordenado previamente una petición PtIm del CC3.

El CM al procesar la petición PtIm del CC3 emite una petición PtObE al CC2, ya que tiene el bloque en exclusividad (MI). En este caso, el CC2 suministra el bloque y además, utiliza la petición PtObE como respuesta a la petición PtXm.

Por otro lado, el CC2 puede recibir una petición PtObL cuando el bloque está en el estado MI. Para ello, el CM ha procesado una petición Pt de un CC1, antes que la petición PtXm del CC2. El CC2 suministra el bloque al procesar la petición PtObL del CM. En el VP del bloque en el CM se identifica a dos CC que tienen copia del bloque.

En consecuencia es necesario discernir en el CC2 entre una petición PtObL y una petición PtObE. Para ello se utiliza el estado MII. En este estado se espera una respuesta RpX del CM a la petición PtXm o una petición PtObE. La respuesta RpX la emite el CM cuando procesa la petición PtXm del CC2 y el estado del bloque en el directorio es L. En este estado no es esperada una petición PtXm. El CM infiere un cruce, responde al CC2 y lo extrae del VP.

En el siguiente diagrama de transiciones entre estados se muestra la propuesta descrita.



Un bloque en el estado LI en el CC2 puede recibir una petición PtObE. Esta petición ha sido emitida por el CM al procesar una petición PtIm de otro CC1, antes de procesar la petición PtXl del CC2. En el CC2 la petición PtObE se utiliza como respuesta del CM a la petición PtXl. El CM no responde a la petición PtXl cuando infiere el cruce de peticiones.

En el CM, la petición PtXl puede procesarse en el estado M o en el estado L, en función del entrelazado de accesos al bloque. Las siguientes secuencias de accesos permite analizar los distintos casos.

accesos A	accesos B
P1 store t	P1 store t
P2 PtXl t	P3 load t
	P2 PtXl t

estado

C1	I
M	2
C2	L

C1	I
M	2
C2	L
C3	I

En las siguientes preguntas muestre en un diagrama temporal la evolución de los estados en el CM y los CC. Determine la información en el VP al procesarse la expulsión e indique si hay que actualizar la memoria, si es el caso. Justifique la respuesta teniendo en cuenta el comportamiento de algún o algunos componentes del multiprocesador.

Pregunta 1: *Utilice la propuesta del ingeniero para mostrar el flujo de mensajes en las secuencias de accesos previas.*

Un bloque en el estado MI en el CC2 puede recibir una petición PtObE. Esta petición ha sido emitida por el CM al procesar una petición PtIm de otro CC1, antes de procesar la petición PtXm del CC2. El CC2 responde a la petición PtObE y la utiliza como respuesta del CM a la petición PtXm. El CM no responde a la petición PtXm cuando infiere el cruce de peticiones.

En el CM, la petición PtXm puede procesarse en el estado M o en el estado L, en función del entrelazado de accesos al bloque. Las siguientes secuencias de accesos permite analizar los distintos casos.

accesos C	
P1 store t	
P2 PtXm t	
estado	
C1	I
M	2E
C2	M

accesos D	
P1 store t	
P3 load t	
P2 PtXm t	
C1	I
M	2E
C2	M
C3	I

Pregunta 2: *Utilice la propuesta del ingeniero para mostrar el flujo de mensajes en las secuencias de accesos previas.*

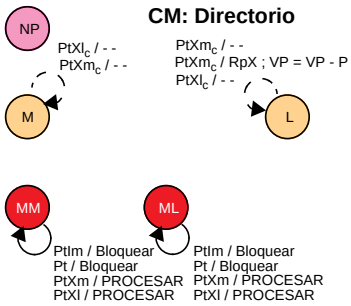
Un bloque en el estado MI en el CC2 puede recibir una petición PtObL. Esta petición ha sido emitida por el CM al procesar una petición Pt de otro CC1, antes de procesar la petición PtXm del CC2. En el CM, la petición PtXm puede procesarse en el estado M o en el estado L, en función del entrelazado de accesos al bloque. Las siguientes secuencias de accesos permite analizar los distintos casos.

accesos E	
P1 load t	
P2 PtXm t	
estado	
C1	I
M	2E
C2	M

accesos F	
P1 load t	
P3 store t	
P2 PtXm t	
C1	I
M	2E
C2	M
C3	I

Pregunta 3: *Utilice la propuesta del ingeniero para mostrar el flujo de mensajes en las secuencias de accesos previas.*

Una optimización, que permite reducir la ocupación en del directorio, es que el CM procese los mensajes de expulsión en estados transitorios, además de en estados estables.



Tenga en cuenta que el VP se actualiza cuando el CM responde al solicitante. En un estado transitorio se mantiene la información que habia en el estado estable del que se proviene.

Respecto a las secuencias de accesos que se deben seleccionar tenga en cuenta que un CC tiene que suministrar un bloque.

Pregunta 4: *Muestre el procesado de una petición PtXl cuando el bloque está en el estado ML en el directorio.*

Pregunta 5: *Muestre el procesado de una petición PtXm cuando el bloque está en el estado MM en el directorio.*

Pregunta 6: *Muestre el procesado de una petición PtXm cuando el bloque está en el estado ML en el directorio.*

Ejercicio 7.5

Consideramos el protocolo de directorio VI denominado A.

Pregunta 1: *Represente mediante un diagrama temporal simplificado las secuencias de acceso mostradas en la Figura 7.18 y en la Figura 7.19.*

Ejercicio 7.6

Un multiprocesador utiliza un esquema de directorio con un protocolo de coherencia con invalidación (MLI) denominado B.

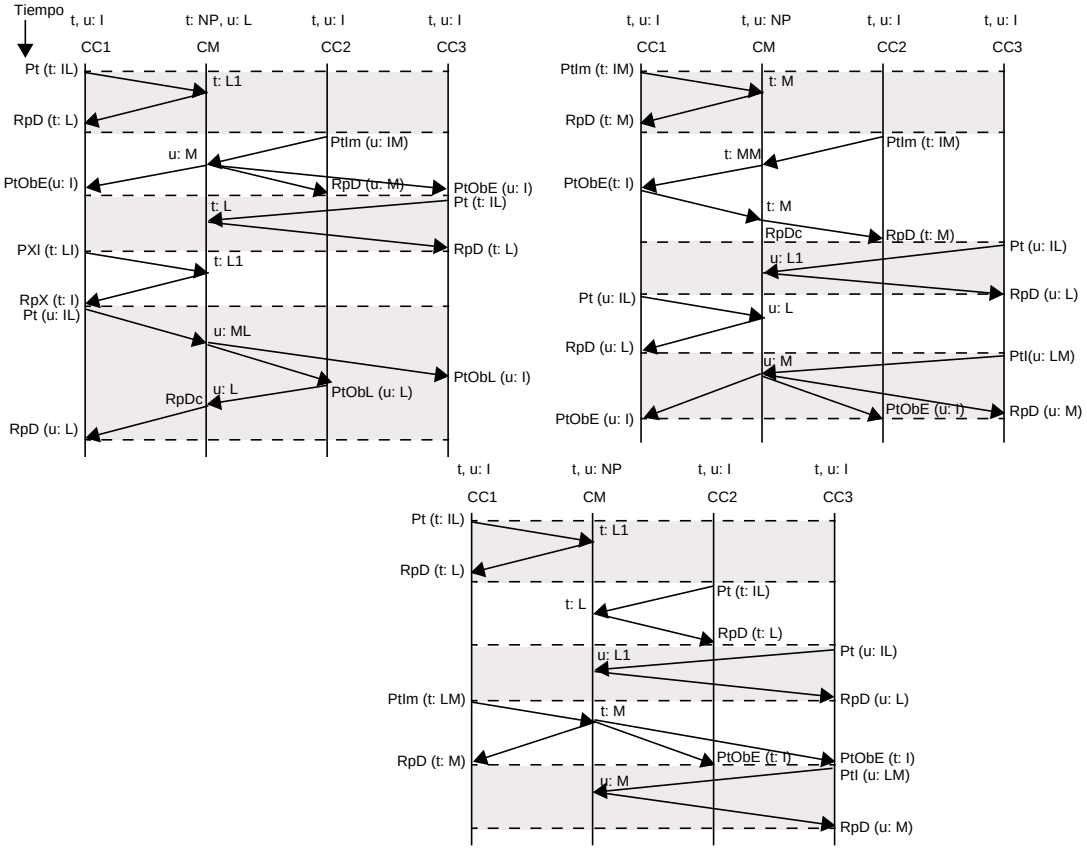
Supondremos un arbitraje por antigüedad en la red de peticiones de los CC al CM y que la cola de peticiones se gestiona de forma FIFO.

Suponga las siguientes secuencias de accesos a memoria. Las variables t y u están ubicadas en bloques distintos.

A. accesos	1. P1 load t	2. P2 store u	3. P3 load t	4. P1 PtXl t	5. P1 load u
B. accesos	1. P1 store t	2. P2 store t	3. P3 load u	4. P1 load u	5. P3 store u
C. accesos	1. P1 load t	2. P2 load t	3. P3 load u	4. P1 store t	5. P3 store u

En la secuencia A otras caches tienen copia de u. En todas las secuencias, los bloques que contienen las variables t y u se almacenan en el mismo contenedor de cache.

Pregunta 1: *En los siguientes diagramas temporales simplificados, indique la transacción o la ausencia de la misma que no cumple el protocolo. Muestre para esta transacción la secuencia de mensajes y estados correcta.*



Podemos decir que el directorio es impreciso. Entonces, un diseñador propone que un CC no notifique al directorio un reemplazo de un bloque en estado L.

Suponga la siguiente secuencia de accesos a memoria. Las caches sólo disponen de un contenedor. Los variables u y t están ubicadas en bloques distintos. Al empezar la secuencia de accesos no hay copia de los bloques en las caches.

D. accesos	1. P1 load t	2. P1 load u	3. P1 load t	4. P1 store t

Pregunta 2: *Utilice la anterior secuencia de accesos a memoria para mostrar que la propuesta del diseñador es desacertada, aunque el protocolo es correcto. Indique el estado de los bloques en la cache y en el directorio una vez el CC y el CM han efectuado todas las acciones necesarias para servir cada acceso a memoria (tenga en cuenta los reemplazos, si es el caso).*

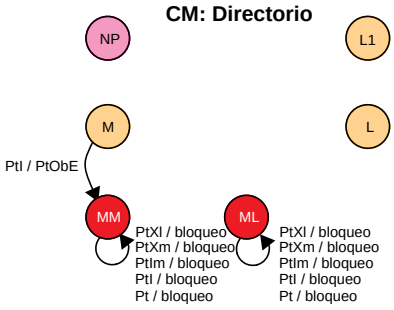
Hay notificación al expulsar un bloque en estado L	Cache		Directorio		Peticiones del CM a los CC
	t	u	t	u	
1. P1 load t					
2. . .					

No hay notificación al expulsar un bloque en estado L	Cache		Directorio		Peticiones del CM a los CC
	t	u	t	u	
1. P1 load t					
2. . . .					

Suponga que en el directorio el estado del bloque que contiene la variable t es L, debido a que caches distintas de C1, C2 y C3 tiene copia del bloque. Las siguientes secuencias de acceso son concurrentes.

E. accesos	1. P2 store t	2. P3 load t	3. P1 load t	Orden de procesado en el CM
F. accesos	1. P2 store t	2. P3 load t	3. P1 store t	

En la siguiente pregunta se analizan cruces de peticiones en los CC. El directorio bloquea el procesado de cualquier petición en un estado transitorio. Estas transiciones entre estados no están especificadas en la descripción del protocolo. Por otro lado, la recepción en el CM de una petición PtI, con el bloque en el estado M, se gestiona de la misma forma que una petición PtIm (el directorio infiere que se ha producido un cruce).



Pregunta 3: *Muestre en un diagrama temporal los cruces de peticiones que se producen en las secuencias E y F. Indique el estado del bloque en las caches cuando se produce el cruce y la respuesta que debe emitir el CC correspondiente, si es el caso.*

Pregunta 4: *Proponga una secuencia de accesos a memoria donde se observe un cruce de peticiones en el estado LM de un bloque en cache.*

Suponga las siguientes secuencias de accesos a memoria. En la dos secuencia el estado del bloque, que contiene la variable t, en la C1 es L y no hay copias en las otras caches. El estado del bloque en el directorio es L, debido a que la cache C3 ha tenido el bloque en el estado L y lo ha expulsado.

EE. accesos	1. P2 store t	2. P3 store t	3. P1 store t	Orden de procesado en el CM
FF. accesos	1. P2 store t	2. P3 load t	3. P1 store t	

Pregunta 5: *Muestre en un diagrama temporal simplificado los cruces de peticiones que se producen en las secuencias EE y FF. Indique el estado del bloque en el directorio cuando se produce el cruce y la acción que efectúa el CM.*

En la siguiente pregunta supondremos que el arbitraje en la red de peticiones no es por antigüedad o la gestión de la cola de peticiones no es FIFO.

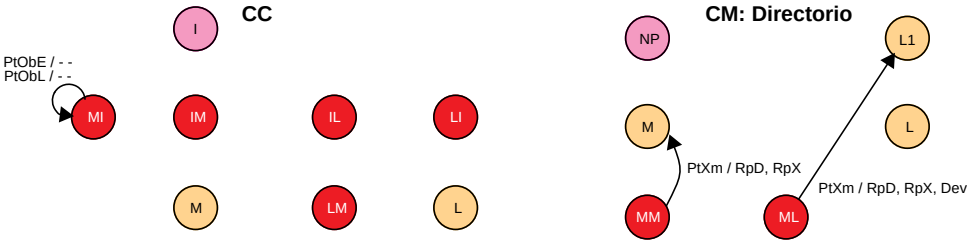
Supongamos la siguiente secuencia de accesos a memoria. El estado del bloque en el directorio es L1 y en la cache P1 es L.

LL. accesos	1. P2 store t	2. P2 PtXm t	3. P3 load t	4. P1 store t	Orden de procesado en el CM

Pregunta 6: *Para la secuencia previa de accesos a memoria, muestre un diagrama temporal simplificado. Indique si el protocolo funciona correctamente. Razone la respuesta.*

En las siguientes preguntas supondremos un arbitraje por antigüedad en la red de peticiones de los CC al CM y que la cola de peticiones se gestiona de forma FIFO.

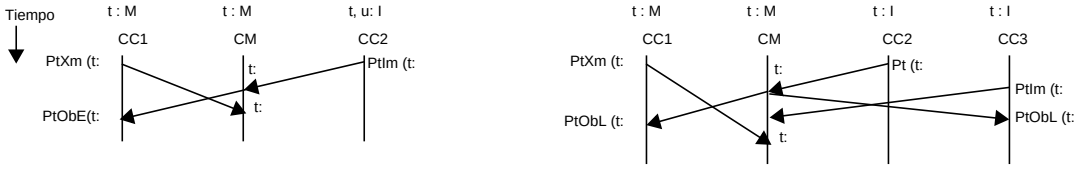
Para gestionar un cruce de peticiones, siendo una de ellas una petición PtXm un ingeniero propone que esta petición se transmita por la red RCM (respuestas) y el CM la gestione, cuando está en un estado transitorio, como una respuesta a su petición. En los siguientes diagramas de transiciones entre estados se muestra el cruce.



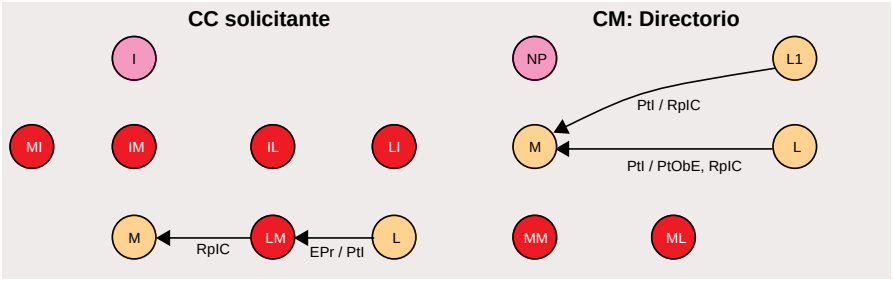
Para analizar los cruces en estados transitorios se utilizan las siguientes dos secuencias de accesos concurrentes a memoria.

G. accesos	1. P2 store t	2. P1 PtXm t	
H. accesos	1. P2 load t	2. P3 store t	3. P1 PtXm t

Pregunta 7: *Dados los siguientes diagramas temporales simplificados, etiquete el estado del bloque en las caches y el CM y añada los mensajes (etiquetandolos) que faltan en el orden temporal adecuado para que finalicen las transacciones iniciadas.*



Un ingeniero propone añadir una nueva respuesta del CM, que se utiliza para responder a una petición PtI. Esta respuesta, cuyo acrónimo es RpIC, no contiene el bloque, ya que un CC utiliza la petición PtI cuando tiene el bloque en estado L.

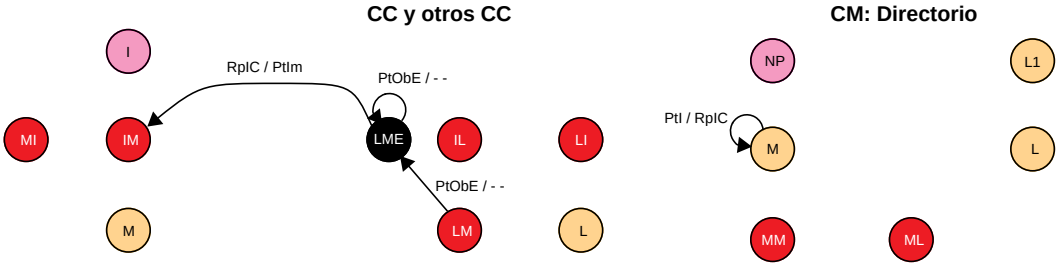


Al analizar cruces de peticiones hay que tener en cuenta, entre otros, el cruce de una petición PtI con una petición PtObE del CM, inducida por una petición PtIm o PtI de otro CC.

La gestión de los cruces se efectúa básicamente en el CC con la colaboración del CM. El CM al procesar una petición PtI en el estado M, infiere un cruce y responde RpIC al CC que ha emitido la petición. El estado no se modifica. El CC inferirá, al recibir está respuesta, en función del estado del bloque en la cache, que la petición PtI ha sido rechazada por el CM.

Un CC, antes de recibir la repuesta a una petición PtI, puede observar una petición PtObE, la cual invalida su copia del bloque. El CC al recibir la respuesta RpIC infiere un cruce de peticiones y solicita el bloque con intención de modificación (PtIm). Para identificar el cruce se utiliza un estado transitorio (LME).

El CC al recibir RpIC, en el estado LME, infiere que el estado del bloque en el directorio es M.



Suponga la siguiente secuencia de accesos a memoria. El estado del bloque en el directorio es L1 y en la cache C1 es L. En las otras caches el estado es I.

I. accesos	1. P2 store t	2. P1 store t	Orden de procesado en el CM

Pregunta 8: *Para la secuencia previa de accesos a memoria, muestre un diagrama temporal simplificado.*

Por otro lado, un CC, después de observar una petición PtObE en el estado LM, puede observar una petición PtObL, antes de recibir la respuesta RpIC a su petición PtI. En estas condiciones el CC infiere que el bloque está en el estado L en el directorio. El CC tiene el bloque invalidado (PtObE) y por tanto, al recibir RpIC emite una petición PtIm.

Por su parte, el CM al procesar la petición PtI, en el estado L, ha establecido como estado del bloque en el directorio M y ha emitido peticiones PtObE a todos los CC, excluyendo al CC que ha emitido PtI. Por tanto, el CM presupone que el CC que ha emitido PtI tiene el bloque en exclusividad; lo cual no es cierto. Sea CC1 el CC que ha emitido PtI y posteriormente emite PtIm.

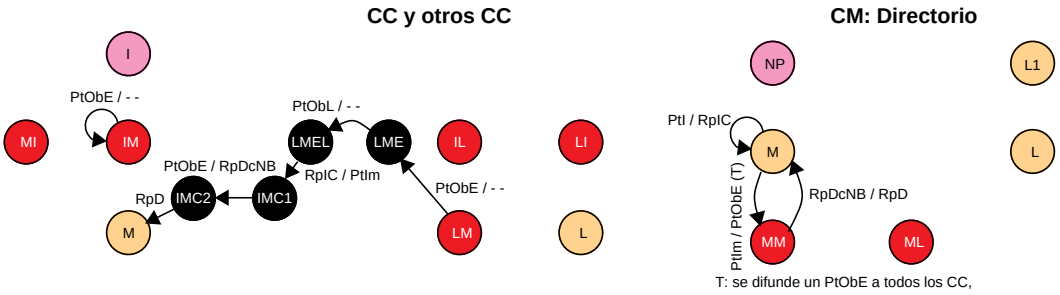
Para simplificar, por ahora, suponemos que el CM procesa la petición PtIm del CC1, antes que peticiones de otros CC (Pt, PtIm, PtI).

Cuando el CM procesa la petición PtIm del CC1 difunde una petición PtObE a todos los CC, excluyendo al CC1. Ninguno de los CC que reciben la petición responde, ya que tienen el bloque en el estado I. En consecuencia se produce un abrazo mortal.

Para soslayar la situación descrita, una alternativa es que el CM al procesar una petición PtIm en el estado M difunda una petición PtObE a todos los CC; sin excluir a ninguno (es una modificación respecto del protocolo base). Entonces, el CC1 responde RpDcNB (nueva respuesta), indicando que no tiene el bloque.

El CM en la situación descrita (estado MM) está esperando una respuesta (RpDc, RpDcNB). Cuando el CM recibe como respuesta RpDcNB infiere que tiene una copia válida del bloque y responde al CC cuya petición está procesando.

Para efectuar el procesado descrito se utilizan tres estados transitorios adicionales en un CC.

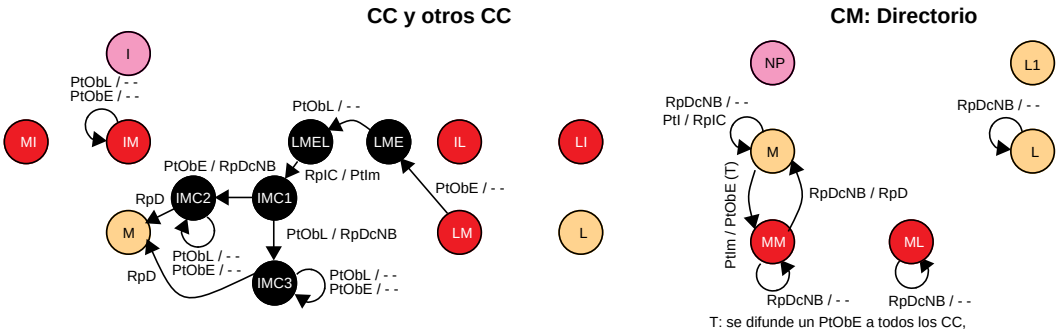


Suponga la siguiente secuencia de accesos a memoria. El estado del bloque en el directorio es L1 y en la cache C1 es L. En las otras caches el estado es I.

J. accesos	1. P2 store t	2. P3 load t	3. P1 store t	Orden de procesado en el CM

Pregunta 9: *Para la secuencia previa de accesos a memoria, muestre un diagrama temporal simplificado.*

El CM, antes de procesar la petición PtIm del CC1, puede procesar peticiones de otros CC (PtIm, Pt, PtI). Mediante la respuesta RpDcNB, el CM infiere que tiene una copia válida del bloque. Entonces, el CC que ha recibido RpIC, también debe responder RpDcNB a una petición PtObL, cuando ha detectado un cruce.

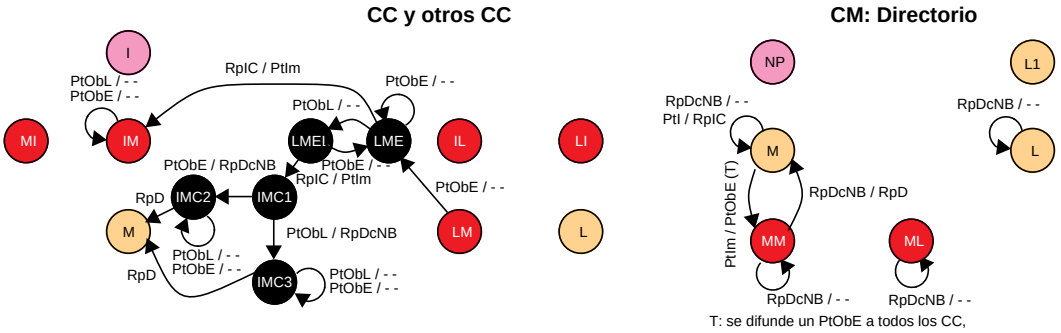


Suponga la siguiente secuencia de accesos a memoria. El estado del bloque en el directorio es L1 y en la cache C1 es L. En las otras caches el estado es I.

K. accesos	1. P2 store t	2. P3 load t	3. P1 store t	4. P4 store t	Orden de procesado en el CM

Pregunta 10: *Para la secuencia previa de accesos a memoria, muestre un diagrama temporal simplificado. Note que algunos CC generan peticiones que no están explicitas en la anterior secuencia. Estas peticiones se procesan en el CM en el orden de llegada. Llegan al CM después de que las especificas en la secuencia de accesos hayan sido emitidas y encoladas en la cola de peticiones.*

Una vez un CC ha emitido un petición PtI, mientras no haya recibido una respuesta RpIC, debe de estar infiriendo el estado del bloque en el directorio (M, L), ya que el CM puede servir peticiones de otros CC. Para ello, hay que disponer de reconocedores de secuencias en el diagrama de estado de un CC. En la siguiente figura se muestra el diseño efectuado por un ingeniero.



Pregunta 11: *Indique las secuencias que se reconocen (expresiones regulares) antes de que un CC reciba una respuesta RpIC, después de haber emitido una petición PtI.*

Ahora supondremos que el arbitraje en la red de peticiones no es por antigüedad o la gestión de la cola de peticiones no es FIFO.

Supongamos la siguiente secuencia de accesos a memoria. El estado del bloque en el directorio es L1 y en la cache P1 es L.

L. accesos	1. P2 store t	2. P2 PtXm t	3. P3 load t	4. P1 store t	Orden de procesado en el CM

Pregunta 12: *Para la secuencia previa de accesos a memoria, muestre un diagrama temporal simplificado. Indique si el protocolo funciona correctamente. Razone la respuesta.*

Pregunta 13: *Un ingeniero propone que un CC notifique la expulsión de un bloque en estado L, pero que no espere respuesta. Indique de forma justificada si la propuesta mantiene la coherencia.*

Ejercicio 7.7

Un multiprocesador utiliza un esquema de directorio con un protocolo de coherencia con invalidación (VI) denominado A. Suponga la siguiente secuencia de accesos a memoria concurrentes.

referencia	referencia	
1 P1 load t P3 store u	3 P1 store t P3 load u	grupos de dos accesos concurrentes
2 P1 load u P2 load t	4 P1 store u P3 store u	

Pregunta 1: *Muestre en un diagrama temporal los mensajes de las transacciones que se inician y el estado en las caches y en la memoria al ejecutarse el entrelazado de accesos a memoria previo. Indique el número de expulsiones y el número de cruces de peticiones. En cada uno de ellos especifique el grupo de accesos a memoria en el cual se producen.*

En las siguientes preguntas utilice una secuencia de dos transacciones concurrentes, iniciadas por CC distintos, para mostrar lo que se solicita. Indique el estado inicial del bloque en las caches y el VP en el directorio y represente las transacciones en un diagrama temporal.

Pregunta 2: *En el CM es necesario procesar una petición PtX en el estado Pr y el CC no esté identificado en el VP.*

Pregunta 3: *En el CM es necesario procesar una petición PtX en el estado NP y la petición no es esperada.*

El diseñador decide que la expulsión de un bloque de cache no se notifica al directorio, lo cual se denomina expulsión silenciosa. Esto es, el directorio es impreciso.

En un diagrama temporal, el estado del bloque al efectuar una expulsión silenciosa se indica en la fase arb de la transacción que expulsa el bloque.

Para efectuar el análisis de las transiciones entre estados que hay que añadir el diseñador utiliza las siguientes secuencias de accesos a memoria concurrentes. En las tres secuencias hay tres grupos de accesos. En ocasiones, en un grupo sólo hay un acceso a memoria.

A	B	C
referencia	referencia	referencia
P1 load t	P1 load t	P1 load t
P1 load u	P1 load u	P1 load u
P2 store t	P2 store t	P2 store t
	P1 load t	P1 store t

Las variables t y u están contenidas en bloque distintos.

Estos bloques se almacenan en el mismo contenedor de cache.

En las siguientes preguntas, en los casos en los cuales no exista una transacción para un par petición-estado, en la descripción del protocolo, determine las acciones que deben efectuarse.

Pregunta 4: *Utilizando expulsión silenciosa, muestre un diagrama temporal para las secuencias A, B y C.*

Pregunta 5: *Muestre en el diagrama de estados, correspondiente al agente procesador de un CC, la transición entre estados en una acción de reemplazo. Muestre también en el agente observador de un CC los estados en los cuales se pueden recibir peticiones que son debidas exclusivamente a que el directorio es impreciso. Indique la transición entre estados y la respuesta.*

Ejercicio 7.8

Un multiprocesador utiliza un esquema de directorio con un protocolo de coherencia con invalidación (VI) denominado A. En este ejercicio la memoria se construye utilizando dos módulos de memoria (M1 y M2) con el correspondiente directorio.

Suponga la siguiente secuencia de accesos a memoria concurrentes, la cual se corresponde con el esqueleto de una sincronización mediante eventos.

referencia	M	Valor almacenado en los registros	Módulo de memoria	Valor en memoria	Contenido en cache
P1 store R4, A	M2	El contenido de R4 es 9	M1: aviso	A: 7	Los bloques que contienen las variables no están almacenado en ninguna cache
P1 store R5, aviso	M1	El contenido de R5 es 1	M2: A	aviso: 0	
P2 load R6, aviso	M1				
P2 load R7, A	M2				

referencia	ciclos 1	2	3	4	5	6	7	8	9	10	11
P1 store R4, A	arb	arb	arb	arb	arb	arb	arb	arb	arb	arb	RI
P1 store R5, aviso		arb	RI								
P2 load R6, aviso	arb	arb	arb	RI							
P2 load R7, A											

En un diseño un ingeniero decide que un procesador puede emitir una petición de escritura sin esperar la confirmación de la anterior petición de escritura. Una petición de lectura espera que finalice el anterior acceso a memoria y también es bloqueante. Debido a conflictos en la RI y acciones del arbitraje, las peticiones de los CC ocupan la RI en los ciclos que se muestran en la figura adjunta.

En un diagrama temporal, en la fase D de una transacción indique el valor leído de memoria. En la fase Mx indique el valor que se ha almacenado en memoria.

Pregunta 1: *Complete el diagrama temporal. Indique si se cumple consistencia secuencial. Para ello indique el orden temporal en el cual la secuencia ha actualizado o leído de la memoria (fase M1, M2). Utilice también el contenido de los registros. En la fase M de una transacción utilice el ordinal del módulo de memoria (x) para indicar el módulo accedido (Mx).*

El lenguaje máquina dispone de la instrucción denominada barrera. El objetivo de la misma es que no se inicie ningún acceso a memoria, que haya sido especificado posteriormente, hasta que todos los accesos previos a la instrucción barrera hayan consolidado.

Pregunta 2: *Indique en las secuencias de instrucciones que se ejecutan en P1 y P2 dónde es necesario incluir instrucciones barrera. El número de inserciones debe ser el mínimo necesario.*

Por otro lado, en otro diseño el ingeniero decide que un procesador puede emitir una petición de lectura sin esperar el valor devuelto

referencia	ciclos 1	2	3	4	5	6	7	8	9	10	11
P1 store R4, A	arb	arb	arb	RI							
P1 store R5, aviso											
P2 load R6, aviso	arb	arb	arb	arb	arb	arb	arb	arb	arb	arb	RI
P2 load R7, A		arb	RI								

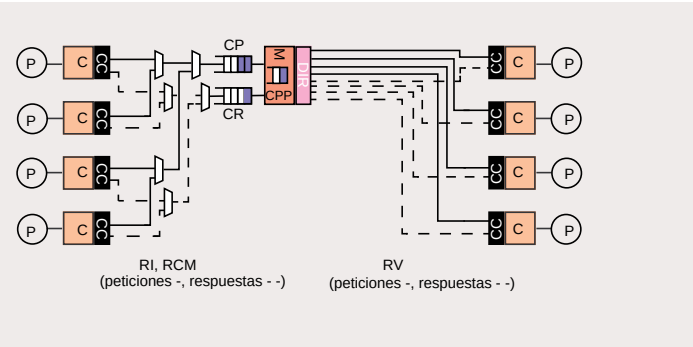
en la petición de lectura previa. Una petición de escritura espera la finalización del anterior acceso a memoria y es bloqueante. Debido a conflictos en la RI y acciones del arbitraje, las peticiones de los CC ocupan la RI en los ciclos que se muestran en la figura adjunta.

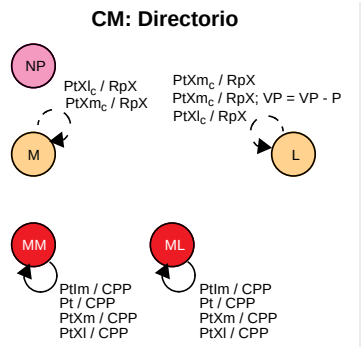
Pregunta 3: *Complete el diagrama temporal. Indique si se cumple consistencia secuencial. Para ello indique el orden temporal en el cual la secuencia ha actualizado o leído de la memoria (fase M1, M2). Utilice también el contenido de los registros. En la fase M de una transacción utilice el ordinal del módulo de memoria (x) para indicar el módulo accedido (Mx).*

Pregunta 4: *Indique en las secuencias de instrucciones que se ejecutan en P1 y P2 dónde es necesario incluir instrucciones barrera. El número de inserciones debe ser el mínimo necesario.*

Ejercicio 7.9

Consideramos el protocolo de directorio MLI denominado A. Para no bloquear el procesado de peticiones en el CM, una petición que no puede procesarse se extrae de la CP y se almacena en una cola de peticiones pendientes (CPP). Entonces se sigue analizando la CP hasta encontrar una petición que pueda procesarse.





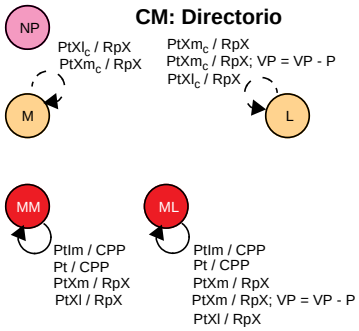
En la CPP se mantiene el orden de llegada de las peticiones. Al finalizar una transacción de 4 pasos el CM analiza la CPP antes que la CP. La CPP se analiza en orden, buscando una petición al mismo bloque que la transacción que acaba de finalizar el procesado.

Pregunta 1: *¿Puede el CM analizar la CP antes que la CPP?*

Cuando no se puede procesar una petición, un diseñador propone que en lugar de utilizar la CPP, el CM extraiga la petición de la CP y la vuelve a encolar después de la última entrada válida.

Pregunta 2: *Justifique si la decisión sigue garantizando un funcionamiento correcto.*

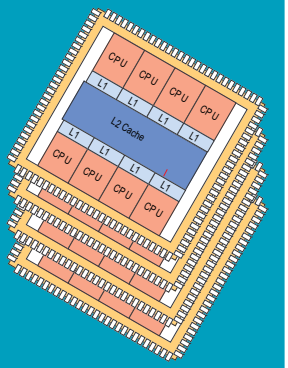
Suponga que el CM responde a las peticiones PtXm y PtXl independientemente del estado en el CM. Esto es, las únicas peticiones que se almacenan en la CPP son peticiones Pt y PtIm que, al ser analizadas por el CM, acceden a un bloque que está en un estado transitorio en el CM.

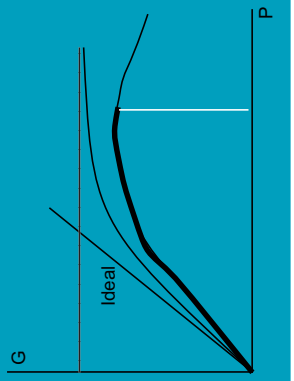


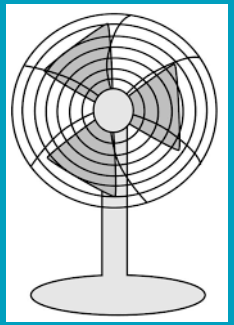
Pregunta 3: *¿Puede el CM analizar la CP antes que la CPP?*

En el contexto de la pregunta previa, un diseñador propone que en lugar de utilizar la CPP, cuando no se puede procesar una petición (Pt, PtIm), el CM extraiga la petición de la CP y la vuelve a encolar después de la última entrada válida.

Pregunta 4: *Justifique si la decisión sigue garantizando un funcionamiento correcto.*







Multiprocesadores

$$P = C_e \cdot v^2 \cdot f + v \cdot I_f$$

J.M. Llabería

Contenido

Capítulo 8
Protocolo de directorio con una red escalable

Una red crossbar es costosa en recursos de cableado y la escalabilidad está limitada cuando se tiene en cuenta el número de puertos[1]. Por otro lado, la longitud de los cables se incrementa al incrementar el número de puertos y con ello el retardo de propagación de la señal, en los mismos, se incrementa[2]. Otra característica es la baja utilización de los recursos dedicados en la implementación de la red.

El objetivo es utilizar otro tipo de redes de interconexión que tengan un menor coste y sean más escalables. La idea es que la topología sea regular y se incremente la utilización de los recursos dedicados. En particular, interesa que la longitud de los enlaces que conectan dos nodos no dependa del número de nodos. La desventaja de este tipo de redes es que no facilitan el establecimiento u observación de un orden lógico global de los accesos a memoria. Por ello, es necesario añadir mecanismos, en los protocolos descritos en el Capítulo 7, para determinar cuando todas las caches, involucradas en una acción de coherencia, han observado una escritura. Esto es, una petición debe serializarse individualmente con respecto a todas las caches que tienen copia. Por ello, son necesarios mensajes explícitos para notificar al agente gestor del bloque que la petición de invalidación ha sido serializada en cada cache que tiene copia.

En el protocolo descrito en el Capítulo 7 no es necesario responder a las peticiones de invalidación, ya que la red crossbar entre el CM y los CC, mediante el arbitraje, mantiene un orden lógico global de los mensajes. En este capítulo las redes de interconexión lógicas mantienen, en algunos casos, el

1. El coste se incrementa cuadraticamente.

2. Una posibilidad es segmentar la propagación para incrementar la concurrencia. Ahora bien, se incrementa la latencia de propagación.

orden de los mensajes entre un emisor y un receptor. En otros casos no existen garantías de recibir en un receptor los mensajes en el mismo orden con el cual son emitios desde un emisor.

En estas condiciones y de forma generica, como no existe un orden global de las peticiones de los CM, es necesario que una petición de invalidación de un CM sea respondida de forma explícita por el CC que la recibe.

Las respuestas de invalidación deben ser recolectadas en un punto o agente. Una vez han sido recolectadas todas las respuestas, se tiene constancia de que la escritura ha sido observada por los CC implicados en la acción de coherencia. Esto es, la escritura está consolidada.

En este capítulo se presentan en primer lugar dos diseños donde las respuestas de invalidación para un bloque son recolectadas en el CM que ha emitido las peticiones de invalidación. En uno de los diseños los mensajes de petición y respuesta de los CM utilizan la misma red de comunicación con los CC y en esta red existe un orden entre un emisor y un receptor. En el otro diseño los mensajes de petición y respuesta de los CM a los CC utilizan redes distintas. Cada una de estas redes mantiene orden entre un emisor y un receptor, pero entre ellas un CC debe inferir el orden de los mensajes que recibe.

En segundo lugar, con el objetivo de reducir la latencia de algunas transacciones, que requieren la colaboración de terceros, se presenta un diseño donde el recolector de las respuestas de invalidación es el CC cuya petición ha inducido que el CM emitiera las peticiones de invalidación.

En todos los protocolos que se ha comentado que se describen, la red que se utiliza para transmitir mensajes de petición de los CM a los CC mantiene el orden entre un emisor y un receptor. El siguiente diseño, que se describe en este capítulo, no mantiene esta propiedad. Los mensajes emitidos desde un CM pueden llegar en un orden distinto al de emisión en el CC destinatario.

En la exposisión que se efectúa en este capítulo se utiliza el protocolo de coherencia MLI.

EL CM ES EL RECEPTOR DE LAS RESPUESTAS

Teniendo en cuenta el encaminamiento de los mensajes desde los CM a los CC podemos distinguir dos diseños: a) los mensajes de petición y respuesta de los CM se encaminan por la misma red y b) los mensajes de petición y respuesta se encaminan por redes distintas.

En los dos casos una red mantiene el orden de los mensajes entre un emisor y un receptor. Entonces, en el caso a) un CC observa el orden de emisión de peticiones y respuestas desde un CM a este CC. En cambio, en el caso b) un CC no observa el orden entre peticiones y respuestas emitidas desde un CM a este CC. Sólo observa orden entre las peticiones y orden entre las respuestas. Por ello, en el protocolo hay que disponer de un mecanismo que permita a un CC inferir el orden entre peticiones y respuestas emitidos por un CM a este CC.

Descripción funcional de la secuencia de mensajes en una transacción

En la Figura 8.1 se muestra la descripción funcional de protocolo del Capítulo 7 y la del protocolo que se describe en el siguiente apartado.

Cuando no hay copia del bloque en otras cache o la transacción es de lectura y una cache no tiene el bloque en exclusividad, la secuencia de mensajes y su serialización es la misma en los dos protocolos (2 pasos, parte superior de la Figura 8.1).

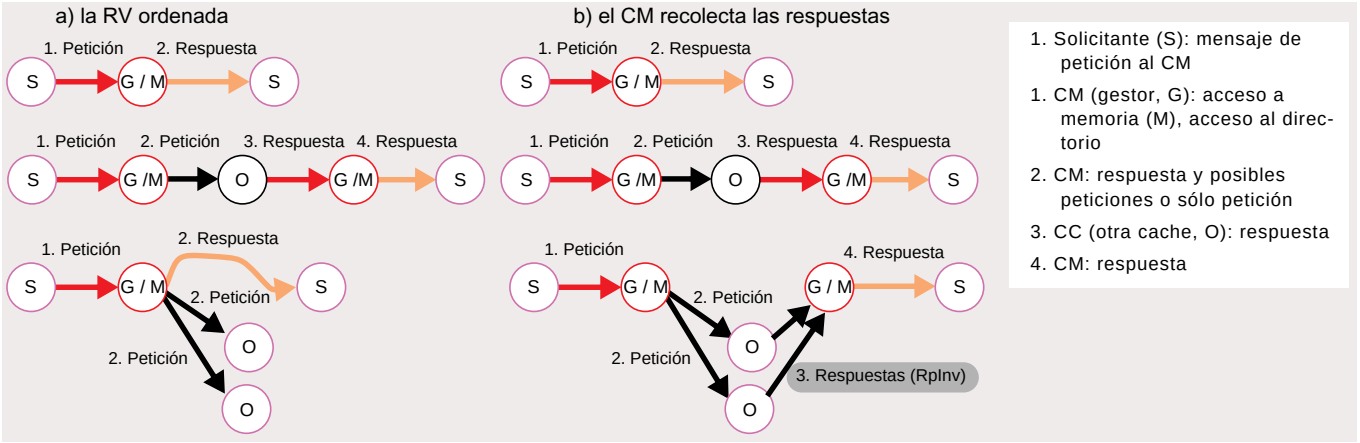


Figura 8.1 Flujos de mensajes en dos protocolo de directorio MLI: a) la red entre los CM y los CC mantiene un orden lógico global de los mensajes (Capítulo 7) y b) el CM recolecta las respuestas de los CC a peticiones de invalidación. El sombreado muestra la diferencia básica con la especificación del Capítulo 7.

Cuando una cache tiene el bloque en exclusividad la secuencia de mensajes es la misma en los dos protocolos (4 pasos). El solicitante recibe el bloque del CM, después de que el CC, que tiene el bloque en exclusividad, ha suministrado el bloque al CM (centro de la Figura 8.1).

Cuando hay caches con copias del bloque y una transacción solicita obtener la exclusividad, las respuestas de invalidación las recolecta el CM (parte inferior de la Figura 8.1). Una vez las respuestas de los CC han sido recolectadas, el CM responde al CC solicitante.

Un CM da por completada o consolidada una transacción una vez emite el mensaje de respuesta al CC solicitante.

PETICIONES Y RESPUESTAS DEL CM SE ENCAMINAN POR LA MISMA RED

El protocolo que se describe es básicamente el mismo que en el Capítulo 7. Ahora bien, el arbitraje en la red que comunica los CM con los CC, y por la que se transmiten peticiones y respuestas del CM, no garantiza que los CC observen el mismo orden lógico global de los mensajes. Sólo se garantiza orden entre un emisor y un destino.

En primer lugar se describe la organización del multiprocesador. Seguidamente se describe el protocolo de coherencia.

Organización del multiprocesador

El protocolo utiliza tres redes lógicas (Figura 8.2). Los CC utilizan la RI para emitir peticiones y la RCM para emitir respuestas al CM. Los CM utilizan la RPR para emitir peticiones y respuestas a los CC.

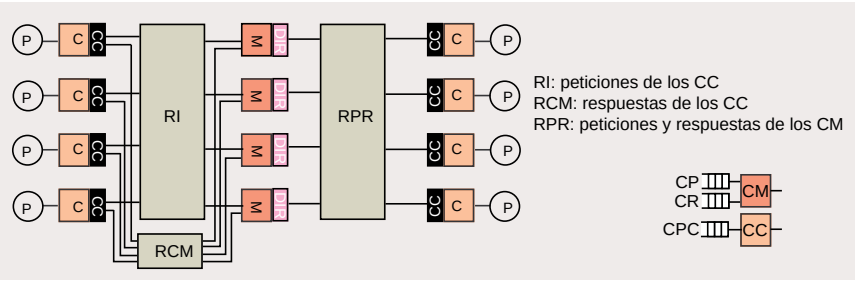


Figura 8.2 Misma red para peticiones y respuestas del CM. Organización del multiprocesador.

La RPR mantiene un orden punto a punto de los mensajes emitidos desde un emisor a un receptor[3].

3. En esta organización la RPR puede corresponderse con una red crossbar, donde el arbitraje es independiente en cada puerto de entrada. También puede corresponderse con una red en malla donde el encaminamiento de los mensajes entre emisor y destino está prefijado. El orden de los mensajes generados por un CM para ser encaminados a un CC se mantiene en los dos casos.

La cola de mensajes de respuestas en un CM se denomina CR. La cola de mensajes de petición y respuesta en un CC se denomina CPC.

Suponemos que un CM genera los mensajes de petición y respuesta en paralelo. Entonces, si los árbitros de cada destino seleccionan los mensajes en el mismo ciclo, los mensajes se propagan en paralelo[4].

Para representar retardo en una red se replica el acrónimo de la red en ciclos consecutivos.

En este capítulo el arbitraje para acceder a un nodo no se puede considerar centralizado. En estas condiciones, en un diagrama temporal no se replicará el acrónimo arb para representar una llegada en serie de los mensajes a un nodo. En su lugar, después del acrónimo arb se utiliza el acrónimo de la cola correspondiente para indicar que el mensaje no se procesa. Ello puede ser debido a dos causas. Una de ellas sería un riesgo estructural. La otra causa sería debida a que el protocolo de coherencia no procesa el mensaje, que está en la cabeza de la cola correspondiente, debido al estado actual del bloque al que hace referencia el mensaje.

En la Figura 8.3 se muestra un ejemplo. En la parte izquierda se muestra la representación utilizada hasta este capítulo y en la parte derecha la representación que se utilizará a partir de ahora.

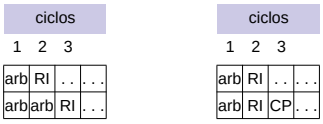


Figura 8.3 Representación en un diagrama temporal de la espera de un mensaje. Ejemplo en la cola de peticiones (CP) del CM.

Cuando varios mensajes llegan en el mismo instante a un destinatario se encolan en paralelo en la cola correspondiente. Un destinatario sólo procesa un mensaje en un instante dado.

Mensajes del protocolo

En la Figura 8.4 se muestra un esquema genérico de la transmisión de mensajes con peticiones y respuestas entre los CC y el CM y viceversa.

En la descripción del protocolo se distinguen los siguientes tipos de mensajes:

4. En los ejemplos y ejercicios usualmente, para simplificar, supondremos que un destinatario puede recibir varios mensajes en un ciclo. Por ejemplo, esta circunstancia es factible en una red en malla. Ahora bien, el procesado en el destinatario es en secuencia.

- Peticiones de un CC a un CM (Pt, PtIm, PtXm, PtXl)
- Peticiones de un CM a los CC (PtObL, PtObE)
- Respuestas de un CM a un CC (RpD, RpX)
- Respuestas de un CC a un CM (RpDc, RpInv)

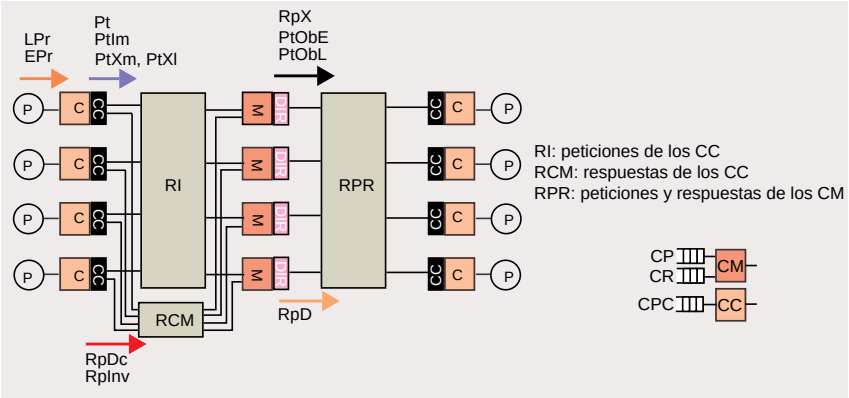


Figura 8.4 Tipos de mensajes en un protocolo MLI.

Los mensajes que se han añadido o modificado respecto al protocolo del Capítulo 7 se muestran en la Tabla 8.1.

Controlador de coherencia Respuestas	Comentario
RpInV: respuesta a una petición de invalidación Id	El CC que recibe una petición PtObE a un bloque en el estado L, invalida el bloque y responde con RpInv.
RpDc: respuesta con el bloque Id | bloque	El CC que recibe la petición PtObE tiene el bloque en exclusividad. Emite una respuesta con el bloque y la acción de invalidación está implícita en el tipo de respuesta. Esta respuesta también se utiliza si la petición del CM es PtObL. En este caso no se invalida el bloque al responder.

Tabla 8.1 Respuesta de un CC a un CM.

Estados y transiciones

Cuando un CM procesa una petición de exclusividad y hay copias del bloque en varias caches, el CM debe esperar las respuestas de invalidación de los CC correspondientes, antes de responder al CC solicitante. Para identificar la espera de las respuestas de invalidación de los CC se utiliza un estado transitorio en el directorio.

Directorio. Los estados estables de un bloque en el directorio son los mismos que en el Capítulo 7 (Figura 8.5).

El número de estados transitorios en el directorio es tres (ML, MM, LM). Se utilizan para identificar la espera de la respuesta con el bloque o respuestas de invalidación de los CC. Respecto al protocolo del Capítulo 7 se ha añadido el estado LM, cuyo objetivo es esperar las respuestas de invalidación.

Cache. En una cache se utilizan los mismos estados estables que en el protocolo descrito en el Capítulo 7 (Figura 8.5). En cuanto a los estados transitorios, se añade otro estado transitorio para gestionar de forma canónica los cruces de peticiones al expulsar un bloque en el estado M.

Para describir el protocolo se utilizan las dos peticiones del procesador que requieren acceder al directorio (fallo en una instrucción load, store o una instrucción store que accede a un bloque sin permiso de exclusividad) y las dos posibilidades de ubicación del bloque solicitado, en memoria o en una cache. Posteriormente se detallan las transiciones entre estados en una expulsión de un bloque de una cache.

En la descripción se muestran las transiciones entre estados de un bloque en el CC que efectúa la petición (agente procesador), en el CM y en otros CC (agente observador).

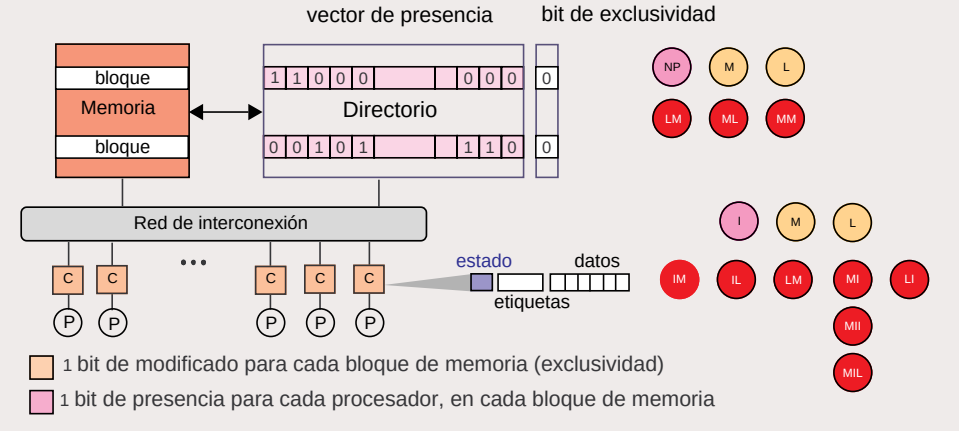


Figura 8.5 Protocolo de directorio MLI y recepción de respuestas de invalidación de los CC en el CM. Estados de un bloque en el directorio y en un contenedor de cache.

Fallo en lectura

Memoria tiene actualizado el bloque. En la parte izquierda de la Figura 8.6 se muestra el flujo de mensajes en un fallo de lectura. El CC emite un mensaje con una petición de lectura de bloque (Pt). El CM accede al direc-

torio para leer el VP y el BE y determina que el estado del bloque es NP o L. El CM emite un mensaje de respuesta (RpD), que incluye el bloque, al CC que ha efectuado la petición.

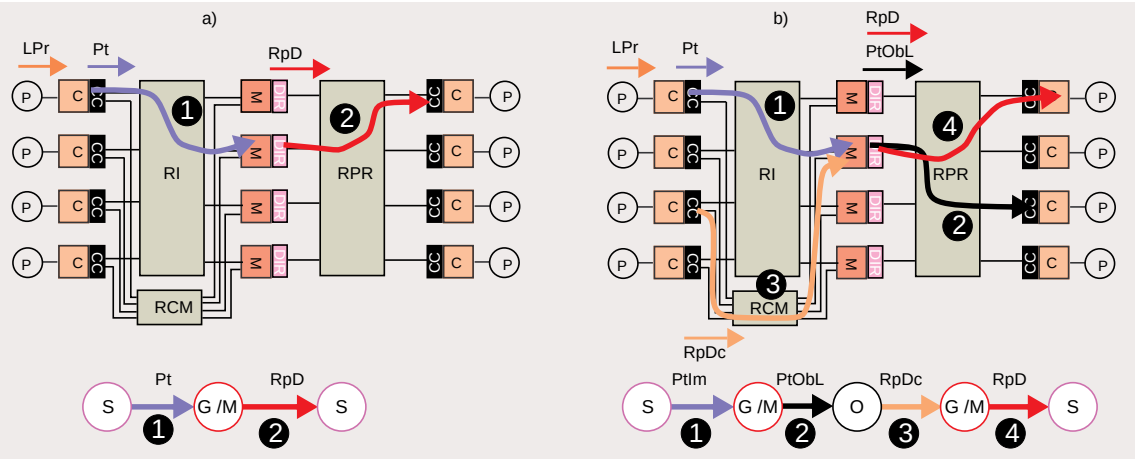


Figura 8.6 Recepción de respuestas de los CC en el CM. Flujo de mensajes en un fallo de lectura: a) memoria está actualizada y b) una cache tiene el bloque en exclusividad.

Una cache tiene el bloque en exclusividad. En la parte derecha de la Figura 8.6 se muestra el flujo de mensajes entre el CC que efectúa la petición y el CM y entre el CM y el CC que tiene el bloque en exclusividad.

El CM emite una petición de observación de lectura (PtObL) al CC que tiene el bloque en exclusividad, para que suministre el bloque. Este CC responde al CM con un mensaje que incluye el bloque (RpDc). El CM actualiza el directorio y la memoria al recibir la respuesta del CC que tiene el bloque en exclusividad. Finalmente el CM responde con el bloque al CC solicitante.

Diagrama de transiciones entre estados. En la Figura 8.7 se muestran las transiciones entre estados en un fallo de lectura en: a) la cache del CC que efectúa la petición (solicitante), b) el directorio (CM) y c) otros CC. El CC envía el mensaje de petición de lectura y establece IL como estado transitorio del bloque. El bloque permanece en este estado hasta que recibe la respuesta del CM (RpD). Entonces, el CC establece como estado estable del bloque el estado L.

El CM al recibir el mensaje de petición de lectura accede al directorio y especulativamente a memoria. Si el estado del bloque en el directorio es NP o L, el CM determina que puede responder a la petición utilizando el bloque leído de memoria. Entonces, el CM, además de enviar la respuesta, actualiza el VP

añadiendo el identificador del CC que ha efectuado la petición. Los otros CC no reciben peticiones de coherencia desde el CM. En la parte izquierda de la Figura 8.8 se muestra un diagrama temporal.

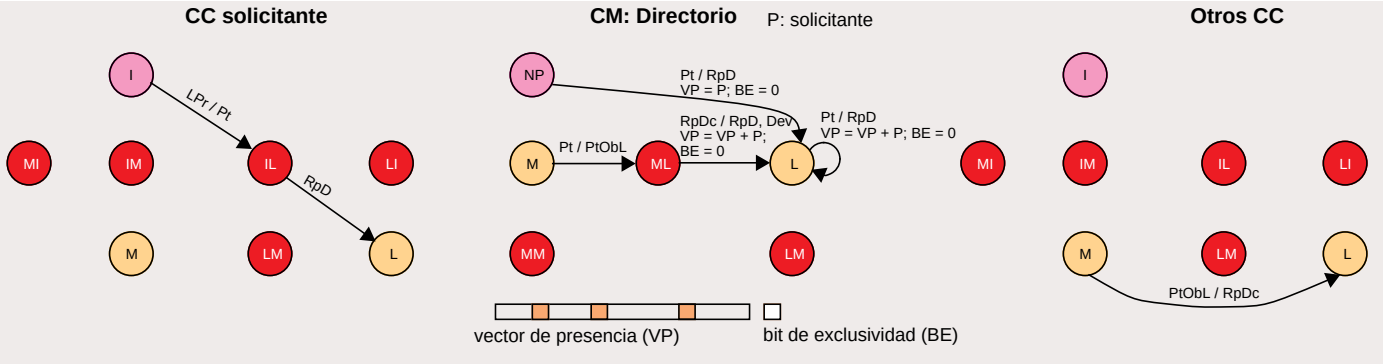


Figura 8.7 Recepción de respuestas de los CC en el CM. Protocolo de directorio MLI. Transiciones entre estados en un fallo de lectura.

Si el estado del bloque en el directorio es M, el CM establece el estado transitorio ML. El CM emite una petición al CC que tiene el bloque en exclusividad (PtObL). Este CC, al recibir la petición, responde al CM con un mensaje, que incluye el bloque (RpDc) y establece como nuevo estado del bloque el estado L. El CM al recibir la respuesta (RpDc) actualiza la memoria con el bloque y modifica el VP y el BE, estableciendo en el directorio como estado estable del bloque el estado L. En la parte derecha de la Figura 8.8 se muestra un diagrama temporal.

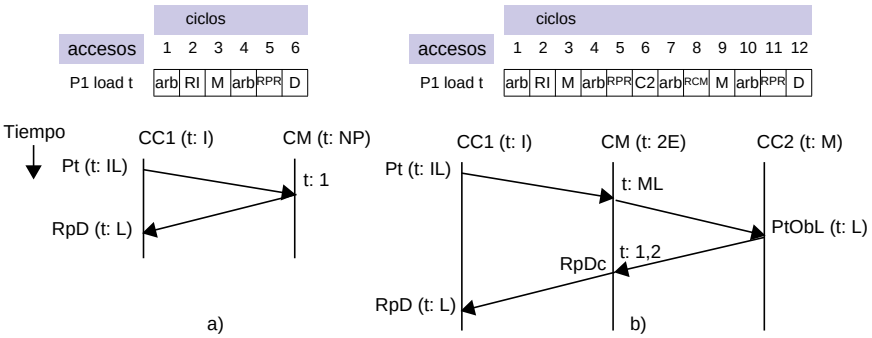


Figura 8.8 Recepción de respuestas de los CC en el CM. Diagrama temporal: a) estado NP o L en el directorio y b) estado M en el directorio.

Fallo de escritura o petición de exclusividad

Memoria tiene el bloque actualizado. En la parte izquierda de la Figura 8.9 se muestra el flujo de mensajes en un fallo de escritura cuando no hay copias del bloque en otras caches. El CC del solicitante emite un mensaje con una petición de bloque con intención de modificación (PtIm). El CM, al recibir el mensaje, lee el estado del bloque en el directorio y accede especulativamente a memoria. El CM determina que el estado del bloque es NP y responde con el bloque al solicitante. Además, actualiza el VP y el BE.

Hay copias del bloque en caches. En la parte derecha de la Figura 8.9 se muestra el flujo de mensajes en un fallo de escritura cuando otras caches tiene copia del bloque. El CM, utilizando la información de estado del bloque en el directorio, determina los CC a los que es necesario enviar una petición de invalidación (PtObE). Una vez el CM ha recolectado las respuestas de los CC (RpInV), el CM responde al CC solicitante con el bloque (RpD). Además, modifica el VP y el BE.

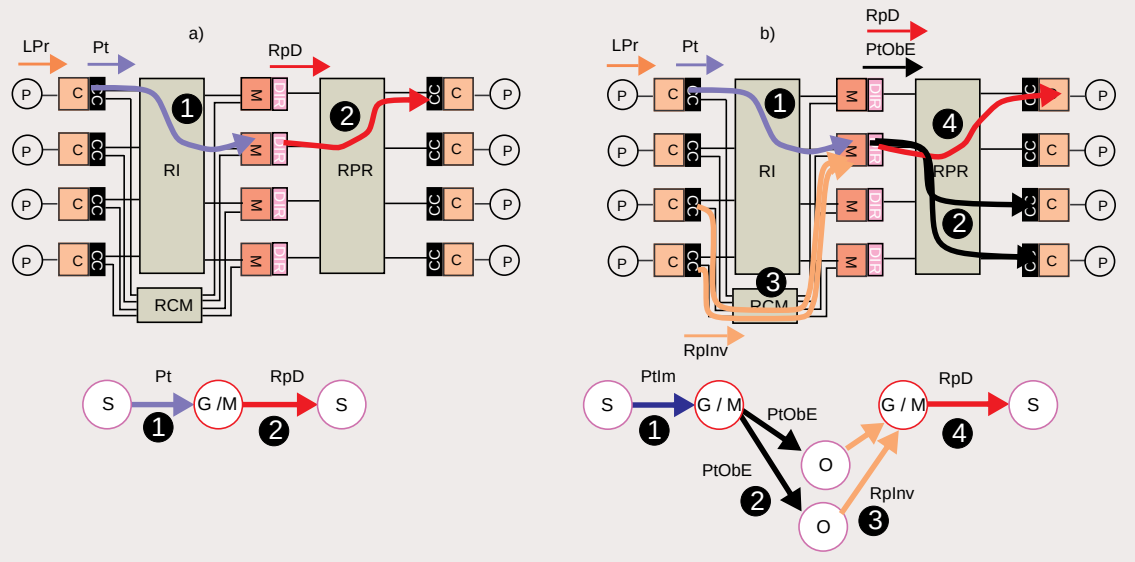


Figura 8.9 Recepción de respuestas de los CC en el CM. Flujo de mensajes en un fallo de escritura: a) no hay copias en caches y b) hay copias en caches.

Una cache tiene el bloque en exclusividad. En la Figura 8.10 se muestra el flujo de mensajes en un fallo de escritura cuando otra cache tiene el bloque en exclusividad. El CM, emite un mensaje PtObE al CC que tiene el bloque en exclusividad. Como el estado del bloque es M, el CC infiere que además de

invalidar el bloque debe suministrarlo (RpDc). El CM, al recibir la respuesta, actualiza el VP para identificar al CC que ahora tiene el bloque en exclusividad y responde al CC solicitante.

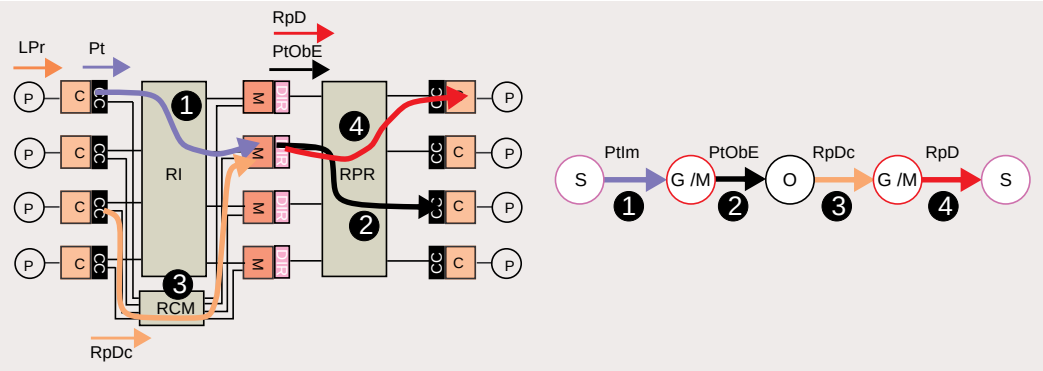


Figura 8.10 Recepción de respuestas de los CC en el CM. Flujo de mensajes en un fallo de escritura y una cache tiene el bloque en exclusividad.

Diagrama de transiciones entre estados. En la Figura 8.11 se muestran las transiciones entre estados en la cache del CC que efectúa la petición, el directorio y otros CC. El CC del solicitante establece un estado transitorio (IM o LM), en función del estado estable inicial, esperando la respuesta.

Cuando no hay copias del bloque o sólo el CC solicitante tiene copia del bloque, el estado del bloque en el directorio pasa de NP a M o de L a M. El VP se actualiza en consecuencia y el BE se activa. Además, el CM emite un mensaje de respuesta al solicitante (RpD). En la parte izquierda de la Figura 8.12 se muestra un diagrama temporal.

Cuando un CC emite una petición PtIm, estando el bloque en estado L, el bit de presencia del CC está activado en el VP. Por tanto, hay que excluirlo de la lista de CC que reciben una petición de observación de escritura PtObE (VP - P). El CM inicializa el contador de respuestas (|VP - P|) y emite las peticiones, para invalidar el bloque, a los CC involucrados en la acción de coherencia. Estos CC al recibir la petición PtObE invalidan la copia del bloque y responden al CM (RpInv). El CM recolecta todas las respuestas de invalidación antes de responder al CC solicitante (RpD). Este CC, al recibir la respuesta siempre almacena el bloque recibido, en el contenedor correspondiente, tanto si el estado estable inicial es L como I. En el centro de la Figura 8.12 se muestra un diagrama temporal. Suponemos que el CM genera todos los mensajes en paralelo. Estos mensajes, cuando no hay conflictos en la RPR para acceder a

los destinatarios, se transmiten en paralelo. Cuando varios mensajes llegan en el mismo instante a un destinatario se encolan. Un destinatario sólo procesa un mensaje en un instante determinado.

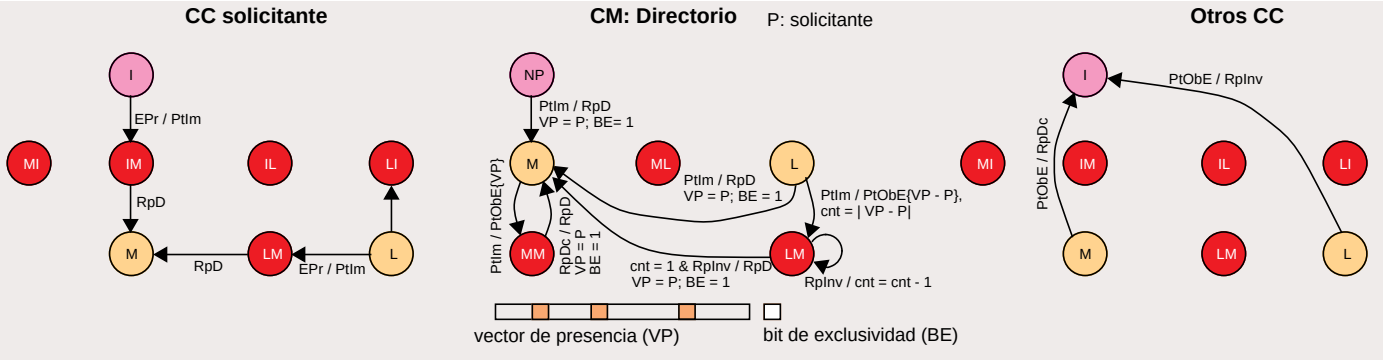


Figura 8.11 Recepción de respuestas de los CC en el CM. Transiciones entre estados en un fallo de escritura.

Cuando el estado del bloque en el directorio es M, el CM emite una petición PtObE al CC que tiene el bloque en exclusividad. Este CC responde con el bloque y establece como estado del bloque en cache el estado I[5]. El CM al recibir la respuesta del CC, que tenía el bloque en exclusividad, responde al CC solicitante. Además, actualiza el VP con el identificador del CC solicitante. En la parte derecha de la Figura 8.12 se muestra un diagrama temporal.

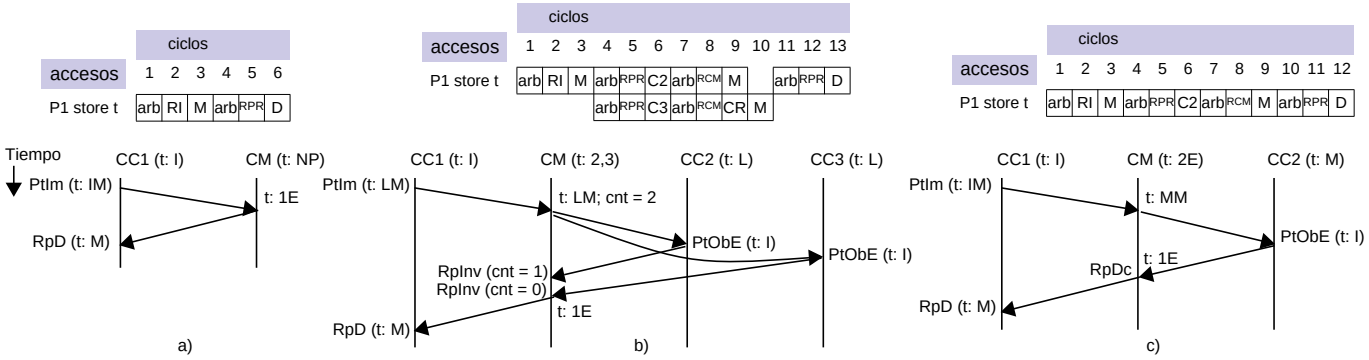


Figura 8.12 Recepción de respuestas de los CC en el CM. Diagrama temporal simplificado: a) estado NP en el directorio o L y VP = P, b) estado L en el directorio y c) estado M en el directorio. El acrónimo CR indica cola de respuestas en un CM.

5. Un CC al recibir una petición PtObE del CM determina, en función del estado del bloque en la cache, si es necesario una acción de suministro del bloque.

Expulsión

El directorio es preciso y una petición de expulsión de un CC espera una respuesta del CM. En la Figura 8.13 se muestra el flujo de mensajes cuando se expulsa un bloque de cache.

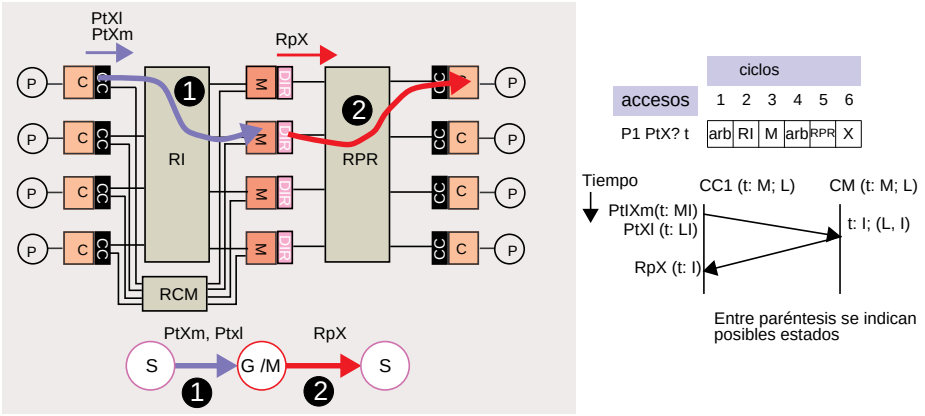


Figura 8.13 Recepción de respuestas de los CC en el CM. Expulsión de un bloque de cache.

El CC efectúa una petición de expulsión (PtXm, PtXI), que va acompañada del bloque si el bloque está en el estado M (PtXm). El CM en cualquier caso actualizada la entrada correspondiente en el directorio. Además, el CM actualiza la memoria cuando la petición es PtXm. El CM responde al CC confirmando que se ha realizado la acción solicitada.

En la Figura 8.50 se muestran las transiciones entre estados en: a) la cache del CC que expulsa el bloque, b) el directorio y c) otros CC. En otros CC no se producen transiciones entre estados.

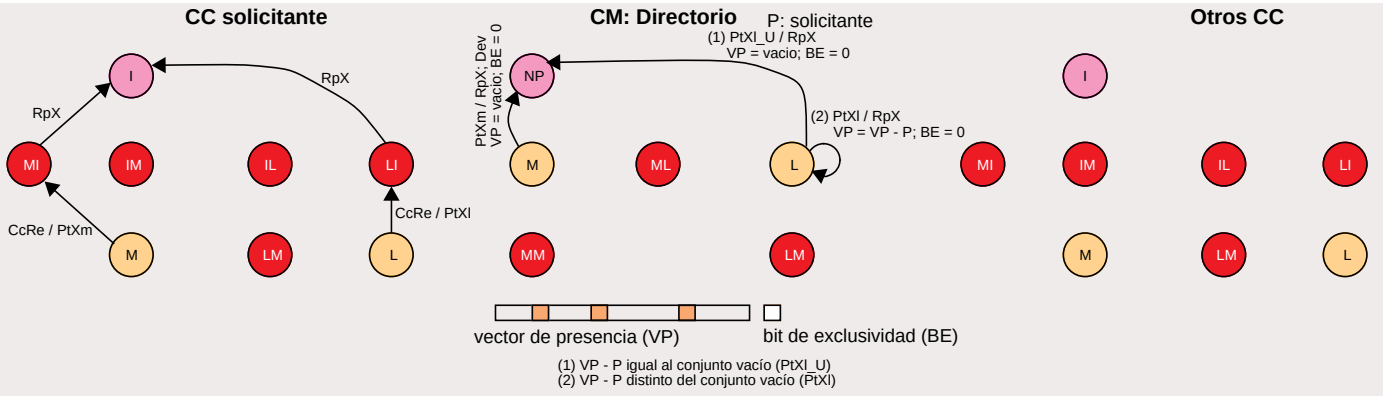


Figura 8.14 Recepción de respuestas de los CC en el CM. Transiciones entre estados cuando se expulsa un bloque.

Para distinguir entre la expulsión de un bloque en el estado M o en el estado L se utilizan distintos tipos de peticiones y también estados transitorios distintos; MI y LI respectivamente. Cuando el CM procesa la petición actualiza memoria si la petición es PtXm y en cualquiera de los dos casos se actualiza la entrada correspondiente del bloque en el directorio (VP). Finalmente el CM envia una respuesta de confirmación. El CC al recibir la respuesta cambia el estado transitorio del bloque al estado I.

En el diagrama del CM (Figura 8.14) hay que distinguir dos casos al procesar la petición PtXI de un CC (PtXI, PtXI_U). El estado final depende de si el CC, que efectúa la petición, es el único que está en el vector de presencia.

Diagrama completo de estados y transiciones

En las Figura 8.15 se muestran los diagramas de transiciones entre estados de un bloque en una cache y en un directorio. En un CC se distinguen, en diagramas separados, las transiciones iniciadas por el agente procesador (CC solicitante) y el agente observador (otros CC). En el diagrama etiquetado como CC solicitante están incluidas las peticiones del procesador que no requieren iniciar transacciones explícitas de coherencia.

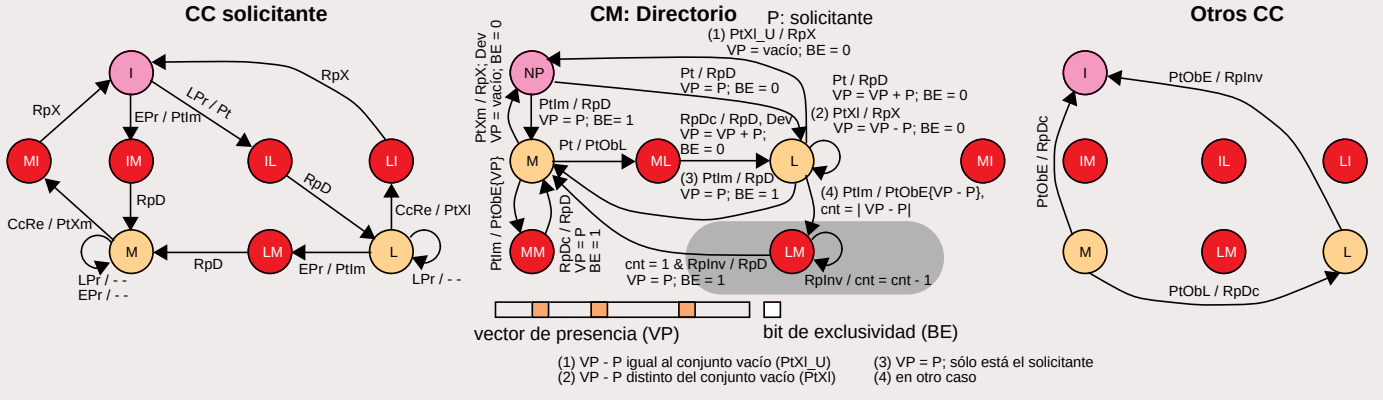


Figura 8.15 Recepción de respuestas de los CC en el CM. Estados y transiciones entre estados de un bloque en una cache y en el directorio.El sombreado muestra la diferencia básica con la especificación del Capítulo 7.

Ejemplo

Los dos trozos de código que se utilizan son el esqueleto del algoritmo de Dekker para exclusión mútua (parte izquierda de la Figura 8.16). Las variables aviso1 y aviso2 están contenidas en bloques que se ubican en los módulos de memoria M1 y M2 respectivamente. Antes de iniciarse la ejecución, la cache

C1 tiene copia del bloque que contiene la variable aviso2 y la cache C2 tiene copia del bloque que contiene la variable aviso1. El valor de las dos variables es cero.

En la Figura 8.16 se muestra un diagrama temporal del ejemplo utilizando el protocolo de este apartado. Suponemos que la petición del CM2 al CC1 experimenta un retardo significativo en la red (varios ciclos RPR).

Las instrucciones store de los procesadores P1 y P2 son fallos en cache. Las peticiones que emiten CC1 y CC2 acceden a CM distintos. Por tanto progresan de forma paralela hasta el CM correspondiente. El CM1 emite una petición PtObE al CC2 y espera la respuesta. En paralelo el CM2 emite una petición PtObE al CC1 y espera la respuesta.

Cuando el CM1 recibe la respuesta de CC2 (RpInv) emite una respuesta al CC1. Seguidamente el procesador P1 ejecuta la instrucción load y accede a la región crítica.

El procesado de la petición de CM2 al CC1 se retrasa por conflictos en la red. Cuando el CM2 recibe la respuesta del CC1 (RpInv) responde al CC2. El procesador P2 ejecuta la instrucción load y detecta un fallo.

P1	P2
store aviso1	store aviso2
load aviso2	load aviso1

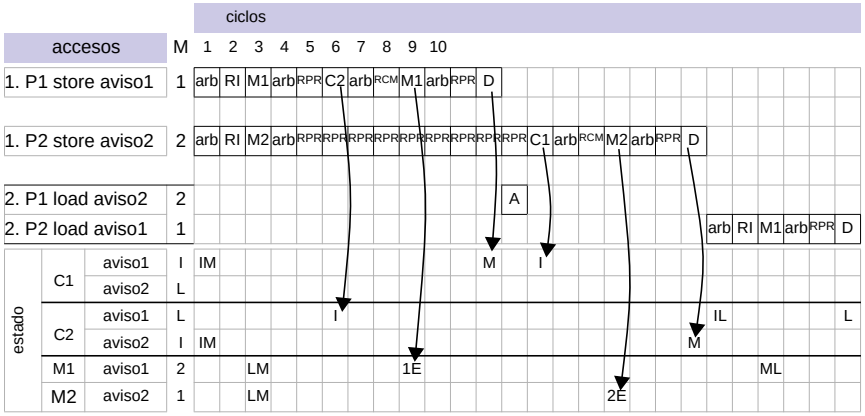


Figura 8.16 Protocolo de directorio MLI. Recepción de respuestas de los CC en el CM. Esqueleto del algoritmo de Dekker.

En la Figura 8.17 se muestra, utilizando el ejemplo previo, una comparación del flujo de mensajes del protocolos MLI del Capítulo 7 y el protocolo de este apartado. En el Capítulo 7 las peticiones de invalidación de un CM no son respondidas por los CC que las reciben. El algoritmo de arbitraje de la RV garantiza un orden lógico global de los mensajes de los CM (Figura 8.17). En el protocolo descrito en este apartado un CC responde a una petición de invalida-

ción del CM. El algoritmo de arbitraje de la red RPR no garantiza un orden lógico global de los mensajes emitidos desde los CM. Sólo garantiza un orden entre un emisor y un receptor.

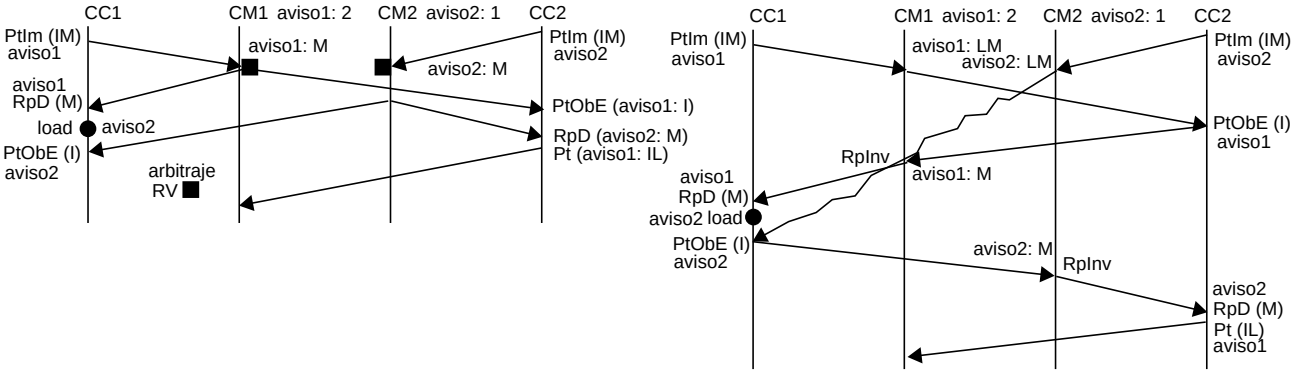


Figura 8.17 Protocolo de directorio MLI. Comparación entre el diseño del Capítulo 7 y el diseño de este apartado (recepción de respuestas de los CC en el CM). Esqueleto del algoritmo de Dekker.

Controlador de memoria. Gestión de cruces de peticiones

Ventana de vulnerabilidad. Mientras el CM está esperando una respuesta puede analizar una petición al mismo bloque (estados transitorios MM, ML y LM).

El procesado en el CM de peticiones a bloques en un estado transitorio es el mismo que en el Capítulo 7. El CM bloquea el análisis de la CP hasta que el CM recibe la respuesta o respuestas necesarias para efectuar la transición a un estado estable (Figura 8.18).

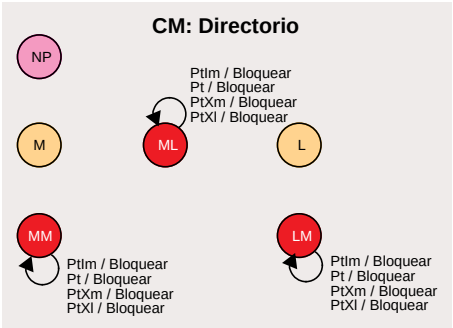


Figura 8.18 Recepción de respuestas de los CC en el CM. Bloqueo del procesado de peticiones en los estados transitorios del directorio.

El CM no procesa peticiones de los CC a un bloque en un estado transitorio. Entonces, el resto de cruces entre peticiones del CM y un CC se observan (infieren) en estados estables. Suponemos que la RI es de tipo crossbar y se utiliza un arbitraje por antigüedad[6]. En estas condiciones, los cruces de peticiones son los mismos que los mostrados en el Capítulo 7. En particular, en el estado estable NP no se infieren cruces de peticiones. En la Figura 8.19 se muestran las transiciones entre estados en las cuales se infieren cruces de peticiones, teniendo en cuenta los estados del protocolo de este apartado.

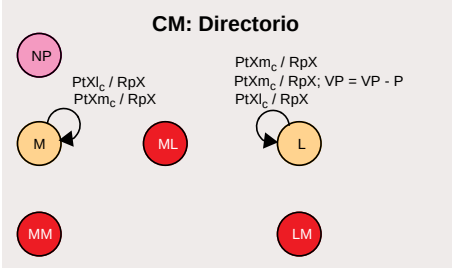


Figura 8.19 Recepción de respuestas de los CC en el CM. Inferencia de cruces en el directorio.

Controlador de coherencia. Gestión de cruces de peticiones

Funcionalmente el conjunto de cruces son los mostrados en el Capítulo 7. La diferencia es que un CC debe responder a todas las peticiones del CM. En la Figura 8.20 se muestran estos cruces de peticiones. Recordemos que la RPR se utiliza para transmitir peticiones y respuestas de los CM a los CC y se mantiene el orden de los mensajes entre un emisor y un receptor.

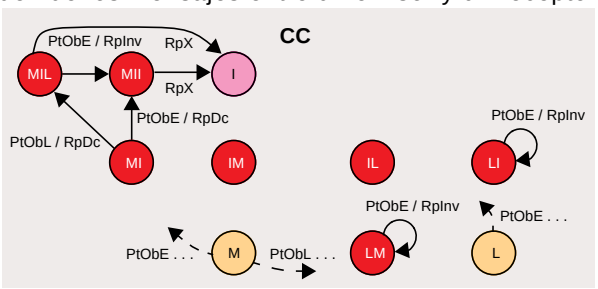


Figura 8.20 Recepción de peticiones de un CM en un CC . Inferencia de cruces en un CC. También se muestran peticiones del CM en estados estables.

6. En el próximo protocolo del capítulo se analiza el caso de utilizar una red de tipo malla.

En el estado LI un CC responde (RpInv) a una petición de observación de escritura (PtObE) y se espera la respuesta de expulsión del bloque (RpX). En el estado LM un CC responde (RpInv) y espera la respuesta del CM con el bloque (RpD).

En el estado MI la actuación depende del tipo de petición que se recibe del CM. En cualquier caso se espera la respuesta del CM a la petición de expulsión. Cuando se recibe una petición PtObE se suministra el bloque al CM (RpDc) y se espera la respuesta del CM en el estado MII. Si la petición es PtObL, posteriormente se puede recibir una petición PtObE. Entonces se utiliza un estado transitorio para identificar el hecho. Este estado se denomina MIL. Si, antes de recibir la respuesta del CM (RpX), se recibe una petición PtObE del CM, el CC responde y establece MII como estado transitorio del bloque[7].

PETICIONES Y RESPUESTAS DEL CM SE ENCAMINAN POR REDES DISTINTAS

El protocolo que se describe básicamente es el mismo que el descrito en el apartado previo. La diferencia reside en que un CC, en un cruce de peticiones, debe inferir si la petición del CM debe procesarse: a) suponiendo el estado estable inicial de la transacción pendiente en el CC o b) hay que esperar a que finalice la transacción pendiente en el CC (procesado en el estado estable final). Esta diferencia es debida a la organización del multiprocesador. En concreto, las redes de interconexión utilizadas.

En la Figura 8.21 se muestra la diferencia básica, en la descripción funcional de la secuencia de mensajes en una transacción, entre el apartado previo y el actual. En la parte izquierda se muestra el apartado previo y en la parte derecha el apartado actual. Para efectuar la inferencia en un CC, cuando se produce un cruce de peticiones, se añade una nueva petición (PtObInv).

7. El diagrama de transiciones entre estados es funcionalmente equivalente al del Capítulo 7, pero el presentado es este apartado es más ortodoxo o canónico.

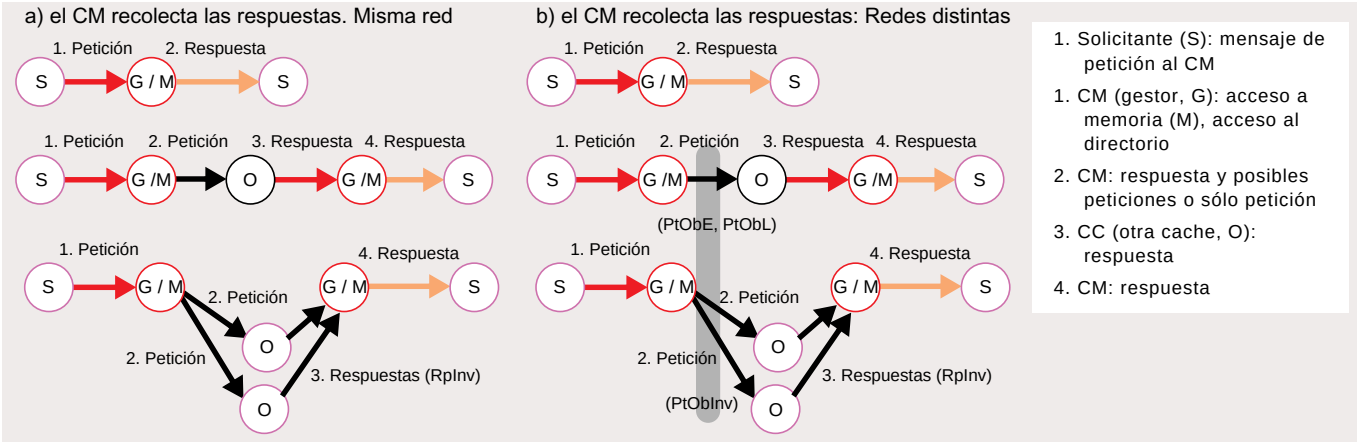


Figura 8.21 Flujos de mensajes en dos protocolo de directorio MLI donde el CM recolecta las respuestas: a) misma red para peticiones y respuestas de los CM a los CC (apartado previo) y b) redes distintas para peticiones y respuestas de los CM a los CC. El sombreado muestra la diferencia básica.

Organización del multiprocesador

El protocolo utiliza cuatro redes lógicas (Figura 8.22): RI, RCM, RP y RMC.

Los CC utilizan la RI para emitir peticiones y la RCM para emitir respuestas al CM. Los CM utilizan la RP para emitir peticiones y la RMC para emitir respuestas a los CC[8].

En esta organización un CC no observa el orden de emisión de peticiones y respuestas desde un CM al CC. Esto es debido a que se utilizan redes distintas para transmitir las peticiones y respuestas de los CM a los CC.

La cola de mensajes de petición en un CC se denomina CPC y la cola de mensajes de respuesta se denomina CRC. Para representar retardo en una red se replica el acrónimo de la red en ciclos consecutivos.

8. Las redes pueden ser de tipo crossbar, anillo o malla. El encaminamiento entre un emisor y un receptor debe estar preestablecido.

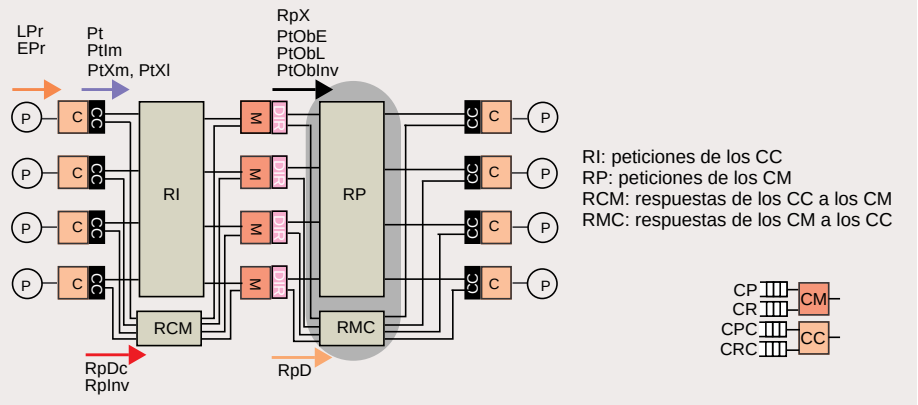


Figura 8.22 Redes distintas para peticiones y respuestas del CM. Organización del multiprocesador. Redes lógicas y mensajes. En sombreado se muestra la diferencia básica con el apartado previo.

Mensajes del protocolo

En la Figura 8.22 se muestra un esquema genérico de la transmisión de mensajes con peticiones y respuestas entre los CC y el CM y viceversa.

En la descripción del protocolo se distinguen los siguientes tipos de mensajes:

- Peticiones de un CC a un CM (Pt, PtIm, PtXm, PtXl)
- Peticiones de un CM a los CC (PtObL, PtObE, PtObInv)
- Respuestas de un CM a un CC (RpD, RpX)
- Respuestas de un CC a un CM (RpDc, RpInv)

Los mensajes que se han añadido o modificado respecto al protocolo del Capítulo 7 se muestran en la Tabla 8.2. Posteriormente se describe la necesidad de la petición PtObInv.

Controlador de memoria	Controlador de coherencia	Comentario
Mensajes de petición a CC y acciones	**Respuestas**	
PtObInv: petición de invalidación op | dirección | Id	RpInV: respuesta a una petición de invalidación Id |	El CC que recibe una petición PtObInv en el estado L invalida el bloque y responde con RpInv.
PtObE: petición de observación de escritura op | dirección | Id	RpDc: respuesta con el bloque Id | bloque	El CC que recibe la petición PtObE tiene el bloque en exclusividad. Emite una respuesta con el bloque y la acción de invalidación está implícita en el tipo de respuesta. Esta respuesta también se utiliza si la petición del CM es PtObL. En este caso no se invalida el bloque al responder.

Tabla 8.2 Petición del CM a los CC y respuesta de estos al CM.

Estados y transiciones

Los estados en el directorio y en la cache son los mismos que en el protocolo previo de este capítulo (Figura 8.5).

La descripción funcional de la secuencia de mensajes también es la misma que en el protocolo previo de este capítulo (Figura 8.6, Figura 8.9, Figura 8.10, Figura 8.13).

La diferencia básica, con el protocolo previo de este capítulo, es que los mensajes de petición y los mensajes de respuesta de los CM a los CC se transmiten por redes distintas. Entonces, en un CC no se observa un orden entre peticiones y respuestas emitidas desde un mismo CM. Cuando sea necesario conocer este orden hay que inferirlo en el CC, a partir del estado del bloque en la cache y la petición del CM. Para ello se introduce la petición PtObInv, la cual utilizan los CM para solicitar la invalidación de un bloque en una cache[9].

Diagrama completo de estados y transiciones

En las Figura 8.23 se muestran los diagramas de transiciones entre estados de un bloque en cache y en el directorio. En el CC se distinguen, en diagramas separados, las transiciones iniciadas por el agente procesador (CC solicitante) y el agente observador (otros CC). En el diagrama etiquetado como CC solicitante están incluidas las peticiones del procesador que no requieren iniciar transacciones explícitas de coherencia. Observe que, en el CM, la petición al efectuar la transición del estado L al estado LM es PtObInv en lugar de PtObE (Figura 8.15). Por tanto, la petición que recibe un CC para efectuar la transición del estado L al estado I es PtObInv.

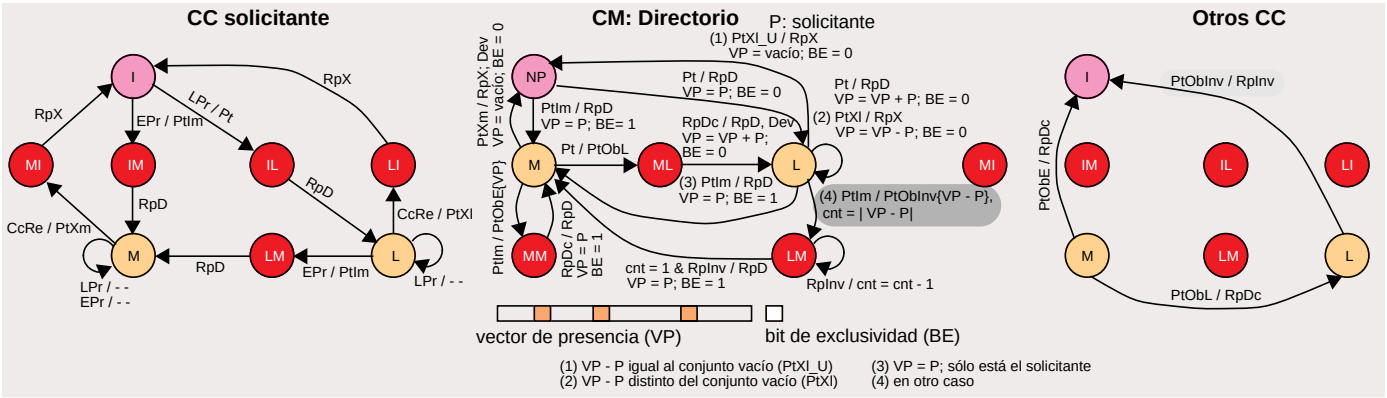


Figura 8.23 Redes distintas para peticiones y respuestas del CM. Estados y transiciones entre estados de un bloque en una cache. En sombreado se muestra la diferencia básica con el apartado previo.

9. Recordemos que en el protocolo descrito previamente en este capítulo se utiliza la petición PtObE.

En el diagrama del CM (Figura 8.23) hay que distinguir dos casos al procesar la petición PtXl de un CC (PtXl, PtXl_U). El estado final depende de si el CC, que efectúa la petición, es el único que está en el vector de presencia.

Inferencia en un CC del orden entre peticiones y respuestas emitidas desde un CM

Un CC tiene que inferir el orden en que un CM ha procesado una petición de este CC respecto de una petición de otro CC al mismo bloque.

En un directorio sólo se identifican estados estables de los bloques en los CC. En consecuencia, la emisión, por parte de un CM, de una petición a un CC, al procesar el CM una petición de otro CC, está determinada por el estado estable del bloque en el CC del cual tiene constancia el CM.

Misma red para transmitir mensajes de petición y mensajes de respuesta desde un CM a un CC. Cuando se utiliza la misma red para transmitir peticiones y respuestas de los CM a los CC, y la red mantiene el orden entre emisor y receptor, una petición del CM a un CC está ordenada respecto de la respuesta previa del CM a este CC. Esta respuesta del CM ha determinado un estado estable en el CC y este estado es del cual tiene constancia el CM. El CM procesa la petición teniendo en cuenta este estado.

Esta propiedad es la que se ha utilizado en el protocolo del apartado previo para inferir que las peticiones de un CM, que se reciben en un CC, estando el bloque en un estado transitorio, han sido ordenadas por el CM antes que la petición pendiente del CC (Figura 8.24). Por tanto, el CC responde al CM como si el bloque estuviera en el estado estable del que proviene[10].

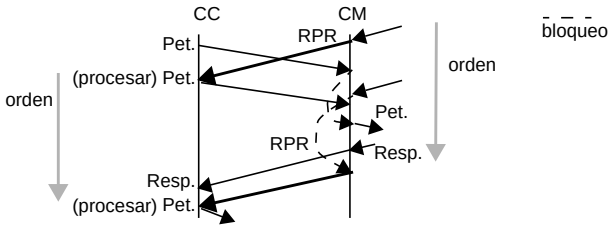


Figura 8.24 Cruce de peticiones en un CC. Comunicación entre un CM y los CC: una red para transmitir mensajes del CM.

En resumen, un CC puede iniciar una nueva transacción cuando un bloque está en un estado estable. Entonces, el CC establece un estado transitorio para el bloque. Si este CC recibe una petición del CM, esta petición ha sido

10. Si en un CC se puede recibir una secuencia de peticiones del CM hay que tener en cuenta las respuestas previas. En el CC hay que recordar que el CM ha ordenado previamente peticiones de otros CC.

emitida por el CM teniendo en cuenta el estado estable del que se proviene para llegar al estado transitorio. Por tanto, el CC tiene que procesar la petición del CM.

Redes distintas para transmitir mensajes de petición y mensajes de respuesta de un CM a un CC. En un CC no existe orden en la recepción de mensajes provenientes de redes distintas, aunque exista orden en la recepción de mensajes provenientes de una misma red. En estas condiciones puede ser factible que (Figura 8.25)

- Una respuesta del CM adelante a una petición previa del CM al mismo CC.
- Una petición del CM adelante a una respuesta previa del CM al mismo CC.

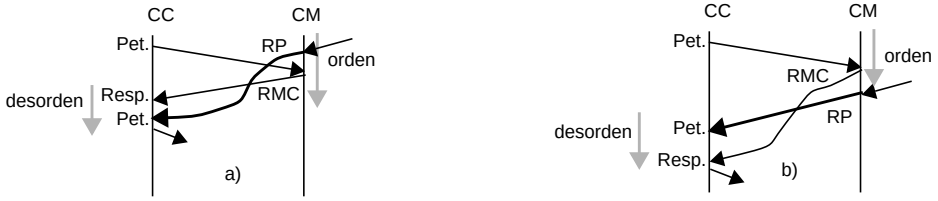


Figura 8.25 Cruce de peticiones en un CC: a) una respuesta adelanta a una petición previa y b) una petición adelanta a una respuesta previa.

Una respuesta del CM no puede adelantar a una petición previa del CM al mismo bloque

En este protocolo no es posible que una respuesta adelante a una petición previa del CM al mismo bloque. Un CM no procesa peticiones que acceden a un bloque en un estado transitorio en el directorio. Además, un CC siempre responde al CM una petición del mismo. Entonces, en la situación a) de la Figura 8.25 el CM procesa, como muy pronto, la petición del CC después de recibir la respuesta a su petición (Figura 8.26).

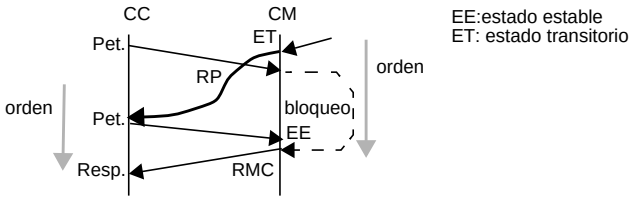


Figura 8.26 Una respuesta del CM no puede adelantar a una petición previa del CM al mismo bloque.

Una petición del CM adelanta a una respuesta previa del CM al mismo bloque

La situación b) de la Figura 8.25 es factible en este protocolo. El CM procesa una petición de otro CC en un estado estable que induce una petición a un CC. El CM ha emitido previamente una respuesta al mismo CC que accede al mismo bloque. Como los mensajes se transmiten por redes distintas el retardo que experimentan puede ser distinto. En estas condiciones, la petición del CM puede llegar antes que la respuesta al CC.

Entonces, un CC al procesar una petición del CM, es un estado transitorio, tiene que determinar si el CM ha emitido la petición teniendo en cuenta el estado inicial o final de la transacción pendiente en el CC. Esto es, la petición del CM ha sido emitida antes que la respuesta o al contrario (Figura 8.27).

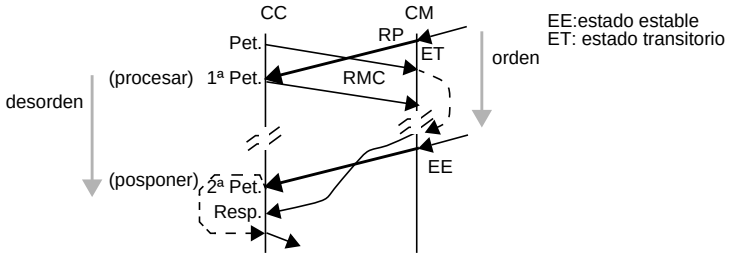


Figura 8.27 Cruce de peticiones en un CC. Comunicación entre los CM y los CC: una red para transmitir peticiones del CM y otra red para transmitir respuestas del CM

Ejemplo. Un bloque en el estado L puede recibir una petición del CM para que se invalide el bloque. Un bloque en el estado M puede recibir una petición para que suministre el bloque y el próximo estado del bloque sea inválido. Cuando el CC está en el estado L y emite una petición para obtener la exclusividad, el próximo estado es el estado transitorio LM.

En el protocolo que se describe, un CM utiliza redes distintas para transmitir las peticiones y las respuesta a un CC. Entonces, un CC puede recibir una petición del CM, para un bloque en el estado LM, que ha sido inducida por una petición de un segundo CC, que el CM ha procesado antes que la transacción pendiente del CC (1ª petición en la Figura 8.27). También, la petición del CM puede haber sido inducida por una petición de un tercer CC, que el CM ha procesado después de la transacción pendiente del CC (2ª petición en la Figura 8.27). Ahora bien, al recibir el CC esta petición del CM, la transacción del CC sigue pendiente, debido a que la respuesta se recibe por otra red (RMC). Por tanto, el procesado en el CC de la petición del CM debe posponerse.

En consecuencia, para distinguir cuando se procesa una petición del CM en un CC, son necesarias peticiones distintas desde el CM[11]. Por ello, se ha añadido la petición PtObInv en el protocolo.

Bloque en el estado L en el directorio. Cuando un CC solicita la exclusividad del bloque, el CM utiliza la petición PtObInv para invalidar las copias del bloque en otras cache (parte izquierda de la Figura 8.28).

Bloque en el estado M en el directorio. Cuando un CC solicita la exclusividad, el CM utiliza la petición PtObE para solicitar el suministro del bloque y la invalidación del bloque a una cache (parte derecha de la Figura 8.28).

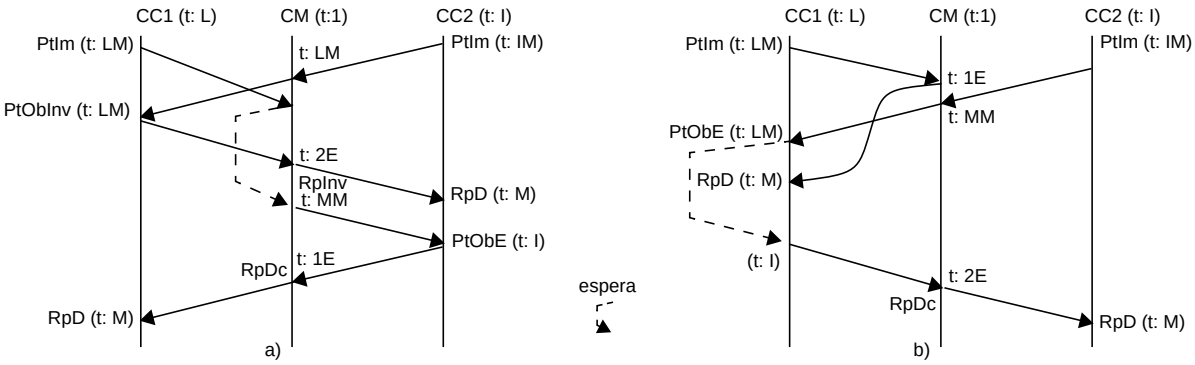


Figura 8.28 Identificación de cruces de peticiones en el CC1: a) El CM ordena la petición de otro CC antes, b) el CM ordena la petición de otro CC después.

En estas condiciones, un CC infiere que una petición PtObInv del CM debe procesarse en el estado transitorio LM y el procesado de una petición PtObE debe posponerse hasta que llegue la respuesta de la petición pendiente (Figura 8.29).

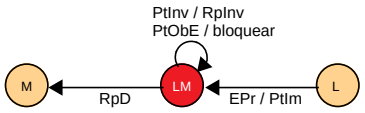


Figura 8.29 En un estado transitorio de un bloque, identificación en un CC del orden de procesado, respecto a la petición propia, de una petición de otro CC en el CM, la cual ha inducido una petición del CM al CC.

11. Recordemos que, en todos los protocolos MLI descritos hasta ahora, la petición PtObE se interpreta en función del estado del bloque en la cache. Si el estado de bloque es L se interpreta como invalidación. Si el estado del bloque es M se interpreta como suministro e invalidación.

Identificación de posibles cruces de peticiones

Funcionalmente los posibles cruces de peticiones, en los cuales el CC infiere que la petición del CM debe procesarse, teniendo en cuenta el estado inicial de la transacción pendiente en el CC, han sido descritos en el Capítulo 7.

Desorden en la recepción de peticiones y respuestas del CM

Los estados transitorios de un bloque, en los cuales un CC está esperando recibir una respuesta de un CM, que incluye el bloque son: IL, IM y LM. Los estados del bloque en la cache al finalizar la transacción pendiente son L, M y M respectivamente.

Para que se produzca un cruce, el CM debe procesar una petición que modifique el estado del bloque en la cache.

Bloque en el estado L. El estado transitorio del que proviene es IL. La petición del CC1 al CM ha sido Pt. El estado del bloque en el directorio es L (Figura 8.30).

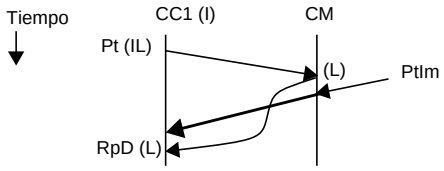


Figura 8.30 Bloque en estado L en la cache C1. Peticiones de otros CC.

El CM puede procesar una petición PtIm de otro CC, la cual induce que el CM emita una petición PtObInv al CC1.

Bloque en el estado M. El estado transitorio del que proviene es IM o LM. La petición del CC1 al CM ha sido PtIm. El estado del bloque en el directorio es M (Figura 8.31).

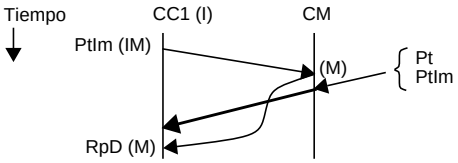


Figura 8.31 Bloque en estado M en la cache C1. Peticiones de otros CC.

El CM puede procesar una petición Pt o PtIm de otro CC, las cuales inducen respectivamente que el CM emita una petición PtObL o PtObE al CC1.

En la Figura 8.32 se muestra el orden de procesado en el CM de varias secuencia de accesos.

Orden de procesado en el CM		
I	J	K
P1 load t	P1 store t	P1 store t
P2 store t	P2 load t	P2 store t

Figura 8.32 Bloque en estado L o M en la cache C1 al finalizar la primera transacción. Orden de procesado en el CM de una secuencia de accesos. La petición del CM, inducida por la petición del CC2, llega al CC1 antes que la respuesta.

En la RI no hay arbitraje por antigüedad: Identificación de cruces de peticiones

En el Capítulo 7 se han analizado los cruces teniendo en cuenta que el arbitraje en la RI mantiene el orden de antigüedad de las peticiones de los CC. Además, el CM extrae de la CP las peticiones en el orden de llegada.

En el protocolo descrito en este apartado no se cumple esta propiedad. Por ejemplo, la latencia efectiva del camino entre un CC y un CM puede ser significativamente distinta, en función del CC y del CM que se comunican. Por otro lado, en una red en malla puede existir congestión en algunos enlaces de comunicación y no existe un arbitraje centralizado.

En este contexto, hay que tener en cuenta la posibilidad de nuevos cruces de peticiones.

El CM puede procesar la petición de exclusividad de un CC2 y la petición de expulsión del bloque por este CC2[12], antes que la petición de expulsión de un CC1, que está identificado en el VP, al procesar la petición de exclusividad del CC2. El CC1 ha emitido la petición al CM antes de recibir la petición del CM, la cual ha sido inducida por la petición de exclusividad del CC2.

Partiendo de los ordenes de procesado en el CM de las secuencias de acceso del Capítulo 7 (Figura 7.30 y Figura 7.32) se pueden construir los ordenes de secuencias de acceso de la Figura 8.33[13].

12. Recordemos que un CC sólo tiene en curso de procesado una petición a un mismo bloque. Por ejemplo, en el orden AX de la Figura 8.33, el procesador P2 no emite la petición de expulsión antes de que haya consolidado la instrución previa: a) llegue el dato o b) recibido la confirmación de que ha consolidado la instrucción store. Si el algoritmo de reemplazo selecciona un contenedor, que aún no contiene el bloque, se suspende la interpretación de la instrucción que determina el reemplazo.

13. Podemos suponer que la expulsión es voluntaria o forzada por un fallo. En este último caso, el bloque al que se accede está almacenado en otro módulo de memoria. Por esta razon no se muestra en el entrelazado. Recordemos que la petición para servir el fallo se puede emitir antes o después de la petición de expulsión.

Orden de procesado en el CM											
AX	BX		CX	DX		EX	FX	GX		HX	
P2 store t	P2 store t	P2 store t	P2 store t	P2 store t	P2 store t	P2 store t	P2 load t	P2 store t	P2 store t	P3 load t	P3 load t
P2 PtXm t	P3 load t	P3 load t	P2 PtXm t	P3 load t	P3 load t	P2 PtXm t	P2 PtXl t	P3 load t	P3 load t	P2 store t	P2 store t
P1 PtXl t	P2 PtXm t	P2 PtXl t	P1 store t	P2 Ptxm t	P2 PtXl t	P1 PtXm t	P1 PtXm t	P2 PtXm t	P2 PtXl t	P3 PtXl t	P3 PtXl t
	P3 PtXl t	P3 PtXl t		P3 PtXl t	P3 PtXl t			P3 PtXl t	P3 PtXl t	P2 PtXm t	P2 PtXl t
	P1 PtXl t	P1 PtXl t		P1 store t	P1 store t			P1 PtXm t	P1 PtXm t	P1 PtXm t	P1 PtXm t

Figura 8.33 Orden de procesado en el CM de secuencias de accesos a memoria. Una secuencia Zx ha sido construida a partir de la secuencia Z de la Figura 7.30 y de la Figura 7.32.

Cruces de peticiones en un CM y en un CC

Controlador de memoria

En un CM, los estados con ventana de vulnerabilidad y los cruces de peticiones en estados estables L y M son los mismos que en el protocolo previo de este capítulo. Las acciones y respuestas del CM también son las mismas (Figura 8.35).

Ahora bien, debido a que el árbitraje en la RI no mantiene la antigüedad de las peticiones, se pueden inferir cruces en el estado NP.

Dado el orden AX de secuencia de accesos, en el CM se infiere un cruce de peticiones en el estado NP (Figura 8.34). Dados los dos ordenes de secuencias de accesos etiquetados como BX, el cruce de peticiones, al procesar la petición de CC1, se infiere en el mismo estado.

En los ordenes CX y los dos DX no se infiere ningún cruce en el CM.

Dado el orden EX de secuencia de accesos, en el CM se infiere un cruce de peticiones en el estado NP. Dados los dos ordenes de secuencias de accesos etiquetados como GX, el cruce de peticiones, al procesar la petición de CC1, se infiere en el mismo estado

Dado el orden FX de secuencia de accesos, en el CM se infiere un cruce de peticiones en el estado L. El CC1 está en el VP, pero el tipo de petición no es esperado.

Dados los dos ordenes de secuencia de accesos HX, en el CM se infiere un cruce de peticiones, al procesar la petición de CC1, en el estado NP.

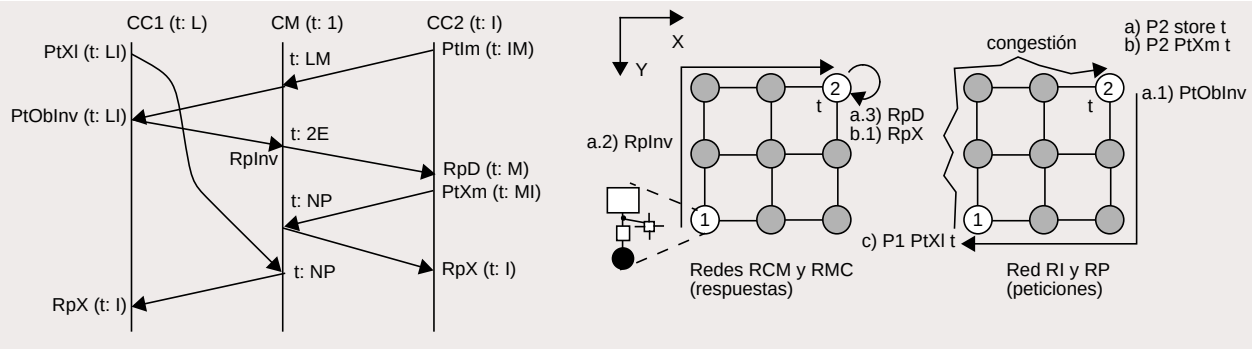


Figura 8.34 Ejemplo de cruce de peticiones. Orden AX y redes en malla. La variable t está ubicada en el módulo de memoria del nodo 2. No se muestra la respuesta RpX al CC1.

Controlador de coherencia.

En un CC, dado un bloque en un estado transitorio, existe un conjunto de peticiones del CM que el CC responde infiriendo que el CM ha procesado la petición de otro CC antes que la petición del CC (Figura 8.35). Estos cruces de peticiones se corresponden con los descritos en el protocolo previo de este capítulo (Figura 8.20).

Identificación de nuevos cruces de peticiones. El otro conjunto de peticiones que un CC infiere debe responderse una vez finaliza la transacción pendiente (IM, IL, LM). Para efectuar la inferencia se utiliza el tipo de petición y el estado transitorio donde se recibe[14]. Para facilitar la comprensión en el siguiente desarrollo, se replica la Figura 8.32 en el margen izquierdo.

Orden de procesado en el CM	
I	J
P1 load t P2 store t	P1 store t P2 load t

K
P1 store t P2 store t

Dado el orden I de secuencias de acceso (Figura 8.32), el CM procesa la petición del CC2 cuando el bloque está en el estado L en el directorio. El CC1 está identificado en el VP. En consecuencia el CM emite una petición PtObInv al CC1. El CC1 recibe esta petición cuando el bloque está en el estado transitorio IL en la cache C1. Teniendo en cuenta el estado y el tipo de petición, el CC1 infiere un cruce y pospone el procesado.

Dado el orden J de secuencias de acceso, el CM procesa la petición de CC2 cuando el bloque está en el estado M en el directorio. El CC1 está identificado en el VP y el BE = 1. En consecuencia el CM emite una petición PtObL al CC1.

14. Este cruce de peticiones se denomina temprano, ya que la petición llega antes de disponer del bloque.

El CC1 recibe esta petición cuando el bloque está en el estado transitorio IM o LM en la cache C1. Teniendo en cuenta el estado y el tipo de petición el CC1 infiere un cruce y pospone el procesado.

Dado el orden K de accesos a memoria, el razonamiento es similar teniendo en cuenta que la petición del CM, inducida por la petición de CC2, es PtObE.

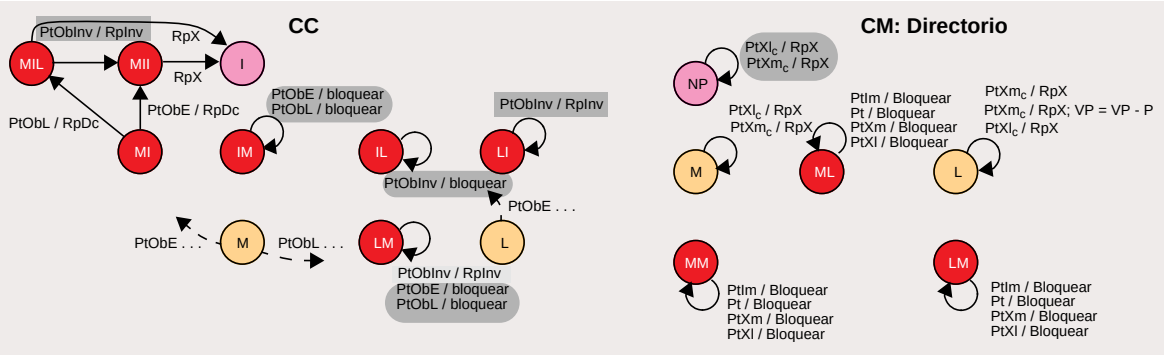


Figura 8.35 Redes distintas para peticiones y respuestas del CM. Diagrama de estados y transiciones de un CM y un CC cuando se producen cruces de peticiones. En sombreado se muestran las diferencias básicas con el protocolo del apartado previo. En sombreado rectangular se muestra el cambio de petición del CM. También se muestran las peticiones del CM en estados estables.

En resumen, en los estados IM y LM un CC puede recibir las peticiones PtObE y PtObL. En los dos casos, el CC infiere un cruce y debe esperar a que llegue la respuesta a su transacción pendiente. Se bloquea el análisis de la cola de peticiones de los CM al CC (CPC). En las dos peticiones el CM está solicitando que el CC suministre el bloque (Figura 8.35)[15].

Un CC, con un bloque en el estado IL, puede recibir una petición PtObInv del CM a este bloque. El CC infiere un cruce de peticiones y pospone el procesado. El CM está solicitando que se invalide el bloque.

Ejemplo. En la Figura 8.36 se muestra un ejemplo donde se observa el procesado, en el CC1, de una petición PtObInv del CM en un estado transitorio y el bloqueo del procesado de una petición PtObE hasta que se dispone del bloque. En el CC1 la petición PtObInv del CM se procesa cuando se recibe (ciclo 6). La petición PtObE del CM llega antes que la respuesta a la petición de CC1 (ciclo 20). El procesado de la misma en el CC se posterga (ciclo 22). La petición del CC3, que ha inducido la petición PtObE del CM al CC1, ha sido procesada en el CM después de la petición de CC1.

15. El bloque lo recibe el CC por la red de respuestas (RMC) y una respuesta del CM siempre se procesa. Las peticiones del CM llegan por la red RP.

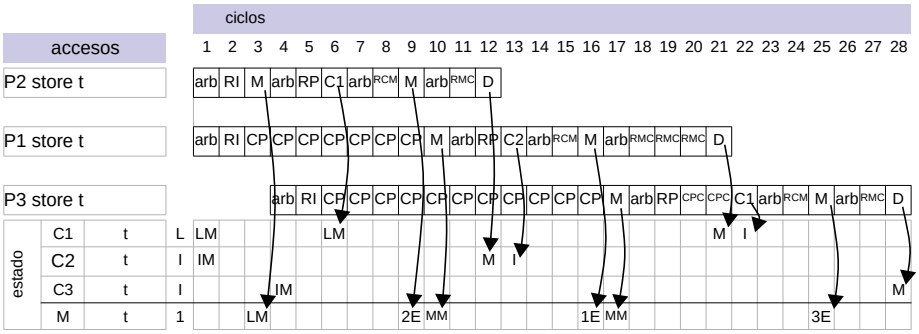


Figura 8.36 Diagrama temporal donde se muestra la inferencia en un CC, cuando un bloque está en un estado transitorio: a) una petición PtOInv se procesa y b) una petición PtObE se posterga hasta disponer del bloque. Varios ciclos en RMC indican retardo en la red. El acrónimo CPC se corresponde con la cola de peticiones en un CC.

En la Figura 8.37 se muestra un ejemplo donde se compara el protocolo del apartado previo (parte izquierda de la figura) con el protocolo de este apartado (parte derecha de la figura).

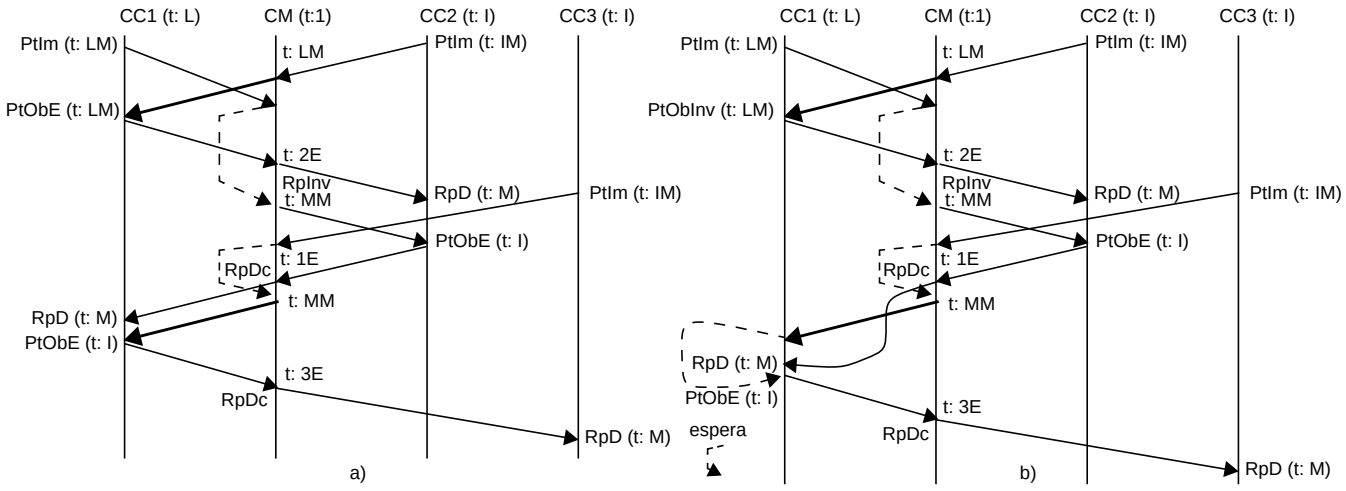


Figura 8.37 Cruces de peticiones en un CC cuando el bloque está en el estado LM. Comunicación entre los CM y los CC: a) una red para transmitir mensajes y b) una red para transmitir peticiones y otra red para transmitir respuestas.

En la parte izquierda de la Figura 8.37 las peticiones y respuestas del CM se reciben por el mismo canal de comunicación (RPR), el cual mantiene el orden de emisión. En el CC1 la segunda petición del CM llega después de la respuesta a la petición del CC1. En la parte derecha de la Figura 8.37

peticiones y respuestas del CM utilizan redes distintas (RP, RCM). En el CC1 la segunda petición del CM llega antes que la respuesta a la petición del CC1. Sin embargo, la petición del CC3, que ha inducido la petición PtObE del CM al CC1, ha sido procesada en el CM después de la petición del CC1.

EL SOLICITANTE ES EL RECEPTOR DE LAS RESPUESTAS

En los protocolos descritos previamente, para obtener la exclusividad de acceso a un bloque, pueden ser necesarios cuatro mensajes en secuencia (4 saltos, Figura 8.1). También son necesarios estos cuatro mensajes cuando una cache tiene el bloque en exclusividad y una transacción accede al bloque. En estos protocolos los CC, involucrados en una acción de coherencia, responden al CM y éste al CC solicitante.

Una forma de reducir la latencia de algunas transacciones es que los CC, involucrados en una acción de coherencia, respondan al CC solicitante en lugar de al CM. En algunas transacciones el número total de mensajes es el mismo, pero la serialización es de tres mensajes (3 saltos).

Por otro lado, se descongestiona al CM. Cuando un CM recibe las respuestas de los CC, involucrados en una acción de coherencia, debe de gestionar hasta P de ellas[16]. Cuando las respuestas de los CC, involucrados en una acción de coherencia, las recibe el CC solicitante, éste debe gestionar una acción de coherencia[17].

Descripción funcional de la secuencia de mensajes en una transacción

En la Figura 8.38 se muestra la descripción funcional de protocolo previo y la del protocolo que se describe en este apartado.

Cuando no hay copia del bloque en otras cache o la transacción es de lectura y una cache no tiene el bloque en exclusividad, la secuencia de mensajes y su serialización es la misma en los dos protocolos (2 pasos, parte superior de la Figura 8.38).

Cuando una cache tiene el bloque en exclusividad la serialización de los mensajes se reduce en un paso. El solicitante recibe el bloque del CC que lo tiene en exclusividad. Si la transacción es de lectura, el CC que suministra el bloque también envia una copia al CM (centro de la Figura 8.38).

16. Si un procesador puede tener k accesos pendientes (por ejemplo, prebúsqueda), el número total es k x P.

17. Si un procesador puede tener k accesos pendientes, el número total es k.

Cuando hay caches con copias del bloque y una transacción solicita obtener la exclusividad, las respuestas de invalidación las recolecta el CC solicitante (parte inferior de la Figura 8.38).

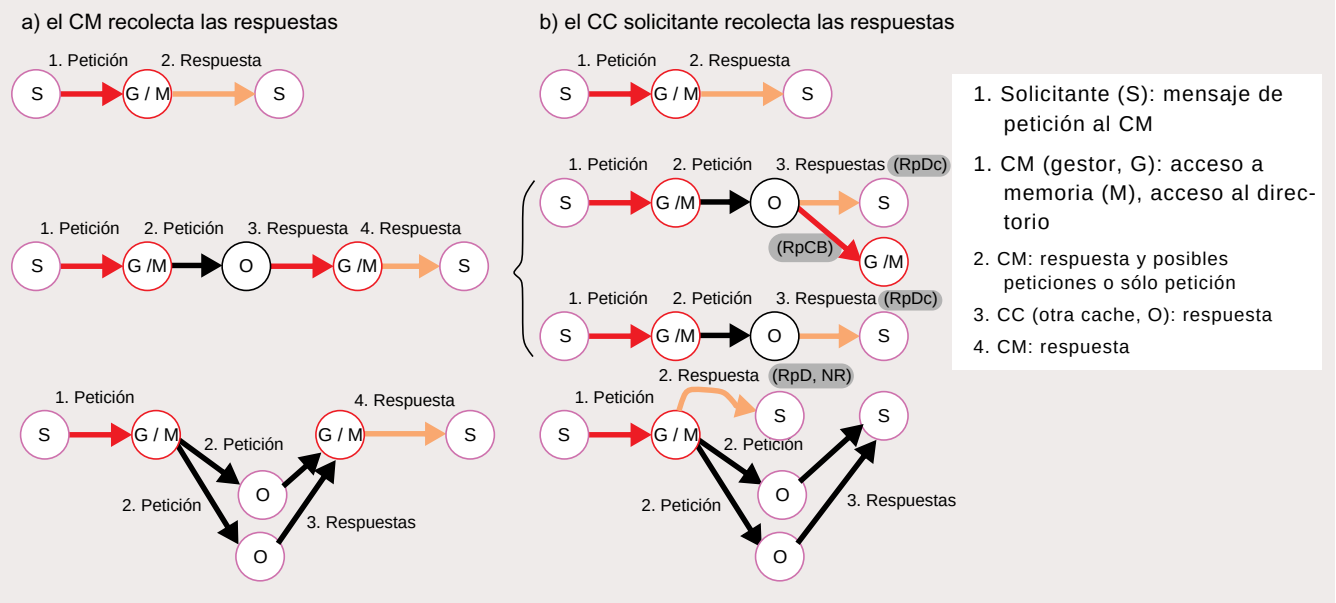


Figura 8.38 Flujos de mensajes en dos protocolo de directorio MLI. a) el CM recolecta las respuestas de los CC y b) el CC solicitante recolecta las respuestas de los CC. En sombreado las diferencias básicas con el protocolo del apartado previo.

En la transacción de la parte inferior de la Figura 8.38, el CM, en la respuesta que emite al solicitante, indica el número de respuestas que debe recolectar para dar por completada la transacción.

El CM da por completada una transacción una vez emite los mensajes de petición y respuesta, excepto en una transacción de lectura y una cache tiene el bloque en exclusividad. En este caso, el CM espera la respuesta, con el bloque, del CC cuya cache tenía el bloque en exclusividad (segundo caso en el centro de la Figura 8.38). El CM utiliza la respuesta para actualizar la memoria.

Notemos que un CM puede empezar a procesar la siguiente transacción a un bloque antes de que los CC involucrados en la transacción previa, al mismo bloque, hayan procesado las peticiones del CM[18]. En estas condiciones, debido a que el CC tiene que inferir el orden entre peticiones y respuestas del CM hay que analizar posibles cruces de mensajes[19]. Estos cruces pueden requerir la modificación del protocolo o la asignación de la red por la cual se transmite un mensaje. En caso contrario el protocolo puede bloquearse.

18. Dos últimos flujos de mensajes de la Figura 8.38.

Organización del multiprocesador

El protocolo requiere que exista una comunicación lógica directa entre los CC. La función de esta comunicación es la transmisión del mensaje de respuesta de cada CC, que participa en la acción de coherencia, al CC solicitante (RDR, Figura 8.39). Esta red lógica también es utilizada por los CM para emitir las respuestas.

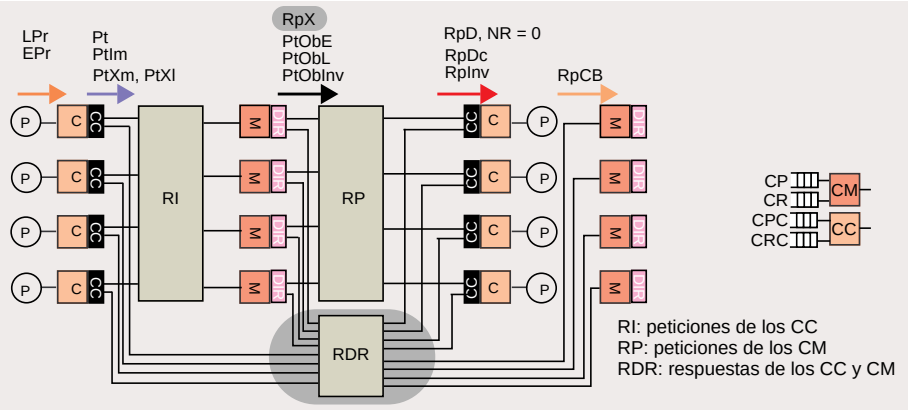


Figura 8.39 Recepción de respuestas de los CC en el solicitante. Organización del multiprocesador. Redes lógicas y mensajes en un protocolo MLI. En sombreado se muestran las diferencias básicas respecto del protocolo del apartado previo.

Un CC genera las respuestas en paralelo. Los mensajes pueden propagarse en paralelo en función de la ocupación de la red.

La cola de mensajes de respuesta en un CC se denomina CRC.

Mensajes del protocolo

En la Figura 8.39 se muestra un esquema genérico de la transmisión de mensajes entre los CC y los CM y viceversa.

En la descripción del protocolo se distiguen los siguientes tipos de mensajes:

- Peticiones de un CC a un CM (Pt, PtIm, PtXm, PtXl)
- Peticiones de un CM a los CC (PtObL, PtObE, PtOInv, RpX)
- Respuestas de un CM a un CC (RpD, NR). La respuesta RpD incluye el bloque y el número de respuestas que debe esperar (NR = K)

19. Un enlace punto a punto mantiene el orden de los mensajes entre emisor y receptor. Sin embargo, las peticiones y respuestas del CM utilizan redes distintas. La red puede ser de tipo crossbar, anillo o malla. El encaminamiento entre un emisor y un receptor debe estar preestablecido.

- Respuestas de un CC a un CC solicitante (RpDc, RpInv)
- Respuesta de un CC a un CM (RpCB)

Los mensajes que se han añadido o modificado respecto al protocolo del Capítulo 7 se muestran en la Tabla 8.3 y en la Tabla 8.4.

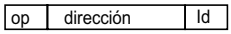
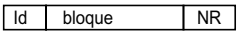
Controlador de coherencia (CC)	Controlador de memoria (CM)	Comentario
Mensajes de petición al CM	**Mensaje de respuesta**	
Pt: petición de bloque 	RpD, NR: respuesta con el bloque y el número de respuestas que debe esperar 	Se lee el bloque de la memoria, se actualiza el directorio, se suministra el bloque y se indica el número de respuestas que el CC solicitante debe recibir (NR).

Tabla 8.3 Petición del CC a un CM y respuesta del CM.

Aunque RpX es una respuesta de un CM a un CC se utiliza la red RP para transmitir el mensaje. Posteriormente se justifica la decisión. El objetivo es que un CC pueda liberar el contenedor al procesar la respuesta del CM, teniendo la seguridad de que posteriormente no recibirá una petición del CM al mismo bloque, que ha sido emitida por el CM antes que la respuesta.

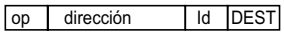
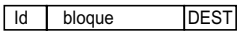
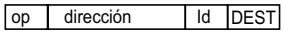
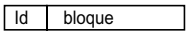
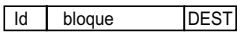
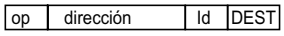
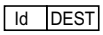
Controlador de memoria	Controlador de coherencia	Comentario
Mensajes de petición a los CC y acciones	**Respuestas**	
PtObE: petición de observación de escritura 	RpDc: respuesta con el bloque e invalidación 	El CC emite una respuesta al solicitante (DEST) con el bloque e invalida el bloque.
PtObL: petición de observación de lectura 	RpCB: respuesta con el bloque al CM  RpDc: respuesta con el bloque al CC 	El CC emite una respuesta al solicitante (DEST) con el bloque y cambia el estado del bloque para indicar que no hay exclusividad. El CC suministra el bloque al CM.
PtObInv: petición de invalidación 	RpInv: respuesta a una petición de invalidación 	El CC que recibe la petición invalida el bloque y responde al solicitante (DEST).

Tabla 8.4 Peticiones de un CM a los CC y respuestas de un CC al CC solicitante y al CM.

Recepción de respuestas en el solicitante

En una transacción donde se solicita la exclusividad, un CC puede recibir dos conjuntos de respuestas[20]: a) la respuesta del CM que, además del bloque, indica el número de respuestas (NR) que deben esperarse y b) las respuestas de confirmación de invalidación de los CC participantes en la acción de coherencia.

La respuesta del CM y las respuestas de los CC se transmiten por la RDR donde, como mucho, se garantiza orden de los mensajes entre emisor y destino. Por tanto, la respuesta del CM y las respuestas de los CC pueden llegar en cualquier orden al CC solicitante. Para gestionar el desorden, en la recepción de las respuestas, se utilizan estados transitorios en el CC.

En la Figura 8.40 se muestra un ejemplo del diagrama de transiciones entre estados. Las respuestas de los CC son RpInv y hay que contabilizar su número. La respuesta del CM es RpD e indica el número de respuestas que deben esperarse (NR).

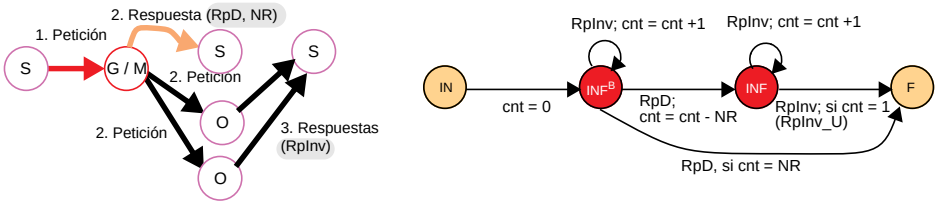


Figura 8.40 Estados transitorio cuandos se esperan varios conjuntos de respuestas.

A medida que llegan las respuestas de invalidación de los CC se contabilizan (INF^B). Cuando llega la respuesta del CM se calcula el número de respuestas de invalidación que aún no han llegado. Cuando han llegado todas las respuestas de invalidación se efectúa la transición a un estado estable (F). En caso contrario, se efectúa una transición a un estado transitorio (INF), que identifica que sólo se esperan respuestas de invalidación. Cuando llega la última respuesta de invalidación (RpInv_U) se efectúa una transición a un estado estable (F). El superíndice B en un estado indica que aún no ha llegada la respuesta del CM, la cual es esperada en la transición entre los estados estables inicial (IN) y final (F).

En resumen, en un CC es necesario utilizar, en ocasiones, un reconocedor de secuencias de respuestas. En este caso se puede expresar como (RpInv*, RpD, RpInv*).

20. El estado del bloque en el directorio es L.

Estados y transiciones

Cuando un CC tiene el bloque en exclusividad y el CM procesa una transacción de lectura hay que actualizar la memoria. En estas condiciones, el CM debe esperar a que responda el CC, con el bloque, antes de procesar otra transacción al mismo bloque. Para identificar la espera de una respuesta de un CC se utiliza un estado transitorio en el CM.

El directorio se utiliza para detectar accesos concurrentes al mismo bloque y gestionarlos (ventana de vulnerabilidad). Esto es, cuando la petición en la cabeza de la CP accede a un bloque en un estado transitorio, el CM efectúa la detección. La gestión utilizada, en todos los protocolos descritos previamente, es bloquear el procesado de esta petición, hasta que el estado del bloque es estable. En este protocolo, en un estado transitorio se procesan peticiones de expulsión.

Directorio. Los estados estables de un bloque en el directorio son los mismos que en el Capítulo 7 (Figura 8.41).

Las respuestas en una transacción de exclusividad son recolectadas por el CC solicitante. Entonces, no son necesarios estados transitorios en el CM para este menester. Ahora bien, en el CM se utiliza un estado transitorio, denominado ML, para identificar la espera de la respuesta del CC que debe suministrar el bloque con el que se actualiza la memoria.

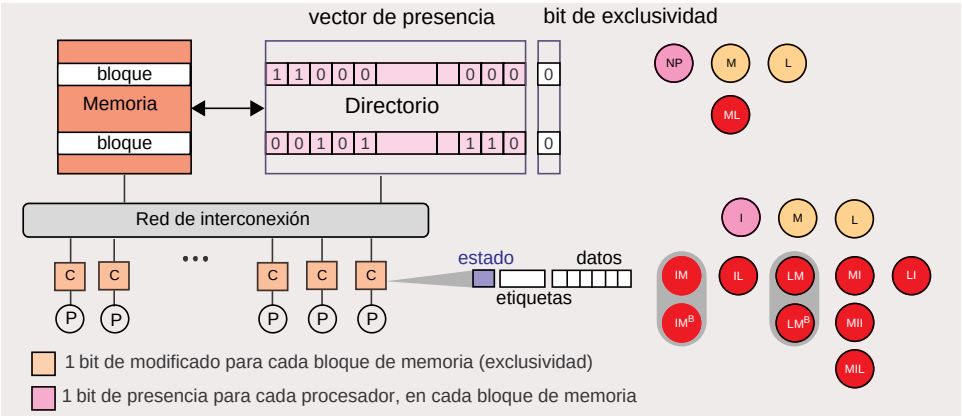


Figura 8.41 Recepción de respuestas de los CC en el solicitante. Estados de un bloque en el directorio y en un contenedor de cache.

Cache. Los estados estables de un bloque en una cache son los mismos que en el Capítulo 7 (Figura 8.41). Para identificar la espera de la respuesta del CM y las respuestas de los CC, a una petición del CC, se utilizan siete estados transitorios (IM^B, IM, IL, LM^B, LM, MI, LI). Los dos nuevos estados transitorios, respecto a los utilizados en el protocolo del apartado previo, se

utilizan para gestionar el desorden en la recepción de la respuesta del CM y las respuestas de los CC, participantes en la acción de coherencia, al CC solicitante.

Para describir el protocolo se utilizan las dos peticiones del procesador que requieren acceder al directorio (fallo en una instrucción load, store o una instrucción store que accede a un bloque sin permiso de exclusividad) y las dos posibilidades de ubicación del bloque solicitado, en memoria o en una cache. Posteriormente se detallan las transiciones entre estados en una expulsión de un bloque de cache.

En la descripción se mostrarán las transiciones entre estados de un bloque, en el CC que efectúa la petición (agente procesador), en el CM y en otros CC (agente observador).

Fallo en lectura

Memoria tiene actualizado el bloque. En la parte izquierda de la Figura 8.42 se muestra el flujo de mensajes en un fallo de lectura. El CC emite un mensaje con una petición de lectura de bloque (Pt). El CM accede al directorio para leer el VP y el BE (estado del bloque) y determina que el estado del bloque es NP o L. El CM emite un mensaje de respuesta (RpD, NR = 0), que incluye el bloque y el número adicional de respuestas que debe esperar, al CC que ha efectuado la petición.

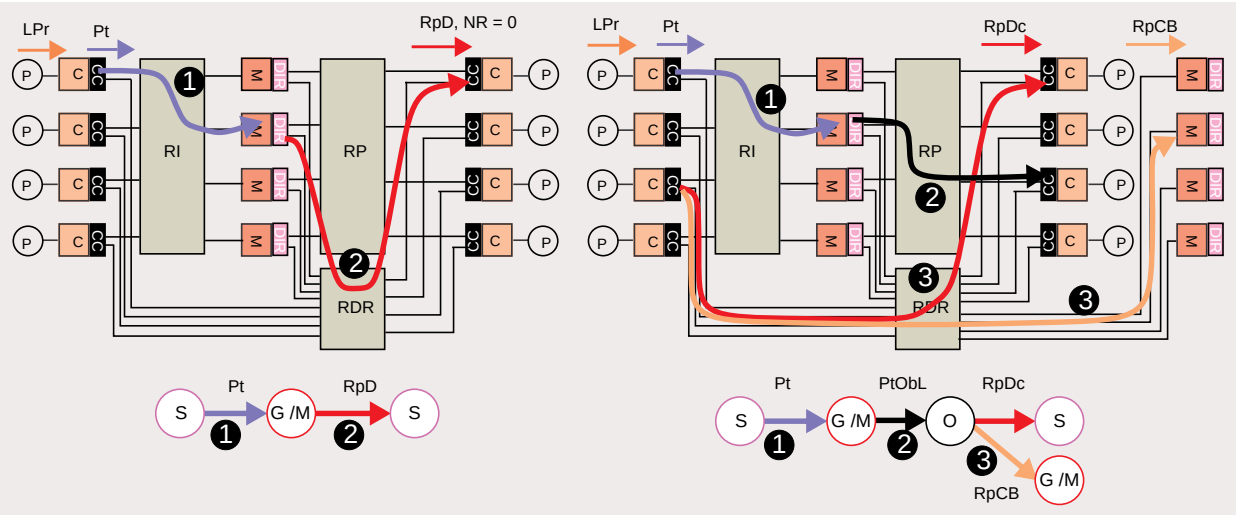


Figura 8.42 Recepción de las respuestas de los CC en el solicitante. Flujo de mensajes en un fallo de lectura.

Una cache tiene el bloque en exclusividad. En la parte derecha de la Figura 8.42 se muestra el flujo de mensajes entre el CC que efectúa la petición y el CM, entre el CM y el CC que tiene el bloque en exclusividad y entre este último y el CC solicitante y el CM.

El CM emite una petición PtObL al CC que tiene el bloque en exclusividad para que suministre el bloque. Este CC responde, con sendos mensajes que incluyen el bloque, al CC solicitante (RpDc) y al CM (RpCB). El CM actualiza el directorio y la memoria al recibir la respuesta del CC.

Diagramas de transiciones entre estados. En la Figura 8.43 se muestran las transiciones entre estados en un fallo de lectura en: a) la cache del CC que efectúa la petición (solicitante), b) el directorio (CM) y c) otros CC. El CC emite el mensaje de petición de lectura y establece IL como estado transitorio del bloque. El bloque permanece en este estado hasta que recibe la respuesta del CM (RpD, Nr = 0) o del CC que tiene el bloque en exclusividad (RpDc), la cual es procesada por el CC que establece como estado estable del bloque el estado L.

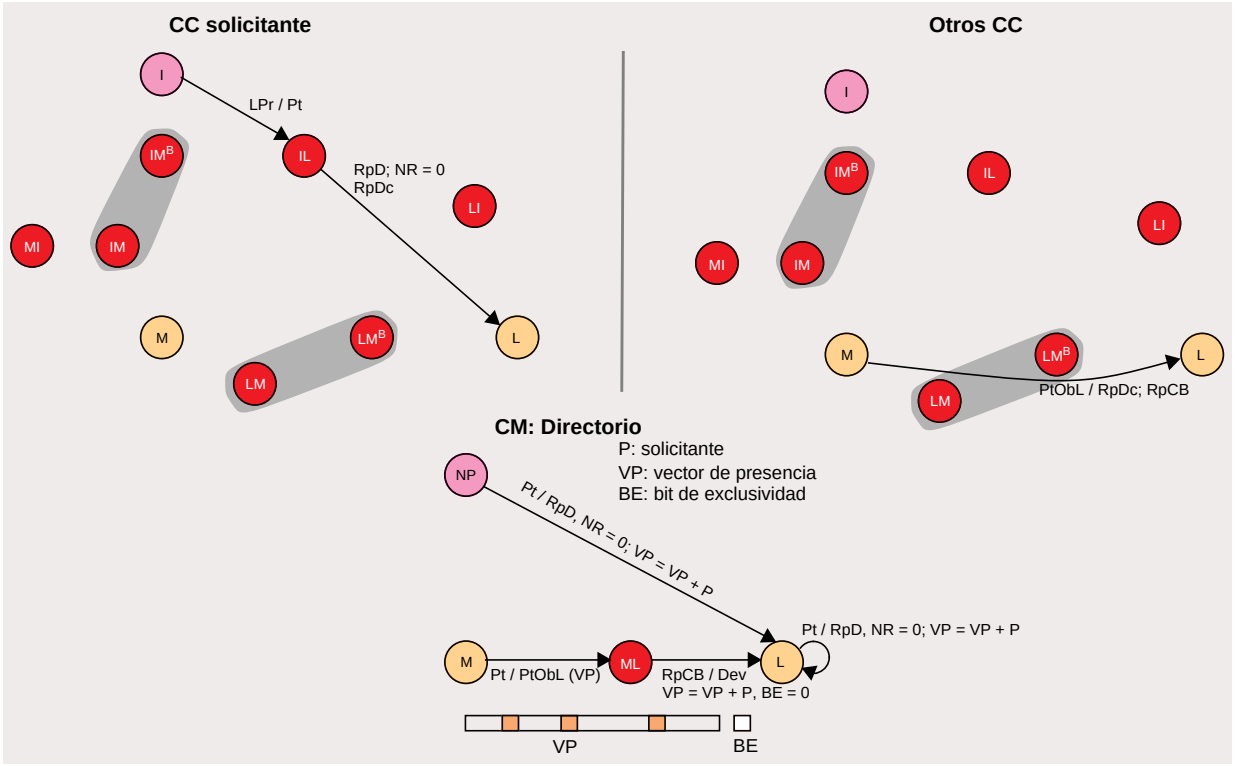


Figura 8.43 Recepción de las respuestas de los CC en el solicitante. Transiciones entre estados en un fallo de lectura.

El CM al recibir el mensaje de petición de lectura accede al directorio y especulativamente a memoria. Si el estado del bloque en el directorio es NP o L, el CM determina que puede responder a la petición utilizando el bloque leído de memoria. El CM, además de enviar la respuesta, actualiza el VP añadiendo al CC, que ha efectuado la petición, en el VP. Los otros CC no reciben peticiones de coherencia desde el CM. En la parte izquierda de la Figura 8.44 se muestra el diagrama temporal.

Si el estado del bloque en el directorio es M, el CM establece el estado transitorio ML. El CM emite una petición al CC que tiene el bloque en exclusividad (PtObL). Este CC, al recibir la petición, responde con mensajes, que incluyen el bloque, tanto al CC (RpDc) como al CM (RpCB) y establece como nuevo estado del bloque el estado L. El CM al recibir la respuesta (RpCB) actualiza la memoria con el bloque y modifica el VP y el BE, estableciendo en el directorio como estado del bloque el estado L. En la parte derecha de la Figura 8.44 se muestra el diagrama temporal[21].

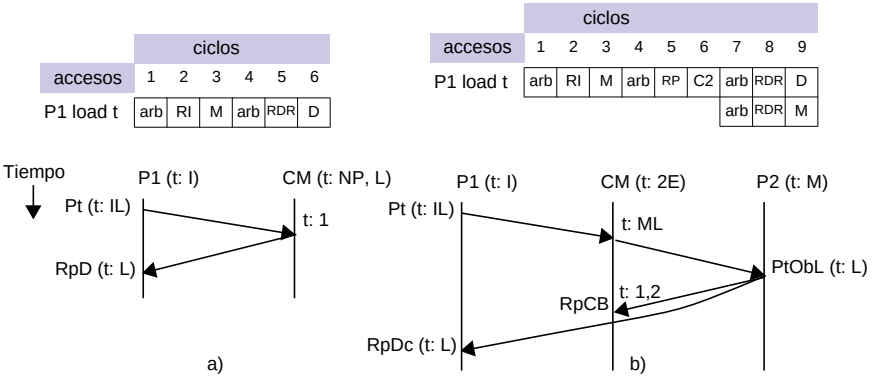


Figura 8.44 Recepción de las respuestas de los CC en el solicitante. Diagrama temporal: a) estado NP o L en el directorio y b) estado M en el directorio.

Fallo de escritura o petición de exclusividad

Memoria tiene el bloque actualizado. En la parte izquierda de la Figura 8.45 se muestra el flujo de mensajes en un fallo de escritura cuando no hay copias del bloque en otras caches. El CC solicitante emite un mensaje con una petición de bloque con intención de modificación (PtIm). El CM, al recibir el mensaje, lee el estado del bloque en el directorio y accede especulativamente a memoria. El CM determina que el estado del bloque es NP y responde con el bloque al CC solicitante. Además, actualiza el VP y el BE.

21. Un CM o un CC generan los mensajes en paralelo. La transmisión de los mismos en paralelo depende de los conflictos en la red correspondiente.

Hay copia del bloque en otras caches. En la parte derecha de la Figura 8.45 se muestra el flujo de mensajes en un fallo de escritura cuando otras caches tiene copia del bloque. El CM, utilizando la información de estado del bloque en el directorio, determina los CC a los que es necesario enviar una petición de invalidación. El CM responde al CC solicitante con el bloque y el número de respuestas que debe recolectar (RpD, Nr = K). Por otro lado, el CM emite un mensaje de invalidación (PtObInv) a cada uno de los CC involucrados en la acción de coherencia. Posterormente, el CM modifica el VP y el BE. Las respuestas a las peticiones de invalidación las recolecta el CC solicitante.

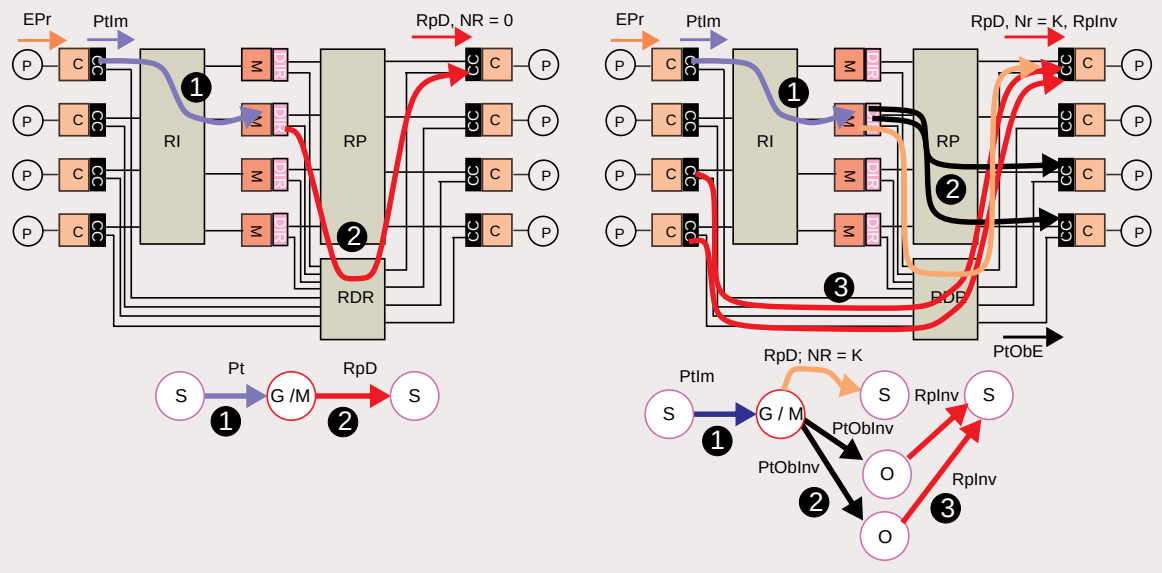


Figura 8.45 Recepción de las respuestas de los CC en el solicitante. Flujo de mensajes en un fallo de escritura.

Una cache tiene el bloque en exclusividad. En la Figura 8.46 se muestra el flujo de mensajes en un fallo de escritura cuando otra cache tiene el bloque en exclusividad. El CM emite un mensaje, de suministro del bloque e invalidación (PtObE), a la cache que tiene el bloque en exclusividad. El CC correspondiente responde al CC solicitante con el bloque, además invalida el bloque en su cache. El CM actualiza el VP para identificar la cache que ahora tiene el bloque en exclusividad.

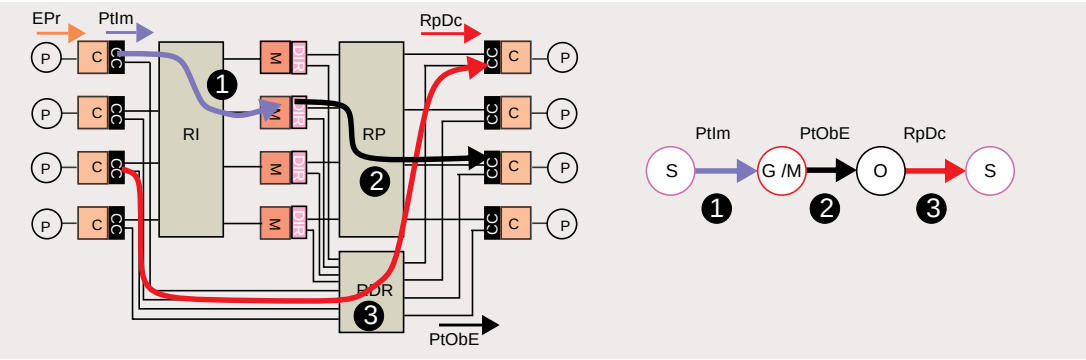


Figura 8.46 Recepción de las respuestas de los CC en el solicitante. Flujo de mensajes en un fallo de escritura cuando una cache tiene el bloque en exclusividad.

Diagramas de transiciones entre estados. En la Figura 8.47 se muestran las transiciones entre estados en la cache del CC que efectúa la petición, el directorio y los otros CC. El CC solicitante establece un estado transitorio (IM o LM), en función del estado estable inicial, esperando la respuesta o respuestas. El patrón de transición entre estados sigue la exposición efectuada en el apartado “Recepción de respuestas en el solicitante”.

En el directorio el estado del bloque pasa de L a M o de NP a M, el VP se actualiza en consecuencia y el BE se activa. Además, el CM emite mensajes de respuesta (RpD, NR = K) y de petición de invalidación (PtObInv) o de petición de observación de escritura (PtObE).

Cuando un CC emite una petición PtIm, estando el bloque en estado L en la cache, el bit de presencia del CC está activado en el VP. Por tanto, hay que excluirlo de la lista de los CC que reciben una petición PtObInv (VP - P). El CC, al recibir la respuesta RpD, NR = K, siempre almacena el bloque recibido en el contenedor correspondiente, tanto si el estado estable inicial es L como I (centro de la Figura 8.48). Un caso particular es que no haya copias del bloque o el CC solicitante tenga la única copia (parte izquierda de la Figura 8.48).

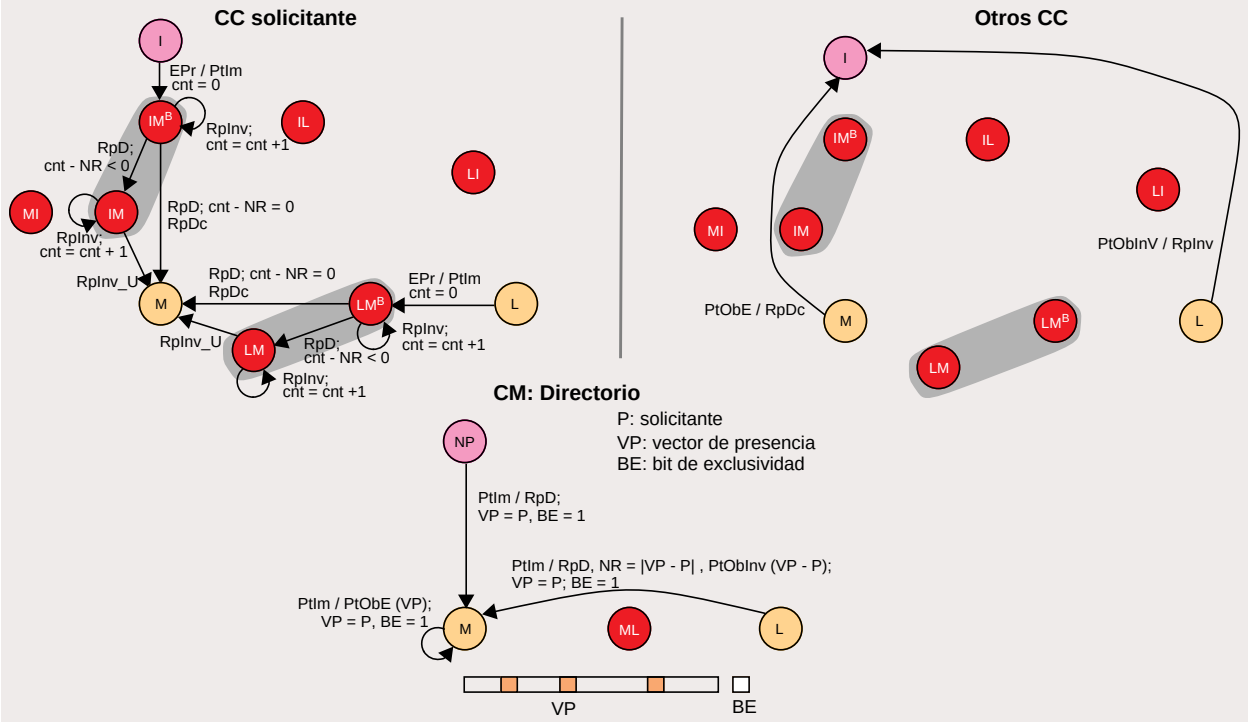


Figura 8.47 Recepción de las respuestas de los CC en el solicitante. Transiciones entre estados en un fallo de escritura o solicitud de exclusividad. RpInv_U indica cnt = 1 y recepción de un mensaje RpInv.

Un CC que tiene copia del bloque, al recibir la petición PtObInv invalida la copia del bloque y responden al CC solicitante (RpInv). Cuando un CC tiene el bloque en exclusividad, el CM, al procesar la petición PtIm, emite una petición PtObE. El CC que tiene el bloque en exclusividad suministra el bloque e invalida el contenedor que almacena el bloque (RpDc, parte derecha de la Figura 8.48).

Un CM genera todas las peticiones y respuestas a los CC, inducidas por una petición de un CC, en paralelo. Su transmisión en paralelo depende de los conflictos en la red correspondiente. Un destinatario de mensajes sólo puede recibir un mensaje en cada instante de tiempo.

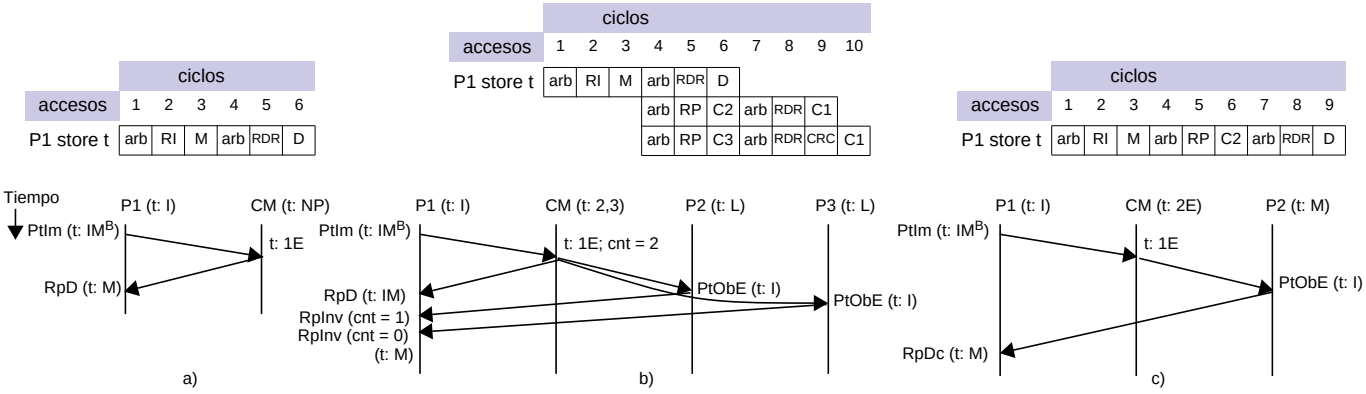


Figura 8.48 Recepción de las respuestas de los CC en el solicitante. Diagrama temporal: a) estado NP en el directorio, b) estado L en el directorio y c) estado M en el directorio. El acrómino CRC indica cola de respuestas en un CC.

Expulsión

El directorio es preciso y las peticiones de expulsión esperan una respuesta del CM. En la Figura 8.49 se muestra el flujo de mensajes cuando se expulsa un bloque de cache.

El CC efectúa una petición de expulsión (PtXm, PtXI) que va acompañada del bloque si el estado es M (PtXm). El CM en cualquier caso se actualiza la entrada correspondiente en el directorio. Además, se actualiza memoria si el bloque expulsado estaba en el estado M en la cache. El CM responde al CC confirmando que se ha realizado la acción solicitada. Esta respuesta se transmite por la red RP.

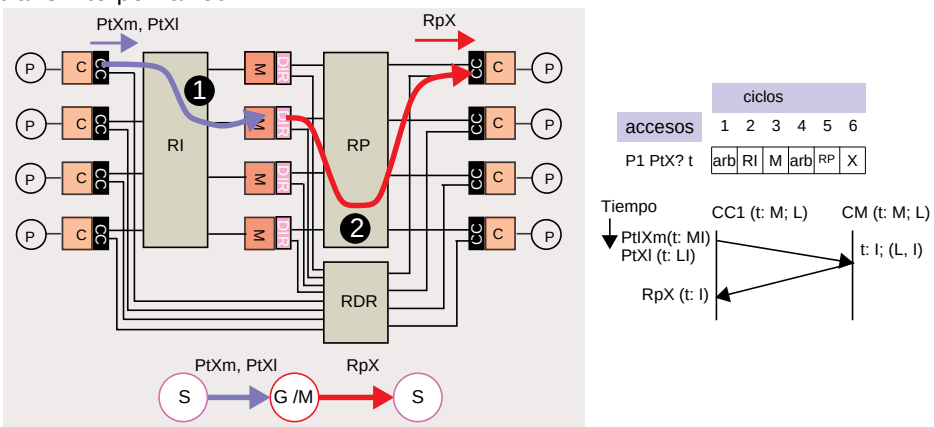


Figura 8.49 Recepción de respuestas de los CC en el solicitante. Expulsión de un bloque de cache.

En la Figura 8.50 se muestran las transiciones entre estados en: a) la cache del CC que expulsa el bloque, b) el directorio y c) otros CC. En otros CC no se producen transiciones entre estados.

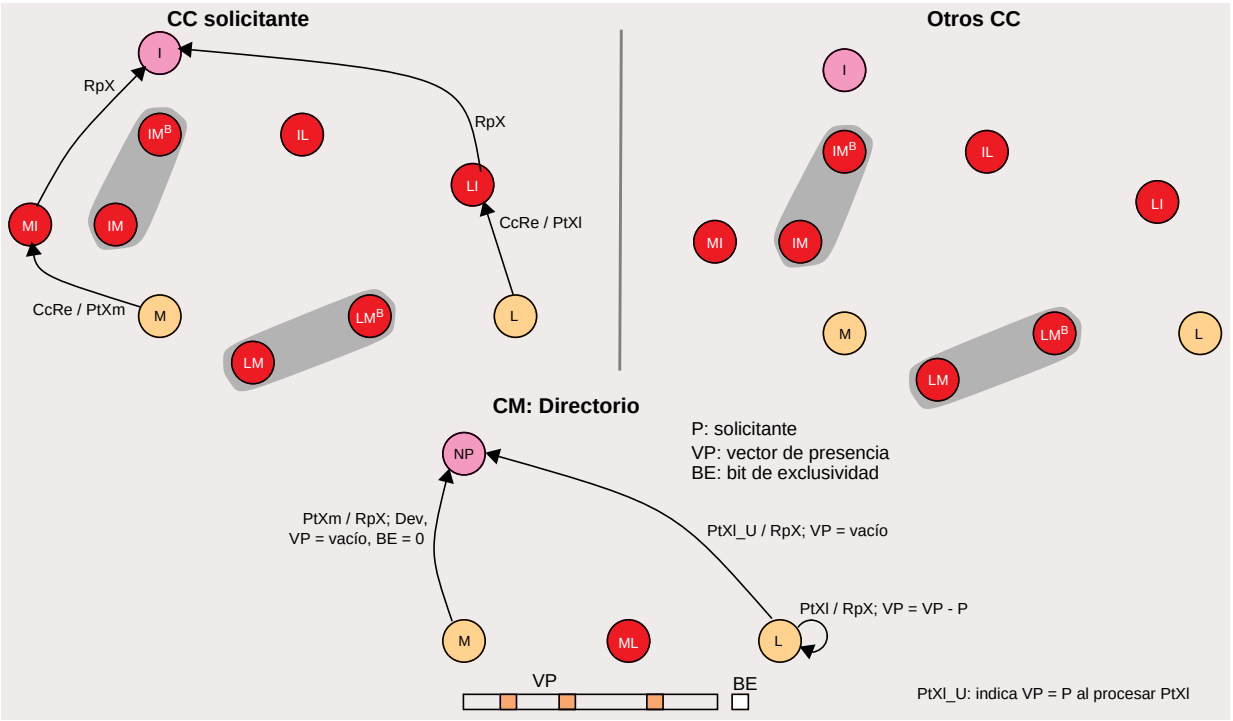


Figura 8.50 Recepción de las respuestas de los CC en el solicitante. Transiciones entre estados cuando se expulsa un bloque.

En un CC, para distinguir entre la expulsión de un bloque en el estado M o en el estado L, se utilizan distintos tipos de peticiones y también estados transitorios distintos; MI y LI respectivamente. Cuando el CM procesa la petición actualiza la memoria si la petición es PtXm y en cualquiera de los dos casos, se actualiza la entrada correspondiente del bloque en el directorio (VP). Finalmente el CM envia una respuesta de confirmación. El CC al recibir la respuesta cambia el estado del bloque al estado I.

Diagrama completo de estados y transiciones

En las Figura 8.51 y Figura 8.52 se muestran, respectivamente, los diagramas de transiciones entre estados de un bloque en cache y en el directorio. En el primer diagrama están incluidas las peticiones del procesador que no requieren iniciar transacciones explícitas de coherencia. En el CC se distinguen, en diagramas separados, las transiciones iniciadas por el agente procesador (CC solicitante) y el agente observador (otros CC).

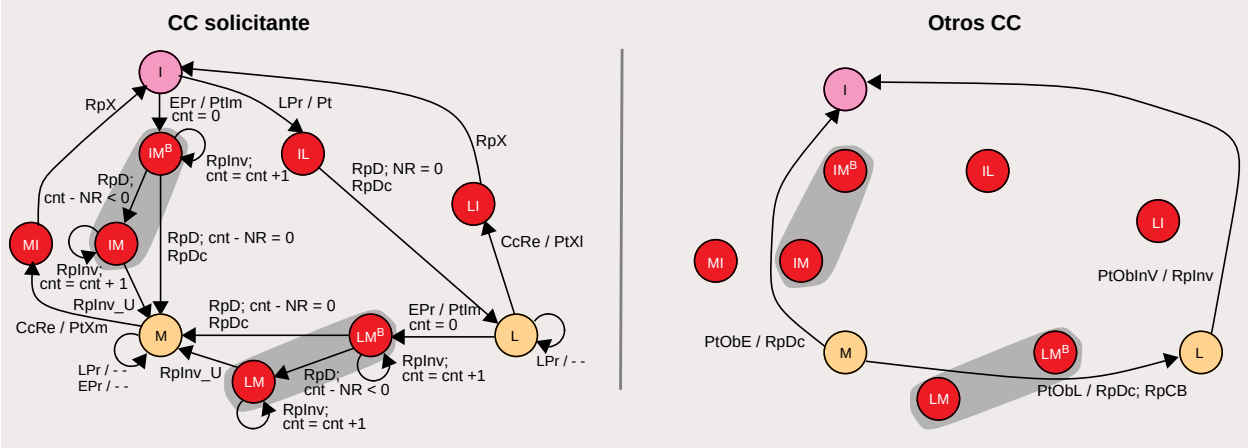


Figura 8.51 Recepción de las respuestas de los CC en el solicitante. Estados y transiciones entre estados de un bloque en una cache.

En el diagrama del CM (Figura 8.52) hay que distinguir dos casos al procesar la petición PtXI de un CC (PtXI, PtXI_U). El estado final depende de si el CC, que efectúa la petición, es el único que está en el vector de presencia.

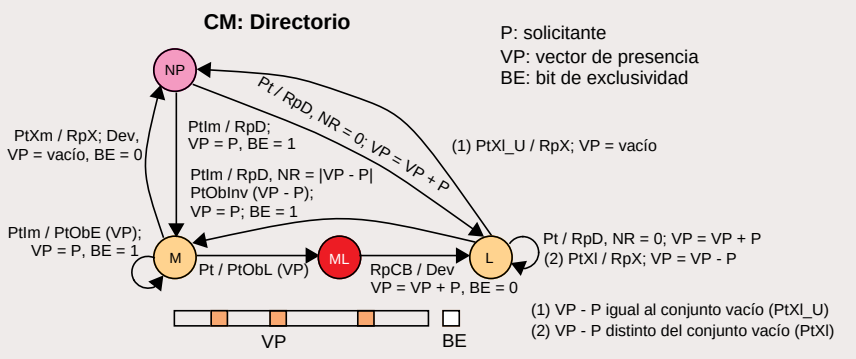


Figura 8.52 Recepción de las respuestas de los CC en el solicitante. Estados y transiciones entre estados de un bloque en el directorio.

Inferencia en un CC del orden entre peticiones y respuestas emitidas desde un CM

En este protocolo, en ocasiones, el CM no conoce el instante en que una transacción ha finalizado[22]. Las respuestas a las peticiones del CM las recolecta el CC solicitante. Esta decisión permite reducir la ocupación del CM,

22. En concreto, en una transición desde los estados L y M al estado M.

ya que el directorio sólo se visita dos veces en un tipo de transacción. En las otras transacciones el directorio se visita una vez. Entonces, el CM puede iniciar el procesado de una petición de un segundo CC antes de que la transacción, al mismo bloque, de un primer CC haya finalizado.

Esta circunstancia puede dar lugar a que la respuesta de un CM a un CC se adelante a una petición que ha efectuado previamente el CM al mismo CC (Figura 8.25 a). Recordemos que la peticiones y respuestas del CM se transmiten por redes distintas.

Una respuesta del CM puede adelantar a una petición previa del CM al mismo bloque

En primer lugar supondremos que un CM no procesa peticiones a un bloque en un estado transitorio en el directorio (ML)[23]. El CM espera la respuesta del CC involucrado en la acción de coherencia para establecer un estado estable del bloque en el directorio. Posteriormente, puede procesar otra petición que accede al bloque.

Otro CC está involucrado en la acción de coherencia. En estados estables el CM procesa peticiones de los CC. En una transición de un bloque del estado L al estado M, el CM no espera respuestas de los CC involucrados en la acción de coherencia (Figura 8.53). El encargado de recolectar las respuestas es el CC que solicita el bloque en exclusividad (CC2). Este CC2 no procesa ninguna petición del CM al mismo bloque hasta que ha recolectado todas las respuestas de los CC involucrados en la acción de coherencia. Por tanto, si el CC1 está incluido en el conjunto de CC que deben responder al CC2, el CC1 no puede recibir una respuesta a su petición, si el CM ha procesado la petición del CC1 después de la petición del CC2, el cual está esperando la respuesta del CC1 entre otras[24].

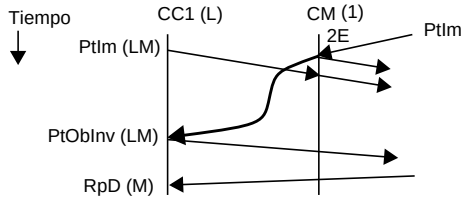


Figura 8.53 Petición de un CC que requieren la colaboración de terceros.

23. Posteriormente se relaja la hipótesis y el protocolo procesa peticiones de expulsión en el estado transitorio.

24. El CC2, que debe emitir la respuesta, está esperando la respuesta del CC1, entre otras, para pasar a un estado estable. Una vez esté en el estado estable responderá al CC1, si es el caso.

Para que se produzca un cruce, el CM debe procesar una petición que modifique el estado del bloque en la cache[25]. Por otro lado, no debe estar involucrado otro CC en la acción de coherencia.

Para que el CM pueda responder a una petición de un CC sin ser necesaria, en ningún caso, la colaboración de otros CC, las peticiones del CC deben ser de expulsión (PtXm y PtXl).

Bloque en el estado L. El estado transitorio del bloque es LI. La petición del CC1 al CM es PXl (Figura 8.54).

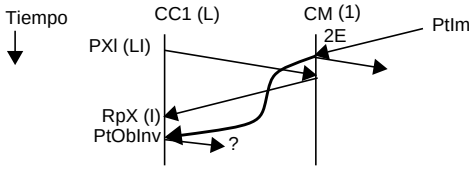


Figura 8.54 Bloque en estado L en la cache C1. Petición de otro CC.

El CM ha procesado una petición PtIm de otro CC, la cual induce que el CM emita una petición PtObInv al CC1. Al procesar el CM la petición PtXl el estado del bloque en el directorio es un estado estable (M). Como la petición PtXl no es esperada, el CM infiere un cruce de peticiones y responde. El CC1 procesa la respuesta RpX antes que la petición PtObInv.

El CC1 al procesar la respuesta RpX libera el contenedor de cache y el estado del bloque en la cache es inválido. Entonces, puede pensarse en que el CC1 responda al CC que ha inducido la petición del CM, ya que la acción de invalidar ha sido efectuada al recibir la respuesta RpX del CM.

Bloque en el estado M. El estado transitorio del bloque es MI. La petición del CC1 al CM es PtXm. El estado del bloque en el directorio es M (Figura 8.55).

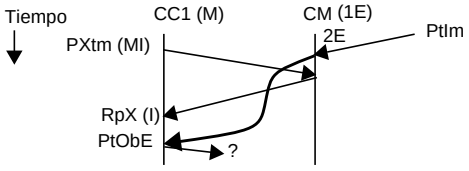


Figura 8.55 Bloque en estado M en la cache C1. Petición de otro CC.

25. Un CM no conoce cuando finaliza una transacción (dos últimos flujos de mensajes de la Figura 8.38). Recordemos que es el CC solicitante el que recolecta las respuestas o respuesta de los CC involucrados en la acción de coherencia. Por otro lado, las peticiones y respuestas del CM utilizan redes distintas. Entonces, una petición de un CM a un CC puede llegar al CC después de que el CC haya recibido una respuesta del CM y esta respuesta haya sido emitida por el CM después de la petición.

El CM ha procesado una petición PtIm de otro CC, la cual induce que el CM emita una petición PtObE al CC1. Al procesar el CM la petición PtXm el estado del bloque en el directorio es un estado estable (M). La petición PtXm no es esperada, ya que el CC1 no está identificado en el VP. Entonces, el CM infiere un cruce de peticiones y responde. El CC1 procesa la respuesta RpX antes que la petición PtObE.

El CC1 al procesar la respuesta RpX libera el contenedor de cache y el estado del bloque en la cache es inválido. Entonces, cuando procesa la petición PtObE no dispone del bloque y no puede responder al CC que ha inducido la petición del CM[26].

En estas condiciones, una alternativa es que la respuesta RpX se transmita por la misma red que las peticiones de los CM. Mediante esta decisión, como la RP mantiene el orden entre emisor y receptor, una petición de un CM y la respuesta a una petición de expulsión de un CC, al mismo bloque, llegan de forma ordenada al CC que efectúa la petición de expulsión (Figura 8.56).

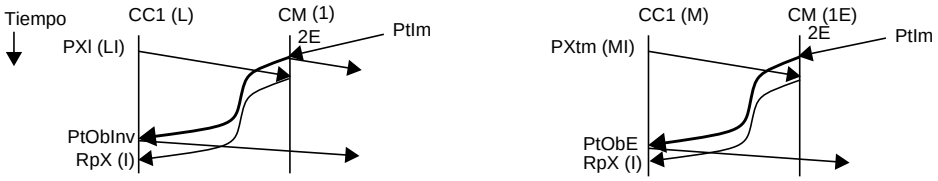


Figura 8.56 Transmisión de la respuesta RpX por la misma red que las peticiones de los CM.

Controlador de memoria

El procesado de una petición en un CM puede necesitar solicitar el bloque a una cache y esperar la respuesta (petición pendiente; ventana de vulnerabilidad). Durante este periodo de tiempo el estado del bloque es transitorio (ML, Figura 8.57).

Respecto de los protocolo de apartados previos se ha incrementado el tipo de transacciones donde el directorio sólo se visita una vez. Sólo es necesario visitar el directorio dos veces cuando una cache tiene el bloque en exclusividad y la petición es de lectura. En este caso, se ha reducido el número de pasos de la transacción (serialización).

26. Se produce abrazo mortal.

Peticiones concurrentes a bloques distintos

Mientras un CM está esperando la respuesta de un CC, puede extraer otra petición de la CP. Si el bloque de memoria, al que accede esta segunda petición, es distinto de los bloques que están en un estado transitorio, el CM puede iniciar el procesado de la petición y por tanto existe concurrencia.

En la Figura 8.57 se muestra la concurrencia en el procesado de peticiones en el CM. Las dos primeras peticiones compiten por acceder al CM. La primera petición accede a un bloque que tiene otra cache en exclusividad. La segunda petición es para obtener la exclusividad y hay copias del bloque en otras caches. Como el CM está ocupado, la petición se espera en la CP. El CC2 recibe el bloque del CM y las respuestas de los CC participantes en la acción de coherencia. La tercera petición es para obtener la exclusividad. Como el CC2 está procesando un mensaje de respuesta de otros CC se espera en la CPC[27]. La cuarta petición accede a un bloque que está actualizado en la memoria.

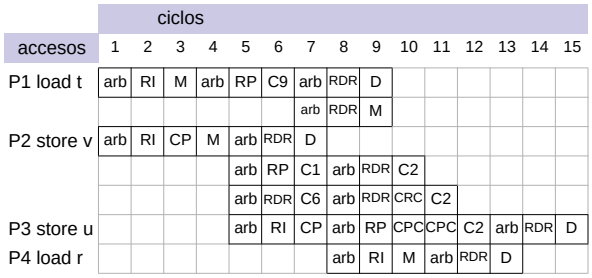


Figura 8.57 Concurrencia en un CM. Las peticiones referencian bloques distintos. Los acrónimos CP, CPC y CRC indica respectivamente cola de peticiones en el CM, cola de peticiones en un CC y cola de respuestas en un CC.

Peticiones concurrentes al mismo bloque

Ventana de vulnerabilidad. Mientras el CM está esperando una respuesta puede analizar una petición al mismo bloque (estado transitorio ML).

El directorio es el punto de ordenación de las peticiones al mismo bloque. En otras palabras es el encargado de serializar las peticiones al mismo bloque[28]. Para ello, el CM bloquea el análisis de la CP, cuando la petición que hay en la cabeza accede a un bloque en un estado transitorio (detección). El bloqueo perdura hasta que el CM recibe la respuesta relacionada con el bloque que determina el bloqueo (gestión). Esto es, cuando el bloque está en un estado estable en el directorio.

27. En un CM y en un CC el procesado de las respuestas es prioritario frente a las peticiones.

28. En este protocolo, el CC que tiene el bloque en exclusividad también colabora.

Un CM no procesa peticiones de los CC (Pt y PtIm) a un bloque en un estado transitorio (Figura 8.58). Ahora bien, en el estado transitorio ML se procesan peticiones de expulsión.

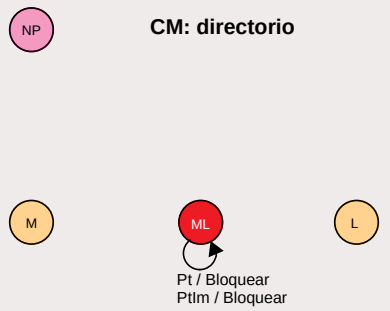


Figura 8.58 Recepción de respuestas de los CC en el solicitante. Bloqueo del procesado de peticiones en el estado transitorio del directorio.

Identificación de cruces de peticiones

En el protocolo del Capítulo 7 y en los protocolos previos de este capítulo, en un CM se bloquea el procesado de peticiones a un bloque en un estado transitorio. La petición bloqueada se procesa al finalizar la transacción pendiente en el CM. Esto es, cuando el bloque está en un estado estable.

En el protocolo descrito en este apartado se procesan peticiones de expulsión (PtXm, PtXl) cuando un bloque está, en el directorio, en el estado transitorio ML. El paso por el estado ML es debido a una petición Pt de un CC, cuando el bloque está en el estado M en el directorio. Esto es, cuando una cache tiene el bloque en exclusividad y el CC de otra cache emite una transacción de lectura.

Desde el estado transitorio ML, un CM efectúa una transición al estado estable L cuando recibe la respuesta del CC involucrado en la acción de coherencia. Entonces, es suficiente analizar los cruces de peticiones que en el Capítulo 7 se infieren cuando un bloque está en el estado L en el directorio. La inferencia del cruce de peticiones en el estado ML o en el estado L depende de la temporalidad del procesado de las peticiones en el CM. En la Figura 8.59 se indican los ordenes de procesado en un CM de las secuencias de acceso a memoria utilizadas en el Capítulo 7.

Orden de procesado en el CM		
B	**F**	**G**
P2 store t	P2 load t	P2 store t
P3 load t	P1 PtXm t	P3 load t
P1 PtXl t		P1 PtXm t

Figura 8.59 Cruces de peticiones en el CM.

Dado el orden B de accesos a memoria, se puede inferir un cruce en el estado ML o en el estado L. En este último caso, la petición del CC1 llega al CM cuando ha finalizado la transacción de CC3 (parte derecha de la Figura 8.60). En los dos estados (ML y L) el CC1 no está identificado en el VP. Dado el orden G se produce la misma inferencia. Además, el tipo de petición del CC1 no es esperada.

Dado el orden F de accesos a memoria, se puede inferir un cruce en el estado ML o en el estado L. En este último caso, la petición del CC1 llega al CM cuando ha finalizado la transacción del CC2. En los dos estados (ML y L) el CC1 está identificado en el VP, pero el tipo de petición no es esperada.

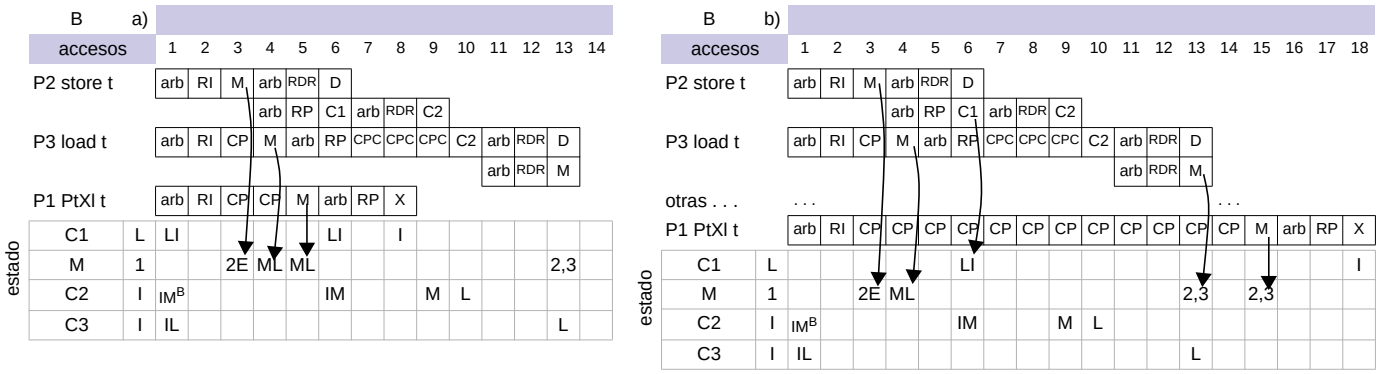


Figura 8.60 Temporalidad en un cruce de peticiones: a) estado ML, b) estado L. En el estado L el CM procesa otras peticiones o el encaminamiento por la RI requiere varios ciclos.

El resto de cruces que se infiere en el directorio son los mismos que en los protocolos previos. En Figura 8.61 se muestran todos los cruces de peticiones.

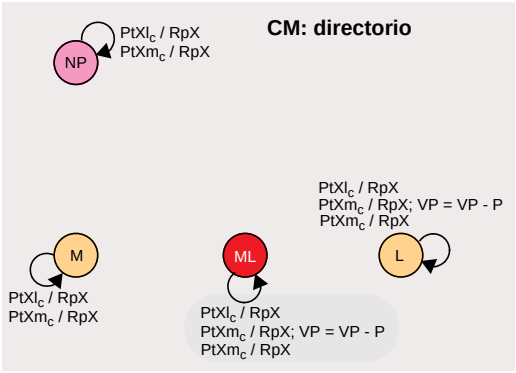


Figura 8.61 Recepción de respuestas de los CC en el solicitante. Cruces de peticiones en el directorio.

Controlador de coherencia. Gestión de cruces de peticiones

Los conjuntos de cruces de peticiones en un CC son los mismos que los identificados en el protocolo descrito previamente en este capítulo, donde se utiliza una red para los mensajes de petición de los CM y otra red para los mensajes de respuesta.

La diferencia es que en el protocolo previo de este capítulo hay un estado transitorio entre los estados I y M y entre los estados L y M. En el protocolo de este apartado hay dos estados transitorios[29]. El cruce se puede inferir en cualquiera de ellos dos y la respuesta es la misma que en el estado equivalente del protocolo previo (Figura 8.62).

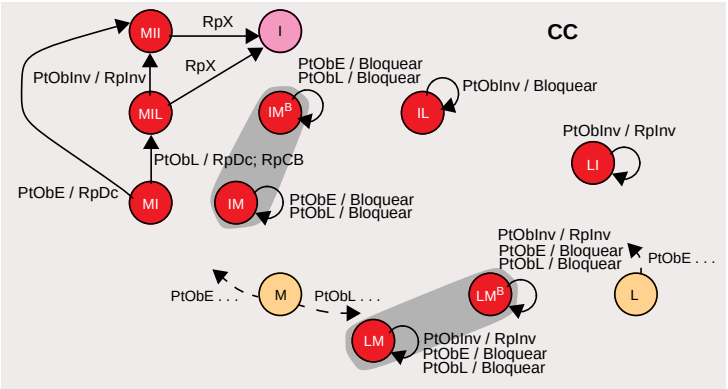


Figura 8.62 Recepción de respuestas de los CC en el solicitante. Diagramas de transiciones en un CC cuando se produce un cruce entre peticiones a un bloque. También se muestran peticiones del CM en estados estables.

29. Los estados IM^B e IM son equivalente al estado IM en el protocolo previo. De la misma forma los estados LM^B y LM son equivalente al estado LM del protocolo previo.

RED GENERICA

Por red genérica se entiende una red que no mantiene ningún orden entre los mensajes transmitidos entre un emisor y un receptor.

En los protocolos descritos previamente en este capítulo la red mantiene el orden de los mensajes de petición emitidos desde un CM a un CC. En este apartado la red no garantiza ningún orden punto a punto. Esto es, la red no garantiza orden en la transmisión de mensajes entre un emisor y un receptor. Recordemos que un CC tampoco dispone de información temporal (orden) entre los mensajes de petición del CM y los mensajes de respuesta.

En un protocolo donde el recolector de las respuestas es el CM, siempre que el CM responde al solicitante éste tiene constancia de que todos los CC, involucrados en la acción de coherencia, la han observado. Por otro lado, durante el lapso de tiempo que requiere el procesado de una transacción, el CM no procesa otras transacciones al mismo bloque. En consecuencia, un CC no puede recibir una segunda petición del CM al mismo bloque sin haber respondido a la primera. Tampoco un CC puede recibir una respuesta a una petición, sin haber respondido antes a una petición del CM, inducida por una petición de otro CC, que el CM ha emitido antes. Un CM antes de procesar la petición de un CC espera las respuestas de la petición previa.

En un protocolo donde el recolector de las respuestas, de una acción de coherencia, es el CC solicitante, es necesario añadir mensajes en el protocolo de coherencia para establecer una ordenación. Estos mensajes aumentan el tráfico en las redes y en ocasiones introducen serialización. Esto es, una transacción tiene más pasos (Figura 8.63).

En este protocolo, donde el CC solicitante recolecta las respuestas, un CM sólo conoce cuando ha finalizado su trabajo correspondiente a una acción de coherencia. Sin embargo, un CM no conoce, en ocasiones, cuando los otros CC, involucrados en la acción de coherencia, la han observado. Es el CC solicitante el que disponde de este conocimiento. En estas condiciones, el CC solicitante puede notificar, mediante un mensaje, al CM la finalización de la transacción. En ocasiones, puede ser el único CC que participa en la acción de coherencia el que emite el mensaje al CM.

El desarrollo se efectúa utilizando un protocolo MLI donde el recolector de las respuestas es el CC solicitante.

Descripción funcional de la secuencia de mensajes en una transacción

En la Figura 8.63 se muestra la descripción funcional del protocolo previo de este capítulo y la del protocolo que se describe en este apartado.

Cuando no hay copia del bloque en otras cache o la transacción es de lectura y una cache no tiene el bloque en exclusividad, la secuencia de mensajes y su serialización es la misma en los dos protocolos (2 pasos, parte superior de la Figura 8.63).

Cuando una cache tiene el bloque en exclusividad el número de pasos es el mismo. El solicitante recibe el bloque del CC que lo tiene en exclusividad. Sin embargo, tanto si la transacción es de lectura como si es de escritura, el CC, que tenía el bloque en exclusividad, notifica al CM la emisión de la respuesta al CC solicitante. En particular, si la transacción es de lectura el CC envia una copia del bloque al CM junto con la notificación (centro de la Figura 8.63).

Cuando hay caches con copias del bloque y una transacción solicita obtener la exclusividad, las respuestas de invalidación las recolecta el CC solicitante (parte inferior de la Figura 8.63). Una vez recolectados todos los mensajes de respuesta, el CC solicitante envia una notificación al CM.

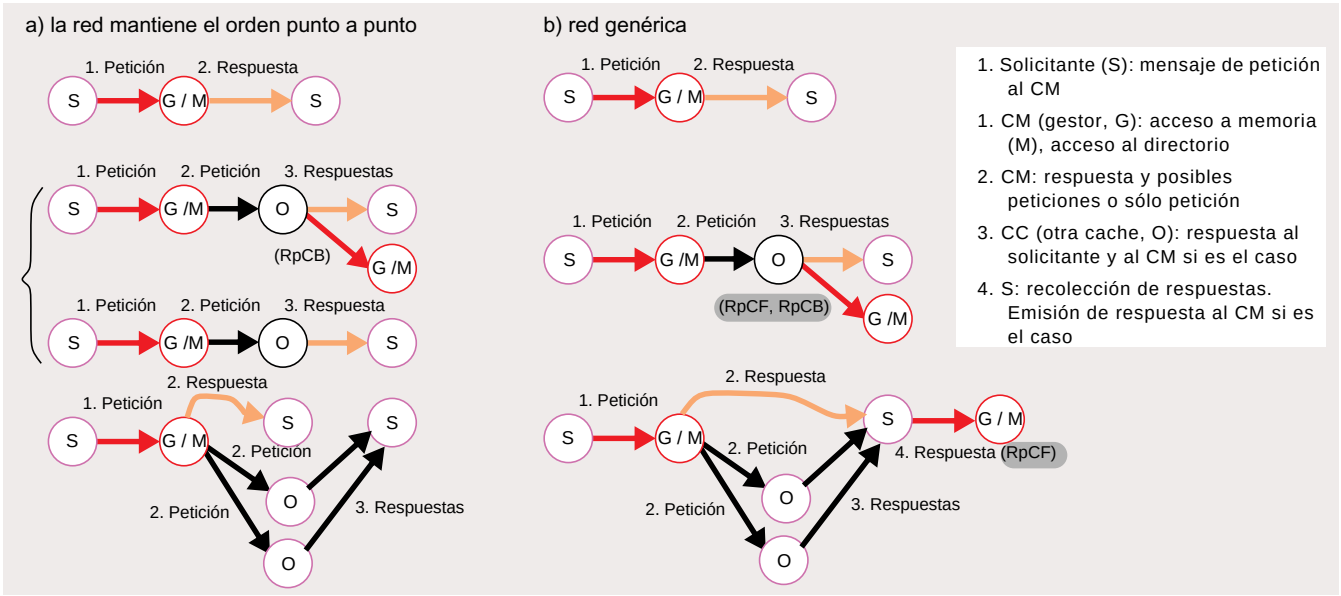


Figura 8.63 Flujos de mensajes en dos protocolo de directorio MLI. a) la red mantiene el orden punto a punto y b) red genérica. En sombreado se muestran las diferencias con el protocolo del apartado previo.

El CM da por consolidada o completada una transacción una vez emite el mensaje de respuesta o recibe la notificación del CC que tiene el bloque en exclusividad o del CC solicitante (casos segundo y tercero en la Figura 8.63)[30]. El CM utiliza la notificación para actualizar la memoria, si es el caso.

Notemos que mediante este protocolo un CM conoce, a falta de la recepción de la respuesta por el CC solicitante, cuándo todas las caches involucradas en la acción de coherencia han procesado la petición del CM. En estas condiciones, en el multiprocesador sólo existe una transacción en curso que hace referencia a un bloque y si es el caso, la respuesta de la transacción previa al mismo bloque.

Organización del multiprocesador

La organización del multiprocesador es la misma que en el protocolo previo. La diferencia es que las redes no mantienen orden entre un emisor y un receptor de mensajes (Figura 8.64)[31].

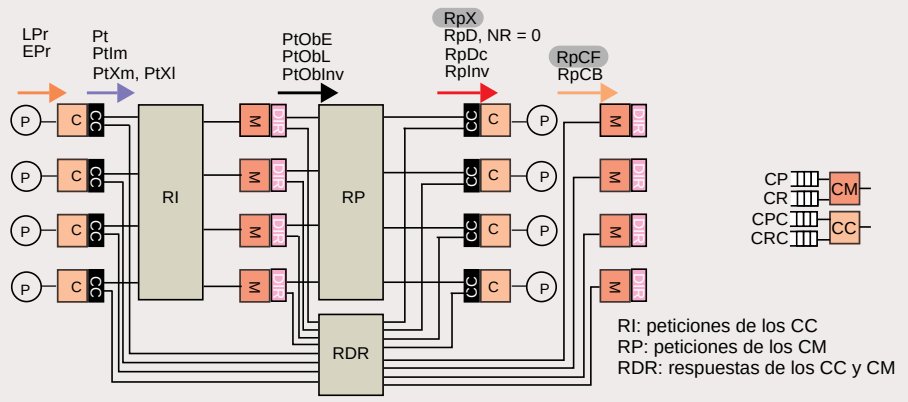


Figura 8.64 Red genérica. Organización del multiprocesador. En sombreado se muestran las diferencias básicas respecto del protocolo del apartado previo.

30. Notemos que la notificación al CM, en el caso de que una cache tenga el bloque en exclusividad, también la puede efectuar el CC solicitante. Sin embargo este diseño incrementa el número de pasos de la transacción.

31. Las redes pueden ser de tipo malla con encaminamiento adaptativo. Esto es, el encaminamiento entre un emisor y un receptor no está preestablecido.

Mensajes del protocolo

En la Figura 8.64 se muestra un esquema genérico de la transmisión de mensajes entre los CC y los CM y viceversa.

En la descripción del protocolo se distiguen los siguientes tipos de mensajes:

- Peticiones de un CC a los CM (Pt, PtIm, PtXm, PtXl)
- Peticiones de un CM a los CC (PtObL, PtObE, PtOInv)
- Respuestas de los CM a un CC (RpD, RpX). La respuesta RpD incluye el bloque y el número de respuestas que debe esperar (NR = K)
- Respuestas de un CC a un CC solicitante (RpDc, RpInv)
- Respuestas de un CC a un CM (RpCB, RpCF)

Los mensajes que se han añadido o modificado respecto del protocolo previo de este capítulo se muestran en la Tabla 8.5.

Controlador de coherencia Respuestas	Comentario
RpCF: notificación al CM de que los CC participantes en la acción de coherencia la han observado Id | bloque	El CC que tenía el bloque en exclusividad notifica al CM que ha emitido la respuesta al solicitante. El CC solicitante notifica al CM que ha recibido todas las respuestas.

Tabla 8.5 Repuesta de un CC a un CM.

A diferencia del protocolo previo la respuesta RpX se transmite por la red RDR.

Mensajes para ordenar las transacciones

En los protocolos descritos en este capítulo el gestor de un bloque (CM) conoce, o en algunos casos es informado, de cuándo su trabajo para procesar una petición ha finalizado. Sin embargo, este hecho no significa que la transacción haya consolidado respecto a todos los nodos involucrados en la acción de coherencia[32]. En estos protocolos hemos analizado la serialización de acciones de coherencia en un CC. En la Figura 8.66 se muestran dos ejemplos utilizando el protocolo descrito previamente en este capítulo, donde el receptor de las respuestas de los CC, participantes en una acción de

32. Recordemos que en los protocolos descritos un CM no procesa una petición (Pt, PtIm) a un bloque que está en un estado transitorio. Un CM inicia el procesado de una petición cuando el bloque está en un estado estable. El procesado de esta petición puede inducir que el CM genere una petición a un CC, cuya petición acaba de procesar el CM, el cual aún espera la respuesta o algunas respuestas a su petición.

coherencia, es el CC solicitante. Las secuencias de accesos a memoria que analizaremos se muestran en la Figura 8.65. En la parte izquierda de la Figura 8.66 se muestra la secuencia X y en la parte derecha la secuencia Y.

Orden de procesado en el CM		
X	**Y**	**Comentario**
P1 load t	P1 store t	X: no hay copias del bloque en las caches
P2 store t	P4 store t	Y: hay copia del bloque en las caches C2 y C3

Figura 8.65 Secuencias de accesos a memoria.

En los dos casos de la Figura 8.66 el CC1 recibe una petición del CM, inducida por una petición de otro CC, antes de la respuesta a su petición que el CM ha procesado previamente. El desorden es entre peticiones y respuestas del CM, debido a que se utilizan redes distintas para transmitirlas.

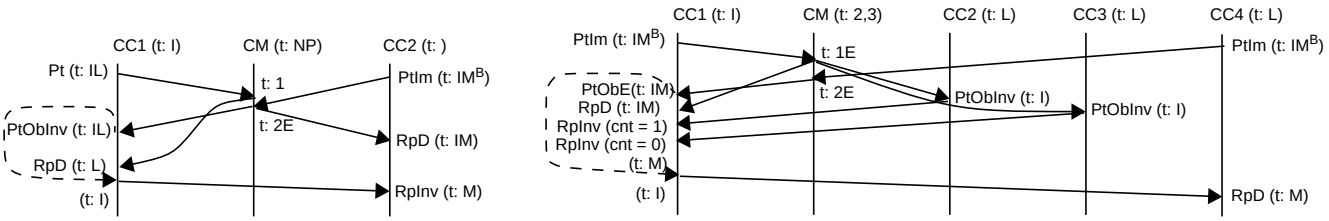


Figura 8.66 Serialización en el CC. Diagrama temporal simplificado.

En el protocolo descrito en este apartado la red de peticiones entre los CM y los CC no mantiene el orden punto a punto. Entonces, un CC puede recibir peticiones de un CM en un orden distinto al de emisión o procesado. Sin embargo, debido a la simplicidad del protocolo MLI veremos que no es posible que se produzca[33].

En el protocolo MLI un CC puede recibir la secuencia de peticiones PtObL y PtObInv a un bloque en estado M (Figura 8.67). Si el bloque está en el estado L sólo puede recibir la petición PtObInv.

En la Figura 8.67 la petición PtIm se queda encolada en la CP hasta que el CM recibe la respuesta a la petición que ha emitido previamente (estado transitorio). Notemos que hasta ese instante no dispone del bloque. Esta serialización en el CM garantiza que el CC2 reciba las peticiones PtObL y PtObInv en el

33. Si el protocolo dispone de un estado, distinto del M, desde el que suministra a otra cache es sencillo observar el desorden. Por ejemplo, un protocolo donde los bloques en cache disponen de un estado desde el que se sigue suministrando el bloque, después de que el bloque haya sido suministrado estando en el estado M. La memoria no está actualizada.

orden en el cual el CM ha procesado las peticiones que las inducen. El CM, para poder emitir la petición PtObinv al CC2, debe esperar a que éste responda a su petición anterior.

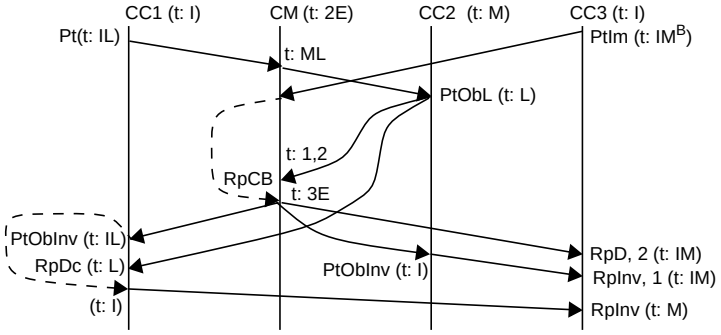


Figura 8.67 Serialización en las peticiones del CM que recibe un CC con un bloque en estado M.

Por otro lado, recordemos que la respuesta RpX del CM se transmite, en el protocolo previo del capítulo, por la red de peticiones de los CM a los CC. El objetivo es que la respuesta RpX no llegue antes al CC, que la petición que el CM ha emitido previamente al mismo CC. Como la red de peticiones no mantiene el orden punto a punto, la solución adoptada no es válida en la organización del multiprocesador de este apartado. Seguidamente analizamos el flujo de mensajes en varias situaciones.

En la parte izquierda de la Figura 8.68 se utiliza el protocolo previo de este capítulo, pero la respuesta RpX se transmite por la red de respuestas. En la figura se muestra el procesado en el CM de la petición PtXm de CC1, después de procesar la petición PtIm de CC2. La respuesta (RpX) puede adelantar a la petición (PtObE). Notemos que el CM procesa la petición de CC1 antes de que la transacción de CC2 haya finalizado[34]. Esto es, antes de que el CC1 haya emitido la respuesta al CC2. El CC1 debe suministrar el bloque, para responder a la petición PtObE, pero no dispone del mismo. El contenedor que almacenaba el bloque ha sido liberado al procesar el CC1 la respuesta RpX.

Para que no se produzca este adelantamiento el CM debe conocer cuándo el CC1 ha procesado la petición inducida por el CC2. Para ello se añade al protocolo un mensaje de respuesta del CC1 al CM (RpCF) cuando el CC procesa una petición PtObE (centro de la Figura 8.68 y centro de la Figura 8.63). El bloque en el directorio está en un estado transitorio (MM),

34. El estado del bloque en el directorio es estable. Estado M. El CM al procesar la petición PtXm detecta un cruce de peticiones, ya que el CC1 no está identificado en el VP.

mientras espera la respuesta del CC1. Una vez llega la respuesta, el CM al procesar la petición PtXm detecta un cruce de peticiones, ya que el CC1 no está identificado en el VP. El CM responde.

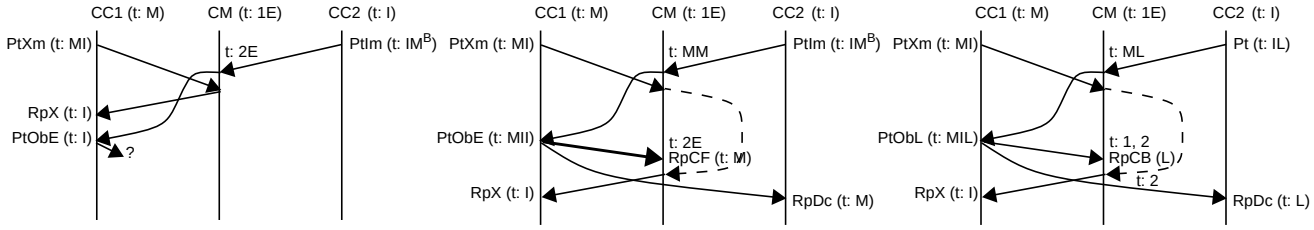


Figura 8.68 Una respuesta RpX adelanta a una petición del CM. La respuesta se corresponde con una petición PtXm. Mensaje de finalización del CC, que tiene el bloque en exclusividad, al CM para establecer orden.

Notemos que cuando la petición de CC2 es Pt no es necesario añadir este mensaje, ya que está en el protocolo de partida (parte derecha de la Figura 8.68 y centro de la Figura 8.63). El CM después de emitir la petición PtObL espera una respuesta del CC1. Durante esta espera el bloque está en el estado ML en el directorio. El CM al procesar la petición PtXm detecta un cruce de peticiones, ya que una petición PtXm no es esperada cuando el bloque está en el estado L. El CM responde y elimina de VP al CC1.

Cuando la petición es PtXI, y se utiliza el protocolo previo de este capítulo, también se puede producir el mismo comportamiento (parte izquierda de la Figura 8.69)[35 36]. Para establecer un orden de procesado en el CC1 es necesario que el solicitante (CC2) le notifique al CM que ha recibido todas las respuestas (parte inferior de la Figura 8.63 y parte derecha de la Figura 8.69). El CM procesa la petición PtXI una vez ha recibido el mensaje de notificación.

35. El CM al procesar la petición PtXI detecta un cruce de peticiones, ya que el CC1 no está identificado en el VP.

36. Procesar la petición RpX en el CC1 determina invalidar el contenedor. Posteriormente, el procesado de la petición PtObInv requiere suponer que el directorio es impreciso, lo cual no es el caso. Esto es, la expulsión de un bloque en el estado L no ha sido notificada al directorio.

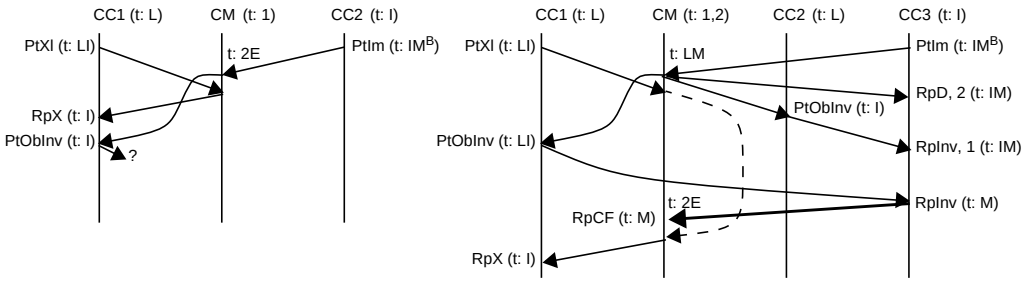


Figura 8.69 Una respuesta RpX adelanta a una petición del CM. La respuesta se corresponde con una petición PtXl. Mensaje de finalización del CC solicitante al CM para establecer orden.

Estados y transiciones

Cuando un CC tiene el bloque en exclusividad y el CM procesa una transacción de lectura hay que actualizar la memoria. En estas condiciones, el CM debe esperar a que responda el CC, con el bloque, antes de procesar otra transacción al mismo bloque. Para identificar la espera de una respuesta de un CC se utiliza un estado transitorio en el CM.

Cuando un CM procesa una petición de exclusividad hay que esperar la respuesta de finalización de la transacción. Esta respuesta la emite el CC que tiene el bloque en exclusividad o el CC solicitante. Para identificar las esperas se utilizan estados transitorios.

El directorio se utiliza para detectar accesos concurrentes al mismo bloque y gestionarlos (ventana de vulnerabilidad). Esto es, cuando la petición en la cabeza de la CP accede a un bloque en un estado transitorio, el CM efectúa la detección. La gestión utilizada en el protocolo de este apartado es bloquear el procesado de esta petición hasta que el estado del bloque sea estable.

Directorio. Los estados estables de un bloque en el directorio son los mismos que en el Capítulo 7 (Figura 8.70).

En el directorio se utilizan tres estados transitorios para: a) esperar a que el CC que tiene el bloque en exclusividad suministre el bloque al CM o notifique al CM el suministro del bloque al CC solicitante y b) esperar a que el CC solicitante notifique al CM la recepción de todas las respuestas de los CC participantes en la acción de coherencia.

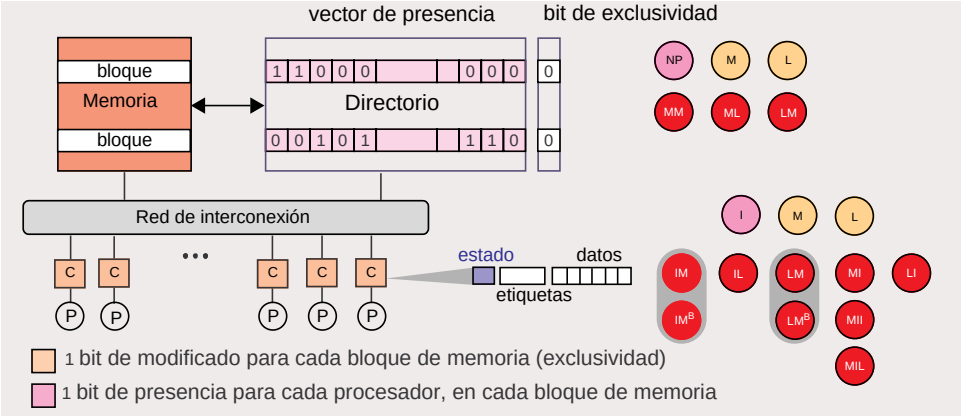


Figura 8.70 Red genérica. Estados de un bloque en el directorio y en un contenedor de cache.

Cache. Los estados estables de un bloque en una cache son los mismos que en el Capítulo 7 (Figura 8.70). Para identificar la espera de la respuesta del CM y las respuestas de los CC, a una petición del CC, se utilizan siete estados transitorios (IM^B, IM, IL, LM^B, LM, MI, LI).

Para describir el protocolo se utilizan las dos peticiones del procesador que requieren acceder al directorio (fallo en una instrucción load, store o una instrucción store que accede a un bloque sin permiso de exclusividad) y las dos posibilidades de ubicación del bloque solicitado, en memoria o en una cache. Posteriormente se detallan las transiciones entre estados en una expulsión de un bloque de cache.

En la descripción se mostrarán las transiciones entre estados de un bloque en el CC que efectúa la petición (agente procesador), en el CM y en otros CC (agente observador).

Fallo en lectura

El flujo de mensajes y transiciones entre estados en un CC y en un CM son idénticos a los descritos en el protocolo previo de este capítulo.

Fallo de escritura o petición de exclusividad

Memoria tiene el bloque actualizado. El flujo de mensajes y las transiciones entre estados son las mismas que en el protocolo previo de este capítulo.

Hay copia del bloque en otras caches. En la parte derecha de la Figura 8.71 se muestra el flujo de mensajes en un fallo de escritura cuando otras caches tiene copia del bloque. El CM, utilizando la información de estado

del bloque en el directorio, determina los CC a los que es necesario enviar una petición de invalidación. El CM responde al CC solicitante con el bloque y el número de respuestas que debe recolectar (RpD, Nr = K) y emite un mensaje de invalidación (PtObInv) a cada uno de los CC involucrados en la acción de coherencia. Posterormente modifica el VP y el BE. El CC solicitante, cuando ha recolectado todas las respuestas, emite un mensaje de notificación al CM (RpCF).

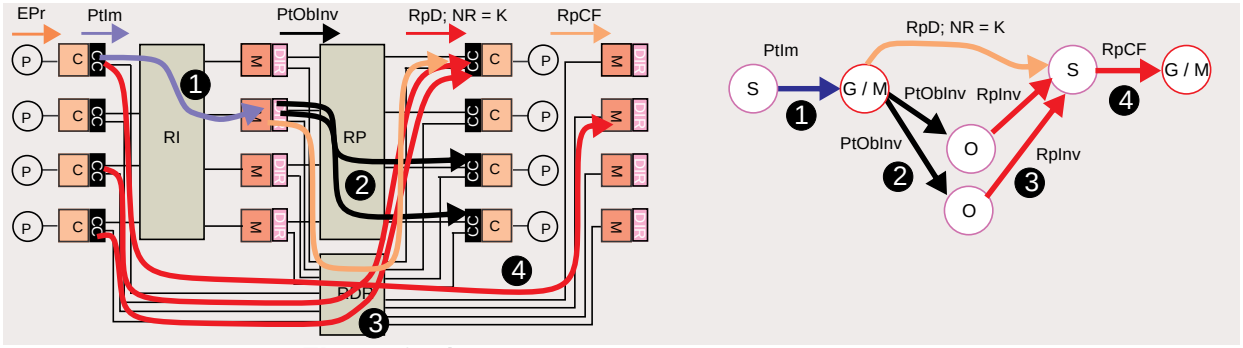


Figura 8.71 Red genérica. Flujo de mensajes en un fallo de escritura y hay copias del bloque en algunas caches.

Una cache tiene el bloque en exclusividad. En la Figura 8.72 se muestra el flujo de mensajes en un fallo de escritura cuando otra cache tiene el bloque en exclusividad. El CM, emite un mensaje, de suministro del bloque e invalidación (PtObE), a la cache que tiene el bloque en exclusividad. El CC correspondiente responde al CC solicitante con el bloque, invalida el bloque en su cache y envia un mensaje de notificación al CM (RpCF). El CM actualiza el VP para identificar la cache que ahora tiene el bloque en exclusividad.

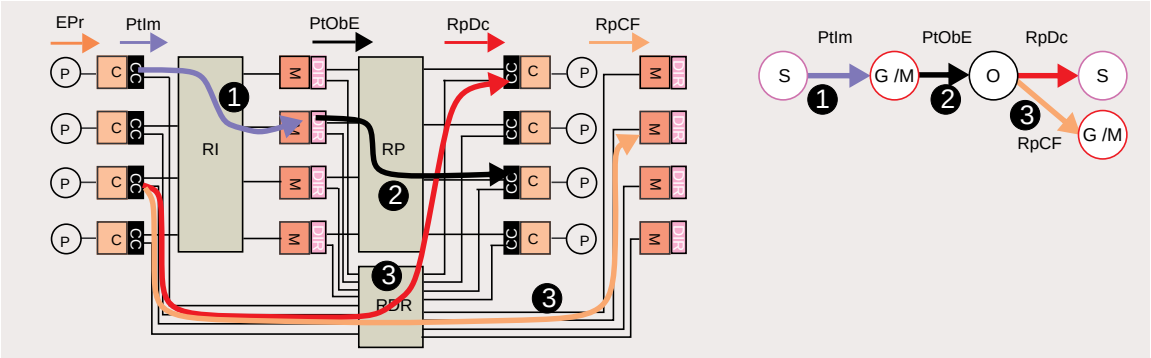


Figura 8.72 Red genérica. Flujo de mensajes en un fallo de escritura cuando una cache tiene el bloque en exclusividad.

Diagramas de transiciones entre estados. En la Figura 8.73 se muestran las transiciones entre estados en la cache del CC que efectúa la petición, el directorio y los otros CC. El CC del solicitante establece un estado transitorio (IM^B o LM^B), en función del estado estable inicial, esperando la respuesta o respuestas.

En el directorio el estado del bloque pasa de L a M, de NP a M o de M a M, el VP se actualiza en consecuencia y el BE se activa. Además, el CM emite mensajes de respuesta (RpD, NR = K) y de petición de invalidación (PtObInv) o de petición de observación de escritura (PtObE).

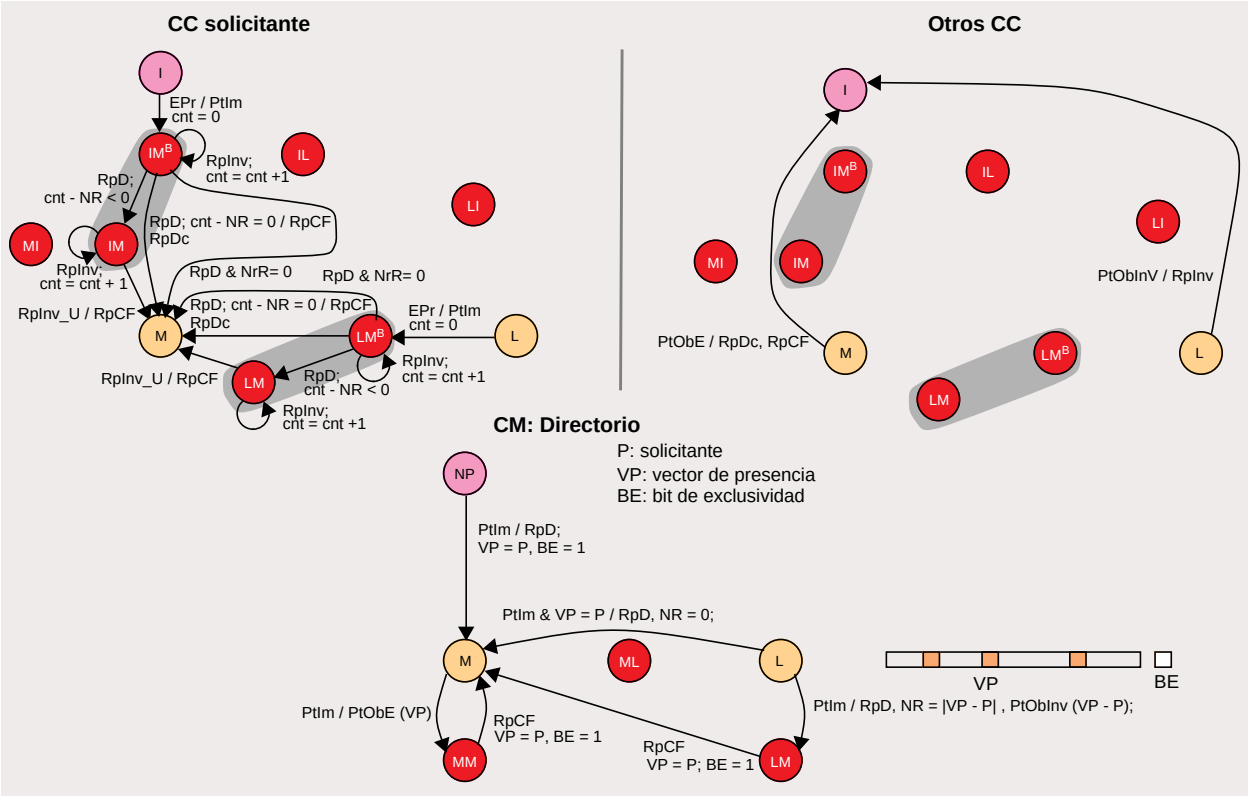


Figura 8.73 Red genérica. Transiciones entre estados en un fallo de escritura o solicitud de exclusividad. RpInv_U indica cnt = 1 y recepción de un mensaje RpInv.

Cuando un CC emite una petición PtIm, estando el bloque en estado L en la cache, el bit de presencia del CC está activado en el VP. Por tanto, hay que excluirlo de la lista de CC que reciben una petición PtObInv (VP - P). El CC, al recibir la respuesta RpD, NR = K, siempre almacena el bloque recibido en el contenedor correspondiente, tanto si el estado estable inicial es L como I

(centro de la Figura 8.74). El CC solicitante después de recibir todas las respuestas de los CC, involucrados en la acción de coherencia, emite una notificación al CM. Un caso particular es que no haya copias del bloque o el CC solicitante tenga la única copia (parte izquierda de la Figura 8.74).

Un CC que tiene copia del bloque, al recibir la petición PtObInv invalida la copia del bloque y responden al CC solicitante (RpInv).

Cuando un CC tiene el bloque en exclusividad, el CM, al procesar la petición PtIm, emite una petición PtObE. El CC que tiene el bloque en exclusividad suministra el bloque, invalida el contenedor que almacena el bloque y emite una notificación al CM (RpDc, RpCF parte derecha de la Figura 8.74).

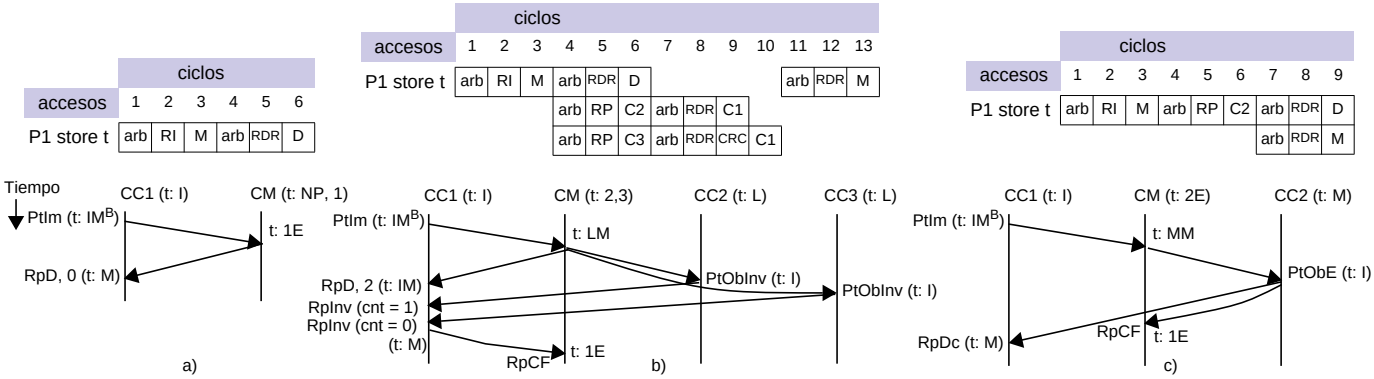


Figura 8.74 Red genérica. Diagrama temporal: a) estado NP o L en el directorio y VP = P, b) estado L en el directorio y c) estado M en el directorio. El acrómino CRC indica cola de respuestas en un CC.

Expulsión

En una expulsión de un bloque el flujo de mensajes y las transiciones entre estados son las mismas que en el protocolo previo de este capítulo.

Diagrama completo de estados y transiciones

En las Figura 8.75 y Figura 8.76 se muestran, respectivamente, los diagramas de transiciones entre estados de un bloque en cache y en el directorio. En el primer diagrama están incluidas las peticiones del procesador que no requieren iniciar transacciones explícitas de coherencia. En el CC se distinguen, en diagramas separados, las transiciones iniciadas por el agente procesador (CC solicitante) y el agente observador (otros CC).

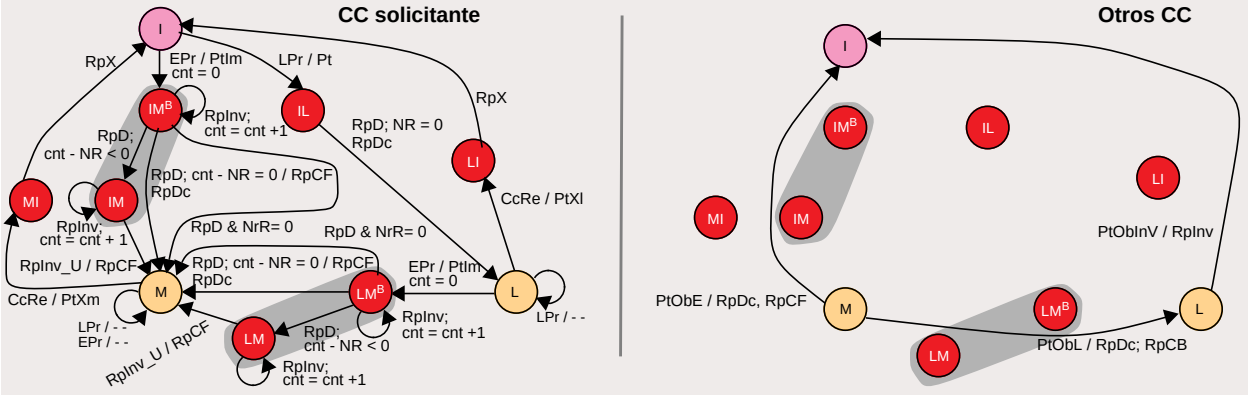


Figura 8.75 Red genérica. Estados y transiciones entre estados de un bloque en una cache.

En el diagrama del CM (Figura 8.76) hay que distinguir dos casos al procesar la petición PtXI de un CC (PtXI, PtXI_U). El estado final depende de si el CC, que efectúa la petición, es el único que está en el vector de presencia.

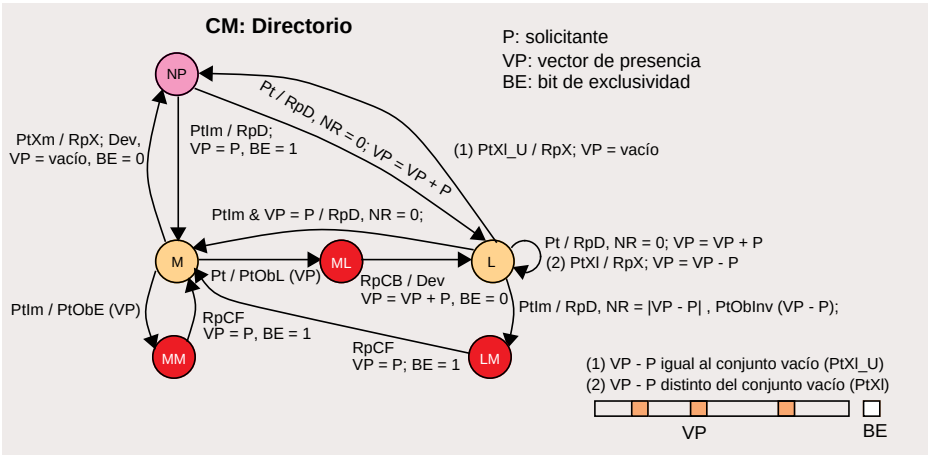


Figura 8.76 Red genérica. Estados y transiciones entre estados de un bloque en el directorio.

Gestión de cruces de peticiones

Controlador de memoria

Los estados transitorios representan una ventana de vulnerabiliad. Durante la estancia de un bloque en estos estados no se procesan peticiones. A diferencia del protocolo previo de este capítulo, en un estado transitorio no se procesan peticiones de expulsión (Figura 8.77).

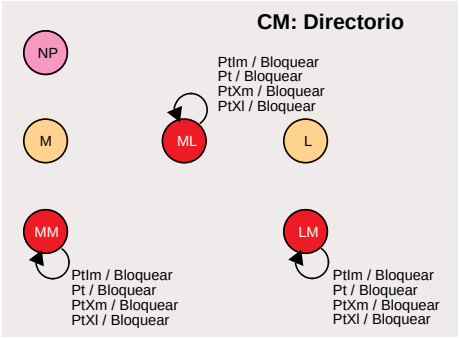


Figura 8.77 Red genérica. Bloqueo del procesado de peticiones en el estado transitorio del directorio.

Los cruces que se infieren en el directorio son los mismos que en los protocolos previos. En Figura 8.78 se muestran todos los cruces de peticiones.

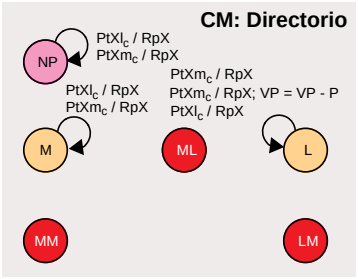


Figura 8.78 Red genérica. Cruce de peticiones en el directorio.

Controlador de coherencia

Los conjuntos de cruces de peticiones en un CC son los mismos que los identificados en el protocolo descrito previamente en este capítulo, donde la red de peticiones entre los CM y los CC mantiene el orden punto a punto (Figura 8.79).

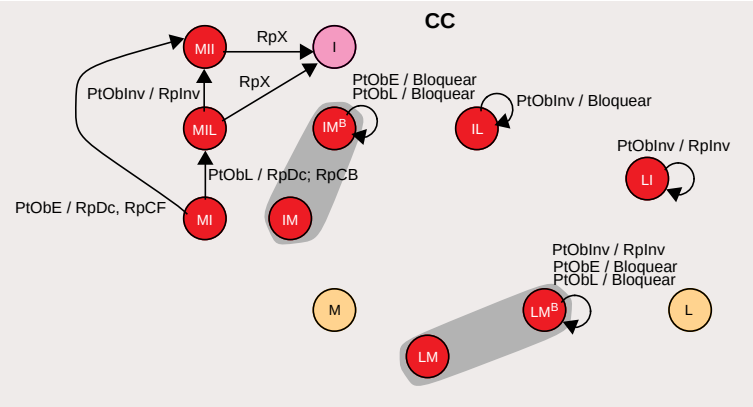


Figura 8.79 Red genérica. Diagramas de transiciones en un CC cuando se produce un cruce entre peticiones a un bloque.

Notemos que en los estados IM y LM no se puede producir un cruce de peticiones, ya que es este CC el que notifica al CM que ha finalizado la transacción (RpCF, Figura 8.63). El CM no procesa ninguna nueva transacción al bloque hasta que recibe RpCF. Por otro lado, en los estado IM^B y LM^B sólo puede haber cruces de peticiones si aún no se han recibido respuestas (RpInv). Si este es el caso, el CM está en un estado transitorio esperando la respuesta RpCF del CC.

APENDICE A: TABLAS DE TRANSICIONES ENTRE ESTADOS

Controlador de coherencia.

Se muestran en formato tabla los estados y las transiciones entre estados de un bloque en una cache. Las casillas que no contienen información indican un error. En un estado determinado no puede llegar el evento que determina la casilla correspondiente en el cruce.

Controlador de memoria.

Se muestran en formato tabla los estados y transiciones entre estados de un bloque en el directorio. En el evento PtXl se distingue el caso de que el CC sea el único que está en el vector de presencia o haya más CC.

Para identificar un cruce, se comprueba si el CC solicitante están en el vector de presencia o no. Además se comprueba si la petición recibida es esperada en el estado actual.

El CM recolecta las respuestas de invalidación

Peticiones y respuestas del CM se encaminan por la misma red

Controlador de coherencia

			Eventos del procesador y reemplazo			Eventos externos (respuestas y peticiones			
			LPr	EPr	CcRe	RpD	RpX	PtObL	PtObE
Estados	Estables	I	Pt; IL	PtIm; IM					
		L	- -; L	PtIm; LM	PtXl; LI				- -; I
		M	- - ; M	- - ; M	PtXm; MI			RpDc: L	RpDc; I
	transitorios	IL				- -; L			
		IM				- -; M			
		LM				- -; M			RpInv; LM
		LI					- -; I		RpInv; LI
		MI					- - ; I	RpDc; MIL	RpDc; MII
		MII					- - ; I		
		MIL					- - ; I		RpInv; MII

Figura 8.80 Tabla de estados y transiciones en un bloque de cache. Las casillas con fondo blanco indican cruces de peticiones.

Controlador de memoria (cuando {VP - P} está vacio no se emiten PtObE)

			Eventos del controlador de coherencia										
			Pt	PtIm	PtXl			PtXm			RpDc	RpInv	
					VP = P	VP ≠ P P ∈ VP	VP ≠ P P ∉ VP	VP = P	VP ≠ P P ∉ VP	VP ≠ P P ∈ VP		cnt > 1	cnt = 1
Estados	Estables	NP	RpD; L, VP = P	RpD; M VP = P, BE = 1									
		L	RpD; L, VP = VP + P	PtObE {VP - P}, RpD; M, VP = P, BE = 1	RpX; NP, VP = vacío	RpX; L VP = VP - P	RpX; L		RpX; L	RpX; L VP = VP - P			
		M	PtObL; ML	PtObE {VP}; MM			RpX; M	RpX; NP, VP = vacío, BE = 0	RpX; M				
	transitorios	ML	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	RpD, Dev; L VP = VP + P, BE = 0		
		MM	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	RpD; M VP = P, BE = 1		
		LM	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo		cnt = cnt -1; LM	RpD; M VP = P, BE =1

Figura 8.81 Tabla de estados y transiciones de un bloque en el directorio. Las casillas con fondo blanco indican cruces de peticiones.

Peticiones y respuestas del CM se encaminan por redes distintas

Controlador de coherencia

Estados			Eventos del procesador y reemplazo			Eventos externos (respuestas y peticiones				
			LPr	EPr	CcRe	RpD	RpX	PtObL	PtObE	PtObInv
	Estables	I	Pt; IL	PtIm; IM						
		L	- - ; L	PtIm; LM	PtXl; LI				- -; I	
		M	- - ; M	- -; M	PtXm; MI			RpDc: L	RpDc; I	
	transitorios	IL				- -; L				Bloqueo
		IM				- -; M		Bloqueo	Bloqueo	
		LM				- -; M		Bloqueo	Bloqueo	RpInv; LM
		LI					- -; I			RpInv; LI
		MI					- - ; I	RpDc; MIL	RpDc; MII	
		MII					- - ; I			
		MIL					- - ; I			RpInv; MII

Figura 8.82 Tabla de estados y transiciones en un bloque de cache. Las casillas con fondo blanco indican cruces de peticiones.

Controlador de memoria (cuando {VP - P} está vacio no se emiten PtObE)

Estados			Eventos del controlador de coherencia										
			Pt	PtIm	PtXl			PtXm			RpDc	RpInv	
					VP = P	VP ≠ P P ∈ VP	VP ≠ P P ∉ VP	VP = P	VP ≠ P P ∉ VP	VP ≠ P P ∈ VP		cnt > 1	cnt = 1
	Estables	NP	RpD; L, VP = P	RpD; M VP = P, BE = 1			RpX; NP		RpX; NP				
		L	RpD; L, VP = VP + P	PtObInv {VP - P}, RpD; M, VP = P, BE = 1	RpX; NP, VP = vacío	RpX; L VP = VP - P	RpX; L		RpX; L	RpX; L VP = VP - P			
		M	PtObL; ML	PtObE {VP}; MM			RpX; M	RpX; NP, VP = vacío, BE = 0	RpX; M				
	transitorios	ML	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	RpD, Dev; L VP = VP + P, BE = 0		
		MM	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	RpD; M VP = P, BE = 1		
		LM	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo	Bloqueo		cnt = cnt -1; LM	RpD; M VP = P, BE =1

Figura 8.83 Tabla de estados y transiciones de un bloque en el directorio. Las casillas con fondo blanco indican cruces de peticiones. El cruce en NP, de las peticiones PtXl y PtXm, se identifica en la columna $P \notin VP$ y $VP \neq P$

El solicitante recolecta las respuestas de invalidación

Controlador de coherencia

Estados			Eventos del procesador y reemplazo			Eventos externos (respuestas y peticiones								
			LPr	EPr	CcRe	RpD, NR	RpD, cnt - NR = 0	RpDc	RpInv cnt > 1	RpInv cnt = 1	RpX	PtObL	PtObE	PtObInv
	Estables	I	Pt; IL	PtIm; IM^B										
		L	- -; L	PtIm; LM^B	PtXl; LI								- -; I	
		M	- -; M	exclu; M	PtXm; MI							RpDc: L	RpDc; I	
	transitorios	IL				- -; L								Bloqueo
		IM^B				cnt - NR < 0; IM	- -; M	- -; M				Bloqueo	Bloqueo	
		IM							cnt = cnt +1; IM	- -; M		Bloqueo	Bloqueo	
		LM^B				cnt - NR < 0; IM	- -; M	- -; M				Bloqueo	Bloqueo	RpInv; LM
		LM							cnt = cnt +1; IM	- -; M		Bloqueo	Bloqueo	
		LI									- -; I			RpInv; LI
		MI									- - ; I	RpDc; MIL	RpDc; MII	
		MII									- - ; I			
		MIL									- - ; I			RpInv; MII

Figura 8.84 Tabla de estados y transiciones en un bloque de cache. Las casillas con fondo blanco indican cruces de peticiones.

Controlador de memoria (cuando {VP - P} está vacio no se emiten PtObInv)

Estados			Eventos del controlador de coherencia								
			Pt	PtIm	PtXl			PtXm			RpCB
					VP = P	VP ≠ P P ∈ VP	VP ≠ P P ∉ VP	VP = P	VP ≠ P P ∉ VP	VP ≠ P P ∈ VP	
	Estables	NP	RpD; L, VP = P	RpD; M VP = P, BE = 1			RpX; NP		RpX; NP		
		L	RpD; L, VP = VP + P	PtObInv {VP - P}, RpD, NR; M, VP = P, BE = 1	RpX; NP, VP = vacío	RpX; L VP = VP - P	RpX; L		RpX; L	RpX; L VP = VP - P	
		M	PtObL; ML	PtObE {VP}; MM			RpX; M	RpX; NP, VP = vacío, BE = 0	RpX; M		
	transitorios	ML	Bloqueo	Bloqueo			RpX; ML		RpX; ML	RpX; ML VP = VP - P	RpD, Dev; L VP = VP + P, BE = 0

Figura 8.85 Tabla de estados y transiciones de un bloque en el directorio. Las casillas con fondo blanco indican cruces de peticiones. El cruce en NP, de las peticiones PtXl y PtXm, se identifica en la columna $P \notin VP$ y $VP \neq P$.

Red genérica

Controlador de coherencia

			Eventos del procesador y reemplazo			Eventos externos (respuestas y peticiones								
			LPr	EPr	CcRe	RpD, NR	RpD, cnt - NR = 0	RpDc	RpInv cnt > 1	RpInv cnt = 1	RpX	PtObL	PtObE	PtObInv
Estados	Estables	I	Pt; IL	PtIm; IM^B										
		L	- -; L	PtIm; LM^B	PtXl; LI								- -; I	
		M	- -; M	exclu; M	PtXm; MI							RpDc: L	RpDc; I	
	transitorios	IL				- -; L								Bloqueo
		IM^B				cnt - NR < 0; IM	RpCF; M	RpCF; M				Bloqueo	Bloqueo	
		IM							cnt = cnt +1; IM	RpCF; M		Bloqueo	Bloqueo	
		LM^B				cnt - NR < 0; IM	RpCF; M	RpCF; M				Bloqueo	Bloqueo	RpInv; LM
		LM							cnt = cnt +1; IM	RpCF; M		Bloqueo	Bloqueo	
		LI									- -; I			RpInv; LI
		MI									- - ; I	RpDc; MIL	RpDc; MII	
		MII									- - ; I			
		MIL									- - ; I			RpInv; MII

Figura 8.86 Tabla de estados y transiciones en un bloque de cache. Las casillas con fondo blanco indican cruces de peticiones.

Controlador de memoria (cuando {VP - P} está vacio no se emiten PtObInv)

			Eventos del controlador de coherencia									
			Pt	PtIm	PtXI			PtXm			RpCB	RpCF
					VP = P	VP ≠ P P ∈ VP	VP ≠ P P ∉ VP	VP = P	VP ≠ P P ∉ VP	VP ≠ P P ∈ VP		
Estados	Estables	NP	RpD; L, VP = P	RpD; M VP = P, BE = 1			RpX; NP		RpX; NP			
		L	RpD; L, VP = VP + P	PtObInv {VP - P}, RpD, NR; M o LM, VP = P, BE = 1	RpX; NP, VP = vacío	RpX; L VP = VP - P	RpX; L		RpX; L	RpX; L VP = VP - P		
		M	PtObL; ML	PtObE {VP}; MM			RpX; M	RpX; NP, VP = vacío, BE = 0	RpX; M			
	transitorios	ML	Bloqueo	Bloqueo			Bloqueo		Bloqueo	Bloqueo	RpD, Dev; L VP = VP + P, BE = 0	
		MM	Bloqueo	Bloqueo			Bloqueo		Bloqueo	Bloqueo		- - / M VP = P, BE =1
		LM	Bloqueo	Bloqueo			Bloqueo		Bloqueo	Bloqueo		- - / M VP = P, BE =1

Figura 8.87 Tabla de estados y transiciones de un bloque en el directorio. Las casillas con fondo blanco indican cruces de peticiones. El cruce en NP, de las peticiones PtXl y PtXm, se identifica en la columna $P \notin VP$ y $VP \neq P$

EJEMPLOS

Recepción de respuestas de los CC en el CM. Peticiones y respuestas del CM se encaminan por la misma red.

En el multiprocesador se ejecuta la siguiente secuencia de accesos a memoria.

accesos	accesos
1. P1 load t	4. P1 load w
1. P2 store u	4. P3 load w
2. P1 load u	5. P2 store w
2. P3 load u	5. P1 store w
3. P1 load w	
3. P2 store u	

Pregunta 1: *Represente mediante un diagrama temporal la secuencia de accesos previa. Las redes no disponen de un arbitraje centralizado. Cuando llegan varias peticiones o respuestas a un componente, el procesado de las mismas se serializa. Las peticiones que no se procesan se esperan en la cola correspondiente.*

Respuesta: El primer par de accesos a memoria son debidos a un fallo de lectura y a un fallo de escritura. El estado transitorio de los bloques al emitir las peticiones es IL en la cache C1 (Pt) e IM en la cache C2 (PtIm). Las dos peticiones llegan a la par al CM. La transacción del CC1 se procesa y la transacción del CC2 se espera en la CP. En las dos transacciones el bloque lo suministra la memoria. El estados de los bloques al finalizar las transacciones es L en la cache C1 y M en la cache C2. En el directorio ha sido activado el BE del bloque que contiene la variable u.

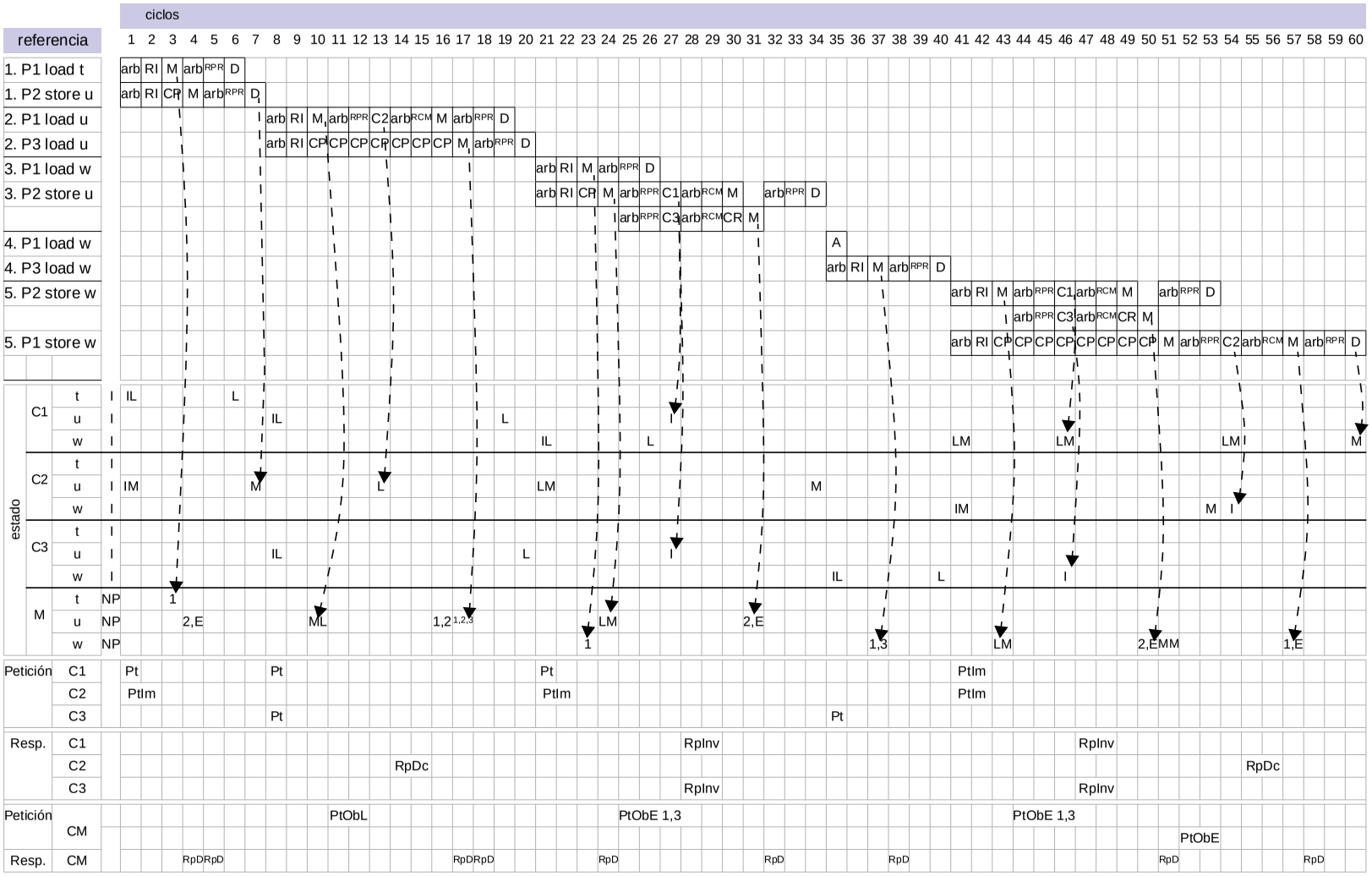
El segundo par de accesos a memoria son fallos de lectura. El CM procesa en primer lugar el mensaje emitido por CC1. El mensaje del CC3 se almacena en la CP. El procesado del mensaje del CC1 requiere que el CM solicite el bloque a la cache C2, mediante una petición de observación de lectura (PtObL). El segundo fallo hace referencia al mismo bloque. Como memoria ha sido actualizada en la transacción previa, el CM suministra directamente el bloque. Al finalizar las dos transacciones el VP del bloque identifica que las caches C1, C2 y C3 tienen copia del bloque.

El tercer grupo de accesos a memoria requiere dos transacciones. La primera petición es de lectura (Pt) y el bloque lo suministra la memoria. La segunda petición es de lectura con intención de modificación (PtIm). El CM emite

peticiones de observación de escritura del bloque (PtObE) a las caches C1 y C3, para invalidar las copias del bloque. El CM espera las respuestas del CC1 y del CC3. Las respuestas llegan a la par y la respuesta del CC3 se almacena en la CR. Una vez el CM ha recibido las respuestas, a las peticiones de invalidación, responde al CC2. El estado estable del bloque, al finalizar la transacción, es M en la cache C2 e I en las caches C1 y C3.

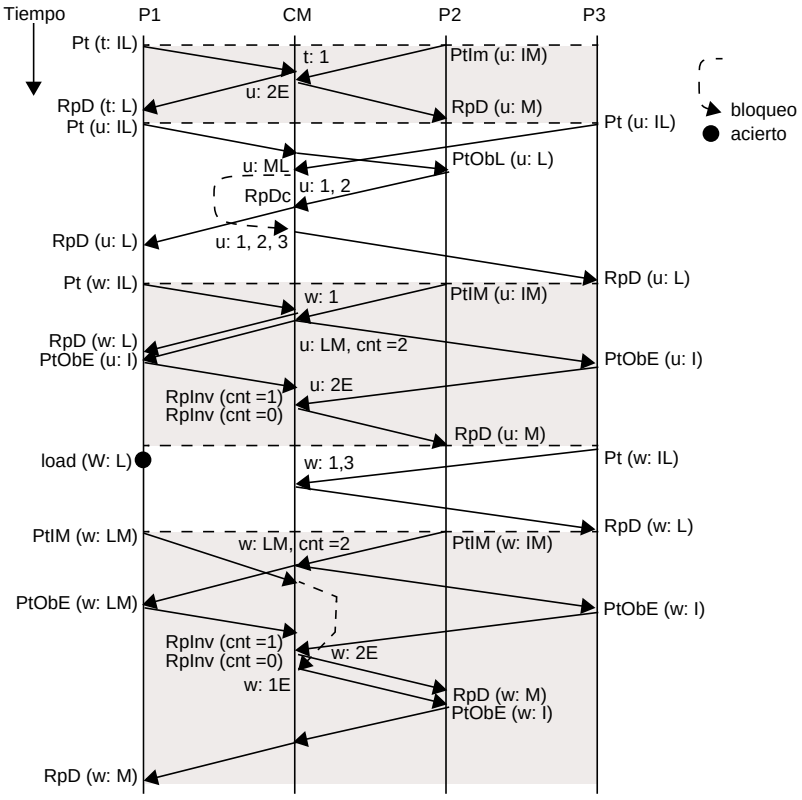
El cuarto par de accesos a memoria requiere sólo una transacción. El primer acceso a memoria efectuado por P1 es un acierto en cache. El segundo acceso a memoria, efectuado por P3, accede al mismo bloque y es un fallo. Al finalizar la transacción, el estado del bloque que contiene la variable w es L en las caches C1 y C3.

El último grupo de accesos a memoria son dos escrituras. Ninguna de las caches accedidas tiene el bloque en exclusividad. La cache C2 no tiene copia del bloque y la cache C1 tiene una copia en el estado L. La petición del CC2 requiere que el CM emita una petición de observación de escritura al CC1 y al CC3, para que invaliden el bloque. El CM espera las respuestas del CC1 y del CC3. Finalmente, el CM emite una respuesta al CC2. Durante este lapso de tiempo la petición del CC1 ha estado almacenada en la CP. Al procesar el CM la petición de CC1, emite un mensaje de petición de observación de escritura al CC2, el cual suministra el bloque al CM. Para finalizar la transacción, el CM suministra el bloque a CC1. Al finalizar las dos transacciones, el VP del bloque en el directorio indica que la cache C1 tiene copia del bloque en exclusividad.


ciclos
referencia
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
1. P1 load t
1. P2 store u
2. P1 load u
2. P3 load u
3. P1 load w
3. P2 store u
4. P1 load w
4. P3 load w
5. P2 store w
5. P1 store w
estado
C1
C2
C3
M
t
u
w
Petición
Resp.
CM
Pt
Ptlm
RpDc
RplnV
PtObL
PtObE 1,3
PtObE
RpD

Pregunta 2: *Represente mediante un diagrama temporal simplificado la secuencia de accesos previa.*

Respuesta: El bloqueo del procesado de una petición en el CM se indica mediante una línea curva a trazos.



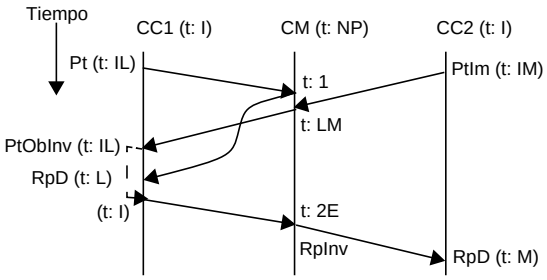
Observemos el bloqueo del procesado de peticiones en el CM en el segundo y último grupo de peticiones.

Recepción de respuestas de los CC en el CM. Peticiones y respuestas del CM se encaminan por redes distintas

Utilice los ordenes de procesado de accesos a memoria en el CM mostrados en la Figura 8.32. Suponga que la respuesta del CM al CC1 se retrasa del orden de 5 ciclos en la RMC.

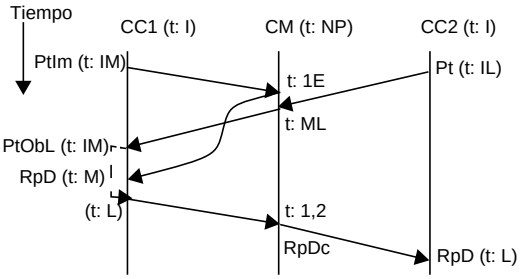
Pregunta 1: *Muestre un diagrama temporal y un diagrama temporal simplificado para cada uno de los órdenes de procesado en el CM.*

Respuesta: En la figura se muestra el diagrama correspondiente al orden I.



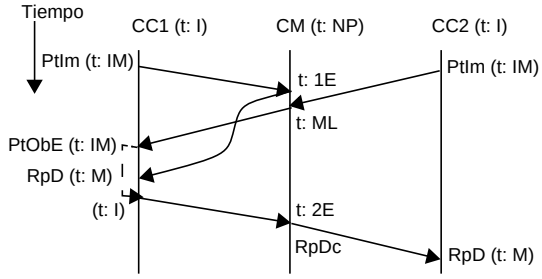
accesos	ciclos																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
P1 load t	arb	RI	M	arb	RMC	RMC	RMC	RMC	RMC	D							
P2 store t		arb	RI	M	arb	RP	CPC	CPC	CPC	CPC	C1	arb	RCM	M	arb	RMC	D

En la figura se muestra el diagrama correspondiente al orden J.



accesos	ciclos																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
P1 store t	arb	RI	M	arb	RMC	RMC	RMC	RMC	RMC	D							
P2 load t		arb	RI	M	arb	RP	CPC	CPC	CPC	CPC	C1	arb	RCM	M	arb	RMC	D

En la figura se muestra el diagrama correspondiente al orden K.

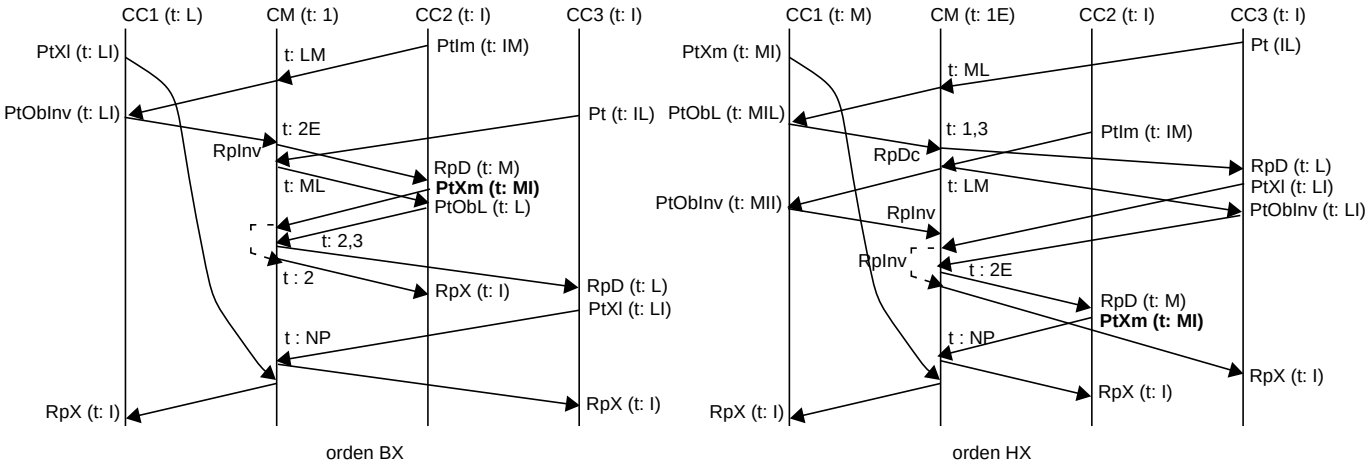


accesos	ciclos																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
P1 store t	arb	RI	M	arb	RMC	RMC	RMC	RMC	RMC	D							
P2 store t		arb	RI	M	arb	RP	CPC	CPC	CPC	CPC	C1	arb	RCM	M	arb	RMC	D

Utilice los ordenes de procesado de accesos a memoria en el CM BX (1ª columna) y HX (1ª columna) mostrados en la Figura 8.33.

Pregunta 2: *Muestre un diagrama temporal simplificado para cada uno de ellos. Debe observarse el procesado de la petición de P1 estando el bloque en el directorio en el estado NP.*

Respuesta: En la parte izquierda de la figura se muestra el diagrama correspondiente al orden BX (1ª columna). En la parte derecha de la figura se muestra el diagrama correspondiente al orden HX (1ª columna).



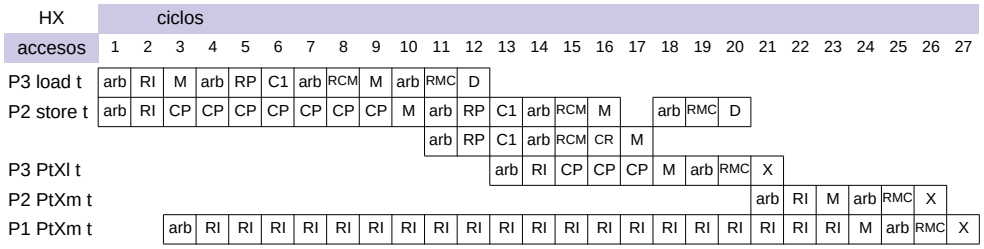
Observemos, que en un CC, una expulsión no se representa hasta que se ha recibido el bloque de la petición previa al mismo bloque.

Pregunta 3: *Muestre un diagrama temporal para cada uno de los órdenes de procesado en el CM. Debe observarse el procesado de la petición de P1 estando el bloque en el directorio en el estado NP. Para representar un retardo en la transmisión de un mensaje replique en ciclos consecutivos el acrónimo de la red correspondiente.*

Respuesta: En la siguiente figura se muestra el diagrama correspondiente al orden BX (1ª columna).

BX	ciclos																										
accesos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
P2 store t	arb	RI	M	arb	RP	C1	arb	RCM	M	arb	RMC	D															
P3 load t	arb	RI	CP	CP	CP	CP	CP	CP	CP	M	arb	RP	C2	arb	RCM	M	arb	RMC	D								
P2 PtXm t													arb	RI	CP	CP	M	arb	RMC	X							
P3 PtXl t																				arb	RI	M	arb	RMC	X		
P1 PtXl t	arb	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	RI	M	arb	RMC	X	

En la siguiente figura se muestra el diagrama correspondiente al orden HX (1ª columna).



En el multiprocesador se ejecuta la siguiente secuencia de accesos a memoria.

accesos	accesos
1. P1 load t	4. P1 load w
1. P2 store u	4. P3 load w
2. P1 load u	5. P2 store w
2. P3 load u	5. P1 store w
3. P1 load w	
3. P2 store u	

Pregunta 4: *Represente mediante un diagrama temporal la secuencia de accesos previa. Suponga que la respuesta del CM al CC2 en el quinto grupo de accesos a memoria experimenta un retardo total de 5 ciclos en la RMC.*

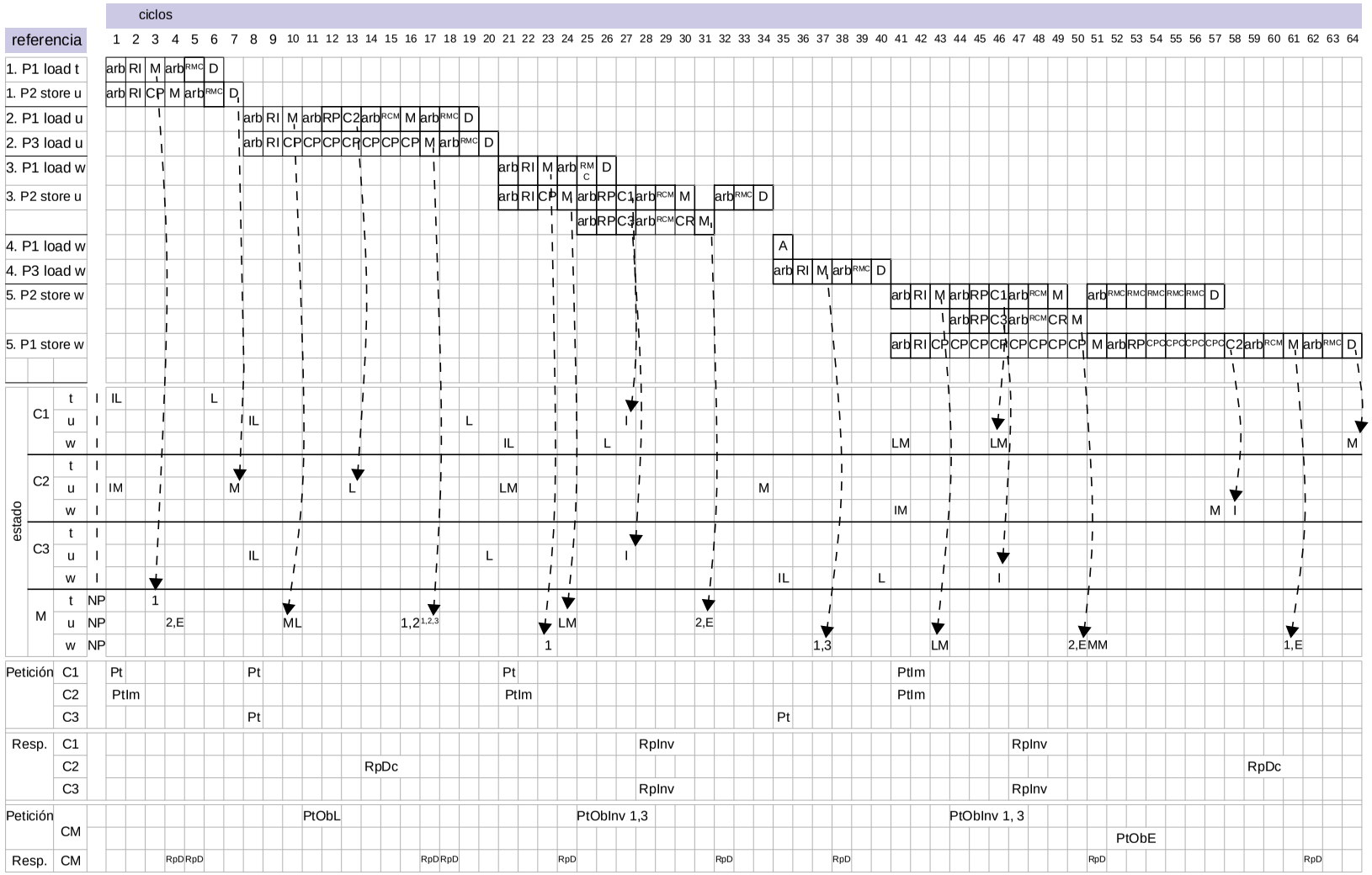
Respuesta: El primer par de accesos a memoria son debidos a un fallo de lectura y a un fallo de escritura. El estado transitorio de los bloques al emitir las peticiones es IL en la cache C1 (Pt) e IM en la cache C2 (PtIm). En las dos transacciones el bloque lo suministra memoria. El estados de los bloques al finalizar las transacciones es L en la cache C1 y M en la cache C2. En el directorio ha sido activado el BE del bloque que contiene la variable u. La petición del CC2 ha estado almacenada en la CP. El CM sólo procesa una petición en un instante determinado.

El segundo par de accesos a memoria son fallos de lectura. El CM procesa en primer lugar el mensaje emitido por el CC1. El procesado del mensaje requiere que el CM solicite el bloque a la cache C2, mediante una petición de observación de lectura (PtObL). El segundo fallo hace referencia al mismo bloque (P3). Durante el lapso de tiempo transcurrido hasta finalizar la transacción del CC1, la petición del CC3 ha estado almacenada en la CP. Como memoria ha sido actualizada en la transacción previa, el CM suministra directamente el bloque. Al finalizar las dos transacciones el VP del bloque identifica que las caches C1, C2 y C3 tienen copia del bloque.

El tercer grupo de accesos a memoria requiere dos transacciones. La primera petición es de lectura (Pt) y el bloque lo suministra la memoria. La segunda petición es de lectura con intención de modificación (PtIm). El CM emite peticiones de invalidación del bloque (PtObInv) a las caches C1 y C3. El CM, antes de responder al CC2, espera las respuestas del CC1 y CC3. El estado estable del bloque, al finalizar la transacción, es M en la cache C2 e I en las caches C1 y C3.

El cuarto par de accesos a memoria requiere sólo una transacción. El primer acceso a memoria es un acierto en cache. El segundo acceso a memoria, accede al mismo bloque, y es un fallo. Al finalizar la transacción, el estado del bloque que contiene la variable w es L en las caches C1 y C3.

El último grupo de accesos a memoria son dos escrituras. Ninguna de las caches accedidas tiene el bloque en exclusividad. La cache C2 no tiene copia del bloque y la cache C1 tiene una copia en el estado L. La petición de CC2 requiere que el CM emita una petición de invalidación al CC1 y al CC3. El CM espera las respuestas del CC1 y CC3 antes de responder al CC2. Esta respuesta al CC2 experimenta un retardo de 5 ciclos en la RMC. El CM, al procesar la petición del CC1, emite un mensaje de petición de observación de escritura al CC2. Esta petición no se procesa en el CC1, ya que no se dispone del bloque. La petición está almacenada en la CPC. Una vez el CC1 dispone del bloque respode al CM y este a su vez al CC1. Finalmente el CM suministra el bloque a CC1. Al finalizar las dos transacciones, el VP del bloque en el directorio indica que la cache C1 tiene copia del bloque en exclusividad.


ciclos
referencia
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
1. P1 load t
1. P2 store u
2. P1 load u
2. P3 load u
3. P1 load w
3. P2 store u
4. P1 load w
4. P3 load w
5. P2 store w
5. P1 store w
estado
C1
C2
C3
M
t
u
w
NP
Petición
Resp.
PtObL
PtObInv 1,3
PtObInv 1, 3
PtObE
CM
RpD

Recepción de respuestas de los CC en el CC solicitante

Utilice la siguiente secuencia de accesos concurrentes a memoria para comparar, el protocolo donde las respuestas se recolectan en el CC solicitante con el segundo protocolo descrito en este capítulo. El bloque que contiene la variable t está almacenado en las caches C2, C3 y C4.

accesos
P1 store t
P5 load t
P6 load v

Pregunta 1: *Utilice un diagrama temporal para cada protocolo con el objetivo de efectuar la comparación.*

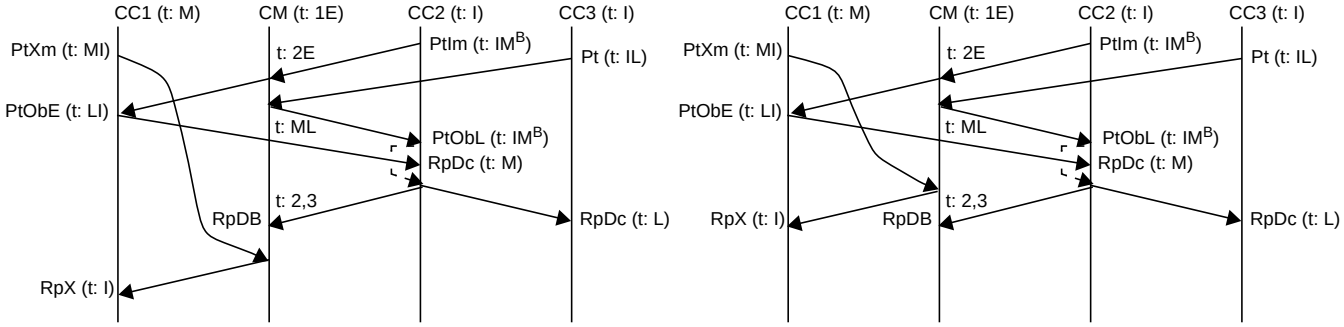
Respuesta: En la siguiente figura se muestra la comparación. Notemos la serialización que introduce el segundo protocolo de este capítulo en el procesado de peticiones en el CM (parte superior de la figura). Ello es debido a que el suministro del bloque al CC solicitante se efectúa siempre desde el CM.

	ciclos																				
referencia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P1 store t	arb	RI	M	arb	RPR	C2	arb	RCM	M			arb	RPR	D							
				arb	RPR	C3	arb	RCM	RCM	M											
				arb	RPR	C4	arb	RCM	RCM	RCM	M										
P5 load t	arb	RI	CP	CP	CP	CP	CP	CP	CP	CP	CP	M	arb	RPR	C1	arb	RCM	M	arb	RPR	D
P6 load v	arb	RI	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	M	arb	RPR	D					

P1 store t	arb	RI	M	arb	RP	D															
				arb	RP	C2	arb	RDR	C1												
				arb	RP	C3	arb	RDR	CRC	C1											
				arb	RP	C4	arb	RDR	RDR	CRC	C1										
P5 load t	arb	RI	CP	M	arb	RP	CPC	CPC	CPC	CPC	CPC	C1	arb	RDR	D						
													arb	RDR	M						
P6 load v	arb	RI	CP	CP	M	arb	RPR	D													

Pregunta 2: *Utilice el orden de procesado G en el CM de una secuencia de accesos para mostrar, mediante sendos diagrama temporales, el cruce de peticiones en el estado ML y en el estado L (Figura 8.59).*

Respuesta: En la parte izquierda de la figura se muestra la inferencia del cruce de peticiones en el estado L. En la parte derecha de la figura se muestra la inferencia del cruce de peticiones en el estado ML.



L

accesos	ciclos																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P2 store t	arb	RI	M	arb	RP	C1	arb	RDR	D									
P3 load t	arb	RI	CP	M	arb	RP	CPC	CPC	CPC	C2	arb	RDR	D					
											arb	RDR	M					
. . .						. .										. . .		
P1 PtXl t	arb	RI	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	M	arb	RP	X

ML

accesos	ciclos												
	1	2	3	4	5	6	7	8	9	10	11	12	13
P2 store t	arb	RI	M	arb	RP	C1	arb	RDR	D				
P3 load t	arb	RI	CP	M	arb	RP	CPC	CPC	CPC	C2	arb	RDR	D
											arb	RDR	M
P1 PtXl t	arb	RI	CP	CP	M	arb	RP	X					

Recordemos que la respuesta RpX se transmite por la red RP

En el multiprocesador se ejecuta la siguiente secuencia de accesos a memoria.

accesos	accesos
1. P1 load t	4. P1 load w
1. P2 store u	4. P3 load w
2. P1 load u	5. P2 store w
2. P3 load u	5. P1 store w
3. P1 load w	
3. P2 store u	

Pregunta 3: *Represente mediante un diagrama temporal la secuencia de accesos previa. Suponga que la respuesta del CM al CC2 en el quinto grupo de accesos a memoria experimenta un retardo total de 5 ciclos en la RP.*

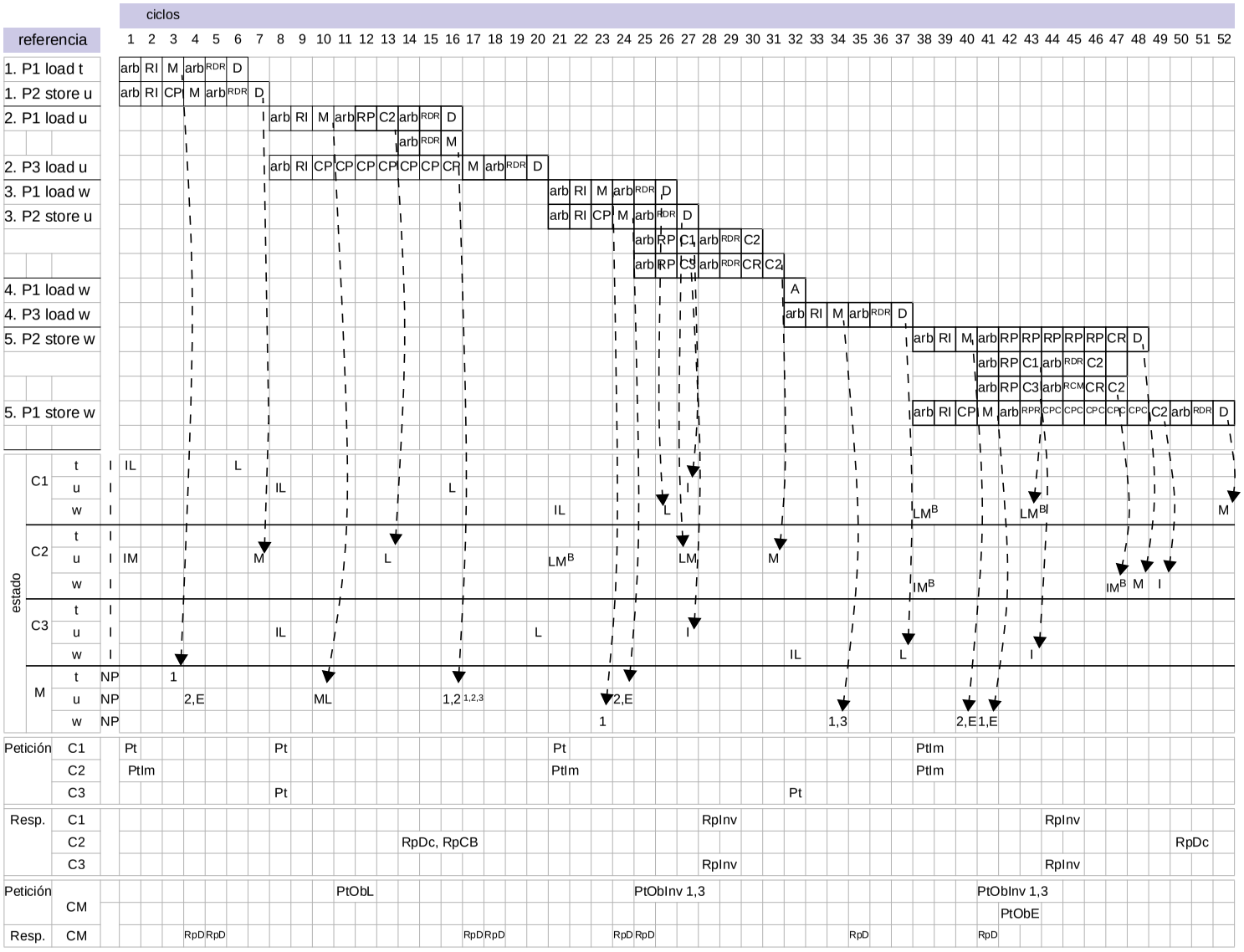
Respuesta: El primer par de accesos a memoria son debidos a un fallo de lectura y a un fallo de escritura. El estado transitorio de los bloques al emitir las peticiones es IL en la cache C1 (Pt) e IM en la cache C2 (PtIm). La petición del CC2 se almacena en la CP. El CM sólo procesa una petición en un instante dado. En las dos transacciones el bloque lo suministra memoria. El estados de los bloques al finalizar las transacciones es L en la cache C1 y M en la cache C2. En el directorio ha sido activado el BE del bloque que contiene la variable u.

El segundo par de accesos a memoria son fallos de lectura. El CM procesa en primer lugar el mensaje emitido por CC1. El procesado del mensaje requiere que el CM solicite el bloque a la cache C2, mediante una petición de observación de lectura (PtObL). El CC2 envia sendas respuestas al CC1 y al CM. El segundo fallo hace referencia al mismo bloque. Como memoria ha sido actualizada en la transacción previa, el CM suministra directamente el bloque. Al finalizar las dos transacciones el VP del bloque identifica que las caches C1, C2 y C3 tienen copia del bloque.

El tercer grupo de accesos a memoria requiere dos transacciones. La primera petición es de lectura (Pt) y el bloque lo suministra la memoria. La segunda petición es de lectura con intención de modificación (PtIm). El CM emite peticiones de invalidación del bloque (PtObInv) a las caches C1 y C3. Los CC correspondientes responden al CC2. El estado estable del bloque, al finalizar la transacción, es M en la cache C2 e I en las caches C1 y C3.

El cuarto par de accesos a memoria requiere sólo una transacción. El primer acceso a memoria es un acierto en cache. El segundo acceso a memoria (P3), accede al mismo bloque, pero es un fallo. Al finalizar la transacción, el estado del bloque que contiene la variable w es L en las caches C1 y C3.

El último grupo de accesos a memoria son dos escrituras. Ninguna de las caches accedidas tiene el bloque en exclusividad. La cache C2 no tiene copia del bloque y la cache C1 tiene una copia en el estado L. La petición de CC2 requiere que el CM emita una petición de invalidación al CC1 y al CC3. El CM también responde al CC2. Esta respuesta experimenta un retardo en la RP. Los CC de las caches C1 y C3 responden al CC2. La respuesta del CC3 se espera en la CR. También se espera en la CR la respuesta del CM. Recordemos que un CC sólo procesa un mensaje en un instante dado. El CM, al procesar la petición del CC1, emite un mensaje de petición de observación de escritura al CC2. El CC2 no dispone del bloque y la petición se queda encolada en la CPC. Cuando el CC2 dispone del bloque responde al CC1. Al finalizar las dos transacciones, el VP del bloque en el directorio indica que la cache C1 tiene copia del bloque en exclusividad.


ciclos
referencia
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
1. P1 load t
1. P2 store u
2. P1 load u
2. P3 load u
3. P1 load w
3. P2 store u
4. P1 load w
4. P3 load w
5. P2 store w
5. P1 store w
arb RI M CP RP C1 C2 C3 CR A D RDR RPR RCM CPC
estado
C1
C2
C3
M
t
u
w
I IL L IM M LM B ML NP 1 2,E 1,2 1,2,3 1,3
Petición
Pt PtIm
Resp.
RpInv RpDc, RpCB RpDc
Petición CM
PtObL PtObInv 1,3 PtObE
Resp. CM
RpD

EJERCICIOS

Descripción de un protocolo de directorio MLI denominado A

Suponga un multiprocesador donde las caches privadas son de mapeo directo y utilizan escritura retardada. El multiprocesador utiliza un directorio para mantener la coherencia y el protocolo de coherencia es de invalidación (MLI).

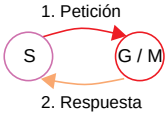
Las caches privadas de los procesadores son bloqueantes. En un fallo de cache o en una solicitud de exclusividad se suspende la interpretación de instrucciones y se reanuda al finalizar la transacción.

El directorio utiliza un vector de presencia y un bit de exclusividad por bloque. El vector de presencia (VP) es un vector de bits, con tantos bits como procesadores y cada bit está asociado a un procesador. El bit de exclusividad (BE) se utiliza para indicar que sólo existe una copia del bloque en una cache privada, la cual está identificada en el vector de presencia.

Las secuencias de mensajes de las transacciones son las siguientes:

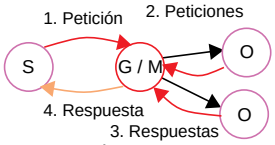
Transacciones

2 pasos



lectura
expulsión

4 pasos



escritura

4 pasos

lectura o escritura y el bloque esta modificado en otra cache

S: solicitante

G/M: controlador de memoria (directorio)

O: otro controlador de coherencia

Las peticiones de procesador y los mensajes utilizados en la transacciones para mantener la coherencia son:

Procesador	Controlador de cache (CC)		Controlador de memoria (CM)		
Peticiones	**Peticiones del CC al CM**	**Respuestas del CM al CC**	**Peticiones del CM a los CC**	**Respuestas del CC al CM**	**Acciones**
LPr : lectura	Pt : petición de bloque	RpD: respuesta con el bloque a una petición Pt o PtIm	PtObE: petición de observación de escritura, inducida por una petición PtIm	RpDc: respuesta con el boque a una petición PtObL o PtObE y el estado del bloque en cache es M	Actualización del directorio
EPr : escritura	PtIm: petición de bloque con intención de modificarlo	RpX: respuesta de confirmación a una petición PtXm o PtXl	PtObL: petición de observación de lectura, inducida por una petición Pt y el estado del bloque en el directorio es M	RpInv: Respuesta a una petición PtObE y el estado del bloque en cache es L	Dev: actualización de memoria
	PtXm: petición de expulsión de un bloque en estado M				
	PtXl: petición (notificación) de expulsión de un bloque en estado L				

El controlador de cache también efectúa acciones de reemplazo cuando es necesario (CcRe). En una acción de reemplazo se distingue la acción de notificación al directorio, ya que éste es preciso y si es el caso, una actualización de memoria con el bloque expulsado, si éste ha sido modificado durante su estancia en la cache. En una petición PtXm se actualiza el directorio y memoria, mientras que en una petición PtXl sólo se actualiza el directorio.

Cuando el servicio de un acceso a memoria requiere un reemplazo, éste se efectúa antes de gestionar el acceso a memoria que produce la acción de reemplazo.

La red de comunicación entre los CM y los CC se utiliza para transmitir peticiones y respuestas. Esta red sólo mantiene orden punto a punto entre un emisor y un receptor.

Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtIm, PtXm, PXl	arb	RI	M
RpD, RpX	arb	RPR	D ó X
PtObE, PtObL	arb	RPR	Cx
RpDc, RpInv	arb	RCM	M

arb: arbitraje en la red correspondiente (RI, RPR, RCM)

RI: red de peticiones desde los CC a los CM

RPR: red de respuestas desde los CM a los CC

RPR: red de peticiones desde los CM a los CC

RCM: red de respuestas desde los CC a los CM

M: memoria (directorio)

D: dato (RpD)

X: confirmación (RpX)

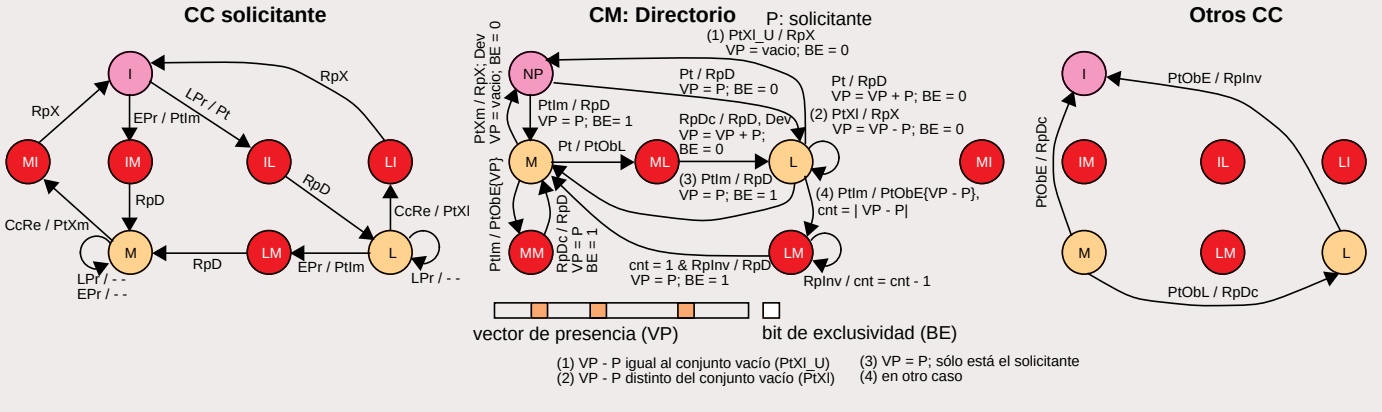
Cx: cache, siendo x el ordinal

Suponemos que un CM genera los mensajes de petición y respuesta en paralelo.Cuando no hay conflictos en la red, los mensajes se propagan en paralelo.

Cuando varios mensajes llegan en el mismo instante a un destinatario se encolan en paralelo en la cola correspondiente. Un destinatario sólo procesa un mensaje en un instante dado.

En un CC se utilizan estados transitorios (IL, LM, IM, LI, MI) mientras se espera la respuesta de una transaccción iniciada previamente. En el CM se utilizan estados transitorios (ML, MM) mientra se esperan las respuestas de los CC involucrados en la acción de coherencia.

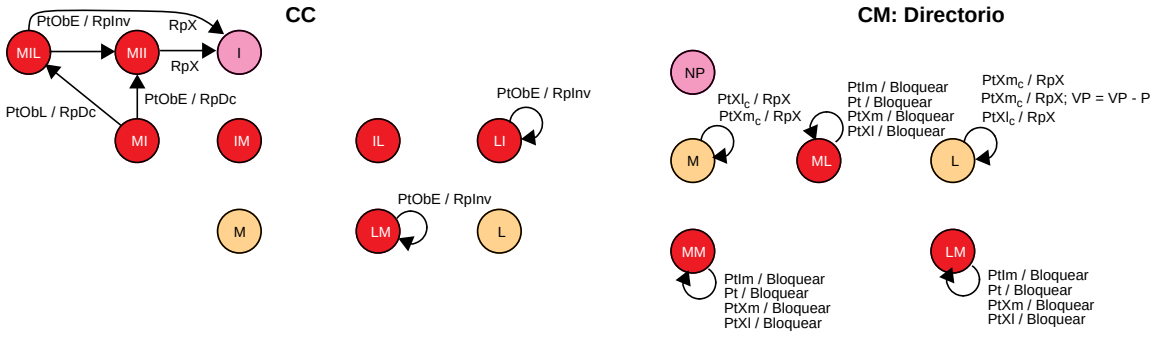
En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio. En el protocolo que se describe, el bit de exclusividad del directorio se activa cuando una cache solicita el bloque para actualizarlo.



Al tener en cuenta la concurrencia de peticiones de los CC hay que considerar los posibles cruces de peticiones en el CM y en los CC. En el CM sólo se procesan peticiones a bloques en estados estables. Una petición en la cabeza de la cola de peticiones (CP) que accede a un bloque en un estado transitorio, determina un bloqueo del análisis de esta petición y las que le siguen en la CP. La cola de respuestas en un CM se etiqueta como CR. En la CR se pueden encolar varios mensajes en el mismo instante de tiempo. Ahora bien, en el CM se procesan en serie.

En una cola de mensajes, el bloqueo del procesado por parte del autómata correspondiente, la espera por riesgo estructural, o cualquier otra causa que indique que no se procesa el mensaje se representa mediante el acrónimo de la cola.

En la siguiente figura se muestra la gestión de peticiones en los cruces. En el diagrama de estados de un bloque en el directorio, se utiliza el subíndice c para identificar una petición del CC que se ha cruzado con una petición del CM y la petición del CC se analiza en un estado estable. Notemos que en un cruce no se actualiza la memoria al procesar la petición PtXm.



Descripción de un protocolo de directorio MLI denominado B

Mensajes. Los mensajes que se han añadido o modificado respecto al protocolo A se muestran en la siguiente tabla.

Controlador de memoria	Controlador de coherencia	Comentario
Mensajes de petición a CC y acciones	**Respuestas**	
PtObInv: petición de invalidación op | dirección | Id	RpInV: respuesta a una petición de invalidación Id |	El CC que recibe una petición PtObInv en el estado L invalida el bloque y responde con RpInv.
PtObE: petición de observación de escritura op | dirección | Id	RpDc: respuesta con el bloque Id | bloque	El CC que recibe la petición PtObE tiene el bloque en exclusividad. Emite una respuesta con el bloque y la acción de invalidación está implícita en el tipo de respuesta. Esta respuesta también se utiliza si la petición del CM es PtObL. En este caso no se invalida el bloque al responder.

Redes. Un CM utiliza redes distintas para transmitir las peticiones y respuestas a los CC. La red de peticiones de los CM a los CC mantiene el orden entre un emisor y un receptor. Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtIm, PtXm, PXl	arb	RI	M
RpD, RpX	arb	RMC	D ó X
PtObE, PtObL	arb	RP	Cx
RpDc, RpInv	arb	RCM	M

arb: arbitraje en la red correspondiente (RI, RP, RCM , RMC)

RI: red de peticiones desde los CC a los CM

RMC: red de respuestas desde los CM a los CC

RP: red de peticiones desde los CM a los CC

RCM: red de respuestas desde los CC a los CM

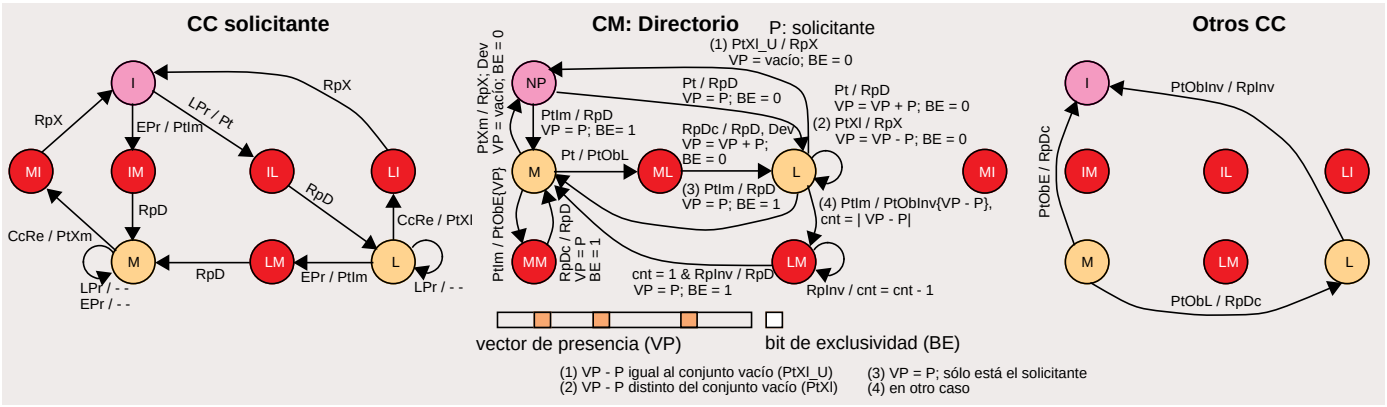
M: memoria (directorio)

D: dato (RpD)

X: confirmación (RpX)

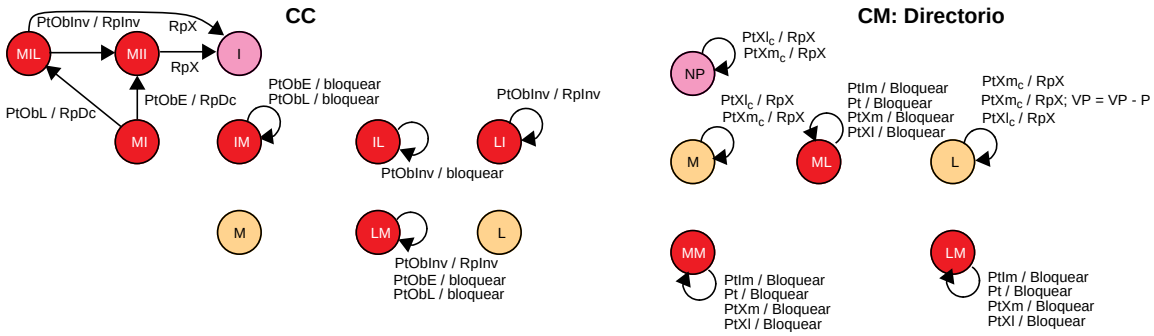
Cx: cache, siendo x el ordinal

En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio. En el protocolo que se describe, el bit de exclusividad del directorio se activa cuando una cache solicita el bloque para actualizarlo.



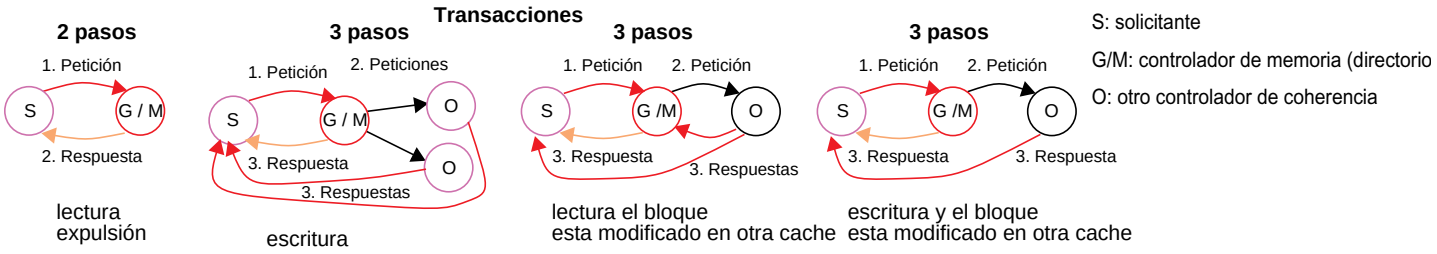
La cola de peticiones en un CC se etiqueta como CPC. En la CPC se pueden encolar varios mensajes en el mismo instante de tiempo. Ahora bien, en el CC se procesan en serie.

En la siguiente figura se muestra la gestión de peticiones en los cruces. En el diagrama de estados de un bloque en el directorio, se utiliza el subíndice c para identificar una petición del CC que se ha cruzado con una petición del CM y la petición del CC se analiza en un estado estable. Notemos que en un cruce no se actualiza la memoria al procesar la petición PtXm.



Descripción de un protocolo de directorio MLI denominado C

Las secuencias de mensajes de las transacciones son los siguientes:



Mensajes. Los mensajes que se han añadido o modificado respecto al protocolo A se muestran en las siguientes tablas.

Controlador de coherencia (CC) Mensajes de petición al CM	Controlador de memoria (CM) Mensaje de respuesta	Comentario
Pt: petición de bloque op | dirección | Id	RpD, NR: respuesta con el bloque y el número de respuestas que debe esperar Id | bloque | NR	Se lee el bloque de la memoria, se actualiza el directorio, se suministra el bloque y se indica el número de respuestas que el CC solicitante debe recibir (NR).

Controlador de memoria Mensajes de petición a los CC y acciones	Controlador de coherencia Respuestas	Comentario
PtObE: petición de observación de escritura op | dirección | Id | DEST	RpDc: respuesta con el bloque e invalidación Id | bloque | DEST	El CC emite una respuesta al solicitante (DEST) con el bloque e invalida el bloque.
PtObL: petición de observación de lectura op | dirección | Id | DEST	RpCB: respuesta con el bloque al CM Id | bloque RpDc: respuesta con el bloque al CC Id | bloque | DEST	El CC emite una respuesta al solicitante (DEST) con el bloque y cambia el estado del bloque para indicar que no hay exclusividad. El CC suministra el bloque al CM.
PtObInv: petición de invalidación op | dirección | Id | DEST	RpInv: respuesta a una petición de invalidación Id | DEST	El CC que recibe la petición invalida el bloque y responde al solicitante (DEST).

Redes. El CM utiliza redes distintas para transmitir las peticiones y respuestas a los CC. Los CC y los CM utilizan la misma red para transmitir las respuestas a los CC y CM. La red de peticiones de los CM a los CC mantiene el orden entre un emisor y un receptor. Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtIm, PtXm, PXl	arb	RI	M
RpD	arb	RDR	D ó X
PtObE, PtObL, RpX	arb	RP	Cx
RpDc, RpInv	arb	RDR	M

arb: arbitraje en la red correspondiente (RI, RP, RDR)

RI: red de peticiones desde los CC a los CM

RDR: red de respuestas desde los CM a los CC

RP: red de peticiones desde los CM a los CC

RDR: red de respuestas desde los CC a los CM

M: memoria (directorio)

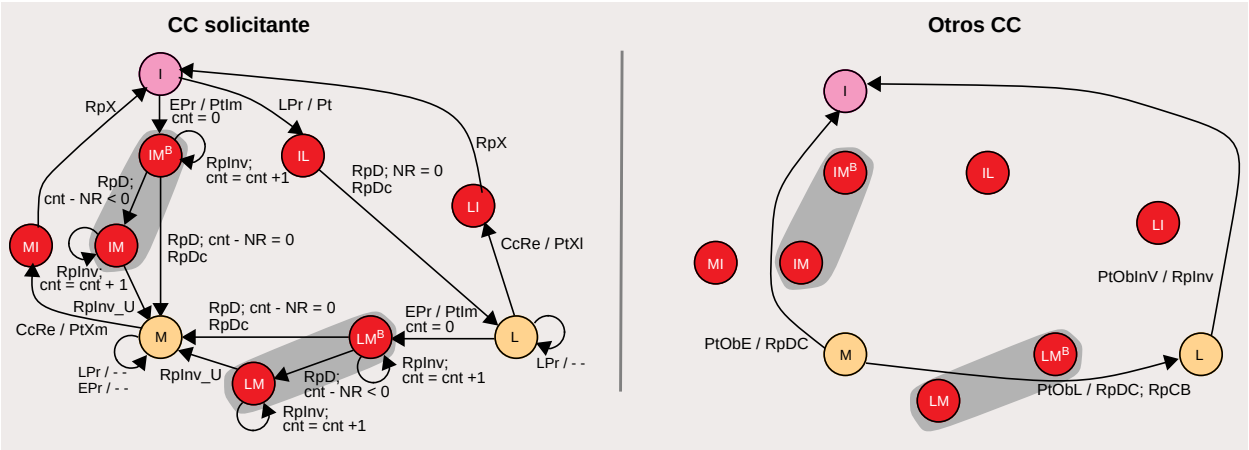
D: dato (RpD)

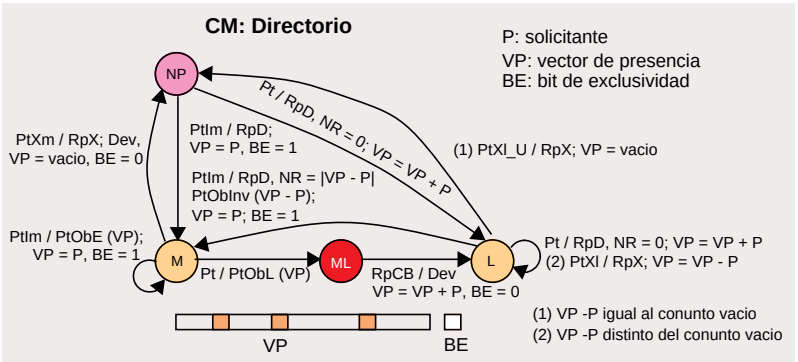
X: confirmación (RpX)

Cx: cache, siendo x el ordinal

La respuesta de un CM a una petición de expulsión se transmite por la red RP.

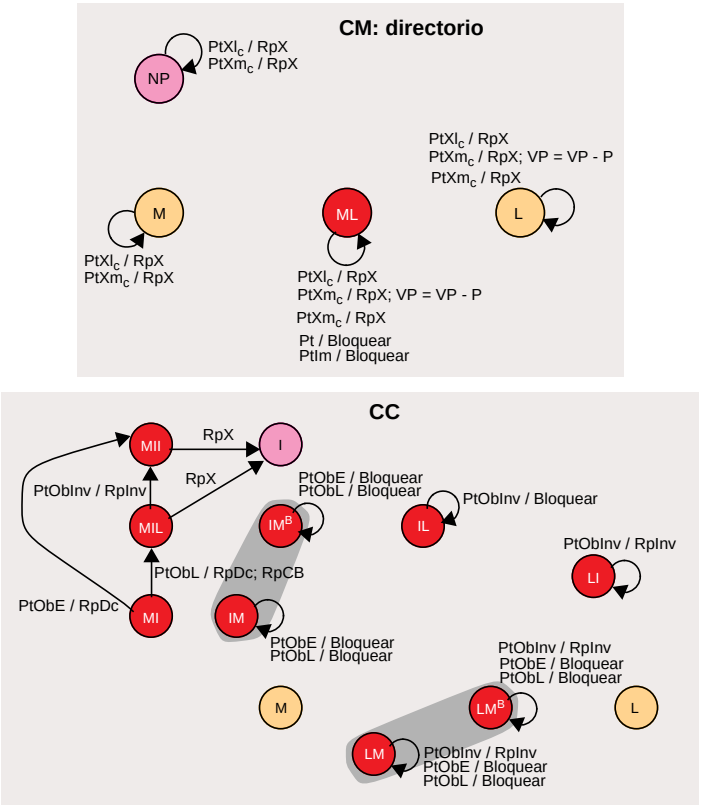
En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio. En el protocolo que se describe, el bit de exclusividad del directorio se activa cuando una cache solicita el bloque para actualizarlo. La recepción de dos tipos de respuestas, sin orden, en los CC requiere utilizar un reconocedor de secuencia: (RpInv*, RpD, RpInv*).





La cola de respuestas en un CC se etiqueta como CRC. En la CRC se pueden encolar varios mensajes en el mismo ciclo. Ahora bien, en el CC se procesan en serie.

En la siguiente figura se muestra la gestión de peticiones en los cruces. En el diagrama de estados de un bloque en el directorio, se utiliza el subíndice c para identificar una petición del CC que se ha cruzado con una petición del CM y la petición del CC se analiza en un estado estable. Notemos que en un cruce no se actualiza la memoria al procesar la petición PtXm.

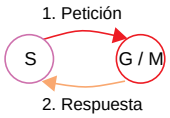


Descripción de un protocolo de directorio MLI denominado D

Las secuencias de mensajes de las transacciones son los siguientes:

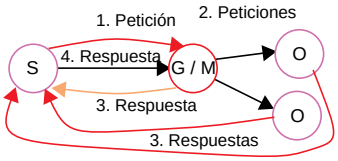
Transacciones

2 pasos



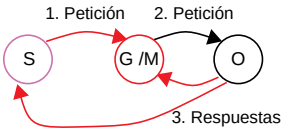
lectura
expulsión

4 pasos



escritura

3 pasos



lectura o escritura y el bloque
esta modificado en otra cache

S: solicitante

G/M: controlador de memoria (directorio)

O: otro controlador de coherencia

Mensajes. Los mensajes que se han añadido o modificado respecto al protocolo A se muestran en las siguientes tablas.

Controlador de coherencia (CC) **Mensajes de petición al CM**	Controlador de memoria (CM) **Mensaje de respuesta**	Comentario
Pt: petición de bloque op | dirección | Id	RpD, NR: respuesta con el bloque y el número de respuestas que debe esperar Id | bloque | NR	Se lee el bloque de la memoria, se actualiza el directorio, se suministra el bloque y se indica el número de respuestas que el CC solicitante debe recibir (NR).

Controlador de memoria **Mensajes de petición a los CC y acciones**	Controlador de coherencia **Respuestas**	Comentario
PtObE: petición de observación de escritura op | dirección | Id | DEST	RpDc: respuesta con el bloque e invalidación Id | bloque | DEST	El CC emite una respuesta al solicitante (DEST) con el bloque e invalida el bloque.
PtObL: petición de observación de lectura op | dirección | Id | DEST	RpCB: respuesta con el bloque al CM Id | bloque RpDc: respuesta con el bloque al CC Id | bloque | DEST	El CC emite una respuesta al solicitante (DEST) con el bloque y cambia el estado del bloque para indicar que no hay exclusividad. El CC suministra el bloque al CM.
PtObInv: petición de invalidación op | dirección | Id | DEST	RpInv: respuesta a una petición de invalidación Id | DEST	El CC que recibe la petición invalida el bloque y responde al solicitante (DEST).
	RpCF: notificación al CM de que los CC participantes en la acción de coherencia la han observado Id | bloque	El CC que tenía el bloque en exclusividad notifica al CM que ha emitido la respuesta al solicitante. El CC solicitante notifica al CM que ha recibido todas las respuestas

Redes. El CM utiliza redes distintas para transmitir las peticiones y respuestas a los CC. Los CC y los CM utilizan la misma red para transmitir las respuestas a los CC y CM. Ninguna de las redes mantiene el orden de los mensajes transmitidos entre dos nodos. Las fases de cada uno de los mensajes son:

mensajes	ciclos 1	2	3
Pt, PtIm, PtXm, PXl	arb	RI	M
RpD	arb	RDR	D ó X
PtObE, PtObL, RpX	arb	RP	Cx
RpDc, RpInv	arb	RDR	M

arb: arbitraje en la red correspondiente (RI, RP, RDR)

RI: red de peticiones desde los CC a los CM

RDR: red de respuestas desde los CM a los CC

RP: red de peticiones desde los CM a los CC

RDR: red de respuestas desde los CC a los CM

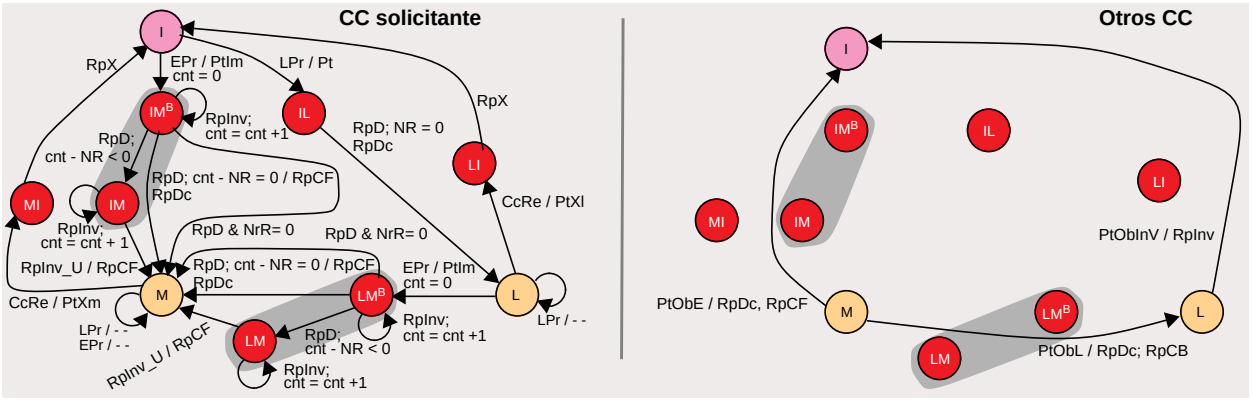
M: memoria (directorio)

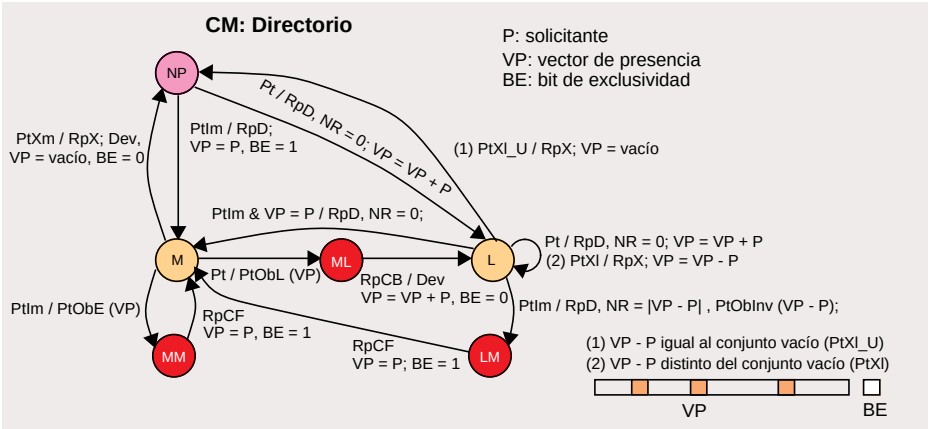
D: dato (RpD)

X: confirmación (RpX)

Cx: cache, siendo x el ordinal

En los siguientes diagramas de estados se muestran todas las transiciones entre estados, ya sean estables o transitorios, de un bloque en cache y en el directorio. En el protocolo que se describe, el bit de exclusividad del directorio se activa cuando una cache solicita el bloque para actualizarlo.

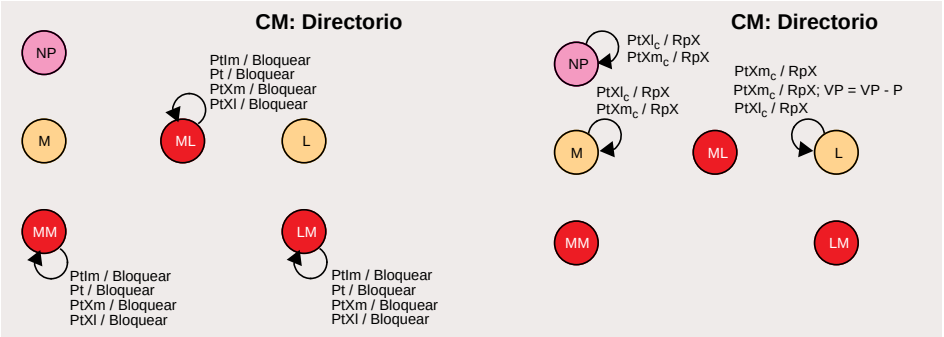


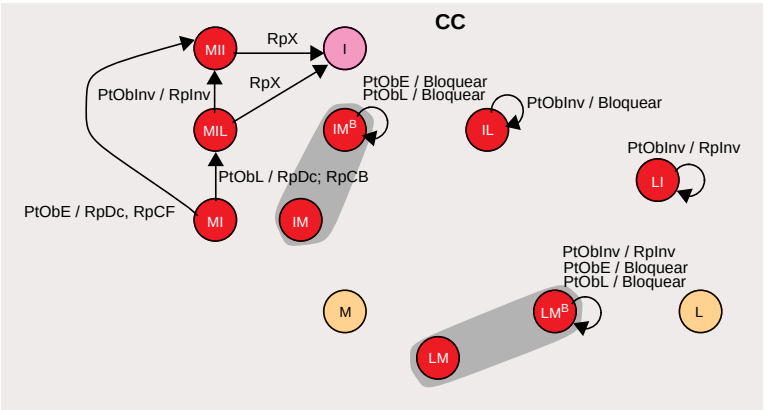


Dada una cola, se pueden encolar varios mensajes en el mismo ciclo. Ahora bien, se procesan en serie.

En una cola de mensajes, el bloqueo del procesado por parte del autómata correspondiente, la espera por riesgo estructural, o cualquier otra causa que indique que no se procesa el mensaje se representa mediante el acrónimo de la cola.

En la siguiente figura se muestra la gestión de peticiones en los cruces. En el diagrama de estados de un bloque en el directorio, se utiliza el subíndice c para identificar una petición del CC que se ha cruzado con una petición del CM y la petición del CC se analiza en un estado estable. Notemos que en un cruce no se actualiza la memoria al procesar la petición PtXm.





Ejercicio 8.1

Un multiprocesador dispone de una red entre los CM y los CC por la que se transmiten peticiones y respuestas. Esta red mantiene el orden entre un emisor y un receptor. El CM es el encargado de recolectar las respuestas de los CC participantes en una acción de coherencia. Después de ello, el CM emite la respuesta al solicitante. El procolo de coherencia utilizado es el denominado A.

En una expulsión silenciosa, el directorio no recibe notificación de la expulsión de un bloque. Esta característica determina que un controlador de coherencia pueda recibir una petición de observación de escritura (PtObE) de un bloque que no tiene almacenado en cache. Esto es, el directorio es impreciso.

Un ingeniero utiliza las siguientes secuencias de accesos a memoria concurrentes para efectuar el análisis de transiciones entre estados cuando se utiliza expulsión silenciosa. En las tres secuencias hay tres grupos de accesos. En ocasiones, en un grupo sólo hay un acceso a memoria.

A	B	C
accesos	accesos	accesos
P1 load t	P1 load t	P1 load t
P1 load u	P1 load u	P1 load u
P2 store t	P2 store t	P2 store t
	P1 load t	P1 store t

Las variables t y u están contenidas en bloque distintos.

Estos bloques se almacenan en el mismo contenedor de cache.

Ahora bien, para simplificar el análisis supondremos que la cache C1 ha expulsado el bloque que contiene la variable t de forma silenciosa. En consecuencia el CC1 está identificado en el VP. Posteriormente se producen las siguientes secuencias de accesos a memoria. El CM las procesa en el orden especificado.

A	B	C	
accesos	accesos	accesos	Estado
P2 store t	P2 store t	P2 store t	C1, C2 : I
	P1 load t	P1 store t	directorio: 1 (L)

Pregunta 1: *Para cada una de las secuencias de acceso muestre un diagrama temporal simplificado. En los casos en que no exista, en la descripción del protocolo, una transición para un par (petición, estado) determine las acciones que deben efectuarse.*

Pregunta 2: *Muestre en el diagrama de estados de un CC, correspondiente al agente procesador, la transición entre estados en una acción de reemplazo. Muestre en el diagrama de estados de un directorio preciso las transiciones que no se utilizan cuando el directorio es impreciso. Muestre también en un*

CC, agente observador, los estados en los cuales se pueden recibir peticiones que son debidas a que el directorio es impreciso. Indique la transición entre estados y la respuesta.

Ejercicio 8.2

Un multiprocesador dispone de una red entre los CM y los CC para transmitir las peticiones y de otra red para transmitir las respuestas. Estas redes mantienen el orden entre un emisor y un receptor. El CM es el encargado de recolectar las respuestas de los CC participantes en la acción de coherencia. Después de ello, el CM emite la respuesta al solicitante. El protocolo de coherencia utilizado es el denominado B.

Un ingeniero decide utilizar expulsión silenciosa. Para efectuar el análisis de las modificaciones en el protocolo de coherencia, el ingeniero utiliza las siguientes secuencia de acceso a memoria concurrente.

A	B	C	
accesos	accesos	accesos	Estado
P2 store t	P2 store t	P2 store t	C1, C2 : I
	P1 load t	P1 store t	directorio: 1 (L)

La cache C1 ha expulsado el bloque que contiene la variable t de forma silenciosa antes de procesar la secuencia previa. En consecuencia el CC1 está identificado en el VP. Las secuencias de accesos se procesan en el CM en el orden especificado.

Pregunta 1: *Para la secuencia de accesos C muestre un diagrama temporal simplificado. En los casos en que no exista, en la descripción del protocolo, una transición para un par (petición, estado) determine las acciones que deben efectuarse.*

Pregunta 2: *Para la secuencia de accesos B muestre un diagrama temporal simplificado. Justifique de forma razonada que se produce una situación de abrazo mortal.*

En estas condiciones hay que diseñar un mecanismo que detecte la posibilidad de abrazo mortal y actuar para que no se produzca. Un ingeniero diseña el siguiente mecanismo, teniendo en cuenta que el CM puede detectar la situación analizando el VP.

Detección: Cuando el CM analiza la cabeza de la CP y determina que la petición es Pt y que ésta accede a un bloque en el estado transitorio LM, analiza el VP. Si el CC, que ha emitido la petición, está en el VP el CM infiere un abrazo mortal.

Actuación: El CM responde al CC (RpD), pero no elimina al CC del VP. El CC bloquea el procesado de una petición PtObInv en el estado IL.

Dada la secuencia de accesos B, podemos decir que el CM modifica el orden de procesado de las peticiones de CC1 y CC2. Había empezado procesando en el orden PtIm, Pt y al detectar la situación procesa Pt, PtIm.

Pregunta 3: *Para la secuencia de accesos B muestre un diagrama temporal simplificado. Utilice el mecanismo descrito para soslayar el abrazo mortal. Refine el mecanismo para que funcione en cualquier caso.*

Pregunta 4: *Muestre en el diagrama de estados de un CC, correspondiente al agente procesador, la transición entre estados en una acción de reemplazo. Muestre en el diagrama de estados de un directorio preciso las transiciones que no se utilizan cuando el directorio es impreciso. Muestre también en un CC, agente observador, los estados en los cuales se pueden recibir peticiones que son debidas a que el directorio es impreciso. Indique la transición entre estados y la respuesta. Muestre las transiciones adicionales en el diagrama de estados del directorio.*

Otro ingeniero decide modificar de forma más radical el protocolo de coherencia del que parte, el cual es el protocolo con un directorio preciso.

Una petición PtObInv del CM es respondida siempre por un CC cuando el bloque está en el estado IL. Por otro lado, si un CC ha procesado una petición PtObInv, con el bloque en el estado IL, el CC descarta la respuesta del CM a su petición Pt. Posteriormente vuelve a emitir la petición Pt.

El CM no efectúa ninguna acción de detección.

Pregunta 5: *Muestre un diagrama de transiciones entre estados en el observador del CC que tenga en cuenta la descripción efectuada.*

Pregunta 6: *Para la secuencia de acceso B muestre un diagrama temporal simplificado. Utilice el último mecanismo descrito para soslayar el abrazo mortal.*

Pregunta 7: *Justifique si se puede producir inanición.*

Para efectuar una comparación con el protocolo con directorio preciso el ingeniero utiliza la siguiente secuencia de accesos a memoria y orden de procesado en el CM.

D

accesos	Estado
P1 load t P2 store t	C1, C2: I directorio: 1 (L)

Pregunta 8: *Para la secuencia de acceso D muestre un diagrama temporal simplificado. Utilice el último mecanismo descrito para soslayar el abrazo mortal. Suponga que la respuesta del CM a la petición del CC1 llega retrasada respecto a la petición del CM inducida por la petición de CC2. Efectúe el mismo análisis cuando el directorio es preciso. En este caso, en el VP no está incluido el CC1 al empezar la secuencia de accesos.*

Pregunta 9: *¿Es factible que el CC, en lugar de descartar la respuesta del CM y volver a efectuar la petición, ejecute la instrucción load que ha inducido la transacción Pt y posteriormente invalide el bloque?. Justifique la respuesta. Razone el efecto del mecanismo en el rendimiento.*

Otro diseñador decide modificar el protocolo de la siguiente forma. Cuando el CM procesa una petición Pt en la cabeza de la CP y el bloque está en el estado transitorio LM y además, el CC está identificado en el VP, el CM contabiliza la petición Pt como una respuesta del CC, que ha emitido la petición Pt, a la petición PtObInv. Entonces, decrementa el contador de respuestas de invalidación.

Cuando el valor del contador es cero el CM responde al CC que ha establecido el estado transitorio LM del bloque en el directorio. Posteriormente, el CM inicia el procesado de la petición Pt y responde al CC que la ha emitido. Finalmente, antes de dar por finalizada esta última transacción, el CM espera la respuesta RpInv del CC que ha emitido la petición Pt. Esta respuesta es descarta por el CM.

Pregunta 10: *Para la secuencia de acceso B muestre un diagrama temporal simplificado. Utilice el mecanismo que se acaba de describir para soslayar el abrazo mortal. Refine el mecanismo para que funcione correctamente en cualquier caso.*

Pregunta 11: *Muestre un diagrama de transiciones entre estados en el directorio que tenga en cuenta la descripción efectuada.*

Ejercicio 8.3

Un multiprocesador dispone de una red entre los CM y los CC para transmitir las peticiones y de otra red para transmitir las respuestas desde los CC y CM a los CC y CM. Estas redes mantienen el orden entre un emisor y un receptor. El CC solicitante es el encargado de recolectar las respuestas de los CC participantes. El protocolo de coherencia utilizado es el denominado C.

Notemos que en este protocolo de coherencia el CM no conoce cuando finaliza una transación de exclusividad (transición de los estados L o M al estado M). Por otro lado, recordemos que la respuesta RpX se transmite por la red de peticiones entre los CM y los CC para eliminar un cruce de peticiones (que la respuesta RpX se adelante a una petición previa del CM).

Un ingeniero decide utilizar expulsión silenciosa. Para efectuar el análisis de las modificaciones en el protocolo de coherencia utiliza las siguientes secuencias de acceso a memoria concurrente.

A	B	C	
accesos	accesos	accesos	Estado
P2 store t	P2 store t	P2 store t	C1, C2 : I
	P1 load t	P1 store t	directorio: 1 (L)

La cache C1 ha expulsado el bloque que contiene la variable t de forma silenciosa antes de procesar la secuencia previa. En consecuencia el CC1 está identificado en el VP. Las secuencias de accesos se procesan en el CM en el orden especificado.

Pregunta 1: *Para la secuencia de acceso C muestre un diagrama temporal simplificado. En los casos en que no exista, en la descripción del protocolo, una transición para un par (petición, estado) determine las acciones que deben efectuarse.*

Pregunta 2: *Para la secuencia de acceso B muestre un diagrama temporal simplificado. Justifique de forma razonada que se produce una situación de abrazo mortal.*

En estas condiciones hay que diseñar un mecanismo que detecte la posibilidad de abrazo mortal y actuar para que no se produzca. Un ingeniero diseña el siguiente mecanismo, teniendo en cuenta que el CM puede detectar la situación analizando el VP.

Detección: En el CC que ha efectuado la petición de exclusividad (CC2) se puede detectar la situación analizando el VP. Para ello el CM transmite el contenido de VP junto con el bloque en la respuesta RpD, NR. Entonces, si el CC que emite la petición Pt está includido en el VP, el CC2, al recibir la

petición PtObL, infiere que el CC1 ha expulsado de forma silenciosa el bloque y hay una situación de abrazo mortal. Recordemos que la petición PtObL se transmite al CC destinatario de la respuesta.

Actuación: Cuando el CC analiza la cabeza de la CPC y es una petición PtObL que referencia un bloque en el estado transitorio IM, analiza el VP. Si el CC, que ha inducido esta petición del CM, está en el VP, el CC infiere un abrazo mortal y responde al CC cuando dispone del bloque.

Pregunta 3: *Para la secuencia de acceso B muestre un diagrama temporal simplificado. Utilice el mecanismo descrito para soslayar el abrazo mortal. Refine el mecanismo para que funcione correctamente en cualquier caso.*

Pregunta 4: *Muestre en el diagrama de estados de un CC, correspondiente al agente procesador, la transición entre estados en una acción de reemplazo. Muestre en el diagrama de estados de un directorio preciso las transiciones que no se utilizan cuando el directorio es impreciso. Muestre también en el agente observador de un CC los estados en los cuales se pueden recibir peticiones que son debidas exclusivamente a que el directorio es impreciso. Indique la transición entre estados y la respuesta.*

Otro ingeniero decide modificar de forma más radical el protocolo de coherencia del que parte, el cual es el protocolo con un directorio preciso.

Una petición PtObInv del CM es respondida siempre por un CC cuando el bloque está en el estado IL. Por otro lado, si un CC ha procesado una petición PtObInv, con el bloque en el estado IL, el CC descarta la respuesta del CM a su petición Pt. Posteriormente vuelve a emitir la petición Pt.

El CM no efectúa ninguna acción de detección.

Pregunta 5: *Para la secuencia de acceso B muestre un diagrama temporal simplificado. Utilice el último mecanismo descrito para soslayar el abrazo mortal.*

Pregunta 6: *Justifique si se puede producir inanición.*

Un ingeniero novel utiliza la siguiente secuencia de accesos a memoria y orden de procesado en el CM para intentar afinar el mecanismo descrito.

D

accesos	Estado
P1 load t	C1, C2: I
P2 store t	directorio: 1 (L)

Pregunta 7: *Para la secuencia de acceso D muestre un diagrama temporal simplificado. Utilice el último mecanismo descrito para soslayar el abrazo mortal. Suponga que la respuesta del CM a la petición del CC1 llega retrasada respecto a la petición del CM inducida por la petición de CC2. Compare este diagrama con el diagrama de la pregunta previa, donde el orden de procesado de las peticiones en el CM está intercambiado. Justifique si un CC, a partir del tipo de respuestas recibidas, puede inferir el orden en el cual el CM ha procesado su petición y mantener la coherencia.*

Pregunta 8: *Muestre un diagrama de transiciones entre estados en el observador del CC que tenga en cuenta la descripción efectuada.*

Ejercicio 8.4

La red que se utiliza en el multiprocesador es genérica y el protocolo MLI utilizado es el denominado D. Utilice las siguiente secuencias de accesos a memoria para identificar cruces de peticiones.

Pregunta 1: *Para las secuencias de accesos a memoria que se muestran en la siguiente tabla, extraidas de la Figura 7.30, muestre diagramas temporales simplificados. Explique la forma de inferir el cruce y las acciones que se toman para mantener la coherencia.*

Orden en el CM	
A	**B**
P2 store t	P2 store t
P1 PtXl t	P3 load t
	P1 PtXl t

Pregunta 2: *Para las secuencias de accesos a memoria que se muestran en la siguiente tabla, extraidas de la Figura 7.32, muestre diagramas temporales simplificados. Explique la forma de inferir el cruce y las acciones que se toman para mantener la coherencia.*

Orden de procesado en el CM			
E	**F**	**G**	**H**
P2 store t	P2 load t	P2 store t	P2 load t
P1 PtXm t	P1 PtXm t	P3 load t	P3 store t
		P1 PtXm t	P1 PtXm t

Pregunta 3: *Para las secuencias de accesos a memoria que se muestran en la siguiente tabla, extraidas de la Figura 8.32, muestre diagramas temporales simplificados. Explique la forma de inferir el cruce y las acciones que se toman para mantener la coherencia.*

Orden de procesado en el CM		
I	**J**	**K**
P1 load t	P1 store t	P1 store t
P2 store t	P2 load t	P2 store t

Pregunta 4: *Para las secuencias de accesos a memoria que se muestran en la siguiente tabla, extraidas de la Figura 8.33, muestre diagramas temporales simplificados. Explique la forma de inferir el cruce y las acciones que se toman para mantener la coherencia.*

Orden de procesado en el CM	
AX	**EX**
P2 store t	P2 store t
P2 PtXm t	P2 PtXm t
P1 PtXl t	P1 PtXm t

Ejercicio 8.5

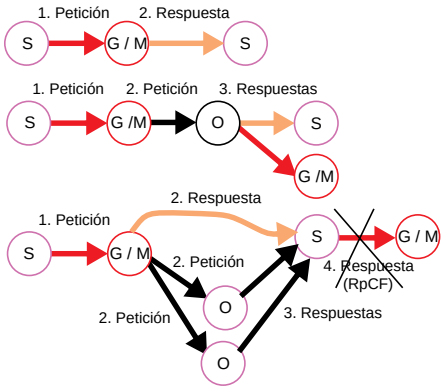
En el protocolo MLI denominado D, el cual se utiliza en una red genérica, se dispone de mensajes de notificación (RpCF) para ordenar las peticiones y respuestas del CM a un CC.

En un directorio preciso, un CC que recibe una petición PtObInv del CM para un bloque no recibirá posteriormente ninguna petición más, a menos que solicite otra vez el bloque.

Por otro lado, un CC recibe una respuesta RpX si ha emitido una petición PtXl de un bloque en el estado L. También recibe una respuesta si ha emitido una petición PtIm, que referencia a un bloque en el estado L o I. El protocolo D garantiza que estas respuestas, si se produce un cruce tardío, se reciben después del procesado en el CC de una petición PtObInv. El CM no procesa la petición PtXl o PtIm hasta que recibe la respuesta RpCF del CC cuya petición ha inducido la emisión de la petición PtObInv.

Una vez el CM procesa una petición PtXl o PtIm (no se produce un cruce tardío), el CC que las ha emitido no recibe una petición PtObInv. En el directorio el CC no está identificado en el VP (PtXl) asociado al bloque, o el estado del bloque en el directorio es M.

La petición PtXI y respuesta asociada son para que el directorio sea preciso. Por otro lado, que el CM espere la respuesta RpCF de un CC, el cual la emite después de recolectar las respuestas RpInv, representa una serialización en el procesado de transacciones en el CM, aunque garantiza que la respuesta RpX no llega al CC, que ha emitido PtXI, antes que una petición PtObInv.



Esta serialización también elimina cruces de peticiones tempranos en los estados IM y LM.

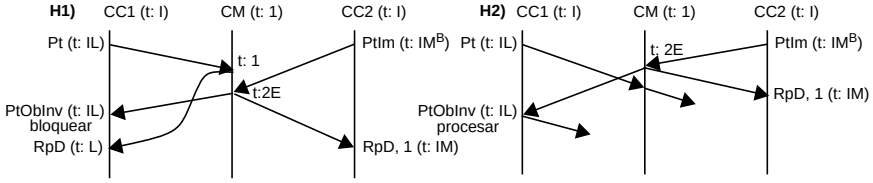
En estas condiciones, la utilización de un directorio impreciso permite eliminar la serialización descrita en el CM y reduce el tráfico de peticiones de los CC (PtXI) y respuestas de los CM (RpX). Por otro lado, puede incrementar el tráfico de peticiones del CM (PtObInv). El coste es gestionar cruces adicionales en un CC.

A este protocolo MLI lo denominamos DL.

Pregunta 1: *Para el protocolo DL, muestre los diagramas de transiciones entre estados de un bloque en el CC y el CM.*

Suponga los siguientes dos accesos a memoria (parte izquierda de la siguiente figura). Utilizando el protocolo DL, en la parte de la derecha se muestran diagrama temporales donde se producen cruces de mensajes (peticiones, respuestas).

Orden en el CM		Estado del bloque en las caches y directorio
H1	H2	
P1 load t	P2 store t	C1, C2: I
P2 store t	P1 load t	directorio: 1 (L), impreciso



Del análisis de los diagramas observamos que en el estado IL es necesario efectuar dos respuestas incompatibles (un oráculo nos indica el orden de procesado en el CM): a) bloquear (el CM ordena 1º Pt), b) procesar (el CM ordena 1º PtIm). Una forma de solventarlo es responder y utilizar un estado transitorio (ILI). Este estado se utiliza para recordar que entre la petición Pt y la recepción de la respuesta (RpD, RpDc) ha sido procesada una petición PtObInv. Cuando éste sea el caso, después de recibir el bloque se ejecuta la instrucción load y se descarta (invalida) el bloque.

Pregunta 2: *Utilice el protocolo DL. Muestre un diagrama de transiciones entre estados de un bloque en el CC donde se tenga en cuenta la descripción que acaba de efectuarse.*

Para efectuar un análisis de las implicaciones de utilizar expulsión silenciosa se utilizan las siguientes secuencias de accesos a memoria.

Orden de procesado en el CM			Estado del bloque en las caches y directorio
A	B	C	
P2 store t	P2 store t	P2 store t	C1, C2: I
	P1 load t	P1 store t	directorio: 1 (L)

Pregunta 3: *Utilice el protocolo DL. Para las secuencias de accesos a memoria relacionadas previamente muestre diagramas temporales simplificados. Analice los cruces en el CC1 y en el CC2. Explique la forma de inferir el cruce y las acciones que se toman para mantener la coherencia.*

Suponga que en el multiprocesador se ejecutan los siguientes dos hilos.

H1	H2	H1	H2	Estado del bloque en las caches y directorio
A = 3.21	while (A == 0) { }	P1 store A	P2 load aviso	A = B = 0; C1, C2: A, B; estado I
aviso = 1	T = A	P1 store aviso	P2 load A	directorio: A: NP, aviso: 2 (L)

Nos centraremos en las instrucciones que acceden a la variable aviso y tendremos en cuenta dos entrelazados.

Entrelazado 1	Entrelazado 2	Comentarios
P2 load aviso	P1 store aviso	En el “entrelazado 1” P2 load aviso se ordena antes que P1 store aviso.
P1 store aviso	P2 load aviso	En el “entrelazado 2” P2 load aviso se ordena después que P1 store aviso
P2 load aviso	P2 load aviso	

Pregunta 4: *Muestre que es necesario serializar en un CC los accesos a una posición de memoria para un funcionamiento correcto (protocolo DL). Esto es hay que inferir el orden que ha determinado el CM. Para ello utilice los entrelazados previos.*

Pregunta 5: *Utilice el protocolo DL. Proponga secuencias de dos accesos a memoria donde se observen los cruces de peticiones en los estado LM^B y LM. Muestre diagramas temporales simplificados. Explique la forma de inferir el cruce y las acciones que se toman para mantener la coherencia.*

Pregunta 6: *Utilice el protocolo DL. Para las secuencias de accesos a memoria relacionadas seguidamente muestre diagramas temporales simplificados, donde se observen cruces tempranos.*

Orden de procesado en el CM		Estado del bloque en las caches y directorio
F	G	
P2 store t	P2 store t	C1, C2, C3: I ; directorio: 1 (L)
P3 load t	P2 PtXm t	
P1 store t	P1 store t	

Pregunta 7: *Utilice el protocolo DL. Para las secuencias de accesos a memoria relacionadas seguidamente muestre diagramas temporales simplificados, donde se observen cruces tempranos.*

Orden de procesado en el CM		Estado del bloque en las caches y directorio
H	I	Dos casos para H e I
P2 store t	P2 store t	1) C1, C2: I; C3: L; directorio: 3 (L)
P1 load t	P1 store t	2) C1: I; C2, C3: L; directorio: 2 ,3 (L)

Pregunta 8: *Utilice el protocolo DL. Para la secuencia de accesos a memoria mostrada seguidamente muestre diagramas temporales simplificados, donde se observen cruces tardíos.*

Orden de el CM	Estado del bloque en las caches y directorio
II	
P1 load t	C1, C2: I; C3: M; directorio: 3E (M)
P2 store t	

El cruce de una petición PtObE o PtObL con una petición PtXm en el CM es una oportunidad para reducir el trafico en la red. Por otro lado, se reduce la serialización en el CM.

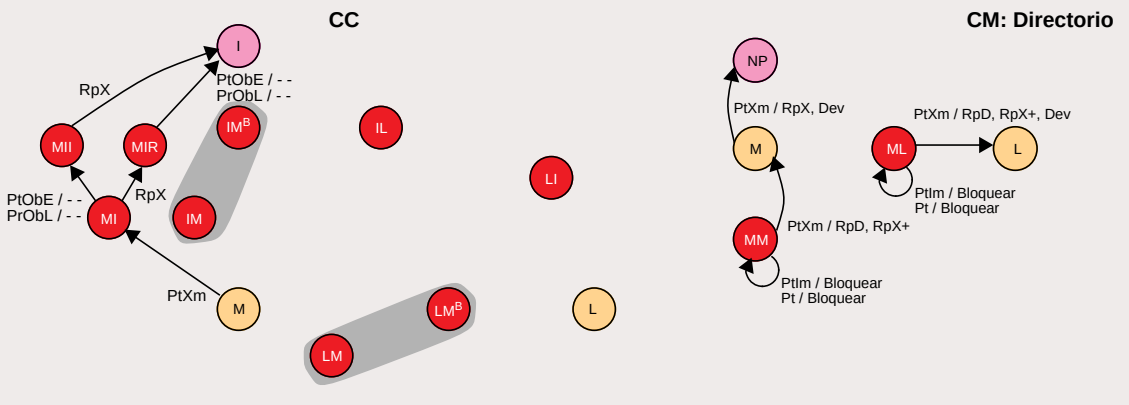
El mecanismo que diseñan los ingenieros es el siguiente:

- Un CM al procesar una petición PtXm en el estado M emite una respuesta al CC (RpX) y actualiza memoria.
- Un CC cuando emite una petición PtXm establece como estado del bloque MI. Un CC no responde a peticiones PtObE o PtObL cuando está en el estado MI. Además, para recordarlo, se utilizan estados transitorios.

- Un CM al procesar una petición PtXm (se transmite por la red de respuestas de los CC a los CM) utiliza esta petición como respuesta, si el bloque está en un estado transitorio en el directorio. También emite una respuesta al CC que ha emitido la petición PtXm, indicando que la ha utilizado como respuesta (RpX+).
- Cuando un CM utiliza una petición PtXm como respuesta emite una respuesta (RpD) al CC que ha establecido el estado transitorio del bloque en el directorio. Además, si es el caso, actualiza memoria.
- Un CC que ha emitido una petición PtXm, cuando recibe una respuesta RpX+ pasa el bloque al estado I, una vez ha recibido una petición PtObE o PtObL (antes o después).

A la modificación del protocolo DL que se acaba de describir la denominamos DLM.

Para el protocolo DLM los ingenieros diseñan los siguientes diagramas de transiciones entre estados de un bloque en el CC y el CM. .



Para comprobar los diagramas de transiciones entre estados utilizan las siguientes secuencias de accesos a memoria.

Orden de procesado en el CM			Estado del bloque en las caches y directorio
AA	BB	CC	
P2 store t	P2 store t	P2 load t	C1: M; C2, C3: I
P2 PtXm t	P3 load t	P3 load t	directorio: 1E (M)
	P2 PtXm t	P2 PtXm t	

Pregunta 9: *Muestre, utilizando las secuencias previas, que los diagramas de transiciones entre estado previos no son correctos. En concreto, en el CM no se serializan adecuadamente los accesos a la misma posición de memoria. Para ello tenga en cuenta que se pueden producir cruces entre mensajes.*

Para efectuar un análisis de la reducción de tráfico en las redes y de la reducción de serialización en el CM, aportada por el protocolo DLM, utilizaremos las siguientes secuencias de accesos a memoria.

Orden de procesado en el CM					Estado del bloque en las caches y directorio
J	K	L	M	N	
P2 store t	P2 store t	P2 load t	P2 load t	P2 load t	C1: M; C2, C3: I
P1 PtXm t	P3 store t	P1 PtXm t	P3 store t	P3 load t	
P3 store t	P1 PtXm t	P3 store t	P1 PtXm t	P1 PtXm t	directorio: 1E (M)

Pregunta 10: *Suponga que sólo se producen cruces en el CC1 y en el CM. Los mensajes del CM al CC1 llegan en el orden de emisión desde el CM. Muestre sendos diagrama temporales para el protocolo D y para el protocolo DLM.*

Pregunta 11: *Un CM utiliza el mensaje PtXm como una respuesta, cuando el estado del bloque en el directorio es transitorio. Entonces, un ingeniero se plantea que el CM no responda a mensajes PtXm. Justifique que no es una opción.*

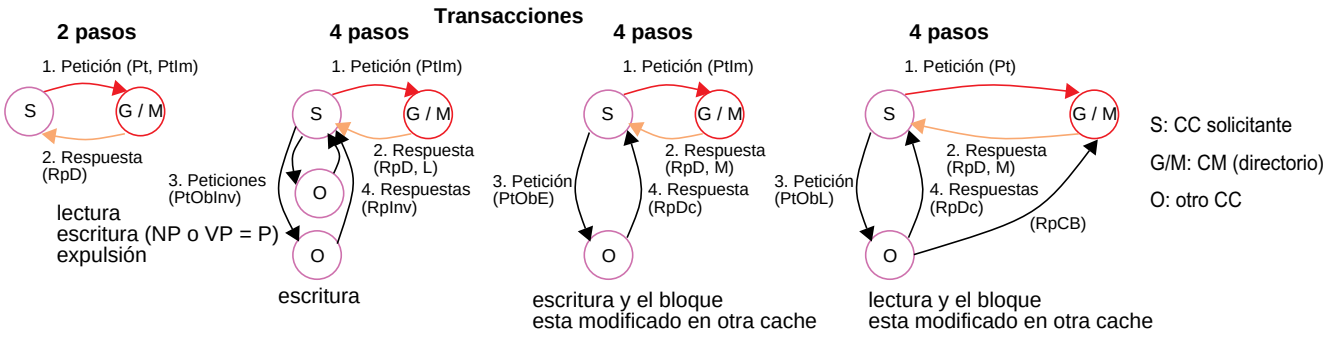
Pregunta 12: *Justifique la necesidad de que el CM, al responder al CC que ha emitido un mensaje PtXm, indique si ha utilizado el mensaje como respuesta. Para ello analice cruces de mensajes al procesarse la siguiente secuencia de accesos a memoria.*

Orden de procesado en el CM				Estado del bloque en las caches y directorio
O	P	Q	R	
P2 load t	P2 store t	P2 load t	P2 store t	C1: M; C2: I
P1 PtXm t	P1 PtXm t	P1 PtXm t	P1 PtXm t	directorio: 1E (M)
P1 load t	P1 load t	P1 store t	P1 store t	

Ejercicio 8.6

El protocolo de directorio que se describe utiliza una comunicación petición-respuesta estricta. Esto es, una petición al procesarse en un directorio genera una respuesta.

En estas condiciones, cuando es necesaria la participación de terceros en una acción de coherencia, el CM responde a un CC con la identidad de los participantes en la acción de coherencia. Entonces, el CC solicitante se encarga de efectuar las peticiones oportunas a los CC participantes y recolectar sus respuestas. En la siguiente figura se muestra el flujo de mensajes en las distintas situaciones.



El directorio es preciso. Cuando es necesaria la participación de terceros, en una acción de coherencia, son necesarios 4 pasos.

Mensajes. Los mensajes que se han añadido o modificado respecto al protocolo D se muestran en las siguientes tablas.

Controlador de coherencia (CC)	Controlador de memoria (CM)	Comentario
Mensajes de petición al CM	**Mensaje de respuesta**	
PtIm: petición de bloque op | dirección | Id	RpD, L: respuesta con el bloque y la lista de nodos que tienen copia del bloque en el estado L Id | bloque | Lista	Se lee el bloque de la memoria, se actualiza el directorio, se suministra el bloque y se indican los nodos que tienen copia del bloque (L).
	RpD, M: respuesta con el identificador del nodo que tiene el bloque en el estado M Id | bloque | IdM	Se actualiza el directorio, se suministra el nodo que tiene el bloque en el estado M.

Controlador de coherencia	Comentario
Respuestas	
RpCB: respuesta con el bloque al CM Id | bloque	El CC suministra el bloque al CM.

Se utiliza una red para transmitir peticiones (RP) y otra red para transmitir respuestas (RR) Las redes de interconexión no mantienen ningún orden entre los mensajes emitidos desde un nodo emisor a un nodo receptor.

Pregunta 1: *Partiendo de la descripción previa, diseñe los diagramas de transiciones entre estados de un bloque en un CC y en un CM. No tenga en cuenta posibles cruces de peticiones. No tenga en cuenta expulsiones de bloques.*

Suponga la siguiente secuencia de accesos a memoria.

Accesos A	Estado del bloque en las caches y directorio
P1 store t	C1, C2, C3: L;
P2 store t	directorio: 1, 2, 3 (L)
P3 load t	
P1 load t	

Pregunta 2: *Para la secuencia previa de accesos a memoria muestre un diagrama temporal simplificado. Un acceso a memoria no se empieza ha procesar hasta que el acceso previo ha finalizado completamente.*

Una transacción de coherencia se implementa mediante una secuencia de mensajes entre nodos. Entonces, mientras se está efectuando una transacción otro nodo puede iniciar otra transacción al mismo bloque. Por tanto, es necesario una serialización de las transacciones para mantener la coherencia.

Para constatar el problema un ingeniero utiliza las siguientes secuencias de accesos a memoria. En una petición PtXm o PtXI, el CC efectúa una transición a un estado transitorio (MI, LI) esperando la respuesta (RpX). Desde el estado MI se suministra el bloque si se recibe una petición. Desde el estado LI se responde RpInv.

Orden de procesado en el CM				Comentarios
B	C	D	E	
P2 load t	P2 store t	P3 store t	P3 store t	B: la respuesta a PtXm llega antes que la petición
P1 PtXm t	P1 PtXI t	P2 store t	P2 load t	C: la respuesta a PtXI llega antes que la petición
				D: C1, C2, C3 : L. Directorio: 1, 2, 3 (L)
				E: C1, C3 : L. C2: I. Directorio: 1, 3 (L)

Otras secuencias de acceso a memoria son:

Orden de procesado en el CM		Comentarios
D1	D2	
P3 store t	P3 store t	DD: C1, C2, C3 : L. C4: I. Directorio: 1, 2, 3 (L)
P2 store t	P2 store t	E: C1, C3 : L. C4: I. Directorio: 1, 3 (L)
P4 store t	P4 load t	

Pregunta 3: *Muestre diagramas temporales simplificados para las secuencias previas de accesos a memoria. Constate la necesidad de serialización, teniendo en cuenta desorden en los mensajes. Analice una serialización en el directorio.*

Una alternativa para solventar el problema descrito, es que un CC no pueda recibir peticiones a un bloque, si el CM ha procesado su petición. La forma de gestionarlo es que el CM reciba una notificación de que la transacción previa al bloque ha finalizado. Esto es, cuando un CC ha recibido respuesta a todas las peticiones que ha emitido a otros CC, emite un mensaje de notificación al CM (RpCF). También, si se permite cierto solapamiento, el CC que tiene el bloque en estado M puede enviar la notificación, después de servir la petición.

Pregunta 4: *Partiendo de la descripción previa, añada los mensajes necesarios al protocolo y diseñe los diagramas de flujo de mensajes. En el caso de existir alternativas tengalas en cuenta. No tenga en cuenta expulsiones de bloques.*

Pregunta 5: *Compare, desde el punto de vista de serialización en el procesado de transacciones al mismo bloque, el protocolo de este ejercicio con el protocolo D.*

Pregunta 6: *Partiendo de la descripción previa, añada los mensajes necesarios al protocolo y diseñe los diagramas de transiciones entre estados de un bloque en un CC y en un CM. En el caso de existir alternativas proponga diagramas para cada una de ellas. No tenga en cuenta expulsiones de bloques.*

Pregunta 7: *Para la secuencia de accesos a memoria previa a la segunda pregunta muestre un diagrama temporal simplificado. Todos los accesos son concurrentes excepto el último que debe ser causal. El orden en el cual se procesan en el CM es el orden de la secuencia de accesos.*

Pregunta 8: *Diseñe las transiciones entre estados en un CC y en un CM para tener en cuenta la expulsión de bloques. El directorio es preciso.*