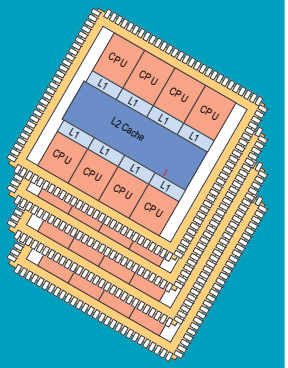
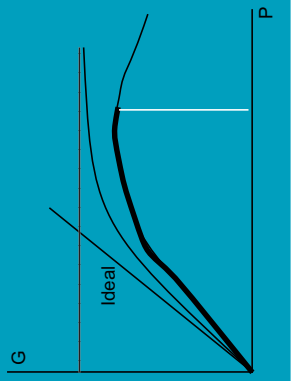
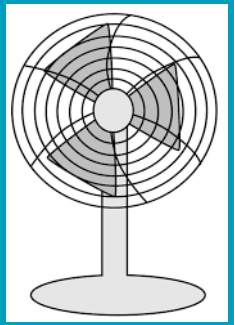






Multiprocesadores

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

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.

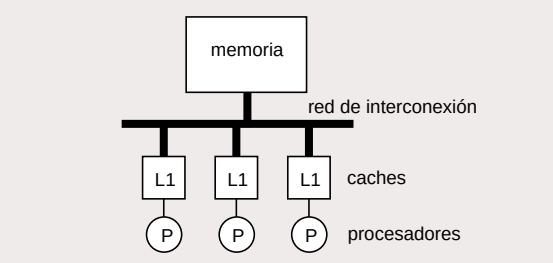


Figura 4.1 *Sistema multiprocesador de memoria compartida.*

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).

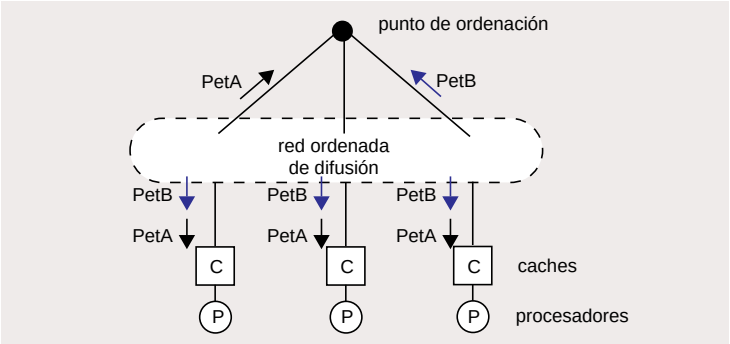


Figura 4.2 Esquema conceptual de una red ordenada.

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.

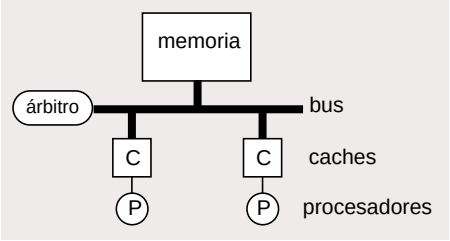
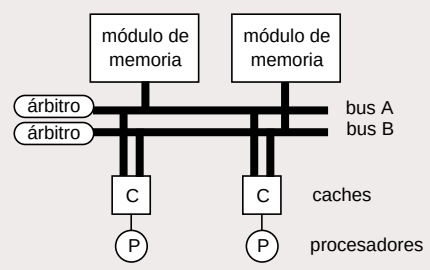


Figura 4.3 Multiprocesador con un bus como red de interconexión.

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).

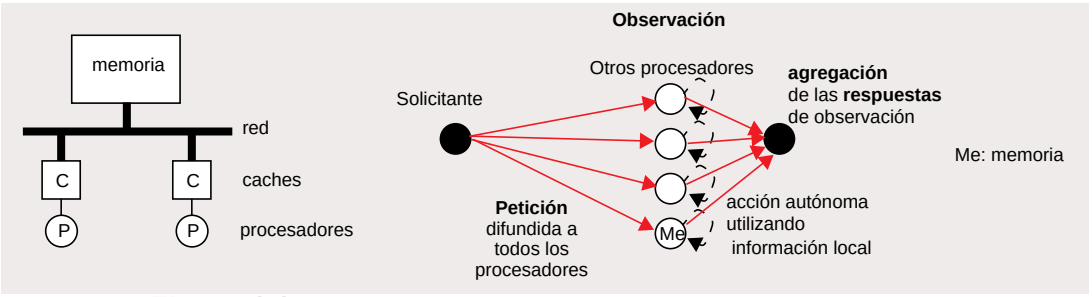


Figura 4.4 Protocolo de observación.

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).

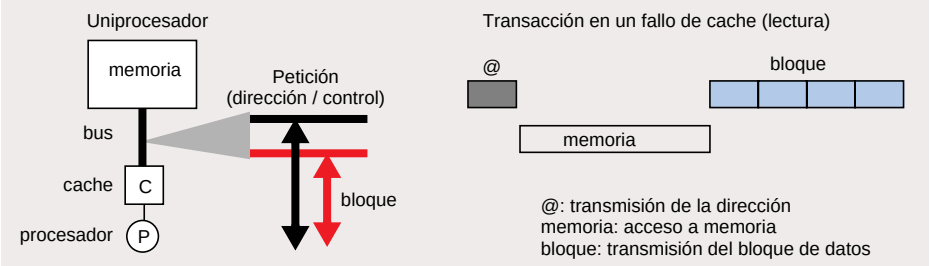


Figura 4.5 Uniprocesador y transacción en un fallo de cache.

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).

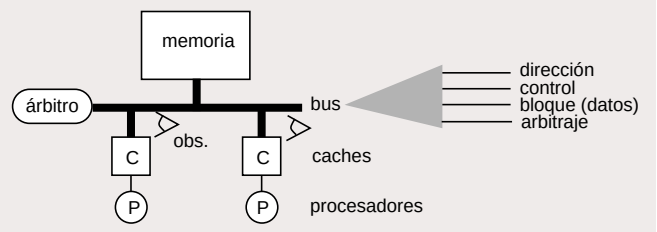


Figura 4.6 Multiprocesador que utiliza un bus como red de interconexión.

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.

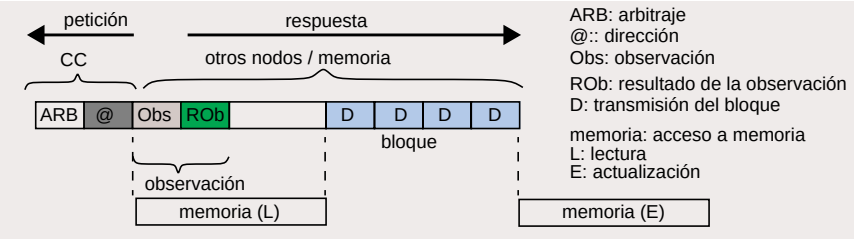


Figura 4.7 Fases en una transacción de bus. La ocupación del bus es independiente del tipo de acceso (lectura o escritura).

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.

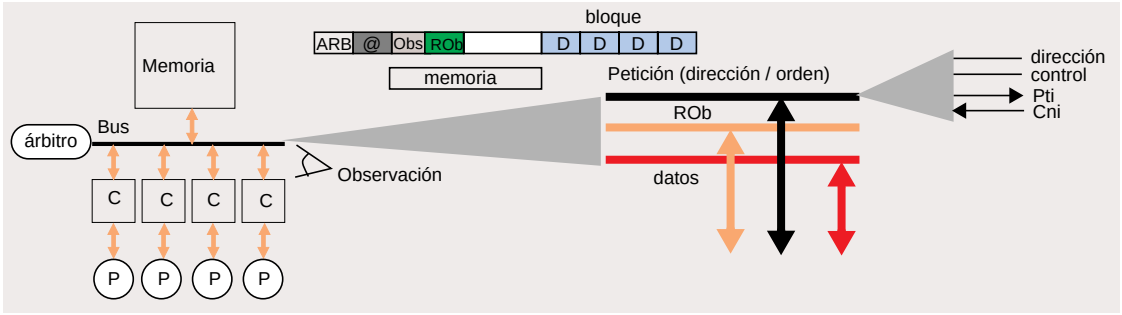
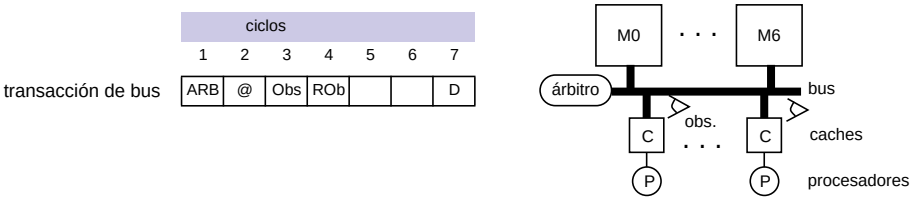


Figura 4.8 Grupos de señales en un bus. Las señales Pti y Cni son respectivamente las señales de petición de bus y concesión de bus. Hay un par de ellas por cada procesador.

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].

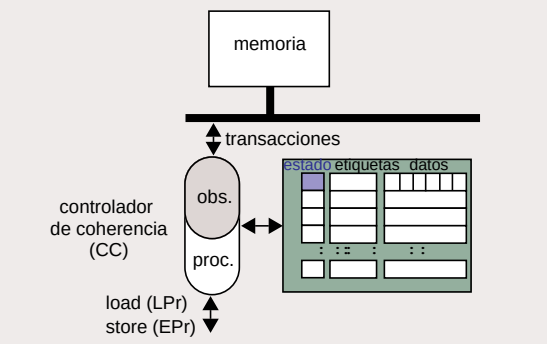


Figura 4.9 Controlador de coherencia: agentes observador y procesador.

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.

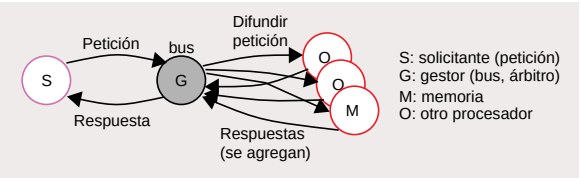


Figura 4.10 Punto de ordenación y fases en una transacción de bus.

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].

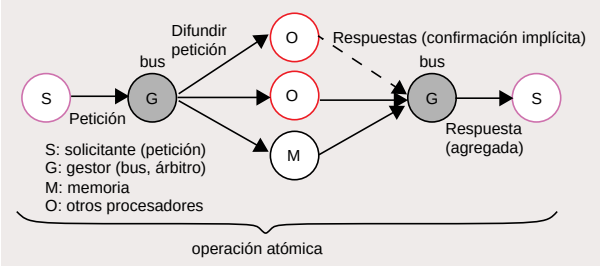


Figura 4.11 Acceso a memoria que requiere una transacción de bus. Operación atómica.

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.

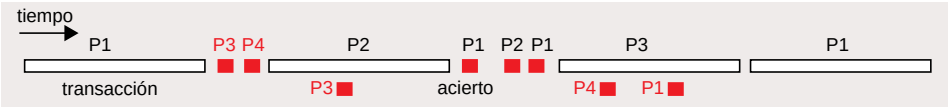


Figura 4.12 Diagrama temporal de accesos a memoria (load o store). Un mecanismo garantiza que sólo existe una transacción en curso en un instante determinado.

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].

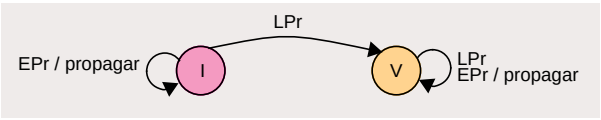


Figura 4.13 Protocolo VI: Transiciones debidas a accesos a memoria del propio procesador.

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.

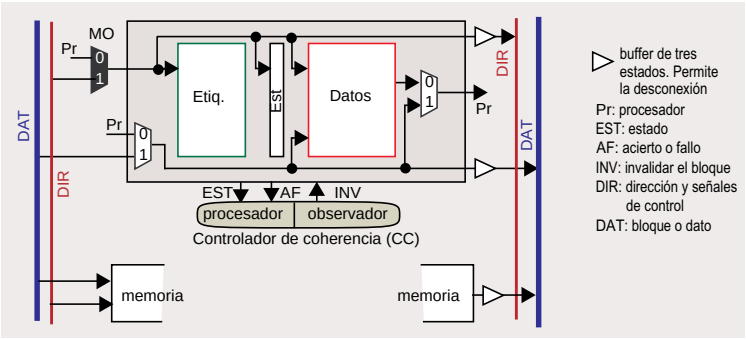


Figura 4.15 Camino de datos simplificado de una cache con escritura inmediata.

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.

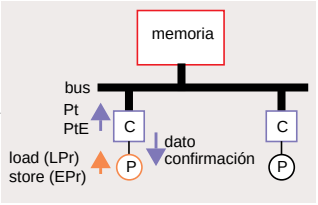


Tabla 4.2 Protocolo VI. Peticiones del procesadores y respuestas.

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

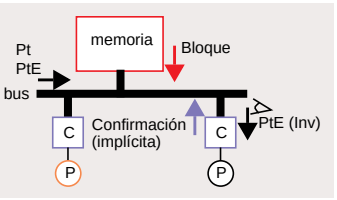


Tabla 4.3 Protocolo VI. Peticiones del CC a memoria, respuestas de memoria y de los otros CC y acciones del CC.

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.

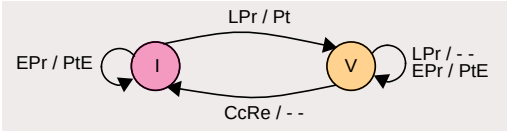


Figura 4.16 Protocolo VI. Transiciones entre estados inducidas por peticiones del procesador.

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.

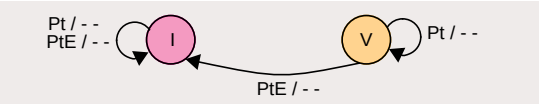


Figura 4.17 Protocolo VI. Transiciones entre estados inducidas por observaciones.

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).

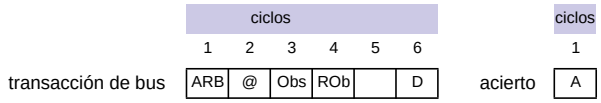


Figura 4.18 Especificación temporal de una transacción y de un acierto en cache.

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.

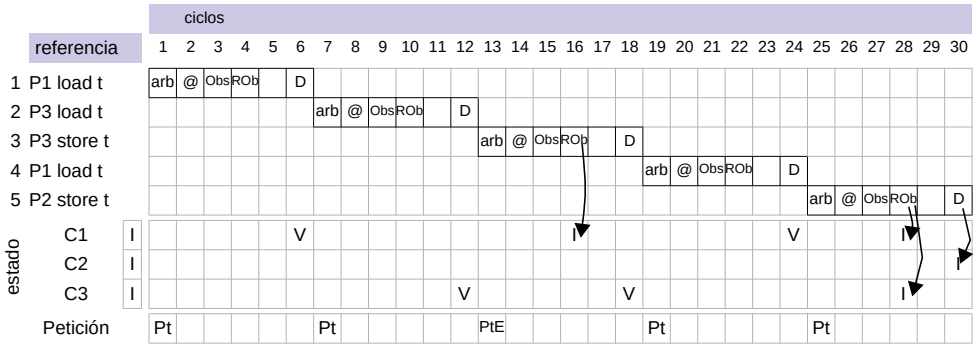


Figura 4.19 Protocolo VI. Diagrama temporal de transacciones y cambios de estado.

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.

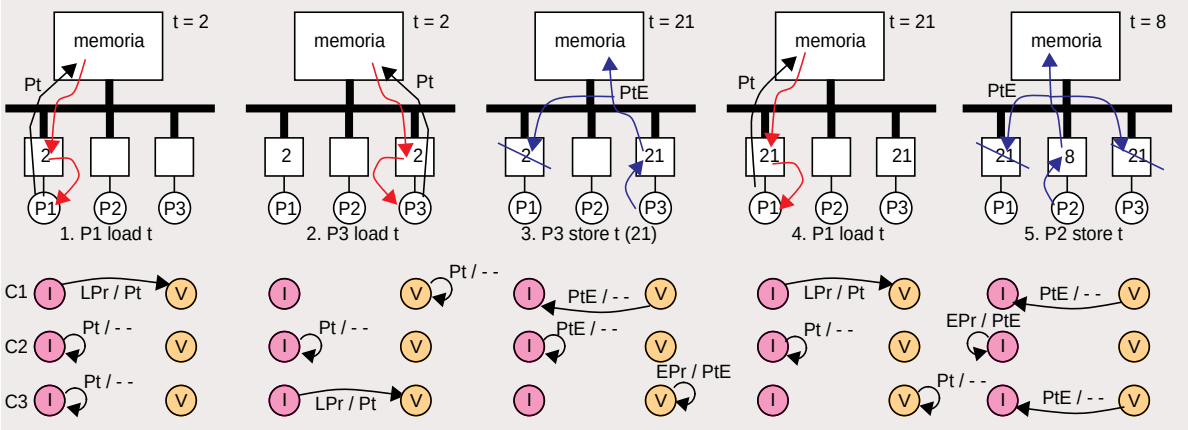


Figura 4.20 Protocolo VI. Fotogramas de las peticiones y respuestas y cambios de estado.

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.

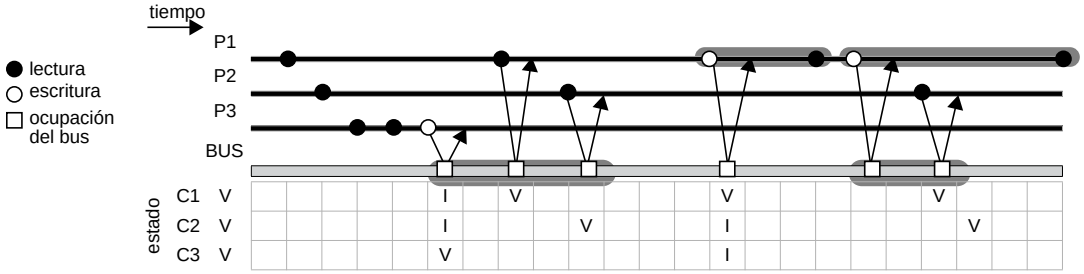


Figura 4.21 Protocolo VI: Verificación no formal de la coherencia de cache. Accesos a la misma posición de memoria.

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).

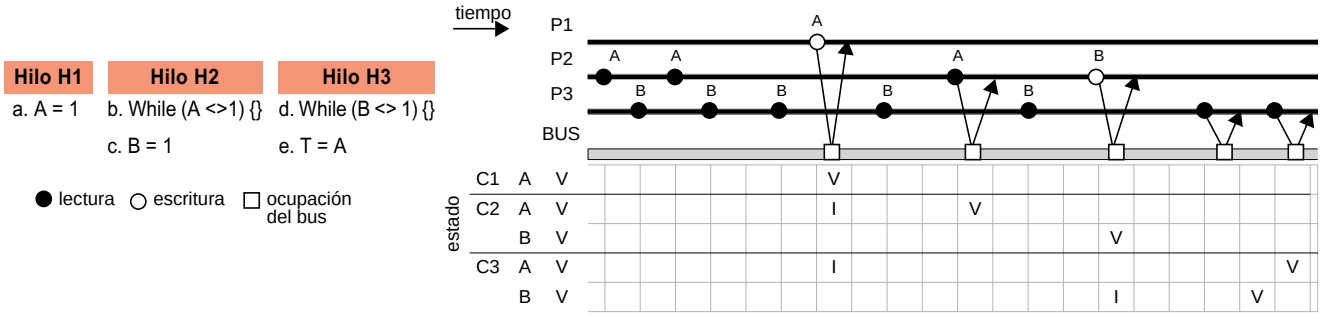


Figura 4.22 Protocolo VI. Verificación no formal de la consistencia de memoria.

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.

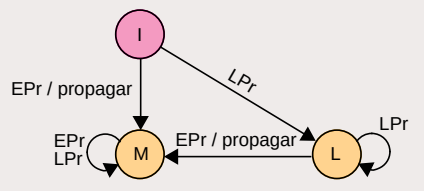


Figura 4.23 Protocolo MLI. Transiciones debidas a accesos a memoria del propio procesador.

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.

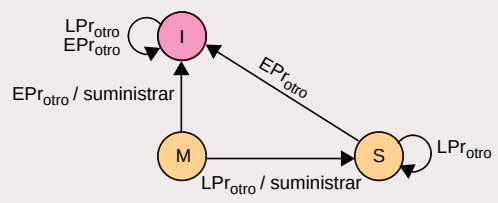


Figura 4.24 Protocolo MLI. Transiciones entre estados inducidas por accesos a memoria de otros procesadores.

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.

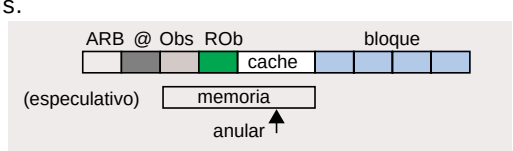


Figura 4.25 Transacción con acceso especulativo a memoria y suministro desde una cache.

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.

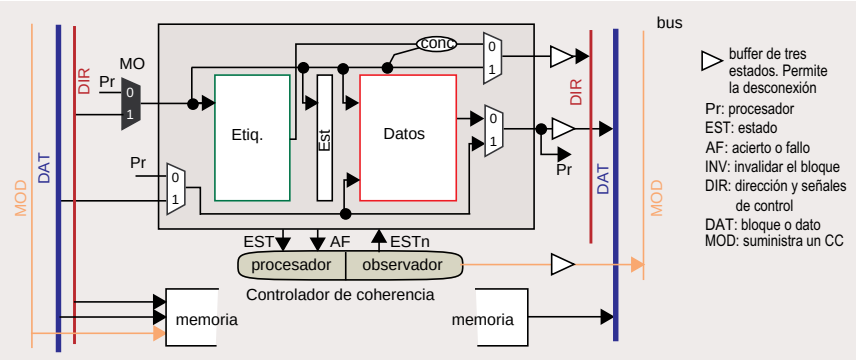


Figura 4.26 Señales de bus y camino de datos de una cache con escritura retardada.

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.

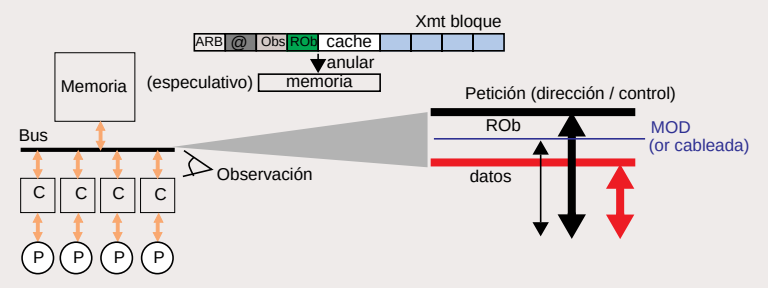


Figura 4.27 Bus: cable de respuesta de observación.

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).

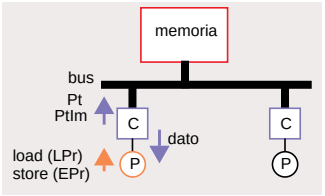


Tabla 4.7 Protocolo MLI. Peticiones del procesador y respuestas.

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.

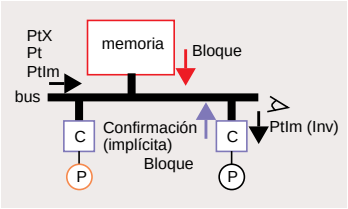


Tabla 4.8 Protocolo MLI. Peticiones de un CC a memoria, respuestas de memoria y de los otros CC y acciones del CC y memoria.

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).

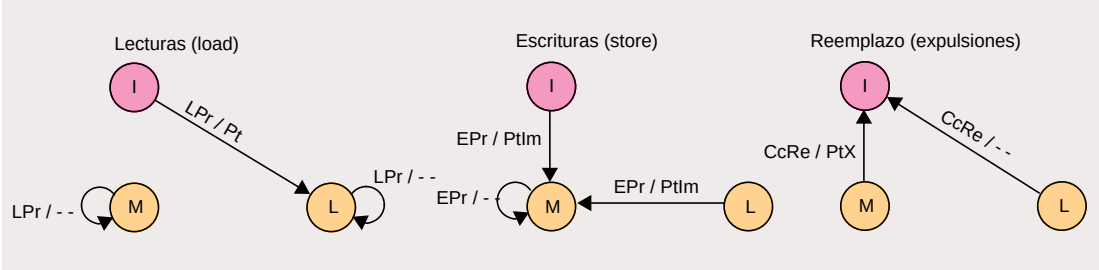


Figura 4.28 Protocolo MLI. Transiciones entre estados debidas a eventos del procesador y acciones de reemplazo.

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.

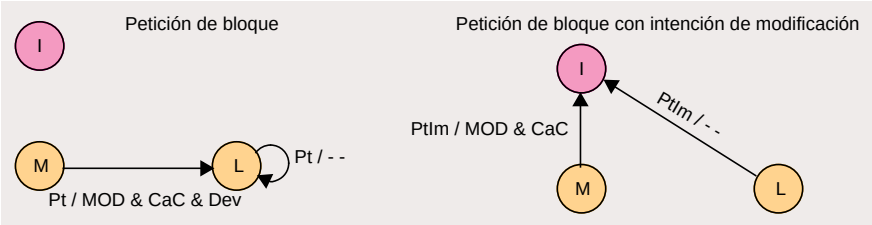


Figura 4.29 Protocolo MLI. Transiciones entre estados inducidas por observaciones.

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.

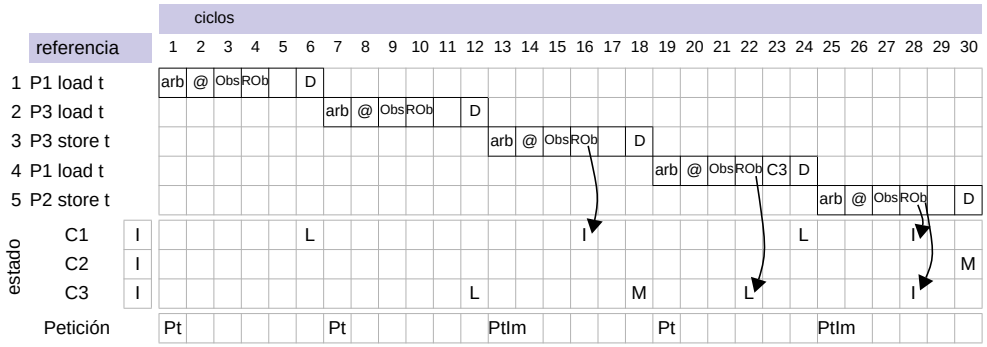


Figura 4.30 Protocolo MLI. Diagrama temporal: transacciones y cambios de estado en una secuencia de accesos a memoria.

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.

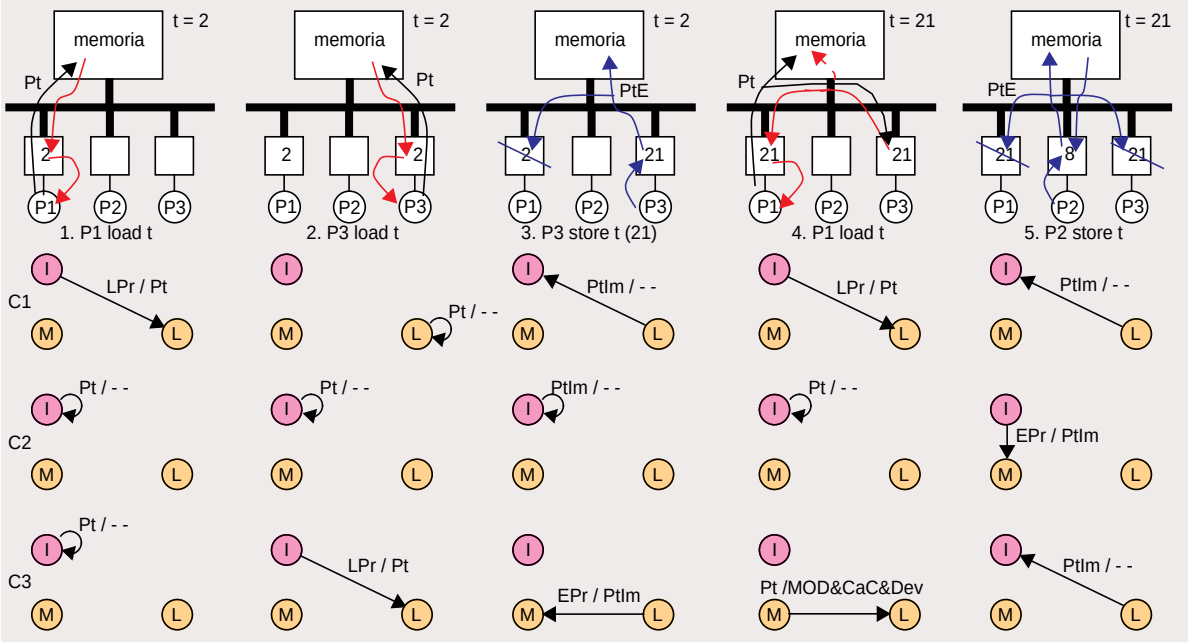


Figura 4.31 Protocolo MLI. Fotogramas de las peticiones y respuestas y cambios de estado en una secuencia de accesos a memoria.

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.

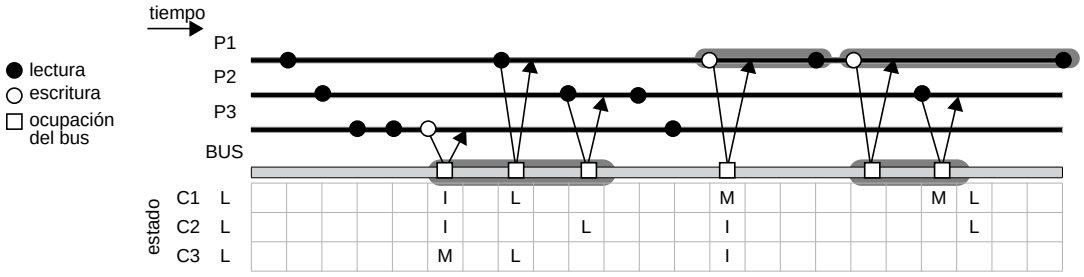


Figura 4.32 Protocolo MLI. Verificación no formal de la coherencia de cache. Accesos a la misma posición de memoria.

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).

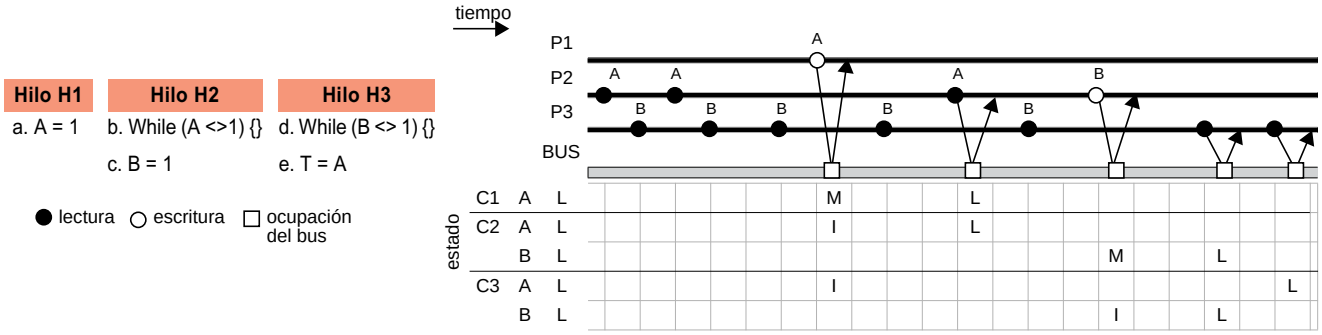


Figura 4.33 Protocolo MLI. Verificación no formal de la consistencia de memoria.

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.

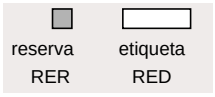
	bus	mem.		C 1			C 2		
acceso	trans.	var.	sum.	cont.	var.	est.	cont.	var.	est.
1. P1 load A1	Pt	A1	mem.	1	A1, A2	L			
2. P2 load A2	Pt	A2	mem.				1	A1, A2	L
3. P1 store A1	PtIm	A1	mem.	1	A1, A2	M	1	A1, A2	I
4. P2 load A2	Pt	A2	C 1	1	A1, A2	L	1	A1, A2	L
5. P2 store A2	PtIm	A2	mem.	1	A1, A2	I	1	A1, A2	M

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.

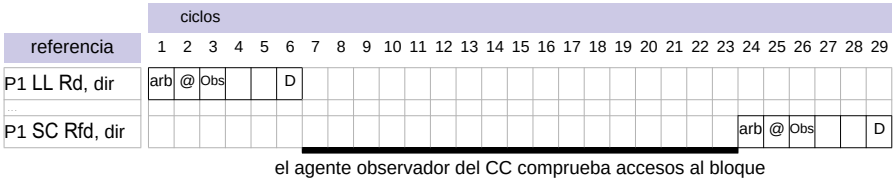


Figura 4.35 Controlador de coherencia: actuación del agente observador para determinar si la ejecución en secuencia de las instrucciones LL y SC se puede considerar atómica.

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.

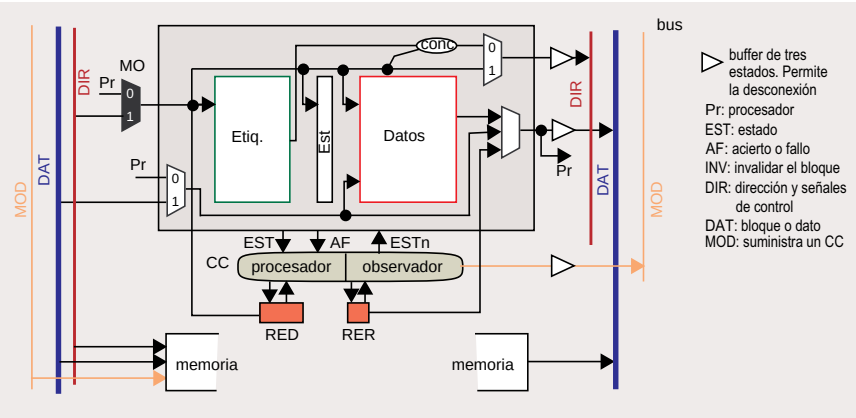


Figura 4.36 Esquema del camino de datos de una cache con registro de enlace.

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.

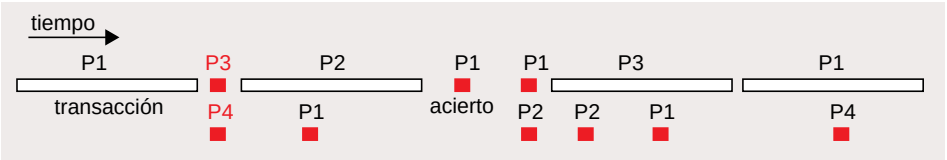


Figura 4.37 Concurrencia de un acceso a memoria y aciertos en cache.

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).

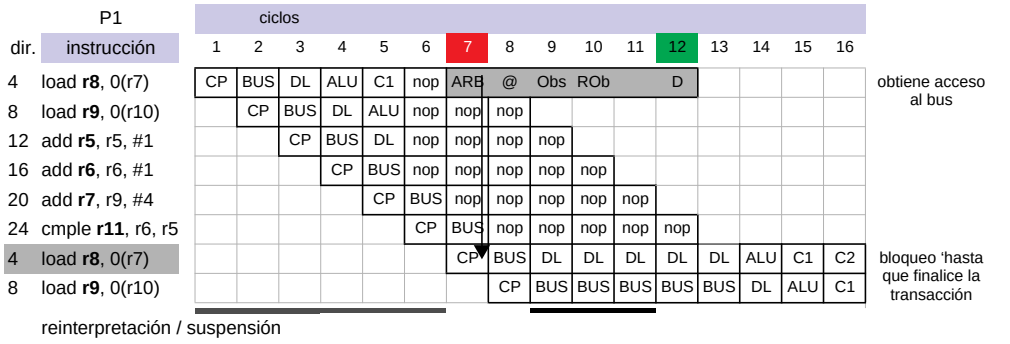


Figura 4.38 Diagrama temporal de un procesador que detecta un fallo de cache y obtiene acceso al bus.

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.

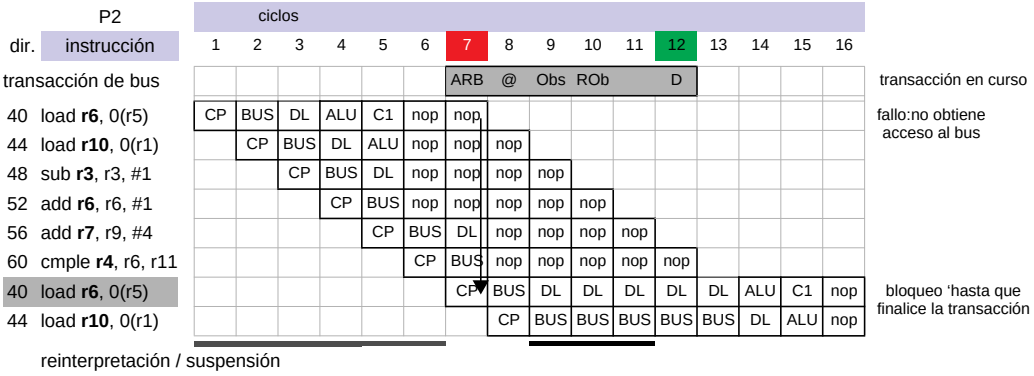


Figura 4.39 Diagrama temporal de un procesador que detecta un fallo de cache y no obtiene acceso al bus.

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.

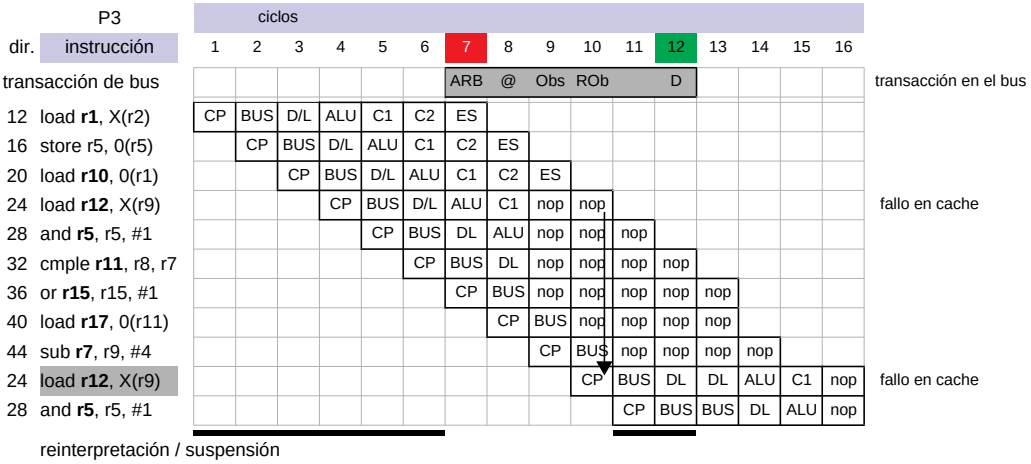


Figura 4.40 Diagrama temporal de un procesador que interpreta instrucciones hasta que detecta un fallo de cache y hay una transacción en curso.

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.

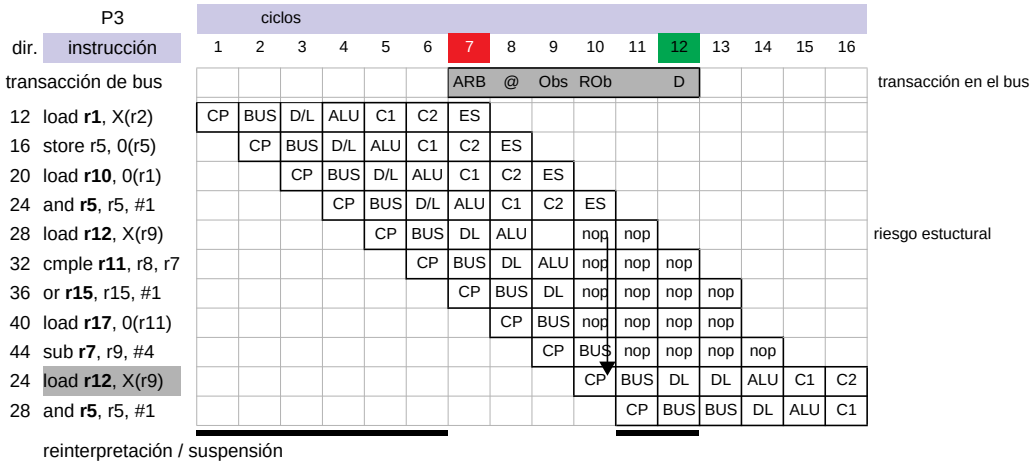


Figura 4.41 Diagrama temporal de un procesador que interpreta instrucciones hasta que detecta un riesgo estructural con una transacción en curso.

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.

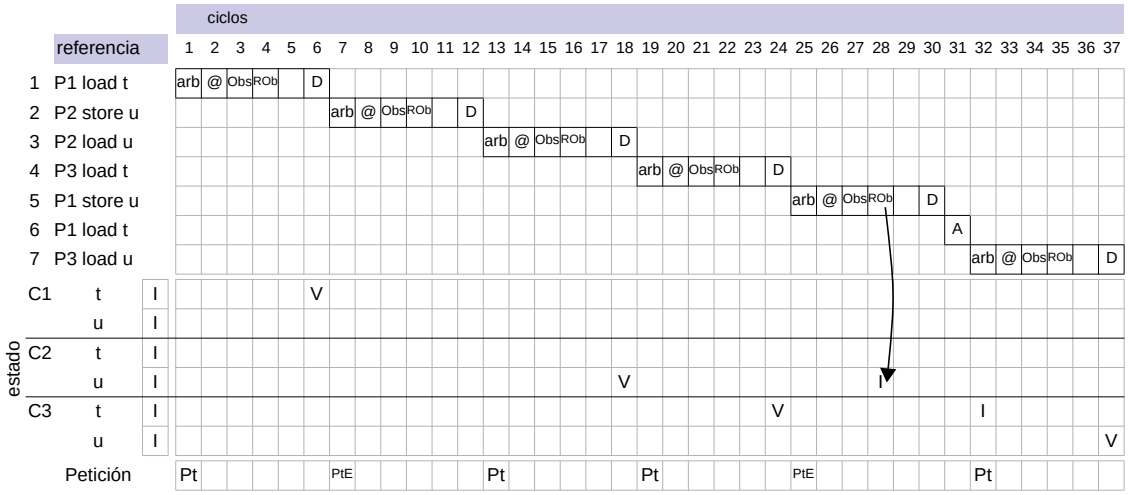
El sexto acceso a memoria es de lectura, lo efectúa el procesador P1 y es un acierto. El séptimo acceso a memoria también es una lectura, efectuada por el procesador P3. Como hay que asignar un contenedor y está ocupado por otro bloque, hay que expulsar el bloque que almacena la variable t. Después de 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).

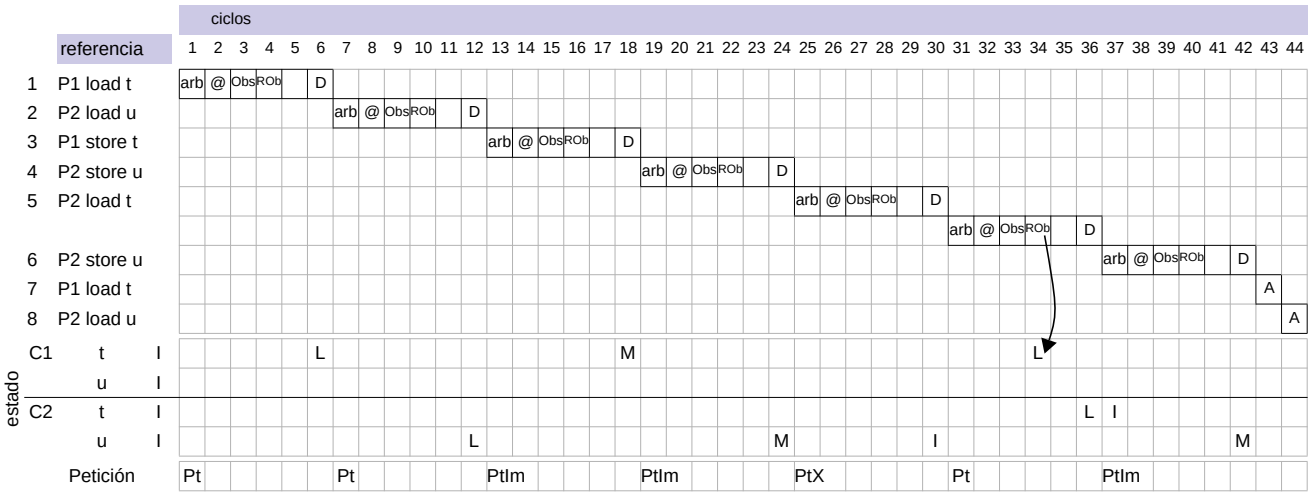
Seguidamente se inicia una transacción de petición de bloque con intención de modificación. Al finalizar la transacción el estado del bloque es M. Los dos últimos accesos a memoria son operaciones de lectura que son aciertos en cache. En el primer caso el bloque está en estado L y en el segundo el bloque 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.

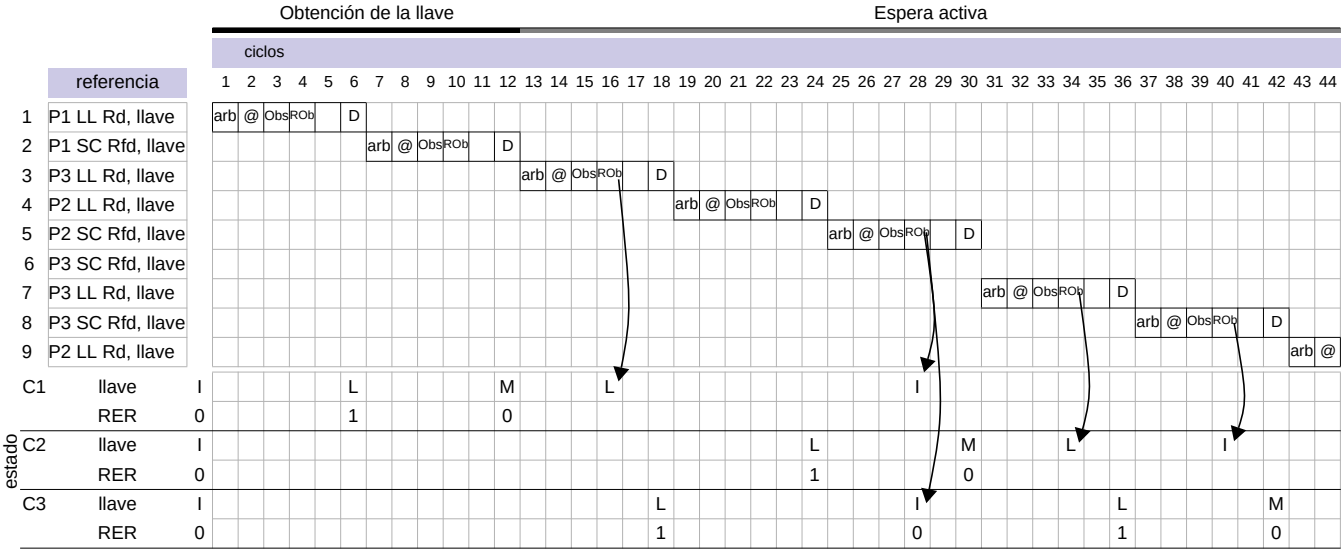
Pregunta 1: *Muestre, mediante una tabla, las transacciones de bus y estado de los bloques en las caches. Muestre también el valor del bit de enlace en cada cache (RER).*

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).*

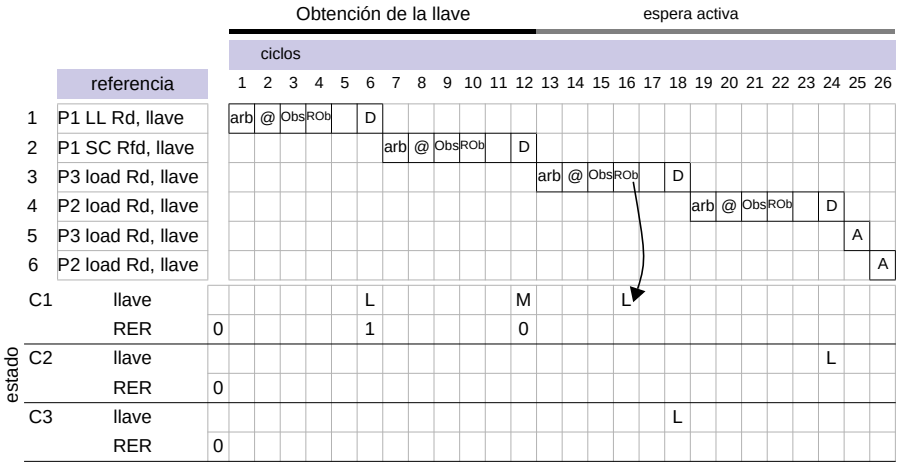
Respuesta: La instrucción LL del procesador P1 acierta en cache y activa el bit de enlace. Al ejecutar el mismo procesador la instrucción SC efectúa la operación intercambio.

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

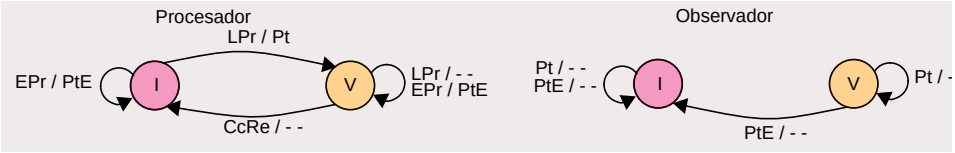
Descripción de un protocolo de observación VI denominado A

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

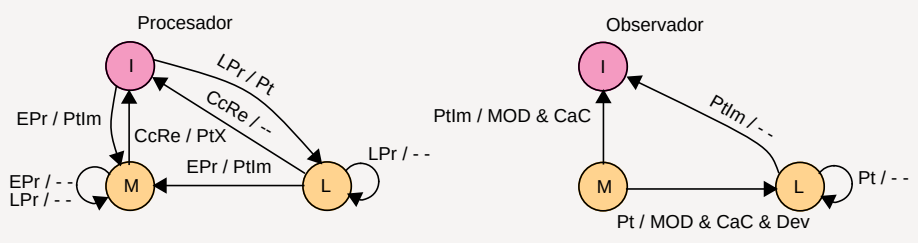
Un multiprocesador utiliza un bus como red de interconexión. Las caches privadas utilizan escritura retardada con asignación de contenedor en fallo. El multiprocesador utiliza la técnica de invalidación para mantener la coherencia. En cada contenedor de cache se dispone de dos bits para identificar los 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

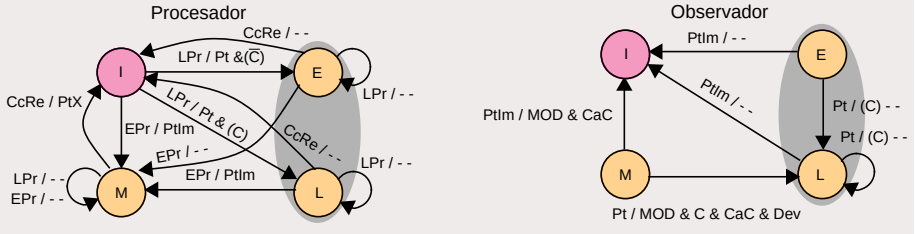
Un multiprocesador dispone de caches privadas que utilizan escritura retardada y la red de interconexión es un bus. Para mantener la coherencia entre las cache privadas se utiliza la técnica de invalidación. El protocolo de coherencia de cache se denomina MELI y tiene cuatro estados (inválido (I), exclusivo (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.

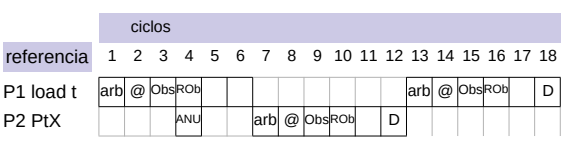
Un procesador inicia los accesos a cache en orden de programa y la cache es bloqueante.

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.

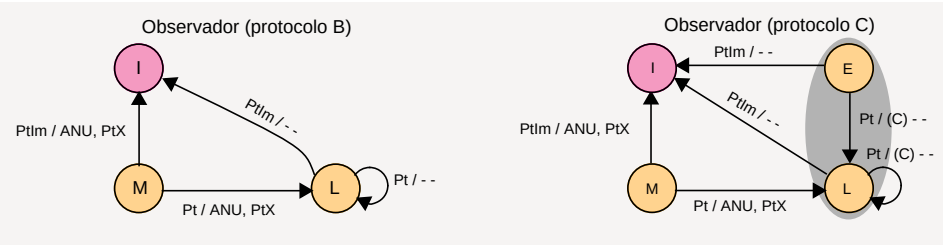
Por intervención indirecta se entiende que una cache nunca suministra un bloque directamente a otra cache. En su lugar, la cache que tiene el bloque en estado M anula la transacción en curso y posteriormente actualiza memoria. Se espera que la actualización de memoria sea antes de que la cache, cuya 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.

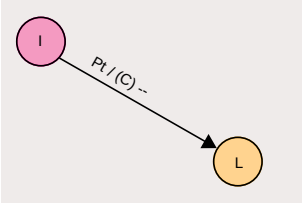
	bus				
acceso	**trans.**	**señal**			**comentario**
P5 load t	Pt	anu C4	. . .		transacción de P5 anulada por C4
	PtX		. . .		actualización de memoria por parte de C4
	Pt		. . .		se repite la transacción de P5

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.

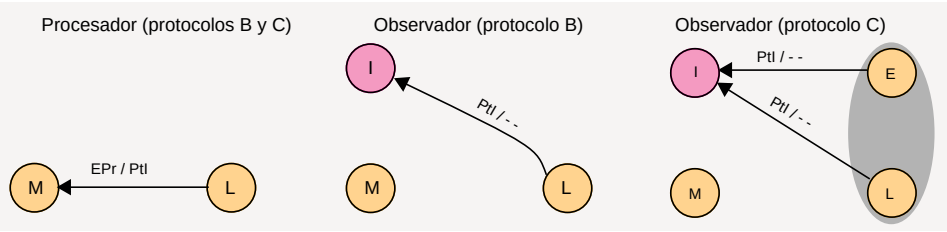
Transacción de exclusividad (petición PtI). Solicitud de acceso a un bloque en exclusividad partiendo del estado L

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
```

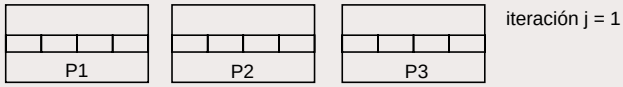
El tamaño de las caches es suficientemente grande para que no existan fallos de capacidad (todas las estructuras de datos caben en la cache). El valor de N es 16, la matriz se almacena por columnas y las direcciones de inicio de las estructuras de datos son 0 para la matriz A(N,N), 256 para el vector 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.*

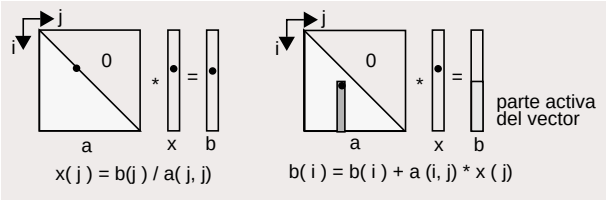
	Bus	Memoria			Cache 1			Cache 2		
acceso	trans.	var.	val.	suministro	cont. / varia	valor	estado	cont. / varia	valor	estado

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.

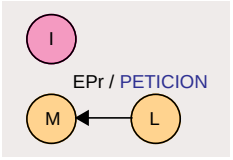
acceso	acceso	memoria (valor inicial)	Caches (almacena)	
			C1	C2
1. P1 load t	6. P1 load t	u: 4	no	no
2. P2 store r (34)	7. P1 store v (37)	t: 5	no	no
3. P1 store t (25)	8. P2 store u (17)	v: 12	no	no
4. P1 store u (71)	9. P1 store r (64)	r: 13	no	no
5. P2 store t (42)	10. P2 load t			

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.*

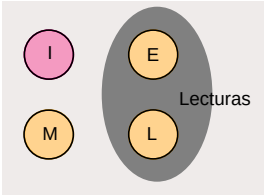
Ejercicio 4.11



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.

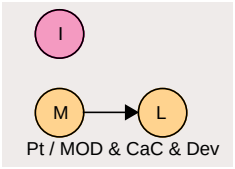
accesos	accesos	memoria (valor inicial)	cache(almacena)		
			C1	C2	C3
1. P1 lee u	5. P3 escribe u (18)	u: 4	u: no	u: no	u: no
2. P2 lee t	6. P1 lee u	t: 5	t: no	t: no	t: no
3. P1 escribe u (32)	7. P2 lee u				
4. P2 escribe t (67)					

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.

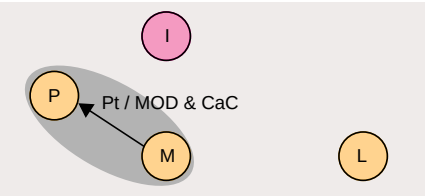


En este ejercicio supondremos que todos los protocolos son del tipo invalidación.

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.

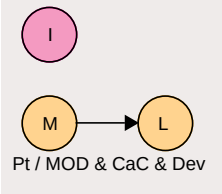
Pregunta 1: *En una cache indique el número mínimo y máximo de comparaciones que se efectúan en la fase de observación de una transacción.*

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.*

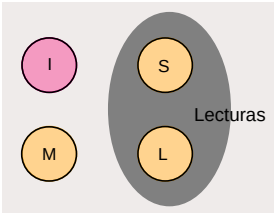
Ejercicio 4.16

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).*

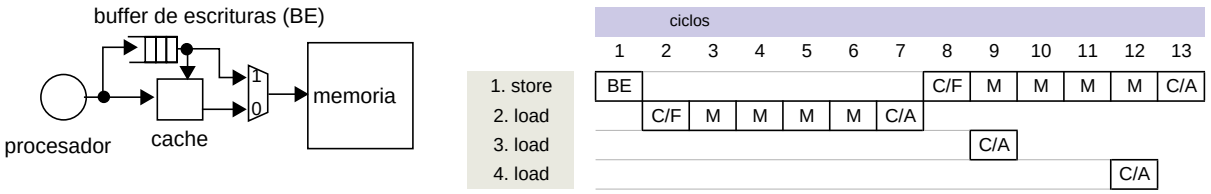
Ejercicio 4.21

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.

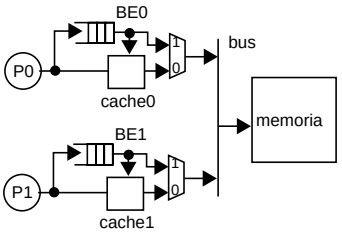
Para que el BE sea transparente a la arquitectura se añade circuitería a su alrededor. Si una instrucción load más joven accede a una posición de memoria almacenada en el BE, el procesador obtiene el valor de la instrucción store más joven que accede a la misma posición de memoria almacenada en el BE. 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.

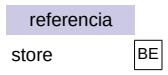
P 0	P 1	valores iniciales
aviso1 = 1 if (aviso2 = 0) R.Critica aviso1 = 0	aviso2 = 1 if (aviso1 = 0) R.Critica aviso2 = 0	aviso1 = 0 aviso2 = 0

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.

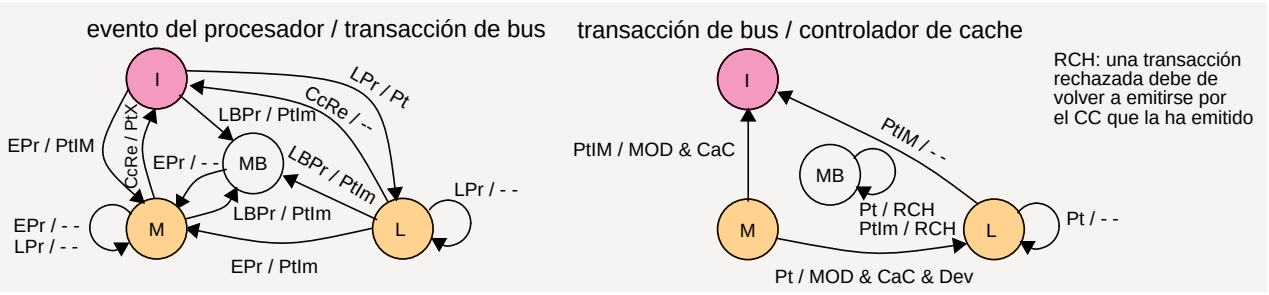


En las preguntan que restan se supone que el BE tiene la misma funcionalidad que en el caso descrito de un procesador. En estas condiciones, en cada procesador pueden competir por el bus: a) la entrada en la cabeza del BE y b) una instrucción load que ha detectado un fallo al acceder a cache. En este 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

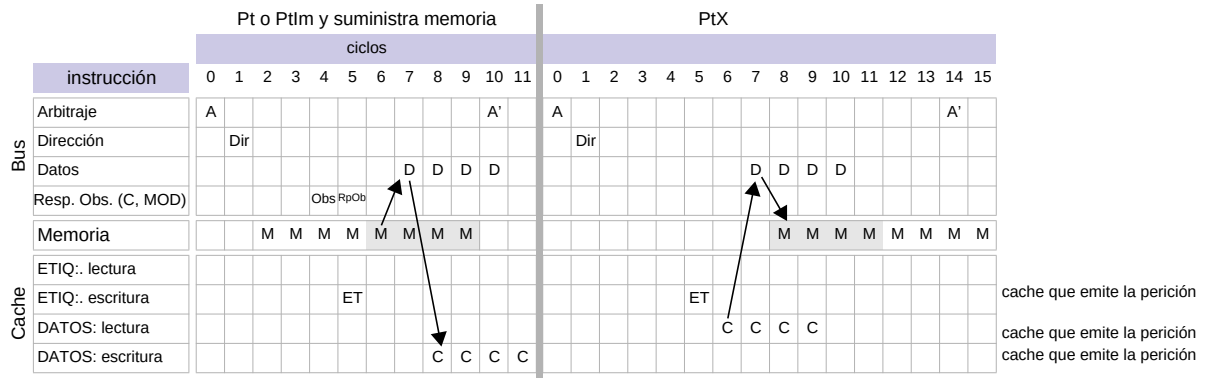
El protocolo de coherencia que se utiliza es el denominado B.

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.

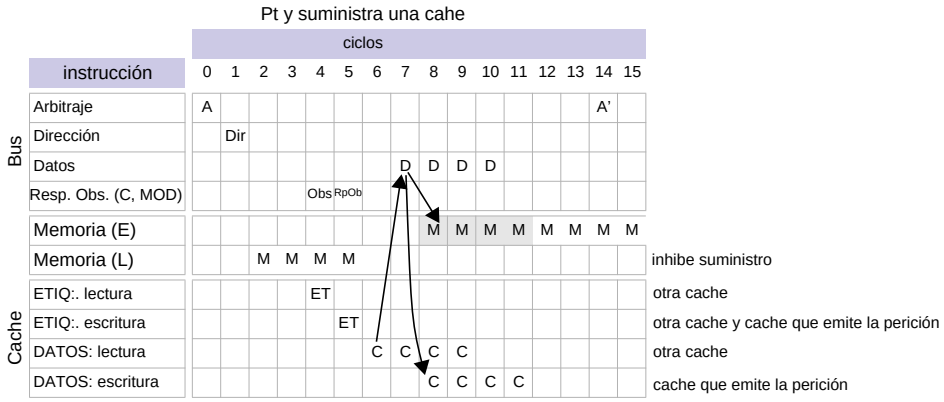


En una petición Pt siempre se accede a memoria aunque posteriormente no suministre el bloque, ya que lo suministra una cache. En una petición PtIm el comportamiento es el mismo que en una petición Pt, pero las caches involucradas en la acción de coherencia invalidan la copia del bloque. En una petición 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.*

.