Bigbounze
www.wuolah.com/student/Bigbounze

Tema 3.pdf

Resumenes Por Temas

2° Arquitectura de Computadores

Grado en Ingeniería Informática

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

Tema 3

- Lección 7

. Clasificación de arquitecturas con TLP explícito y una instancia de SO

- Multiprocesador: Ejecutan varios threads en paralelo en un computador con varios cores (cada thread en un core distinto).
- Multiprocesador en un chip o CMP: ejecutan varios threads en paralelo en un chip de procesamiento multicore (cada thread en un core distinto).
- Core multithread: core que modifican su arquitectura ILP para ejecutar threads concurrentemente o en paralelo.

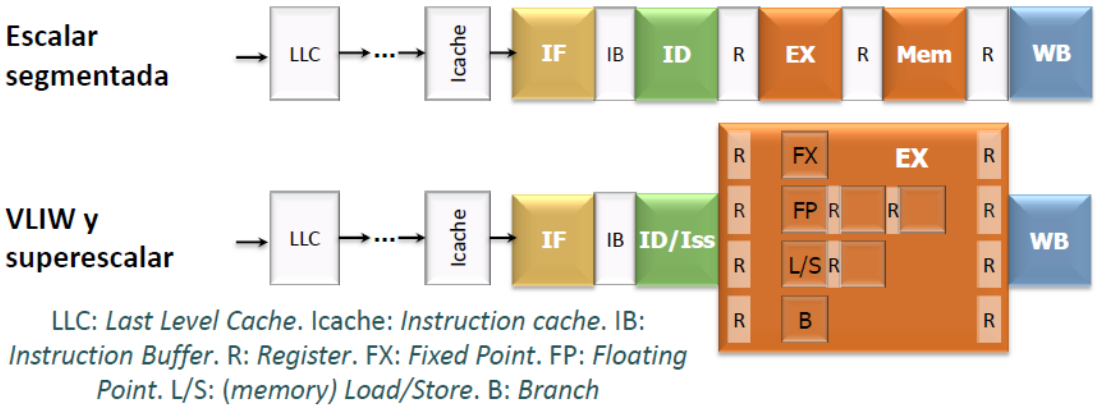
. Multiprocesadores

Existen dos criterios de clasificación, por sistemas de memoria y por nivel de empaquetamiento.

- Sistemas de memoria
    - Multiprocesador con memoria centralizada (UMA):
        - Mayor latencia, poco escalable.
        - Controlador de memoria en chipset.
        - Red: bus (medio compartido)
    - Multiprocesador con memoria distribuida (NUMA):
        - Menor latencia, escalable, pero requiere para ello distribución de datos/código.
        - Controlador de memoria en chip del procesador.
        - Red: enlaces (conexiones punto a punto) y conmutadores (en el chip del procesador).
- Nivel de empaquetamiento (de mayor a menor):
    - Sistema
    - Armario
    - Placa
    - Chip

. Cores Multithread

- Arquitecturas ILP



LLC: *Last Level Cache*. Icache: *Instruction cache*. IB: *Instruction Buffer*. R: *Register*. FX: *Fixed Point*. FP: *Floating Point*. L/S: *(memory) Load/Store*. B: *Branch*

- Etapa de captación de instrucciones (**I**nstruction **F**etch)
- Etapa de decodificación de instrucciones y emisión a unidades funcionales (**I**nstruction **D**ecode)
- Etapa de ejecución (**Ex**ecution)
- Etapa de acceso a memoria (**Mem**ory)
- Etapa de almacenamiento de resultados (**W**rite-**B**ack)
- Procesadores segmentados: ejecutan instrucciones concurrentemente segmentando el uso de sus componentes
- Procesadores VLIW (Very Large Instruction Word) y superescalares: ejecutan instrucciones concurrentemente (segmentación) y en paralelo (tienen múltiples unidades funcionales y emiten múltiples instrucciones en paralelo a unidades funcionales)
  - VLIW: Las instrucciones que se ejecutan en paralelo se captan juntas de memoria. Este conjunto de instrucciones conforma la palabra de instrucción muy larga a la que hace referencia la denominación VLIW. El hardware presupone que las instrucciones de una palabra son independientes: no tiene que encontrar construcciones que pueden emitirse y ejecutarse en paralelo.

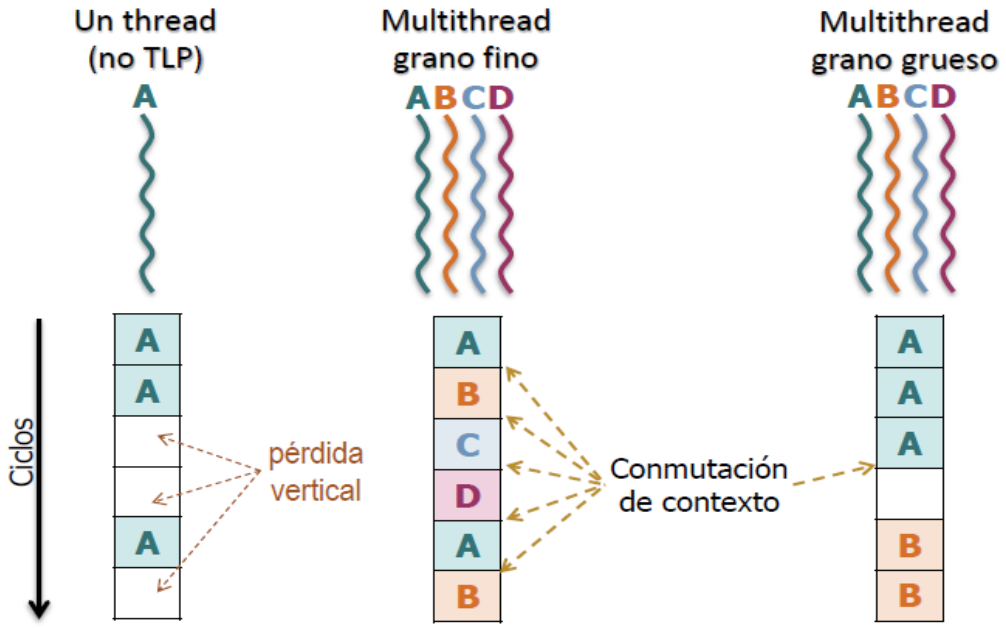
  - Superescalares: Tiene que encontrar instrucciones que puedan emitirse y ejecutarse en paralelo (tiene hardware para extraer paralelismo a nivel de instrucción.

- El almacenamiento y el hardware se multiplexa, se reparte o se comparte entre threads.

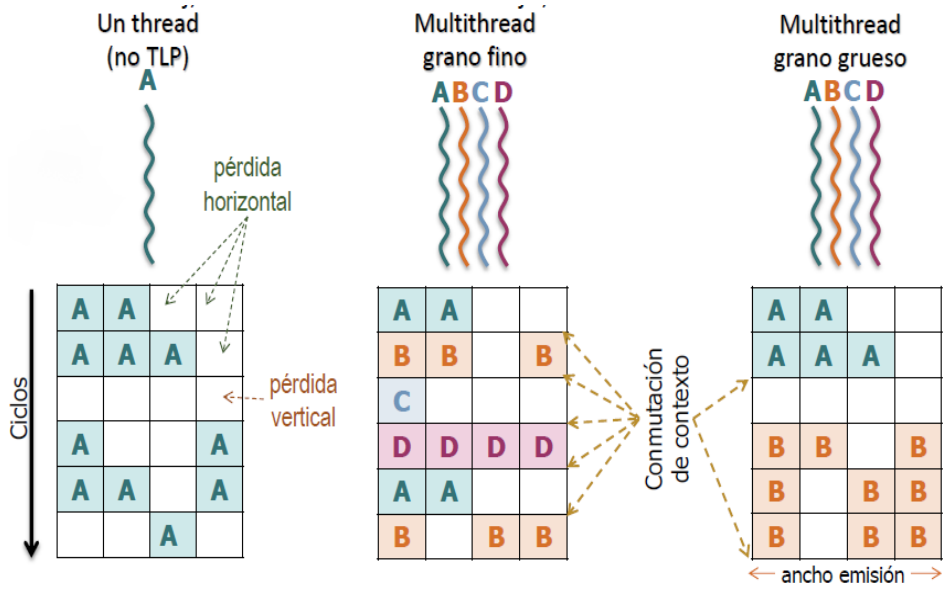
- Clasificación de cores Multithread
  - Temporal Multithreading (TMT)
    - Ejecutan varios threads concurrentemente en el mismo core.
    - La conmutación entre threads la decide y controla el hardware
    - Emite instrucciones de un único thread en un ciclo
  - Simultaneous Multitheading (SMT)
    - Ejecutan, en un core superescalar varios threads en paralelo
    - Pueden emitir instrucciones de varios threads en un ciclo
- Clasificación de cores con TMT

- Fine-grain multithreading (FGMT): la conmutación entre threads la decide el hardware cada ciclo, por round-robin o por eventos de cierta latencia combinado con alguna técnica de planificación.
- Coarse-grain multithreading (CGMT): la conmutación entre threads la decide el hardware tras intervalos de tiempo prefijados o por eventos de cierta latencia.

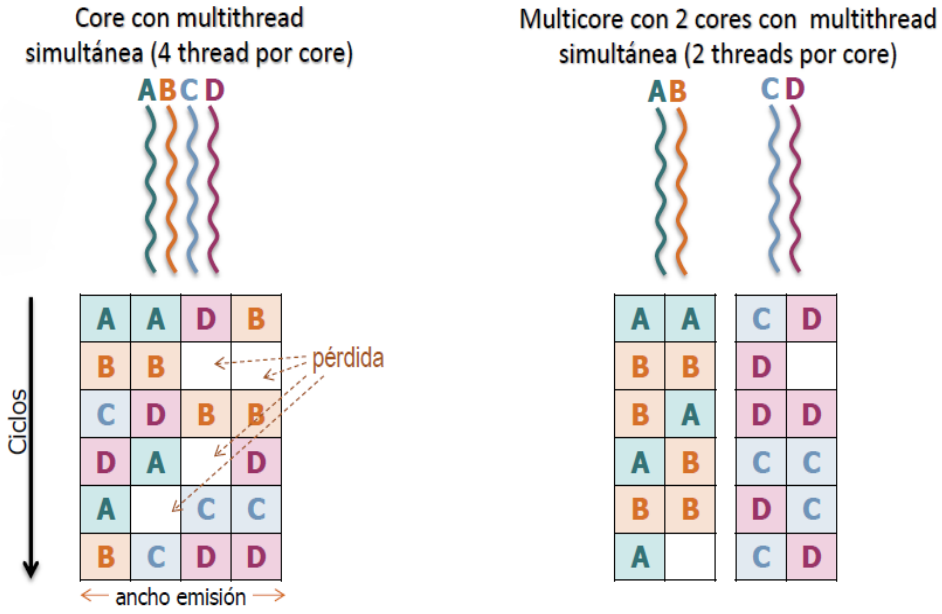
- Clasificación de cores con CGMT con conmutación por eventos
  - Estática:
    - Conmutación:
      - Explícita: instrucciones explícitas para conmutación.
      - Implícita: instrucciones de carga, almacenamiento, salto.
    - Ventaja/Inconveniente:
      - Coste cambio contexto bajo (0 o 1 ciclo) / cambios de contexto innecesarios
  - Dinámica:
    - Conmutación típicamente por: fallo en la última cache dentro del chip de procesamiento, interrupción, …
    - Ventaja/Inconveniente:
      - Reduce cambios de contexto innecesarios / mayor sobrecarga al cambiar de contexto
- Alternativas en un core escalar segmentado
  - En un core escalar se emite una instrucción cada ciclo de reloj



- Alternativas en un core con emisión múltiple de instrucciones de un thread
  - En un core superescalar o VLIW se emiten más de una instrucción cada ciclo de reloj.



- Core multithread simultánea y multicores
  - En un multicore y en un core superescalar con SMT se pueden emitir instrucciones de distintos threads cada ciclo de reloj



. Hardware y arquitecturas TLP en un chip

Hardware	CGMT	FGMT	SMT	CMP
Registros	**replicado** (al menos PC)	**replicado**	**replicado**	**replicado**
Almacenamiento	multiplexado	multiplexado, compartido, repartido o **replicado**	compartido, repartido o **replicado**	**replicado**
Otro hardware de las etapas del cauce	multiplexado	Captación: repartida o compartida; Resto: multiplexadas	UF: compartidas; Resto: repartidas o compartidas	**replicado**
Etiquetas para distinguir el thread de una instr.	Sí	Sí	Sí	**No**
Hardware para conmutar entre threads	Sí	Sí	**No**	**No**

- Lección 8

. Sistema de memoria en multiprocesadores

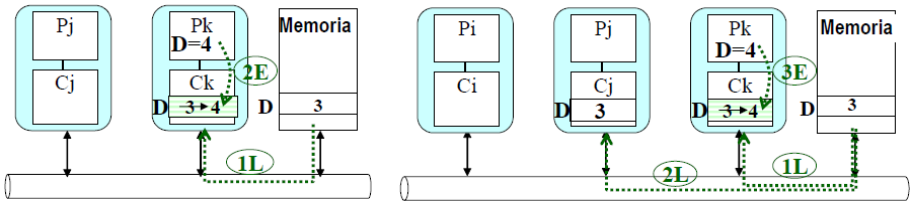
Está formado por:

- Caches de todos los nodos
- Memoria principal
- Controladores
- Buffers
- Medio de comunicación de todos estos componentes

La comunicación de datos entre procesadores la realiza el sistema de memoria. La lectura de una dirección debe devolver lo último que ha escrito.

. Incoherencia en el sistema de memoria

Clases de estructuras de datos	Eventos que ponen de manifiesto faltas de coherencia	Tipos de Falta de coherencia
Datos modificables	E/S	Cache-MP
Datos modificables compartidos	Fallo de cache	Cache-MP
Datos modificables privados	Emigra thread/proceso→Fallo cache	Cache-MP
Datos modificables compartidos	Lectura de cache no actualizada	Cache-Cache



. Métodos de actualización de memoria principal implementados en caches

1- Escritura inmediata: cada vez que un procesador escribe en su cache escrible también en memoria principal.
2- Posescritura: se actualiza memoria principal escribiendo todo el bloque cuando se desaloja de la cache.

Por los principios de localidad temporal y espacial sería más rentable si se escribe todo el bloque una vez realizadas múltiples escrituras.

. Alternativas para propagar una escritura en protocolos de coherencia de cache

1- Escritura con actualización: cada vez que un procesador escribe en una dirección en su cache se escribe en las copias de esa dirección en otras caches.
2- Escritura con invalidación: antes que un procesador modifique una dirección en su cache se invalidan las copias del bloque de la dirección en otras caches

. Requisitos del sistema de memoria para evitar problemas por incoherencia

- Propagar las escrituras en una dirección. La escritura en una dirección debe hacerse visible en un tiempo finito a otros procesadores.
- Serializar las escrituras en una dirección. Las escrituras en una dirección deben verse en el mismo orden por todos los procesadores.
- Propagar escrituras en una dirección:
    - Usando difusión: los paquetes de actualización/invalidación se envían a todas las caches.
    - Para conseguir mayor escalabilidad: se debería enviar paquetes de actualización/invalidación sólo a caches con copia del bloque. Mantener en un directorio, para cada bloque, los nodos con copia del mismo.
- Serializar escrituras en una dirección. El orden en el que las peticiones de escritura llegan a su home o al directorio centralizado sirve para serializar en sistemas de comunicación que garantizan el orden de transferencia entre dos puntos.

. Alternativas para implementar el directorio

- Centralizado:
    - Compartido por todos los nodos
    - Contiene información de los bloques de todos los módulos de memoria.
- Distribuido:

- Las filas se distribuyen entre los nodos
- Típicamente el directorio de un nodo contiene información de los bloques de sus módulos de memoria

. Clasificación de protocolos para mantener coherencia en el sistema de memoria

- Protocolos de espionaje (snoopy): para buses y en general sistemas con una difusión eficiente bien porque el número de nodos es pequeño o porque la red implementa difusión.
- Protocolos basados en directorios: para redes sin difusión o escalables.
- Esquemas jerárquicos: jerarquía de buses, jerarquía de redes escalables, redes escalables-buses.

. Facetas de diseño lógico de protocolos para coherencia

- Política de actualización de memoria principal: escritura inmediata, posescritura, mixta.
- Política de coherencia entre caches: escritura con invalidación, escritura con actualización, mixta.
- Describir comportamiento:
  - Definir posibles estados de los bloques en cache, y en memoria.
  - Definir transferencias a generar ante eventos.
  - Definir transiciones de estados para un bloque en cache y en memoria.

. Protocolo de espionaje de tres estados (MSI)

- Estados de un bloque en cache:
  - Modificado (M): es la única copia del bloque válida en todo el sistema.
  - Compartido (C, S): está válido, también valido en memoria y puede que haya copia válida en otras caches.
  - Invalido (I): se ha invalidado o no está físicamente.
- Estados de un bloque en memoria:
  - Válido: puede haber copia válida en una o varias caches.
  - Inválido: habrá copia valida en una cache
- Transferencias generadas por un nodo con cache
  - Petición de lectura de un bloque (PtLec): por lectura con fallo de cache del procesador del nodo
  - Petición de acceso exclusivo (PtLecEx): por escritura del procesador en bloque compartido o inválido.
  - Petición de posescritura (PtPEsc): por el reemplazo del bloque modificado (el procesador del nodo no espera respuesta)
  - Respuesta con bloque (RpBloque): al tener un estado modificado el bloque solicitado por una PtLec o PtLecEx recibida.

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

. Protocolo MESI de espionaje

- Estados de un bloque en cache:
  - Modificado (M): es la única copia del bloque válida en todo el sistema
  - Exclusivo (E): es la única copia del bloque válida en caches, la memoria también está actualizada
  - Compartido (C, Shared): es válido, también válido en memoria y en al menos otra cache
  - Invalido (I): se ha invalidado o no está físicamente
- Estados de un bloque en memoria:
  - Válido: puede haber copia válida en una o varias caches
  - Invalido: habrá copia valida en una cache

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

. Protocolo MSI basado en directorios con o sin difusión

- MSI con directorios sin difusión:
    - Estados de un bloque en cache:
        - Modificado
        - Compartido
        - Inválido
    - Estados de un bloque en MP:
        - Válido
        - Invalido
    - Transferencias (tipos de paquetes):
        - Nodos:
            - Solicitantes (S)
            - Origen (O)
            - Modificado (M)
            - Propietario (P)
            - Compartido (C)
        - Petición de nodo S a O: lectura de un bloque (PtLec), lectura con acceso exclusivo (PtLecEx), petición de acceso exclusivo sin lectura (PtEx), posescritura (PtPEsc)
        - Reenvío de petición de nodo O a nodos con copia (P, M, C) invalidación, lectura.
        - Respuesta de:
            - Nodo P a O: respuesta con bloque, respuesta con o sin bloque confirmando invalidación
            - Nodo O a S respuesta con bloque, respuesta con o sin bloque confirmando fin invalidación
- MSI con directorios con difusión
    - Estados de un bloque en cache
        - Modificado
        - Compartido
        - Inválido
    - Estados de un bloque en Memoria Principal
        - Válido
        - Inválido
    - Transferencias (tipos de paquetes):
        - Nodos:
            - Solicitantes (S)
            - Origen (O)
            - Modificado (M)
            - Propietario (P)
            - Compartido (C)
        - Difusión de petición del nodo S a:
            - O y P: lectura de un bloque, lectura con acceso exclusivo, petición de acceso exclusivo sin lectura
            - O: posescritura
        - Respuesta de:

- Nodo P a O: respuesta con bloque, respuesta con o sin bloque confirmando inavalidación
- Nodo O a S: respuesta con bloque, respuesta con o sin bloque confirmando fin invalidación

- Lección 9

. Concepto de consistencia de memoria

Un modelo de consistencia de memoria especifica el orden en el cual las operaciones de acceso a memoria deben parecer haberse realizado. La coherencia abarca operaciones realizadas por múltiples componentes en una misma dirección.

. Consistencia secuencial (SC)

SC es el modelo de consistencia que espera el programador de las herramientas de alto nivel.

SC requiere que:

- Todas las operaciones de un único procesador parezcan ejecutarse en el orden descrito por el programa de entrada al procesador.
- Todas las operaciones de memoria parezcan ser ejecutadas una cada vez.

SC presenta el sistema de memoria a los programadores como una memoria global conectada a todos los procesadores a través de un conmutador central.

. Modelos de consistencia relajados

Los modelos de memoria relajados difieren en cuanto a los requisitos para garantizar SC que relajan:

- Orden del programa: Hay modelos que permiten que se relaje en el código ejecutado en un procesador el orden entre dos accesos a distintas direcciones (W → R, W→ W, R → RW).
- Atomicidad: Hay modelos que permiten que un procesador pueda ver el valor escrito por otro antes de que este valor sea visible al resto de los procesadores del sistema

Los modelos relajados comprenden:

- Los órdenes de acceso a memoria que no garantiza el sistema de memoria.
- Mecanismos que ofrece el hardware para garantizar un orden cuando sea necesario

. Modelo que relaja W → R

Permiten que una lectura pueda adelantar a una escritura precia en el orden del programa; pero evita dependencias RAW. Lo implementan los sistemas con buffer de escritura para los procesadores. Generalmente permiten que el procesador pueda leer una dirección directamente del buffer. Para garantizar un orden correcto se pueden utilizar instrucciones de serialización.

Hay sistemas en los que se permite que un procesador pueda leer la escritura de otro antes que el resto de procesadores (acceso no atómico). Para garantizar acceso atómico se puede utilizar instrucciones de lectura-modificación-escritura atómicas.

. Modelo que relaja W → y W → W

Tiene buffer de escritura que permite que lecturas adelanten a escrituras en el buffer.

Permiten que el hardware solape escrituras a memoria a distintas direcciones, de forma que pueden llegar a la memoria principal o a caches de todos procesadores fuera del orden del programa.

En sistemas con este modelo se proporciona hardware para garantizar los dos órdenes. Este modelo no se comporta como SC.

. Modelo de ordenación débil

Relaja W → R, W → W y R → RW.

Si S es una operación de sincronización, ofrece hardware para garantizar el orden.

- S → WR
- WR → S

. Consistencia de liberación

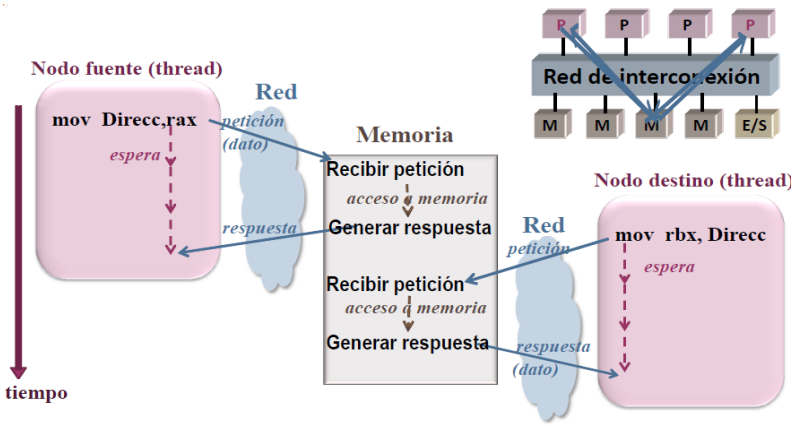
Relaja W → R, W → W y R → RW.

Si SA es una operación de adquisición y SL de liberación, ofrece hardware para garantizar el orden:

- SA → WR
- WR → SL

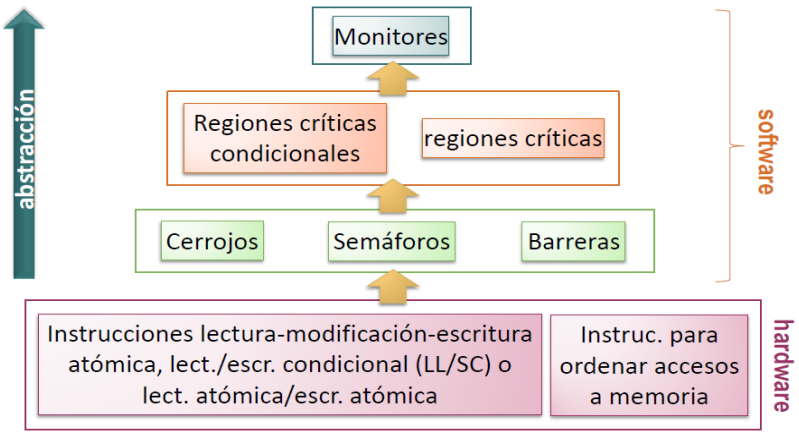
- Lección 10

. Comunicación en multiprocesadores y necesidad de usar código de sincronización



- Comunicación uno-a-uno:
  - Se debe garantizar que el proceso que recibe lea la variable compartida cuando el proceso que envía haya escrito en la variable el dato a enviar
  - Si se reutiliza la variable para comunicación, se debe garantizar que no se envía un nuevo dato en la variable hasta que no se haya leído el anterior
- Comunicación colectiva:
  - La lectura-modificación-escritura se debería hacer en exclusión mutua (es una sección crítica)
    - Sección crítica: Secuencia de instrucciones con una o varias direcciones compartidas que se deben acceder en exclusión mutua

. Soporte software y hardware de sincronización



. Cerrojos

Permiten sincronización mediante dos operaciones:

- Cierre del cerrojo
  - Intenta adquirir el derecho a acceder a una sección crítica. Si varios procesadores intentan la adquisición a la vez sólo uno de ellos lo debe conseguir, el resto debe pasar a una etapa de espera.
  - Todos los procesos que ejecuten el cerrojo con este ya cerrado deben quedar esperando.
- Apertura del cerrojo
  - Libera a uno de los threads que esperan el acceso a una sección crítica.
  - Si no hay threads en espera, permitirá que el siguiente que ejecute el cierre adquiera el cerrojo sin espera.

. Barreras

La función barrera se utiliza para sincronizar entre sí, en algún punto del código, los procesos que colaboran en la ejecución de un trozo de código. Las barreras proporcionan un medio para asegurar que ningún proceso un punto del código hasta que todos los procesos hayan llegado a ese punto.