Clase 20 - Coherencia Caché

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl

Resumen de la clase pasada

Funciones de correspondencia - *Fully associative*

• En este esquema, cada bloque de memoria puede asociarse a **cualquier línea de caché**

• Aprovecha todo el espacio y maximiza *hit-rate*

• Como se puede ocupar cualquier línea, la composición del tag para identificar si un dato se encuentra en caché se **complejiza**

Funciones de correspondencia - *N way associative*

- En este esquema, cada bloque de memoria puede asociarse a **cualquier línea de caché**
- Aprovecha todo el espacio y maximiza *hit-rate*
- Como se puede ocupar cualquier línea, la composición del tag para identificar si un dato se encuentra en caché se **complejiza**

Políticas de reemplazos

- **Bélády**: El bloque que se utilizará más lejos en el futuro se saca. Óptimo no alcanzable en la práctica.
- **First-in First-out (FIFO)**: El primer bloque en entrar es el primero en salir. Simple, pero tonto.
- **Least Frequently Used (LFU)**: El bloque con menos accesos se saca. Mejor que FIFO, solo un poco más complejo.
- **Least Recently Used (LRU)**: El bloque con mayor tiempo sin accesos se saca. Complejo, requiere timestamp. En general el de mejor rendimiento.
- **Random**: Muy rápido y con rendimiento algo inferior a LFU y LRU.

Políticas de escritura

- **Write-through:** el bloque modificado es escrito inmediatamente en la memoria principal
- **Write-back:** el bloque modificado es escrito en la memoria principal sólo cuando va a ser sustituido

¿Dudas?

Arquitectura de Computadores: Mejoras y extensiones

En lo que hemos visto hasta ahora

- Una máquina programable que ejecuta programas
- Interacción con dispositivos de entrada y salida

Lo que nos queda ver **mejoras:**

- Mejora los accesos a memoria 
- Mejora en lo que respecta a la acceso a memoria en más de un procesador
- Mejora en lo que respecta a la ejecución de instrucciones en la CPU.

Arquitectura de Computadores: Mejoras y extensiones

En lo que hemos visto hasta ahora

- Una máquina programable que ejecuta programas
- Interacción con dispositivos de entrada y salida

Lo que nos queda ver **mejoras:**

- Mejora los accesos a memoria 
- **Mejora en lo que respecta a la acceso a memoria en más de un procesador**
- Mejora en lo que respecta a la ejecución de instrucciones en la CPU.

Sistemas multiprocesador con memoria compartida

- Los procesadores comparten en algún nivel su memoria
- Cada uno podría tener su propia jerarquía de memoria
- Si todos los procesadores se incorporan en un único chip, hablamos de un procesador **multicore**

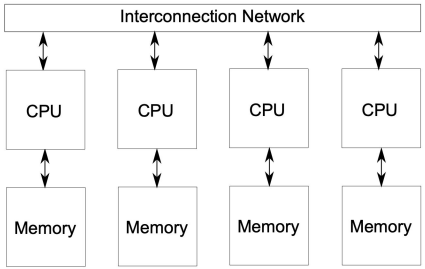


Figura 10: Diagrama multiprocesador de paso de mensajes.

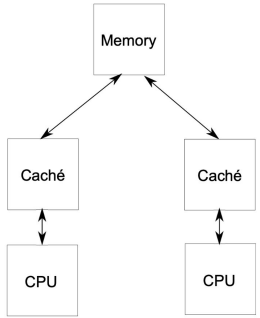
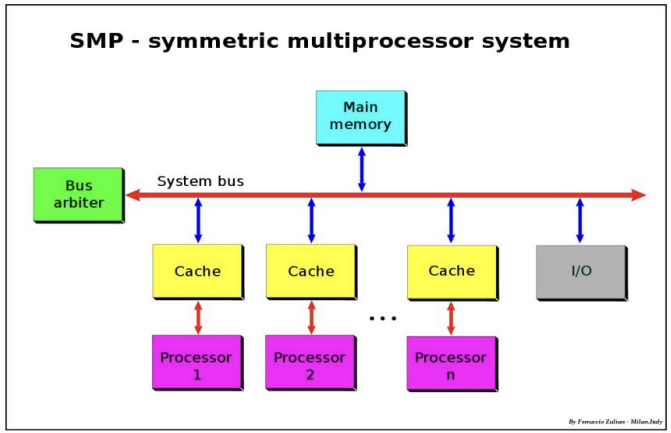


Figura 12: Diagrama de multiprocesador de memoria compartida, con cachés individuales.

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/12%20-%20Paralelismo%20avanzado.pdf

Esquemas de memoria compartida según tipo de acceso

• **Uniform Memory Access** (UMA): Todos los procesadores acceden de **la misma forma** a la memoria. Puede ser a través de un único bus o un **crossbar switch**



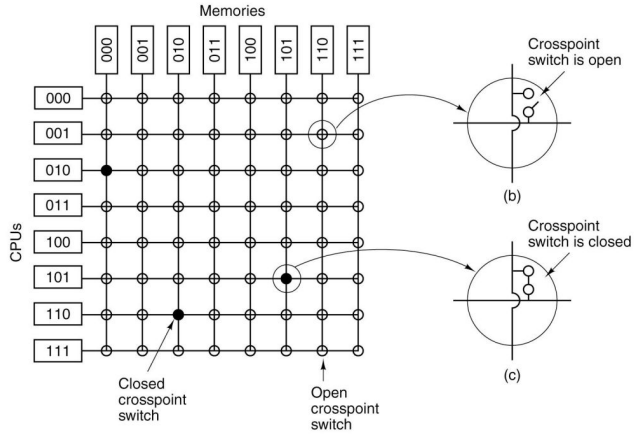


Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/12%20-%20Paralelismo%20avanzado.pdf

Esquemas de memoria compartida según tipo de acceso

• **Non-Uniform Memory Access (NUMA)**: Los procesadores se reparten en nodos donde cada uno puede acceder a un segmento específico de la memoria

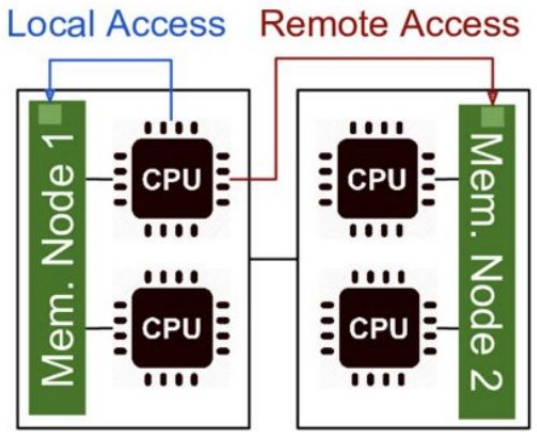


Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/12%20-%20Paralelismo%20avanzado.pdf

¿Dudas?

Tipos de arquitectura paralela - Memoria compartida

• Una política de escritura **write-through**, no habrá problemas ya que la memoria será consistente para todos los procesadores (más tiempo)

• Actualmente se opta por velocidad por tanto hay se sigue el protocolo de escritura **write-back**, pero de este nace un nuevo problema: la **coherencia de caché**

Coherencia de caché - Snooping

- La coherencia de caché es la consistencia en la memoria caché de dos o más procesadores

- Debemos asegurar que los otros procesadores que accedan al mismo dato lean la versión más actualizada.

- **Bus snooping:** Los controladores de caché deben revisar las solicitudes de lectura y escritura que se realizan a sus contrapartes a través de un bus compartido

Coherencia de caché - Snarfing

- **Snarfing:** cuando un controlador de caché detecta una solicitud de escritura a una dirección de memoria que contiene, actualiza inmediatamente

- Esto se utiliza menos por el aumento de latencia en los accesos a memoria y el aumento de tráfico en el bus compartido de datos.

¿Dudas?

Coherencia de caché - Protocolo MSI

- **Modified:** Indica que el contenido de la línea fue modificado y no es consistente con el bloque de la memoria principal. Este puede existir solo en una caché para un mismo bloque compartido.
- **Shared:** Indica que el contenido de la línea se encuentra en al menos una memoria caché, pero no se ha modificado.
- **Invalid:** Indica que el contenido de la línea es inválido, que puede ser por dato no existente o por solicitud desde el bus compartido.

Coherencia de caché - Protocolo MSI

Para entender el protocolo, se definen los siguientes eventos:

- Eventos del procesador actual
  - **PrRd:** Solicitud de lectura (Read - Rd).
  - **PrWr:** Solicitud de escritura (Write - Wr).
- Eventos del bus compartido -
  - **BusRd:** Solicitud de un bloque de datos **sin** intención de modificarlo.
  - **BusRdX:** Solicitud de un bloque de datos **con** intención de modificarlo.
  - **Flush:** Transferencia de línea de caché a la memoria principal.

Protocolo MSI - Cambios por eventos del procesador

- Si se hace la lectura de un dato que no está en memoria o que se invalidó por una modificación (estado **Invalid**)
- Entonces se lee el bloque de memoria y se copia; el estado de la línea de caché pasa a estado **Shared** y se emite la señal **BusRd** al bus compartido.
- Si se hace una lectura en estado **Shared** o lectura/escritura en estado **Modified**, no se emite ninguna señal en el bus compartido

Protocolo MSI - Cambios por eventos del procesador

- Si se hace una escritura en estado Invalid, se lee el bloque de memoria y se copia
- Entonces se modifica el contenido directamente en caché; el estado de la línea pasa a estado **Modified** y se emite la señal **BusRdX** al bus compartido.
- Si se hace una escritura en estado **Shared**, se realizan las mismas acciones pero se evita la copia

¿Dudas?

Protocolo MSI - Cambios por eventos del BUS

- Si un controlador recibe la señal **BusRd** o **BusRdX** para una de sus líneas en estado **Invalid**, mantiene su estado sin hacer nada

- Lo mismo ocurre para la señal **BusRd** y una línea en estado **Shared**

- Si un controlador recibe la señal **BusRdX** para una de sus líneas en estado **Shared**, se actualiza a estado **Invalid**

Protocolo MSI - Cambios por eventos del BUS

- Si un controlador recibe la señal **BusRd** para una de sus líneas en estado Modified, actualiza su estado a **Shared** y realiza **Flush**

- Si un controlador recibe la señal **BusRdX** para una de sus líneas en estado **Modified**, también realiza **Flush** pero pasa a estado **Invalid**

Protocolo MSI - Diagramas

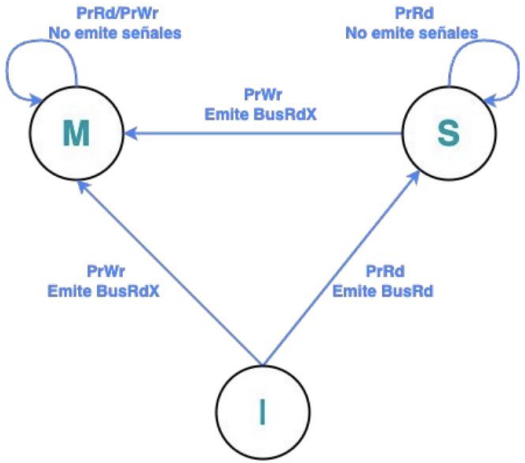


Diagrama de estado de una línea de caché para el protocolo MSI y eventos del procesador.

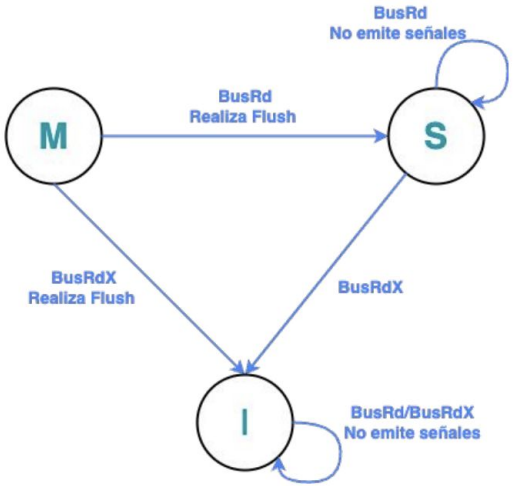


Diagrama de estado de una línea de caché para el protocolo MSI y eventos del bus compartido.

Imagen Referencia: https://github.com/IIC2343/Syllabus-2025-2/blob/main/Clases/S3/Clase%2011%20-%20Coherencia%20de%20Cache%CC%81.pdf

Protocolo MSI - Diagrama General

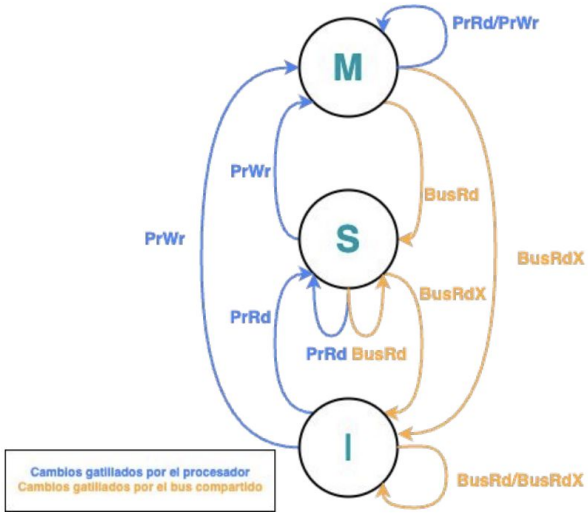


Diagrama de estado de una línea de caché para el protocolo MSI.

Imagen Referencia: https://github.com/IIC2343/Syllabus-2025-2/blob/main/Clases/S3/Clase%2011%20-%20Coherencia%20de%20Cache%CC%81.pdf

¿Dudas?

Clase 20 - Coherencia Caché

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl