



Organización del Computador II

Microarquitectura - El Sistema de Memoria

Alejandro Furfaro

Departamento de Computación - FCEyN - UBA

13 de abril de 2018

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

3

Memoria Cache

- Principio de Funcionamiento

- Clasificación de memorias
- Memorias y velocidad del Procesador

- Hardware dedicado = + complejidad
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

Evolución



Pioneros:

Maurice Wilkes en 1947 con la primer memoria de tanque de mercurio para la computadora EDSAC. Capacidad 2 bytes.

Visionarios:

"640K debe ser suficiente memoria para cualquiera. . ."

Bill Gates. 1981

Evolución



Pioneros:

Maurice Wilkes en 1947 con la primer memoria de tanque de mercurio para la computadora EDSAC. Capacidad 2 bytes.

Visionarios:

"640K debe ser suficiente memoria para cualquiera. . ."

Bill Gates. 1981

Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

- Clasificación de memorias
- Memorias y velocidad del Procesador

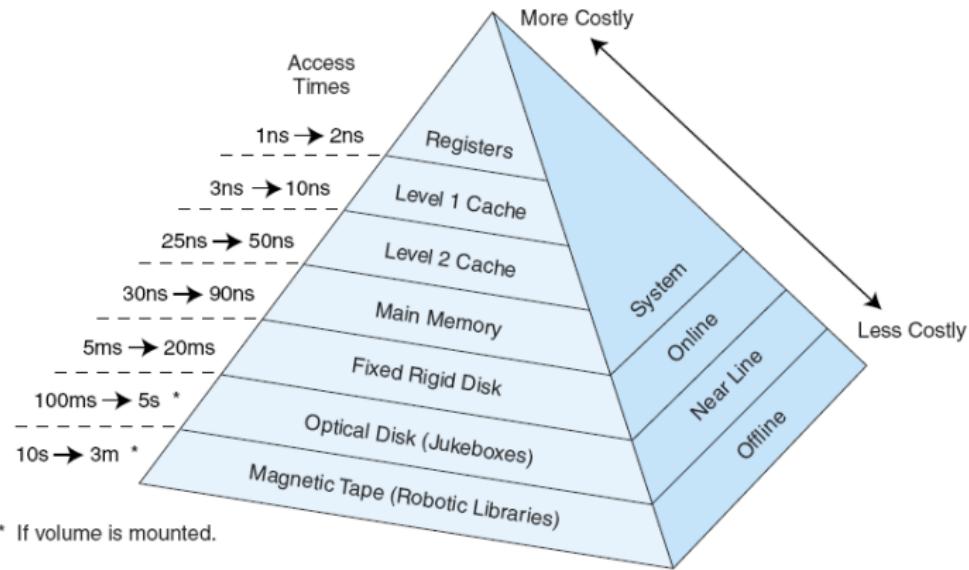
3

Memoria Cache

- Principio de Funcionamiento

- Hardware dedicado = + complejidad
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

Baja Latencia o alta capacidad



Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- **Principio de Vecindad o Localidad**

2

Tecnologías de Memoria

- Clasificación de memorias
- Memorias y velocidad del Procesador

3

Memoria Cache

- Principio de Funcionamiento

- Hardware dedicado = + complejidad
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

Principio de funcionamiento... ¿black magic?

- Para nada.
- El comportamiento de los algoritmos de software que se emplean habitualmente, es el que rige como y para que se diseña el hardware.
- El controlador cache trabaja mediante dos principios:

Principio de funcionamiento... ¿black magic?

- Para nada.
- El comportamiento de los algoritmos de software que se emplean habitualmente, es el que rige como y para que se diseña el hardware.
- El controlador cache trabaja mediante dos principios:

Principio de funcionamiento... ¿black magic?

- Para nada.
- El comportamiento de los algoritmos de software que se emplean habitualmente, es el que rige como y para que se diseña el hardware.
- El controlador cache trabaja mediante dos principios:

Principio de funcionamiento... ¿black magic?

- Para nada.
- El comportamiento de los algoritmos de software que se emplean habitualmente, es el que rige como y para que se diseña el hardware.
- El controlador cache trabaja mediante dos principios:

Principio de funcionamiento... ¿black magic?

- Para nada.
- El comportamiento de los algoritmos de software que se emplean habitualmente, es el que rige como y para que se diseña el hardware.
- El controlador cache trabaja mediante dos principios:

Principio de vecindad temporal

Una dirección de memoria que está siendo accedida actualmente, tiene muy alta probabilidad de seguir siendo accedida en el futuro inmediato.

Principio de funcionamiento... ¿black magic?

- Para nada.
- El comportamiento de los algoritmos de software que se emplean habitualmente, es el que rige como y para que se diseña el hardware.
- El controlador cache trabaja mediante dos principios:

Principio de vecindad temporal

Una dirección de memoria que está siendo accedida actualmente, tiene muy alta probabilidad de seguir siendo accedida en el futuro inmediato.

Principio de vecindad espacial

Si se está accediendo a una dirección determinada de memoria actualmente, la probabilidad de que esta dirección y sus direcciones vecinas sean accedidas en el futuro inmediato es muy alta.

Ejemplo: Algoritmo de Convolución

```
1   for ( i = 0 ; i < 256 ; i++ )  
2   {  
3       suma = 0.0f;  
4       for ( j = 0 ; ( j <= i && j < 256) ; j++)  
5           suma += v0[ i-j ] * v1[ j ];  
6       fAux[ i ] = suma;  
7   }
```

- Las variables *i*, *j*, **suma**, se utilizan a menudo.
- Por lo tanto si se mantienen en el cache, el tiempo de acceso a estas variables por parte del procesador es óptimo.

Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

● Clasificación de memorias

- Memorias y velocidad del Procesador

3

Memoria Cache

- Principio de Funcionamiento

- Hardware dedicado = + complejidad
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

Memorias No volátiles

- Capaces de retener la información almacenada cuando se les desconecta la alimentación.
- Han evolucionado tecnológicamente pasando por múltiples modelos que paulatinamente permitieron su modificación offline, y actualmente, son modificables en tiempo real.
 - Partieron desde las viejas memorias denominadas ROM (por Read Only Memory), que en sus primeras implementaciones debían ser grabadas por el fabricante del chip, y no eran modificables.
 - Evolucionaron a los chips EEPROM (Electrically Erasable Programmable Read-Only Memory), que permitían la modificación en tiempo real, pero solo una vez por cada bit.
 - Finalmente llegaron las memorias Flash, que permiten la modificación en tiempo real y muchas veces por cada bit.

Memorias No volátiles

- Capaces de retener la información almacenada cuando se les desconecta la alimentación.
- Han evolucionado tecnológicamente pasando por múltiples modelos que paulatinamente permitieron su modificación offline, y actualmente, son modificables en tiempo real.
 - Partieron desde las viejas memorias denominadas **ROM** (por Read Only Memory), que en sus primeras implementaciones debían ser grabadas por el fabricante del chip, y no eran modificables.
 - Pasaron por componentes programables solo una vez, factibles de ser borrados con luz ultravioleta de una determinada longitud de onda, etc.
 - Hasta llegar las actuales memorias flash que pueden ser grabadas por algoritmos de escritura on the fly por el usuario, y cuyo ejemplo más habitual son los discos de estado sólido de los equipos portátiles modernos.

Memorias No volátiles

- Capaces de retener la información almacenada cuando se les desconecta la alimentación.
- Han evolucionado tecnológicamente pasando por múltiples modelos que paulatinamente permitieron su modificación offline, y actualmente, son modificables en tiempo real.
 - Partieron desde las viejas memorias denominadas **ROM** (por Read Only Memory), que en sus primeras implementaciones debían ser grabadas por el fabricante del chip, y no eran modificables.
 - Pasaron por componentes programables solo una vez, factibles de ser borrados con luz ultravioleta de una determinada longitud de onda, etc.
 - Hasta llegar las actuales memorias flash que pueden ser grabadas por algoritmos de escritura on the fly por el usuario, y cuyo ejemplo más habitual son los discos de estado sólido de los equipos portátiles modernos.

Memorias No volátiles

- Capaces de retener la información almacenada cuando se les desconecta la alimentación.
- Han evolucionado tecnológicamente pasando por múltiples modelos que paulatinamente permitieron su modificación offline, y actualmente, son modificables en tiempo real.
 - Partieron desde las viejas memorias denominadas **ROM** (por Read Only Memory), que en sus primeras implementaciones debían ser grabadas por el fabricante del chip, y no eran modificables.
 - Pasaron por componentes programables solo una vez, factibles de ser borrados con luz ultravioleta de una determinada longitud de onda, etc.
 - Hasta llegar las actuales memorias flash que pueden ser grabadas por algoritmos de escritura on the fly por el usuario, y cuyo ejemplo más habitual son los discos de estado sólido de los equipos portátiles modernos.

Memorias No volátiles

- Capaces de retener la información almacenada cuando se les desconecta la alimentación.
- Han evolucionado tecnológicamente pasando por múltiples modelos que paulatinamente permitieron su modificación offline, y actualmente, son modificables en tiempo real.
 - Partieron desde las viejas memorias denominadas **ROM** (por Read Only Memory), que en sus primeras implementaciones debían ser grabadas por el fabricante del chip, y no eran modificables.
 - Pasaron por componentes programables solo una vez, factibles de ser borrados con luz ultravioleta de una determinada longitud de onda, etc.
 - Hasta llegar las actuales memorias flash que pueden ser grabadas por algoritmos de escritura on the fly por el usuario, y cuyo ejemplo mas habitual son los discos de estado sólido de los equipos portátiles modernos.

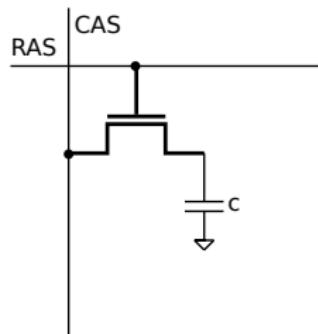
Volátiles

- Conocidas como RAM (Random Access Memory), se caracterizan por que una vez interrumpida la alimentación eléctrica, la información que almacenaban se pierde.
- Sin embargo estas memorias pueden almacenar mayores cantidades de información y modificarla en tiempo real a gran velocidad en comparación con las No Volátiles.
- Se clasifican de acuerdo con la tecnología y su diseño interno en dinámicas (DRAM) y estáticas (SRAM).

Uso en un computador

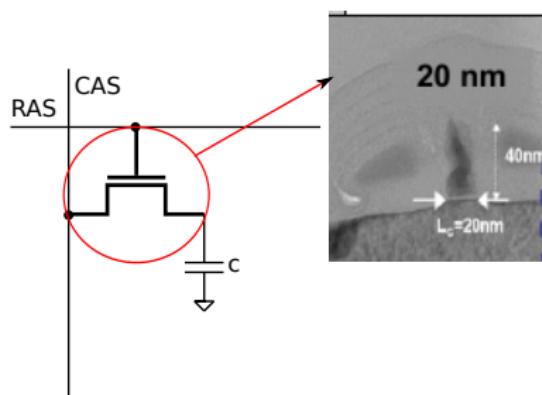
- La memoria no volátil se usa fundamentalmente para almacenar el programa de arranque de cualquier sistema.
- Se conectan en un espacio de direcciones determinado por el propio microprocesador de acuerdo a la dirección en la que éste irá a buscar la primer instrucción luego de encender el equipo.
- El resto es RAM y allí el sistema copia incluso buena parte del código de arranque para que se ejecute mas rápido (recordemos que las memorias volátiles tienen menor tiempo de acceso (menos ciclos de clock para acceder al contenido de una dirección)).
- Nos concentraremos a continuación en las memorias RAM, que es donde en general residen nuestros programas y el propio sistema operativo en un computador.

Memorias dinámicas



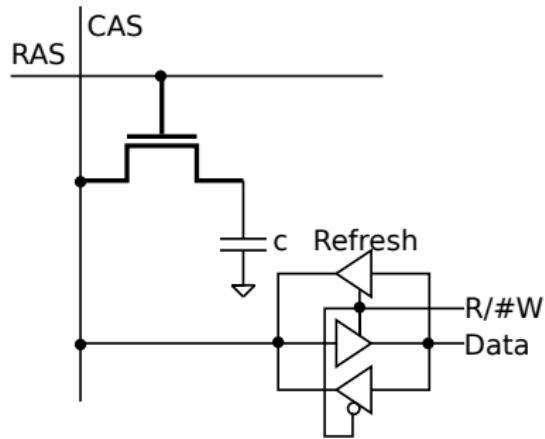
- Almacena la información en forma de estado de carga en un capacitor y la sostiene durante un breve lapso con la ayuda de un transistor.
- Una celda (un bit) se implementa con un solo transistor => máxima capacidad de almacenamiento por CI.
- Ese transistor está generalmente en estado de Corte. Consume mínima energía.

Memorias dinámicas



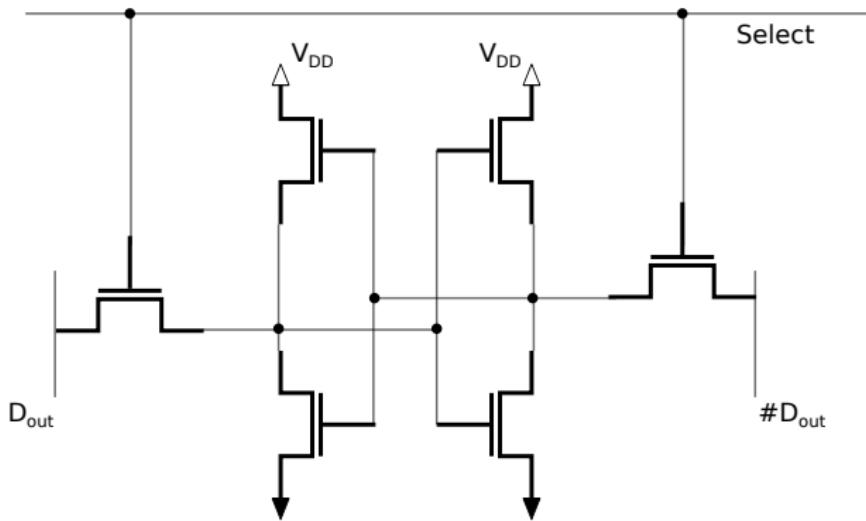
- Almacena la información en forma de estado de carga en un capacitor y la sostiene durante un breve lapso con la ayuda de un transistor.
- Una celda (un bit) se implementa con un solo transistor => máxima capacidad de almacenamiento por CI.
- Ese transistor está generalmente en estado de Corte. Consume mínima energía.

Memorias dinámicas



- Al leer el bit, se descarga la capacidad (lectura destructiva).
- Necesita regenerar la carga** cada vez que se la lee.
- Esta operación se realiza por realimentación mediante buffers.
- Aumenta entonces el tiempo total que demanda el acceso de la celda, ya que no libera la operación hasta no haber repuesto el estado de carga del capacitor.

Memorias estáticas



Memorias estáticas

- Almacena la información en un biestable.
- Una celda (un bit) se compone de seis transistores. Por lo tanto tiene menor capacidad de almacenamiento por CI.
- Tres de los seis transistores están saturados (conducen la máxima corriente posible en forma permanente) y los otros tres al corte (conducen una corriente prácticamente insignificante, pero no nula). Esto genera mayor consumo de energía por celda.
- La lectura es directa y no destructiva por lo cual el tiempo de acceso es muy bajo en comparación con las memorias dinámicas. De hecho, luego de los registros del procesador, son el medio de almacenamiento de menor tiempo de acceso.

Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

- Clasificación de memorias
- **Memorias y velocidad del Procesador**

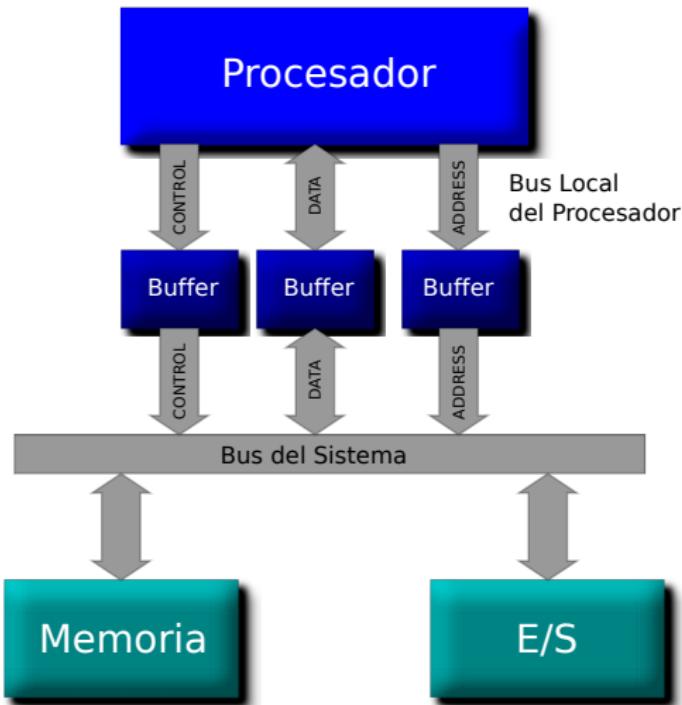
3

Memoria Cache

- Principio de Funcionamiento

- Hardware dedicado = + complejidad
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

Conexión básica (Según Von Newmann)



- Desde fines de los años 80, los procesadores desarrollaban velocidades muy superiores a los tiempos de acceso a memoria.
- En este escenario, el procesador necesita generar wait states para esperar que la memoria esté lista ("READY") para el acceso.
- ¿Tiene sentido lograr altos clocks en los procesadores si no puede aprovecharlos por tener que esperar (wait) a la memoria?

El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema.
Hay dos opciones...

- **RAM dinámica (DRAM)**

• Baja velocidad de acceso

• Alto consumo relativo

- **RAM estática (SRAM)**

• Alto consumo relativo

Conclusión:

Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos. Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema.
Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

Conclusión:

Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos. Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema.
Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

Conclusión:

Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos. Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema.
Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

Conclusión:

Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos. Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema.
Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

Conclusión:

Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos. Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema.
Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

Conclusión:

Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos. Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema.
Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

Conclusión:

Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos. Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

- Clasificación de memorias
- Memorias y velocidad del Procesador

3

Memoria Cache

- Principio de Funcionamiento

- Hardware dedicado = + complejidad
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

La solución... Memoria Cache

- Se trata de un banco de SRAM de muy alta velocidad, que contiene una copia de los datos e instrucciones que están en memoria principal.
- El arte consiste en que esta copia esté disponible justo cuando el procesador la necesita permitiéndole acceder a esos ítems sin recurrir a wait states.
- Combinada con una gran cantidad de memoria DRAM, para almacenar el resto de códigos y datos, resuelve el problema mediante una solución de compromiso típica.
- Requiere de hardware adicional que asegure que este pequeño banco de memoria cache contenga los datos e instrucciones mas frecuentemente utilizados por el procesador.

La solución... Memoria Cache

- Se trata de un banco de SRAM de muy alta velocidad, que contiene una copia de los datos e instrucciones que están en memoria principal.
- El arte consiste en que esta copia esté disponible justo cuando el procesador la necesita permitiéndole acceder a esos ítems sin recurrir a wait states.
- Combinada con una gran cantidad de memoria DRAM, para almacenar el resto de códigos y datos, resuelve el problema mediante una solución de compromiso típica.
- Requiere de hardware adicional que asegure que este pequeño banco de memoria cache contenga los datos e instrucciones mas frecuentemente utilizados por el procesador.

La solución... Memoria Cache

- Se trata de un banco de SRAM de muy alta velocidad, que contiene una copia de los datos e instrucciones que están en memoria principal.
- El arte consiste en que esta copia esté disponible justo cuando el procesador la necesita permitiéndole acceder a esos ítems sin recurrir a wait states.
- Combinada con una gran cantidad de memoria DRAM, para almacenar el resto de códigos y datos, resuelve el problema mediante una solución de compromiso típica.
- Requiere de hardware adicional que asegure que este pequeño banco de memoria cache contenga los datos e instrucciones mas frecuentemente utilizados por el procesador.

La solución... Memoria Cache

- Se trata de un banco de SRAM de muy alta velocidad, que contiene una copia de los datos e instrucciones que están en memoria principal.
- El arte consiste en que esta copia esté disponible justo cuando el procesador la necesita permitiéndole acceder a esos ítems sin recurrir a wait states.
- Combinada con una gran cantidad de memoria DRAM, para almacenar el resto de códigos y datos, resuelve el problema mediante una solución de compromiso típica.
- Requiere de hardware adicional que asegure que este pequeño banco de memoria cache contenga los datos e instrucciones mas frecuentemente utilizados por el procesador.

La solución... Memoria Cache

- Se trata de un banco de SRAM de muy alta velocidad, que contiene una copia de los datos e instrucciones que están en memoria principal.
- El arte consiste en que esta copia esté disponible justo cuando el procesador la necesita permitiéndole acceder a esos ítems sin recurrir a wait states.
- Combinada con una gran cantidad de memoria DRAM, para almacenar el resto de códigos y datos, resuelve el problema mediante una solución de compromiso típica.
- Requiere de hardware adicional que asegure que este pequeño banco de memoria cache contenga los datos e instrucciones mas frecuentemente utilizados por el procesador.

Características y métricas

El tamaño del banco de memoria cache debe ser:

- ① Suficientemente grande para que el procesador resuelva la mayor cantidad posible de búsquedas de código y datos en esta memoria asegurando una alta performance.
- ② Suficientemente pequeña para no afectar el consumo ni el costo del sistema.

Hit cuando se accede a un ítem (dato o código) y éste *se encuentra* en la memoria cache

Miss cuando se accede a un ítem (dato o código) y éste *no se encuentra* en la memoria cache

$$\text{hit rate} \text{ hitrate} = \frac{\text{Cantidad de Accesos con hit}}{\text{Cantidad de Accesos Totales}}$$

Se espera un hit rate lo mas alto posible

Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

3

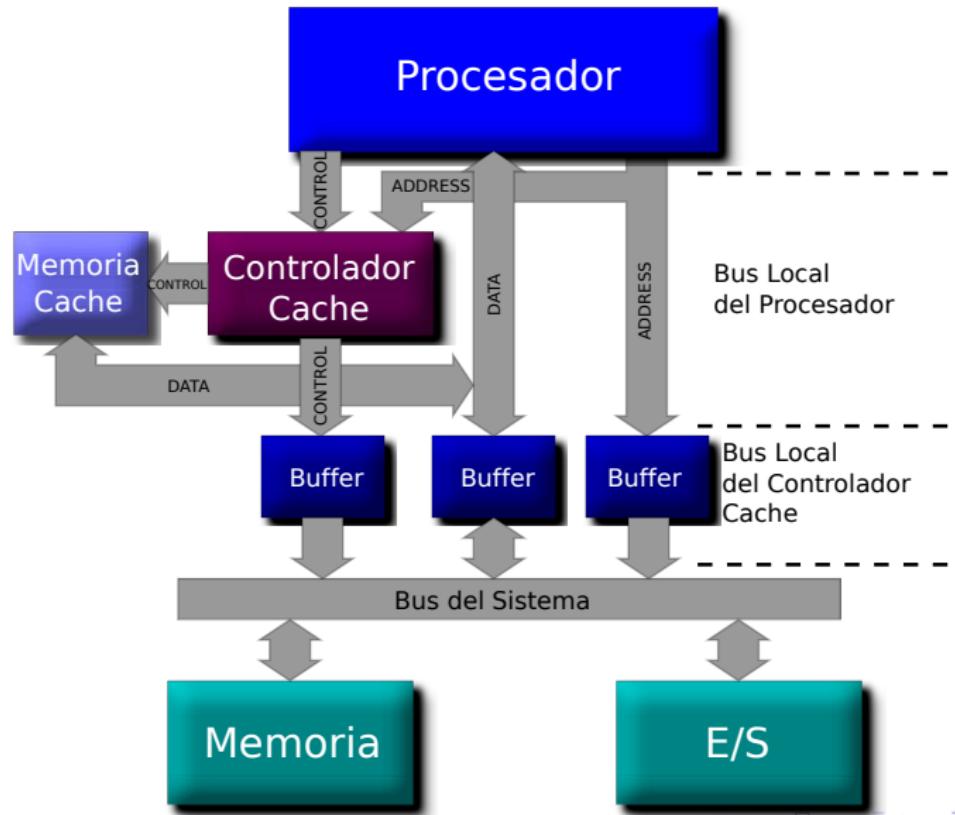
Memoria Cache

- Principio de Funcionamiento

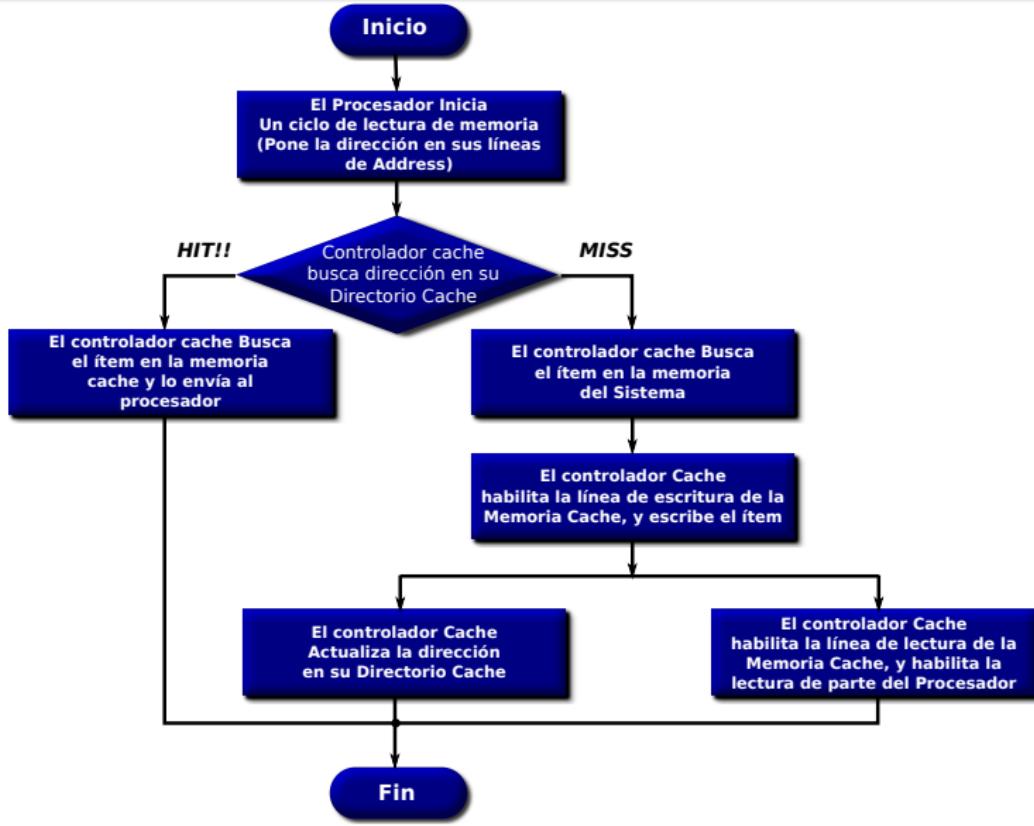
- Clasificación de memorias
- Memorias y velocidad del Procesador

- **Hardware dedicado = + complejidad**
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

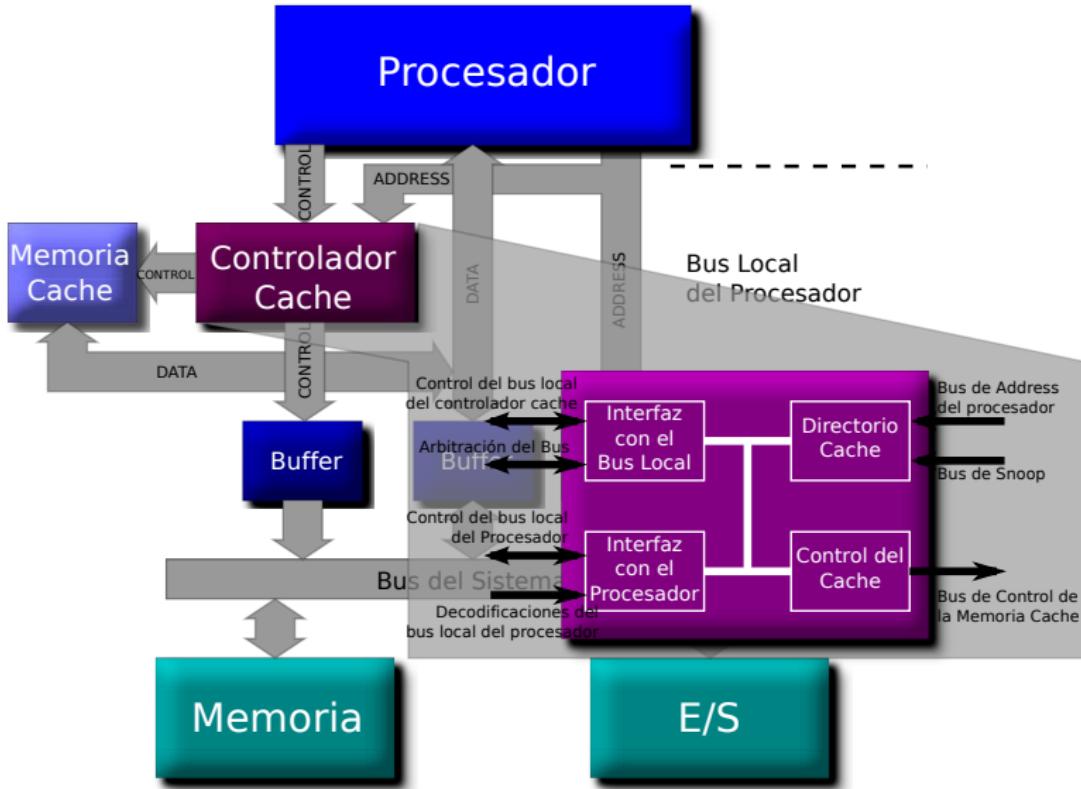
Subsistema Cache de Hardware



Operación de acceso a memoria para lectura



El Controlador Cache



Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

3

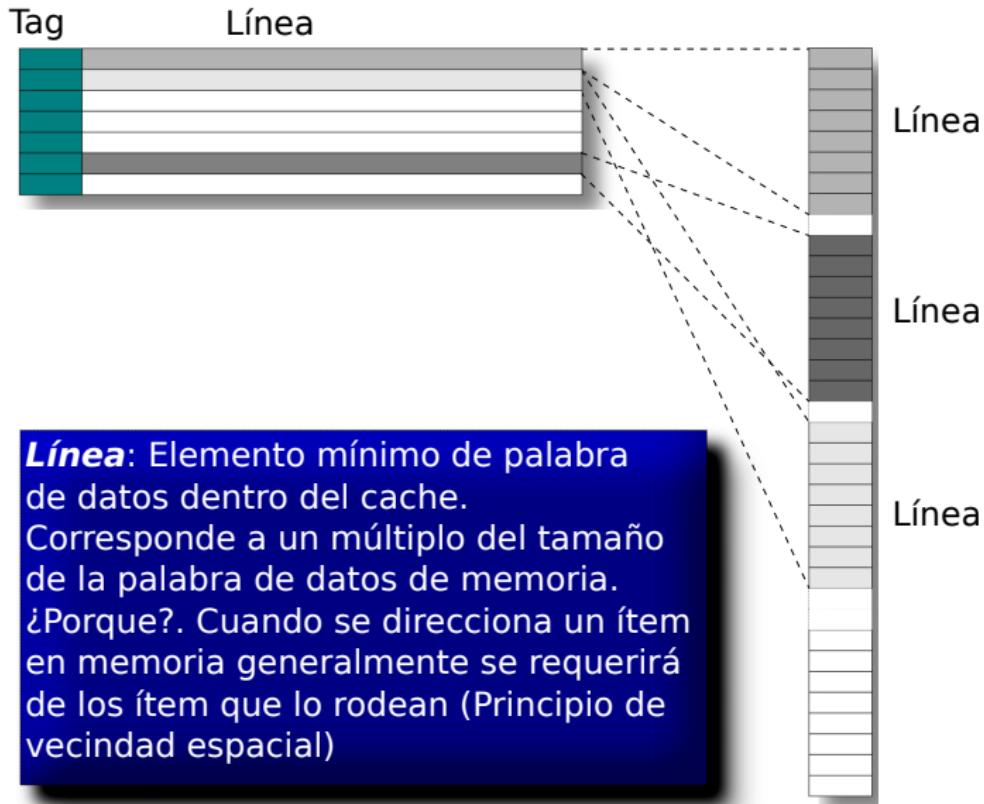
Memoria Cache

- Principio de Funcionamiento

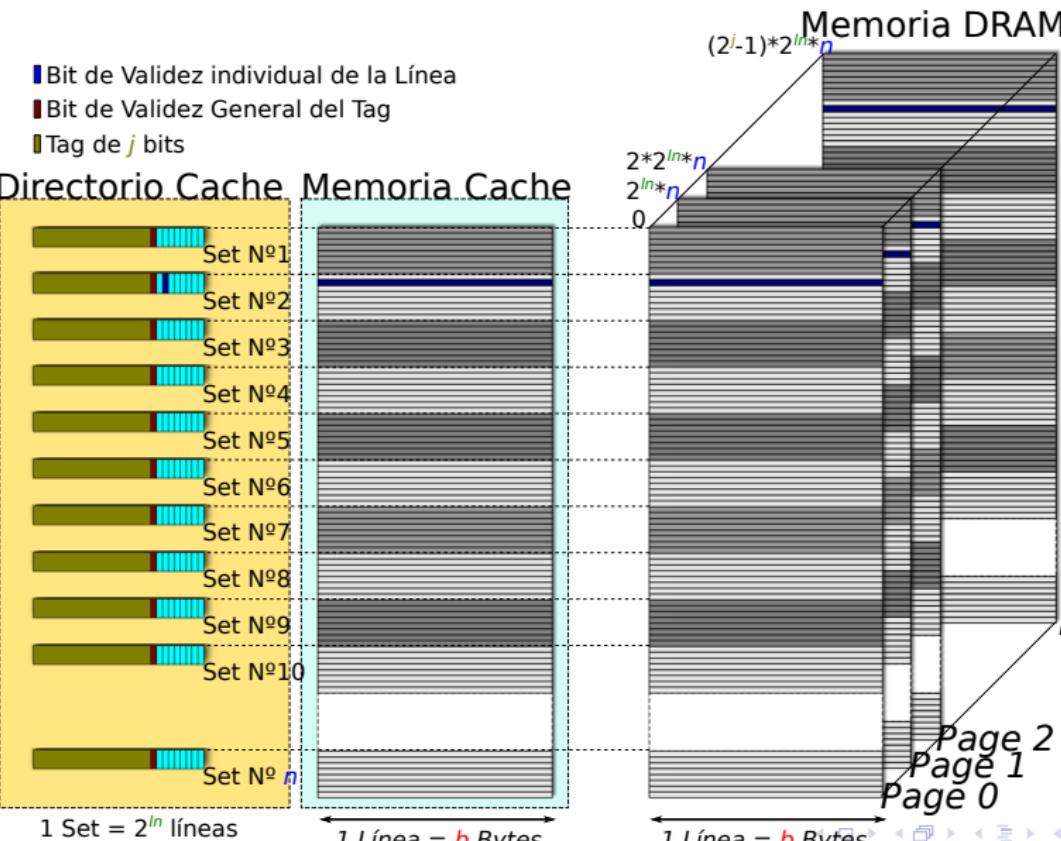
- Clasificación de memorias
- Memorias y velocidad del Procesador

- Hardware dedicado = + complejidad
- **organización de un cache**
- Coherencia de un cache
- Arquitecturas de cache avanzadas

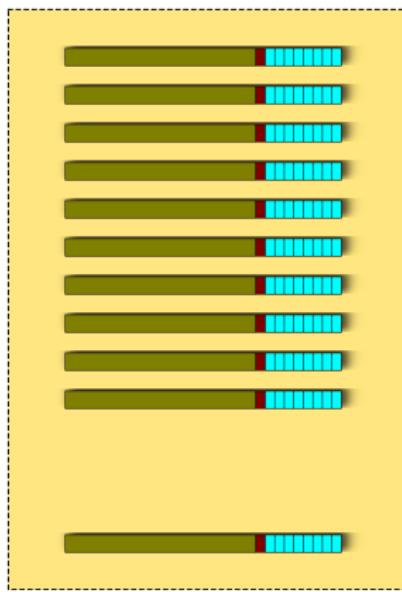
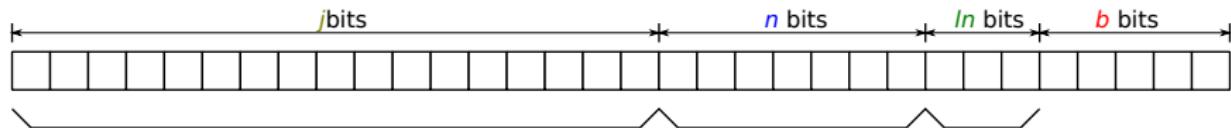
Organización del cache. Líneas



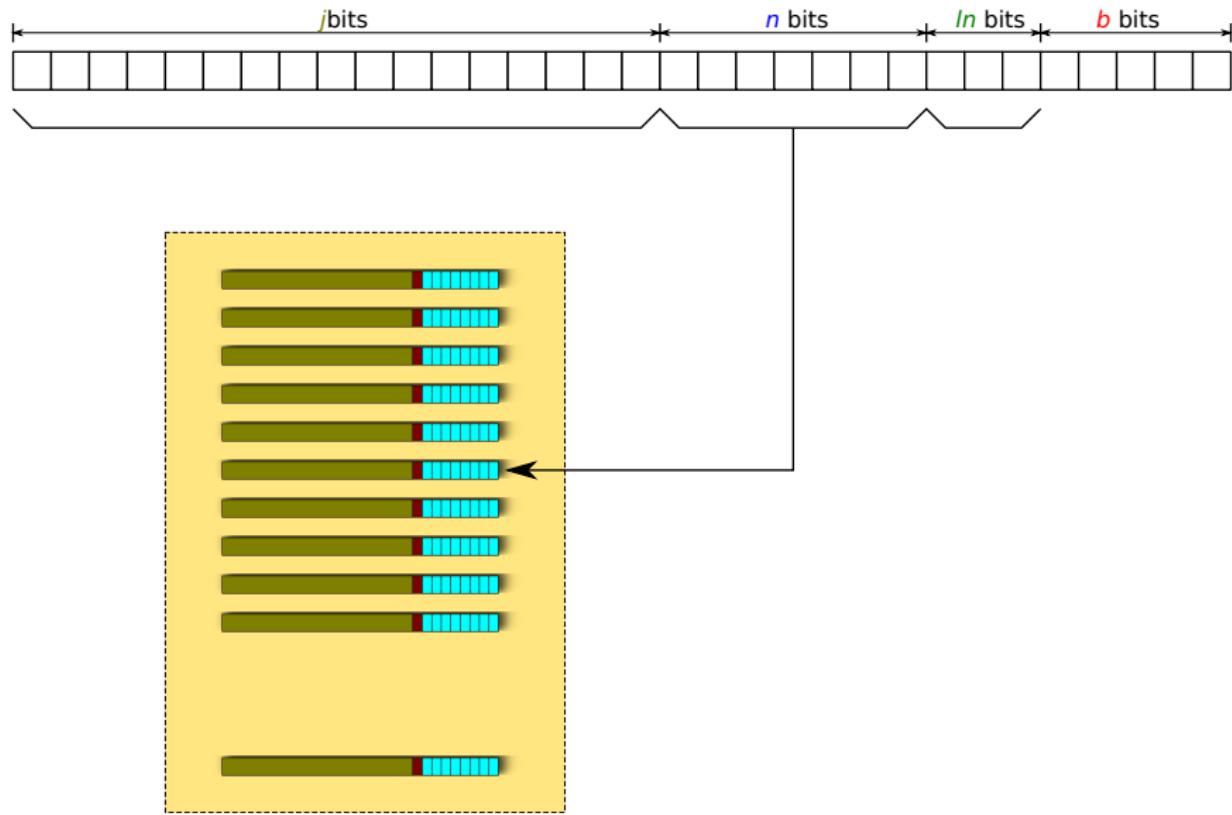
Sistema Cache de Mapeo Directo



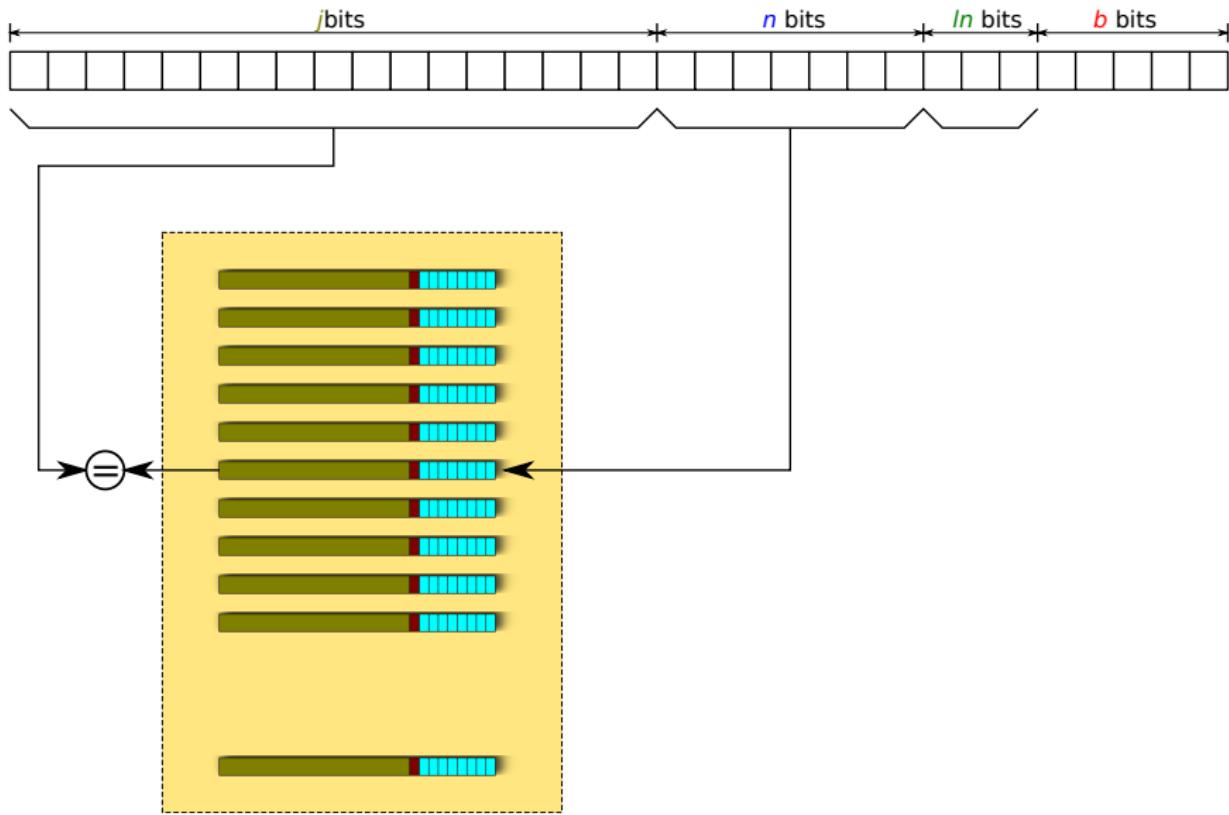
Sistema Cache de Mapeo Directo



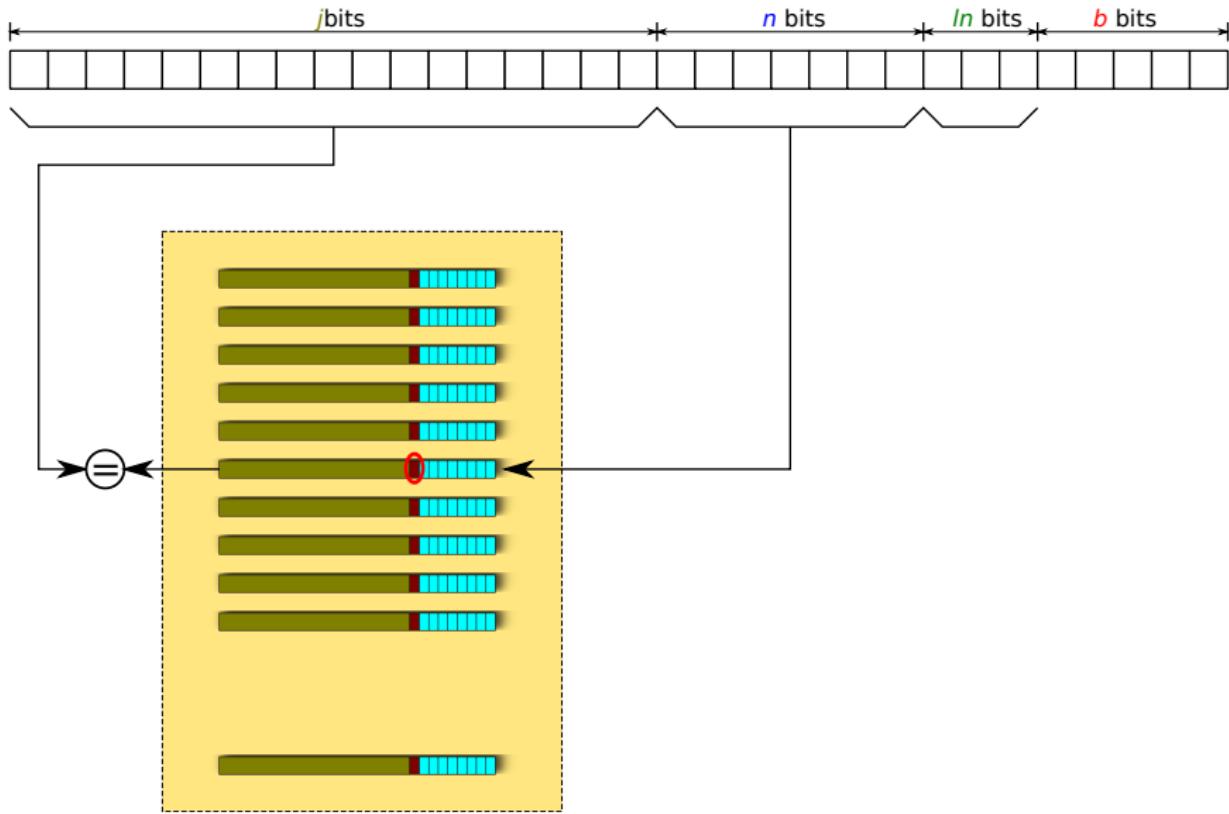
Sistema Cache de Mapeo Directo



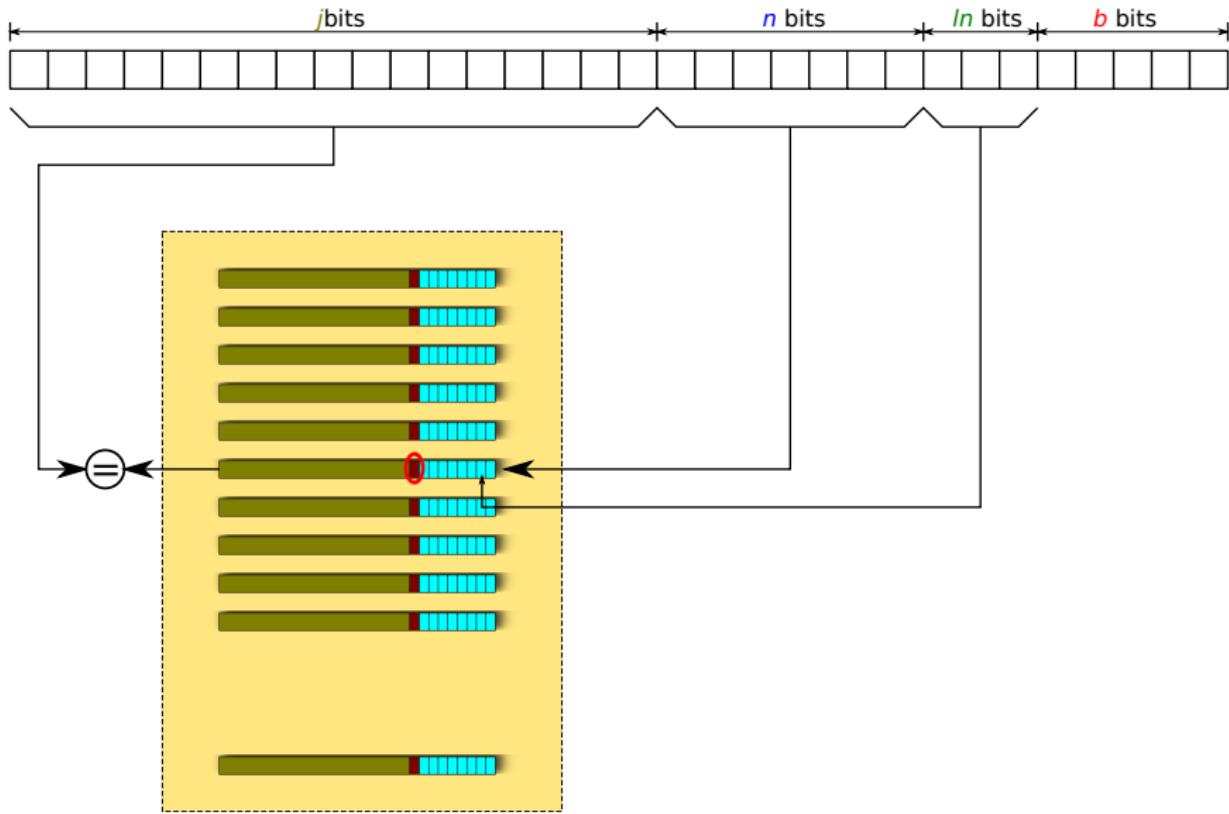
Sistema Cache de Mapeo Directo



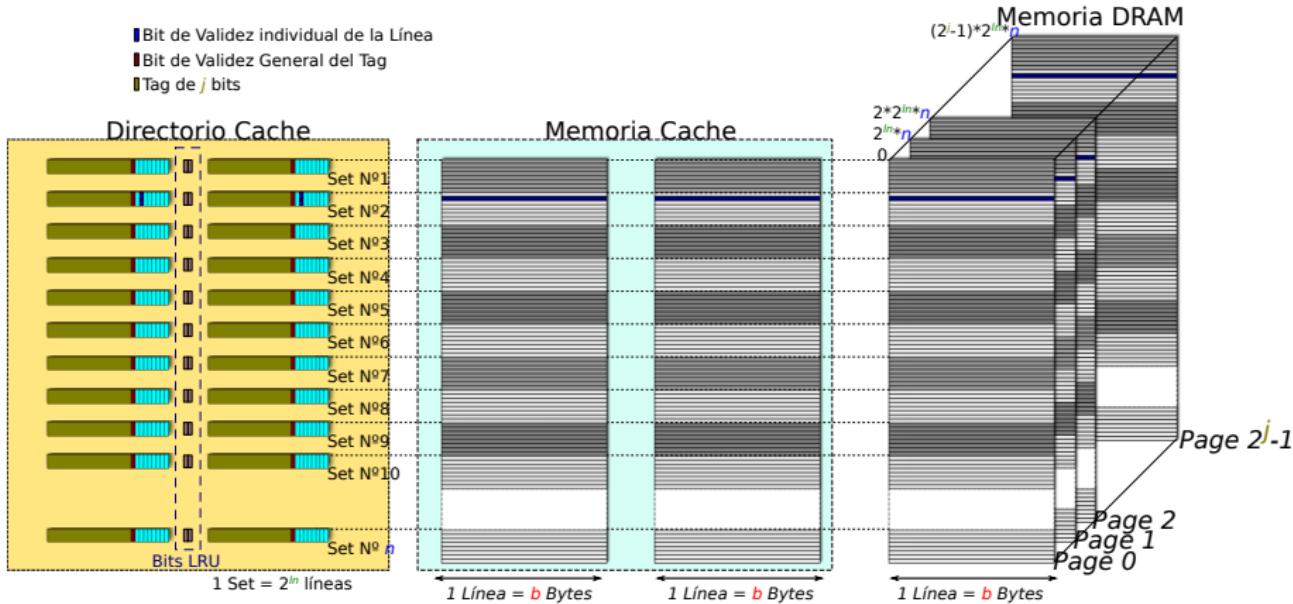
Sistema Cache de Mapeo Directo



Sistema Cache de Mapeo Directo



Sistema Cache Asociativo de 2 Vías



Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

3

Memoria Cache

- Principio de Funcionamiento

- Clasificación de memorias
- Memorias y velocidad del Procesador

- Hardware dedicado = + complejidad
- organización de un cache
- **Coherencia de un cache**
- Arquitecturas de cache avanzadas

¿Que ocurre durante las escrituras?

- Una variable que está en el caché también está alojada en alguna dirección de la DRAM.
- Idealmente ambos copias de la variable deben mantener el mismo valor (al menos siempre que sea posible).
- Cuando el procesador la modifica ésta condición deja de cumplirse. Hay varios modos de actuar, dependiendo de si el sistema dispone de una sola CPU o mas de una.
- Estas alternativas de acción en las escrituras se conocen como políticas de escritura y constituyen una de las decisiones más importantes en el diseño del sistema de memoria.

¿Que ocurre durante las escrituras?

- Una variable que está en el caché también está alojada en alguna dirección de la DRAM.
- Idealmente ambos copias de la variable deben mantener el mismo valor (al menos siempre que sea posible).
- Cuando el procesador la modifica ésta condición deja de cumplirse. Hay varios modos de actuar, dependiendo de si el sistema dispone de una sola CPU o mas de una.
- Estas alternativas de acción en las escrituras se conocen como políticas de escritura y constituyen una de las decisiones más importantes en el diseño del sistema de memoria.

¿Que ocurre durante las escrituras?

- Una variable que está en el caché también está alojada en alguna dirección de la DRAM.
- Idealmente ambos copias de la variable deben mantener el mismo valor (al menos siempre que sea posible).
- Cuando el procesador la modifica ésta condición deja de cumplirse. Hay varios modos de actuar, dependiendo de si el sistema dispone de una sola CPU o mas de una.
- Estas alternativas de acción en las escrituras se conocen como políticas de escritura y constituyen una de las decisiones más importantes en el diseño del sistema de memoria.

¿Que ocurre durante las escrituras?

- Una variable que está en el caché también está alojada en alguna dirección de la DRAM.
- Idealmente ambos copias de la variable deben mantener el mismo valor (al menos siempre que sea posible).
- Cuando el procesador la modifica ésta condición deja de cumplirse. Hay varios modos de actuar, dependiendo de si el sistema dispone de una sola CPU o mas de una.
- Estas alternativas de acción en las escrituras se conocen como políticas de escritura y constituyen una de las decisiones más importantes en el diseño del sistema de memoria.

¿Que ocurre durante las escrituras?

- Una variable que está en el caché también está alojada en alguna dirección de la DRAM.
- Idealmente ambos copias de la variable deben mantener el mismo valor (al menos siempre que sea posible).
- Cuando el procesador la modifica ésta condición deja de cumplirse. Hay varios modos de actuar, dependiendo de si el sistema dispone de una sola CPU o mas de una.
- Estas alternativas de acción en las escrituras se conocen como políticas de escritura y constituyen una de las decisiones más importantes en el diseño del sistema de memoria.

Políticas de escritura

Write through el procesador escribe en la DRAM y el controlador cache refresca el cache con el dato actualizado. Garantiza coherencia entre ambos datos de manera absoluta. Pero a cambio penaliza las escrituras con el tiempo de acceso a DRAM siempre. La performance se degrada en las escrituras

Write through buffered Es una opción mejorada de **Write through**, ya que el procesador actualiza la SRAM cache, y el controlador cache luego actualiza la copia en memoria DRAM mientras el procesador continúa ejecutando instrucciones y usando datos de la memoria cache. Para ello el Controlador cache debe disponer de un buffer de escrituras para encolar en él las operaciones de escritura a memoria.

Políticas de escritura

Write through el procesador escribe en la DRAM y el controlador cache refresca el cache con el dato actualizado. Garantiza coherencia entre ambos datos de manera absoluta. Pero a cambio penaliza las escrituras con el tiempo de acceso a DRAM siempre. La performance se degrada en las escrituras

Write through buffered Es una opción mejorada de **Write through**, ya que el procesador actualiza la SRAM cache, y el controlador cache luego actualiza la copia en memoria DRAM mientras el procesador continúa ejecutando instrucciones y usando datos de la memoria cache. Para ello el Controlador cache debe disponer de un buffer de escrituras para encolar en él las operaciones de escritura a memoria.

Políticas de escritura

Write through el procesador escribe en la DRAM y el controlador cache refresca el cache con el dato actualizado. Garantiza coherencia entre ambos datos de manera absoluta. Pero a cambio penaliza las escrituras con el tiempo de acceso a DRAM siempre. La performance se degrada en las escrituras

Write through buffered Es una opción mejorada de **Write through**, ya que el procesador actualiza la SRAM cache, y el controlador cache luego actualiza la copia en memoria DRAM mientras el procesador continúa ejecutando instrucciones y usando datos de la memoria cache. Para ello el Controlador cache debe disponer de un buffer de escrituras para encolar en él las operaciones de escritura a memoria.

Políticas de escritura

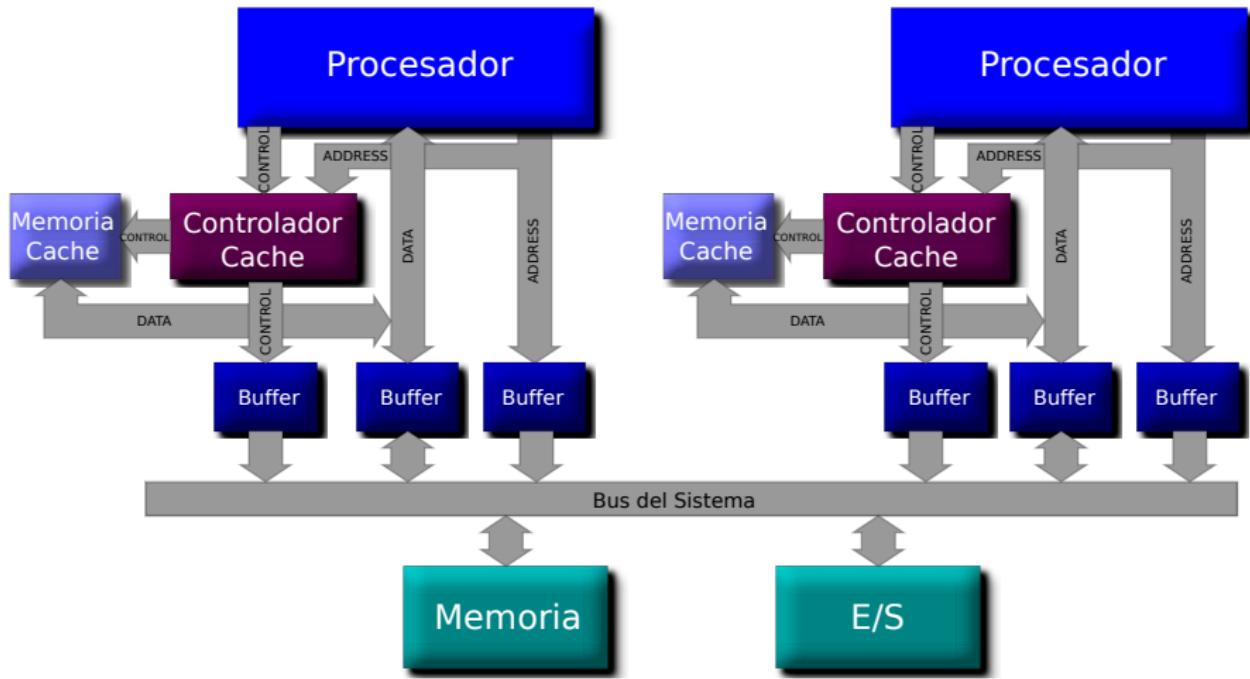
Copy back Se marcan como modificadas (o dirty) las líneas de la memoria cache que contienen variables en cuya dirección el procesador escribió. Nunca se actualiza la copia de memoria, excepto en el momento de eliminar la línea del caché. En ese momento, el controlador cache deberá actualizar la copia de DRAM.

Políticas de escritura

Copy back Se marcan como modificadas (o dirty) las líneas de la memoria cache que contienen variables en cuya dirección el procesador escribió. Nunca se actualiza la copia de memoria, excepto en el momento de eliminar la línea del caché. En ese momento, el controlador cache deberá actualizar la copia de DRAM.

Si el procesador realiza un miss mientras el controlador cache está accediendo a la DRAM para actualizar el valor, deberá esperar hasta que controlador cache termine la actualización para recibir desde éste la habilitación de las líneas de control para acceder a la DRAM.

Coherencia en sistemas SMP



Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - Con una latencia de写回 (Write Back) que es la duración entre la modificación en el cache y la actualización en la memoria principal.
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - ① Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - ② Lo antes posible (sin afectar performance), si el Controlador Cache implementa Write Through buffered.
 - ③ Cuando el dato sea descartado del cache (si usa copy back).
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - ① Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - ② Lo antes posible (sin afectar performance), si el Controlador Cache implementa Write Through buffered.
 - ③ Cuando el dato sea descartado del cache (si usa copy back).
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - ① Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - ② Lo antes posible (sin afectar performance), si el Controlador Cache implementa Write Through buffered.
 - ③ Cuando el dato sea descartado del cache (si usa copy back).
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - ① Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - ② Lo antes posible (sin afectar performance), si el Controlador Cache implementa Write Through buffered.
 - ③ Cuando el dato sea descartado del cache (si usa copy back).
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - ① Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - ② Lo antes posible (sin afectar performance), si el Controlador Cache implementa Write Through buffered.
 - ③ Cuando el dato sea descartado del cache (si usa copy back).
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

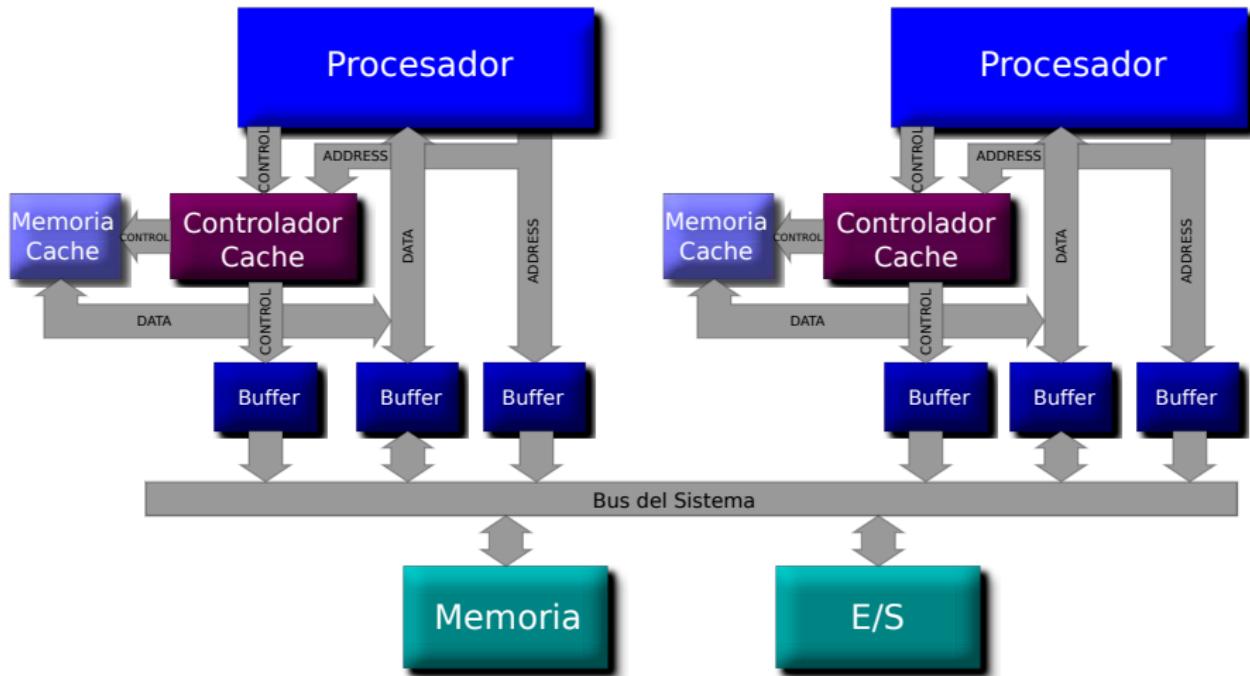
Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - ① Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - ② Lo antes posible (sin afectar performance), si el Controlador Cache implementa Write Through buffered.
 - ③ Cuando el dato sea descartado del cache (si usa copy back).
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

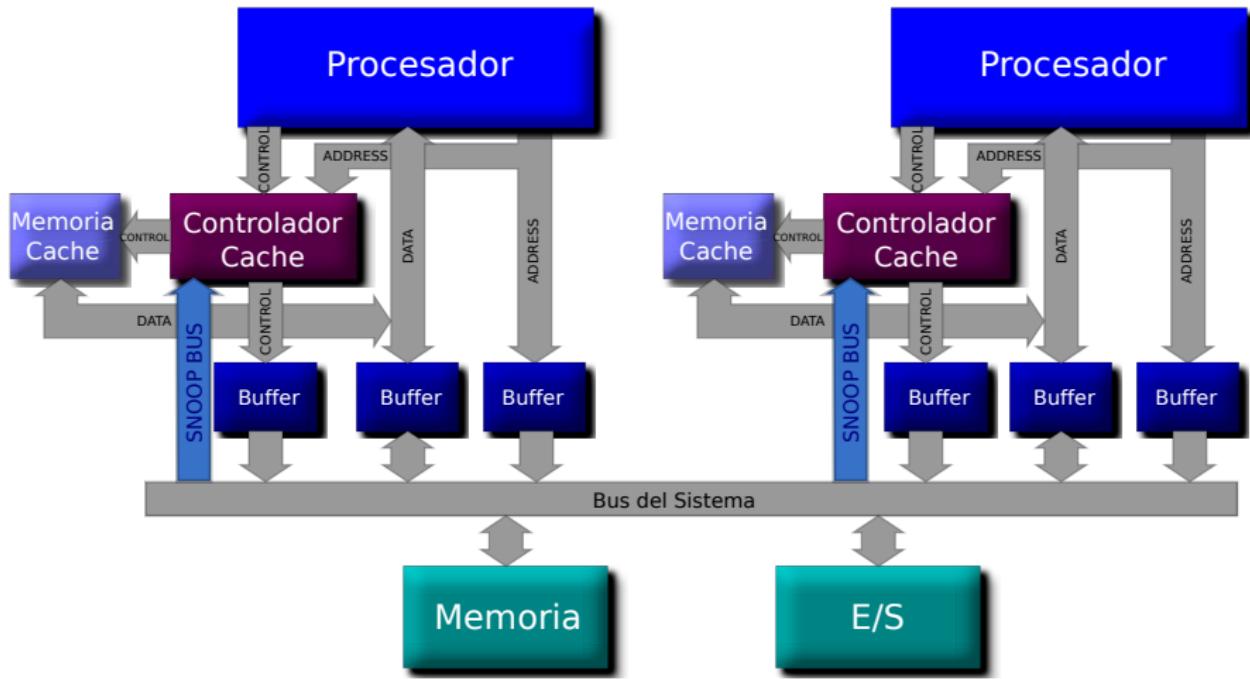
Coherencia en sistemas SMP

- En el diagrama anterior cuando un procesador modifica una variable, esta se modifica en el cache.
- De acuerdo con la política de escritura el dato se refrescará en la DRAM :
 - ① Inmediatamente (pero con un alto costo de performance) si el Controlador Cache implementa Write Through.
 - ② Lo antes posible (sin afectar performance), si el Controlador Cache implementa Write Through buffered.
 - ③ Cuando el dato sea descartado del cache (si usa copy back).
- En esta situación el otro procesador necesita enterarse cuanto antes del cambio, por si tiene esa misma variable en su cache
- En principio parece que copy back no puede usarse cuando hay mas de una CPU.
- Pero antes de analizar esto necesitamos que el segundo procesador se entere del cambio en la variable.

Coherencia en sistemas SMP



Solución: hardware adicional: El Snoop bus



SNOOP Bus

- El Controlador Cache marca en su directorio cache interno las direcciones de memoria que tiene almacenadas en el cache.
- Para saber si una variable que termina de ser modificada está o no en su cache necesita conocer su dirección en la DRAM (dirección física).
- El Snoop bus es un *conjunto de líneas entrantes al controlador cache* provenientes del **bus de Address** del sistema.
- De este modo el Controlador Cache espía (snoop) por el snoop bus cada operación de lectura y escritura sobre direcciones en la memoria del sistema y chequea si la tiene en su cache.
- Si la tiene en su caché, y detecta una escritura, la invalida.
- Además de las líneas del **bus de Address** del sistema también componen al Snoop Bus *las líneas MEMRD y MEMWR del bus de Control*, para saber si la dirección fue leída o escrita.

SNOOP Bus

- El Controlador Cache marca en su directorio cache interno las direcciones de memoria que tiene almacenadas en el cache.
- Para saber si una variable que termina de ser modificada está o no en su cache necesita conocer su dirección en la DRAM (dirección física).
- El Snoop bus es un *conjunto de líneas entrantes al controlador cache* provenientes del **bus de Address** del sistema.
- De este modo el Controlador Cache espía (snoop) por el snoop bus cada operación de lectura y escritura sobre direcciones en la memoria del sistema y chequea si la tiene en su cache.
- Si la tiene en su caché, y detecta una escritura, la invalida.
- Además de las líneas del **bus de Address** del sistema también componen al Snoop Bus *las líneas MEMRD y MEMWR del bus de Control*, para saber si la dirección fue leída o escrita.

SNOOP Bus

- El Controlador Cache marca en su directorio cache interno las direcciones de memoria que tiene almacenadas en el cache.
- Para saber si una variable que termina de ser modificada está o no en su cache necesita conocer su dirección en la DRAM (dirección física).
- El Snoop bus es un **conjunto de líneas entrantes al controlador cache** provenientes del **bus de Address** del sistema.
- De este modo el Controlador Cache espía (snoop) por el snoop bus cada operación de lectura y escritura sobre direcciones en la memoria del sistema y chequea si la tiene en su cache.
- Si la tiene en su caché, y detecta una escritura, la invalida.
- Además de las líneas del **bus de Address** del sistema también componen al Snoop Bus **las líneas MEMRD y MEMWR del bus de Control**, para saber si la dirección fue leída o escrita.

SNOOP Bus

- El Controlador Cache marca en su directorio cache interno las direcciones de memoria que tiene almacenadas en el cache.
- Para saber si una variable que termina de ser modificada está o no en su cache necesita conocer su dirección en la DRAM (dirección física).
- El Snoop bus es un **conjunto de líneas entrantes al controlador cache** provenientes del **bus de Address** del sistema.
- De este modo el Controlador Cache espía (snoop) por el snoop bus cada operación de lectura y escritura sobre direcciones en la memoria del sistema y chequea si la tiene en su cache.
- Si la tiene en su caché, y detecta una escritura, la invalida.
- Además de las líneas del **bus de Address** del sistema también componen al Snoop Bus **las líneas MEMRD y MEMWR del bus de Control**, para saber si la dirección fue leída o escrita.

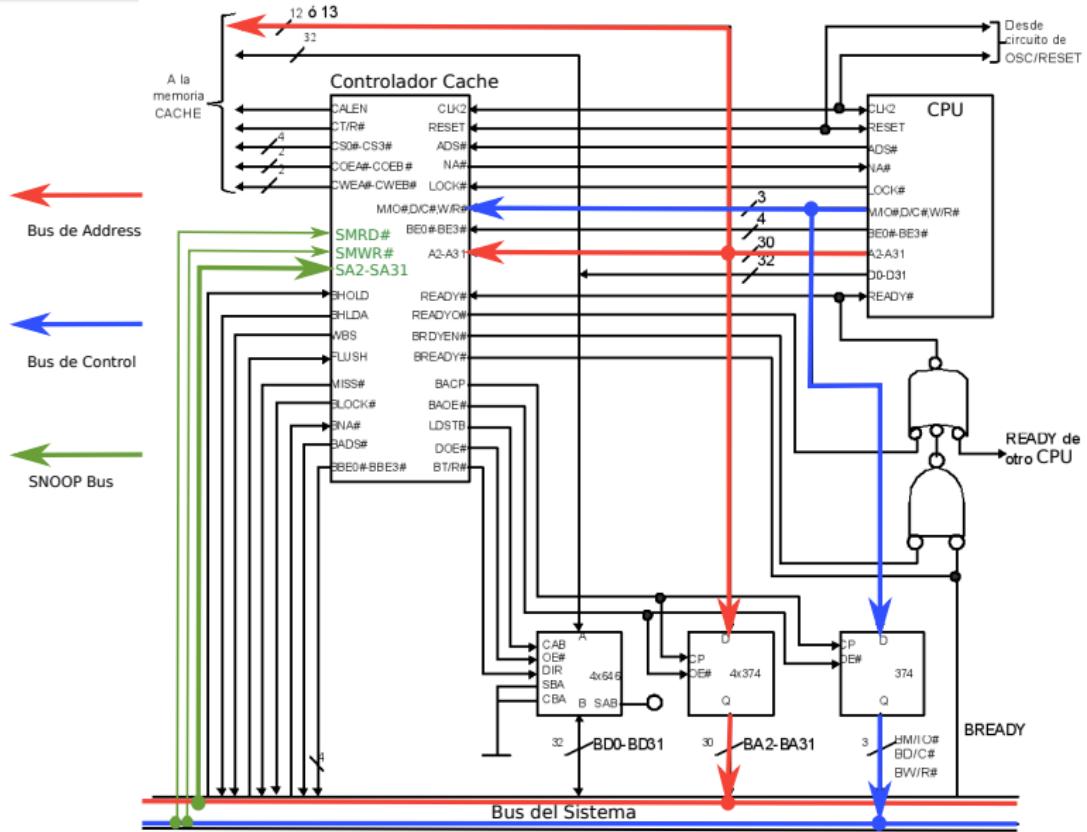
SNOOP Bus

- El Controlador Cache marca en su directorio cache interno las direcciones de memoria que tiene almacenadas en el cache.
- Para saber si una variable que termina de ser modificada está o no en su cache necesita conocer su dirección en la DRAM (dirección física).
- El Snoop bus es un **conjunto de líneas entrantes al controlador cache** provenientes del **bus de Address** del sistema.
- De este modo el Controlador Cache espía (snoop) por el snoop bus cada operación de lectura y escritura sobre direcciones en la memoria del sistema y chequea si la tiene en su cache.
- Si la tiene en su caché, y detecta una escritura, la invalida.
- Además de las líneas del **bus de Address** del sistema también componen al Snoop Bus **las líneas MEMRD y MEMWR del bus de Control**, para saber si la dirección fue leída o escrita.

SNOOP Bus

- El Controlador Cache marca en su directorio cache interno las direcciones de memoria que tiene almacenadas en el cache.
- Para saber si una variable que termina de ser modificada está o no en su cache necesita conocer su dirección en la DRAM (dirección física).
- El Snoop bus es un **conjunto de líneas entrantes al controlador cache** provenientes del **bus de Address** del sistema.
- De este modo el Controlador Cache espía (snoop) por el snoop bus cada operación de lectura y escritura sobre direcciones en la memoria del sistema y chequea si la tiene en su cache.
- Si la tiene en su caché, y detecta una escritura, la invalida.
- Además de las líneas del **bus de Address** del sistema también componen al Snoop Bus **las líneas MEMRD y MEMWR del bus de Control**, para saber si la dirección fue leída o escrita.

SNOOP Bus - Diagrama detallado



SNOOP Bus - Diagrama detallado

- Si bien el diagrama del slide anterior es algo antiguo en términos tecnológicos, permite ver a simple vista las señales involucradas. Este es el efecto que buscamos.
- Sirve para apreciar por ejemplo que *el SNOOP bus no es un bus entre los diferentes controladores*, sino una conexión que cada controlador cache toma desde el bus del sistema para espiar que es lo que hace el resto de los procesadores con la memoria.
- Observar que las líneas **SA31-SA32** son las mismas líneas del Bus de Address pero entrantes al controlador cache.
- Lo mismo ocurre con las señales **SMRD#** y **SMWR#**, que toman las señales de lectura y escritura de memoria pertenecientes al bus de control.
- De este modo el controlador puede espiar lo que hace el resto de los procesadores en el bus del sistema.

SNOOP Bus - Diagrama detallado

- Si bien el diagrama del slide anterior es algo antiguo en términos tecnológicos, permite ver a simple vista las señales involucradas. Este es el efecto que buscamos.
- Sirve para apreciar por ejemplo que ***el SNOOP bus no es un bus entre los diferentes controladores***, sino una conexión que cada controlador cache toma desde el bus del sistema para espiar que es lo que hace el resto de los procesadores con la memoria.
- Observar que las líneas **SA31-SA32** son las mismas líneas del Bus de Address pero entrantes al controlador cache.
- Lo mismo ocurre con las señales **SMRD#** y **SMWR#**, que toman las señales de lectura y escritura de memoria pertenecientes al bus de control.
- De este modo el controlador puede espiar lo que hace el resto de los procesadores en el bus del sistema.

SNOOP Bus - Diagrama detallado

- Si bien el diagrama del slide anterior es algo antiguo en términos tecnológicos, permite ver a simple vista las señales involucradas. Este es el efecto que buscamos.
- Sirve para apreciar por ejemplo que ***el SNOOP bus no es un bus entre los diferentes controladores***, sino una conexión que cada controlador cache toma desde el bus del sistema para espiar que es lo que hace el resto de los procesadores con la memoria.
- Observar que las líneas **SA31-SA32** son las mismas líneas del Bus de Address pero entrantes al controlador cache.
- Lo mismo ocurre con las señales **SMRD# y SMWR#**, que toman las señales de lectura y escritura de memoria pertenecientes al bus de control.
- De este modo el controlador puede espiar lo que hace el resto de los procesadores en el bus del sistema.

SNOOP Bus - Diagrama detallado

- Si bien el diagrama del slide anterior es algo antiguo en términos tecnológicos, permite ver a simple vista las señales involucradas. Este es el efecto que buscamos.
- Sirve para apreciar por ejemplo que ***el SNOOP bus no es un bus entre los diferentes controladores***, sino una conexión que cada controlador cache toma desde el bus del sistema para espiar que es lo que hace el resto de los procesadores con la memoria.
- Observar que las líneas **SA31-SA32** son las mismas líneas del Bus de Address pero entrantes al controlador cache.
- Lo mismo ocurre con las señales **SMRD# y SMWR#**, que toman las señales de lectura y escritura de memoria pertenecientes al bus de control.
- De este modo el controlador puede espiar lo que hace el resto de los procesadores en el bus del sistema.

SNOOP Bus - Diagrama detallado

- Si bien el diagrama del slide anterior es algo antiguo en términos tecnológicos, permite ver a simple vista las señales involucradas. Este es el efecto que buscamos.
- Sirve para apreciar por ejemplo que ***el SNOOP bus no es un bus entre los diferentes controladores***, sino una conexión que cada controlador cache toma desde el bus del sistema para espiar que es lo que hace el resto de los procesadores con la memoria.
- Observar que las líneas **SA31-SA32** son las mismas líneas del Bus de Address pero entrantes al controlador cache.
- Lo mismo ocurre con las señales **SMRD# y SMWR#**, que toman las señales de lectura y escritura de memoria pertenecientes al bus de control.
- De este modo el controlador puede espiar lo que hace el resto de los procesadores en el bus del sistema.

Protocolos de coherencia

- El Snoop bus resuelve el problema de la coherencia
- Sin embargo, hay lugar para algunas optimizaciones considerando las políticas de escritura.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización.
- Pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.
- Para poder utilizar este método de escritura siempre que sea posible y reemplazarlo solo cuando la misma dirección física está presente en por lo menos dos caches, se han desarrollado protocolos de coherencia.
- En más popular es M.E.S.I.

Protocolos de coherencia

- El Snoop bus resuelve el problema de la coherencia
- Sin embargo, hay lugar para algunas optimizaciones considerando las políticas de escritura.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización.
- Pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.
- Para poder utilizar este método de escritura siempre que sea posible y reemplazarlo solo cuando la misma dirección física está presente en por lo menos dos caches, se han desarrollado protocolos de coherencia.
- En más popular es M.E.S.I.

Protocolos de coherencia

- El Snoop bus resuelve el problema de la coherencia
- Sin embargo, hay lugar para algunas optimizaciones considerando las políticas de escritura.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización.
- Pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.
- Para poder utilizar este método de escritura siempre que sea posible y reemplazarlo solo cuando la misma dirección física está presente en por lo menos dos caches, se han desarrollado protocolos de coherencia.
- En más popular es M.E.S.I.

Protocolos de coherencia

- El Snoop bus resuelve el problema de la coherencia
- Sin embargo, hay lugar para algunas optimizaciones considerando las políticas de escritura.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización.
- Pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.
- Para poder utilizar este método de escritura siempre que sea posible y reemplazarlo solo cuando la misma dirección física está presente en por lo menos dos caches, se han desarrollado protocolos de coherencia.
- En más popular es M.E.S.I.

Protocolos de coherencia

- El Snoop bus resuelve el problema de la coherencia
- Sin embargo, hay lugar para algunas optimizaciones considerando las políticas de escritura.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización.
- Pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.
- Para poder utilizar este método de escritura siempre que sea posible y reemplazarlo solo cuando la misma dirección física está presente en por lo menos dos caches, se han desarrollado protocolos de coherencia.
- En más popular es M.E.S.I.

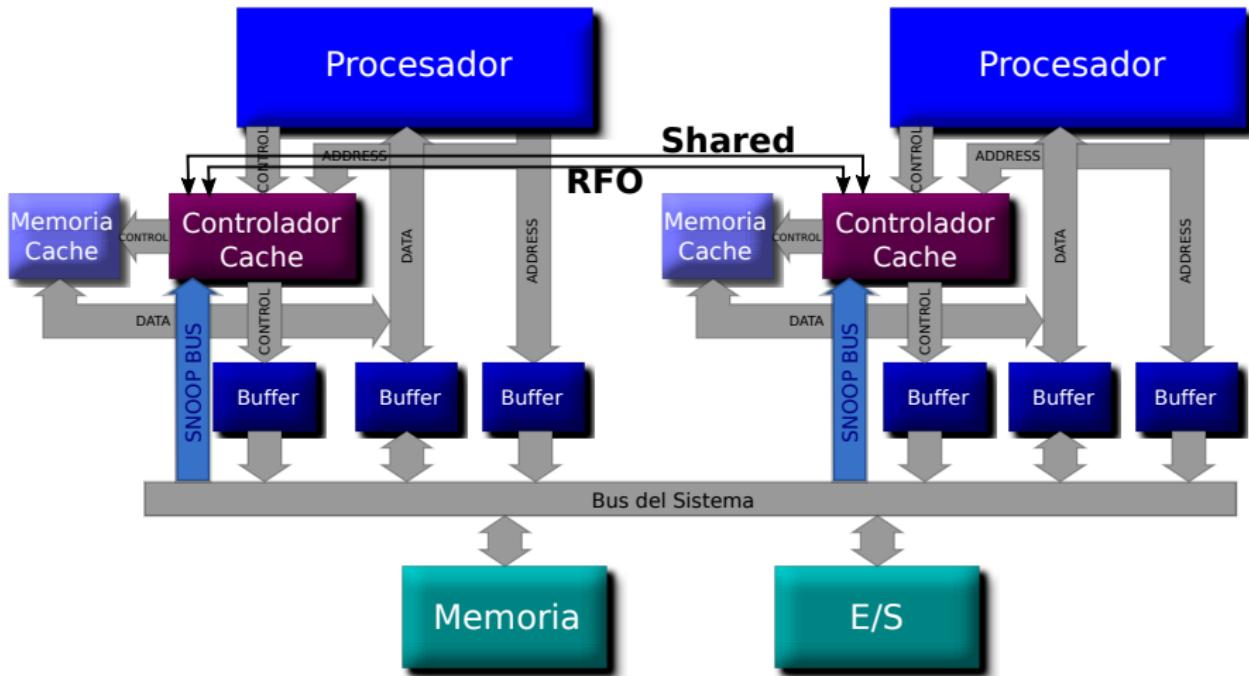
Protocolos de coherencia

- El Snoop bus resuelve el problema de la coherencia
- Sin embargo, hay lugar para algunas optimizaciones considerando las políticas de escritura.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización.
- Pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.
- Para poder utilizar este método de escritura siempre que sea posible y reemplazarlo solo cuando la misma dirección física está presente en por lo menos dos caches, se han desarrollado protocolos de coherencia.
- En más popular es M.E.S.I.

Protocolo MESI

- M - Modified** : Línea presente solamente en éste cache que varió respecto de su valor en memoria del sistema (dirty). Requiere write back hacia la memoria del sistema antes que otro procesador lea desde allí el dato (que ya no es válido).
- E – Exclusive** Línea presente solo en esta cache, que coincide con la copia en memoria principal (clean).
- S – Shared** Línea del cache presente y puede estar almacenada en los caches de otros procesadores.
- I – Invalid** Línea de cache no es válida.
- Aplica a cache L1 de datos y L2/L3
 - Para cache L1 de código solo Shared e Invalid

Coherencia en sistemas SMP con MESI



Operación del Protocolo MESI

- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- Las líneas inválidas se buscan en DRAM, o las provee otro Controlador Cache si tiene esa línea (en ese caso activa la señal Shared para indicar que está compartida). El estado en el cache que recibe la pasa a **Shared** o a **Exclusive**.
- El controlador cache que posee líneas en estado **Exclusive**, monitorea a través del snoop bus cada transacción sobre la DRAM.

Operación del Protocolo MESI

- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- Las líneas inválidas se buscan en DRAM, o las provee otro Controlador Cache si tiene esa línea (en ese caso activa la señal Shared para indicar que está compartida). El estado en el cache que recibe la pasa a **Shared** o a **Exclusive**.
- El controlador cache que posee líneas en estado **Exclusive**, monitorea a través del snoop bus cada transacción sobre la DRAM.
 - Si detecta un acceso a una línea que tiene almacenada en su cache, la cambia a **Shared** y activa la línea Shared para enviar un broadcast al resto, y activa la lectura en su cache enviar el valor de la línea por el bus de datos del sistema.

Operación del Protocolo MESI

- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- Las líneas inválidas se buscan en DRAM, o las provee otro Controlador Cache si tiene esa línea (en ese caso activa la señal Shared para indicar que está compartida). El estado en el cache que recibe la pasa a **Shared** o a **Exclusive**.
- El controlador cache que posee líneas en estado **Exclusive**, monitorea a través del snoop bus cada transacción sobre la DRAM.
 - Si detecta un acceso a una línea que tiene almacenada en su cache, la cambia a **Shared** y activa la línea Shared para enviar un broadcast al resto, y activa la lectura en su cache enviar el valor de la línea por el bus de datos del sistema.
 - El que la está leyendo recibirá el broadcast lee la línea y la marca **Shared**

Operación del Protocolo MESI

- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- Las líneas inválidas se buscan en DRAM, o las provee otro Controlador Cache si tiene esa línea (en ese caso activa la señal Shared para indicar que está compartida). El estado en el cache que recibe la pasa a **Shared** o a **Exclusive**.
- El controlador cache que posee líneas en estado **Exclusive**, monitorea a través del snoop bus cada transacción sobre la DRAM.
 - Si detecta un acceso a una línea que tiene almacenada en su cache, la cambia a **Shared** y activa la línea Shared para enviar un broadcast al resto, y activa la lectura en su cache enviar el valor de la línea por el bus de datos del sistema.
 - El que la está leyendo recibirá el broadcast lee la línea y la marca **Shared**

Operación del Protocolo MESI

- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- Las líneas inválidas se buscan en DRAM, o las provee otro Controlador Cache si tiene esa línea (en ese caso activa la señal Shared para indicar que está compartida). El estado en el cache que recibe la pasa a **Shared** o a **Exclusive**.
- El controlador cache que posee líneas en estado **Exclusive**, monitorea a través del snoop bus cada transacción sobre la DRAM.
 - Si detecta un acceso a una línea que tiene almacenada en su cache, la cambia a **Shared** y activa la línea Shared para enviar un broadcast al resto, y activa la lectura en su cache enviar el valor de la línea por el bus de datos del sistema.
 - El que la está leyendo recibirá el broadcast lee la línea y la marca **Shared**

Operación del Protocolo MESI

- Una línea en estado **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere actualizar previamente la DRAM.
- Una línea en estado **Modified** o **Exclusive** puede escribirse desde su CPU en cualquier momento. En el caso de **Exclusive** pasará a **Modified**.
- En el caso en que se necesite escribir en una línea cuyo estado sea **Shared**, el protocolo indica que todas las demás caches que tienen esa línea la Invaliden previamente.
- Para ello se emplea una operación broadcast denominada **Request For Ownership**

Operación del Protocolo MESI

- Una línea en estado **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere actualizar previamente la DRAM.
- Una línea en estado **Modified** o **Exclusive** puede escribirse desde su CPU en cualquier momento. En el caso de **Exclusive** pasará a **Modified**.
- En el caso en que se necesite escribir en una línea cuyo estado sea **Shared**, el protocolo indica que todas las demás caches que tienen esa línea la Invaliden previamente.
- Para ello se emplea una operación broadcast denominada **Request For Ownership**

Operación del Protocolo MESI

- Una línea en estado **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere actualizar previamente la DRAM.
- Una línea en estado **Modified** o **Exclusive** puede escribirse desde su CPU en cualquier momento. En el caso de **Exclusive** pasará a **Modified**.
- En el caso en que se necesite escribir en una línea cuyo estado sea **Shared**, el protocolo indica que todas las demás caches que tienen esa línea la Invaliden previamente.
- Para ello se emplea una operación broadcast denominada **Request For Ownership**

Operación del Protocolo MESI

- Una línea en estado **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere actualizar previamente la DRAM.
- Una línea en estado **Modified** o **Exclusive** puede escribirse desde su CPU en cualquier momento. En el caso de **Exclusive** pasará a **Modified**.
- En el caso en que se necesite escribir en una línea cuyo estado sea **Shared**, el protocolo indica que todas las demás caches que tienen esa línea la Invaliden previamente.
- Para ello se emplea una operación broadcast denominada Request For Ownership

Operación del Protocolo MESI

- Una línea en estado **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere actualizar previamente la DRAM.
- Una línea en estado **Modified** o **Exclusive** puede escribirse desde su CPU en cualquier momento. En el caso de **Exclusive** pasará a **Modified**.
- En el caso en que se necesite escribir en una línea cuyo estado sea **Shared**, el protocolo indica que todas las demás caches que tienen esa línea la Invaliden previamente.
- Para ello se emplea una operación broadcast denominada **Request For Ownership**

Operación del Protocolo MESI

- Un cache que tiene una línea en estado **Modified** y detecta por el snoop bus una lectura de esa misma línea debe de algún modo “insertar” el dato mantenido en su línea ya que está incoherente con la DRAM (**Modified** y **Exclusive** usan copy back).
- Para esto realiza lo siguiente:
 - Activa la Línea RFO para indicar al lector que ese dato está incoherente.
- Una línea en estado **Shared** pasa **Invalid** cuando se recibe un **RFO**.

Operación del Protocolo MESI

- Un cache que tiene una línea en estado **Modified** y detecta por el snoop bus una lectura de esa misma línea debe de algún modo “insertar” el dato mantenido en su línea ya que está incoherente con la DRAM (**Modified** y **Exclusive** usan copy back).
- Para esto realiza lo siguiente:
 - ① Activa la Línea RFO para indicar al lector que ese dato está incoherente.
 - ② Escribe en DRAM el valor actual de la línea. El lector copia ese valor correcto a su cache cuando aparece en el bus de datos.
 - ③ Ambos pondrán esa línea en estado **Shared**.
- Una línea en estado **Shared** pasa **Invalid** cuando se recibe un **RFO**.

Operación del Protocolo MESI

- Un cache que tiene una línea en estado **Modified** y detecta por el snoop bus una lectura de esa misma línea debe de algún modo “insertar” el dato mantenido en su línea ya que está incoherente con la DRAM (**Modified** y **Exclusive** usan copy back).
- Para esto realiza lo siguiente:
 - ① Activa la Línea RFO para indicar al lector que ese dato está incoherente.
 - ② Escribe en DRAM el valor actual de la línea. El lector copia ese valor correcto a su cache cuando aparece en el bus de datos.
 - ③ Ambos pondrán esa línea en estado **Shared**.
- Una línea en estado **Shared** pasa **Invalid** cuando se recibe un **RFO**.

Operación del Protocolo MESI

- Un cache que tiene una línea en estado **Modified** y detecta por el snoop bus una lectura de esa misma línea debe de algún modo “insertar” el dato mantenido en su línea ya que está incoherente con la DRAM (**Modified** y **Exclusive** usan copy back).
- Para esto realiza lo siguiente:
 - ① Activa la Línea RFO para indicar al lector que ese dato está incoherente.
 - ② Escribe en DRAM el valor actual de la línea. El lector copia ese valor correcto a su cache cuando aparece en el bus de datos.
 - ③ Ambos pondrán esa línea en estado **Shared**.
- Una línea en estado **Shared** pasa **Invalid** cuando se recibe un **RFO**.

Operación del Protocolo MESI

- Un cache que tiene una línea en estado **Modified** y detecta por el snoop bus una lectura de esa misma línea debe de algún modo “insertar” el dato mantenido en su línea ya que está incoherente con la DRAM (**Modified** y **Exclusive** usan copy back).
- Para esto realiza lo siguiente:
 - ① Activa la Línea RFO para indicar al lector que ese dato está incoherente.
 - ② Escribe en DRAM el valor actual de la línea. El lector copia ese valor correcto a su cache cuando aparece en el bus de datos.
 - ③ Ambos pondrán esa línea en estado **Shared**.
- Una línea en estado **Shared** pasa **Invalid** cuando se recibe un **RFO**.

Operación del Protocolo MESI

- Un cache que tiene una línea en estado **Modified** y detecta por el snoop bus una lectura de esa misma línea debe de algún modo “insertar” el dato mantenido en su línea ya que está incoherente con la DRAM (**Modified** y **Exclusive** usan copy back).
- Para esto realiza lo siguiente:
 - ① Activa la Línea RFO para indicar al lector que ese dato está incoherente.
 - ② Escribe en DRAM el valor actual de la línea. El lector copia ese valor correcto a su cache cuando aparece en el bus de datos.
 - ③ Ambos pondrán esa línea en estado **Shared**.
- Una línea en estado **Shared** pasa **Invalid** cuando se recibe un **RFO**.

Operación del Protocolo MESI

- Los estados **Modified** y **Exclusive** siempre son precisos: apuntan a una línea de cache que solo está en este cache (señala ownership!).
- Las líneas **Shared** son imprecisas. Aunque otros caches descarten esta línea, esta acción no se informa, ni hay modo en que cada cache pueda llevar la cuenta de cuantos caches tienen la misma línea **Shared**. Por lo tanto nunca puede pasar a **Exclusive**.
- En este sentido el estado **Exclusive** es el mas apto para optimizar el mínimo de transacciones en el bus ya que al ser escrito cambia a **Modified** pero no informa nada al resto.

Operación del Protocolo MESI

- Los estados **Modified** y **Exclusive** siempre son precisos: apuntan a una línea de cache que solo está en este cache (señala ownership!).
- Las líneas **Shared** son imprecisas. Aunque otros caches descarten esta línea, esta acción no se informa, ni hay modo en que cada cache pueda llevar la cuenta de cuantos caches tienen la misma línea **Shared**. Por lo tanto nunca puede pasar a **Exclusive**.
- En este sentido el estado **Exclusive** es el mas apto para optimizar el mínimo de transacciones en el bus ya que al ser escrito cambia a **Modified** pero no informa nada al resto.

Operación del Protocolo MESI

- Los estados **Modified** y **Exclusive** siempre son precisos: apuntan a una línea de cache que solo está en este cache (señala ownership!).
- Las líneas **Shared** son imprecisas. Aunque otros caches descarten esta línea, esta acción no se informa, ni hay modo en que cada cache pueda llevar la cuenta de cuantos caches tienen la misma línea **Shared**. Por lo tanto nunca puede pasar a **Exclusive**.
- En este sentido el estado **Exclusive** es el mas apto para optimizar el mínimo de transacciones en el bus ya que al ser escrito cambia a **Modified** pero no informa nada al resto.

Read For Ownership

Definición

- El protocolo de coherencia combina una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por un controlador cache que trata de escribir una línea en estado **Shared** o **Invalid**.
- Resultado: el resto de los caches que tienen esta línea almacenada la invalidan.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en la memoria DRAM.
- O sea es una operación exclusiva que fuerza al resto de los controladores que tienen esa línea almacenada a invalidarla.

Read For Ownership

Definición

- El protocolo de coherencia combina una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por un controlador cache que trata de escribir una línea en estado **Shared** o **Invalid**.
- Resultado: el resto de los caches que tienen esta línea almacenada la invalidan.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en la memoria DRAM.
- O sea es una operación exclusiva que fuerza al resto de los controladores que tienen esa línea almacenada a invalidarla.

Read For Ownership

Definición

- El protocolo de coherencia combina una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por un controlador cache que trata de escribir una línea en estado **Shared** o **Invalid**.
- Resultado: el resto de los caches que tienen esta línea almacenada la invalidan.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en la memoria DRAM.
- O sea es una operación exclusiva que fuerza al resto de los controladores que tienen esa línea almacenada a invalidarla.

Read For Ownership

Definición

- El protocolo de coherencia combina una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por un controlador cache que trata de escribir una línea en estado **Shared** o **Invalid**.
- Resultado: el resto de los caches que tienen esta línea almacenada la invalidan.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en la memoria DRAM.
- O sea es una operación exclusiva que fuerza al resto de los controladores que tienen esa línea almacenada a invalidarla.

Read For Ownership

Definición

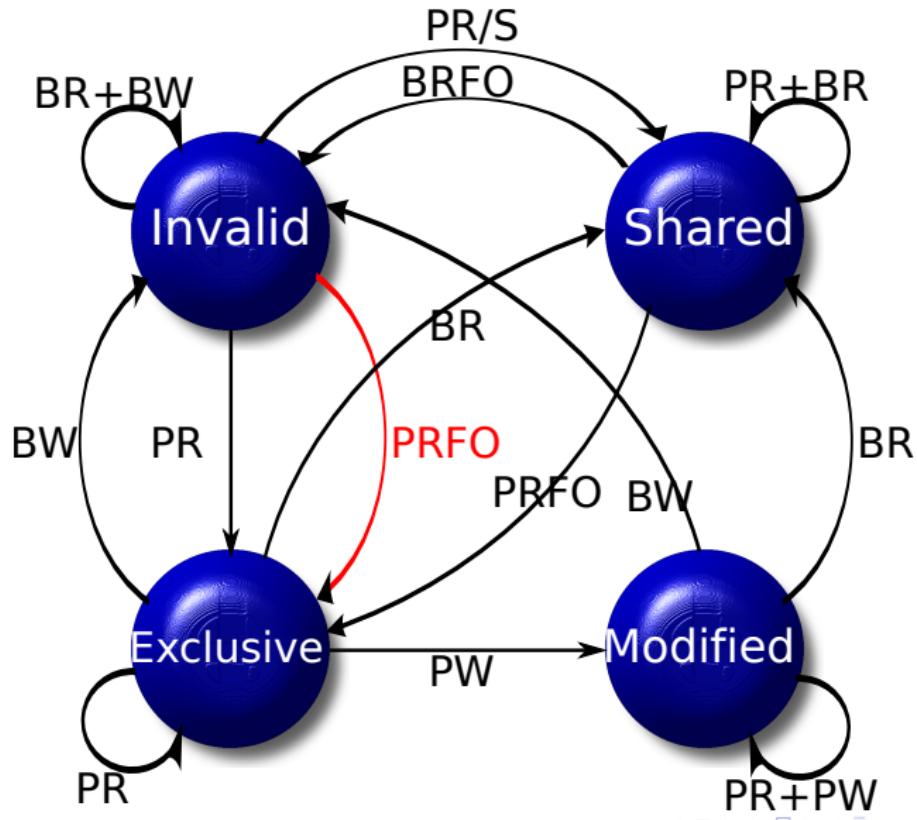
- El protocolo de coherencia combina una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por un controlador cache que trata de escribir una línea en estado **Shared** o **Invalid**.
- Resultado: el resto de los caches que tienen esta línea almacenada la invalidan.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en la memoria DRAM.
- O sea es una operación exclusiva que fuerza al resto de los controladores que tienen esa línea almacenada a invalidarla.

Read For Ownership

Definición

- El protocolo de coherencia combina una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por un controlador cache que trata de escribir una línea en estado **Shared** o **Invalid**.
- Resultado: el resto de los caches que tienen esta línea almacenada la invalidan.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en la memoria DRAM.
- O sea es una operación exclusiva que fuerza al resto de los controladores que tienen esa línea almacenada a invalidarla.

Protocolo MESI - Diagrama de estados



Temario

1

El sistema de Memoria

- Jerarquía de Memorias
- Principio de Vecindad o Localidad

2

Tecnologías de Memoria

3

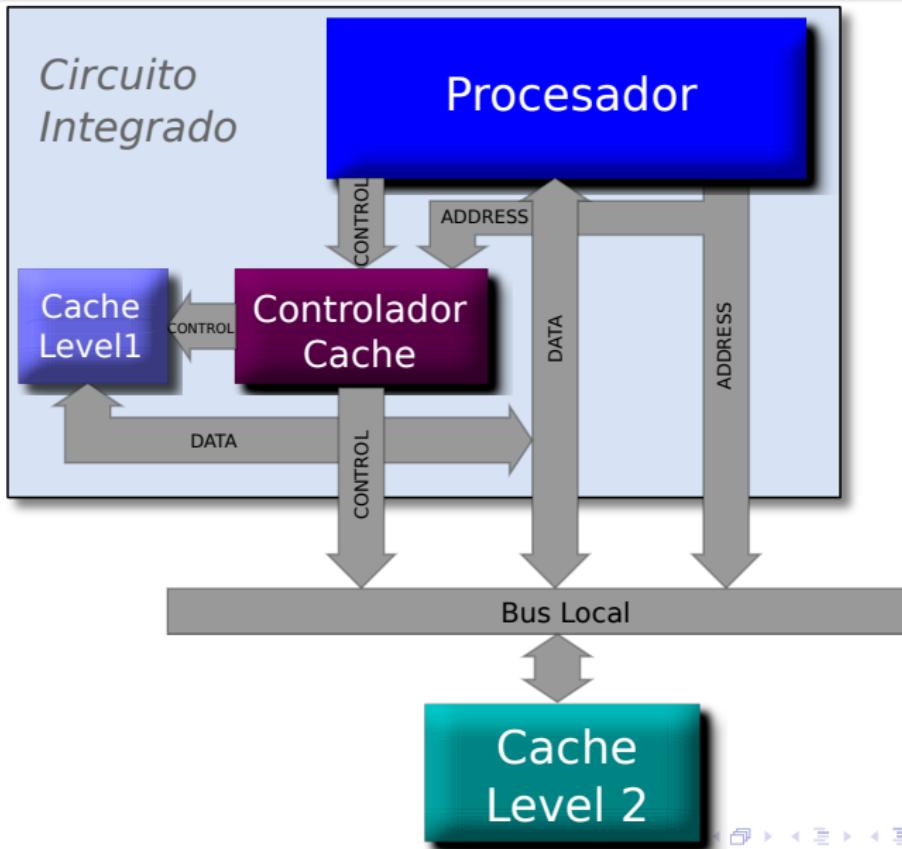
Memoria Cache

- Principio de Funcionamiento

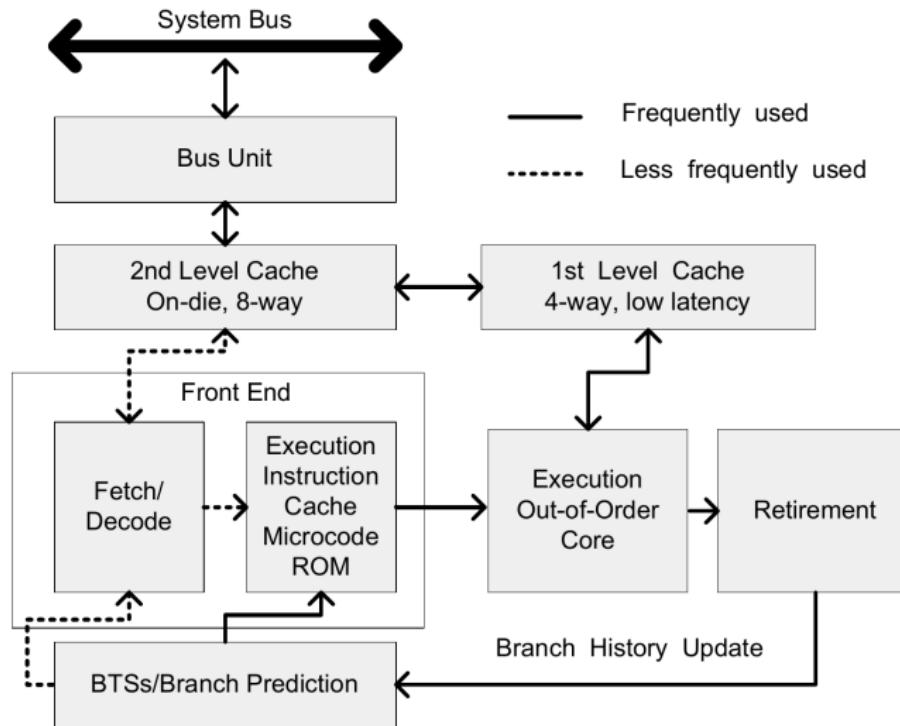
- Clasificación de memorias
- Memorias y velocidad del Procesador

- Hardware dedicado = + complejidad
- organización de un cache
- Coherencia de un cache
- Arquitecturas de cache avanzadas

Cache Multinivel

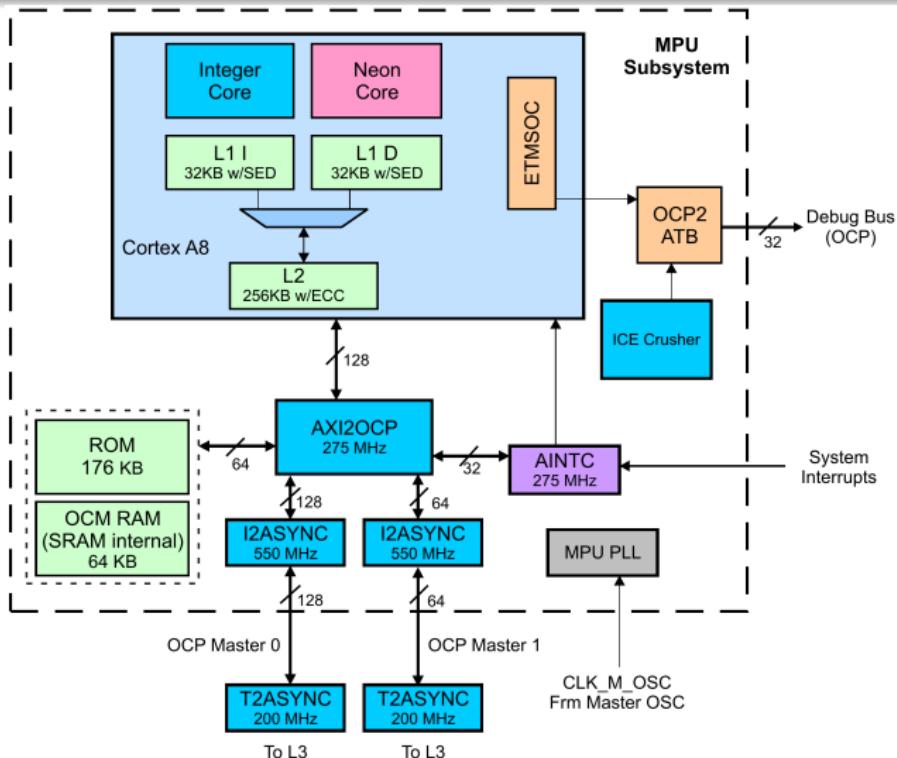


2do. Nivel On chip



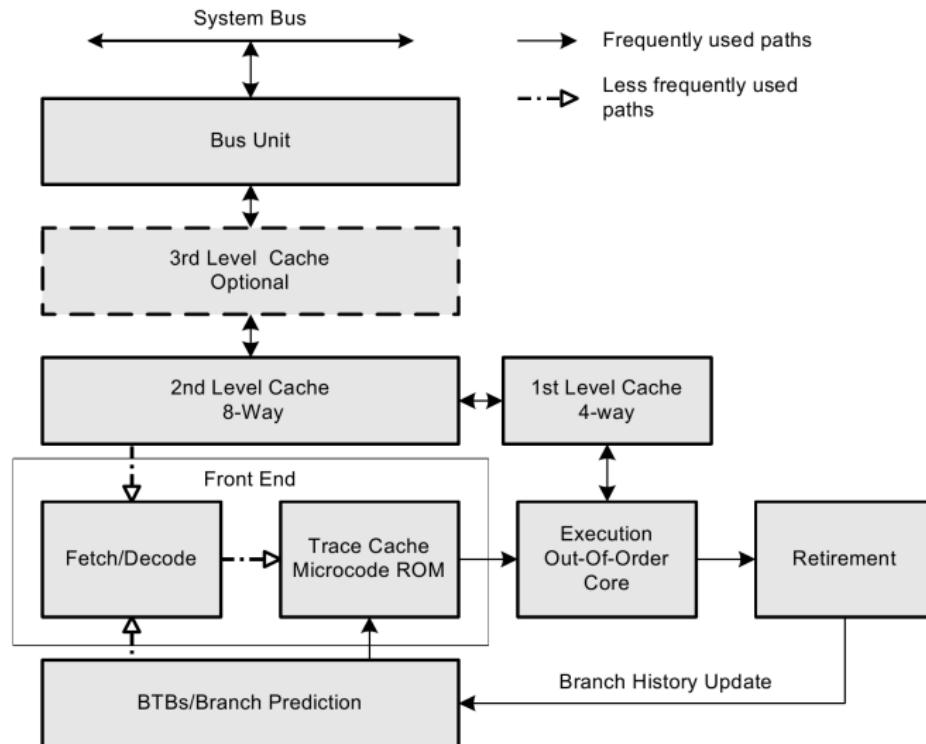
Intel P6 Architecture

2do. Nivel On chip



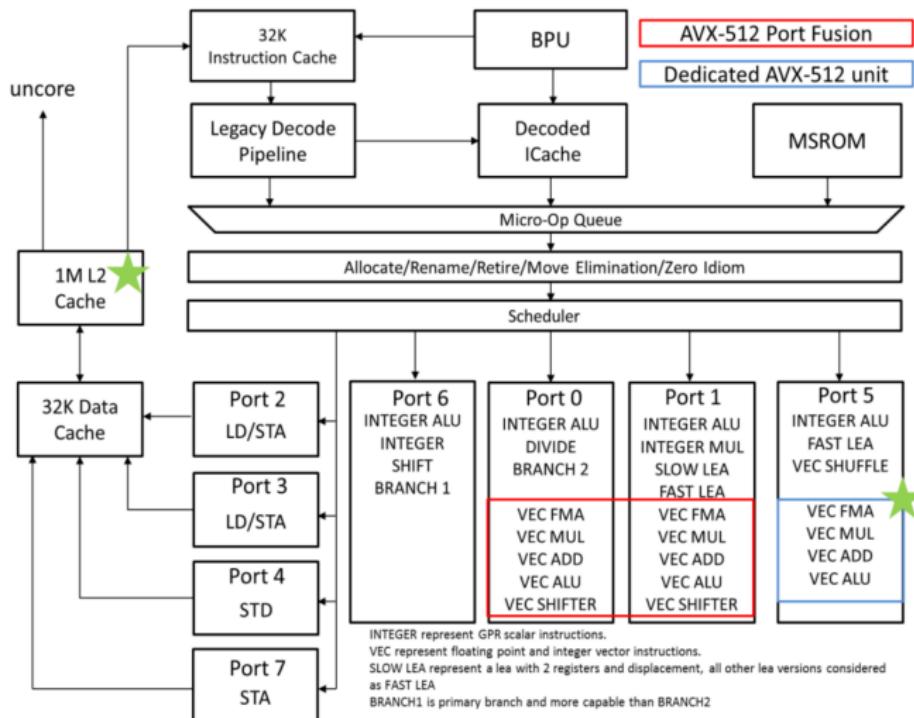
Cortex A8 Architecture

3er. Nivel On chip



Intel Netburst Architecture

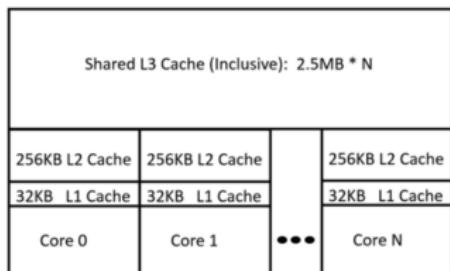
3er. Nivel On chip



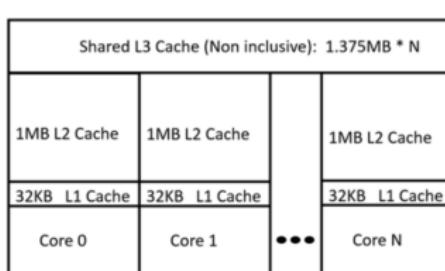
Intel Skylake Server Architecture

3er. Nivel On chip

Broadwell Server Cache Structure



Skylake Server Cache Structure



Intel Skylake vs Broadwell Server Architecture

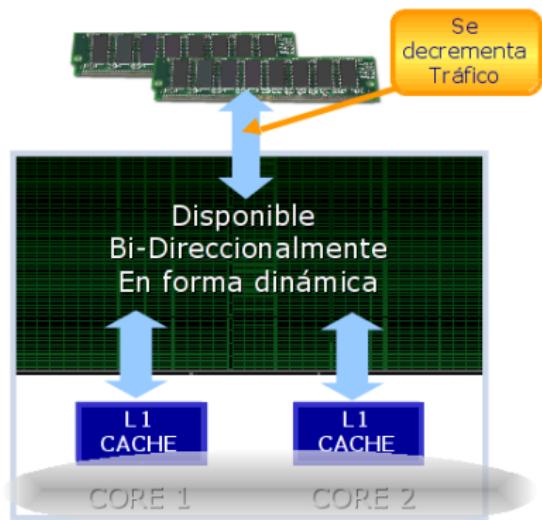
3er. Nivel On chip

Level	Capacity / Associativity	Line Size (bytes)	Fastest Latency ¹	Peak Bandwidth (bytes/cyc)	Sustained Bandwidth (bytes/cyc)	Update Policy
First Level Data	32 KB/ 8	64	4 cycle	96 (2x32B Load + 1*32B Store)	~81	Writeback
Instruction	32 KB/8	64	N/A	N/A	N/A	N/A
Second Level	256KB/4	64	12 cycle	64	~29	Writeback
Third Level (Shared L3)	Up to 2MB per core/Up to 16 ways	64	44	32	~18	Writeback

Intel Skylake Server Architecture - Parámetros de Cache

Smart Cache

L2 Compartida Microarquitectura Core



L2 Independiente

