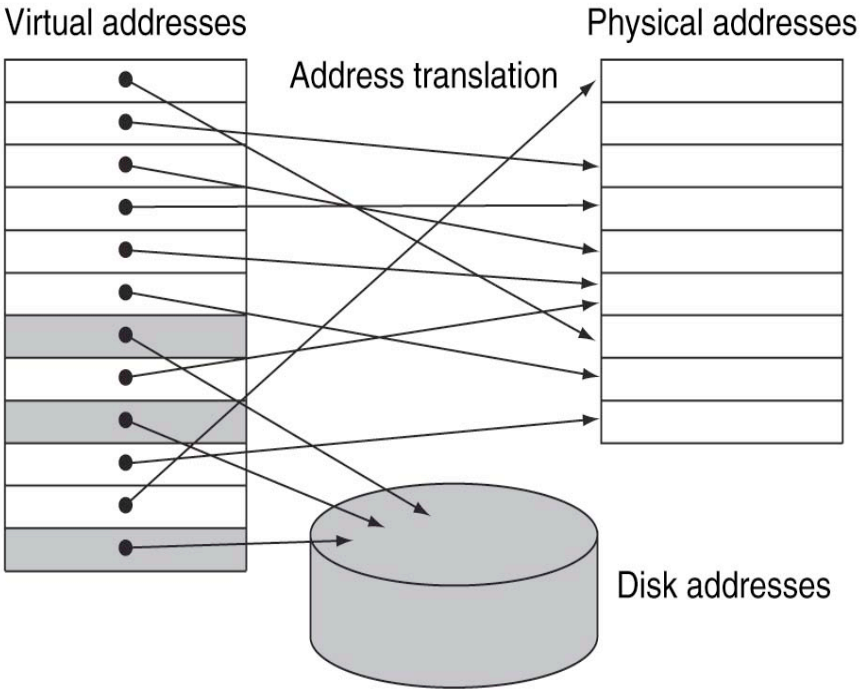
Memoria Virtual

- Es el nivel de la jerarquía que maneja la cache entre memoria principal y memoria secundaria.
- Permite que los programas se expandan más allá de los límites de la memoria principal.
- Permite compartir memoria de manera protegida.
- Es un desafío por la enorme penalidad.
- Técnicas:
  - Páginas grandes para reducir la tasa de fallos.
  - La correspondencia entre direcciones virtuales y físicas es asociativa
  - El S.O. reemplaza una página considerando LRU y bit de referencia

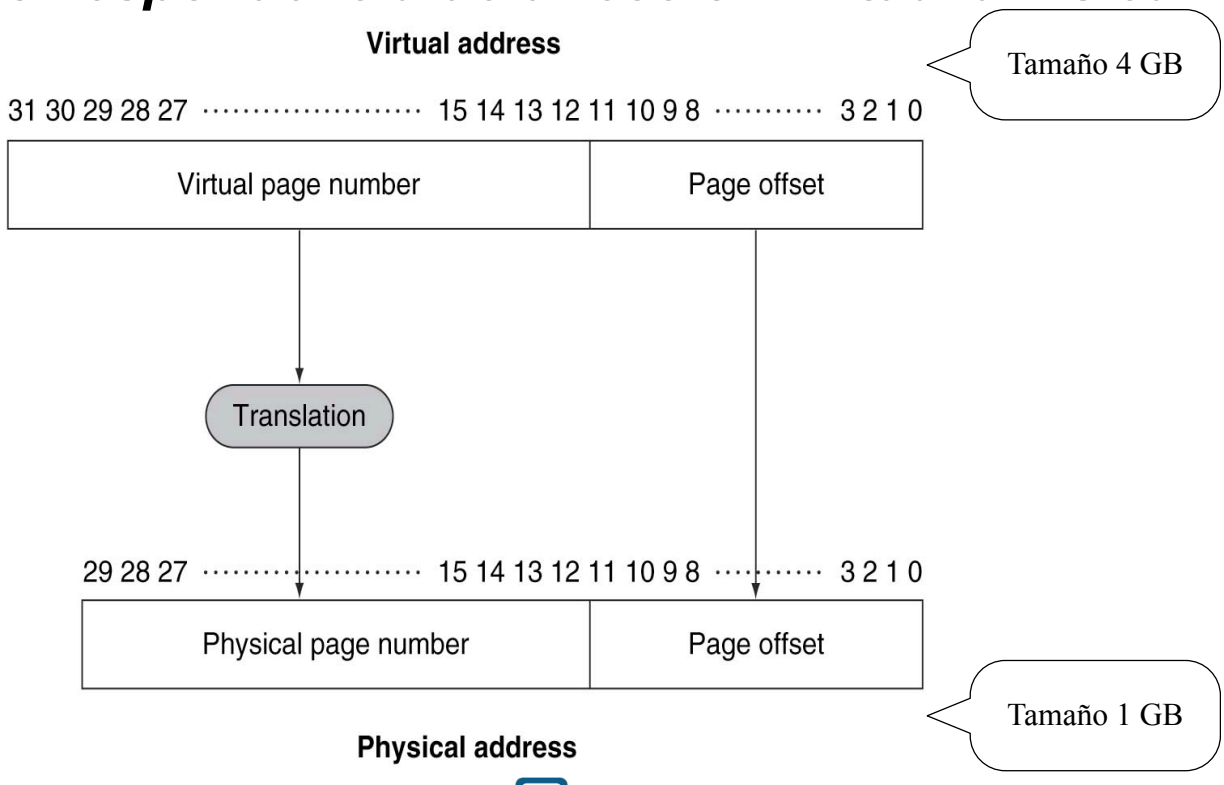
Memoria Virtual

- Usa write-back y Dirty Bit por el costo de escritura.
- Traduce la dirección virtual del programa a una dirección física de la memoria. Permite:
  - Compartir memoria de manera protegida.
  - Simplifica la asignación de memoria.
- Sólo el SO cambia la traducción de direcciones.
- Las páginas compartidas se controlan con bits de acceso.
- Se evitan accesos a la **tabla de páginas** con TLB.
- Cache, Memoria Virtual y Translation-Lookaside Buffer comparten principios y políticas.

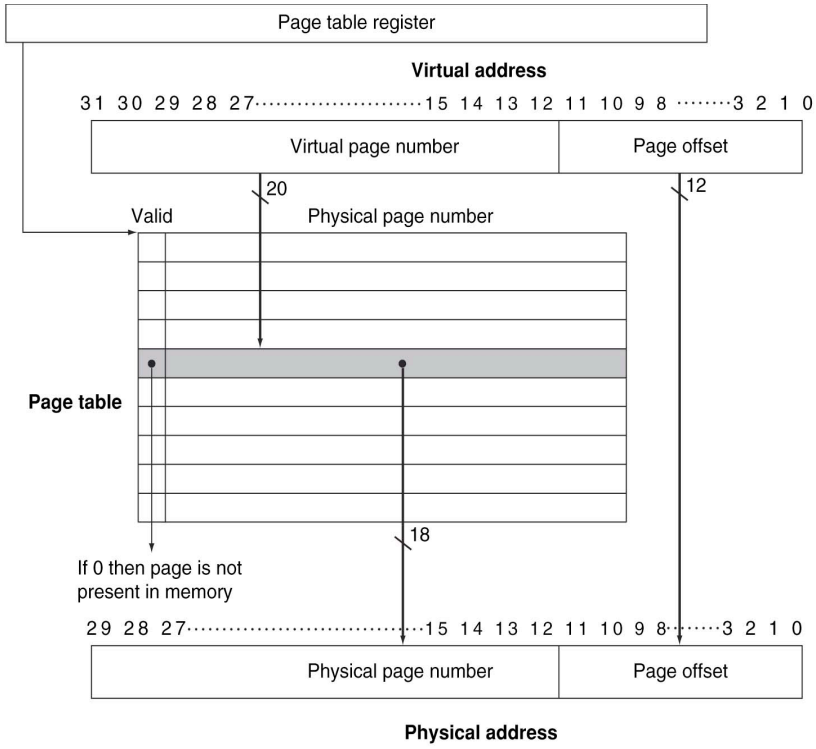
Dirección Virtual y Dirección Física



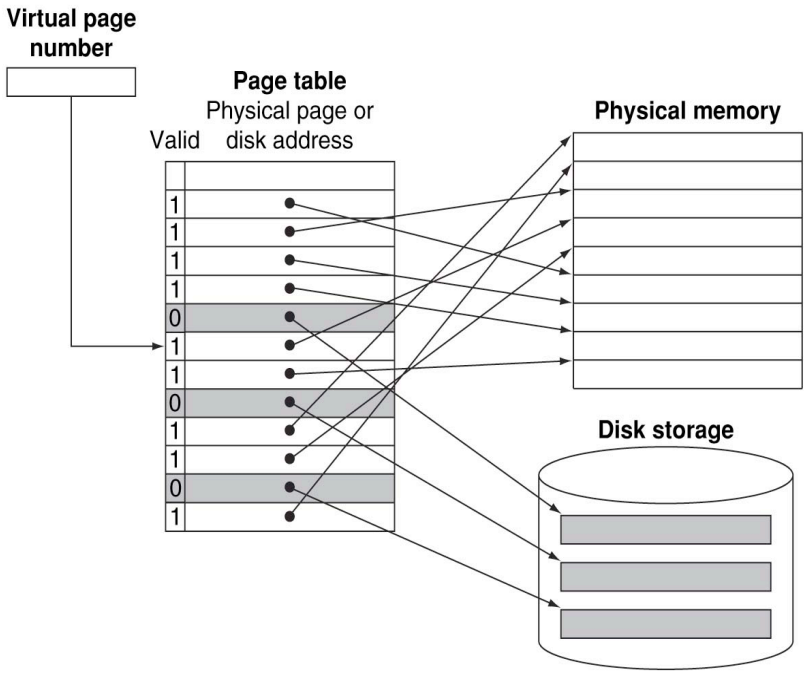
Correspondencia de dirección Virtual a Física



Traducción de dirección Virtual a Física



La tabla de páginas



Características compartidas por la jerarquía de memoria

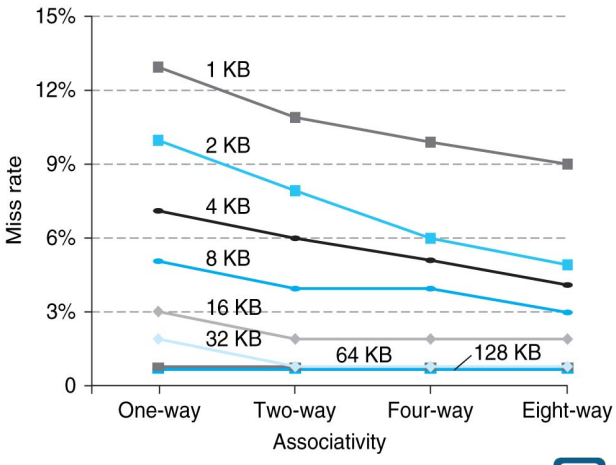
Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	15,000–50,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	500–4000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%

- ¿Dónde se puede ubicar un bloque?
- ¿Cómo se encuentra un bloque?
- ¿Qué bloque reemplazar en un miss?
- ¿Qué ocurre con las escrituras?
- Las tres C´s: modelo de comportamiento de la jerarquía.

¿Dónde se puede ubicar un bloque?

Varaciones sobre el esquema de conjunto asociativo

Nombre del esquema	Número de conjuntos	Bloques por conjunto
Correspondencia directa	Nro de bloque en cache	1
Conjunto asociativo	$\frac{\text{Nro de bloque en cache}}{\text{Asociatividad}}$	Asociatividad(2-16)
Totalmente asociativa	1	Nro de bloque en cache



Tasa de fallo a medida que aumenta la asociatividad, considerando ocho tamaños de caches

¿Cómo encontrar un bloque?

Esquemas de ubicación de los bloques

Asociatividad	Método de ubicación	Número de comparaciones
Correspondencia directa	Índice	1
Conjunto asociativo	Indexar el conjunto y buscar entre los elementos	Asociatividad(2-16)
Completa	Buscar en todas las entradas	Nro de bloques en cache
	Tabla Externa	0

- El esquema usado depende del costo del miss y del hard.
- Usar L2 permite mayor asociatividad.
- Totalmente Asociativas para caches pequeñas, con:
  - pocos comparadores
  - mejoras significativas

¿Qué bloque reemplazar?

- Principales estrategias: Aleatorio y LRU
- LRU (aproximado) en cuatro vías se pueden usar 2 bits:
  - 1 para un par de bloques LRU.
  - 1 para el bloque LRU en el par.
- Con caches de mayor asociatividad:
  - Algoritmo simple en hardware. Miss rate: Random>LRU (1,1)
  - En las grandes ambas fallan
- Memoria virtual usa LRU aproximado:
  - Reduce el miss rate, importante porque el miss penalty es grande.
  - En soft, usando bits de referencia. Razón: miss costosos y pocos.

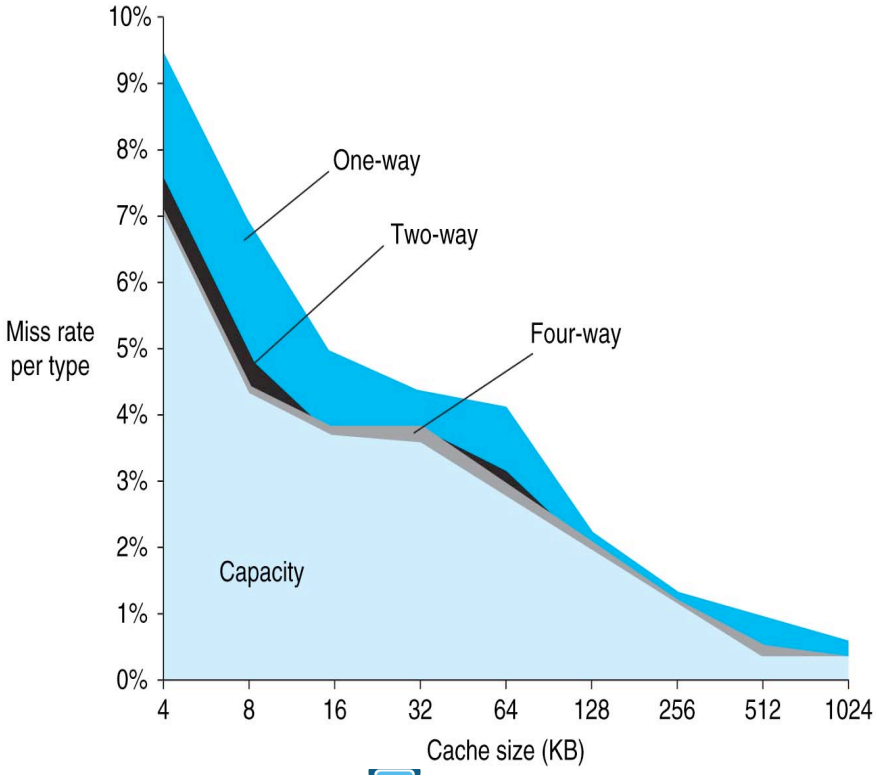
¿Qué ocurre en una escritura?

- Ventajas de write-back
  - Las palabras se escriben a la tasa de la cache.
  - Múltiples escrituras requieren un sólo acceso a memoria.
  - Puede escribir el bloque completo (high-bandwidth).
- Ventajas de write-through
  - Misses simples y baratos. Nunca escribe un bloque entero.
  - Más fácil de implementar. Usará un buffer de escritura.
- Memoria virtual sólo usa write-back porque:
  - Gran latencia al escribir en el nivel inferior.
  - La memoria no soporta la tasa de escritura del procesador.

Las tres C´s

- Origen de los misses.
- Cómo los cambios afectan los misses.
- Compulsory (Obligatorios) misses.
  - Primer acceso. Comienzo en frío
- Capacity (Por capacidad) misses.
  - No hay espacio suficiente. Bloques reemplazados y recargados.
- Conflict (Por conflictos) misses.
  - Conflictos por conjuntos de bloques. No se darían en Full Asociat.

Las tres fuentes de misses



Desafíos del diseño de la jerarquía

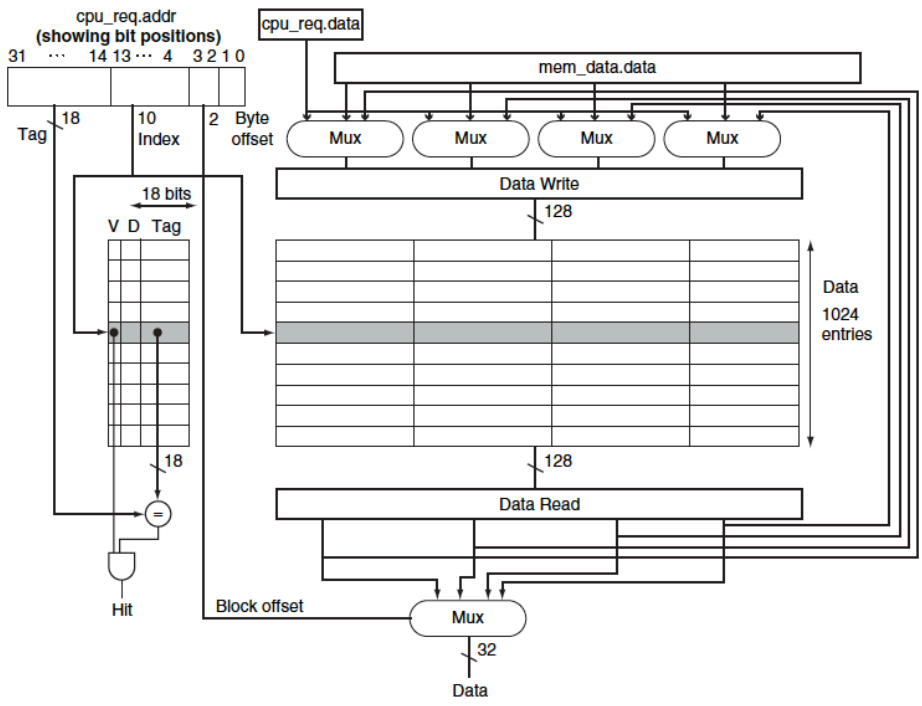
Design change	Effect on miss rate	Possible negative performance effect
Increase cache size	Decreases capacity misses	May increase access time
Increase associativity	Decreases miss rate due to conflict misses	May increase access time
Increase block size	Decreases miss rate for a wide range of block sizes due to spatial locality	Increases miss penalty. Very large block could increase miss rate

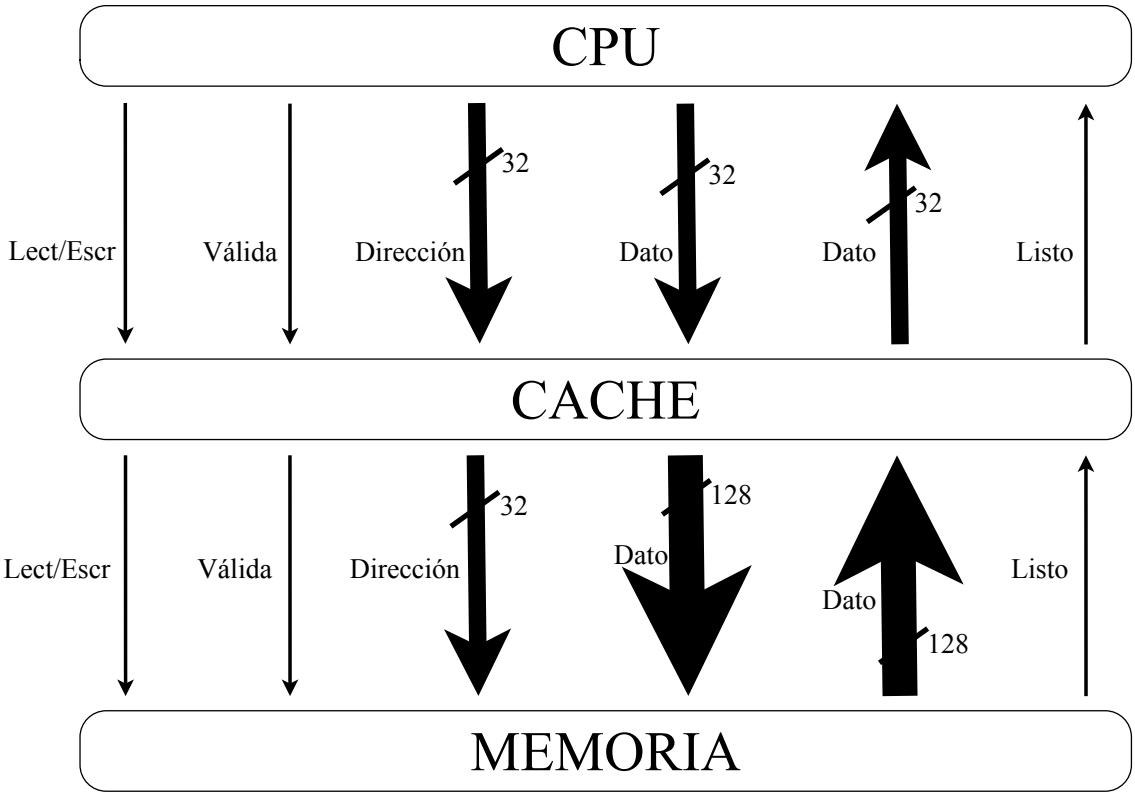
- Los cambios para mejorar la tasa de fallos pueden afectar todo el desempeño.
- La combinación de efectos positivos y negativos hace interesante el diseño de la jerarquía de memoria.

Diseño del controlador para una Cache simple

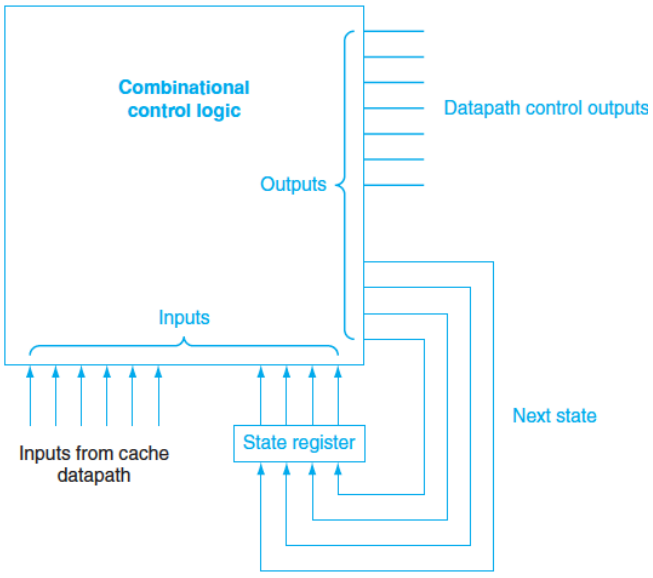
- Correspondencia Directa.
- Escritura demorada usando alojamiento del bloque.
- Tamaño del bloque 4 palabras.
- Tamaño de cache 16KiB, mantiene 1024 bloques.
- Direcciones de 32 bits.
- Cada bloque tiene bit de validez y modificado.

Diagrama de una cache simple

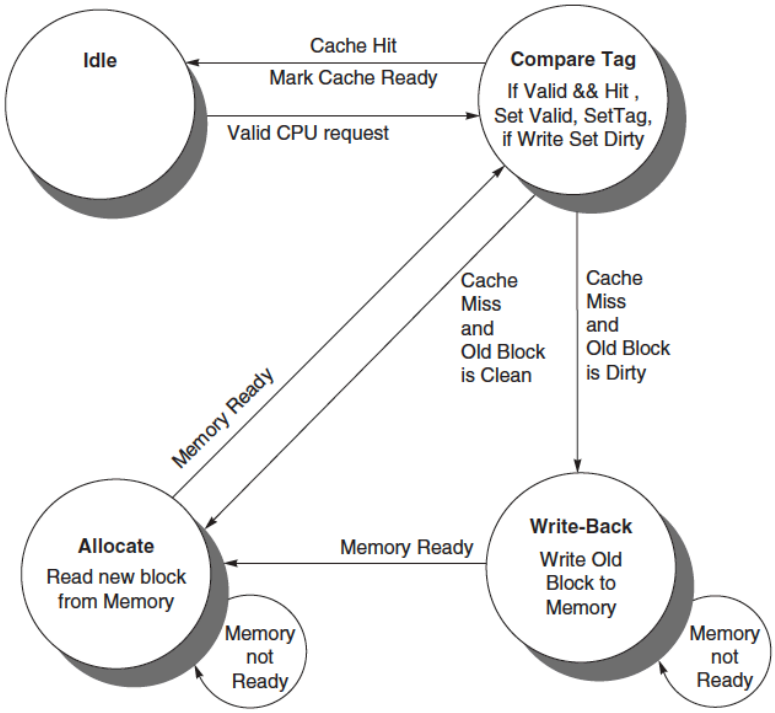




Implementación de un autómata



Controlador simple



Una traza de accesos y sus estados

		TAG (18)						INDEX (10)	2 2		0	1	2	3
0		0	A	B	C	3	4	5	C		✓	✓		✓
0		0	A	B	C	3	4	6	0		✓	✓		✓
0		0	A	B	C	3	4	6	4		✓	✓		
3		0	D	A	A	F	0	0	0		✓	✓		✓
0		0	A	B	C	3	4	6	8		✓	✓		
0		0	A	B	C	3	4	6	C		✓	✓		
0		0	A	B	C	3	4	7	0		✓	✓		✓
2		0	D	A	A	7	0	0	F		✓	✓	✓	✓

El problema de coherencia de cache

✓Definición informal: “Un sistema de memoria es coherente si cualquier lectura de un ítem de dato devuelve el último valor escrito sobre ese ítem”.

✓Esta definición no es precisa y presenta una visión simplificada del problema.

✓Aspectos del comportamiento del sistema de memoria:

-Coherencia: que valores son retornados en una lectura.

-Consistencia: cuando un valor escrito será retornado por una lectura.

Esquema básico para forzar la coherencia

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A stores 1 into X	1	0	1

Sistema de memoria coherente - Propiedades

✓Una lectura de un procesador P a una ubicación X luego de una escritura de P en X, sin otra escritura en X por otro procesador entre la escritura y la lectura por P, siempre retorna el valor escrito por P.

✓Una lectura de un procesador a una ubicación X luego de la escritura de otro procesador en X, retorna el valor escrito si la lectura y la escritura están suficientemente separadas en tiempo, y ninguna otra escritura en X ocurre entre los dos accesos.

✓Un par de escrituras a la misma ubicación se llaman *serializadas,* si son vistas en el mismo orden por todos los procesadores.

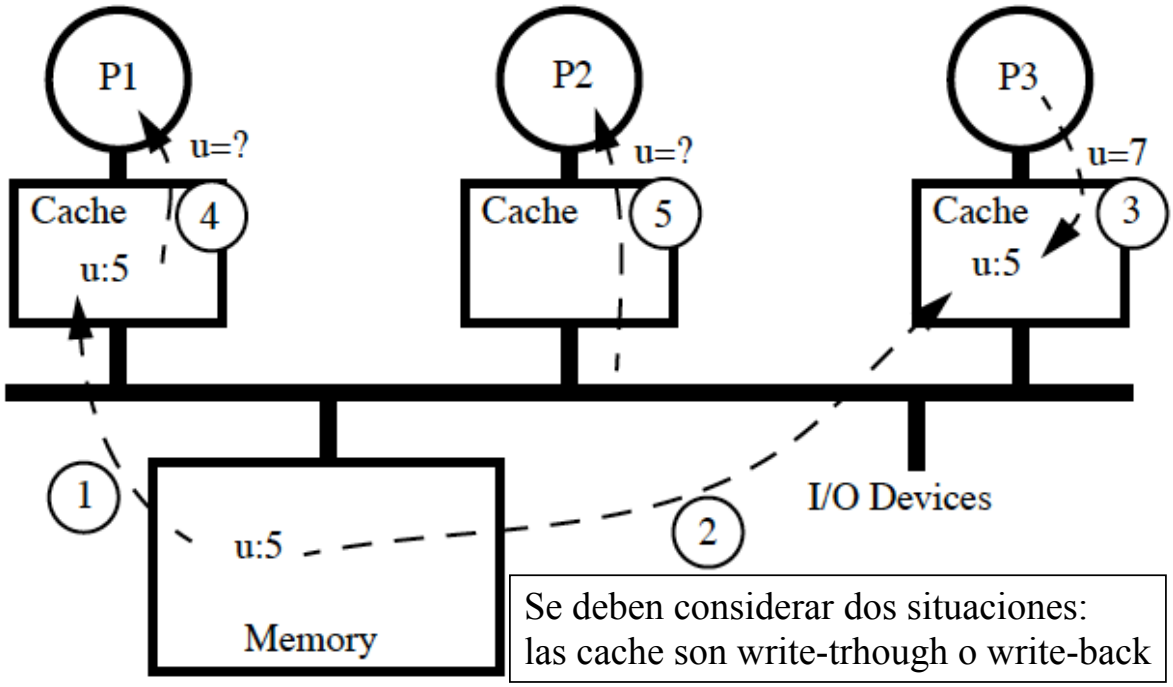
Esquema básico para forzar la coherencia

✓ La cache provee migración y replicación de datos compartidos.

✓Migración: los datos son movidos a la cache local y usados en forma transparente, reduce la latencia de acceso y el bandwidth de la memoria compartida.

✓Replicación: los datos pueden ser simultáneamente compartidos para lectura, reduce la latencia de acceso y la ***contención*** en la lectura de datos compartidos.

Ejemplo del problema de coherencia de cache



Se deben considerar dos situaciones:
las cache son write-trhough o write-back

Protocolos de Snooping

- Acceso exclusivo para escribir. Invalidación por escritura.
- No existen otras copias válidas después de una escritura.
- Ejemplo de invalidación en cache write-back:
  - Una escritura seguida por una lectura de otro procesador.
  - Como las copias son invalidadas, se produce miss.
  - Otras escrituras requieren acceso exclusivo.
  - Escrituras simultáneas: condición de carrera.
  - Fuerza la serialización.

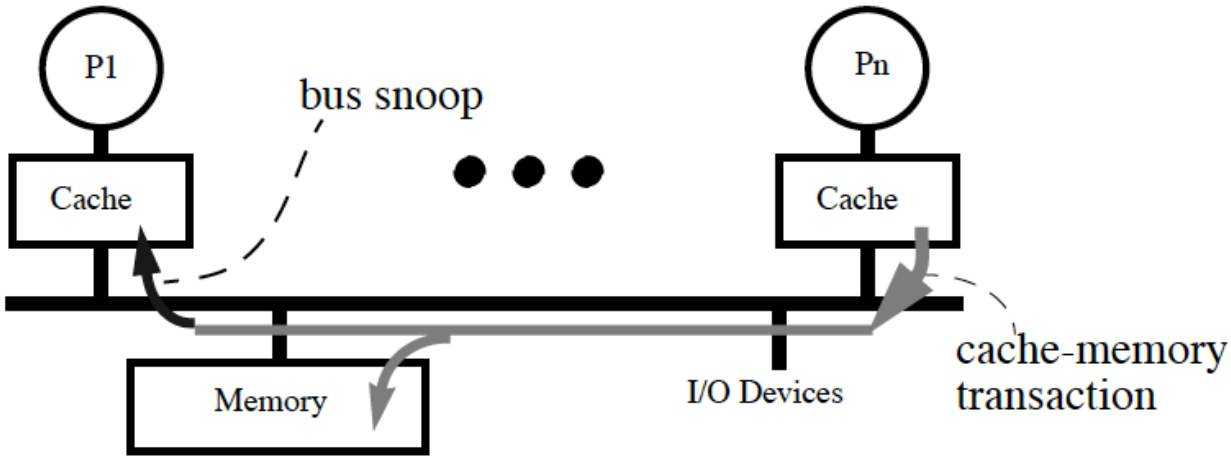
Protocolo de invalidación

Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidation for X	1		0
CPU B reads X	Cache miss for X	1	1	1

El "dueño", en este caso A, responde con el valor y cancela el acceso a memoria.

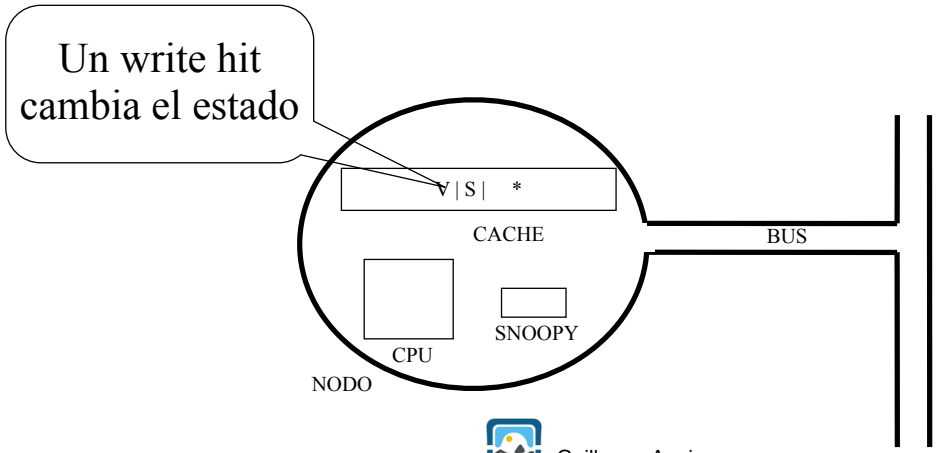
Ejemplo para cache write-back

Coherencia de cache en multi-procesadores mediante Snoopy

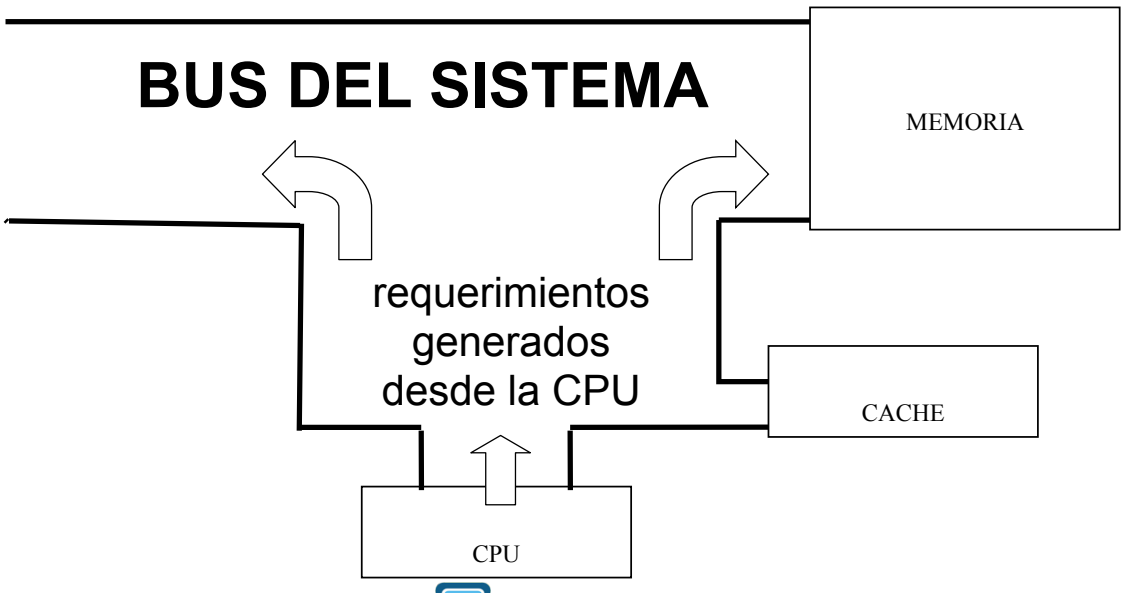


El protocolo Snoopy

- ✓Se controlan actividades desde el procesador y desde el bus.
- ✓Se mantiene el estado de los bloques.
- ✓A través del bus se acceden e invalidan los datos.



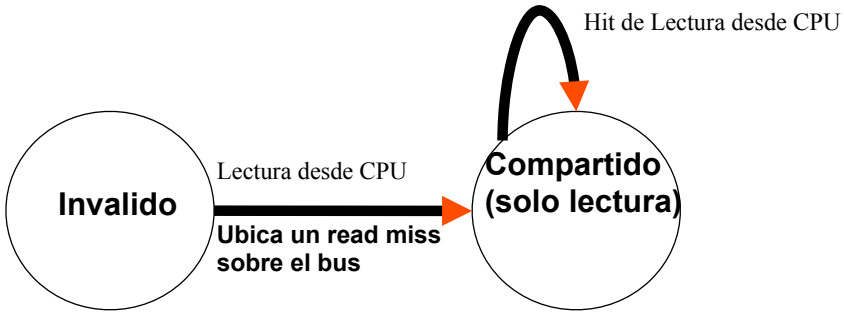
Requerimientos generados desde la CPU



Requerimientos y Acciones

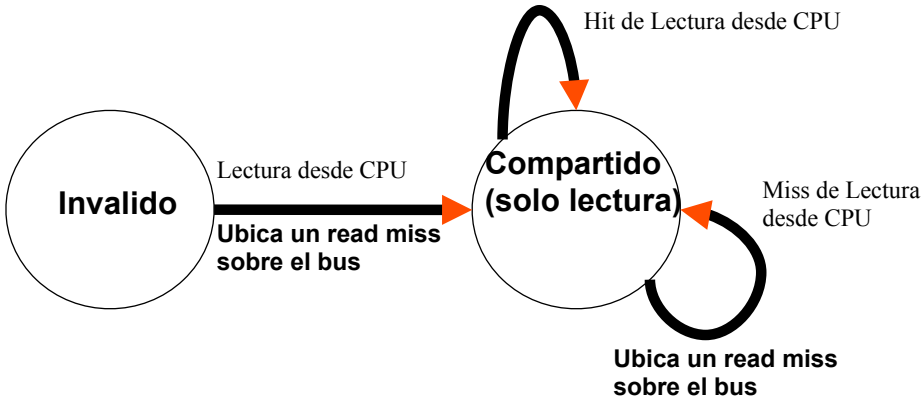
Request	Source	State of addressed cache block	Type of cache action	Function and explanation
Read hit		shared or modified	normal hit	Read data in cache.
Read miss		invalid	normal miss	Place read miss on bus.
Read miss		shared	replacement	Address conflict miss: place read miss on bus.
Read miss		modified	replacement	Address conflict miss: write back block, then place read miss on bus.
Write hit		modified	normal hit	Write data in cache.
Write hit		shared	coherence	Place invalidate on bus. These operations are often called upgrade or *ownership* misses, since they do not fetch the data but only change the state.
Write miss		invalid	normal miss	Place write miss on bus.
Write miss		shared	replacement	Address conflict miss: place write miss on bus.
Write miss		modified	replacement	Address conflict miss: write back block, then place write miss on bus.
Read miss		shared	no action	Allow memory to service read miss.
Read miss		modified	coherence	Attempt to share data: place cache block on bus and change state to shared.
Invalidate		shared	coherence	Attempt to write shared block; invalidate the block.
Write miss		shared	coherence	Attempt to write block that is shared; invalidate the cache block.
Write miss		modified	coherence	Attempt to write block that is exclusive elsewhere: write back the cache block and make its state invalid.

Transiciones de estado en base a los requerimientos desde la CPU



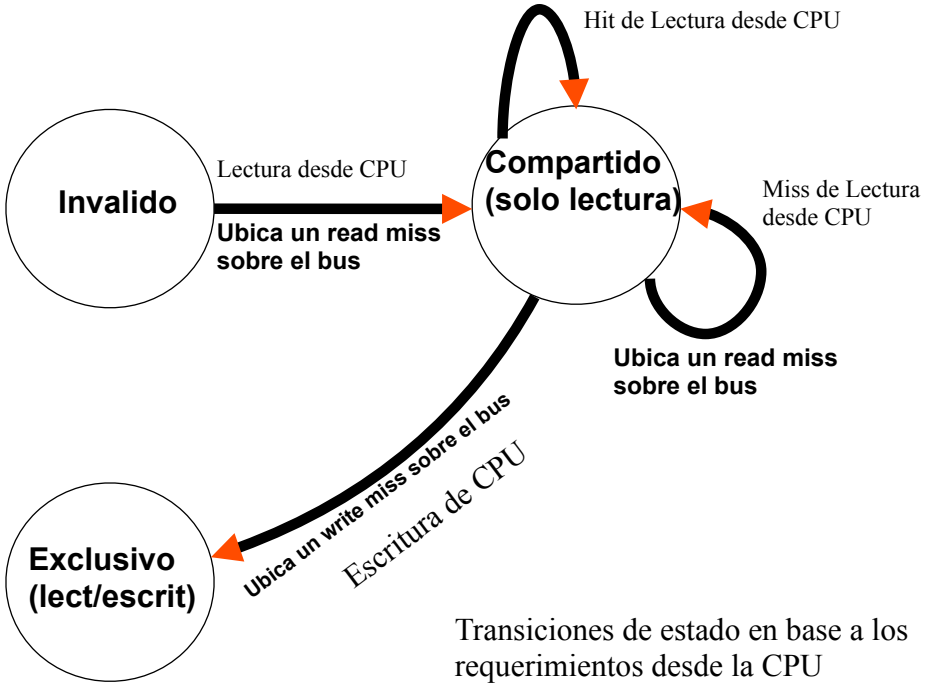
Exclusivo (lect/escrit)

Transiciones de estado en base a los requerimientos desde la CPU

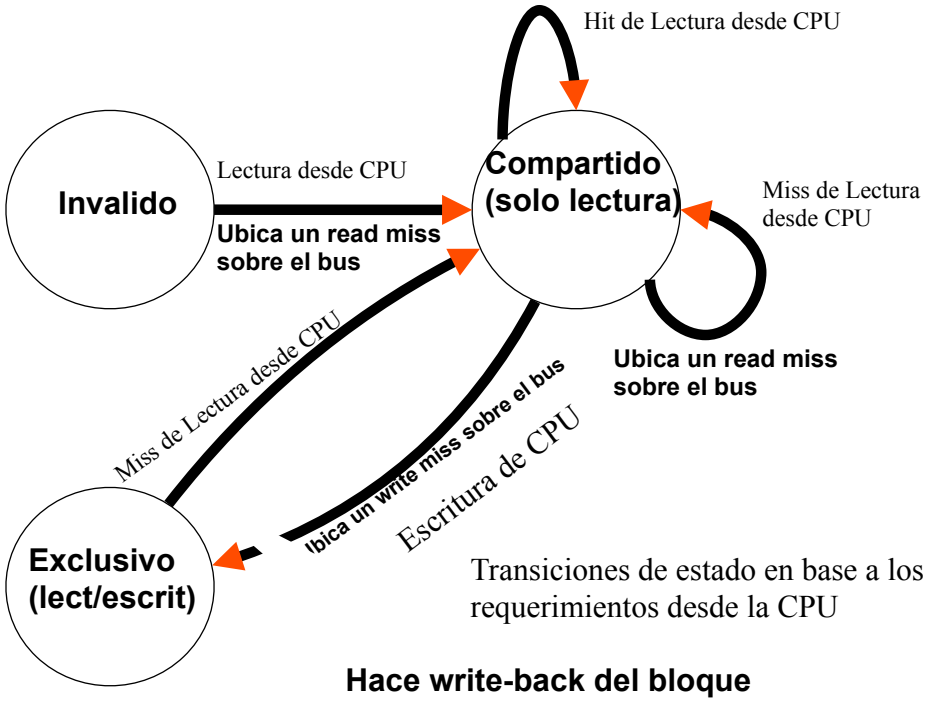


Exclusivo (lect/escrit)

Transiciones de estado en base a los requerimientos desde la CPU

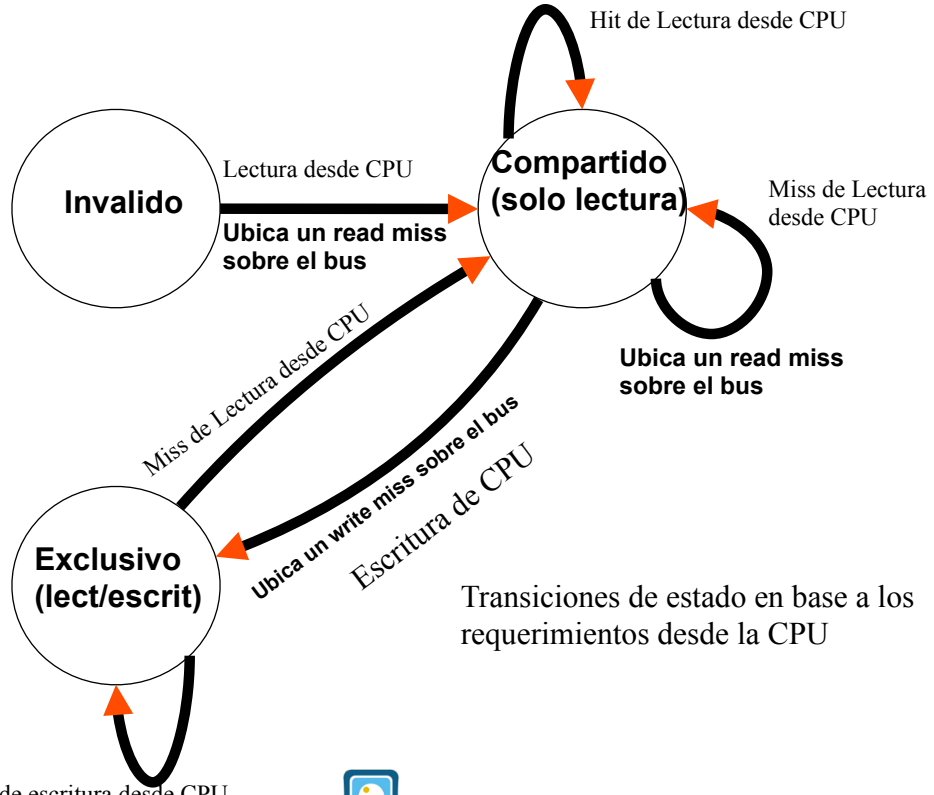


Transiciones de estado en base a los requerimientos desde la CPU

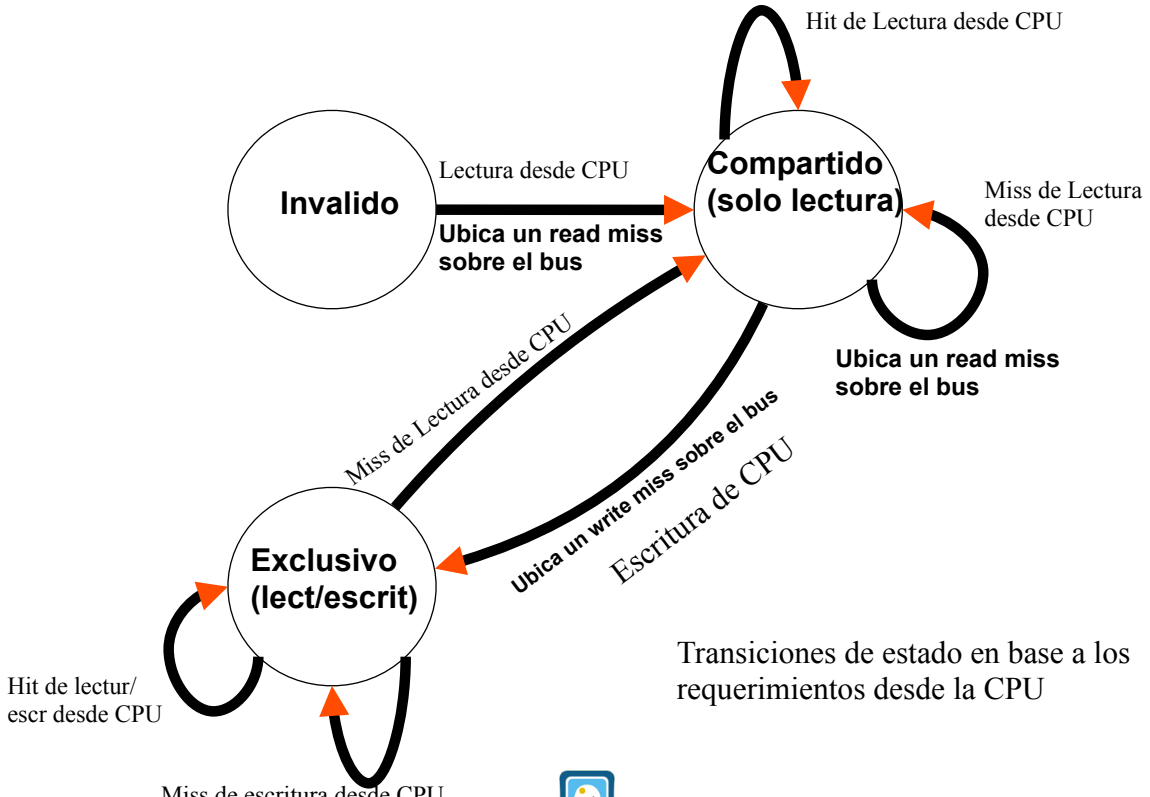


Transiciones de estado en base a los requerimientos desde la CPU

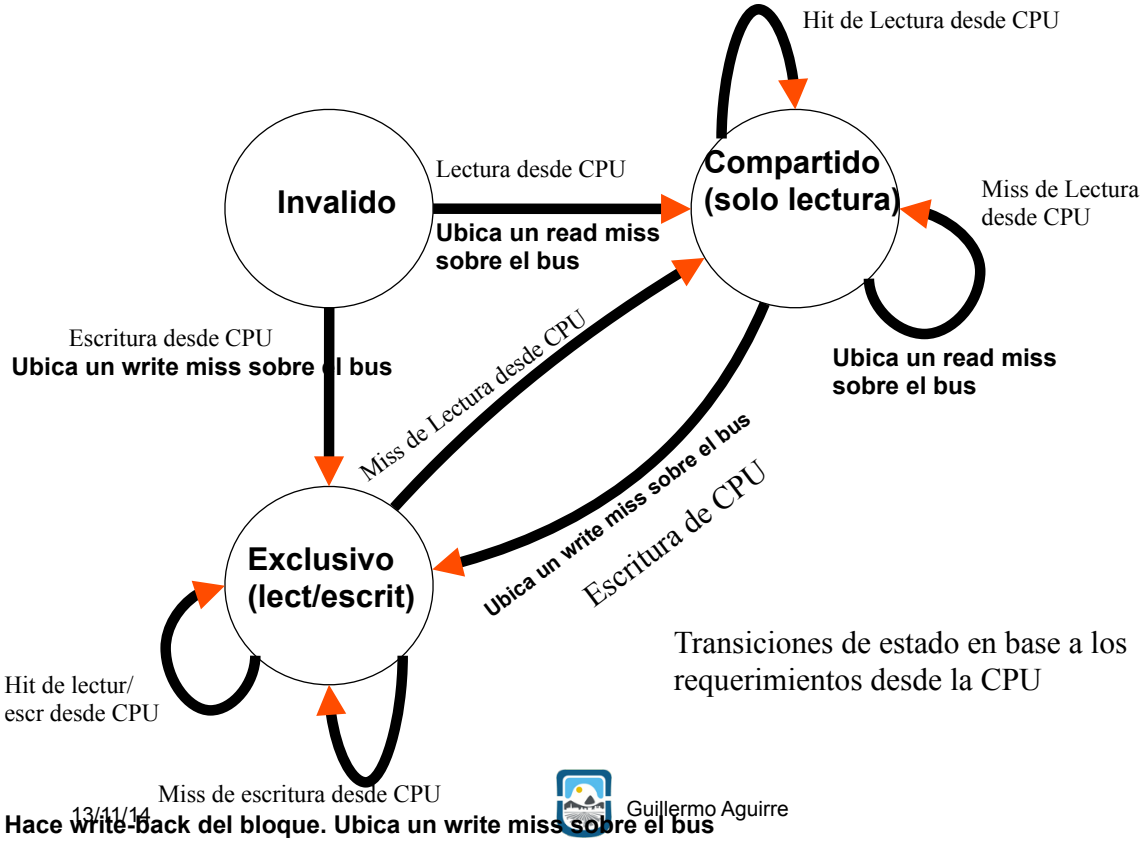
Hace write-back del bloque
Ubica un read miss en el bus



Transiciones de estado en base a los requerimientos desde la CPU

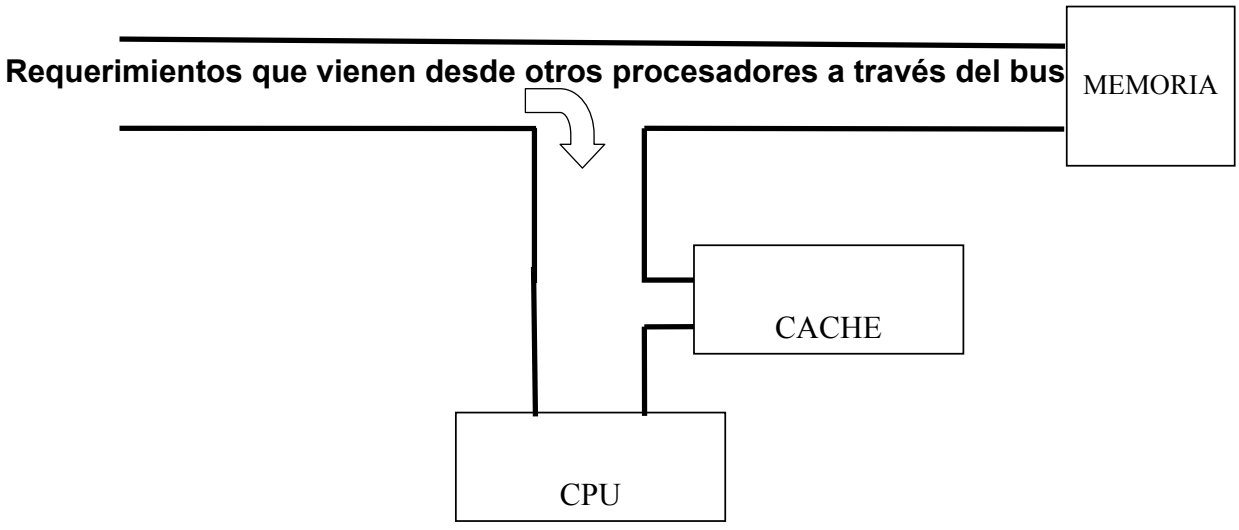


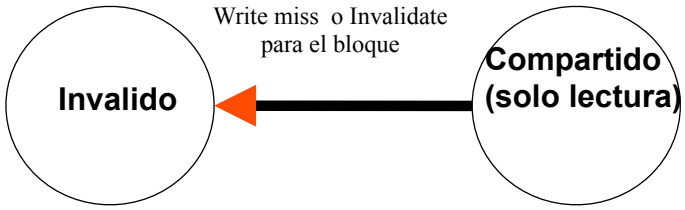
Transiciones de estado en base a los requerimientos desde la CPU



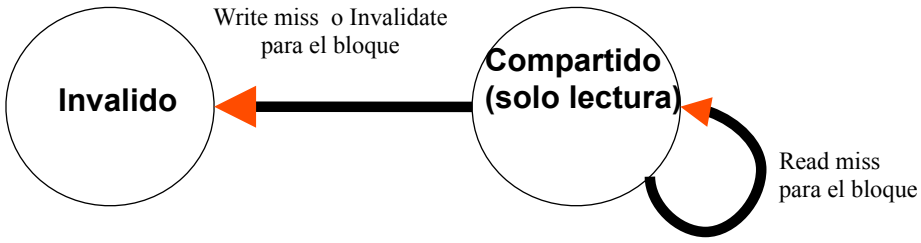
Transiciones de estado en base a los requerimientos desde la CPU

Estados para los requerimientos generados desde el bus

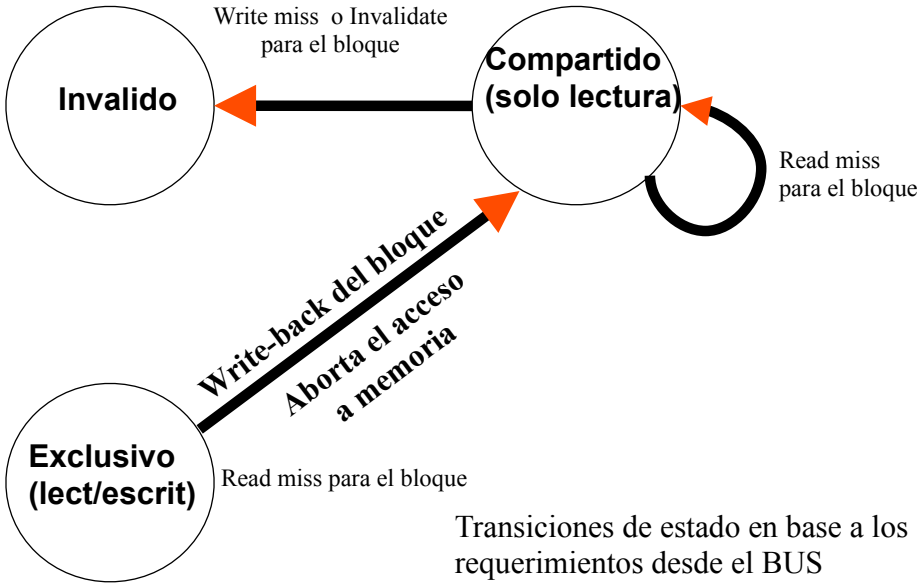




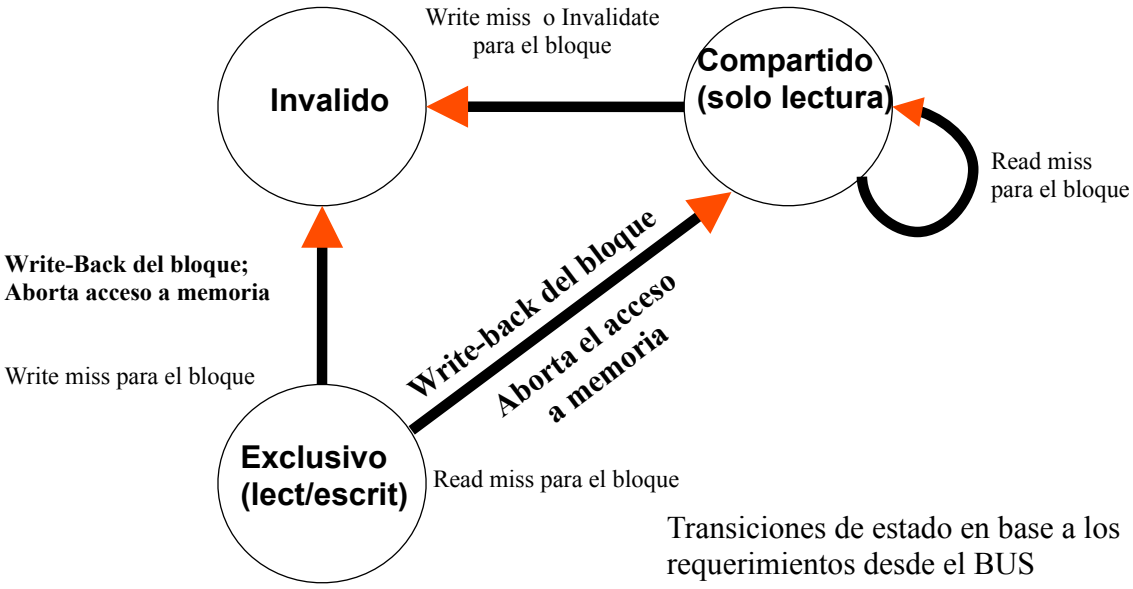
Transiciones de estado en base a los requerimientos desde el BUS



Transiciones de estado en base a los requerimientos desde el BUS



Transiciones de estado en base a los requerimientos desde el BUS

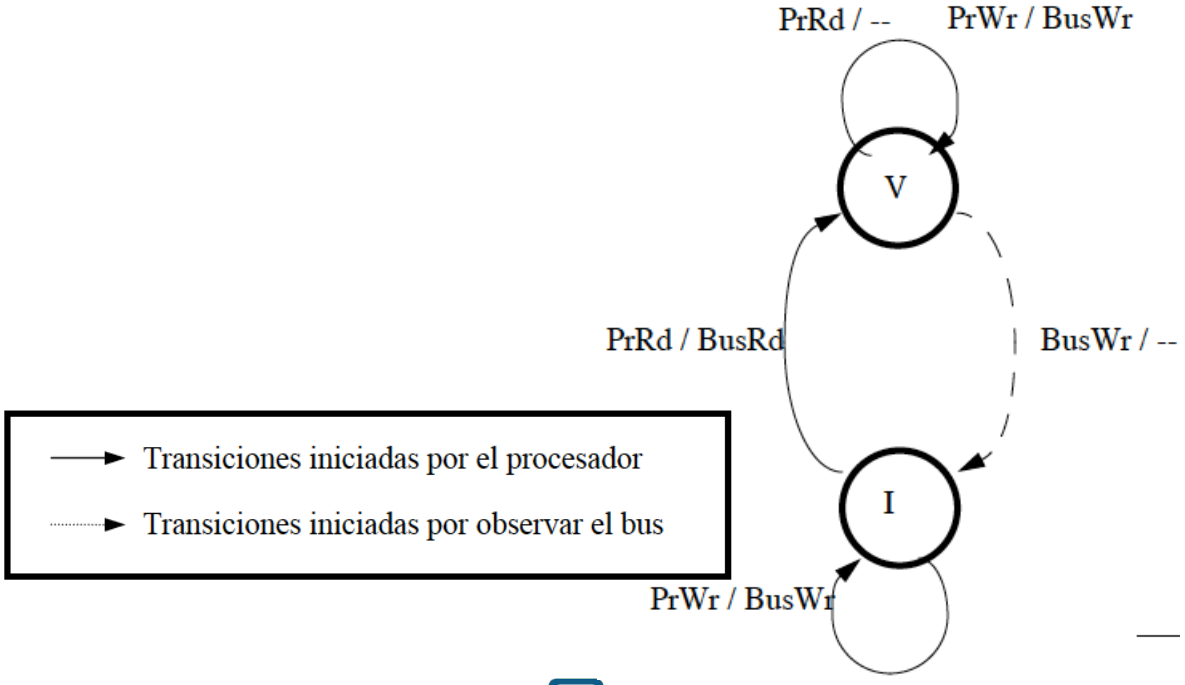


Transiciones de estado en base a los requerimientos desde el BUS

Variantes del protocolo Snoopy

- La red de interconexión común es el **bus**, puede ser crosbar
- Si la cache es write-through, sólo se necesitan dos estados.
- En algunos casos se hace **difusión** del bloque actualizado, pero lo habitual es hacer **invalidación** de las copias.
- Para caches write-back existen tres protocolos:
  - MSI: de tres estados.
  - MESI: agrega un estado exclusivo.
  - Dragon: 4 estados, no tiene I porque es de actualización

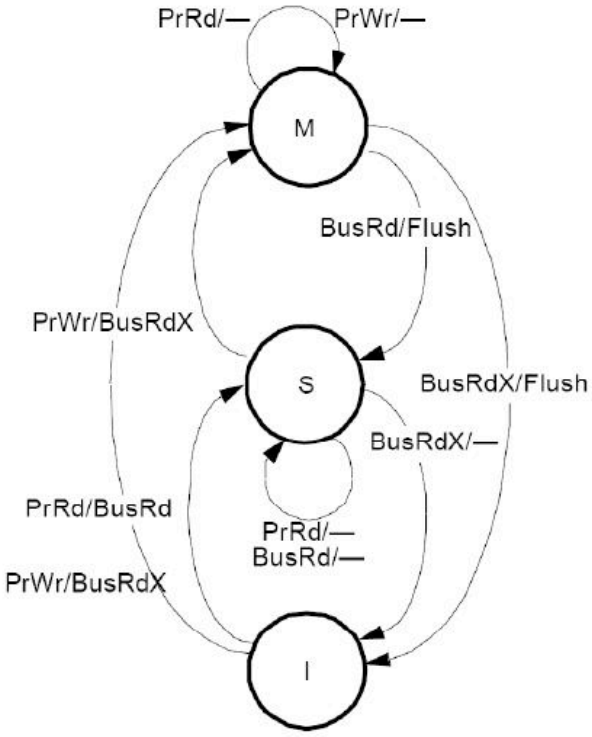
Protocolo de invalidación para una caché coherente de escritura directa y sin alojamiento-en-escritura



Protocolo MSI

Acciones del controlador:
*PrRd
*PrWr

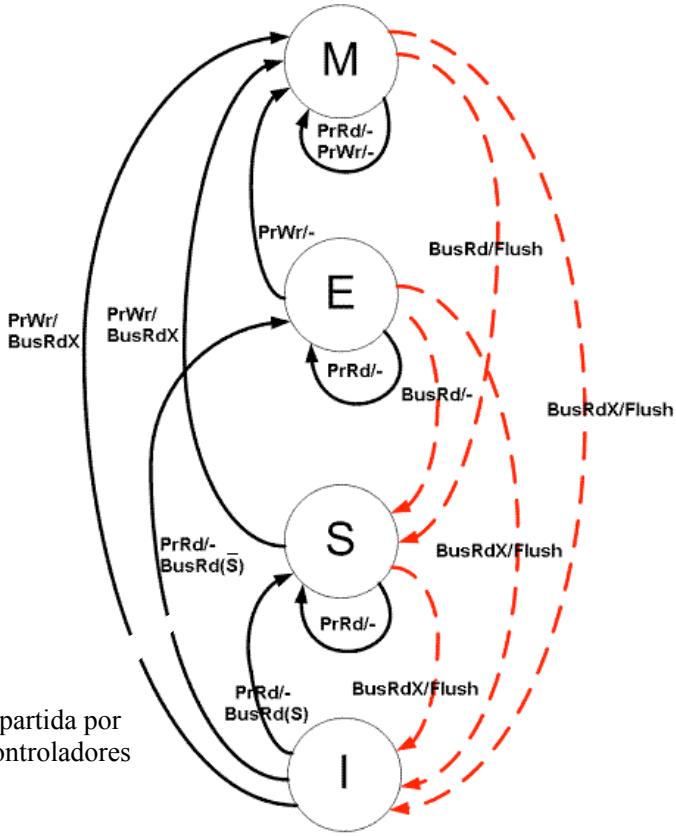
Acciones del bus:
*BusRd
*BusRdX
*Flush



Protocolo MESI

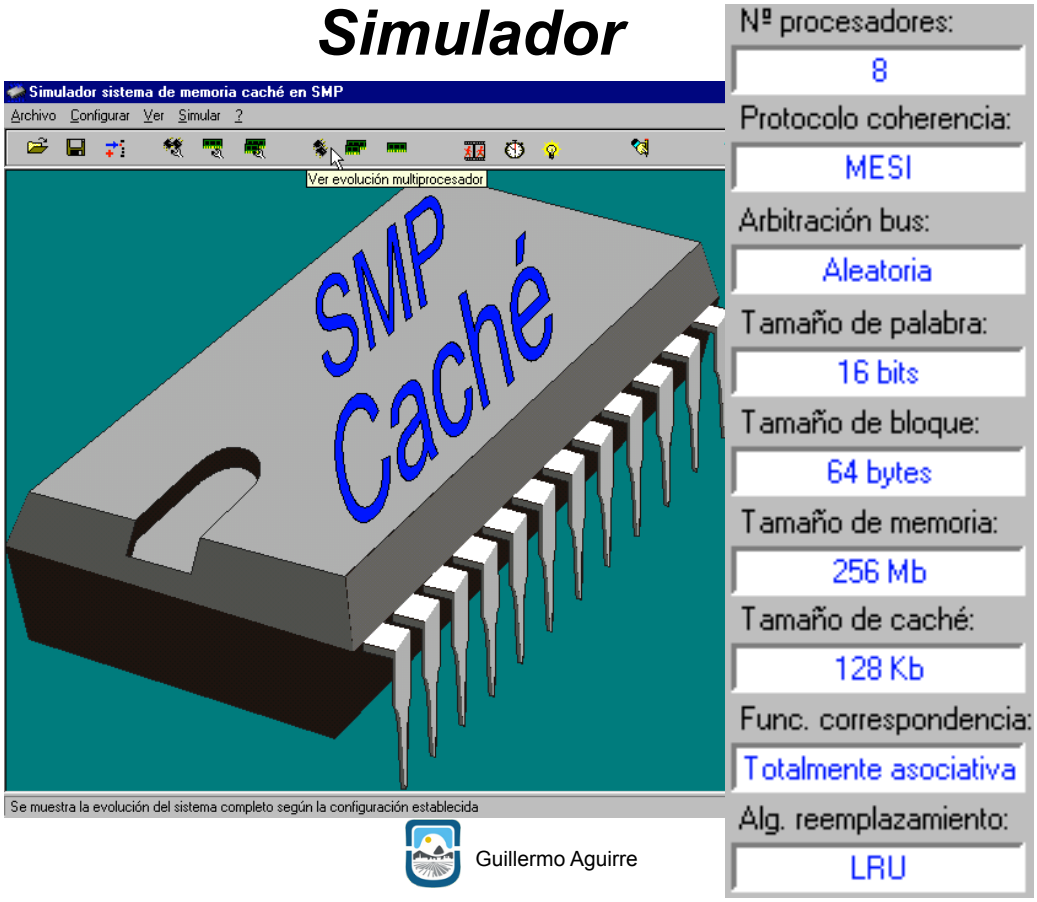
Acciones del controlador:
*PrRd
*PrWr

Acciones del bus:
*BusRd [BusRd(S/$\bar{S}$)]
*BusRdX
*Flush

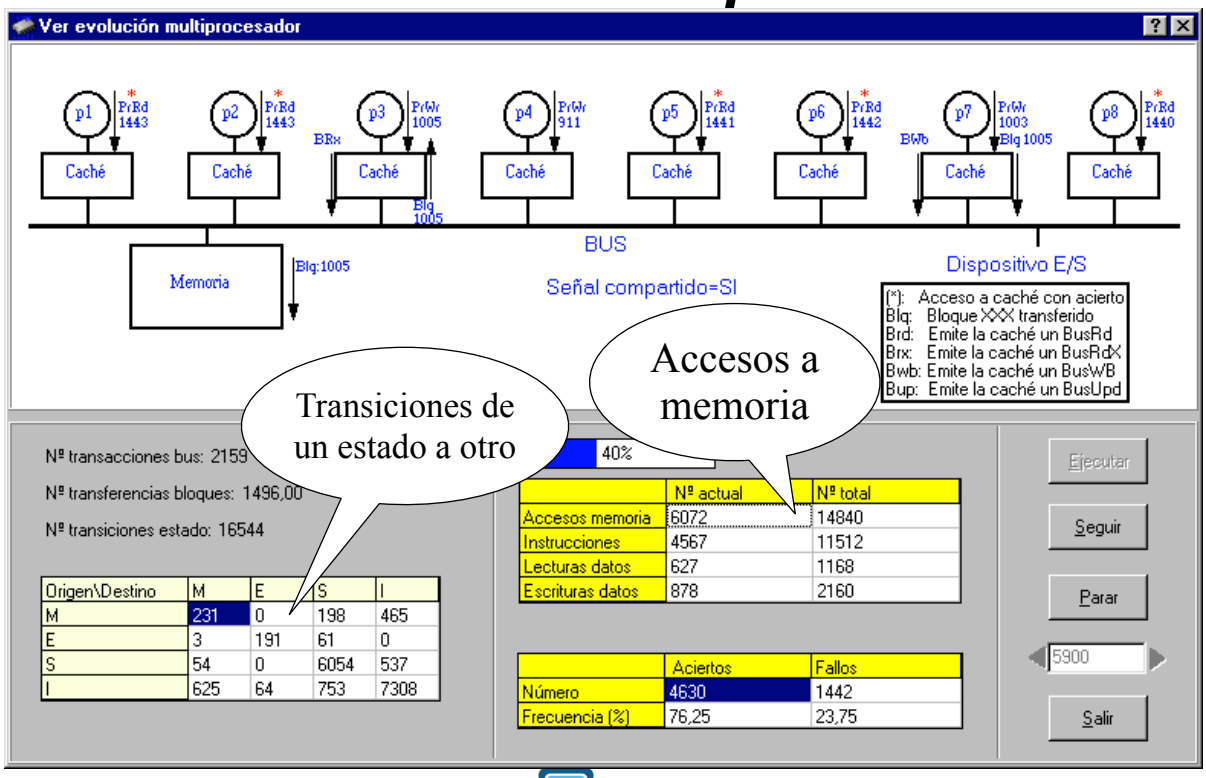


Señal compartida por todos los controladores

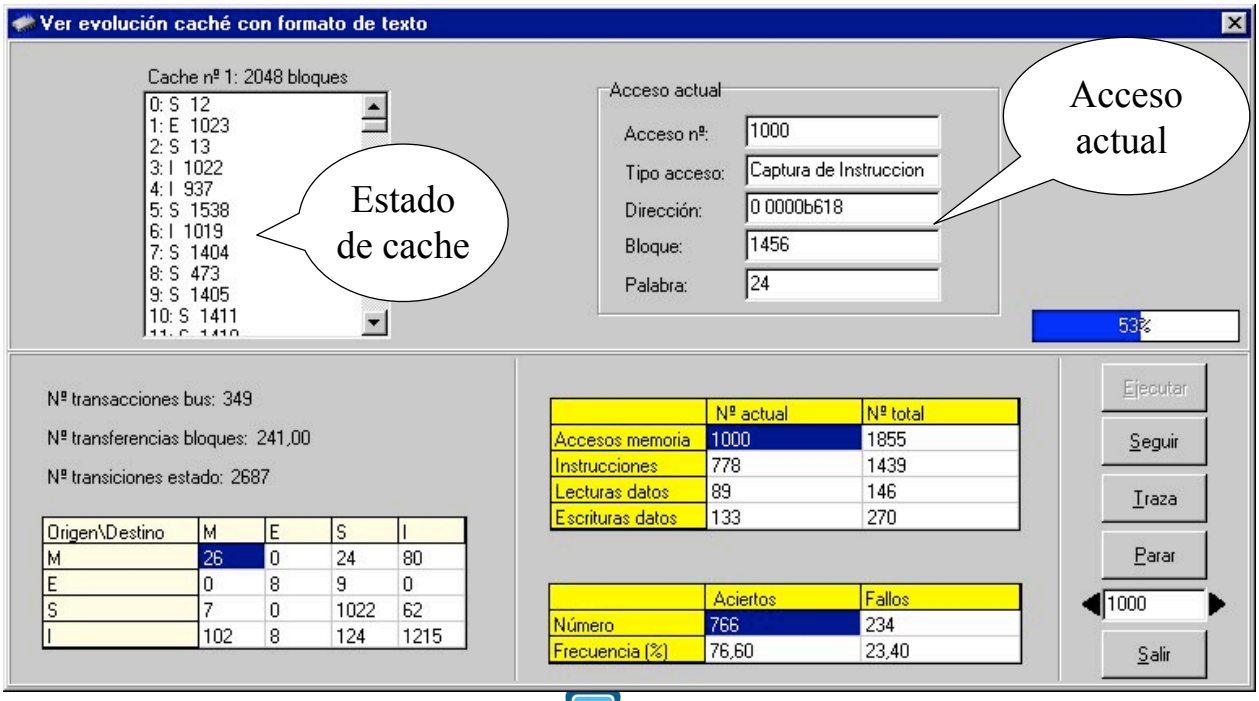
Simulador



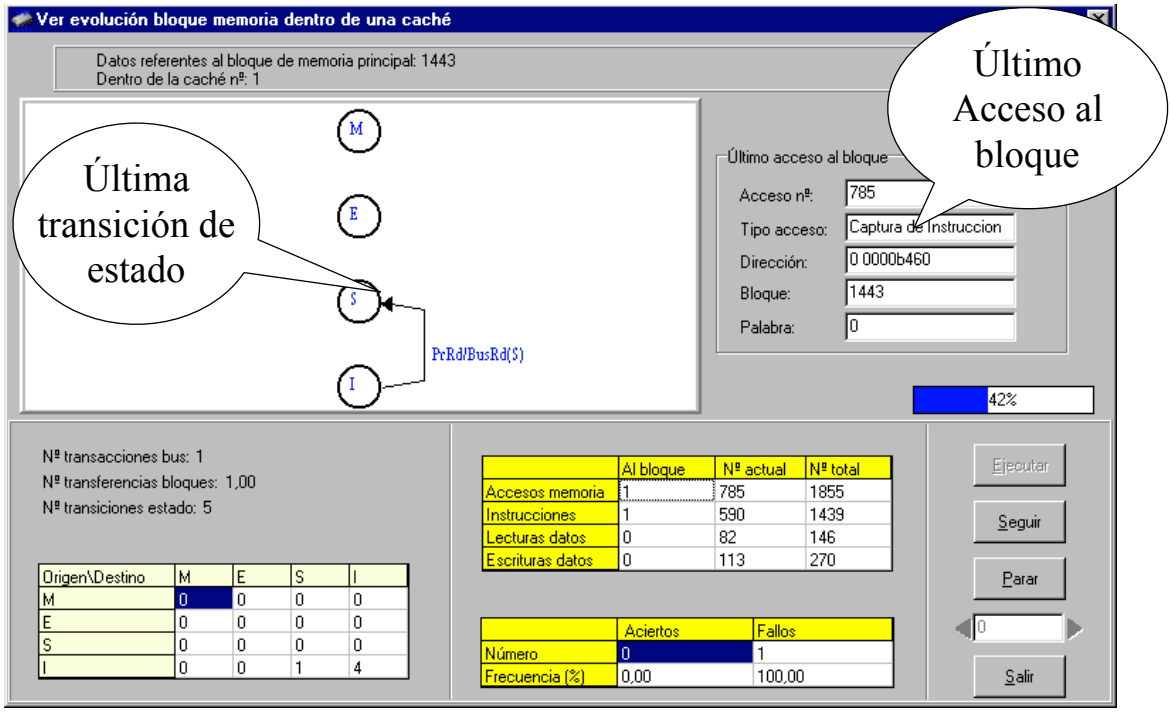
Evolución del multiprocesador



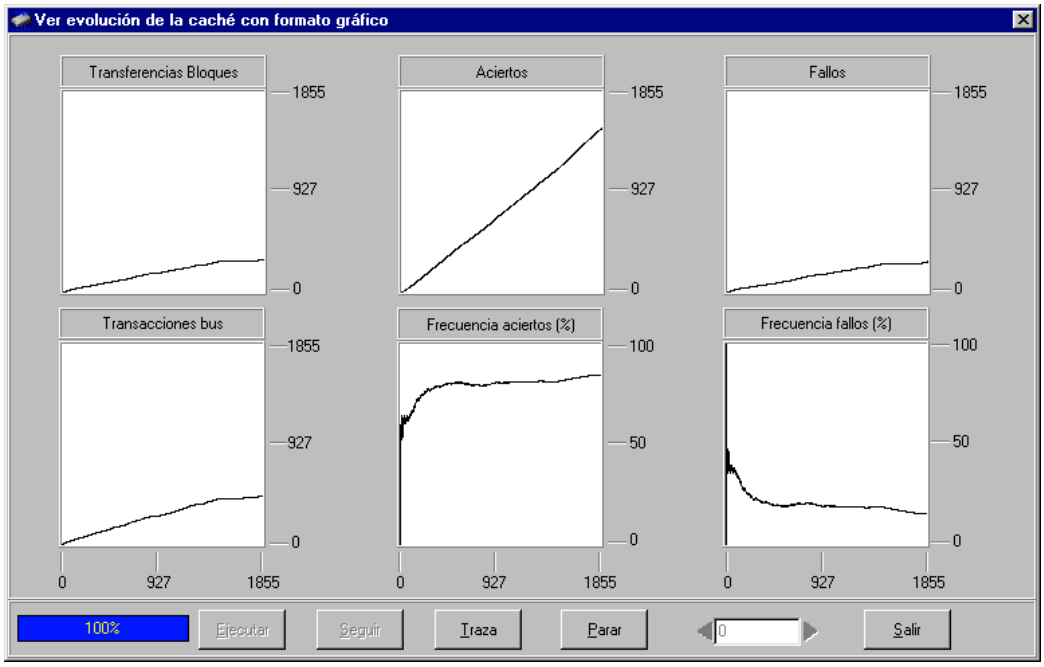
Datos en formato texto de una cache



Evolución de un bloque



Gráficas de la evolución de la cache



¿Qué vimos?

- Memoria Virtual
- Cuatro preguntas a la jerarquía
- Las tres C´s
- Coherencia de cache
- Comportamiento del controlador de Cache
- Esquemas para mantener la coherencia de cache
- Protocolo Snoopy
- Protocolo MESI
- Simulador SMP Cache