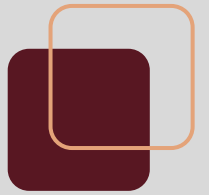
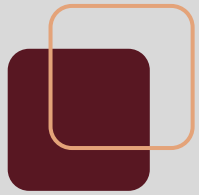


Aspectos avanzados de arquitectura de computadoras Jerarquía de Memoria II

Facultad de Ingeniería - Universidad de la República

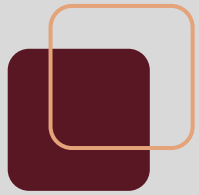
Curso 2017

Técnicas Básicas (1/5)
Mayor Tamaño de Caché



- Mejora obvia: Aumentar el tamaño de la caché.
- Ventaja:
  - Menor Miss Rate
- Desventajas:
  - Costo (+)
  - Consumo (+)

Técnicas Básicas (2/5)
Mayor Tamaño de Bloque



- Bloques más grandes provocan que se traigan a caché mayor cantidad de datos por miss.
- Ventajas:
  - Menor Miss Rate (si no se abusa)
  - Mayor aprovechamiento de las transferencias en ráfaga de memoria y buses.

Técnicas Básicas (3/5)
Mayor Tamaño de Bloque

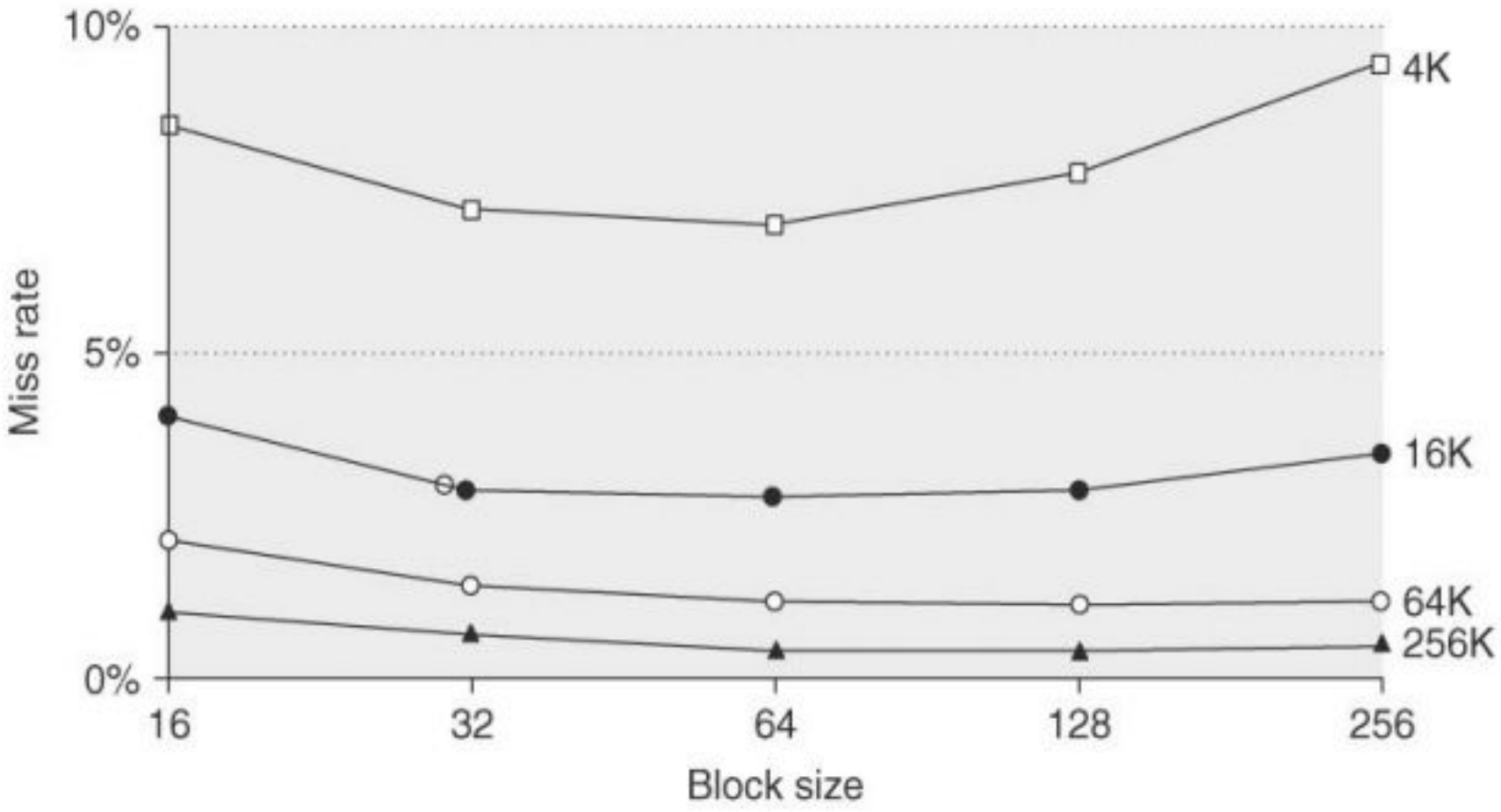
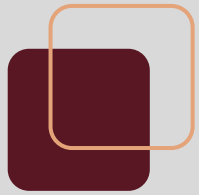


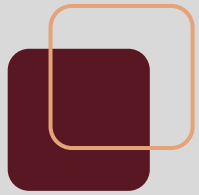
Imagen del libro Computer Architecture: A Quantitative Approach (5ta edición). John L. Hennessy y David Patterson

Técnicas Básicas (4/5)
Mayor Asociatividad



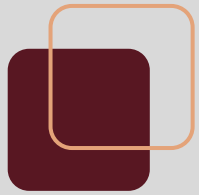
- Mayor asociatividad provoca menos conflictos por reemplazos (no reduce los misses por capacidad)
- Ventajas:
  - Miss Rate (-)
- Desventajas:
  - Hit Time (+)
- En general se busca aquella que minimiza AMAT.

Técnicas Básicas (5/5)
Diseño de Múltiples Niveles



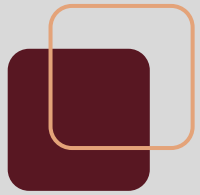
- Mayor cantidad de niveles de caché permite hacer mejor uso de los principios sobres los que se basa la jerarquía de memoria
- Técnicas:
  - L1 chica y rápida -> menor hit time.
  - Write through entre L1 y L2
  - L1 de instrucciones, L1 de datos y L2 unificada.

Técnicas Avanzadas
Cache Pipelining (1/3)

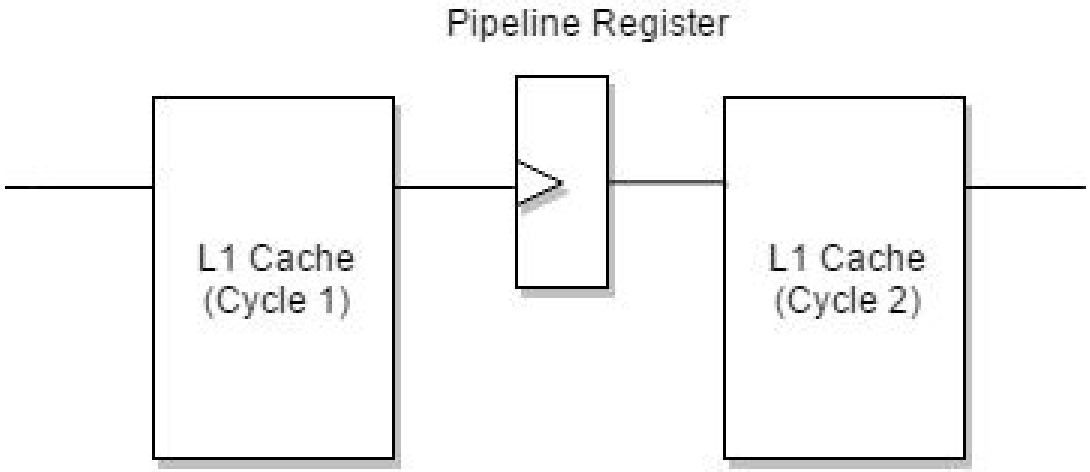


- Típicamente:
  - cache access time > ciclo de reloj de CPU.
- Una solución para acompasar los relojes es realizar *cache pipelining*.
- Ej:
  - Ciclo 1: Comparación de Tags
  - Ciclo 2: Acceso a bloque

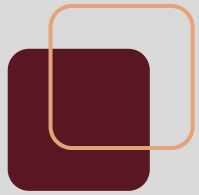
Técnicas Avanzadas
Cache Pipelining (2/3)



- A ojos del CPU, un caché L1 con estas características se puede representar como una unidad funcional de dos etapas:



Técnicas Avanzadas
Cache Pipelining (3/3)



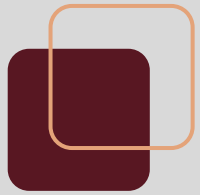
- Ejemplo:

```
ld      R1, 0(R2)
ld      R3, 0(R6)
add R7, R1, R3
```

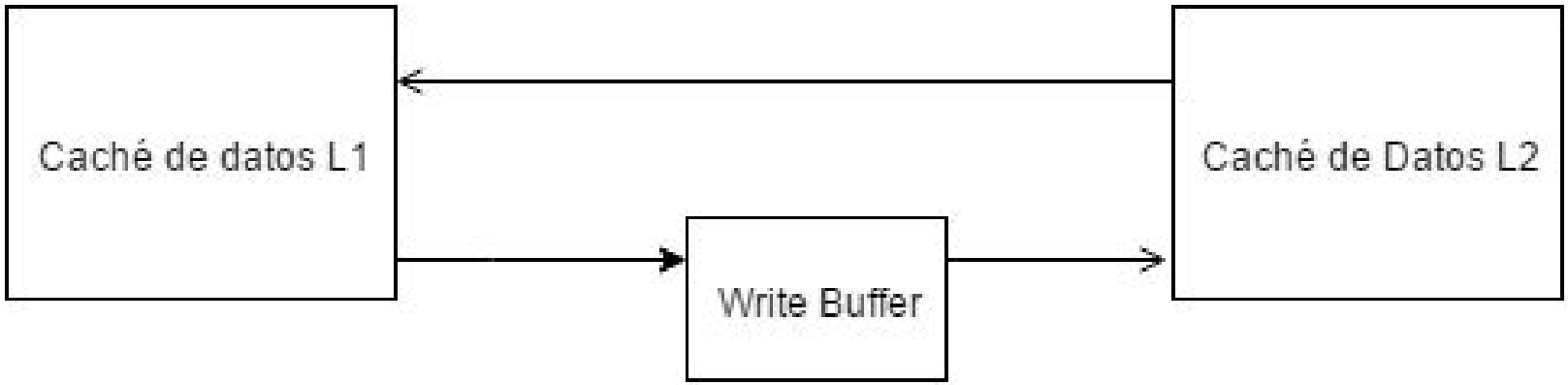
	1	2	3	4	5	6	7	8
ld R1, 0(R2)	IF	ID	EX	MEM1	MEM2	WB		
ld R3, 0(R6)		IF	ID	EX	MEM1	MEM2	WB	
add R7, R1, R3			IF	ID	ID	ID	EX	WB

- Se asume MEM1-MEM2 como una unidad funcional más.
- Forwarding entre etapas en azul!

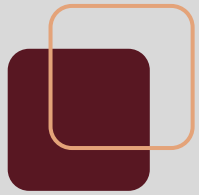
Técnicas Avanzadas
Write Buffers (1/3)



- Buffers de escritura encargados de realizar las escrituras desde cachés de menor jerarquía hasta el siguiente.
- Liberan al caché para continuar procesando el acceso

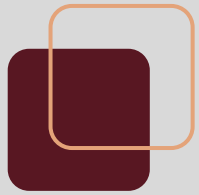


Técnicas Avanzadas
Write Buffers (2/3)



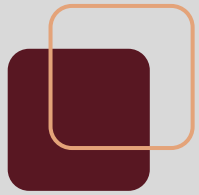
- Problema: el write buffer puede tener el valor más actualizado de una línea en el sistema.
  - Se debe buscar en el write buffer al tener un read miss.
- Mejora miss penalty en write-back y write hit time en write-through.

Técnicas Avanzadas
Write Buffers (3/3)



- En un caché con write-through, es posible tener varias escrituras seguidas al mismo bloque.
- Esto puede mejorarse escribiendo accesos consecutivos directamente en el write buffer (*write merging*)

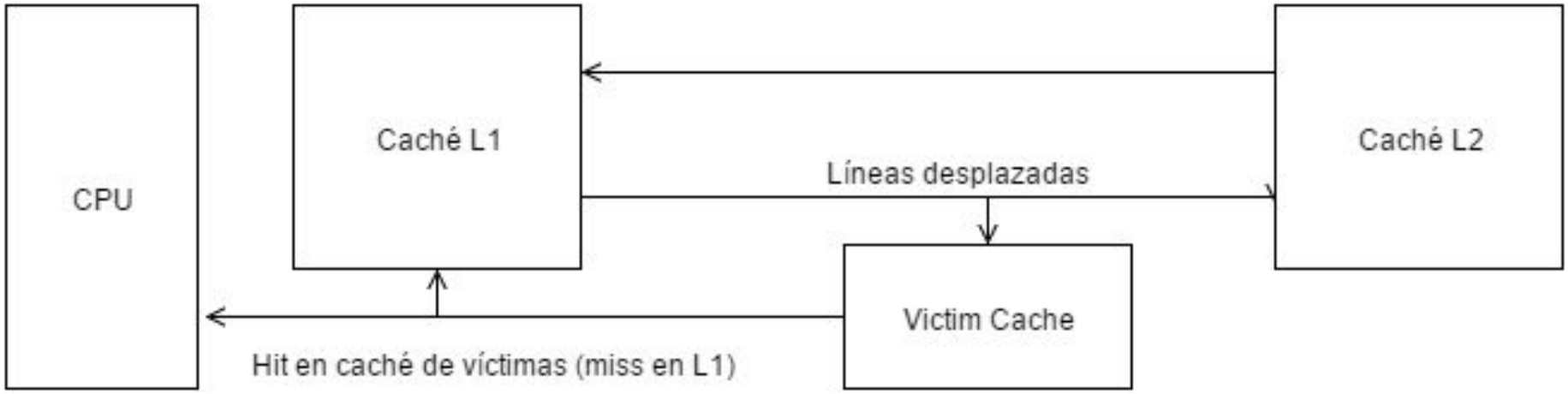
Técnicas Avanzadas
Victim Cache (1/5)



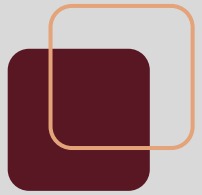
- Una caché de víctimas es una pequeña memoria caché totalmente asociativa, colocada entre dos niveles de jerarquía, que mantiene las últimas líneas desplazadas del caché.
- Al obtener un miss en el caché, se busca la dirección en la caché de víctimas.

Técnicas Avanzadas
Victim Cache (2/5)

- El efecto neto de una caché de víctimas es aumentar temporalmente la asociatividad de *todos* los conjuntos a un costo bastante bajo.
- Miss penalty (-)



Técnicas Avanzadas
Victim Cache (3/5)



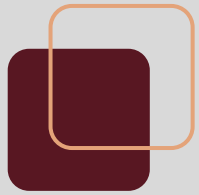
- Ejemplo: Caché L1 de 128 KB con direcciones de 32 bits. Líneas de 256 bytes en conjuntos de 2 vías. Caché de víctimas de 4 bloques. Reemplazo LRU
- Dirección para Caché L1:

Tag (16 bits)	Conjunto (8 bits)	Byte (8 bits)

- Dirección para Caché de Víctimas:

Tag (24 bits)	Byte (8 bits)

Técnicas Avanzadas
Victim Cache (4/5)



Caché L1

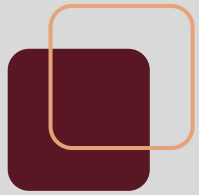
Conjunto	Timestamp	Tag	Contenido
1	20000	0x1000	101010….
1	10000	0x1410	00011…..
2	350000	0x3000	11001…..
2	150000	0x1520	11001…..
...			

Victim Cache

Tag	Contenido
(Vacío)	
(Vacío)	
(Vacío)	
(Vacío)	

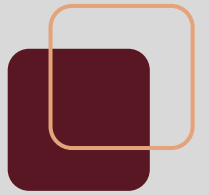
- Se acceden a las direcciones: 0x15000134, 0x20000256, 0x14100152, 0x15200252.

Técnicas Avanzadas
Victim Cache (5/5)



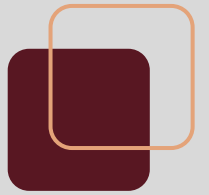
- 0x15000134 - MISS, se trae bloque 0x15000100 al conjunto 1 y desplaza el bloque 0x14100100 al caché de víctimas.
- 0x20000256 - MISS, se trae bloque 0x20000200 al conjunto 2 y desplaza el bloque 0x15200200 al caché de víctimas
- 0x14100152 - MISS en L1, hit en caché de víctimas
- 0x15200252 - MISS en L1, hit en caché de víctimas.

Técnicas Avanzadas
Early Restart



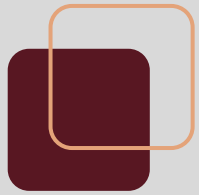
- Durante un miss en caché, se debe traer todo el bloque correspondiente, tarea que usualmente demora varios ciclos de reloj y en la técnica más simple se espera a recibir todo el bloque antes de resumir el acceso.
- La técnica *early restart* consiste en retomar la actividad en el caché al obtener la palabra que ocasionó el miss.

Técnicas Avanzadas
Critical Word First



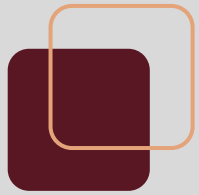
- Consiste en cambiar el orden en que se devuelve el bloque solicitado en caso de un miss, devolviendo primero la palabra que ocasionó el miss, y devolviéndola al CPU inmediatamente.
- CWF y ER disminuyen el miss penalty.
- CWF es bastante más compleja de implementar que ER.

Técnicas Avanzadas
Non-Blocking Caches (1/2)



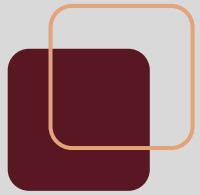
- Para procesadores *out of order*, la detención de una instrucción no significa que se debe detener la ejecución.
- Una caché no bloqueante permite seguir recibiendo accesos durante un *miss*, es decir, mientras se está esperando que el bloque correspondiente llegue desde memoria.

Técnicas Avanzadas
Non-Blocking Caches (2/2)



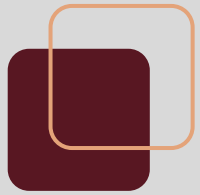
- Las cachés no bloqueantes logran una reducción del *miss penalty*, al ocultar el tiempo en que se busca el bloque desde los niveles bajos de la jerarquía.
- Además, logran un mayor *ancho de banda*.
- Los misses por los cuales se está esperando resultado se guardan en una estructura llamada mshr (Miss Status Holding Register)

Técnicas Avanzadas
Multibanked Caches

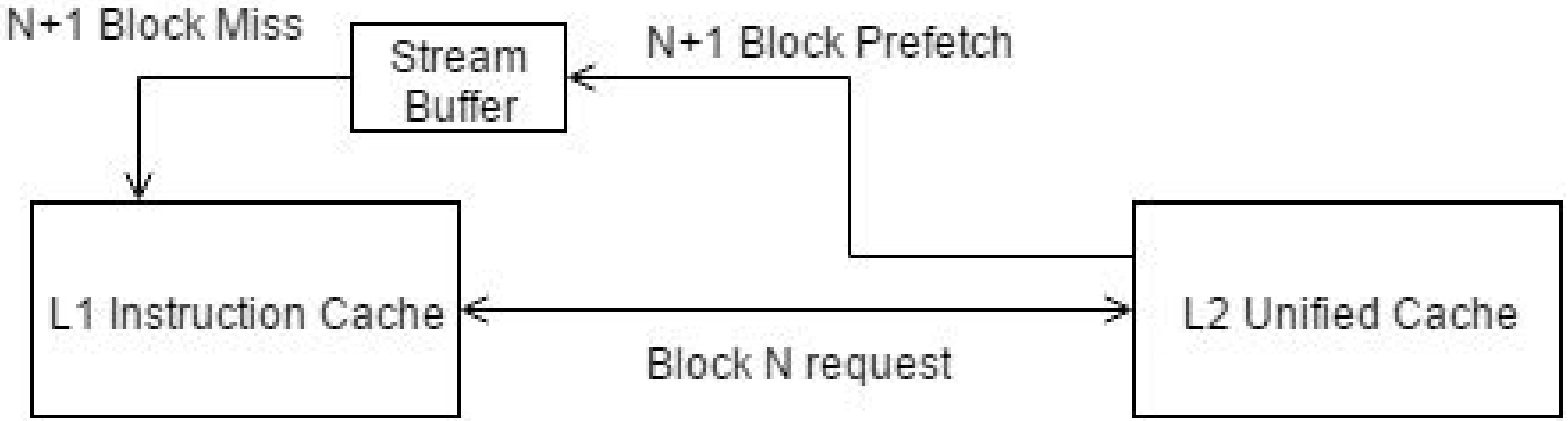


- Similar a la técnica *interleaving* de memoria, los conjuntos de la caché se pueden dividir en bancos, cada cual conteniendo múltiples conjuntos.
- Se permiten múltiples accesos a la caché, siempre que estos sean en *bancos* diferentes.
- Efecto:
  - Ancho de banda (+)

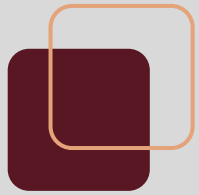
Técnicas Avanzadas
Instruction Prefetching (1/2)



- Siguiendo la idea del *prefetch* realizado por la *fetch unit* del CPU, al leer un bloque de la caché de instrucciones, se puede buscar el siguiente.

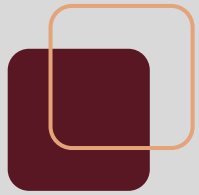


Técnicas Avanzadas
Instruction Prefetching (2/2)



- *Block Lookahead*: Al presentarse un miss en el bloque *N*, buscar dicho bloque y además el *N + 1* en el stream buffer.
- *Strided prefetch*: Si se observan accesos a los bloques b, b + N, b + 2N, realizar prefetch de **b + 3N** en el siguiente miss.

Fin



¿Preguntas?