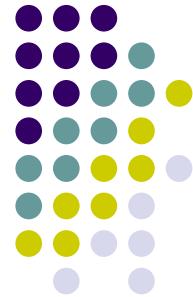


Jerarquía de Memoria

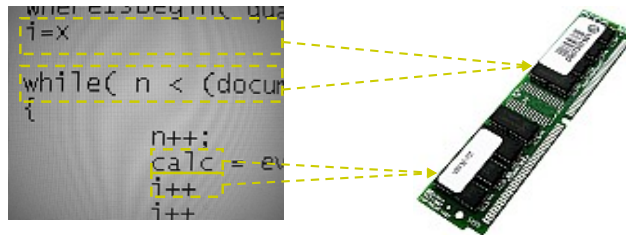
Memoria Cache

Elías Todorovich - Marcelo Tosini
Arquitectura I - Curso 2019



Introducción

Los programas comparten en la memoria tanto su **código** como sus **datos**.



Estrategia de optimización de rendimiento

- posibilitar a la CPU el acceso **ilimitado y rápido** tanto al código como a los datos.

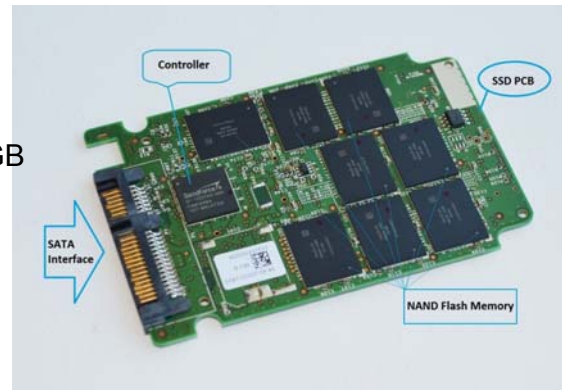
Inconveniente

- **tecnológicamente**, cuanto más rápidas son, más costosas resultan.



Tecnologías de memoria

- Static RAM (SRAM) (valores de 2015)
 - 0.5ns – 2.5ns, USD 500 – USD 1000 per GB
- Dynamic RAM (DRAM)
 - 50ns – 70ns, USD 10 – USD 20 per GB
- Memoria flash
 - 5 μ s – 50 μ s, 0,75 – 1 USD per GB
- Disco magnético
 - 5ms – 20ms, 0.05 – 0,10 USD per GB
- Memoria ideal
 - Tiempo de acceso de la SRAM
 - Capacidad y costo/GB del disco magnético.



Cache - 2016

3

Introducción



Ley de localidad

“Todo programa favorece una parte de su espacio de direcciones en cualquier instante de tiempo.”

2 dimensiones:

Localidad temporal. Si se referencia un elemento tenderá a ser referenciado pronto.

Localidad espacial. Si se referencia un elemento, los elementos cercanos a él tenderán a ser referenciados pronto.

Cache - 2016

4



Introducción

Jerarquía de memoria

Es la reacción natural a la localidad y tecnología

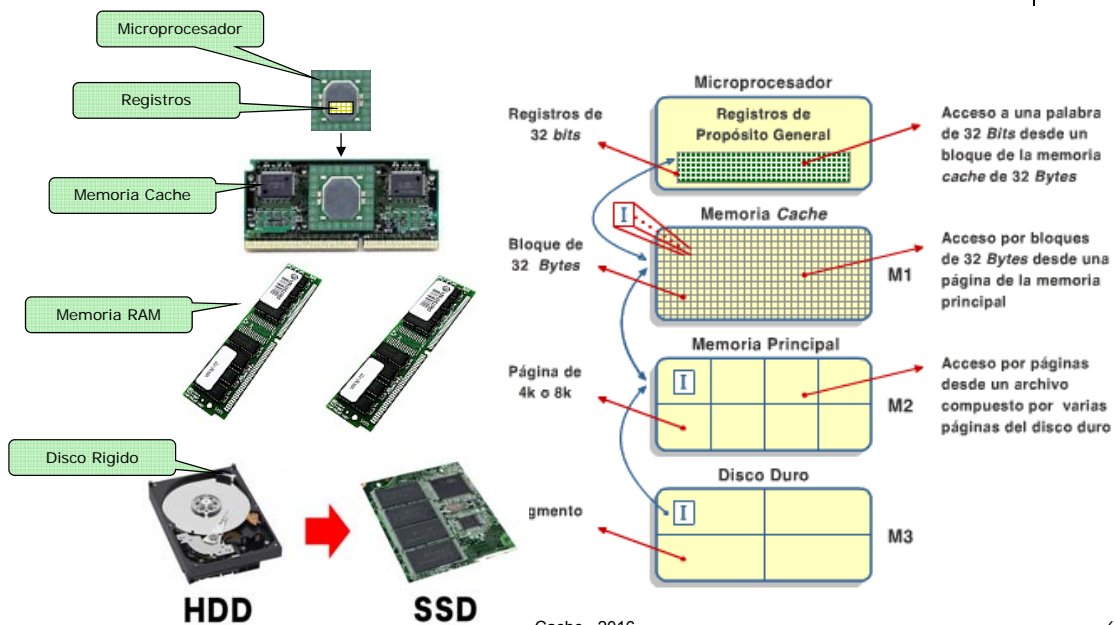
El principio de localidad y la directriz que el hardware más rápido es más caro, mantienen el concepto de una jerarquía basada en diferentes localidades y tamaños.

Organizada en varios niveles -cada uno más pequeño, más caro y más rápido que el anterior.

Todos los datos de un nivel se encuentran también en el nivel siguiente, y todos los datos de ese nivel inferior se encuentran también en el siguiente a él, hasta el extremo inferior de la jerarquía. Cache - 2016

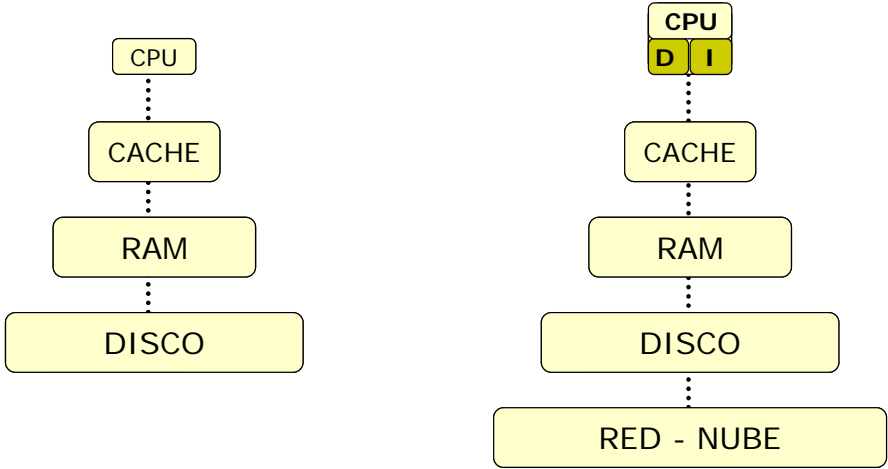
5

Jerarquía en un sistema actual

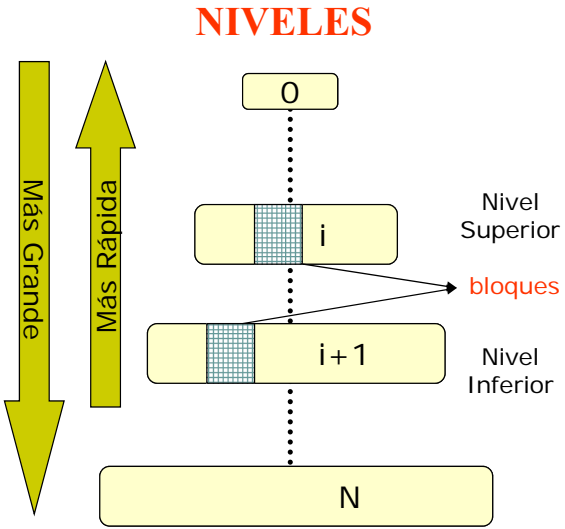


6

Ejemplos de jerarquías de memoria

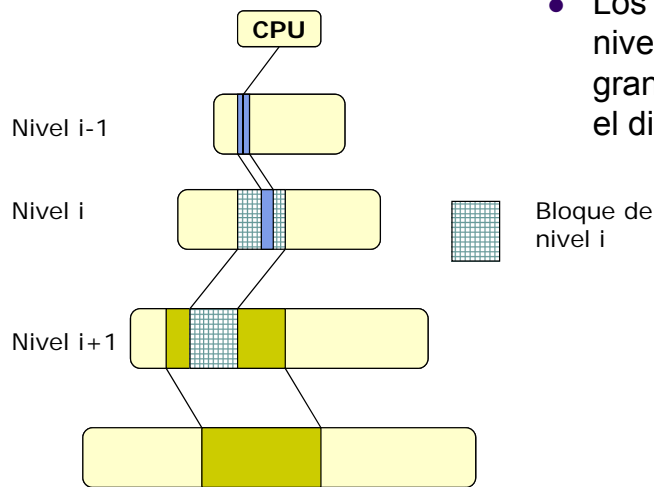


Terminología básica



- BLOQUE**
- Mínima unidad de información en una jerarquía de dos niveles.
 - El nivel superior, el más cercano al procesador, es más rápido y pequeño que el nivel inferior.

Comunicación entre niveles



- Los tamaños dependen del nivel de la jerarquía, la granularidad de los datos, el diseño, etc.

Terminología básica (1)



- ★ **Acierto** (hit) : un acceso a un bloque de memoria que se encuentra en el nivel superior
- ★ **Fallo** (miss) : el bloque no se encuentra en ese nivel
- ★ **Frecuencia de aciertos** : fracción de accesos a memoria encontrados en el nivel superior
- ★ **Frecuencia de fallos** ($1 - \text{frecuencia de aciertos}$) : fracción de accesos a memoria no encontrados en el nivel superior

Terminología básica (2)



- ★ **Tiempo de acierto** : tiempo necesario para acceder a un dato presente en el nivel superior de la jerarquía

incluye el tiempo necesario para saber si el acceso es un acierto o un fallo

- ★ **Tiempo de fallo** : tiempo necesario para sustituir un bloque de nivel superior por el correspondiente bloque de nivel inferior
- ★ **Penalización de fallo** : tiempo de fallo + tiempo de acierto

Tiempo de fallo...



2 componentes:

tiempo de acceso : tiempo necesario para acceder a la primera palabra de un bloque en un fallo

relacionado con la latencia del nivel más bajo

tiempo de transferencia : tiempo para transferir las restantes palabras del bloque

relacionado con el ancho de banda entre las memoria de nivel más bajo y más alto





Rendimiento de la jerarquía

$$T_{\text{medioAcc}} = f_{\text{acierto}} * T_{\text{acierto}} + f_{\text{fallo}} * P_{\text{fallos}}$$

$$= f_{\text{acierto}} * T_{\text{acierto}} + f_{\text{fallo}} * (T_{\text{acierto}} + T_{\text{fallo}})$$

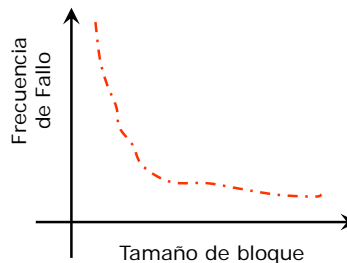
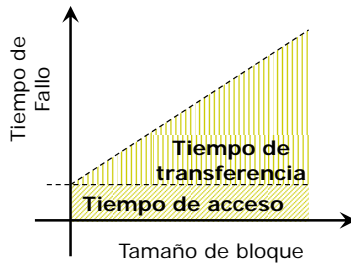
$$= f_{\text{acierto}} * T_{\text{acierto}} + f_{\text{fallo}} * T_{\text{acierto}} + f_{\text{fallo}} * T_{\text{fallo}}$$

$$= f_{\text{acierto}} * T_{\text{acierto}} + (1 - f_{\text{acierto}}) * T_{\text{acierto}} + f_{\text{fallo}} * T_{\text{fallo}}$$

$$= \cancel{f_{\text{acierto}} * T_{\text{acierto}}} + T_{\text{acierto}} - \cancel{f_{\text{acierto}} * T_{\text{acierto}}} + f_{\text{fallo}} * T_{\text{fallo}}$$

$$T_{\text{medioAcc}} = T_{\text{acierto}} + f_{\text{fallo}} * T_{\text{fallo}}$$

f = Frecuencia
 T = Tiempo
 P = Penalización



Cache - 2016

13

Modificaciones de la CPU



Los procesadores diseñados sin jerarquía de memoria son más simples (*los accesos a memoria siempre emplean la misma cantidad de tiempo*)

Los fallos en una jerarquía de memoria implican que la CPU debe manejar tiempos de acceso a memoria variables

- penalización del orden de decenas de ciclos
la CPU espera a que se termine el ciclo de memoria
- penalización del orden de cientos de ciclos
 - Se interrumpe a la CPU para atender otras tareas mientras dura el fallo.
 - *La CPU debe soportar excepciones por accesos a memoria.*

Cache - 2016

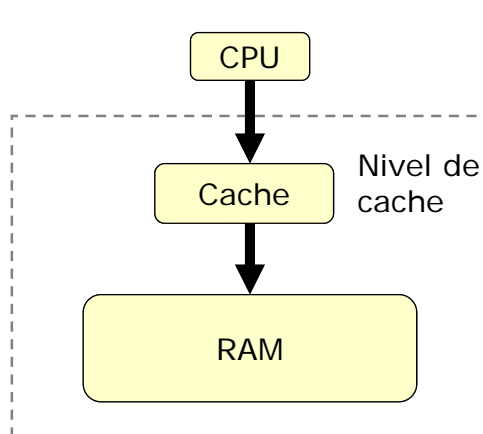
14

Clasificación de las jerarquías de memoria



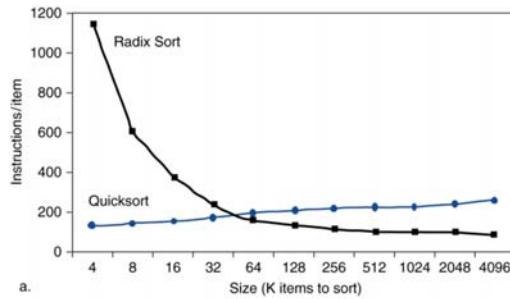
- ★ Ubicación del bloque
Dónde puede ubicarse un bloque en el nivel superior
- ★ Identificación del bloque
Cómo se encuentra un bloque en el nivel superior
- ★ Sustitución de bloque
Qué bloque debe reemplazarse en caso de fallo
- ★ Estrategia de escritura
Qué ocurre en una escritura

Primer nivel : Memoria CACHE

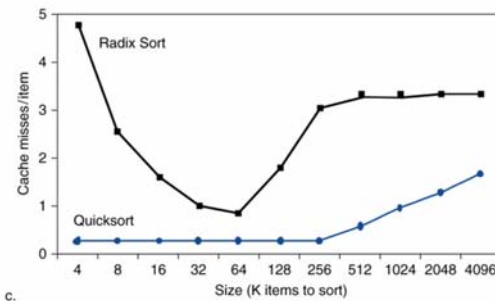
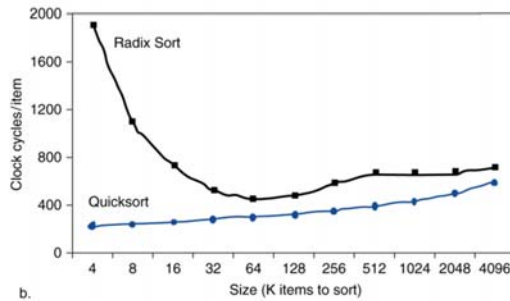


- ★ Memorias muy rápidas
- ★ Poca capacidad
- ★ Se ubican entre el procesador y la memoria principal

Interacción con el Software



- Los fallos dependen de los patrones de acceso a la memoria
 - Complejidad Algoritmo/Memoria
 - Quicksort $O(n \log n)$ / $O(\log n)$
 - Radix sort $O(nw)$ / $O(n)$
 - Optimizaciones del compilador para acceso a memoria



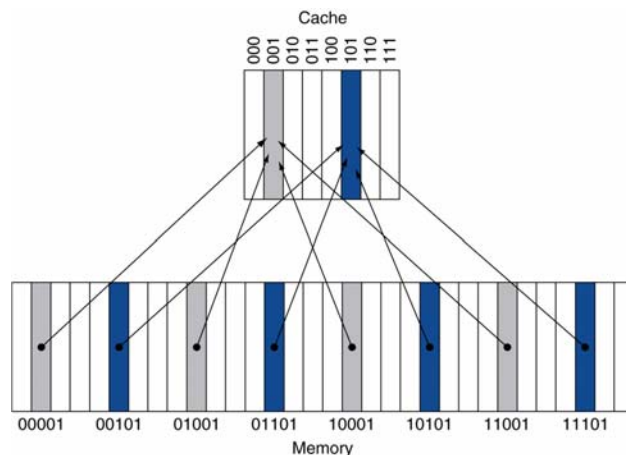
Cache - 2016

Ver paper de LaMarca and Ladner, 1996.

Cache de Correspondencia Directa



- La dirección determina la ubicación
 - Hay una sola opción
 - $(\#bloque) \text{ modulo } (\#\text{Bloques en cache})$



- $\#(\text{Bloques en la cache})$ es una potencia de 2.
- El módulo es simplemente la parte baja de la dirección.

Cache - 2016

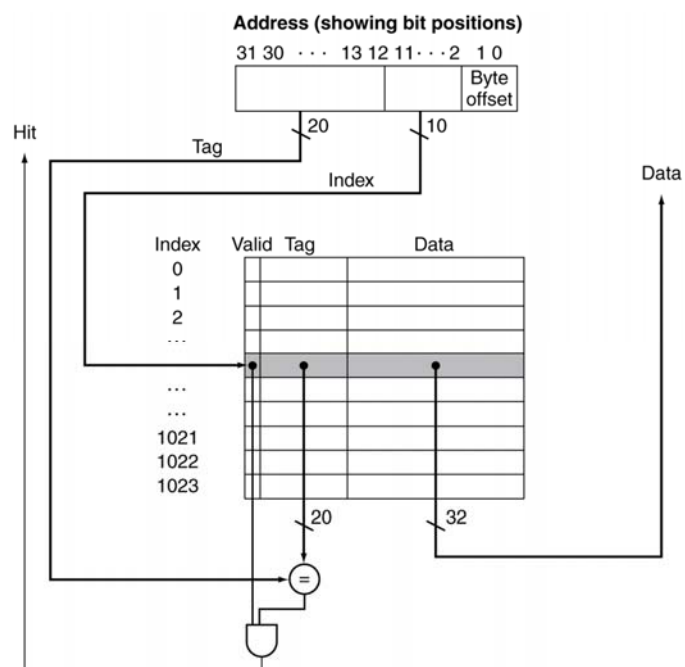


Etiquetas y bits de validez

- Cómo saber cuál es el bloque que se guardó en una línea de cache?
 - Se guarda la dirección del bloque, lo mismo que los datos
 - En realidad, solo se necesita guardar la parte alta de la dirección, llamada etiqueta
- Cómo se indica si no hay datos en una línea de cache?
 - Valid bit: 1 = presente, 0 = ausente
 - Inicialmente 0



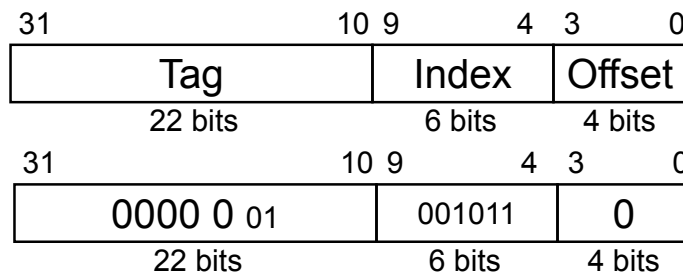
Subdivisión de las direcciones



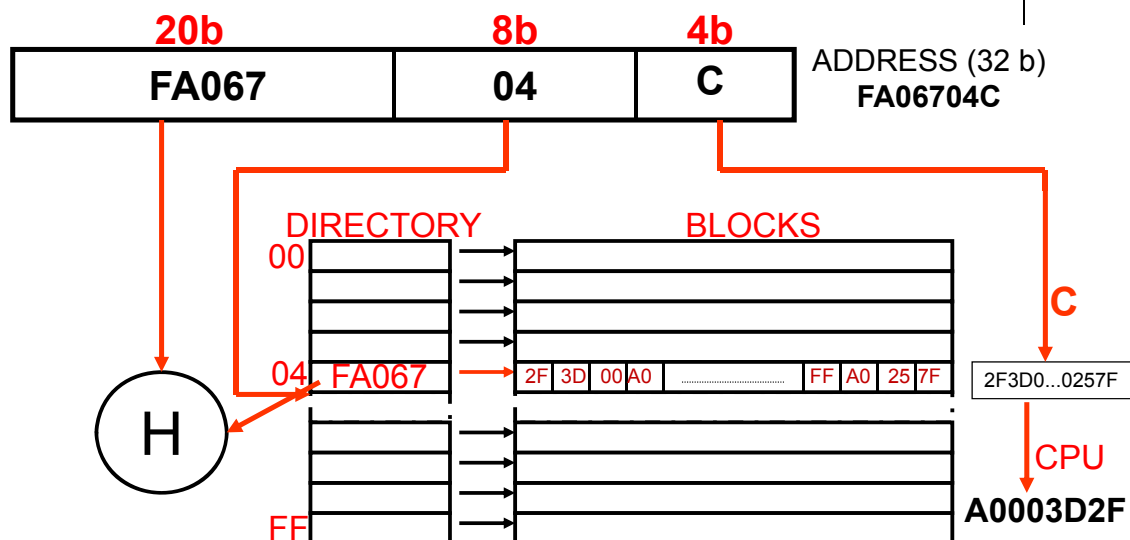
Ejemplo: aumentando el tamaño del bloque



- 64 bloques, 16 bytes/bloque
 - A qué bloque se mapea la dirección 1200?
 - $1200 = 0x0000\ 04B0$
- Dirección de bloque = $\lfloor 1200/16 \rfloor = 75$
- Número de bloque = $75 \bmod 64 = 11$



Ejemplo: Cache 4 KB, 16 bytes/bloque



Resolver



Una computadora tiene una memoria principal de 16 MB y emplea un bus de datos de 32 bits, dispone de una unidad caché de 8 KB.

Determinar el número de bits de cada campo de la dirección según las siguientes organizaciones:

- a) Correspondencia directa con una palabra por bloque.
- b) Correspondencia directa con ocho palabras por bloque.

Cache - 2016

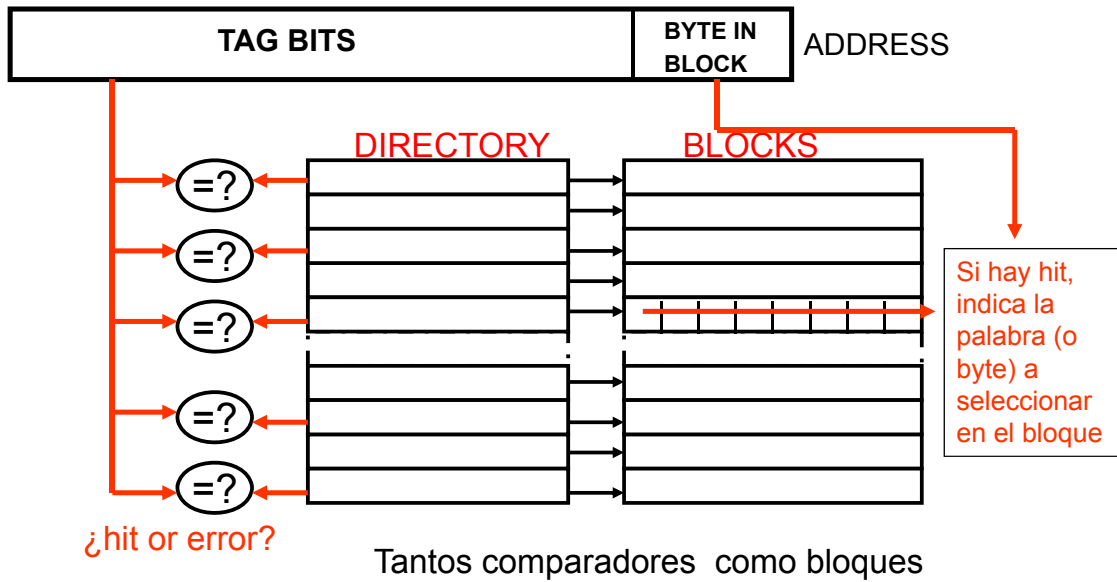
23

Caches Completamente Asociativa

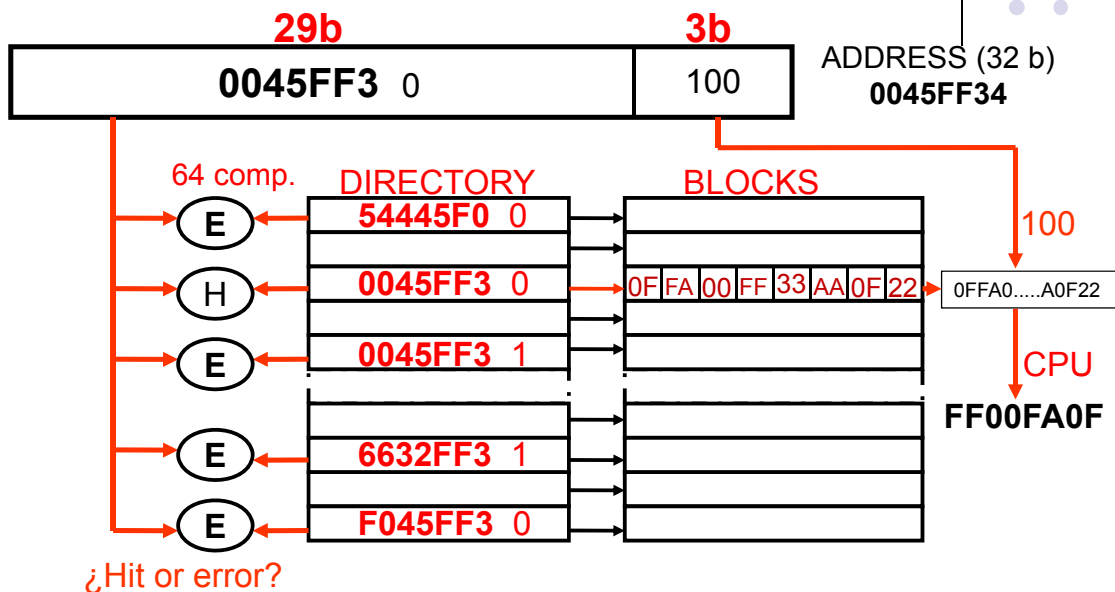


- Los bloques pueden guardarse en cualquier entrada de la cache
- Eso requiere que se deba buscar en todas las entradas
- Un comparador por entrada (costoso)

Completamente Asociativa (FA)



Ejemplo: Cache FA 512 bytes, 8 bytes/bloque



Cache Asociativa por Conjuntos

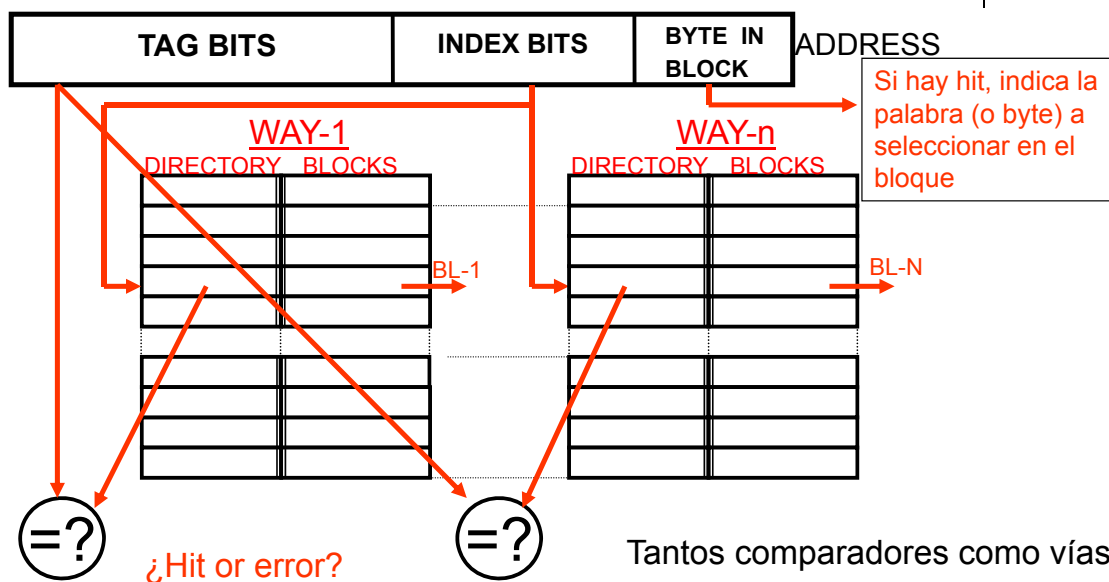


- *Asociativa por conjuntos de n-vías*
 - Cada conjunto contiene n entradas
 - El #bloque determina el conjunto
 - $(\# \text{ bloque}) \bmod (\# \text{ conjuntos en cache})$
 - La etiqueta se busca en todas las vías del conjunto
 - n comparadores (menos costosa que la completamente asociativa)

Cache - 2016

27

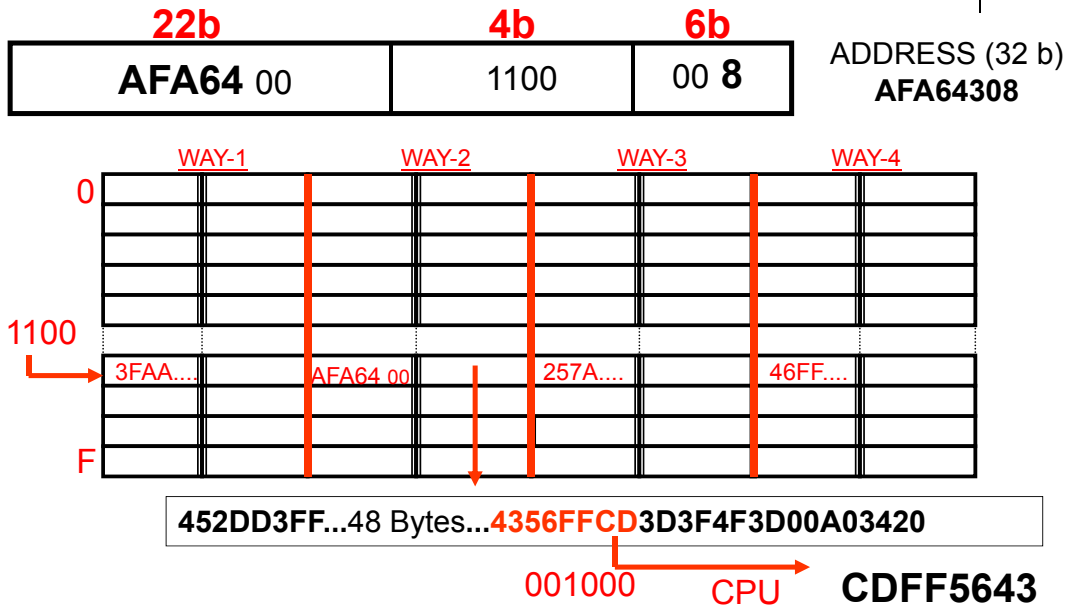
Asociativa por conjuntos, N vías (N-A)



Cache - 2016

28

Ejemplo: Cache N-A 4 vías, 4KB, 64 bytes/bloque



Cache - 2016

29

Cuánta asociatividad?

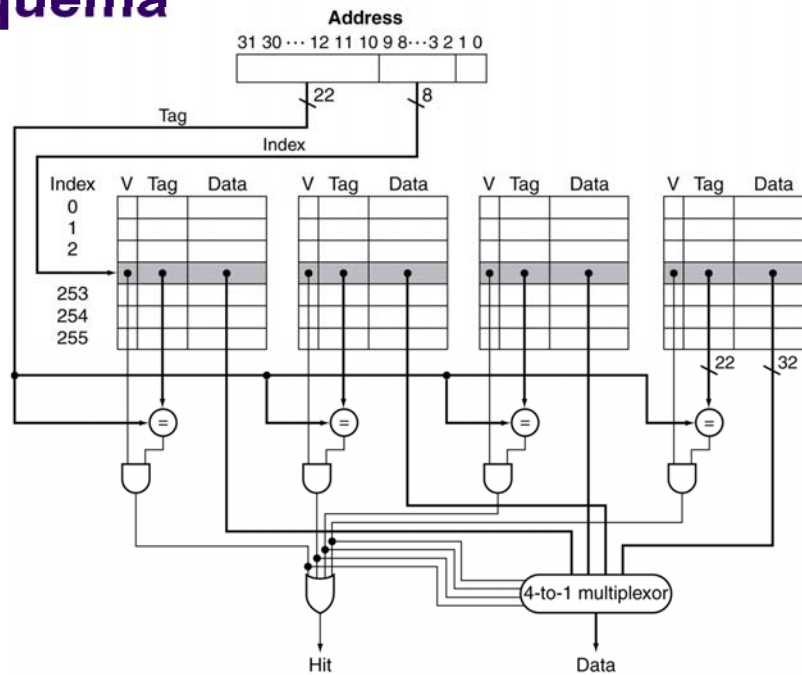


- Incrementar la asociatividad disminuye la tasa de fallos
 - Con rendimientos decrecientes
- Simulación de un sistema con D-cache de 64KB, bloques de 16 palabras, SPEC2000
 - 1-way: 10.3%
 - 2-way: 8.6%
 - 4-way: 8.3%
 - 8-way: 8.1%

Cache - 2016

30

Cache asociativa por conjuntos: esquema



Cache - 2016

31

Un mayor grado de detalle



- Hasta aquí se vieron las principales características y estructura de las caches.
- Ahora vamos a ver algunos detalles.

5.2.1. Running Cachegrind

To run Cachegrind on a program `prog`, run:

```
valgrind --tool=cachegrind prog
```

The program will execute (slowly). Upon completion, summary statistics that look like this will be printed:

```
==31751== I refs:      27,742,716
==31751== I1 misses:    276
==31751== L1 misses:    275
==31751== I1 miss rate:  0.0%
==31751== L1 miss rate:  0.0%
==31751==
==31751== D refs:      15,430,290 (10,955,517 rd + 4,474,773 wr)
==31751== D1 misses:    41,185 ( 21,905 rd + 19,280 wr)
==31751== L1d misses:   23,085 ( 3,987 rd + 19,098 wr)
==31751== D1 miss rate:  0.2% ( 0.1% + 0.4%)
==31751== L1d miss rate: 0.1% ( 0.0% + 0.4%)
==31751==
==31751== LL misses:    23,360 ( 4,262 rd + 19,098 wr)
==31751== LL miss rate:  0.0% ( 0.0% + 0.4%)
```

32

Consideraciones para el tamaño del bloque



- Bloques más grandes reducen la tasa de fallos
 - Por localidad espacial
- Pero si la cache es de tamaño fijo
 - Bloques grandes \Rightarrow menos bloques
 - Más competencia \Rightarrow incrementa la tasa de fallos
- Mayor penalidad de fallo
 - Puede contrarrestar el beneficio de la reducción de la tasa de fallo
 - Ayudaría aplicar *early restart* y *critical-word-first*

Fallos de Cache



- En los aciertos de cache, la CPU procede normalmente
- Cuando hay fallos de cache
 - Se bloquea el pipeline de la CPU
 - Se captura un bloque del siguiente nivel en la jerarquía de memoria (RAM)



Write-Through

- En escrituras, si hay acierto, solo se actualiza el bloque en la cache
 - Cache y memoria estarán inconsistentes por un tiempo
- Write through: también actualiza la memoria
 - Pero alarga el tiempo de escritura
 - Ej., Si el CPI base = 1, 10% de *stores*, y la escritura a memoria lleva 100 ciclos
 - $CPI\ efectivo = 1 + 0.1 \times 100 = 11$
- Solución: write buffer
 - CPU continúa inmediatamente
 - Solo se bloquea la CPU en una escritura si el buffer está lleno



Write-Back

- Alternativa: En un acierto en escritura, solo se actualiza el bloque en la cache
 - Se mantiene info. si eso sucedió (dirty bit)
- Cuando se reemplaza un bloque con el dirty bit activo
 - Escribir el bloque a memoria
 - Se puede usar un buffer de escritura para permitir que el nuevo bloque se lea antes

Ubicación de los datos escritos



- Qué ocurre ante un fallo de escritura?
- Alternativas para write-through
 - Allocate on miss: se captura el bloque
 - Write around: no se captura el bloque
 - Los programas suelen escribir un bloque completo antes de leerlo (ej., inicialización)
- Para write-back
 - Usualmente se captura el bloque

Política de reemplazos



- Correspondencia directa: no hay alternativa
- Asociativa por conjuntos
 - Least-recently used (LRU)
 - Se elige la entrada que no se usó por más tiempo
 - Simple para 2 vías, manejable para 4 vías, complejo más allá
 - Random
 - Con asociatividad alta, se obtiene casi el mismo rendimiento que LRU

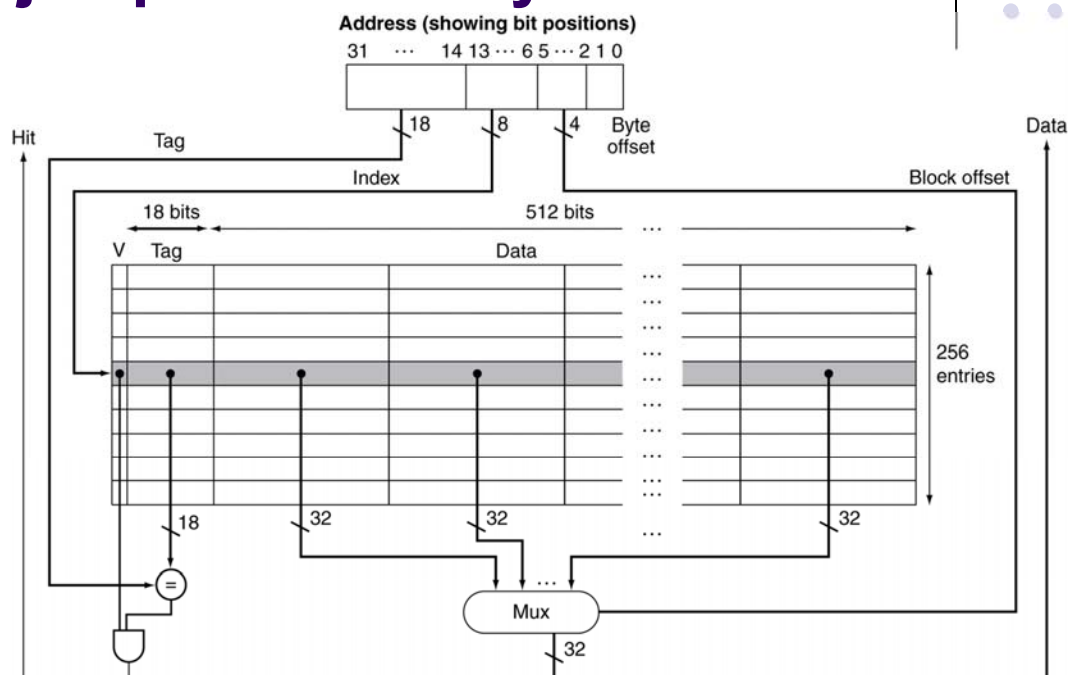


Ejemplo: Intrinsity FastMATH

- Procesador MIPS embebido
 - Pipeline de 12 etapas
- Split cache: I-cache y D-cache separadas
 - Cada una de 16KB: 256 bloques × 16 palabras/bloque
 - D-cache: write-through o write-back
- Tasas de fallos SPEC2000
 - I-cache: 0.4%
 - D-cache: 11.4%
 - Weighted average: 3.2%



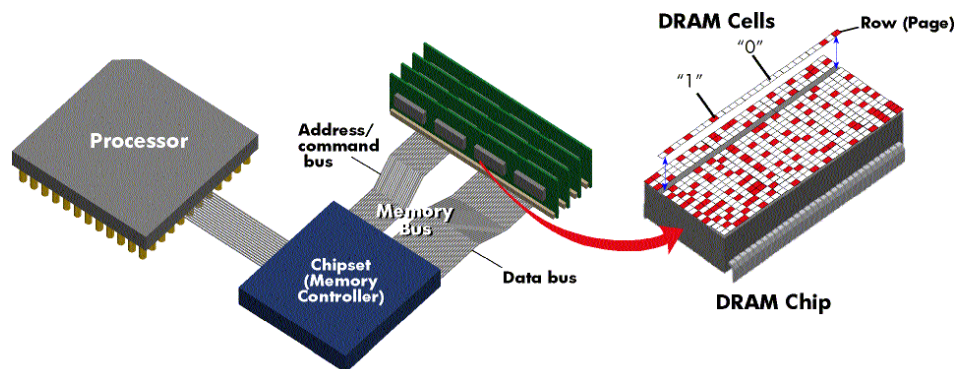
Ejemplo: Intrinsity FastMATH



Memoria principal que soporta Caches



- Se usan DRAMs como memoria principal
 - Ancho fijo (ej., 1 word)
 - Conectado por buses síncronos de ancho fijo
 - El reloj del bus es típicamente más lento que el de la CPU



Cache - 2016

41

Memoria principal que soporta Caches

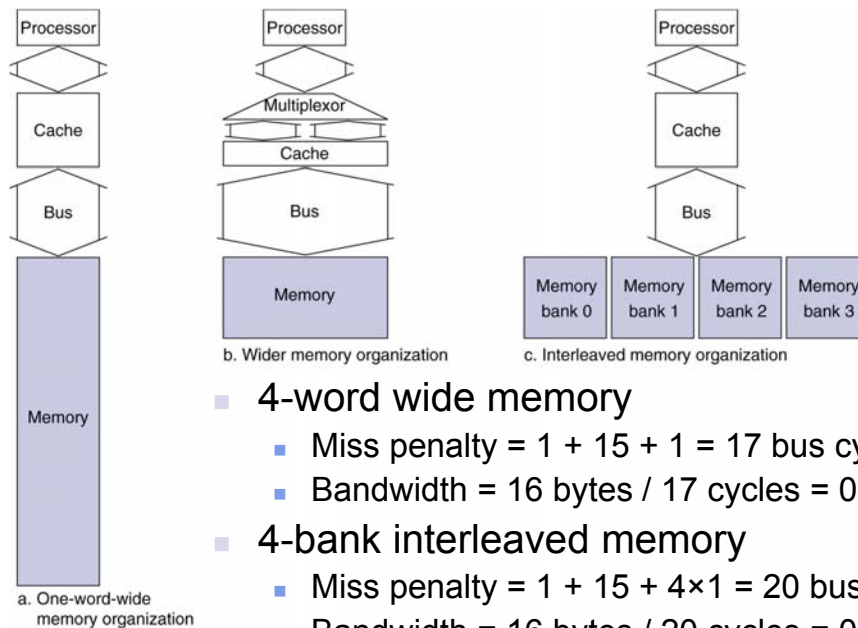


- Ejemplo de lectura de un bloque de cache
 - 1 ciclo de bus para transferir la dirección
 - 15 ciclos de bus por acceso a la DRAM
 - 1 ciclo de bus para transferir el dato
- Para un bloque de 4 words, 1-word-wide DRAM
 - Miss penalty = $1 + 4 \times 15 + 4 \times 1 = 65$ ciclos de bus
 - Bandwidth = $16 \text{ bytes} / 65 \text{ cycles} = 0.25 \text{ B/ciclo}$

Cache - 2016

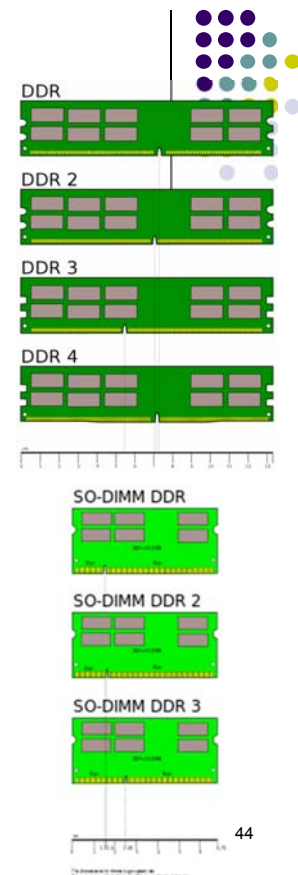
42

Incrementando el ancho de banda de la memoria



DRAMs más avanzadas

- Los bits en la DRAM se organizan en forma rectangular
 - La DRAM accede a una fila entera
 - Burst mode: Provee las palabras sucesivas de una fila con menor latencia
- Double data rate (DDR) DRAM
 - Transfiere en ambos flancos del reloj
- Quad data rate (QDR) DRAM
 - Entrada y salida de la DDR separadas





DDR4 vs DDR3



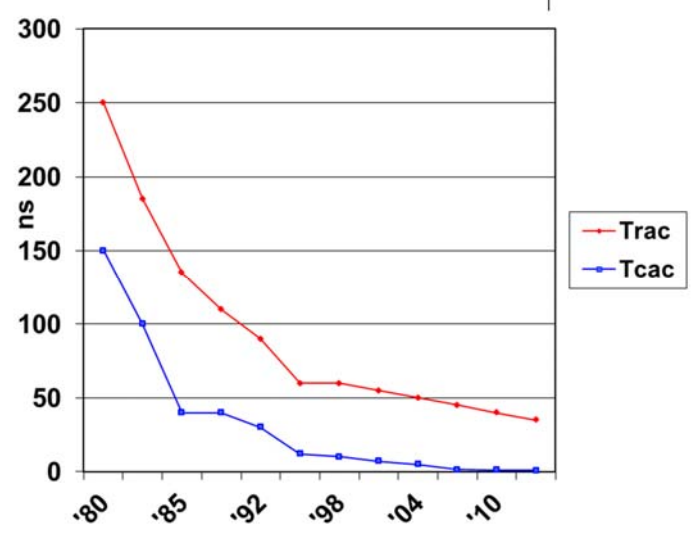
Badnwidth
12800 – 25600 MB/s

6400 – 17000 MB/s

Evolución de las DRAMs



Año	Capacidad	USD/GB
1980	64Kbit	\$1500000
1983	256Kbit	\$500000
1985	1Mbit	\$200000
1989	4Mbit	\$50000
1992	16Mbit	\$15000
1996	64Mbit	\$10000
1998	128Mbit	\$4000
2000	256Mbit	\$1000
2004	512Mbit	\$250
2007	1Gbit	\$50
2010	2Gbit	\$30
2012	4Gbit	\$1





Caches Multinivel

- La cache primaria está conectada a la CPU
 - Pequeña, pero rápida
- La cache de nivel 2 da servicio a los fallos de la cache primaria
 - Más grande, más lenta, pero más rápida que la memoria principal
- La memoria principal da servicio a los fallos de la cache de nivel 2
- Los sistemas de alto rendimiento incluyen una cache de nivel 3



Ejemplo de Cache Multinivel

- Datos
 - CPI base de CPU = 1, clock rate = 4GHz
 - Tasa de fallos/instrucción = 2%
 - Tiempo de acceso a memoria principal = 100ns
- Con una cache simple
 - Penalidad de fallo = $100\text{ns}/0.25\text{ns} = 400$ ciclos
 - CPI efectivo = $1 + 0.02 \times 400 = 9$



Ejemplo (cont.)

- Ahora agregamos cache L-2
 - Tiempo de acceso = 5ns
 - Tasa de fallos global a memoria = 0.5%
- L-1 miss con hit en L-2
 - Penalidad = $5\text{ns}/0.25\text{ns} = 20$ cycles
- L-1 miss con miss en L-2
 - Penalidad extra = 400 cycles
- $\text{CPI} = 1 + 0.02 \times 20 + 0.005 \times 400 = 3.4$
- $\text{Aceleración} = 9/3.4 = 2.6$

Consideraciones Cache Multinivel



- Cache L-1
 - Se enfoca en un tiempo de acierto mínimo
- Cache L-2
 - Se enfoca en una tasa de fallos mínima para evitar accesos a memoria
 - El tiempo de acierto tiene menos impacto
- Resultados
 - L-1 es más chica que usando cache única
 - Los bloques en L-1 son más chicos que en L-2

Jerarquía de Memoria

Memoria Cache

Elías Todorovich - Marcelo Tosini
Arquitectura I - Curso 2019

