



TEMA 6: Jerarquía de la Memoria

Introducción

Tecnología de Memorias

- Static RAM (SRAM) < 0.5ms - 2.5ns / 6B
- Dynamic RAM (DRAM) < 10ns - 50ns / 6B
- Magnetic Disk < 5ms - 20ms / 0.05 - 0.10 / 6B

Principio de localidad → programas acceden a memoria

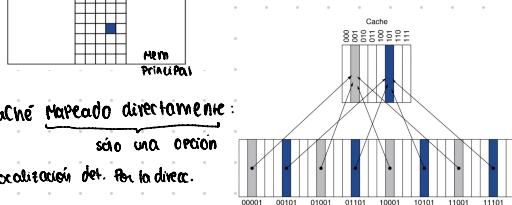
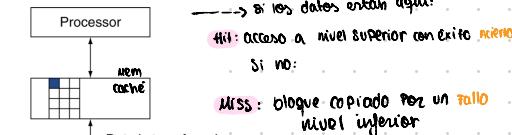
Localidad temporal

- Es probable que se accedan a los datos cercanos a los que se acceden pronto.
- Ej: instrucciones en loops

Localidad espacial

- Es probable que se accedan a los datos cercanos a los que se acceden pronto.
- Ej: array

Niveles de Jerarquía



Buffer de Escritura

Cuando hit → hay q actualizar cache y memoria
[sinco cache inconsistente]

Solución: buffer de escritura

Write-Back

Cuando data-write hit → actualizar cache +
OG si el bloque esté "Sucio"

Cuando un bloque "Sucio" → Substituido:
• Write-back en la memoria.
(Puede usar) + buffer de BT para permitir q se use
+ un bloq de reemplazo

Rendimiento Cache

$$\text{Ciclos stall} = \frac{\text{acceso memoria}}{\text{Programa}} \times \text{Tasa Miss} \times \text{Tasa Penalty}$$

↓
Penalidades

$$= \frac{\text{Instruc.}}{\text{Programa}} \times \frac{\text{Misses}}{\text{Instruc.}} \times \text{Tasa Penalidades}$$

Ejemplo:

- Given
- Cache miss rate = 2%
 - Cache miss rate = 4%
 - Miss penalty = 100 cycles
 - Base CPI (ideal cache) = 2
 - Load & stores are 36% of instructions
 - Miss cycles per instruction = $0.02 \times 100 = 2$
 - L-cache: 0.02 × 100 = 2
 - D-cache: $0.36 \times 100 = 100 = 1.44$
 - Actual CPI = $2 + 2 + 1.44 = 5.44$
 - Ideal CPU is $5.44/2 = 2.72$ times faster

I-cache

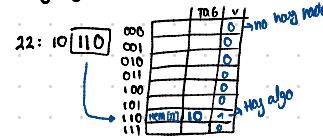
o-cache → incluye lru/sw

ideal cache

$$\rightarrow CPI = 0.02 \times 100 \text{ miss} + 0.36 \times 100 \text{ CPI} + \text{Miss} \times 2$$

Miss

Tags y V: bits válidos



Acerto: vos a caché y ya está → H

Fallo: vos a caché y no está → N
Lo sacas de la memoria principal y lo traes

010 [mem [23]] 10 1

y querés 10: 10 010 → reescribo

010 [mem [18]] 11 1

Ejemplo: 8 bloques, 1 word/block, mapeo directo: Ej: Mem cache de 32b PREGUNTARI!!
direcc. q queremos leer: 22, 26, 22, 16, 16, 3

| | | TAGS | | V |
|-----|----------|------|---|---|
| 000 | mem [16] | 10 | 1 | |
| 001 | | 0 | | |
| 010 | mem [26] | 10 | 1 | |
| 011 | | 0 | | |
| 100 | mem [32] | 00 | 1 | |
| 101 | | 0 | | |
| 110 | mem [22] | 10 | 1 | |
| 111 | | 0 | | |

22: 10 110 → 10 110: H

26: 11 010

16: 10 000

3: 00 011

TEMA 6: Jerarquía de la Memoria

Caches Asociativos

Fully associative

- Permiten a un bloque dado ir a cualquier entrada del cache
- Requiere → Todas las Es sean 1 de una vez
- Comparador por entrada (\$\$)

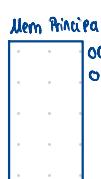
n-Way set associative

- cada una → contiene n entradas
- nom bloque dato en q set va
- 1 todos los Es en el set dado de una vez
- n comparadores (menos \$\$)

2-way associative:

| set 0 | set 1 |
|------------|------------|
| Mem [0000] | Mem [0001] |
| | |
| | |
| | |

2 conjuntos de
4 bloques



Ejemplo con 2-ways : 0, 8, 0, 6, 8

| set 0 | set 1 |
|------------|----------|
| Mem [0/18] | yo/Julio |
| | |
| 2 Mem [6] | Julio |
| 3 | |

$$0 \bmod 4 \rightarrow \frac{0+4}{0+4} = 0$$

$$8 \bmod 4 \rightarrow \frac{8+4}{0+4} = 0$$

$$6 \bmod 4 \rightarrow \frac{6+4}{2+4} = 2$$

| direcc bloq | índice | hit/miss | get 0 | get 1 |
|-------------|--------|----------|---------|---------|
| 0 | 0 | MISS | Mem[0] | |
| 8 | 0 | MISS | Mem[03] | Mem[03] |
| 0 | 0 | HIT | Mem[0] | Mem[08] |
| 6 | 2 | MISS | Mem[6] | |

1-way set associative (direct mapped)

| Block | Tag | Data |
|-------|-----|------|
| 0 | | |
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

2-way set associative

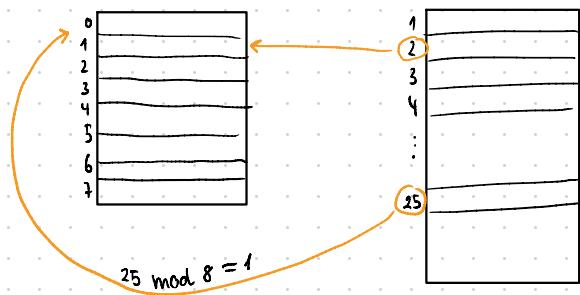
| Set | Tag | Data | Tag | Data |
|-----|-----|------|-----|------|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

4-way set associative

| Set | Tag | Data | Tag | Data | Tag | Data | Tag | Data |
|-----|-----|------|-----|------|-----|------|-----|------|
| 0 | | | | | | | | |
| 1 | | | | | | | | |

8-way set associative (fully associative)

| Tag | Data |
|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|
| | | | | | | | | | | | | | |



Tema 6: Jerarquía de la Memoria

Ejercicio: Asumiendo una caché de 4096 bloques, un tamaño de 4 palabras por bloque y una dirección de 32 bits, encuentre los bits para tag, index offset.

a) Mapo Directo

vamos directos a la dirección → nos dice la memoria → 1 conjunto por bloque → 4096 conjuntos

| TAG | INDEX | OFFSET |
|---------------|-------|--------|
| 32-12-4 = 16b | 12b | 4b |

↓
4096 = 2^{12}

4 words = 16B
2¹²

b) 2-way → conjuntos: $\frac{4096}{2} = 2048$

| TAG | INDEX | OFFSET |
|---------------|-------|--------|
| 32-11-4 = 17b | 11b | 4b |

↓
2048 = 2^11

c) 4-way conjuntos $\frac{4096}{4} = 1024$

| TAG | INDEX | OFFSET |
|-----|-------|--------|
| 18b | 10b | 4b |

d) Fully Associative 1 solo conjunto

| TAG | INDEX | OFFSET |
|-----------|-------|--------|
| 32-4 = 28 | 0b | 4b |

Asuma que un procesador genera la secuencia de direcciones de 8 bits que se muestra en la tabla siguiente. La cache del procesador utiliza bloques de 4 bytes, es asociativa por conjuntos de 2 vías, tiene una capacidad de 4 bloques y utiliza una política de remplazo de tipo LRU.

Asuma la cache inicialmente vacía y determine los campos TAG, INDEX, SET (offset) y rellene la tabla.

| tiempo | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----------|----------|----------|----------|----------|----------|----------|----------|----------|
| Dirección | 10001101 | 10110010 | 10111111 | 10001100 | 10011100 | 11101001 | 11111110 | 11101001 |
| TAG | 10001 | 10110 | 10111 | 10001 | 00011 | 11101 | 11111 | 11101 |
| SET | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| INDEX | 00 | 10 | 11 | 00 | 00 | 01 | 10 | 01 |
| | | | | | | | | |
| | | | | | | | | |

↓
bloque 0

↓
bloque 1

En la figura siguiente marque para cada acceso el TAG, el bit LRU y el bit de Hit.

| Acceso 0 | | Acceso 1 | |
|----------|-----------|----------|-----------|
| Bloque 0 | Bloque 1 | Bloque 0 | Bloque 1 |
| Set 0 | | Set 0 | 10110,0,0 |
| Set 1 | 10001,0,0 | Set 1 | 10001,0,0 |

↓
TAG LRU Hit

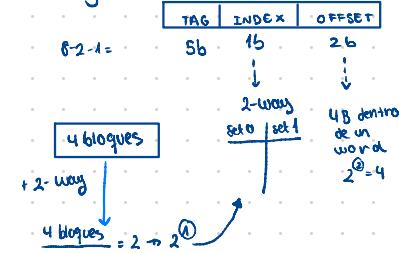
| Acceso 2 | | Acceso 3 | |
|----------|-----------|----------|-----------|
| Bloque 0 | Bloque 1 | Bloque 0 | Bloque 1 |
| Set 0 | 10110,0,0 | Set 0 | 10110,0,0 |
| Set 1 | 10001,0,0 | Set 1 | 10001,0,0 |

↓
1 pg no ha sido utilizado por mucho

↓
ya lo tenemos pero solo en este acceso

LRU bit = 1 → si el bloque es el menos utilizado recientemente
Last Recently Used: the one unused for a long time

2-way associative



| Acceso 4 | Bloque 0 | Bloque 1 | Acceso 5 | Bloque 0 | Bloque 1 |
|----------|-----------|-----------|----------|-----------|-----------|
| Set 0 | 10110,0,0 | | Set 0 | 10110,0,0 | |
| Set 1 | 10001,0,0 | 10001,0,0 | Set 1 | 10001,0,0 | 10001,0,0 |

10011 G set 1

11101 G set 0

10110 G set 0

11101 G set 1

11101 G set 0

11101 G set 1

Los bloques menos utilizados, cuando hay empate: tomar los de abajo

10110 G set 0

11101 G set 1

11101 G set 0

10110 G set 1