

Práctica 8 - Memoria Caché

Organización del Computador I

Paula Vergelet
en base al material realizado por Verónica Coy

Departamento de Computación
FCEyN - UBA

2 de noviembre de 2017

Tiempo de acceso (Intro. performance)

- Pregunta: ¿puedo hacer que el cómputo sea más rápido?
- La CPU **es** muy rápida mientras que el acceso a Memoria Principal no.
- ¿Puedo usar **muchá memoria, muy rápida, barata** y que consuma **poca energía**?

Tiempo de acceso (Intro. performance)

- Pregunta: ¿puedo hacer que el cómputo sea más rápido?
- La CPU **es** muy rápida mientras que el acceso a Memoria Principal no.
- ¿Puedo usar **muchá memoria, muy rápida, barata** y que consuma **poca energía**?

Problema

- ① La memoria **rápida es cara** y **consume mucha energía** (ej. Registros).
- ② La memoria **barata** y de **bajo consumo** es **lenta** (ej. RAM).

Tiempo de acceso (Intro. performance)

- Pregunta: ¿puedo hacer que el cómputo sea más rápido?
- La CPU **es** muy rápida mientras que el acceso a Memoria Principal no.
- ¿Puedo usar **muchá memoria, muy rápida, barata** y que consuma **poca energía**?

Problema

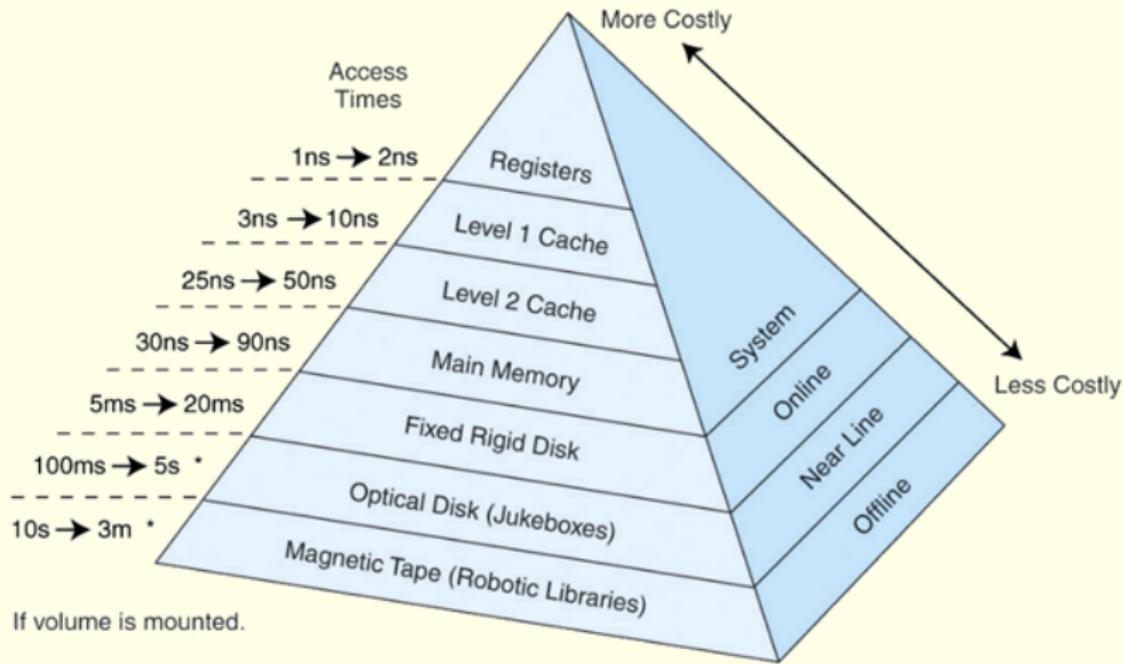
- ① La memoria **rápida es cara** y **consume mucha energía** (ej. Registros).
- ② La memoria **barata** y de **bajo consumo** es **lenta** (ej. RAM).

Una solución

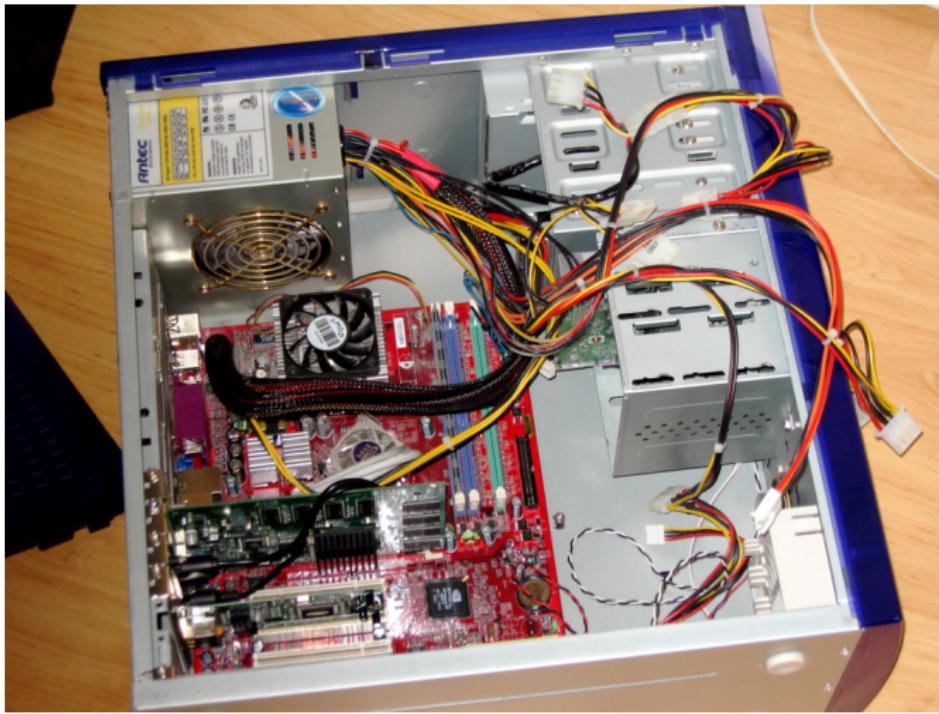
- ① Uso mucha memoria de la barata y lenta,
- ② y un poquito de memoria de la cara y rápida,
- ③ en la que mantengo una copia de los datos *más usados*.

Jerarquía de Memoria

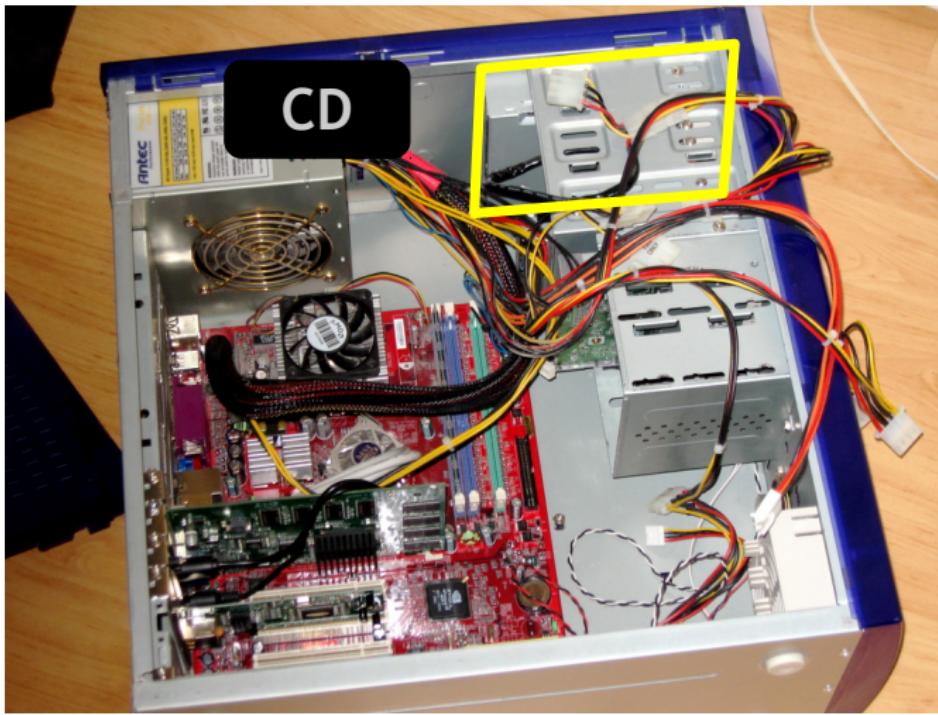
Jerarquía de Memoria



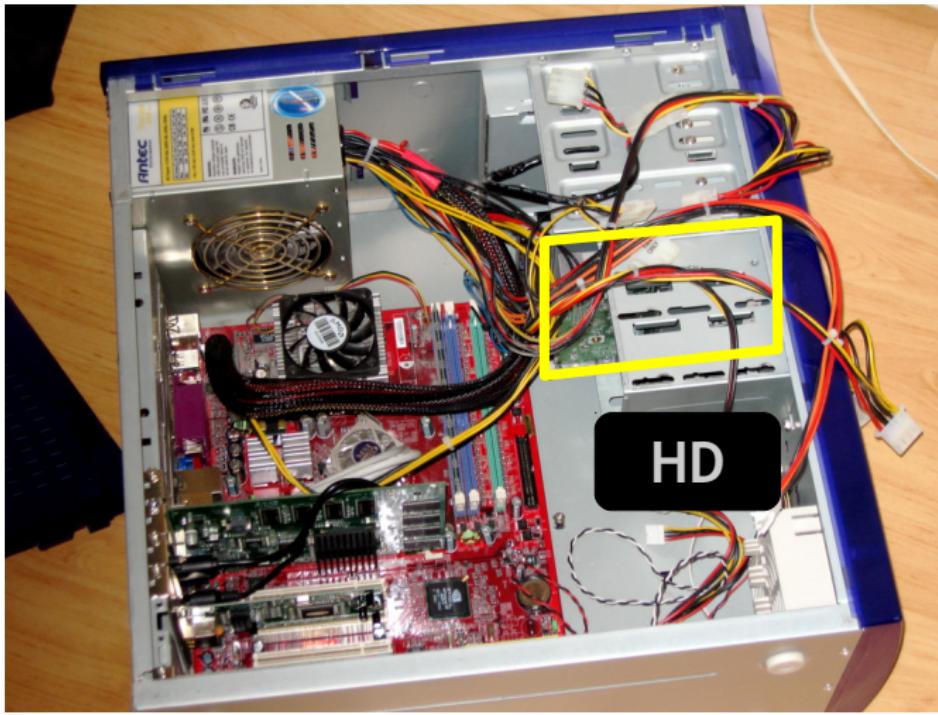
Jerarquía de Memoria (Versión Ilustrada)



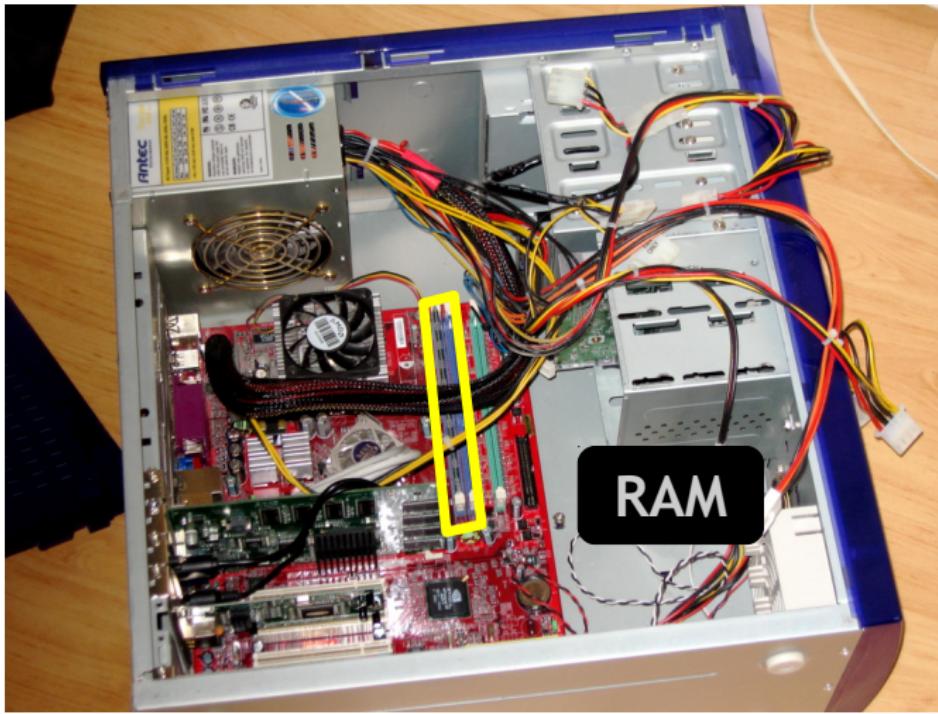
Jerarquía de Memoria (Versión Ilustrada)



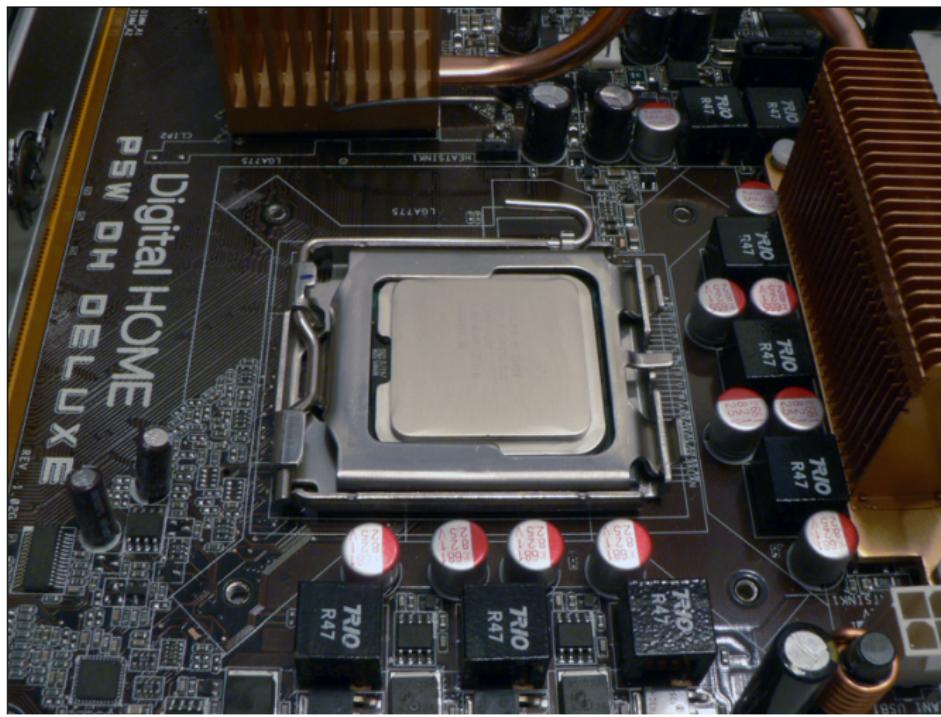
Jerarquía de Memoria (Versión Ilustrada)



Jerarquía de Memoria (Versión Ilustrada)



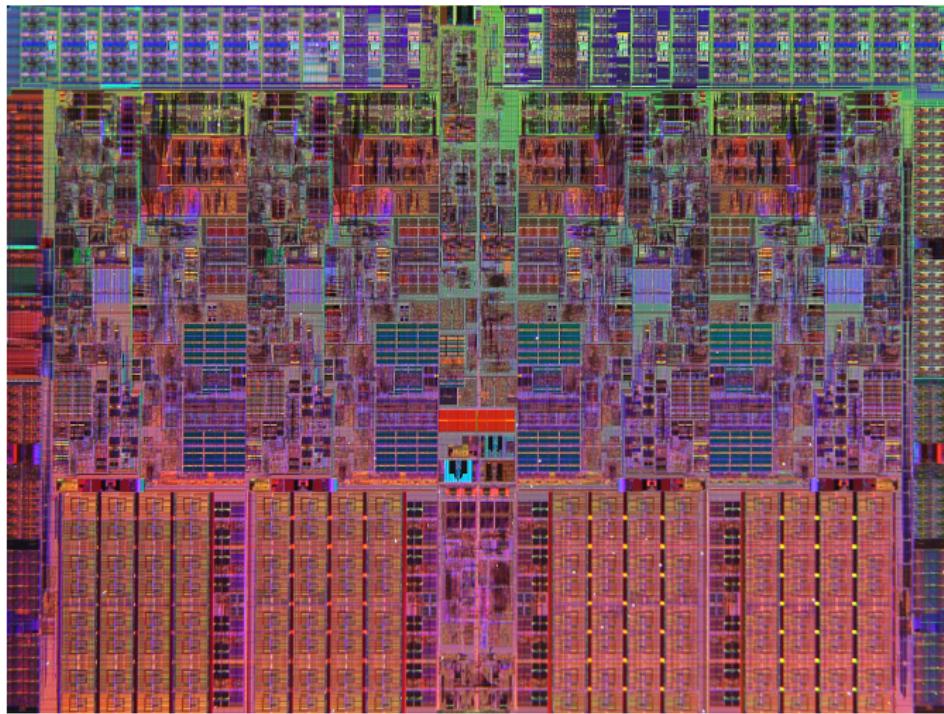
Jerarquía de Memoria (Versión Ilustrada)



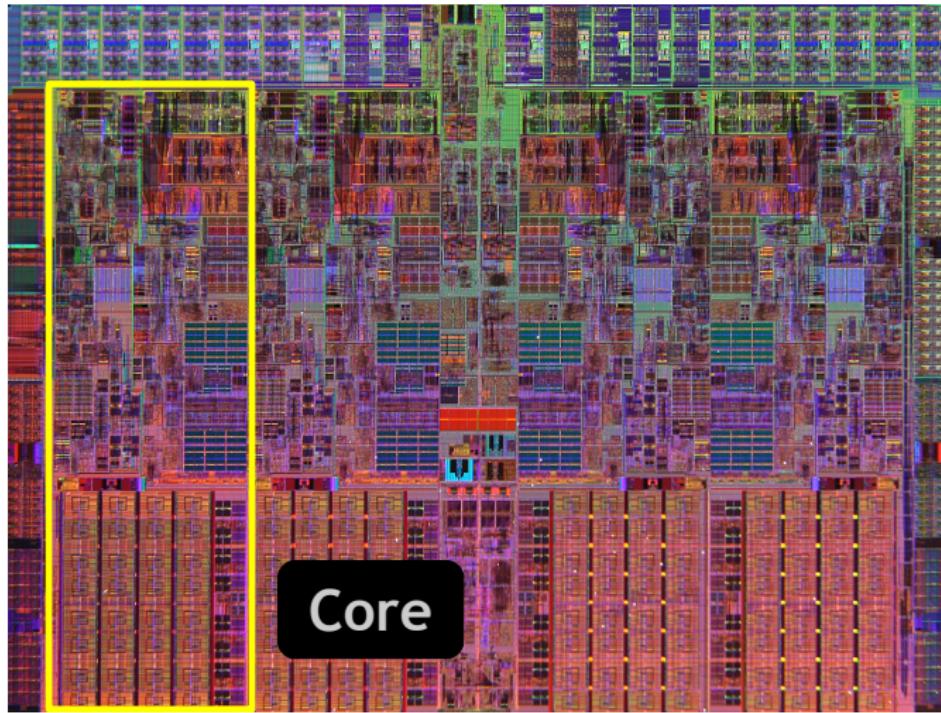
Jerarquía de Memoria (Versión Ilustrada)



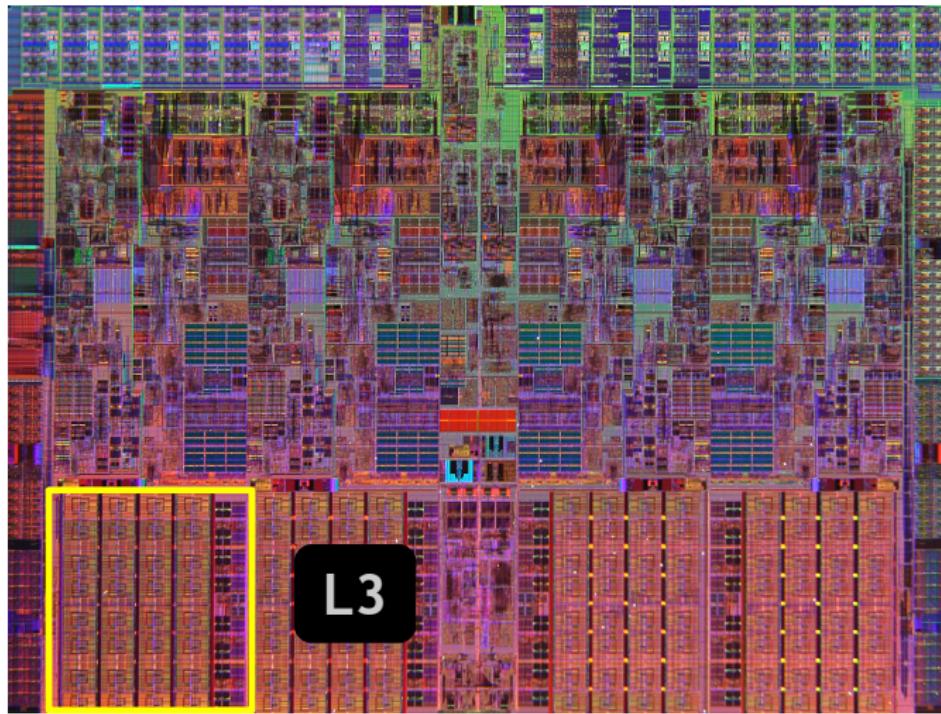
Jerarquía de Memoria (Versión Ilustrada): processor die map



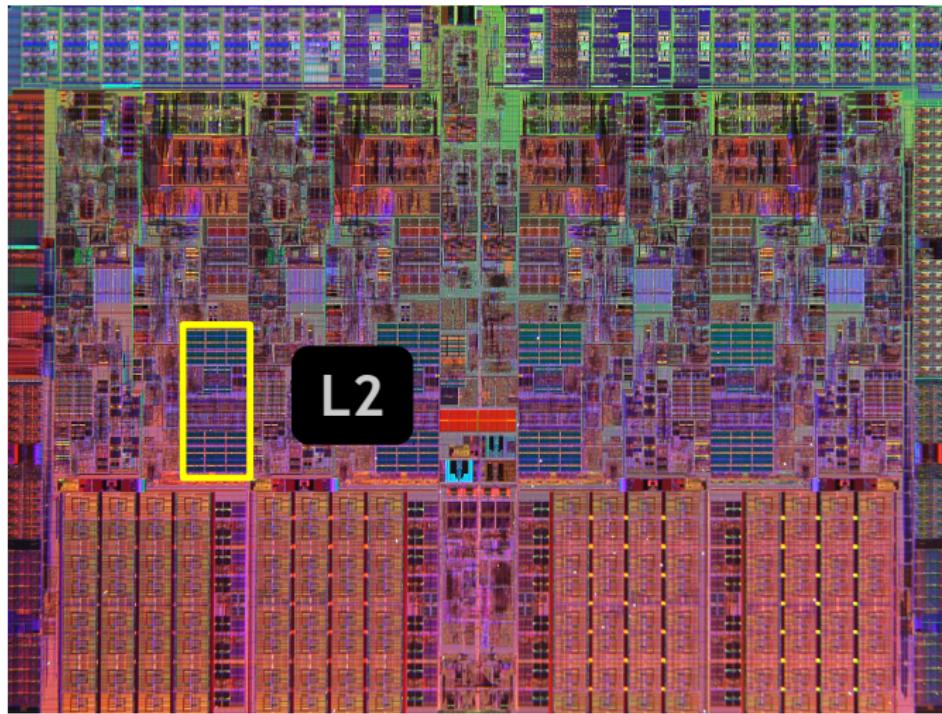
Jerarquía de Memoria (Versión Ilustrada): processor die map



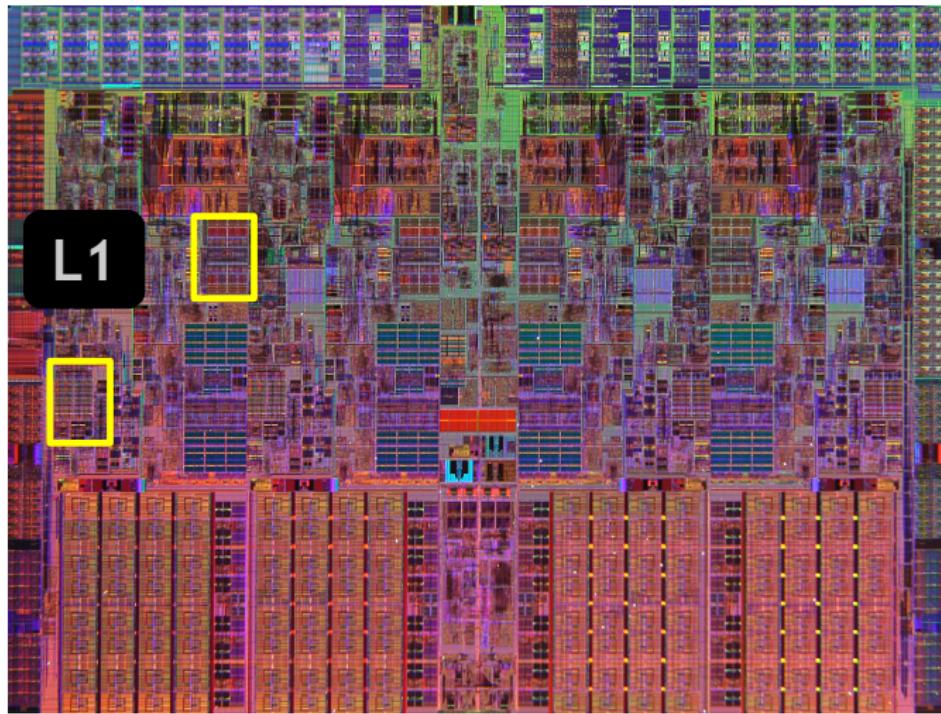
Jerarquía de Memoria (Versión Ilustrada): processor die map



Jerarquía de Memoria (Versión Ilustrada): processor die map



Jerarquía de Memoria (Versión Ilustrada): processor die map



La Memoria Caché

¿Qué es una memoria Caché?

Es un espacio de almacenamiento intermedio entre el CPU y la memoria principal.

¿Por qué la necesitamos?

Porque el crecimiento de la velocidad de los CPUs en las últimas décadas ha sido exponencial respecto al de las memorias. El uso de la caché sirve para minimizar el tiempo de acceso a los datos mejorando así la performance del sistema.

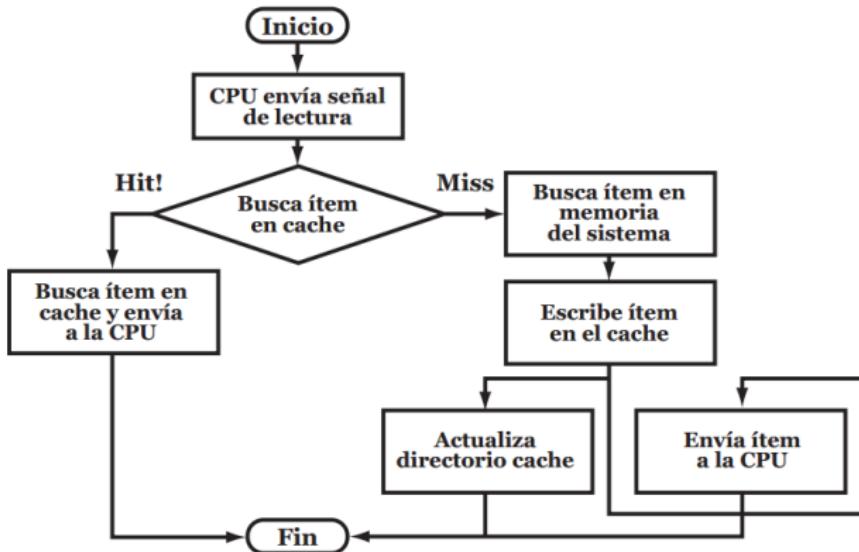
¿Cuánto más rápida es?

Entre 10 y 100 veces mas rápida que la memoria principal.

¿Por qué entonces no usamos sólo memoria caché?

Porque la memoria caché es muy cara.

¿Cómo funciona?



Los pedidos a la caché se clasifican como:

- **Hit:** Si el dato de la dirección solicitada por el CPU se encuentra en caché.
- **Miss:** En caso contrario.

Métricas para performance

En base a estos dos eventos, *miss* y *hit*, se obtienen métricas para estimar el desempeño de una caché:

$$\textbf{Hit Rate} = \frac{\text{cantidad de hits}}{\text{cantidad de pedidos}}$$

$$\textbf{Miss Rate} = \frac{\text{cantidad de miss}}{\text{cantidad de pedidos}}$$

Observación:

$$\textit{Hit Rate} + \textit{Miss Rate} = 1$$

Métricas para performance

En base a estos dos eventos, *miss* y *hit*, se obtienen métricas para estimar el desempeño de una caché:

$$\textbf{Hit Rate} = \frac{\text{cantidad de hits}}{\text{cantidad de pedidos}}$$

$$\textbf{Miss Rate} = \frac{\text{cantidad de miss}}{\text{cantidad de pedidos}}$$

Observación:

$$\textit{Hit Rate} + \textit{Miss Rate} = 1$$

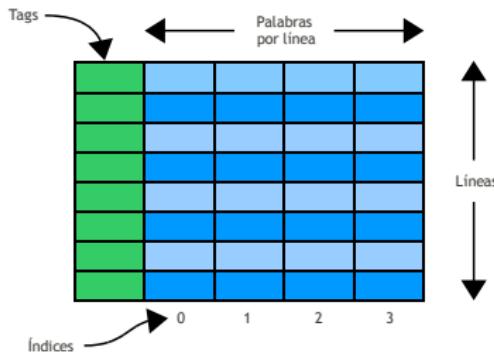
Nuestro objetivo es lograr qué el **hit rate** sea lo más alto posible.

Pensando una memoria caché

¿Qué cosas debería tener en cuenta a la hora de pensar una memoria caché?

- **Tamaño:** Debería ser lo suficientemente grande como para maximizar los **hits**, pero a la vez, que sea lo suficientemente pequeña como para no afectar el nivel de consumo.
- **Organización:** Establecer de qué forma vinculo las direcciones de memoria con las posiciones de mi caché.
- **Política de desalojo:** Qué hacer cuando se llena la caché.

Estructura de una Caché



- **Línea:** Las memorias caché se componen por unidades denominadas **líneas** cuyo tamaño es múltiplo del tamaño de palabra que utiliza el CPU. Es la unidad que se carga cada vez que se hace un acceso a memoria.
- **Índice:** Un índice permite ubicar una palabra dentro de una línea
- **Tag:** Es el identificador usado para mantener la correspondencia entre la línea en caché y la ubicación de dicho contenido en memoria. El modo de trazar la correspondencia varía según el tipo de caché.

Función de correspondencia

Entre Memoria Principal y Memoria Caché.

Tres soluciones:

- Correspondencia Directa (o Mapeo directo).
- Totalmente Asociativa.
- Asociativa por Conjunto de N vías.

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

0x0	0x1	0x2	0x3
0x0	0xA	0x0	0x0
0x4	0x5	0x6	0x7
0x0	0x0	0xF	0x0
0x8	0x9	0xA	0xB
0x0	0x0	0x5	0x0
0xC	0xD	0xE	0xF
0x0	0x0	0x0	0x0

Permite que cada bloque de memoria principal pueda cargarse en cualquier línea de la caché.

Cada una de estas líneas identifica únicamente un bloque de la memoria principal por medio del tag o etiqueta.

Para decidir si una línea esta en la caché se debe examinar todas los tags almacenados en la caché.

Cache

índice→ tag↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

0000	0001	0010	0011
0000	1010	0000	0000
0100	0101	0110	0111
0000	0000	1111	0000
1000	1001	1010	1011
0000	0000	0101	0000
1100	1101	1110	1111
0000	0000	0000	0000

Cache

indice → tag ↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

¿Cómo se traza la correspondencia entre las posiciones de memoria y la línea de caché?

Dividiendo la memoria en bloques y numerándolos. ¿Cuántos bloques?

$$\begin{aligned} \# \text{bloques}_{mem} &= \frac{|\text{memoria}|}{|\text{linea}_{cache}|} \\ &= \frac{|\text{memoria}|}{\frac{|\text{cache}|}{\#\text{lineas}_{cache}}} \\ &= \frac{16 \cancel{\text{unidades}}}{8 \cancel{\text{unidades/bloque}}} \\ &\quad 2 \end{aligned}$$

= 4 bloques → 2 bits para tags

Cada bloque de memoria se puede asociar con cualquier línea de caché.

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

índice → tag ↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

¿Cómo se traza la correspondencia entre las posiciones de memoria y la línea de caché?

Dividiendo la memoria en bloques y numerándolos. ¿Cuántos bloques?

$$\begin{aligned} \#\text{bloques}_{\text{mem}} &= \frac{|\text{memoria}|}{|\text{linea}_{\text{cache}}|} \\ &= \frac{|\text{memoria}|}{\frac{|\text{cache}|}{\#\text{lineas}_{\text{cache}}}} \\ &= \frac{16 \cancel{\text{unidades}}}{8 \cancel{\text{unidades / bloque}}} \\ &\quad 2 \end{aligned}$$

= 4 bloques → 2 bits para tags

Cada bloque de memoria se puede de asociar con cualquier línea de caché.

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Pregunta:

¿Qué ocurre cuando el CPU solicita la dirección 0x6?

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

índice→ tag↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

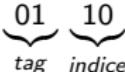
indice→ tag↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

Pregunta:

¿Qué ocurre cuando el CPU solicita la dirección 0x6?

Respuesta:

Tomo los dos primeros bits de la dirección para identificar el tag:

0x6 → 
tag *indice*

Como el tag no está cargado en caché se produce un **miss**. Voy a la memoria, busco la línea y la traigo.

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

indice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
--	---	---	---	---

Pregunta:

¿Qué ocurre cuando el CPU solicita la dirección 0x6?

Respuesta:

Tomo los dos primeros bits de la dirección para identificar el tag:

$$0x6 \rightarrow \underbrace{01}_{\text{tag}} \underbrace{10}_{\text{indice}}$$

Como el tag no está cargado en caché se produce un **miss**. Voy a la memoria, busco la línea y la traigo.

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Pregunta:

¿Y si ahora quiere traer la dirección 0x1?

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

índice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
--	---	---	---	---

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

indice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
--	---	---	---	---

Pregunta:

¿Y si ahora quiere traer la dirección 0x1?

Respuesta:

Idem:

0x1 → 
tag *indice*

Como el tag no está cargado en caché se produce un **miss**. Voy a la memoria, busco la línea y la traigo.

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

indice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
00	0000	1010	0000	0000

Pregunta:

¿Y si ahora quiere traer la dirección 0x1?

Respuesta:

Idem:

0x1 → $\underbrace{00}_{tag} \underbrace{01}_{indice}$

Como el tag no está cargado en caché se produce un **miss**. Voy a la memoria, busco la línea y la traigo.

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Pregunta:

Qué ocurriría si el CPU hiciera los siguientes pedidos:

0x6 → 01 10

0x1 → 00 01

⋮ ⋮ ⋮

0x6 → 01 10

0x1 → 00 01

Cache

indice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
00	0000	1010	0000	0000

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

indice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
00	0000	1010	0000	0000

Pregunta:

Qué ocurriría si el CPU hiciera los siguientes pedidos:

0x6 → 01 10

0x1 → 00 01

⋮ ⋮ ⋮

0x6 → 01 10

0x1 → 00 01

Respuesta:

Nada, puesto que esas direcciones ya están en caché. Serían todos **hits!**

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Pregunta:

Ahora, ¿qué ocurriría si el CPU pidiera la dirección 0xA?:

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

índice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
00	0000	1010	0000	0000

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

indice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
00	0000	1010	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU pidiere la dirección 0xA?:

Respuesta:

0xA → $\underbrace{10}_{\text{tag}}$ $\underbrace{10}_{\text{indice}}$

Miss de nuevo, pero ahora tengo la caché llena. Debería desalojar alguna línea, ¿pero cuál?

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

indice→ tag↓	00	01	10	11
01	0000	0000	1111	0000
00	0000	1010	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU pidiere la dirección 0xA?:

Respuesta:

0xA → $\underbrace{10}_{\text{tag}}$ $\underbrace{10}_{\text{indice}}$

Miss de nuevo, pero ahora tengo la caché llena. Debería desalojar alguna línea, ¿pero cuál?

- Random
- FIFO (First In First Out)
- LFU (Least Frequently Used)
- LRU (Least Recently Used)

Caché Totalmente Asociativa

Caché Totalmente Asociativa

Memoria

B.0	0000	0001	0010	0011
	0000	1010	0000	0000
B.1	0100	0101	0110	0111
	0000	0000	1111	0000
B.2	1000	1001	1010	1011
	0000	0000	0101	0000
B.3	1100	1101	1110	1111
	0000	0000	0000	0000

Cache

indice→ tag↓	00	01	10	11
10	0000	0000	0101	0000
00	0000	1010	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU pidiere la dirección 0xA?:

Respuesta:

0xA → $\underbrace{10}_{\text{tag}}$ $\underbrace{10}_{\text{indice}}$

Miss de nuevo, pero ahora tengo la caché llena. Debería desalojar alguna línea, ¿pero cuál?

- Random
- FIFO (First In First Out)
- LFU (Least Frequently Used)
- LRU (Least Recently Used)

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Por cada línea tengo que guardar a qué bloque pertenece.

Memoria

0x0	0x1	0x2	0x3
0x0	0xA	0x0	0x0
0x4	0x5	0x6	0x7
0x0	0x0	0xF	0x0
0x8	0x9	0xA	0xB
0x0	0x0	0x0	0x0
0xC	0xD	0xE	0xF
0x0	0x5	0x0	0x0

Cache

indice→ tag↓	00	01	10	11
--	--	--	--	--
--	--	--	--	--

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

¿Cómo se traza la correspondencia entre las posiciones de memoria y la línea de caché?

Memoria

0000	0001	0010	0011
0000	1010	0000	0000
0100	0101	0110	0111
0000	0000	1111	0000
1000	1001	1010	1011
0000	0000	0000	0000
1100	1101	1110	1111
0000	0101	0000	0000

Dividiendo la memoria en bloques y numerándolos:

$$\# \text{bloques}_{mem} = \frac{|\text{memoria}|}{|\text{lnea}_{cache}|}$$

$$= \frac{16 \text{ unidades}}{4 \text{ unidades/bloque}}$$

= 4 bloques → 2 bits

- 1 para el tag del bloque.
- 1 para indicar la línea de la caché.

Cache

indice→	00	01	10	11
tag↓	---	---	---	---
---	---	---	---	---
---	---	---	---	---

Línea 0
Línea 1

Correspondencia:

$$i = j \bmod m$$

i = nro. línea caché

j = nro. bloque memoria

m = # líneas en la caché

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

¿Cómo se traza la correspondencia entre las posiciones de memoria y la línea de caché?

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Dividiendo la memoria en bloques y numerándolos:

$$\# \text{bloques}_{mem} = \frac{|\text{memoria}|}{|\text{lnea}_{cache}|}$$

$$= \frac{16 \text{ unidades}}{4 \text{ unidades/bloque}}$$

= 4 bloques → 2 bits

- 1 para el tag del bloque.
- 1 para indicar la línea de la caché.

Cache

indice→ tag↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

Línea 0
Línea 1

Correspondencia:

$$i = j \bmod m$$

i = nro. línea caché

j = nro. bloque memoria

m = # líneas en la caché

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

¿Qué ocurre cuando el CPU solicita la dirección 0x6?

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Cache

indice→ tag↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

¿Qué ocurre cuando el CPU solicita la dirección 0x6?

Memoria

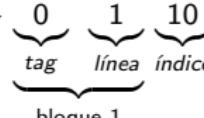
0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Los 2 primeros bits indican el nro. de bloque en memoria.

$$0x6 = 0\ 1\ 1\ 0$$

$$\text{nro. línea caché} = 1 \bmod 2 = 1$$

Tomo el primer bit de la dirección para identificar el tag y el siguiente para identificar la línea:

0x6 → 

Cache

indice→ tag↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

Línea 0
Línea 1

Como el tag no está cargado en caché se produce un **miss**. Pido a memoria toda la línea.

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

¿Qué ocurre cuando el CPU solicita la dirección 0x6?

Memoria

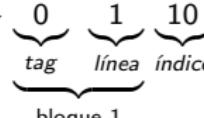
0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Los 2 primeros bits indican el nro. de bloque en memoria.

$$0x6 = 0\ 1\ 1\ 0$$

$$\text{nro. línea caché} = 1 \bmod 2 = 1$$

Tomo el primer bit de la dirección para identificar el tag y el siguiente para identificar la línea:

0x6 → 

Cache

indice→ tag↓	00	01	10	11
--	---	---	---	---
0	0000	0000	1111	0000

Línea 0
Línea 1

Como el tag no está cargado en caché se produce un **miss**. Pido a memoria toda la línea.

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Pregunta:

¿Y si ahora pide la dirección 0x1?

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Cache

indice→ tag↓	00	01	10	11
--	--	--	--	--
0	0000	0000	1111	0000

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Cache

indice→ tag↓	00	01	10	11
--	--	--	--	--
0	0000	0000	1111	0000

Línea 0

Línea 1

Pregunta:

¿Y si ahora pide la dirección 0x1?

Respuesta:

Ídem:

0x1 → 

Como el tag ya está cargado en memoria, pero la línea 0 no, se produce un **miss**. Pido a memoria toda la línea.

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Cache

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
0	0000	0000	1111	0000

Línea 0

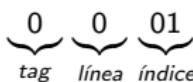
Línea 1

Pregunta:

¿Y si ahora pide la dirección 0x1?

Respuesta:

Ídem:

0x1 → 

Como el tag ya está cargado en memoria, pero la línea 0 no, se produce un **miss**. Pido a memoria toda la línea.

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Pregunta:

Memoria

Ahora, ¿qué ocurriría si el CPU pidiera la dirección 0xD?

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Cache

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
0	0000	0000	1111	0000

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Cache

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
0	0000	0000	1111	0000

Línea 0
Línea 1Pregunta:

Ahora, ¿qué ocurriría si el CPU pidiera la dirección 0xD?

Respuesta:

0xD → 
 tag línea índice

Como el tag no está cargado en caché, se produce un **miss** porque sé que esa línea no está en caché.

Pero ahora, como mi caché está llena, voy a tener que *reemplazar la línea correspondiente* en caché.

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Cache

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
1	0000	0101	0000	0000

Línea 0

Línea 1

Pregunta:

Ahora, ¿qué ocurriría si el CPU pidiera la dirección 0xD?

Respuesta:

0xD → 

Como el tag no está cargado en caché, se produce un **miss** porque sé que esa línea no está en caché.

Pero ahora, como mi caché está llena, voy a tener que *reemplazar la línea correspondiente* en caché.

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU realizara la siguiente secuencia de pedidos?

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

⋮ ⋮ ⋮

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

Cache

indice→ tag↓	00	01	10	11
--	---	---	---	---
--	---	---	---	---

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU realizara la siguiente secuencia de pedidos?

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

⋮ ⋮ ⋮

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

Cache

indice→ tag↓	00	01	10	11
--	---	---	---	---
0	0000	0000	1111	0000

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU realizará la siguiente secuencia de pedidos?

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

⋮ ⋮ ⋮

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

Cache

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
0	0000	0000	1111	0000

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU realizara la siguiente secuencia de pedidos?

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

⋮ ⋮ ⋮

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

Cache

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
1	0000	0101	0000	0000

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU realizará la siguiente secuencia de pedidos?

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

⋮ ⋮ ⋮

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

Cache

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
0	0000	0000	1111	0000

Línea 0

Línea 1

Caché de Correspondencia Directa

Caché de Correspondencia Directa (o Mapeo Directo)

Memoria

0	0000	0001	0010	0011
	0000	1010	0000	0000
1	0100	0101	0110	0111
	0000	0000	1111	0000
2	1000	1001	1010	1011
	0000	0000	0000	0000
3	1100	1101	1110	1111
	0000	0101	0000	0000

Pregunta:

Ahora, ¿qué ocurriría si el CPU realizará la siguiente secuencia de pedidos?

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

⋮ ⋮ ⋮

0x6 → 0 1 10

0x1 → 0 0 01

0xD → 1 1 01

Cache

Y así sucesivamente...

indice→ tag↓	00	01	10	11
0	0000	1010	0000	0000
0	0000	0000	1111	0000

Línea 0

Línea 1

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

0x0	0x1
0x0	0xD
0x2	0x3
0xA	0x0
0x4	0x5
0x0	0x0
0x6	0x7
0x0	0x8
0x8	0x9
0x0	0x0
0xA	0xB
0x0	0xE

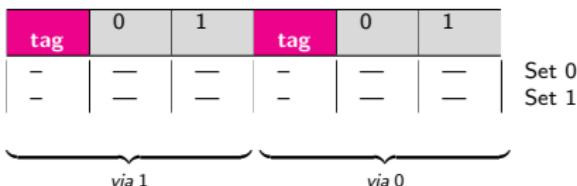
Es un tipo de caché que combina características de los dos tipos de caché que vimos previamente.

Es muy similar a la Caché de Correspondencia Directa, pero con el agregado de vías que permiten persistir las líneas de memoria en caché por más tiempo.

Vamos a ver un ejemplo de 2 vías, es extensible a N vías.

Cache

En este caso la caché se divide en conjuntos de 2 líneas (2 vías).



$$\begin{aligned} \# \text{ líneas caché} &= \\ \# \text{ conjuntos} \times \# \text{ vías} & \end{aligned}$$

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

¿Cómo se traza la correspondencia entre las posiciones de memoria y la línea de caché?

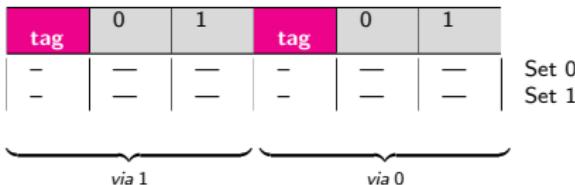
Memoria

0000	0001
0000	1101
0010	0011
1010	0000
0100	0101
0000	0000
0110	0111
0000	1000
1000	1001
0000	0000
1010	1011
0000	1110

Dividiendo la memoria en bloques y numerándolos:

$$\begin{aligned}\#\text{bloques}_{\text{mem}} &= \frac{|\text{memoria}|}{|\text{via}|} \\ &= \frac{|\text{memoria}|}{|\text{cache}|} \\ &= \frac{|\text{memoria}|}{(\#\text{conjunto} \times \#\text{vías})} \\ &= \frac{12 \text{ unidades}}{8 \text{ unidades / bloque}} \\ &= 4\end{aligned}$$

Cache



- = 6 bloques → 3 bits.
- 1 bit para el conjunto.
- 2 bits para tag.

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

¿Cómo se traza la correspondencia entre las posiciones de memoria y la línea de caché?

Dividiendo la memoria en bloques y numerándolos:

$$\# \text{bloques}_{\text{mem}} = \frac{|\text{memoria}|}{|\text{via}|}$$

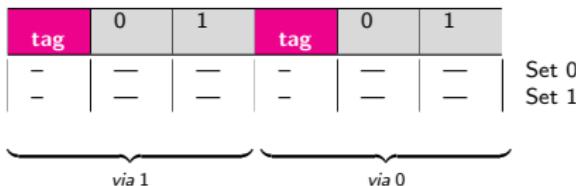
$$= \frac{|\text{memoria}|}{\frac{|\text{cache}|}{(\#\text{conjunto} \times \#\text{vias})}}$$

$$= \frac{12 \text{ unidades}}{\frac{8 \text{ unidades / bloque}}{4}}$$

= 6 bloques → 3 bits.

- 1 bit para el conjunto.
- 2 bits para tag.

Cache



Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:

Memoria

¿Qué ocurre cuando el CPU solicita la dirección 0x2?

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Cache

tag	0	1	tag	0	1	
-	--	--	-	--	--	Set 0
-	--	--	-	--	--	Set 1

The diagram illustrates the organization of a cache with two sets. Each set contains two cache blocks. Each block has a 'tag' field and two 'data' fields, labeled 'via 1' and 'via 0'. The 'via 1' fields are grouped by a brace under 'Set 0' and 'Set 1'. The 'via 0' fields are also grouped by a brace under 'Set 0' and 'Set 1'.

Caché Asociatiya por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

		0	0000	0001
		1	0000	1101
		2	0010	0011
		3	1010	0000
		4	0100	0101
		5	0000	0000
		6	0110	0111
		7	0000	1000
		8	1000	1001
		9	0000	0000
		A	1010	1011
		B	0000	1110

Cache

tag	0	1	tag	0	1
-	-	-	-	-	-
-	-	-	-	-	-

via 1
via 0

Set 0
 Set 1

Pregunta:

¿Qué ocurre cuando el CPU solicita la dirección 0x2?

Respuesta:

Tomo los dos primeros bits de la dirección para identificar el tag y el siguiente para identificar el set:

0x2 → 

Como el tag no está cargado en caché se produce un **miss**. Pido a memoria la línea entera.

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:

¿Qué ocurre cuando el CPU solicita la dirección 0x2?

Respuesta:

Tomo los dos primeros bits de la dirección para identificar el tag y el siguiente para identificar el set:

$0x2 \rightarrow \underbrace{00}_{\text{tag}} \underbrace{1}_{\text{set}} \underbrace{0}_{\text{índice}}$
bloque 1

Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Cache

tag	0	1	tag	0	1
-	--	--	-	--	--
-	--	--	00	1010	0000

{ via 1 } { via 0 }

Set 0
 Set 1

Como el tag no está cargado en caché se produce un **miss**. Pido a memoria la línea entera.

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:

Memoria

¿Y si ahora pide la dirección 0x1?

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Cache



Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

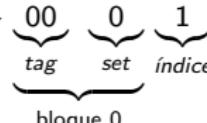
0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Pregunta:

¿Y si ahora pide la dirección 0x1?

Respuesta:

Ídem:

0x1 → 

Cache

tag	0	1	tag	0	1
-	--	--	-	--	--
-	--	--	00	1010	0000



Set 0
Set 1

Como la línea no está cargada en caché se produce un **miss**. Pido a memoria la línea entera.

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

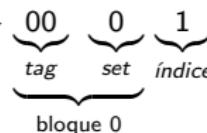
0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Pregunta:

¿Y si ahora pide la dirección 0x1?

Respuesta:

Ídem:

0x1 → 

tag set índice
 { } { }
 bloque 0

Cache

tag	0	1	tag	0	1
-	--	--	00	0000	1101
-	--	--	00	1010	0000



Set 0
Set 1

Como la línea no está cargada en caché se produce un **miss**. Pido a memoria la línea entera.

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:

Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Ahora, ¿qué ocurriría si el CPU pidiera la dirección 0xB?

Cache

tag	0	1	tag	0	1	
-	--	--	00	0000	1101	Set 0
-	--	--	00	1010	0000	Set 1



Caché Asociativa por Conjuntos de N Vías

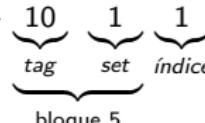
Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Ahora, ¿qué ocurriría si el CPU pidiera la dirección 0xB?

Respuesta:

0xB → 

Como el tag no está cargado en caché, se produce un **miss**.

Cache

tag	0	1	tag	0	1
-	--	--	00	0000	1101
-	--	--	00	1010	0000



 Set 0 Set 1

via 1 via 0

La dirección pedida corresponde al set 1 y como tengo espacio puedo traerla sin desalojar !

Caché Asociativa por Conjuntos de N Vías

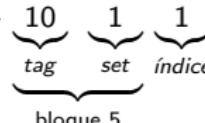
Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Ahora, ¿qué ocurriría si el CPU pidiera la dirección 0xB?

Respuesta:

0xB → 
 tag set índice

 bloq 5

Como el tag no está cargado en caché, se produce un **miss**.

Cache

tag	0	1	tag	0	1		
-	—	—	00	0000	1101		
10	0000	1110	00	1010	0000		

{ via 1 } { via 0 }

Set 0
 Set 1

La dirección pedida corresponde al set 1 y como tengo espacio puedo traerla sin desalojar !

Caché Asociatiya por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

			0000	0001
			0000	1101
			0010	0011
			1010	0000
			0100	0101
			0000	0000
			0110	0111
			0000	1000
			1000	1001
			0000	0000
			1010	1011
			0000	1110
5	4	3	2	1
				0

Cuando el CPU solicita nuevamente la dirección 0x2:

Tomo los dos primeros bits de la dirección para identificar el tag y el siguiente para identificar el set:

0x2 → 

Como el tag está cargado en caché se produce un **hit**.

tag	0	1	tag	0	1
-	—	—	00	0000	1101
10	0000	1110	00	1010	0000

{ via 1 } { via 0 }

Caché Asociativa por Conjuntos de N Vías

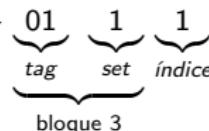
Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
	1010	1011
5	0000	1110

¿Y si después el CPU pide la dirección 0x7?

Respuesta:

0x7 → 

Cache

tag	0	1	tag	0	1
-	—	—	00	0000	1101
10	0000	1110	00	1010	0000

Set 0
Set 1

Set 0

Set 1

via 0

via 1

Como el tag no está cargado en caché, se produce un **miss**.

De nuevo, la dirección pedida corresponde al set 1. Sin embargo, ya tengo todas las vías de mi set completas. ¿Qué hago?

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

			0000	0001
			0000	1101
		0	0010	0011
		1	1010	0000
	0	2	0100	0101
	1	3	0000	0000
	2	4	0110	0111
	3	5	0000	1000
	4		1000	1001
	5		0000	0000
			1010	1011
			0000	1110

Cache

tag	0	1	tag	0	1
-	—	—	00	0000	1101
10	0000	1110	00	1010	0000

via 1
via 0

Pregunta:

¿Y si después el CPU pide la dirección 0x7?

Respuesta:

0x7 → 

Como el tag no está cargado en caché, se produce un **miss**.

De nuevo, la dirección pedida corresponde al set 1. Sin embargo, ya tengo todas las vías de mi set completas. ¿Qué hago?

Desalojo la línea de alguna de las vías del set 1 usando alguna de las políticas que vimos antes. Por ejemplo: LFU.

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Cache

tag	0	1	tag	0	1
-	—	—	00	0000	1101
01	0000	1000	00	1010	0000

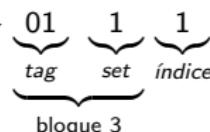
↴ vía 1 ↴ vía 0

Set 0
 Set 1

Pregunta:

¿Y si después el CPU pide la dirección 0x7?

Respuesta:

0x7 → 

Como el tag no está cargado en caché, se produce un **miss**.

De nuevo, la dirección pedida corresponde al set 1. Sin embargo, ya tengo todas las vías de mi set completas. ¿Qué hago?

Desalojo la línea de alguna de las vías del set 1 usando alguna de las políticas que vimos antes. Por ejemplo: LFU.

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Entonces, ¿qué ocurriría si el CPU hiciera la siguiente secuencia de pedidos?

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

⋮

⋮

⋮

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

Cache

tag	0	1	tag	0	1	
-	—	—	—	—	—	Set 0
-	—	—	00	1010	0000	Set 1

{ via 1 } { via 0 }

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Entonces, ¿qué ocurriría si el CPU hiciera la siguiente secuencia de pedidos?

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

⋮

⋮

⋮

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

Cache

tag	0	1	tag	0	1		
-	—	—	00	0000	1101	Set 0	
-	—	—	00	1010	0000	Set 1	



Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Entonces, ¿qué ocurriría si el CPU hiciera la siguiente secuencia de pedidos?

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

⋮

⋮

⋮

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

Cache

tag	0	1	tag	0	1	
—	—	—	00	0000	1101	Set 0
10	0000	1110	00	1010	0000	Set 1

{ via 1 } { via 0 }

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Entonces, ¿qué ocurriría si el CPU hiciera la siguiente secuencia de pedidos?

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

⋮

⋮

⋮

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

Cache

tag	0	1	tag	0	1	
—	—	—	00	0000	1101	Set 0
01	0000	1000	00	1010	0000	Set 1

{ via 1 } { via 0 }

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Entonces, ¿qué ocurriría si el CPU hiciera la siguiente secuencia de pedidos?

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

⋮

⋮

⋮

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

Cache

tag	0	1	tag	0	1	
—	—	—	00	0000	1101	Set 0
10	0000	1110	00	1010	0000	Set 1

{ via 1 } { via 0 }

Caché Asociativa por Conjuntos de N Vías

Caché Asociativa por Conjuntos de N Vías

Pregunta:

Memoria

0	0000	0001
	0000	1101
1	0010	0011
	1010	0000
2	0100	0101
	0000	0000
3	0110	0111
	0000	1000
4	1000	1001
	0000	0000
5	1010	1011
	0000	1110

Entonces, ¿qué ocurriría si el CPU hiciera la siguiente secuencia de pedidos?

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

⋮

⋮

⋮

0x2 → 0010

0x1 → 0001

0xB → 1011

0x2 → 0010

0x7 → 0111

Cache

tag	0	1	tag	0	1
—	—	—	00	0000	1101
01	0000	1000	00	1010	0000

Set 0
Set 1



Y así sucesivamente...

Ejercicio 1 - Correspondencia directa

Ejercicio 1 - Correspondencia directa

Memoria Principal: 2^{20} bytes, direccionable a byte.

Cache: 32 líneas de 16 bytes cada una.

Responder:

- ① ¿Cuánto mide un bloque para esta configuración?
- ② ¿Cuántos bloques entran en memoria principal?
- ③ ¿Cuántos bits son necesarios para identificar una línea de memoria en la caché?
- ④ ¿Cómo puedo saber si está cargada la línea donde se encuentra la palabra referida por la dirección C34A6?

Ejercicio 1 - Correspondencia directa

1) ¿Cuánto mide un bloque para esta configuración?

Depende del espacio disponible en una línea de caché, en este caso el enunciado lo indica: 16 bytes.

2) ¿Cuántos bloques entran en memoria principal?

$$\frac{\text{capacidad memoria}}{\text{tamaño bloque}} = \frac{2^{20}B}{2^4B/\text{bloque}} = 2^{16} \text{ bloque}$$

3) ¿Cuántos bits son necesarios para identificar una línea de memoria en la caché?

De los 20 bits que son necesarios para direccionar a byte toda la memoria hay que utilizar:

4 bits para mantener el índice a byte, quedan 16 bits para identificar el bloque, de los cuales:

5 bits determinan la línea ($2^5 = 32$ líneas de caché) y 11 bits para etiquetar los bloques.

④ ¿Cómo puedo saber si está cargada la línea donde se encuentra la palabra referida por la dirección C34A6?

Primero me fijo cuánto mide cada campo de una dirección de memoria para esta configuración de cache: **tag 11 bits, línea 5 bits, índice 4 bits**. Después, paso la dirección a binario para saber el valor de los campos correspondientes a esa dirección.

- La dirección en binario:

<i>C</i>	3	4	<i>A</i>	6
1100	0011	0100	1010	0110

- Agrupada según los campos **tag, línea e índice**:

61A	A	6
110 0001	1010 0	1010 0110

Finalmente, me tengo que fijar si en el lugar reservado para las líneas número 0xA, está cargada la línea correspondiente al bloque número 0x61A (el tag 0x61A); si es así, la línea correspondiente a la dirección pedida está cargada en cache.

Ejercicio 2 - Asociativa por conjuntos

Ejercicio 2 - Asociativa por conjuntos

Memoria Principal: 1 MB, direccionable a byte.

Cache: 32 líneas de 64 bytes cada una, 2 vías.

Responder:

- ① ¿Cuánto mide un bloque para esta configuración?
- ② ¿Cuántos bloques entran en memoria principal?
- ③ ¿Cuántos bits son necesarios para identificar una línea de memoria en la caché?
- ④ ¿Cómo puedo saber si está cargada la línea donde se encuentra la palabra referida por la dirección C34A6?

① ¿Cuánto mide un bloque para esta configuración?

Depende del espacio disponible en una vía de caché, en este caso puede obtenerse del enunciado:

$$\frac{64 \text{ bytes}}{2 \text{ vías}} = 32 \text{ bytes} = 2^5 \text{ bytes.}$$

② ¿Cuántos bloques entran en memoria principal?

$$1 \text{ MB} = 1024 \text{ KB} = 2^{10} \times 2^{10} \text{ bytes} = 2^{20} \text{ bytes.}$$

$$\frac{2^{20}}{32} = \frac{2^{20}}{2^5} = 2^{15} \text{ bloques.}$$

③ ¿Cuántos bits son necesarios para identificar una línea de memoria en la caché?

De los 20 bits que son necesarios para direccionar a byte toda la memoria hay que utilizar:

5 bits para mantener el índice a byte en una vía de un conjunto, quedan 15 bits para identificar el bloque, de los cuales:

5 bits determinan el conjunto ($2^5 = 32$ líneas de caché) y 10 bits para etiquetar los bloques.

Ejercicio 2 - Asociativa por conjuntos

4) ¿Cómo puedo saber si está cargada la línea donde se encuentra la palabra referida por la dirección C34A6?

Primero me fijo cuánto mide cada campo de una dirección de memoria para esta configuración de cache: **tag 10 bits, conjunto 5 bits, índice 5 bits**. Después, paso la dirección a binario para saber el valor de los campos correspondientes a esa dirección.

- La dirección en binario:

<i>C</i>	<i>3</i>	<i>4</i>	<i>A</i>	<i>6</i>
1100	0011	0100	1010	0110

- Agrupada según los campos **tag, conjunto e índice**:

<i>61A</i>	<i>5</i>	<i>06</i>
110 0001	1010	0101 0 0110

Finalmente, me tengo que fijar si en el lugar reservado para las líneas número 0x5 está cargada la línea correspondiente al bloque número 0x61A, en cualquiera de los dos espacios del conjunto; si es así, la línea correspondiente a la dirección pedida está cargada en cache.

Ejercicio 3

Se estudia agregar una memoria cache a una computadora cuyas palabras y direcciones de memoria son de 16 *bits*, y que trabaja con **direcccionamiento a byte**.

Hasta el momento se barajan dos opciones:

- Una caché asociativa por conjuntos de 4 vías
- Una de correspondencia directa (o *mapeo directo*)

Cada cache puede almacenar hasta **1024 bytes** de información (sin contar el espacio necesario para los *tags*) y se organiza en 64 líneas en **total**.

Ejercicio 3

- a) Indique cómo se distribuyen los bits de una dirección de memoria en los campos correspondientes para cada una de las caches mencionadas.

Ejercicio 3

- a) Indique cómo se distribuyen los bits de una dirección de memoria en los campos correspondientes para cada una de las caches mencionadas.

- b) Se conoce que este fragmento de código insume gran parte del tiempo de cómputo.

Junto a cada instrucción se indica el acceso a memoria necesario para el *fetch* de la instrucción, no así los necesarios para acceder a los datos.

	Código	Pedido a memoria del <i>fetch</i>
	MOV R6, 0x10	0x9C13
	MOV R2, 0x801A	0x9C15, 0x9C17
mejillón:	ADD R5, [R2]	0x9C19
	ADD R5, [R2 + 0x15]	0x9C1B
	SUB R6, 0x01	0x9C1D
	CMP R6, 0x00	0x9C1F
	JNE mejillón	0x9C21

Simule los accesos a memoria que realiza este programa hasta la comparación (inclusive) utilizando la cache de **mapeo directo**. **Tenga en cuenta los accesos a datos, que no fueron detallados en la tabla.** Indique en cada paso el contenido de la cache, y cuando corresponda detalle si se producen *hits*, *misses*, desalojos (señalando la línea desalojada) y/o accesos desalineados. ¿Cuál es el *hit rate* de esta ejecución parcial?

Ejercicio 3

- a) Indique cómo se distribuyen los bits de una dirección de memoria en los campos correspondientes para cada una de las caches mencionadas.

- b) Se conoce que este fragmento de código insume gran parte del tiempo de cómputo.

Junto a cada instrucción se indica el acceso a memoria necesario para el *fetch* de la instrucción, no así los necesarios para acceder a los datos.

	Código	Pedido a memoria del <i>fetch</i>
	MOV R6, 0x10	0x9C13
	MOV R2, 0x801A	0x9C15, 0x9C17
mejillón:	ADD R5, [R2]	0x9C19
	ADD R5, [R2 + 0x15]	0x9C1B
	SUB R6, 0x01	0x9C1D
	CMP R6, 0x00	0x9C1F
	JNE mejillón	0x9C21

Simule los accesos a memoria que realiza este programa hasta la comparación (inclusive) utilizando la cache de **mapeo directo**. **Tenga en cuenta los accesos a datos, que no fueron detallados en la tabla**. Indique en cada paso el contenido de la cache, y cuando corresponda detalle si se producen *hits*, *misses*, desalojos (señalando la línea desalojada) y/o accesos desalineados. ¿Cuál es el *hit rate* de esta ejecución parcial?

- c) ¿Cuál de las dos caches es más conveniente para la ejecución completa del fragmento del programa presentado?

Ejercicio 3

a) Correspondencia directa

- 1024 bytes distribuidos en 64 líneas, entonces 16 bytes por línea.
- Direcciones de 16 bits, 2^{16} bytes de memoria
- Cantidad de bloques de memoria:
$$\frac{2^{16}}{2^4} = 2^{12}$$
 bloques.
- El direccionamiento es a byte y hay que indexar 16 bytes, así que tendría que reservar los 4 bits de orden más bajo para el índice.
- Hay 64 líneas de caché que identificar para hacer la correspondencia, son necesarios 6 bits.
- De la dirección de 16 bits, los 6 bits de orden más alto se utilizan para la etiqueta.

|| tag (6 bits) || nro. de línea (6 bits) || indice (4 bits) ||

Ejercicio 3

(a) Asociativa por conjuntos

- # líneas caché = # conjuntos x # vías
$$\# \text{ conjuntos} = \frac{64 \text{ líneas}}{4 \text{ vías}} = 16 \text{ conjuntos}$$
- 1024 bytes distribuidos en 64 líneas, entonces 16 bytes por línea.
- 16 bytes por línea de caché de 4 vías. Entonces $\frac{16 \text{ bytes}}{4} = 4 \text{ bytes por vía.}$
- Direcciones de 16 bits, 2^{16} bytes de memoria
- Cantidad de bloques de memoria:
$$\frac{2^{16}}{2^2} = 2^{14} \text{ bloques.}$$
- El direccionamiento es a byte y hay que indexar 4 bytes, así que tendría que reservar los 2 bits de orden más bajo para el índice.
- Hay 16 conjuntos que identificar para hacer la correspondencia, son necesarios 4 bits.
- De la dirección de 16 bits, los 10 bits de orden más alto se utilizan para la etiqueta.

|| tag (10 bits) || nro. de conjunto (4 bits) || indice (2 bits) ||

Ejercicio 3

(b) Accesos a memoria - Acceso directo

Dirección	Tag	Línea	Índice	Resultado	Estado Cache	Notas
0x9C13 <u>27</u> <u>01</u> <u>3</u> 10 0111 00 0001 0011	27	1	3	Miss	{1: 27}	cargué 1:27
0x9C15 <u>27</u> <u>01</u> <u>5</u> 10 0111 00 0001 0101	27	1	5	Hit	{1: 27}	
0x9C17 <u>27</u> <u>01</u> <u>7</u> 10 0111 00 0001 0111	27	1	7	Hit	{1: 27}	
0x9C19 <u>27</u> <u>01</u> <u>9</u> 10 0111 00 0001 1001	27	1	9	Hit	{1: 27}	
0x801A <u>20</u> <u>01</u> <u>A</u> 10 0000 00 0001 1010	20	1	A	Miss	{1: 20}	desalojé 1: 27; cargué 1:20
0x9C1B <u>27</u> <u>01</u> <u>B</u> 10 0111 00 0001 1011	27	1	B	Miss	{1: 27}	desalojé 1: 20; cargué 1:27
0x802F <u>20</u> <u>02</u> <u>F</u> 10 0000 00 0010 1111	20	2	F	Miss	{1: 27} {2: 20} {3: 20}	acc. desalineado; cargué 2: 20; cargué 3: 20
0x9C1D <u>27</u> <u>01</u> <u>D</u> 10 0111 00 0001 1101	27	1	D	Hit		
0x9C1F <u>27</u> <u>01</u> <u>F</u> 10 0111 00 0001 1111	27	1	F	Miss	{1: 27} {2: 27} {3: 20}	acc. desalineado; desalojé 2:20; cargué 2: 27

$$\text{Hit rate} = \frac{4}{9} \approx 0.44\%$$

(c) Comparacion - Tarea

Idea de solución:

Para este bloque de código sería más conveniente la caché asociativa por conjuntos de 4 vías, ya que se eliminarían los desalojos gracias al más alto nivel de asociatividad.

Ejercicio 3

Acceso Desalineado

Un acceso desalineado es una situación especial que se da cuando se realiza un acceso a memoria en una posición cuya palabra sobrepasa el límite de la línea de caché. En este caso, para satisfacer el último pedido de lectura desde caché, será necesario cargar dos líneas de caché en vez de una.

Un acceso desalineado es considerado un **Hit** si ambas líneas ya están en la caché. Si falta una o las dos es considerado un único **Miss**.

Ejercicio 3

Por qué funciona la *caché* (localidad espacial)

```
int arreglo[10000] = { /* ... */ };
int contador = 0;

for(int i = 0; i < 10000; i++)
{
    if (arreglo[i] > 4)
        contador++;
}
```

Ejercicio 3

Por qué funciona la *caché* (localidad temporal)

```
int arreglo[10000] = { /* ... */ };
int contador = 0;

for(int i = 0; i < 10000; i++)
{
    if (arreglo[i] > 4)
        contador++;
}
```

Fin

Preguntas?

