

Tema 5: Jerarquía de memoria — Caché

Marzo de 2011

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Agenda

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Introducción

- Los programadores quieren memorias grandes y rápidas.
 - Problema: no existe hoy en día una memoria a la vez grande y rápida a un coste razonable.
 - **Crearemos la ilusión** de una memoria de las características deseadas **combinando** distintos tipos de memoria → **Jerarquía de memoria**.
- Analogía: mesa en la biblioteca
 - Una buena selección de libros en la mesa (memoria pequeña y rápida).
 - Gran probabilidad de encontrar lo buscado sin ir a la estantería.
 - Visión de memoria grande (biblioteca) a la que se accede rápido (tiempo de coger un libro de la mesa).
- Objetivo: detectar zonas del programa con más probabilidad de ser accedidas y llevarlas a la memoria más rápida

Agenda

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Principio de localidad

En un momento concreto, los programas acceden a una parte relativamente pequeña de su espacio de direcciones.

Dos tipos:

- **Localidad temporal:** si se consulta un dato, seguramente será consultado próximamente.
 - Un libro de la mesa será consultado varias veces.
- **Localidad espacial:** si se consulta un dato, seguramente otros datos cercanos a él serán consultados próximamente.
 - Traemos un libro del estante → los libros que están próximos versarán sobre el mismo tema.

Principio de localidad

- **Localidad temporal de un programa.**

- Bucles: datos e instrucciones.

```
for i:=1 to 1000 do
  a := a + 2;
  b := b * 3;
  c := c div 2;
end;
```

- **Localidad espacial de un programa.**

- Instrucciones: ejecución secuencial.
- Datos: estructuras de datos como las tablas y arrays.

```
for i := 1 to 1000 do begin
  a[i] := 1;
  b[i] := 2;
end;
```

Agenda

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Conceptos generales

- El objetivo de organizar la memoria de un ordenador como una jerarquía de memorias es aprovechar el principio de localidad.
- **Consideración:** las memorias más rápidas son las más caras por bit y por tanto suelen ser más pequeñas.

Tabla: Tiempos de acceso y precios de las distintas tecnologías de memoria más comunes (año 2009).

Tecnología	Tiempo de acceso típico	€ por MB
SRAM	1 ns	20 €
SDRAM	5 ns	0,01 €
Disco magnético	8.500.000 ns	0,0001 €

- **Objetivo:** mucha memoria + tecnología barata + alta velocidad

Conceptos generales

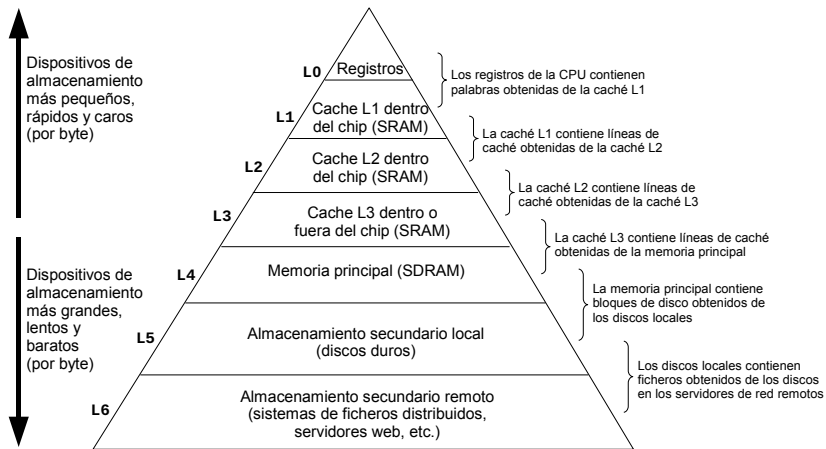


Figura: Estructura básica de la jerarquía de memoria.

Conceptos generales

- **Sacar provecho de la localidad temporal de un programa:** mantener los datos accedidos más recientemente cerca del procesador.
- **Sacar provecho de la localidad espacial de un programa:** moviendo bloques de varios datos contiguos a los niveles próximos al procesador.

Conceptos generales

- La jerarquía consta de múltiples niveles.
- Los datos sólo se copian entre niveles adyacentes.
 - Nivel superior: más cercano al procesador, más caro, más rápido.
 - Nivel inferior: más lejano al procesador, más barato, más lento.
- **Bloque**: unidad mínima de información.
- **Acierto** (*hit*): el elemento requerido por la CPU está en el nivel superior.
- **Fallo** (*miss*): el elemento requerido por la CPU **no** está en el nivel superior \Rightarrow Se accede al nivel inferior para recuperar el bloque que contiene al elemento requerido.

Conceptos generales

- **Tasa de aciertos** (*hit rate*): porcentaje de accesos a memoria encontrados en el nivel superior.
- **Tasa de fallos** (*miss rate*): $(1 - \text{tasa de aciertos})$. Porcentaje de accesos a memoria no encontrados en el nivel superior.
- **Tiempo de acierto**: tiempo necesario acceder al nivel superior de la memoria, incluyendo el tiempo para determinar si el acceso es un acierto o un fallo.
- **Penalización por fallo**: es el tiempo necesario para reemplazar un bloque del nivel superior por el correspondiente bloque del nivel inferior, más el tiempo de suministrar este bloque al procesador.

Conceptos generales

Nivel superior más pequeño y rápido



Tiempo de acierto \ll Penalización por fallo

Los programas pasan mucho tiempo accediendo a memoria (leer código instrucción y posibilidad de otro acceso a memoria para leer/escribir operandos)



El sistema de memoria es un factor de gran importancia para determinar el rendimiento general de una máquina

Agenda

- 1 **Introducción**
 - Principio de localidad
 - Conceptos generales
- 2 **Memoria caché**
 - Memoria caché de correspondencia directa
 - Rendimiento de la caché
 - Memoria caché asociativa por conjuntos
- 3 **Tratamiento de los fallos de caché**
 - Tratamiento de los fallos de lectura
 - Tratamiento de los fallos de escritura
- 4 **Memorias caché multinivel**
- 5 **Características de la memoria caché en algunos sistemas actuales**

Memoria caché

- Caché fue el término escogido para representar el nivel de la jerarquía de memorias entre la CPU y la memoria principal.
- Ejemplo: caché muy simple y bloques de una palabra.

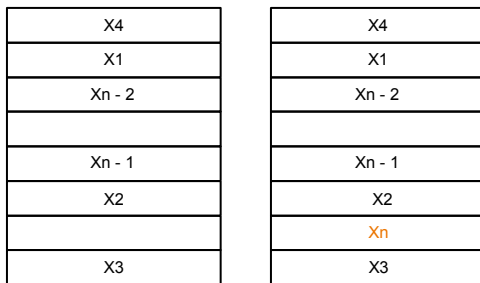
a. Antes de la referencia a X_n b. Después de la referencia a X_n

Figura: Una memoria caché antes y después de la referencia a la palabra X_n .

Memoria caché

- Dos preguntas:
 - ¿Cómo se sabe si un dato está en la caché?
 - Y si está, ¿cómo se localiza?
- Respuesta:
 - Dependerá de la estructura y funcionamiento de la memoria caché.

Agenda

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Memoria caché de correspondencia directa

Cada dirección de memoria se corresponde con una única posible posición en la caché

posición en caché = (dirección de bloque en memoria)
MODULO (n° de posiciones de la caché)

Bloques monopalabra (bloque = 1 palabra)

posición en caché = (dirección de palabra en memoria)
MODULO (n° de posiciones de la caché)

Correspondencia directa con bloques monopalabra

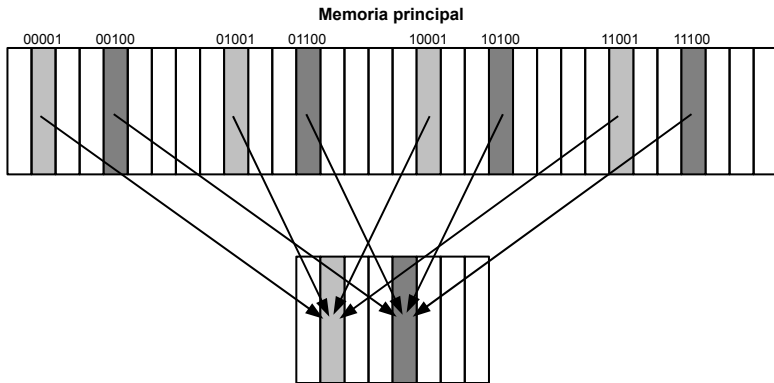
dirección palabra = (dirección byte) DIV (bytes por palabra)

Tabla: Relación entre las direcciones de byte de memoria y las direcciones de palabra para direcciones de 7 bits y palabras de 4 bytes (32 bits).

Dir. de palabra	Dir. de byte	Byte de memoria
00000	0000000	
	0000001	
	0000010	
	0000011	
00001	0000100	
	0000101	
	0000110	
	0000111	
00010	0001000	
	0001001	
	0001010	
	0001011	
00011	0001100	
	0001101	
	0001110	
	0001111	
...

Correspondencia directa con bloques monopalabra

- Ejemplo: caché de correspondencia directa con 8 posiciones. Memoria principal de 32 palabras.
- Una palabra de dirección **X** se lleva a caché a la posición **X MOD 8**: los 3 ($\log_2 8$) bits de menor peso servirán de índice de la caché.



Correspondencia directa con bloques monopalabra

- Cada posición de la caché puede contener los datos de unas cuantas direcciones de memoria.



- ¿Cómo se sabe si una palabra solicitada por el procesador es la que está en caché en esa posición o es otra la que ocupa su puesto?

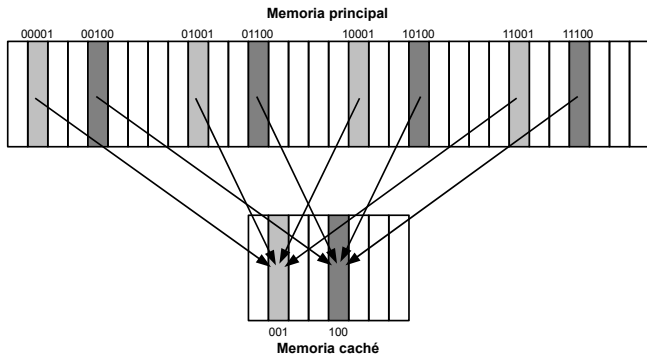


Etiquetas (*tags*): información para distinguir cuál de las posibles palabras que pueden ocupar una posición de caché es la que la ocupa en un momento dado.

- La etiqueta más simple está formada por aquellos bits de la dirección en los que difieren las diferentes palabras que pueden ocupar una posición.

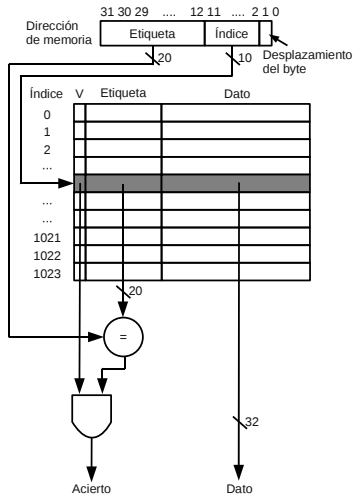
Correspondencia directa con bloques monopalabra

- En el ejemplo anterior, a las palabras de dirección $00001)_2$, $01001)_2$, $10001)_2$ y $11001)_2$, les corresponderá la posición de caché $001)_2$ tomando el valor de sus 3 bits menos significativos. **Etiqueta:** los restantes 2 bits más significativos: $00)_2$, $01)_2$, $10)_2$ y $11)_2$.



Correspondencia directa con bloques monopalabra

- **Bit de validez:** indica si una posición de caché contiene datos válidos.



Correspondencia directa con bloques monopalabra

- Continuando con el ejemplo de la memoria principal de 32 palabras y la caché de 8 palabras:

Tabla: Secuencia de ocho peticiones de acceso a memoria a una caché de correspondencia directa de 8 posiciones de un byte cada una.

Peticiones	Dir) ₁₀	Dir) ₂	Posición caché	Etiqueta caché	Acierto o Fallo
1°	22	10110	110	10	Fallo
2°	26	11010	010	11	Fallo
3°	22	10110	110	10	Acierto
4°	26	11010	010	11	Acierto
5°	16	10000	000	10	Fallo
6°	4	00100	100	00	Fallo
7°	16	10000	000	10	Acierto
8°	18	10010	010	10	Fallo

Correspondencia directa con bloques monopalabra

Tabla: Detalles de los contenidos de una caché durante ocho peticiones de palabras después de cada petición que provoca fallo de caché (1/3).

Índice	V	Etiqueta	Dato	Índice	V	Etiqueta	Dato
000	N			000	N		
001	N			001	N		
010	N			010	N		
011	N			011	N		
100	N			100	N		
101	N			101	N		
110	N			110	S	10	M(10110)
111	N			111	N		
a) Estado inicial de la caché después de arrancar				b) Después de tratar el fallo de la dirección de memoria (10110)			

Correspondencia directa con bloques monopalabra

Tabla: Detalles de los contenidos de una caché durante ocho peticiones de palabras después de cada petición que provoca fallo de caché (2/3).

Índice	V	Etiqueta	Dato	Índice	V	Etiqueta	Dato
000	N			000	S	10	M(10000)
001	N			001	N		
010	S	11	M(11010)	010	S	11	M(11010)
011	N			011	N		
100	N			100	N		
101	N			101	N		
110	S	10	M(10110)	110	S	10	M(10110)
111	N			111	N		
c) Después de tratar el fallo de la dirección de memoria (11010)				d) Después de tratar el fallo de la dirección de memoria (10000)			

Correspondencia directa con bloques monopalabra

Tabla: Detalles de los contenidos de una caché durante ocho peticiones de palabras después de cada petición que provoca fallo de caché (3/3).

Índice	V	Etiqueta	Dato	Índice	V	Etiqueta	Dato
000	S	10	M(10000)	000	S	10	M(10000)
001	N			001	N		
010	S	11	M(11010)	010	S	10	M(10010)
011	N			011	N		
100	S	00	M(00100)	100	S	00	M(00100)
101	N			101	N		
110	S	10	M(10110)	110	S	10	M(10110)
111	N			111	N		
e) Después de tratar el fallo de la dirección de memoria (00100)				f) Después de tratar el fallo de la dirección de memoria (10010)			

Tamaño de una caché de correspondencia directa

Tamaño total de una caché (en bits) = n° de posiciones \times
(n° bits control + tamaño etiqueta + tamaño bloque)

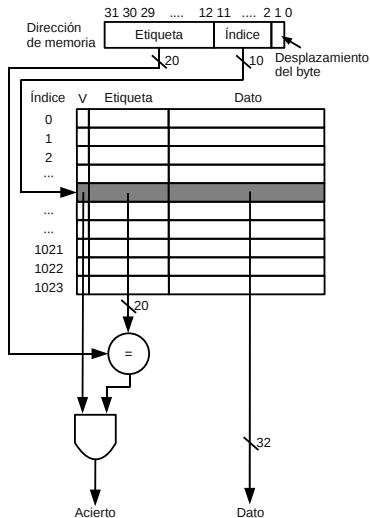
donde, para una caché de correspondencia directa con bloques monopalabra:

- Tamaño bloque = (n° de palabras por bloque) \times
(n° de bits por palabra) = $1 \times (n^{\circ}$ de bits por palabra).
- Tamaño etiqueta = (tamaño dirección) – (n° bits para índice) –
(n° bits para el desplazamiento de byte).
- N° bits para índice = $\log_2 (n^{\circ}$ de posiciones).
- N° bits para el desplazamiento de byte =
 $\log_2 (n^{\circ}$ de bytes por palabra).

Tamaño de una caché de correspondencia directa

- Tamaño total de una caché, según el ejemplo:
 - N° bits para desplazamiento de byte = $\log_2(4) = 2$ bits.
 - N° bits para índice = $\log_2(2^{10}) = 10$ bits.
 - Tamaño etiqueta = $(32 - 10 - 2) = 20$ bits.
 - Tamaño bloque = 1×32 bits.
- Por lo tanto:

$$N^{\circ} \text{ total de bits} = 2^{10} \times (1 + 20 + 32) = 54272 \text{ bits} = 6784 \text{ bytes.}$$



Correspondencia directa con bloques multipalabra

- La **localidad espacial** se da de forma natural en los programas. Conviene aprovecharla.
- **Bloques multipalabra**: tras un fallo, se traerá a caché la palabra buscada y un grupo de palabras de direcciones adyacentes a ella y que, por tanto, tendrán una alta probabilidad de ser referenciadas en breve.

Posición en caché = (dirección de bloque en memoria)
MODULO (número de bloques de la caché)

dirección de bloque en memoria = (dirección de palabra en memoria)
DIV (número de palabras por bloque)

Correspondencia directa con bloques multipalabra

Tabla: Relación entre direcciones de byte, de palabra y de bloque para direcciones de 7 bits, palabras de 4 bytes (32 bits) y bloques de 2 palabras (64 bits).

Dir. de bloque	Dir. de palabra	Dir. de byte	Byte de memoria
0000	00000	0000000	
		0000001	
		0000010	
		0000011	
	00001	0000100	
		0000101	
		0000110	
		0000111	
0001	00010	0001000	
		0001001	
		0001010	
		0001011	
	00011	0001100	
		0001101	
		0001110	
		0001111	
...

Correspondencia directa con bloques multipalabra

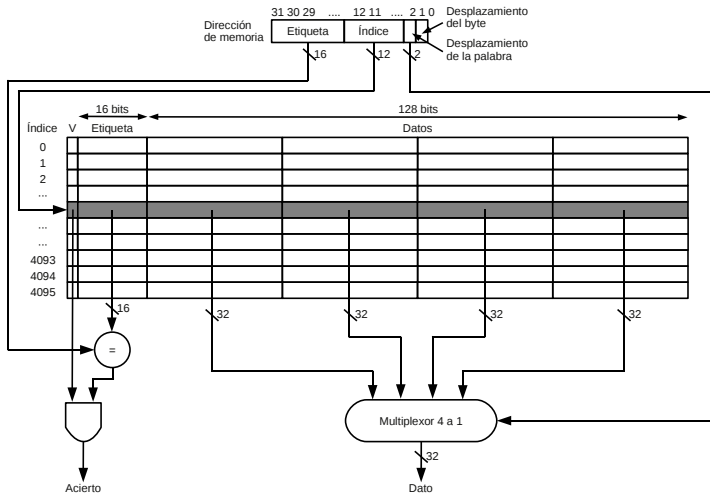


Figura: Memoria caché de 64 KB con bloques de cuatro palabras y palabras de 32 bits.

Correspondencia directa con bloques multipalabra

- La **dirección de un dato solicitado por el procesador** se descompone en:
 - **Desplazamiento del byte dentro de la palabra** (*byte offset*): los 2 bits menos significativos (como siempre, el número de bits del campo desplazamiento de byte será igual a \log_2 del número de bytes de la palabra).
 - **Dirección de palabra**: los 30 bits más significativos.
- Como el dato solicitado es una palabra completa, los 2 bits del campo *desplazamiento de byte* se desprecian directamente.

Correspondencia directa con bloques multipalabra

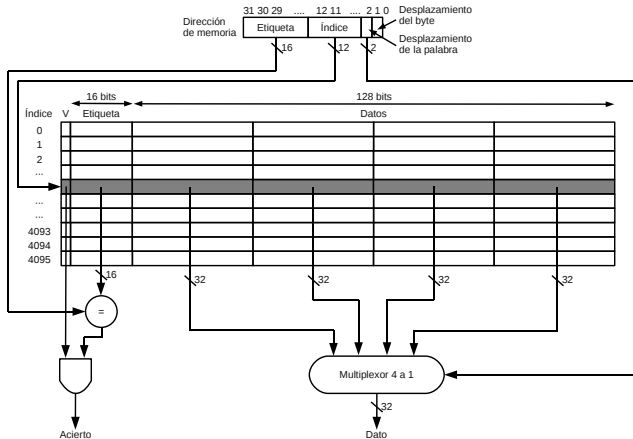


Figura: Memoria caché de 64 KB con bloques de cuatro palabras y palabras de 32 bits.

Correspondencia directa con bloques multipalabra

La **dirección de palabra** se descompone en:

- **Desplazamiento de la palabra dentro del bloque** (*word offset*): los 2 bits menos significativos (el número de bits de este campo será igual a \log_2 del número de palabras del bloque, que en este ejemplo es 4).
- **Dirección de bloque**: los restantes 28 bits más significativos.

Correspondencia directa con bloques multipalabra

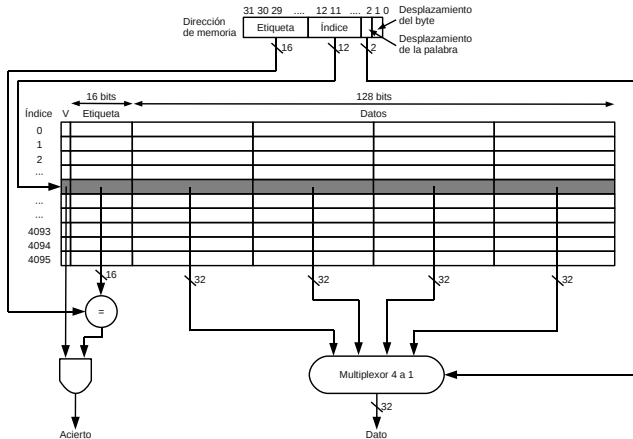


Figura: Memoria caché de 64 KB con bloques de cuatro palabras y palabras de 32 bits.

Correspondencia directa con bloques multipalabra

La **dirección de bloque** es la que se utiliza para acceder a la caché:

- **Posición en caché** (*index*): los 12 bits menos significativos (la posición en caché la marcarán los \log_2 (número de posiciones de caché) bits menos significativos de la dirección de bloque).
- **Etiqueta** (*tag*): los restantes 16 bits se deben comparar con la etiqueta existente en la posición accedida para saber si el dato que buscamos está o no en caché.

Si el dato está en caché, se utiliza el **desplazamiento de la palabra dentro del bloque** para seleccionar esta palabra en el multiplexor de salida.

Correspondencia directa con bloques multipalabra

Tabla: Uso de las direcciones de memoria para el acceso a una caché de 64 KB organizada en 2^{12} bloques de 4 palabras de 4 bytes cada una (16 bytes por bloque).

Dirección del dato solicitado por el procesador			
32 bits			
Dirección de palabra			Desplazamiento de byte
30 bits			2 bits
Dirección de bloque		Desplazamiento de palabra	Desplazamiento de byte
28 bits		2bits	2 bits
Etiqueta	Posición en caché	Desplazamiento de palabra	Desplazamiento de byte
16 bits	12 bits	2bits	2 bits

Tamaño de bloque

- En general, nos interesa un tamaño de bloque grande, pero ...
- **Problema 1:** tamaño de bloque demasiado grande provoca:
 - Número de bloques que pueden estar en caché disminuye.
 - Localidad espacial entre las palabras del bloque desciende.

- **Consecuencia 1:** la tasa de fallos deja de disminuir, incluso aumenta.

- **Problema 2:** aumenta la penalización por fallo.

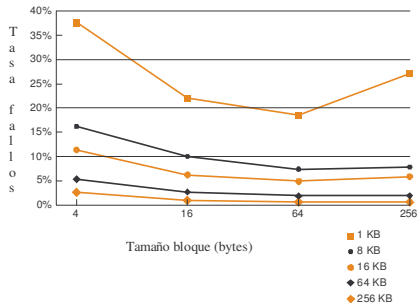


Figura: Tasa de fallos respecto al tamaño de bloque en cachés de diferentes tamaños.

Agenda

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Rendimiento de la caché

Tiempo total de un programa

=

(Ciclos de ejecución de la CPU + ciclos de bloqueo por memoria)

×

Tiempo de ciclo

Ciclos de bloqueo por Memoria

=

Nº accesos a memoria en el programa × tasa de fallos ×
penalización por fallo

Rendimiento de la caché

Ejemplo:

- Programa P:
 - $TF_{inst} = 2\%$.
 - 36% de las instrucciones acceden a algún dato de memoria.
 - $TF_{dat} = 4\%$.
- Máquina M:
 - Frecuencia del reloj de 50 Mhz.
 - $CPI_{ideal} = 2$ ciclos.
 - $PF = 40$ ciclos.

¿Cuánto más rápida sería una máquina con una caché perfecta (sin fallos)?

Rendimiento de la caché

Respuesta:

- $CPI_{real} = CPI_{ideal} + \text{ciclos bloqueo por fallo de instrucción} + \text{ciclos bloqueo por fallo de datos}.$
- $\text{Ciclos bloqueo por fallo de instrucción} = TF_{inst} * PF = 2\% \times 40 = 0,80 \text{ ciclos}.$
- $\text{Ciclos bloqueo por fallo de datos} = \text{tasa de instrucciones con acceso a memoria} \times TF_{dat} \times PF = 36\% \times 4\% \times 40 = 0,576 \text{ ciclos}.$
- $CPI_{real} = 2 + 0,80 + 0,576 = 3,376 \text{ ciclos}.$
- Con una caché perfecta: $CPI_{real} = CPI_{ideal} = 2 \text{ ciclos}.$

Resultado:

Una máquina con caché perfecta es $\frac{CPI_{real}}{CPI_{ideal}} = \frac{3,376}{2} = 1,69$ veces más rápida.

Rendimiento de la caché

¿Si el procesador es más rápido, pero la memoria es la misma?

- Si suponemos que se reduce el CPI_{ideal} a 1 ciclo, el sistema con una caché perfecta sería: $2,376 / 1 = 2,376$ veces más rápido.
- La cantidad de tiempo destinada a bloqueos de memoria tendría un porcentaje más alto del tiempo de ejecución: $1,376 / 3,376 = 41\% \rightarrow 1,376/2,376 = 58\%$.

Rendimiento de la caché

¿Si el procesador es más rápido, pero la memoria es la misma? Si suponemos que se dobla la frecuencia a 100 Mhz manteniendo el $CPI_{ideal} = 2$ ciclos:

- El sistema de memoria es el mismo, este tiempo de acceso no varía (medido en segundos), pero medido en los ciclos de reloj de su CPU sería el doble que anteriormente, o sea, 80 ciclos:
 $CPI_{real} = 2 + (2\% \times 80) + (36\% \times 4\% \times 80) = 4,752$ ciclos.
- Rendimiento relativo de esta nueva máquina respecto a la antigua:

$$\frac{\text{Tiempo de ejecución a 50 Mhz}}{\text{Tiempo de ejecución a 100 Mhz}} = \frac{CPI_{real} \times T_{ciclo}}{CPI_{real} \times T_{ciclo}} = \frac{3,376 \times 1/(50 \times 10^6)}{4,752 \times 1/(100 \times 10^6)} = 1,42$$

- La máquina con un reloj el doble de rápido es sólo 1,42 veces más rápida y no el doble de rápida.

Agenda

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Memoria caché asociativa por conjuntos

- **Memoria caché de correspondencia directa:** cada bloque puede ir sólo a una posición de la caché.
- **Memoria caché asociativa por conjuntos:**
 - **m** posiciones.
 - En cada posición hay espacio para almacenar **n** bloques de datos.

A cada bloque que se trae de memoria principal le corresponde una posición concreta de caché, pero dentro de esa posición el bloque puede ocupar cualquiera de los **n** huecos disponibles. Cada **posición** de caché se convierte en un **conjunto de n elementos**.

Memoria caché asociativa por conjuntos

Localizar un bloque en caché

posición en caché = (dirección de bloque)
MODULO (número de posiciones de la caché)

- **Diferencia con las de correspondencia directa:** una vez conocida la posición de caché, habrá que buscar entre todos los diferentes bloques que forman el conjunto de esa posición, comparando con todas las correspondiente etiquetas. (En la de correspondencia directa hay que comparar con la etiqueta del único bloque existente en esa posición).

Memoria caché asociativa por conjuntos

**Caché 1-asociativa
(correspondencia directa)**

Posición	Etiqueta	Datos
0		
1		
2		
3		
4		
5		
6		
7		

Caché 2-asociativa

Conjunto	Etiqueta	Datos	Etiqueta	Datos
0				
1				
2				
3				

Caché 4-asociativa

Conjunto	Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos
0								
1								

**Caché 8-asociativa
(totalmente asociativa)**

Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos	Etiqueta	Datos

Figura: Cuatro posibilidades para una caché con capacidad para 8 bloques.

Memoria caché asociativa por conjuntos

- **Ventaja de incrementar el grado de asociatividad:** descende la tasa de fallos gracias a una asignación más flexible del espacio disponible en caché a los bloques llegados desde memoria principal.
- **Desventaja:** incremento en el tiempo de acierto.

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché de correspondencia directa de 4 posiciones.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Pos 0	Pos 1	Pos 2	Pos 3

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché de correspondencia directa de 4 posiciones.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Pos 0	Pos 1	Pos 2	Pos 3
0	0000	00	Fallo	M[0]			

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché de correspondencia directa de 4 posiciones.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Pos 0	Pos 1	Pos 2	Pos 3
0	0000	00	Fallo	M[0]			
8	1000	00	Fallo	M[8]			

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché de correspondencia directa de 4 posiciones.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Pos 0	Pos 1	Pos 2	Pos 3
0	0000	00	Fallo	M[0]			
8	1000	00	Fallo	M[8]			
0	0000	00	Fallo	M[0]			

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché de correspondencia directa de 4 posiciones.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Pos 0	Pos 1	Pos 2	Pos 3
0	0000	00	Fallo	M[0]			
8	1000	00	Fallo	M[8]			
0	0000	00	Fallo	M[0]			
6	0110	10	Fallo	M[0]		M[6]	

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché de correspondencia directa de 4 posiciones.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Pos 0	Pos 1	Pos 2	Pos 3
0	0000	00	Fallo	M[0]			
8	1000	00	Fallo	M[8]			
0	0000	00	Fallo	M[0]			
6	0110	10	Fallo	M[0]		M[6]	
8	1000	00	Fallo	M[8]		M[6]	

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché asociativa con 2 conjuntos de 2 bloques.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Conjunto 0		Conjunto 1	

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché asociativa con 2 conjuntos de 2 bloques.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Conjunto 0		Conjunto 1	
0	0000	0	Fallo	M[0]			

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché asociativa con 2 conjuntos de 2 bloques.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Conjunto 0		Conjunto 1	
0	0000	0	Fallo	M[0]			
8	1000	0	Fallo	M[0]	M[8]		

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché asociativa con 2 conjuntos de 2 bloques.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Conjunto 0		Conjunto 1	
0	0000	0	Fallo	M[0]			
8	1000	0	Fallo	M[0]	M[8]		
0	0000	0	Acierto	M[0]	M[8]		

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché asociativa con 2 conjuntos de 2 bloques.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Conjunto 0		Conjunto 1	
0	0000	0	Fallo	M[0]			
8	1000	0	Fallo	M[0]	M[8]		
0	0000	0	Acierto	M[0]	M[8]		
6	0110	0	Fallo	M[0]	M[6]		

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché asociativa con 2 conjuntos de 2 bloques.

Dir. bloque		Pos. caché	Acierto o fallo	Contenido de la caché			
Base 10	Base 2			Conjunto 0		Conjunto 1	
0	0000	0	Fallo	M[0]			
8	1000	0	Fallo	M[0]	M[8]		
0	0000	0	Acierto	M[0]	M[8]		
6	0110	0	Fallo	M[0]	M[6]		
8	1000	0	Fallo	M[8]	M[6]		

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché totalmente asociativa.

Dir. bloque		Acierto o fallo	Contenido de la caché			
Base 10	Base 2		Conjunto único			

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché totalmente asociativa.

Dir. bloque		Acierto o fallo	Contenido de la caché			
Base 10	Base 2		Conjunto único			
0	0000	Fallo	M[0]			

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché totalmente asociativa.

Dir. bloque		Acierto o fallo	Contenido de la caché			
Base 10	Base 2		Conjunto único			
0	0000	Fallo	M[0]			
8	1000	Fallo	M[0]	M[8]		

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché totalmente asociativa.

Dir. bloque		Acierto o fallo	Contenido de la caché			
Base 10	Base 2		Conjunto único			
0	0000	Fallo	M[0]			
8	1000	Fallo	M[0]	M[8]		
0	0000	Acierto	M[0]	M[8]		

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché totalmente asociativa.

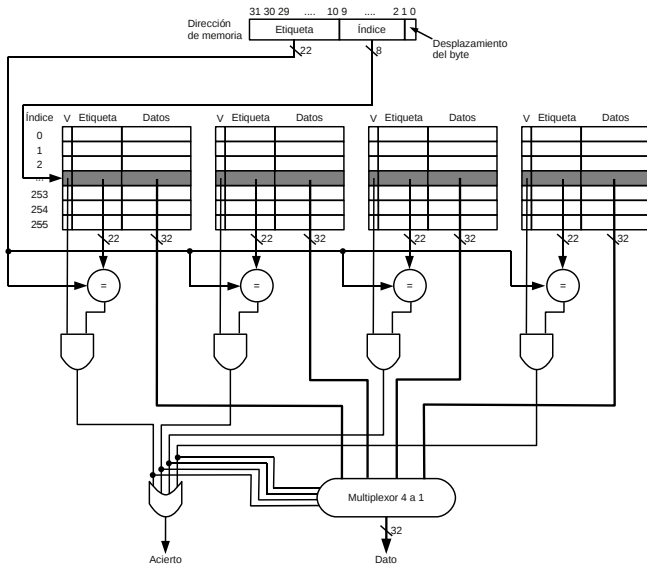
Dir. bloque		Acierto o fallo	Contenido de la caché			
Base 10	Base 2		Conjunto único			
0	0000	Fallo	M[0]			
8	1000	Fallo	M[0]	M[8]		
0	0000	Acierto	M[0]	M[8]		
6	0110	Fallo	M[0]	M[8]	M[6]	

Memoria caché asociativa por conjuntos

Tabla: Evolución del contenido de una memoria caché totalmente asociativa.

Dir. bloque		Acierto o fallo	Contenido de la caché			
Base 10	Base 2		Conjunto único			
0	0000	Fallo	M[0]			
8	1000	Fallo	M[0]	M[8]		
0	0000	Acierto	M[0]	M[8]		
6	0110	Fallo	M[0]	M[8]	M[6]	
8	1000	Acierto	M[0]	M[8]	M[6]	

Construcción de una caché 4-asociativa



Memoria caché asociativa por conjuntos

- Aumentar la asociatividad requiere más comparadores, así como más bits para las etiquetas de los bloques de caché.
- Ejemplo: caché de 4096 (2^{12}) bloques monopalabra. Direcciones de 32 bits.

	Nº Conjuntos	Nº bits para una etiqueta	Nº total de bits para etiquetas
Corresp. directa	4096 (2^{12})	30 - 12 = 18	4096 x 18 = 72 Kb
2-asociativa	2048 (2^{11})	30 - 11 = 19	4096 x 19 = 76 Kb
4-asociativa	1024 (2^{10})	30 - 10 = 20	4096 x 20 = 80 Kb
Total. asociativa	1	30	4096 x 30 = 120 Kb

Agenda

- 1 **Introducción**
 - Principio de localidad
 - Conceptos generales
- 2 **Memoria caché**
 - Memoria caché de correspondencia directa
 - Rendimiento de la caché
 - Memoria caché asociativa por conjuntos
- 3 **Tratamiento de los fallos de caché**
 - Tratamiento de los fallos de lectura
 - Tratamiento de los fallos de escritura
- 4 **Memorias caché multinivel**
- 5 **Características de la memoria caché en algunos sistemas actuales**

Agenda

- 1 **Introducción**
 - Principio de localidad
 - Conceptos generales
- 2 **Memoria caché**
 - Memoria caché de correspondencia directa
 - Rendimiento de la caché
 - Memoria caché asociativa por conjuntos
- 3 **Tratamiento de los fallos de caché**
 - **Tratamiento de los fallos de lectura**
 - Tratamiento de los fallos de escritura
- 4 **Memorias caché multinivel**
- 5 **Características de la memoria caché en algunos sistemas actuales**

Tratamiento de los fallos de lectura

- **Tras producirse un fallo:** traer desde memoria principal a caché el bloque que contiene la palabra solicitada y colocarlo en su posición de caché:
 - **Caché de correspondencia directa:** si esa posición está ocupada por otro bloque, éste se sustituye por el nuevo.
 - **Caché asociativa:** si el conjunto de la posición de caché que le corresponde al nuevo bloque está lleno, necesitamos un **algoritmo de sustitución** para decidir qué bloque de ese conjunto será sustituido. Dos posibles algoritmos:
 - **Aleatorio:** se elige un bloque al azar.
 - **Usado Menos Recientemente** (*Last Recently Used* o LRU): el bloque reemplazado es el que ha estado más tiempo sin utilizar.
- Se implementan en hardware para mayor rapidez.
Costoso implementar LRU puro \Rightarrow Pseudo-LRU (*bit de uso*).

Agenda

- 1 **Introducción**
 - Principio de localidad
 - Conceptos generales
- 2 **Memoria caché**
 - Memoria caché de correspondencia directa
 - Rendimiento de la caché
 - Memoria caché asociativa por conjuntos
- 3 **Tratamiento de los fallos de caché**
 - Tratamiento de los fallos de lectura
 - Tratamiento de los fallos de escritura
- 4 **Memorias caché multinivel**
- 5 **Características de la memoria caché en algunos sistemas actuales**

Tratamiento de los fallos de escritura

- **Si la palabra a escribir está en caché (acierto):**
 - **Política de escritura directa (*write through*):** la palabra se escribe tanto en el bloque de caché como en el bloque de memoria principal.
 - **Política de postescritura (*write back*):** la palabra se escribe sólo en el bloque de caché.
Además, se activa un *bit de modificación*: el bloque de caché modificado se escribirá completamente en la memoria principal cuando sea sustituido por otro.

Tratamiento de los fallos de escritura

- Si la palabra a escribir NO está en caché (fallo):
 - Política de escritura directa (*write through*): se escribe la palabra en memoria principal, pero su bloque no se trae a caché.
 - Política de postescritura (*write back*): si la caché contiene bloques multipalabra:
 - 1 Se trae a caché el bloque al que pertenece la palabra.
 - 2 Se escribe la palabra en ese bloque de caché.

Si los bloques son monopalabra: sólo el paso 2.

Tratamiento de los fallos de escritura

- **Ventajas de la política de postescritura:**

- Las palabras se pueden escribir muy rápidamente, pues sólo se escriben en caché y no en memoria principal.
- Múltiples escrituras dentro de un bloque requieren sólo una escritura en la memoria principal (cuando el bloque es sustituido por otro).
- Las escrituras en memoria principal son siempre de bloques completos \Rightarrow uso efectivo de un alto ancho de banda en esa comunicación.

Tratamiento de los fallos de escritura

- **Ventajas de la política de escritura directa:**

- Los fallos son menos costosos:
 - En los fallos de lectura, si un bloque es expulsado de caché para ser sustituido por otro, no es necesario escribirlo en memoria principal.
 - En los fallos de escritura, sólo se escribe una palabra en memoria principal.
- Más fácil de realizar desde el punto de vista del hardware necesario:
 - Aunque, para ser práctico, en un sistema muy rápido este esquema necesita de un **buffer de escrituras** a memoria.

Agenda

1 Introducción

- Principio de localidad
- Conceptos generales

2 Memoria caché

- Memoria caché de correspondencia directa
- Rendimiento de la caché
- Memoria caché asociativa por conjuntos

3 Tratamiento de los fallos de caché

- Tratamiento de los fallos de lectura
- Tratamiento de los fallos de escritura

4 Memorias caché multinivel

5 Características de la memoria caché en algunos sistemas actuales

Memorias caché multinivel

Objetivo: reducir la diferencia de velocidad CPU – memoria principal



Incorporar, al menos, un nivel adicional de caché.

- Si el segundo nivel de caché contiene los datos deseados, la penalización por fallo del primer nivel será el tiempo de acceso al segundo nivel (mucho menor que el de la memoria principal).
- Si ni el nivel primario ni el secundario de caché contienen los datos, se requiere acceder a la memoria principal y se obtiene una penalización por fallo mayor.

Memorias caché multinivel

- **Ejemplo:**

- $CPI_{ideal} = 1$ ciclo.
- Reloj a 500 MHz.
- Tiempo de acceso a la memoria principal de 200 ns.
- Tasa de fallos por instrucciones en la caché primaria del 5 %.

¿Cuánto más rápida será la máquina al añadir una segunda caché que tiene un tiempo de acceso de 20 ns y una tasa de fallos del 2 %?

Memorias caché multinivel

- **Ejemplo:**

- **Máquina original:**

La penalización por fallo es de: $200 \text{ ns} / 2 \text{ ns/ciclo} = 100$ ciclos.

$CPI_{\text{real}} = CPI_{\text{ideal}} + \text{ciclos de bloqueo de memoria} = 1 + 5\% \times 100 = 6$ ciclos.

- **Máquina con 2 niveles de caché:**

Ciclos necesarios para acceder a la caché secundaria es: $20 \text{ ns} / (2 \text{ ns/ciclo}) = 10$ ciclos.

$CPI_{\text{real}} = CPI_{\text{ideal}} + \text{ciclos de bloqueo de memoria} = 1 + 5\% \times (10 + 2\% \times 100) = 1,6$ ciclos.

- **Conclusión:**

La máquina con 2 niveles de cache es $(6/1,6) = 3,75$ veces más rápida que la original.

Agenda

- 1 **Introducción**
 - Principio de localidad
 - Conceptos generales
- 2 **Memoria caché**
 - Memoria caché de correspondencia directa
 - Rendimiento de la caché
 - Memoria caché asociativa por conjuntos
- 3 **Tratamiento de los fallos de caché**
 - Tratamiento de los fallos de lectura
 - Tratamiento de los fallos de escritura
- 4 **Memorias caché multinivel**
- 5 **Características de la memoria caché en algunos sistemas actuales**

Memorias caché en algunos sistemas actuales

Tabla: Características de las cachés L1 del AMD Phenom II y el IBM Power6. Hay una caché de datos y otra de instrucciones por cada núcleo.

Característica	AMD Phenom II	IBM Power6
Organización de la caché	Cachés I+D separadas	Cachés I+D separadas
Tamaño de la caché	64 KB cada una	64 KB cada una
Asociatividad	2-asociativa	4-asociativa (I) / 8-asociativa (D)
Reemplazo	ND	ND
Tamaño de bloque	ND	128 bytes
Política de escritura	ND	Escritura directa

Memorias caché en algunos sistemas actuales

Tabla: Características de las cachés L2 del AMD Phenom II y el IBM Power6. Hay una caché por núcleo y es compartida por datos e instrucciones.

Característica	AMD Phenom II	IBM Power6
Organización de la caché	Caché compartida	Caché compartida
Tamaño de la caché	512 KB	4 MB
Asociatividad	16-asociativa	8-asociativa
Reemplazo	ND	Pseudo-LRU
Tamaño de bloque	ND	128 bytes
Política de escritura	ND	Postescritura