

## UT 3. Subsistema de memoria

### Tema 3.1 Prestaciones del subsistema de memoria

J. Flich, P. López, V. Lorente,  
A. Pérez, S. Petit, J.C. Ruiz, S. Sáez, J. Sahuquillo

Departamento de Informática de Sistemas y Computadores  
Universitat Politècnica de València



**DOCENCIA VIRTUAL**

**Finalidad:**  
Prestación del servicio Público de educación superior (art. 1 LOU)

**Responsable:**  
Universitat Politècnica de València.

**Derechos de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento conforme a políticas de privacidad:**  
<http://www.upv.es/contenidos/DPD/>

**Propiedad intelectual:**  
Uso exclusivo en el entorno de aula virtual.  
Queda prohibida la difusión, distribución o divulgación de la grabación de las clases y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes.  
La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa o civil



UNIVERSITAT  
POLITECNICA  
DE VALÈNCIA





## Índice

- 1 Repaso de la jerarquía de memoria.
- 2 Repaso de la estructura y funcionamiento de las caches
- 3 Evaluación de las prestaciones del subsistema de memoria.

### Bibliografía

 John L. Hennessy and David A. Patterson.

*Computer Architecture, Fifth Edition: A Quantitative Approach.*  
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5  
edition, 2012.

## Índice

- 1 Repaso de la jerarquía de memoria.
- 2 Repaso de la estructura y funcionamiento de las caches
- 3 Evaluación de las prestaciones del subsistema de memoria.

# 1. Repaso de la jerarquía de memoria.

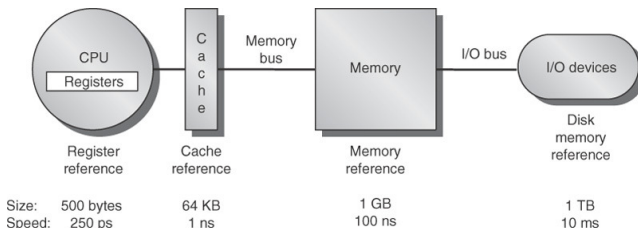
## Jerarquía de memoria

“...los programadores pretenden acceder a cantidades ilimitadas de memoria rápida...”

→ No existe ninguna tecnología capaz de proporcionarlo

**Solución:** organización jerárquica utilizando distintas tecnologías.

**Jerarquía de memoria** organización de la memoria en diferentes niveles, donde cada nivel es más pequeño, más rápido y más caro que el nivel inferior.



© 2007 Elsevier, Inc. All rights reserved.

# 1. Repaso de la jerarquía de memoria.

## Jerarquía de memoria (cont.)

¿Por qué la jerarquía proporciona buenas prestaciones?

- **Objetivo:** velocidad cercana al más rápido.
- Principio de localidad. Los programas tienden a reutilizar el código y los datos utilizados recientemente.

**Localidad temporal:** los elementos ya accedidos serán accedidos nuevamente en el futuro próximo.

- Alta en el código: “El 10 % del código se ejecuta durante el 90 % del tiempo de ejecución de un programa”.

**Localidad espacial:** los elementos con direcciones cercanas tienden a referenciarse cercanamente en el tiempo. → organización en bloques.

¿Por qué es una solución eficiente?

- Combina tecnologías con distintas características.
  - Rápidas pero caras para prestaciones.
  - Densas pero baratas (coste por bit) para almacenamiento.



# 1. Repaso de la jerarquía de memoria.

## Requisitos diferentes según tipo de computador

**Computador de sobremesa** Un usuario, una aplicación.

- Objetivo: reducir latencia.

**Servidores** Múltiples usuarios, múltiples aplicaciones.

- Objetivos: ancho de banda, protección.

**Computadores empotrados** Una aplicación, a veces sin sistema operativo. Memoria principal pequeña.

Objetivos:

- Tiempo-real (importante conocer las prestaciones del peor-caso).
- Bajo consumo.



# 1. Repaso de la jerarquía de memoria.

## Memorias *cache*

**“Cache:** *un sitio seguro para esconder o almacenar cosas*”

Cache: primer nivel de la jerarquía de memoria.

El término cache se emplea actualmente cuando se almacena información que se reutilizará: caches de ficheros, cache de disco, cache de nombres, etc.

**Acierto:** el procesador encuentra el dato en la cache.

**Fallo:** en caso contrario. El *bloque* que contiene la palabra accedida se trae desde la memoria principal a la cache.

La gestión de la cache: aciertos, fallos y reemplazos, se realiza mediante *hardware*.

# 1. Repaso de la jerarquía de memoria.

## Memoria Virtual

Si el sistema soporta memoria virtual, los objetos referenciados por un programa deben estar en memoria principal o en disco.

El espacio de direccionamiento se divide en *páginas* que deben estar en memoria para ejecutarse.

Si se produce un *fallo de página*, la página entera se transfiere desde el disco a la memoria principal.

Los fallos de página se gestionan por *software* y no detienen el procesador. El procesador cambia de contexto, ejecutando otra tarea mientras se realiza el acceso al disco.

## Índice

- 1 Repaso de la jerarquía de memoria.
- 2 Repaso de la estructura y funcionamiento de las caches
- 3 Evaluación de las prestaciones del subsistema de memoria.

## 2. Repaso de la estructura y funcionamiento de las caches

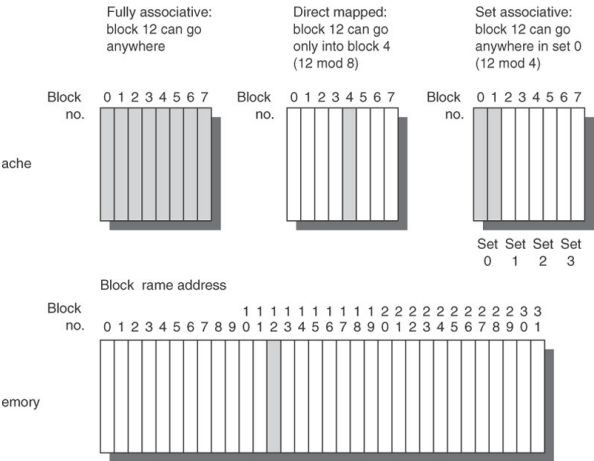
### Características de las memorias cache

Cualquier nivel de la jerarquía de memoria puede caracterizarse respondiendo a las preguntas siguientes:

- Ubicación de un bloque. ¿Dónde puede ubicarse un bloque?
- Identificación de un bloque. ¿Cómo se encuentra un bloque?
- Reemplazamiento. ¿Qué bloque se elimina ante un fallo?
- Política ante escrituras. ¿Qué se hace ante una escritura?

# 2. Repaso de la estructura y funcionamiento de las caches

## Ubicación de un bloque



© 2007 Elsevier, Inc. All rights reserved.

## 2. Repaso de la estructura y funcionamiento de las caches

### Ubicación de un bloque (cont.)

**Correspondencia *directa*** Un bloque sólo puede estar almacenado en una línea de la cache.

$\#línea = \#bloque\_referenciado \bmod \#líneas\_de\_cache$

**Correspondencia *totalmente asociativa*** Un bloque puede almacenarse en cualquier línea de la cache.

**Correspondencia *asociativa por conjuntos de n vías*** Un bloque sólo puede almacenarse en un conjunto que contiene n líneas de la cache.

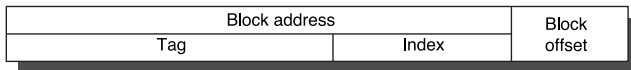
$\#conjunto = \#bloque\_referenciado \bmod \#conjuntos\_de\_cache$

## 2. Repaso de la estructura y funcionamiento de las caches

### Identificación de un bloque

Cada bloque almacenado en la cache tiene asociada una etiqueta. Para saber si un bloque se encuentra en la cache, se compara el campo etiqueta de la dirección del bloque con las etiquetas del conjunto destino.

Partes de una dirección emitida por el procesador:



© 2003 Elsevier Science (USA). All rights reserved.

Un bit ‘válido’ (V) indica si una línea tiene o no información válida. Solo se comparan aquellas etiquetas que tienen el bit válido activo.

## 2. Repaso de la estructura y funcionamiento de las caches

### Identificación de un bloque (cont.)

¿Cómo comparar?

- En paralelo con todas las etiquetas. Con correspondencia directa, sólo una comparación.
- No hace falta incluir la palabra dentro del bloque (*offset*), ya que, el bloque completo está presente o ausente.

Para un mismo tamaño de cache, al aumentar la asociatividad aumenta el número de bloques por conjunto y disminuye el número de conjuntos, por lo que se reduce el tamaño del índice y aumenta el de la etiqueta.



## 2. Repaso de la estructura y funcionamiento de las caches

### Reemplazamiento

Cuando hay un fallo de cache, el bloque referenciado se trae desde la memoria principal. Si el conjunto está lleno, ¿cuál se elimina?

Con correspondencia directa es trivial (solo 1 candidato).

Con correspondencia asociativa pueden emplearse varias estrategias:

**LRU** Menos recientemente usado. Explota la localidad temporal.

**SeudoLRU** Menos costoso que el LRU, prestaciones similares con muchas vías.

**Aleatoria** Se elije un candidato al azar. Fácil de implementar. Útil para estructuras con poca localidad.

## 2. Repaso de la estructura y funcionamiento de las caches

### Políticas de escritura

Solo se modifica un dato (byte, palabra, . . . ) del bloque.  
El bloque debe actualizarse también en la memoria principal (MP).

Estrategias en caso de acierto:

*Write-through* Se escribe tanto en la cache como en MP.

- Más fácil de implementar.
- La memoria principal siempre está actualizada.

*Write-back* Solo se escribe en la cache.

- Los bloques "sucios" (bit dirty activo) se actualizan en MP cuando se reemplazan.
- Emplea menos ancho de banda de memoria que *Write-through*.

## 2. Repaso de la estructura y funcionamiento de las caches

### Políticas de escritura (cont.)

Estrategias en caso de fallo:

*Write allocate* El bloque se trae a la cache.

Después, se llevan a cabo las acciones de escritura con acierto.

Habitual con *write-back*

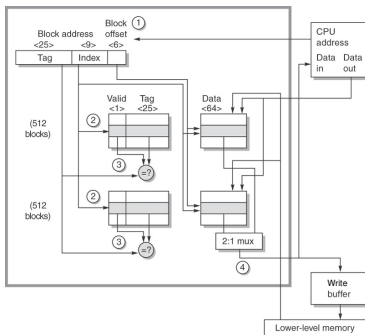
*No-write allocate* El bloque no se trae a la cache. Sólo se modifica en el nivel inferior. Habitual con *write-through*

## 2. Repaso de la estructura y funcionamiento de las caches

### Políticas de escritura (cont.)

#### Ejemplo: Cache de datos del Opteron

- 64KB, 2 vías, 64 bytes/bloque. Reemplazo LRU
- Implementa direcciones físicas de 40 bits (de los 64 bits posibles)
- Write-back, write-allocate



© 2007 Intel, Inc. All rights reserved.

## 2. Repaso de la estructura y funcionamiento de las caches

### Políticas de escritura (cont.)

- 1 El procesador envía la dirección (40 bits)  
= Dirección de bloque: 34 bits + Desplazamiento: 6 bits
- 2 Selección del conjunto (Índice):

$$2^{\text{Índice}} = \frac{\text{Capacidad cache}}{\text{Tam.bloque} \cdot \text{Num.vías}} = \frac{65536}{64 \cdot 2} = 512 = 2^9$$

Etiqueta:  $34 - 9 = 25$  bits

- 3 Lectura de las dos etiquetas del conjunto y comparación con la emitida por la CPU. En paralelo, lectura de los dos datos del conjunto.
- 4 En caso de acierto, selección de la entrada del multiplexor y envío del dato a la CPU.

## Índice

- 1 Repaso de la jerarquía de memoria.
- 2 Repaso de la estructura y funcionamiento de las caches
- 3 Evaluación de las prestaciones del subsistema de memoria.

### 3. Evaluación de las prestaciones del subsistema de memoria.

#### Tiempo de acceso medio

$$T_{\text{acceso}} = TA + TF \times PF$$

donde TA: tiempo de acierto en cache

TF: tasa de fallos

PF: penalización de fallo

**Modificación de la ecuación del tiempo de ejecución**  
para incluir el tiempo de la cache:

$$T_{\text{ej}} = T_{\text{ej cpu}} + T_{\text{extra memoria}}$$

donde  $T_{\text{ej cpu}}$  incluye el tiempo de acierto de cache y  
 $T_{\text{extra memoria}}$  el tiempo para gestionar los fallos.

### 3. Evaluación de las prestaciones del subsistema de memoria.

#### Procesador con ejecución en orden

Los fallos detienen al procesador.

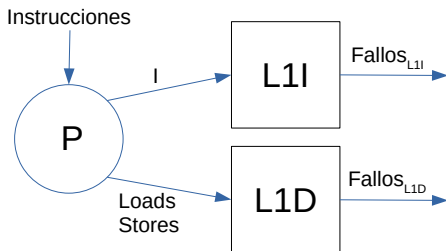
- $T_{ej\ cpu} = I \times CPI \times T_{cpu}$
- $T_{extra\ memoria} = \text{Ciclos parada mem.} \times T_{mem}$
- $\text{Ciclos parada mem.} = N^{\circ} \text{ de fallos} \times \text{Penaliz. por fallo} = NF \times PF$
- $N^{\circ} \text{ de fallos} = N^{\circ} \text{ de accesos} \times \frac{\text{Fallos}}{\text{Accesos}} = NA \times TF$
- $N^{\circ} \text{ de accesos} = \text{Instrucciones} \times \frac{\text{Accesos}}{\text{Instrucciones}} = I \times API$

$$T_{extra\ memoria} = I \times API \times TF \times PF \text{ (en ciclos)} \times T_{mem}$$



### 3. Evaluación de las prestaciones del subsistema de memoria.

#### Cache separadas de instrucciones y datos



Si consideramos caches separadas para instrucciones (L1I) y datos (L1D):

$$API = \frac{\text{Accesos}}{\text{Instrucciones}} = \frac{(\text{Accesos}_{L1I}) + (\text{Accesos}_{L1D})}{\text{Instrucciones}} = \frac{(\text{Instrucciones}) + (\text{Loads} + \text{Stores})}{\text{Instrucciones}}$$

Número medio de accesos por instrucción a L1I :  $API_{L1I} = \frac{\text{Instrucciones}}{\text{Instrucciones}} = 1$

Número medio de accesos por instrucción a L1D :  $API_{L1D} = \frac{\text{Loads} + \text{Stores}}{\text{Instrucciones}}$

Desglose del API:

$$API = API_{L1I} + API_{L1D}$$

### 3. Evaluación de las prestaciones del subsistema de memoria.

#### Cache separadas de instrucciones y datos (cont.)

$$T_{\text{extra mem.}} = (\text{Fallos}_{L1I} + \text{Fallos}_{L1D}) \times PF \times T_{\text{mem}}$$

Puesto que:

$$\blacksquare TF_{L1I} = \frac{\text{Fallos}_{L1I}}{\text{Accesos}_{L1I}} = \frac{\text{Fallos}_{L1I}}{I} \rightarrow \text{Fallos}_{L1I} = I \times TF_{L1I}$$

$$\blacksquare TF_{L1D} = \frac{\text{Fallos}_{L1D}}{\text{Accesos}_{L1D}} = \frac{\text{Fallos}_{L1D}}{\text{Loads} + \text{Stores}} \rightarrow \text{Fallos}_{L1D} = (\text{Loads} + \text{Stores}) \times TF_{L1D}$$

Entonces:

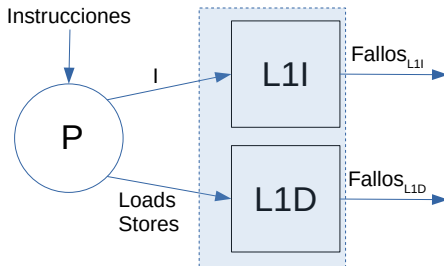
$$\begin{aligned} T_{\text{extra mem.}} &= (I \times TF_{L1I} + (\text{Loads} + \text{Stores}) \times TF_{L1D}) \times PF \times T_{\text{mem}} \\ &= I \times \left( \frac{I}{I} \times TF_{L1I} + \frac{(\text{Loads} + \text{Stores})}{I} \times TF_{L1D} \right) \times PF \times T_{\text{mem}} \end{aligned}$$

Sustituyendo  $API_{L1I}$  y  $API_{L1D}$ , queda:

$$T_{\text{extra mem.}} = I \times (API_{L1I} \times TF_{L1I} + API_{L1D} \times TF_{L1D}) \times PF \times T_{\text{mem}}$$

### 3. Evaluación de las prestaciones del subsistema de memoria.

#### Tasa de fallos unificada



$$TF' = \frac{\text{Fallos}_{L1I} + \text{Fallos}_{L1D}}{I + \text{Loads} + \text{Stores}} = \frac{TF_{L1I} \times I + TF_{L1D} \times (\text{Loads} + \text{Stores})}{I + \text{Loads} + \text{Stores}} = \frac{TF_{L1I} \times \frac{I}{I + \text{Loads} + \text{Stores}} + TF_{L1D} \times \frac{\text{Loads} + \text{Stores}}{I + \text{Loads} + \text{Stores}}}{1}$$

$$TF' = TF_{L1I} \times \frac{API_{L1I}}{API} + TF_{L1D} \times \frac{API_{L1D}}{API}$$

Si se usa la tasa de fallos unificada no es necesario desglosar el API:

$$T_{\text{extra mem.}} = I \times API \times TF' \times PF \times T_{\text{mem}}$$

### 3. Evaluación de las prestaciones del subsistema de memoria.

#### Diferentes penalizaciones de fallo

Si las penalizaciones de fallo son diferentes (por ejemplo, entre L1I o L1D, o entre lecturas y escrituras en L1D), no es posible utilizar la tasa de fallos unificada y es necesario desglosar el API:

$$T_{\text{extra mem.}} = I \times \frac{I}{I} \times TF_{L1I} \times PF_{L1I} \times T_{\text{mem}} + \\ I \times \frac{\text{Loads}}{I} \times TFL_{L1D} \times PFL_{L1D} \times T_{\text{mem}} + \\ I \times \frac{\text{Stores}}{I} \times TFE_{L1D} \times PFE_{L1D} \times T_{\text{mem}}$$

Donde:

- $TF_{L1I} = \frac{\text{Fallos}_{L1I}}{I}$ ,  $TFL_{L1D} = \frac{\text{Fallos de Lectura}_{L1D}}{\text{Loads}}$ , y  $TFE_{L1D} = \frac{\text{Fallos de Escritura}_{L1D}}{\text{Stores}}$
- $PF_{L1I}$ ,  $PFL_{L1D}$ , y  $PFE_{L1D}$  son las penalizaciones correspondientes a cada tipo de fallo.



### 3. Evaluación de las prestaciones del subsistema de memoria.

#### Procesador con ejecución fuera de orden (cont.)

Por lo tanto:

$$T_{\text{extra mem.}} = I \times TF_{LII} \times PF_{LII} \times T_{mem} + \\ I \times \frac{\text{Loads}}{I} \times TFL_{LID} \times PFL_{LID} \times FNS \times T_{mem}$$