Conceptos de Arquitectura de Computadoras

Clase 7

Memoria

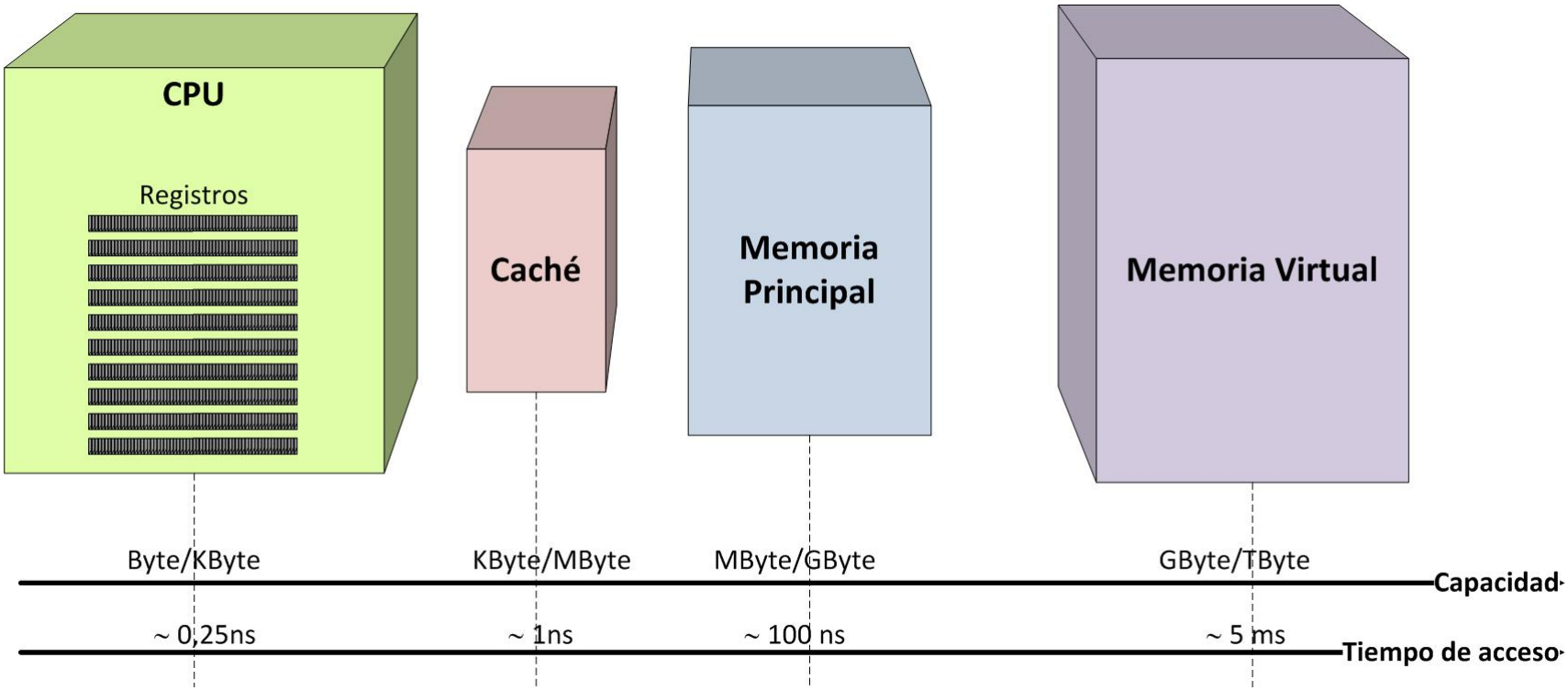
Sistema de Memoria

- Los programadores desean acceder a cantidades ilimitadas de memoria rápida !!
- Solución práctica:

Jerarquía de memoria

  - organizada en niveles que son ubicados en distintos lugares físicos
  - fabricados con tecnologías diferentes que se gestionan de manera independiente

Jerarquía de memoria



Jerarquía de memoria (2)

- **Objetivo:** la velocidad del sistema deberá ser, aproximadamente, la del nivel más rápido al costo del nivel mas barato.
- A medida que nos alejamos de la CPU, cada nivel inferior es más grande, más lento y más barato que el nivel previo (o superior) en la jerarquía.
- Debe haber correspondencia de direcciones en los distintos niveles.

Jerarquía de memoria (3)

Propiedades a cumplir

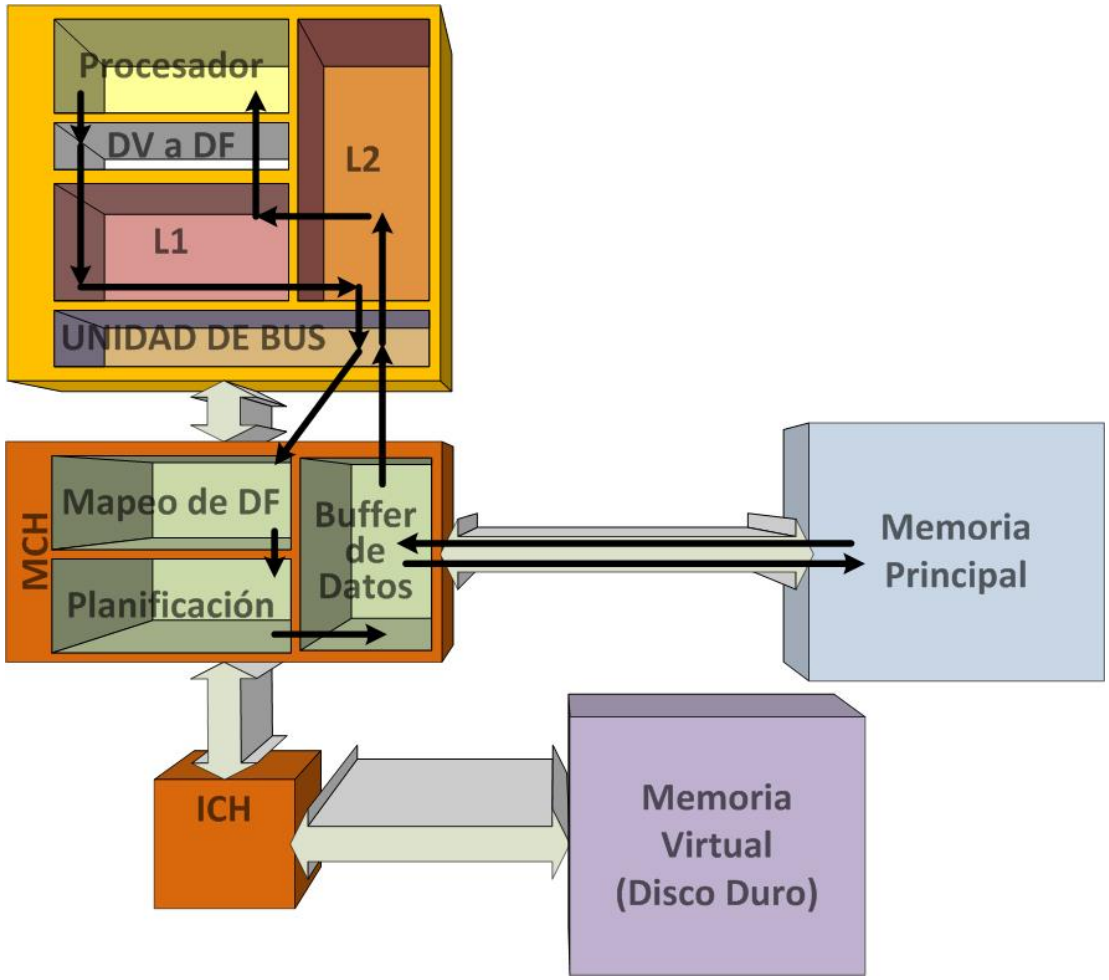
- Inclusión
  - Los datos almacenados en un nivel han de estar almacenados en los niveles inferiores a él
- Coherencia
  - Las copias de la misma información en los distintos niveles deben contener los mismos valores.

¿Porqué funciona la jerarquía?

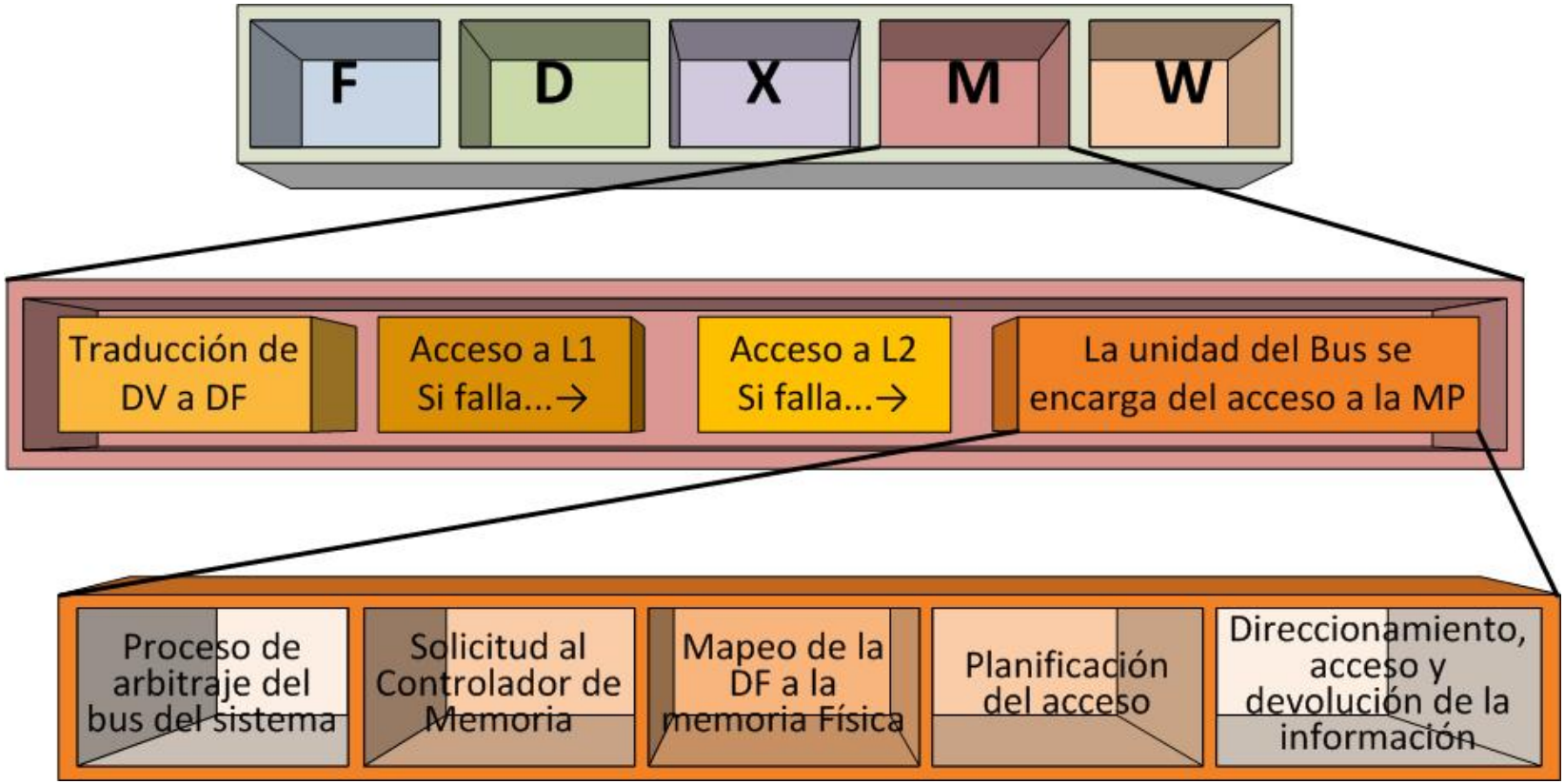
Principio de localidad de referencias

- **Localidad Temporal**: los elementos de memoria referenciados recientemente (datos o instrucciones), volverán a serlo en un futuro próximo => subo la palabra de nivel
- **Localidad Espacial**: los elementos de memoria cuyas direcciones están próximas a los últimos referenciados serán referenciados.
  => subo un bloque (con la palabra) de nivel

Mecanismo de acceso memoria

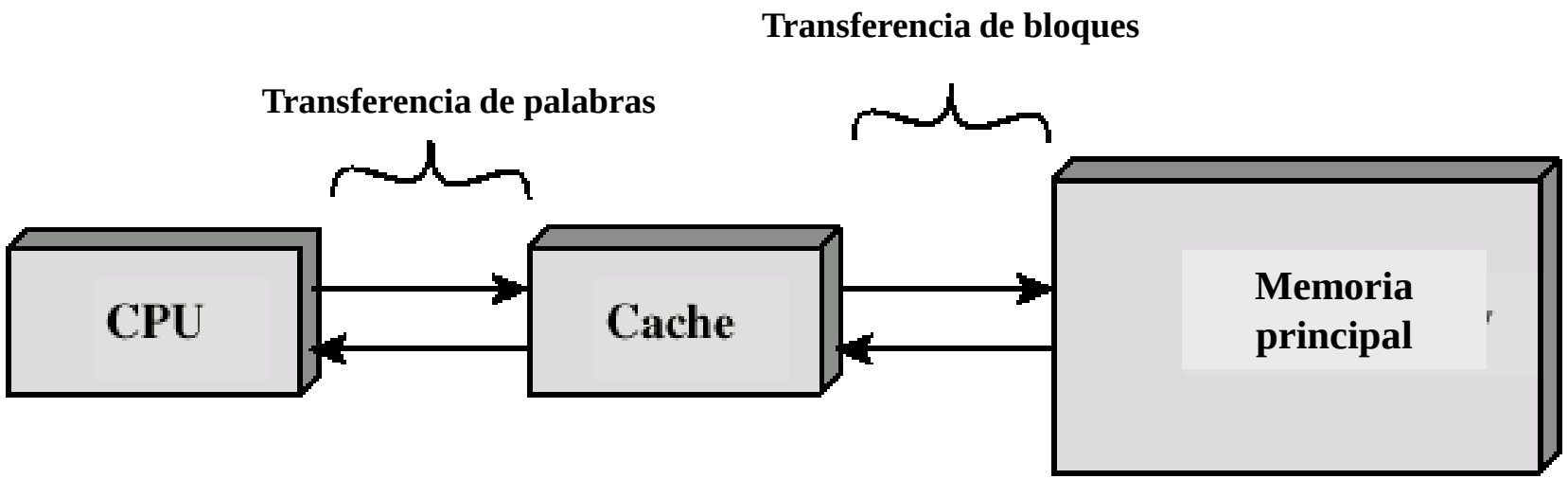


Trabajo de etapa M del cauce

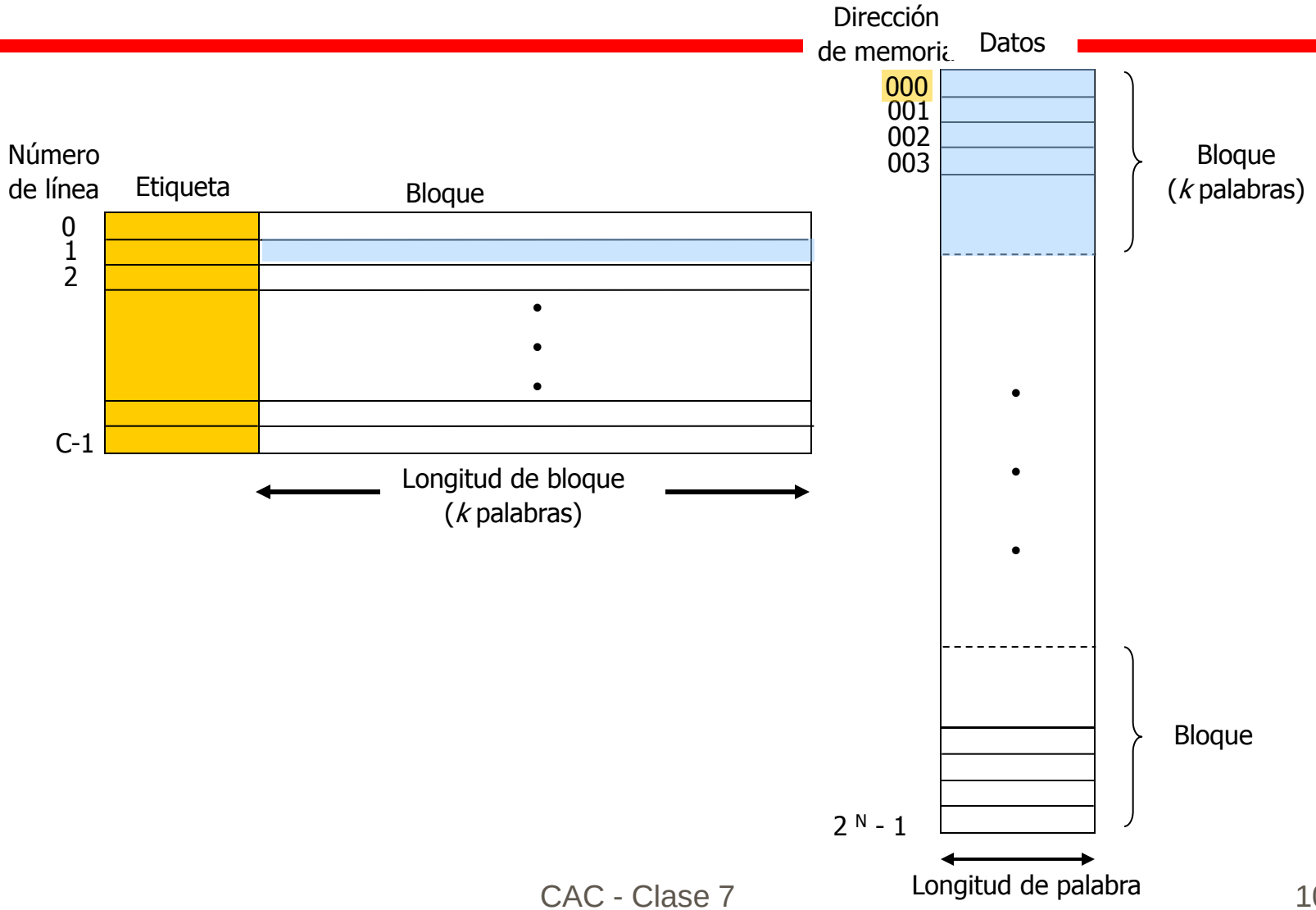


Memoria Cache

- Cantidad pequeña de memoria rápida.
- Se ubica entre la memoria principal y la CPU.
- Puede localizarse en un chip o en módulo CPU.



Memoria Cache y Principal

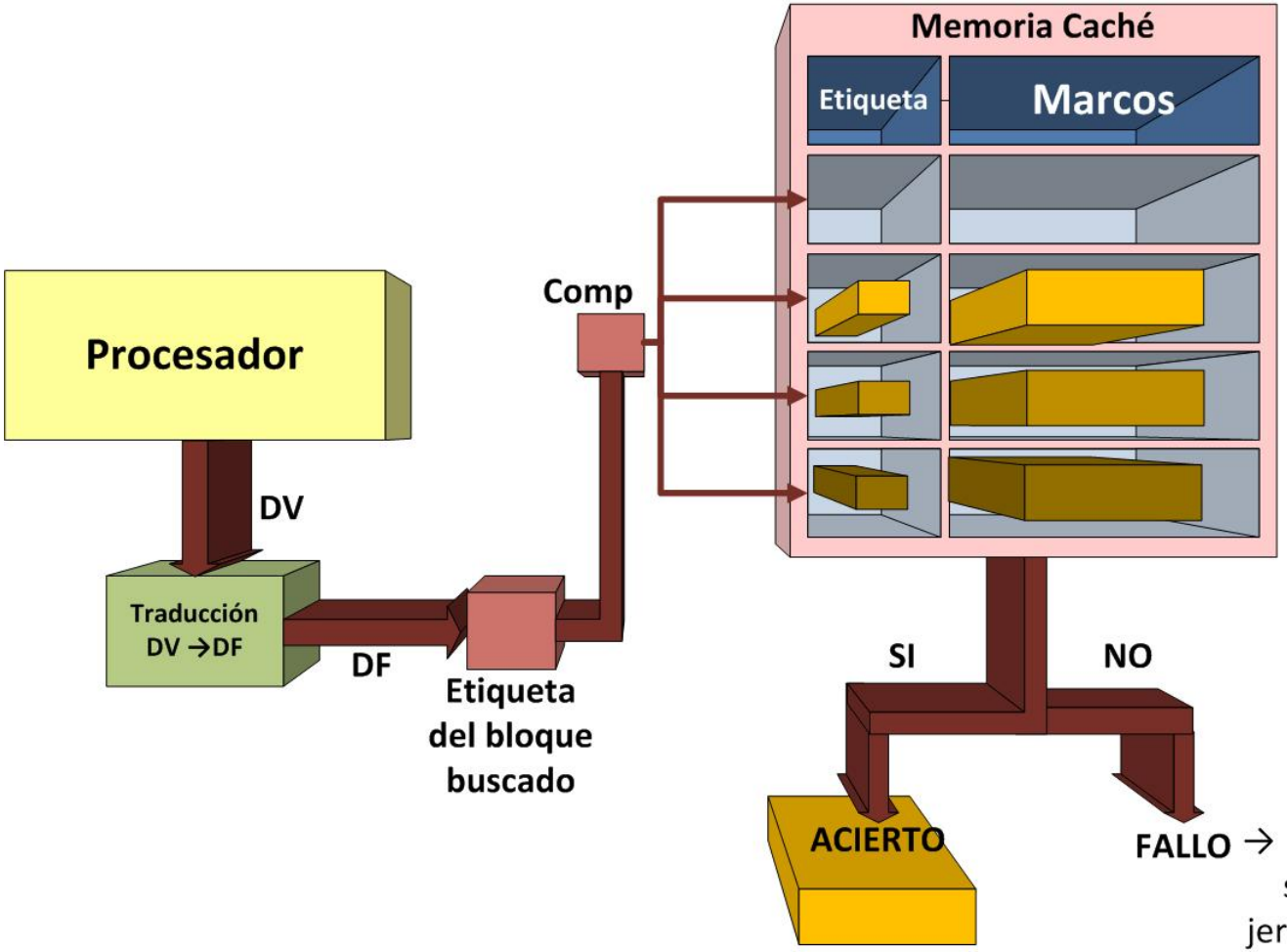


Funcionamiento de la cache

- La CPU solicita contenido de 1 dirección de memoria.
- La cache ¿tiene ese dato?
  - Si es así, la obtiene de la cache (rápidamente).
  - Si no está, se lee el bloque que contiene esa dirección desde la memoria principal y copia en la cache.
    - Después, la cache entrega el dato requerido a la CPU.

La cache incluye etiquetas para identificar qué bloque de la memoria principal está en cada una de sus líneas.

Funcionamiento de la cache (2)



Conceptos básicos

Acierto ***(hit):*** se encuentra en la caché el dato solicitado

Fallo ***(miss):*** **no se** encuentra en la caché el dato solicitado

- un bloque que contiene la palabra accedida se copia de la memoria principal a una línea de caché.
- ***Tiempo para servir un fallo:*** depende de la latencia y ancho de banda de la memoria principal.
  - ***Latencia:*** tiempo necesario para completar un acceso a memoria.
  - ***Ancho de banda:*** cantidad de información por unidad de tiempo que puede transferirse desde/hacia la memoria.
- Los **fallos de caché** se gestionan mediante hardware y causan que el procesador se detenga hasta que el dato esté disponible.

Prestaciones de la jerarquía

Tiempo de acceso medio a memoria

$$T_{acceso} = T_{acierto} + T_{fallos\_memoria}$$

$$T_{fallos\_memoria} = \text{Tasa de fallos x Penalización\_fallo}$$

$$T_{acceso} = T_{acierto} + TF \times PF$$

Para mejorar las prestaciones

- Reducir el tiempo en caso de acierto ($T_{acierto}$)
- Reducir la tasa de fallos (TF)
- Reducir la penalización por fallo (PF)

Jerarquía perfecta vs real

Supongamos que un procesador con frecuencia de reloj de 2 GHz y un CPI=1 ejecuta código de 100 instrucciones.

Caso 1: Se incorpora 2 memorias caches ideales MI y MD (NO habrá fallos y $t_{acierto}$ es despreciable).

$$t = t_{cpu} + t_{mem}$$

$$t = t_{cpu} + 0 = t_{cpu}$$

$$t = \text{nI} \cdot \text{CPI} \cdot \text{T} = 100 \cdot 1 \cdot (1/2\text{G})$$

$$t = 50\text{ns}$$

Jerarquía perfecta vs real (2)

Caso 2: Se incorpora 2 caches reales. Cache MI con TF=4% y PF=100ns y Cache MD con TF=6% y PF=115ns. El 25% del código accesa datos.

$$t = t_{cpu} + t_{mem}$$

$$t_{mem} = accesosMI.(t_{aciertoMI} + TF_{MI}.PF_{MI}) + accesosMD.(t_{aciertoMD} + TF_{MD}.PF_{MD})$$

$$t_{mem} = 100(0+0{,}04x100ns) + 25(0+0{,}06x115ns)$$

$$t_{mem} = 400ns + 172{,}5ns = 572{,}5ns$$

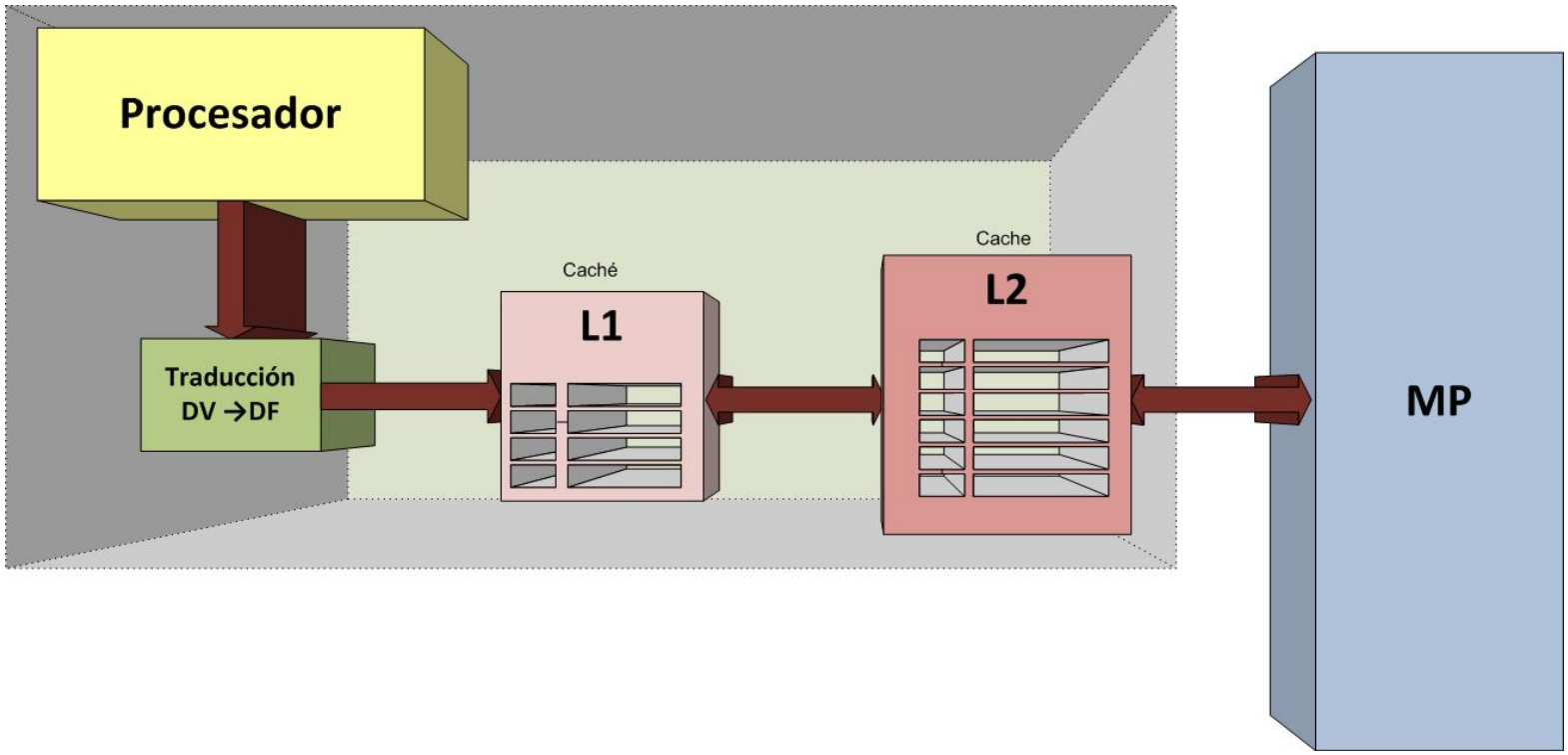
$$t = t_{cpu} + t_{mem} = 622{,}5ns$$

Diseño de la cache

- Organización (tamaño y cantidad)
- Política de ubicación
  - Tipo de función de correspondencia
- Política de reemplazo
  - Algoritmo de sustitución
- Política de escritura

Organización de la cache

- Tamaño - Costo - Niveles.



Ubicación de un bloque

- Correspondencia **directa.** Un bloque sólo puede estar almacenado en un lugar de la caché.

  Nº línea caché = Nº bloque ref. ***mod*** Nº líneas caché

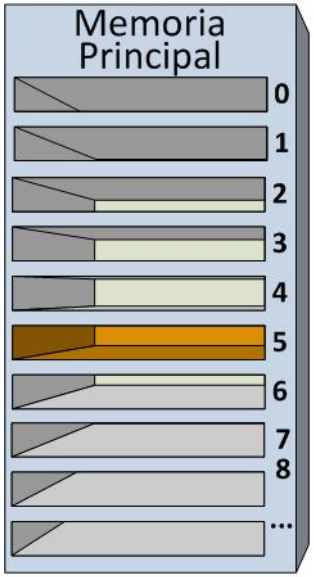
- Correspondencia **totalmente asociativa.** Un bloque puede almacenarse en cualquier lugar de la caché.
- Correspondencia **asociativa por conjuntos.** Un bloque puede almacenarse en un conjunto restringido de lugares en la caché.

  Un conjunto es un grupo de lineas de la caché.

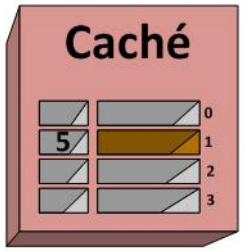
  Nº conjunto = Nº bloque ref. ***mod*** Nº conjuntos caché

Tipos de correspondencia

¿dónde se ubica el bloque 5 de la MP si la cache posee 4 lineas?

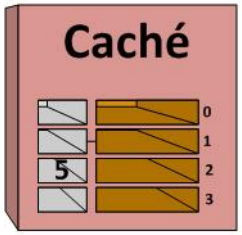


Directa

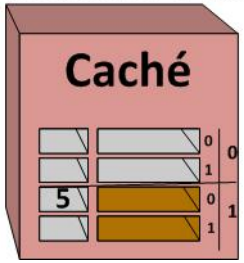


Bloque 5 mod 4 líneas = 1

Asociativa



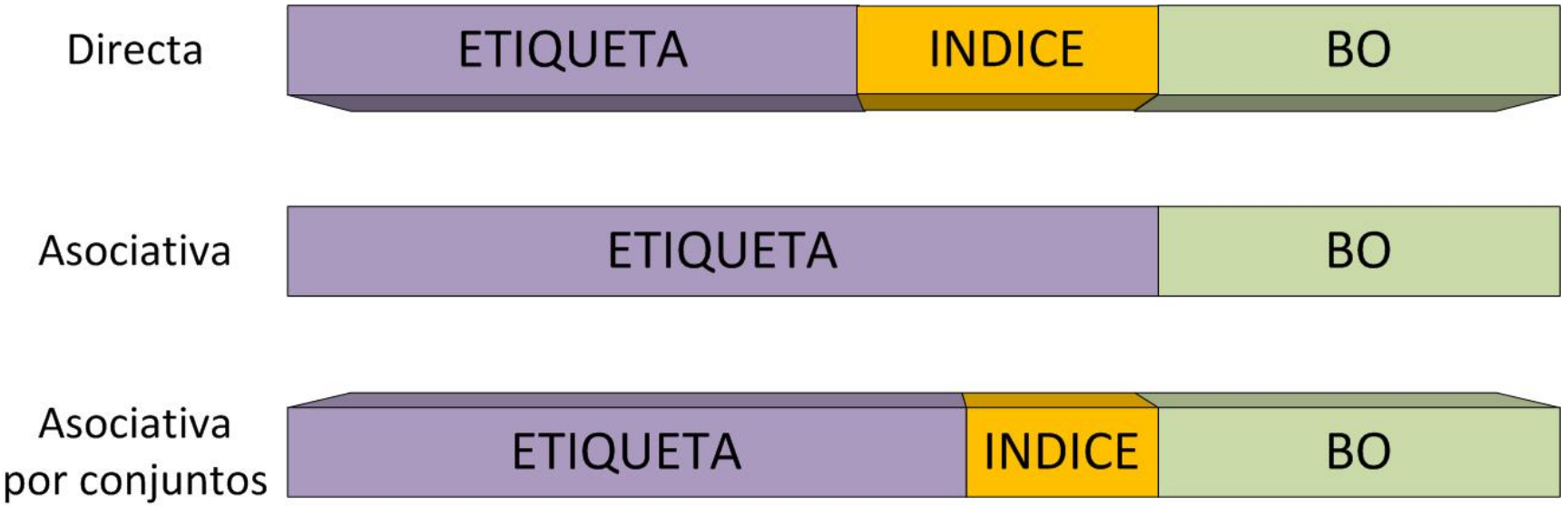
Asociativa 2 vías



Bloque 5 mod 2 conjuntos = 1

Tipos de correspondencia (2)

La interpretación de la dirección física depende del tipo que se utilice. INDICE indicará la línea ó el conjunto que le corresponde. BO representa todas las direcciones que pertenecen al bloque.



Tipos de corresponden cia (3)

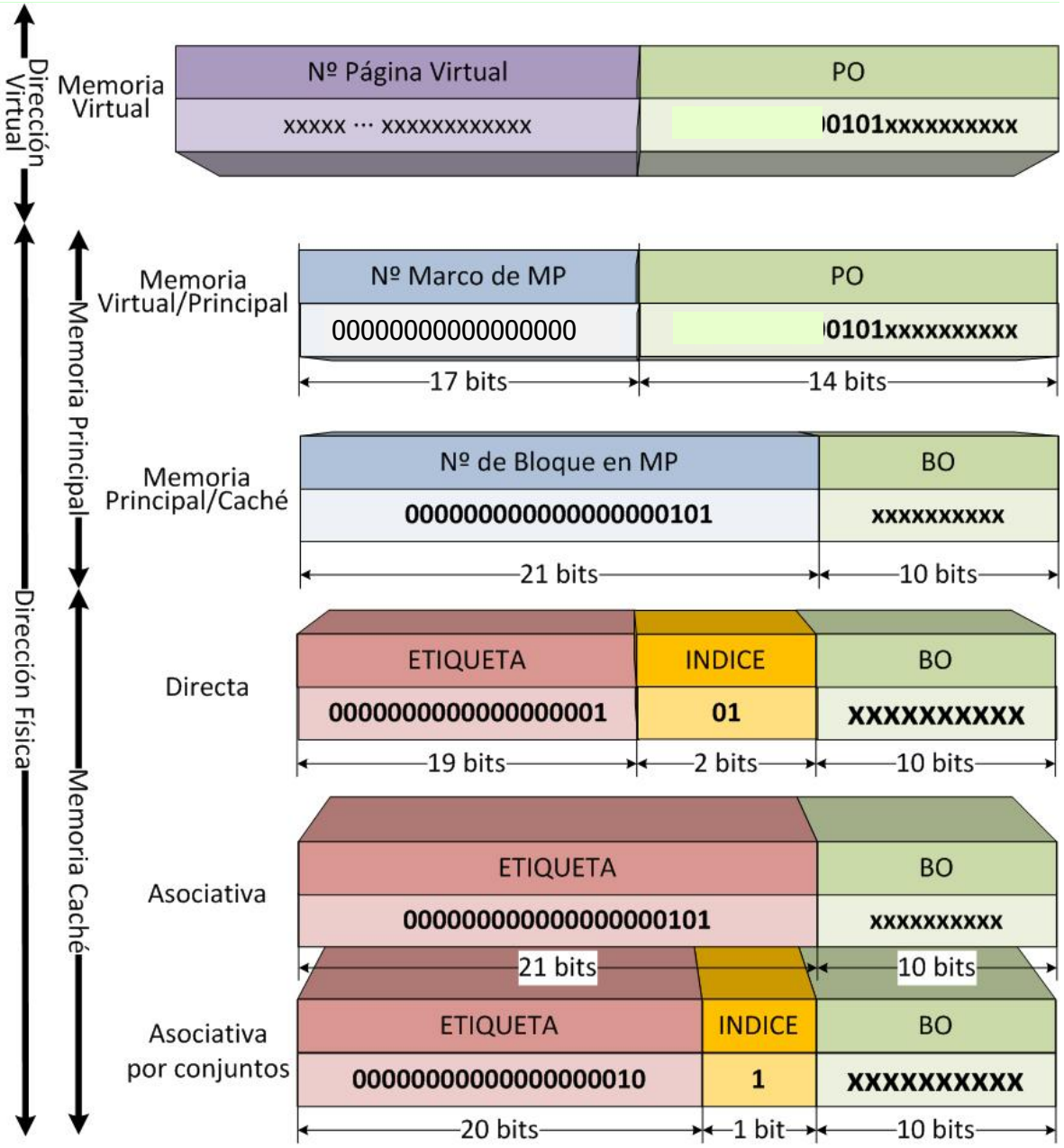
Supongamos que en el ejemplo anterior:

MP es de 2GB

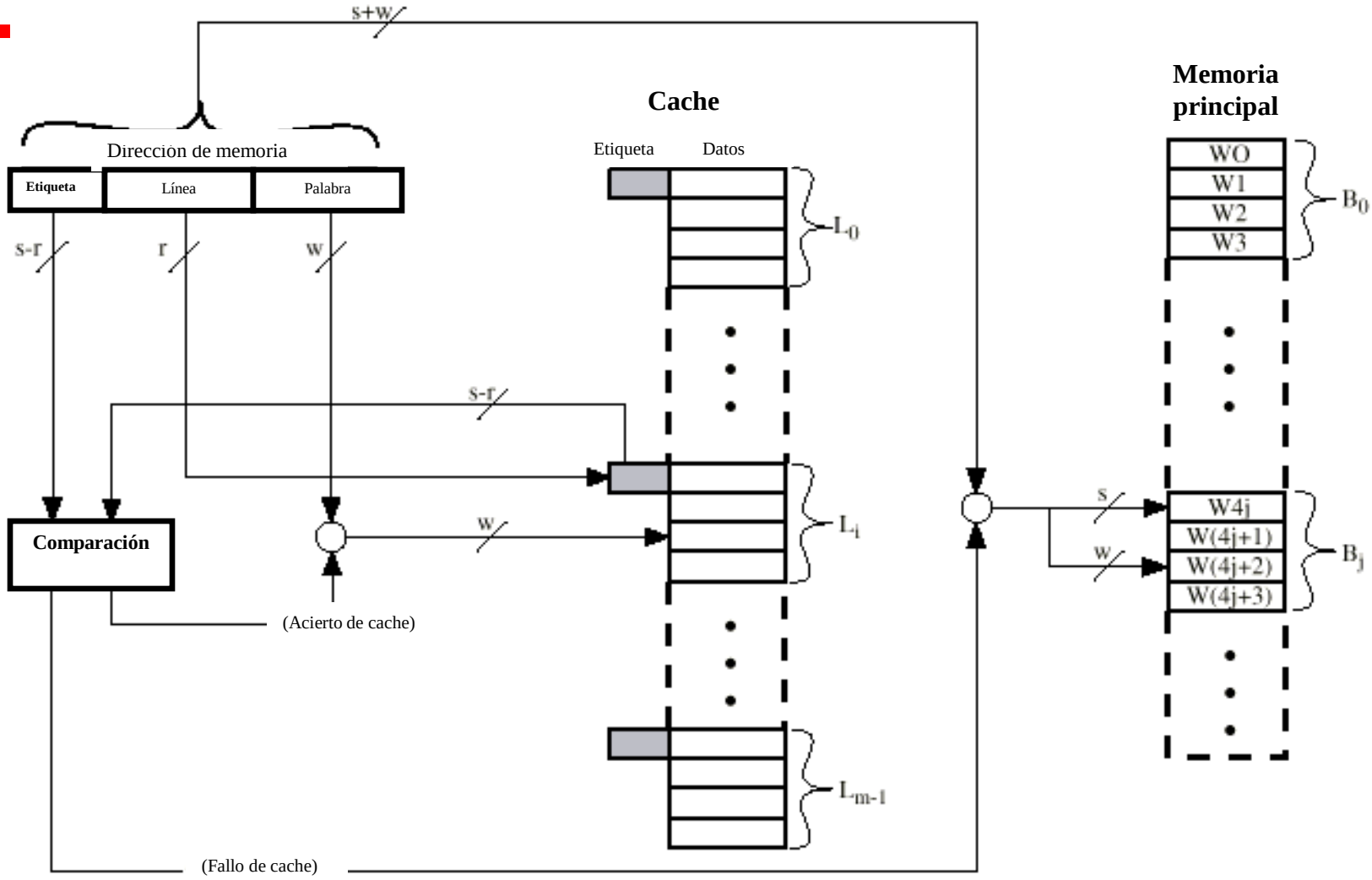
Páginas de 16KB

Bloques de 1KB

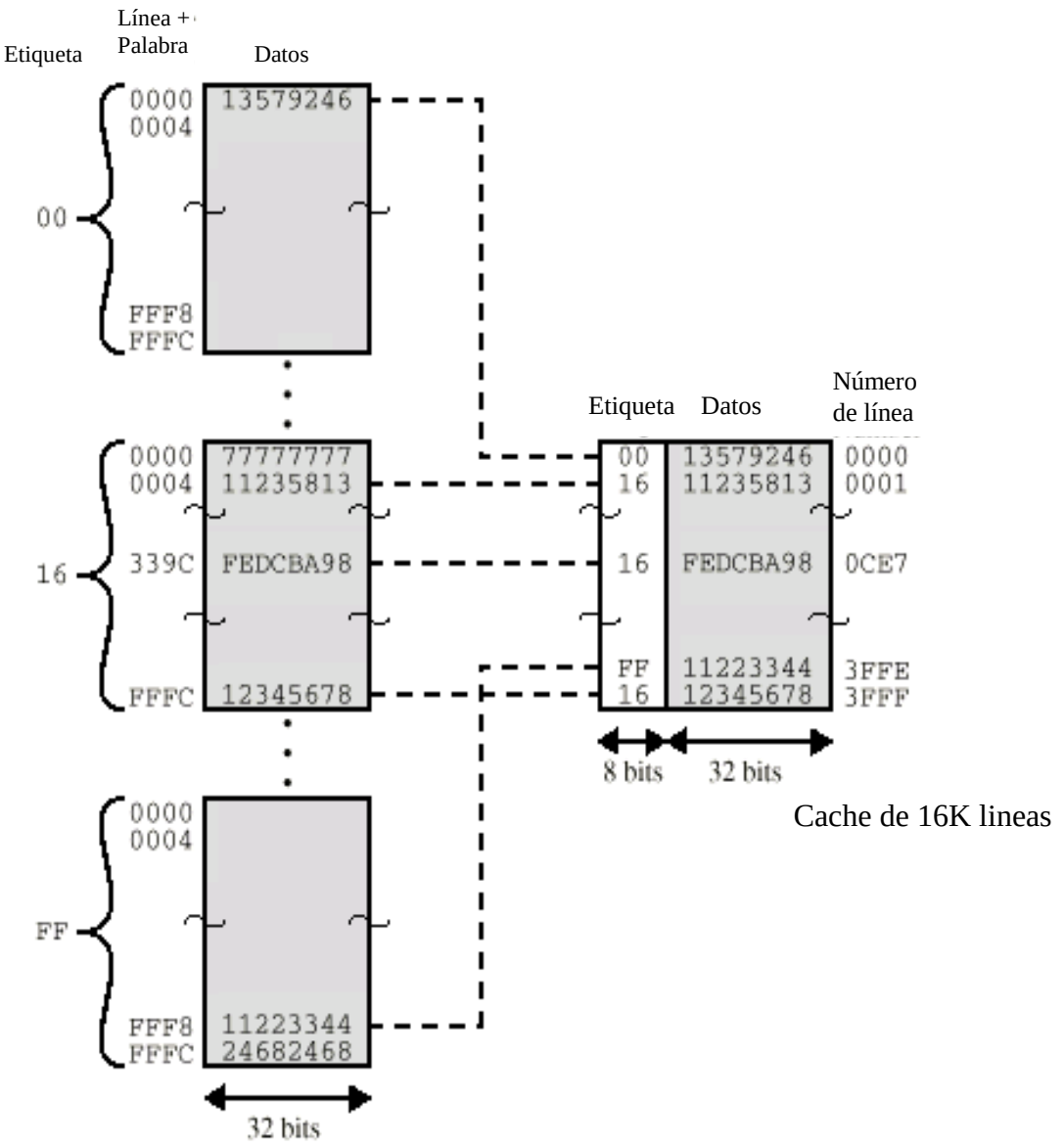
Cualquier dirección del bloque 5 es como en la figura



Correspondencia directa: Organización de cache



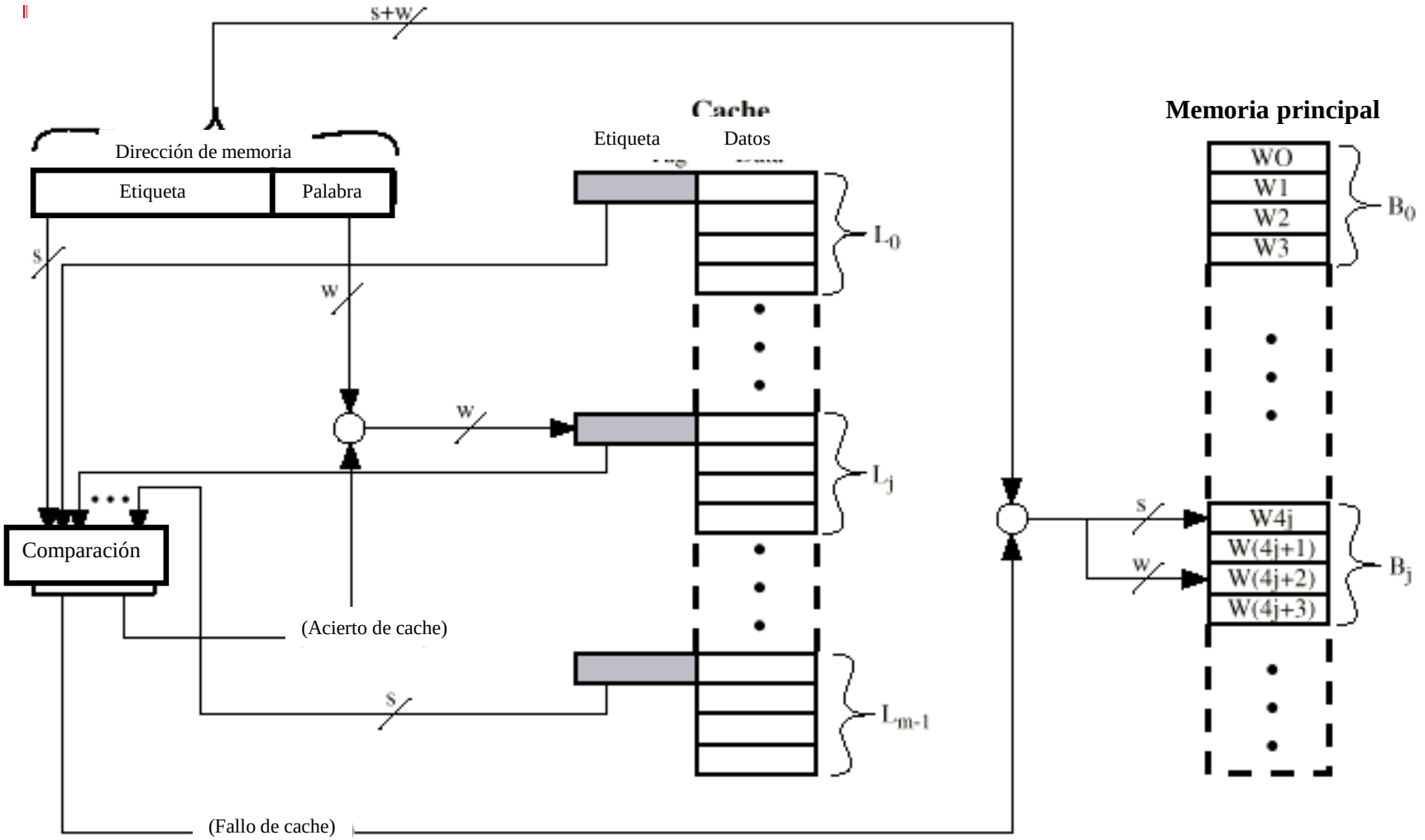
Ejemplo de correspon-dencia directa



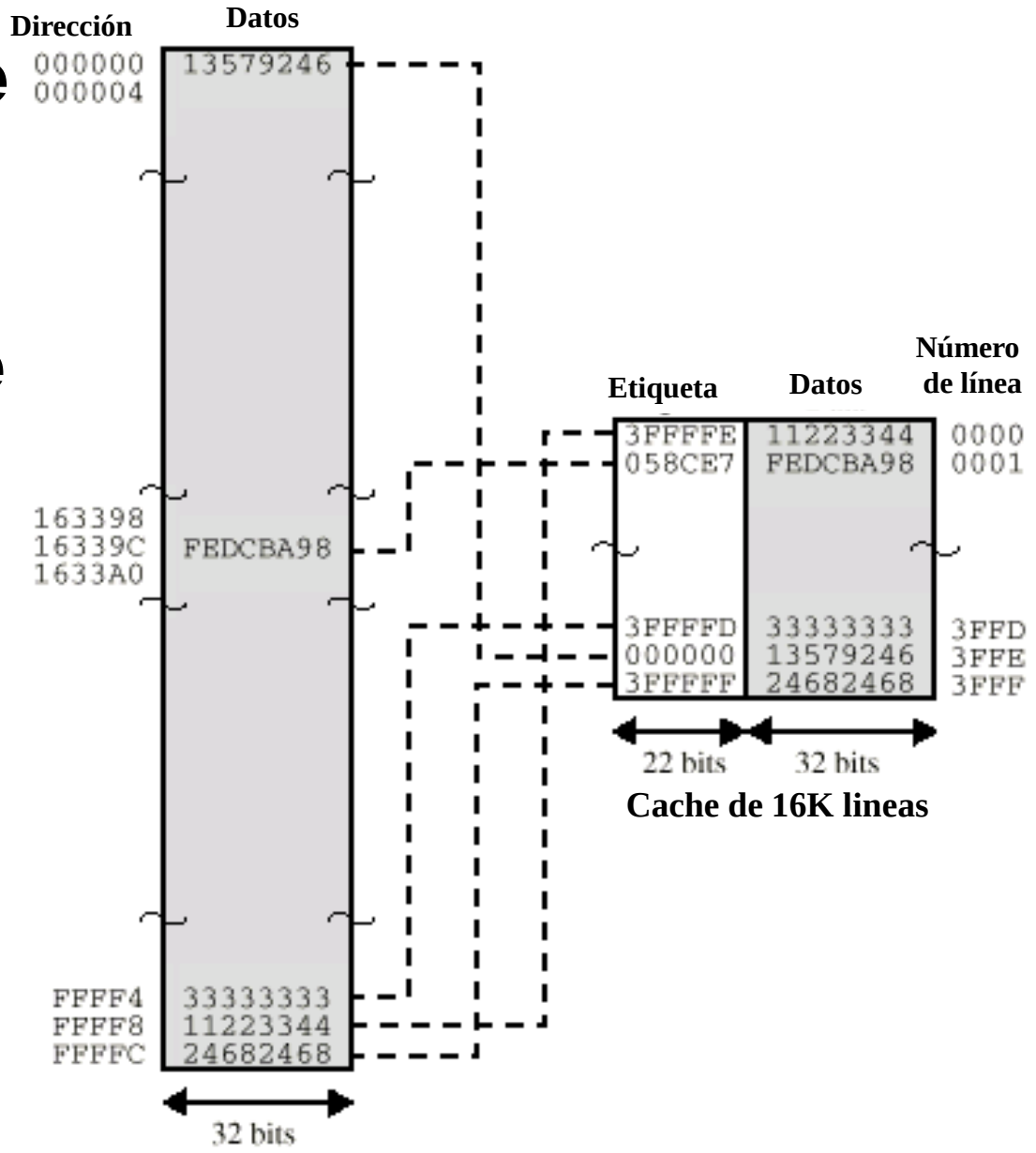
Correspondencia Directa: ventajas y desventajas

- Simple.
- Poco costosa.
- Hay una posición concreta para cada bloque dado:
  - si un programa accede a dos bloques que se corresponden a la misma línea (diferentes bloques de memoria principal) de forma repetida, las pérdidas de cache (desaciertos) serán muy grandes.

Organización de cache totalmente asociativa



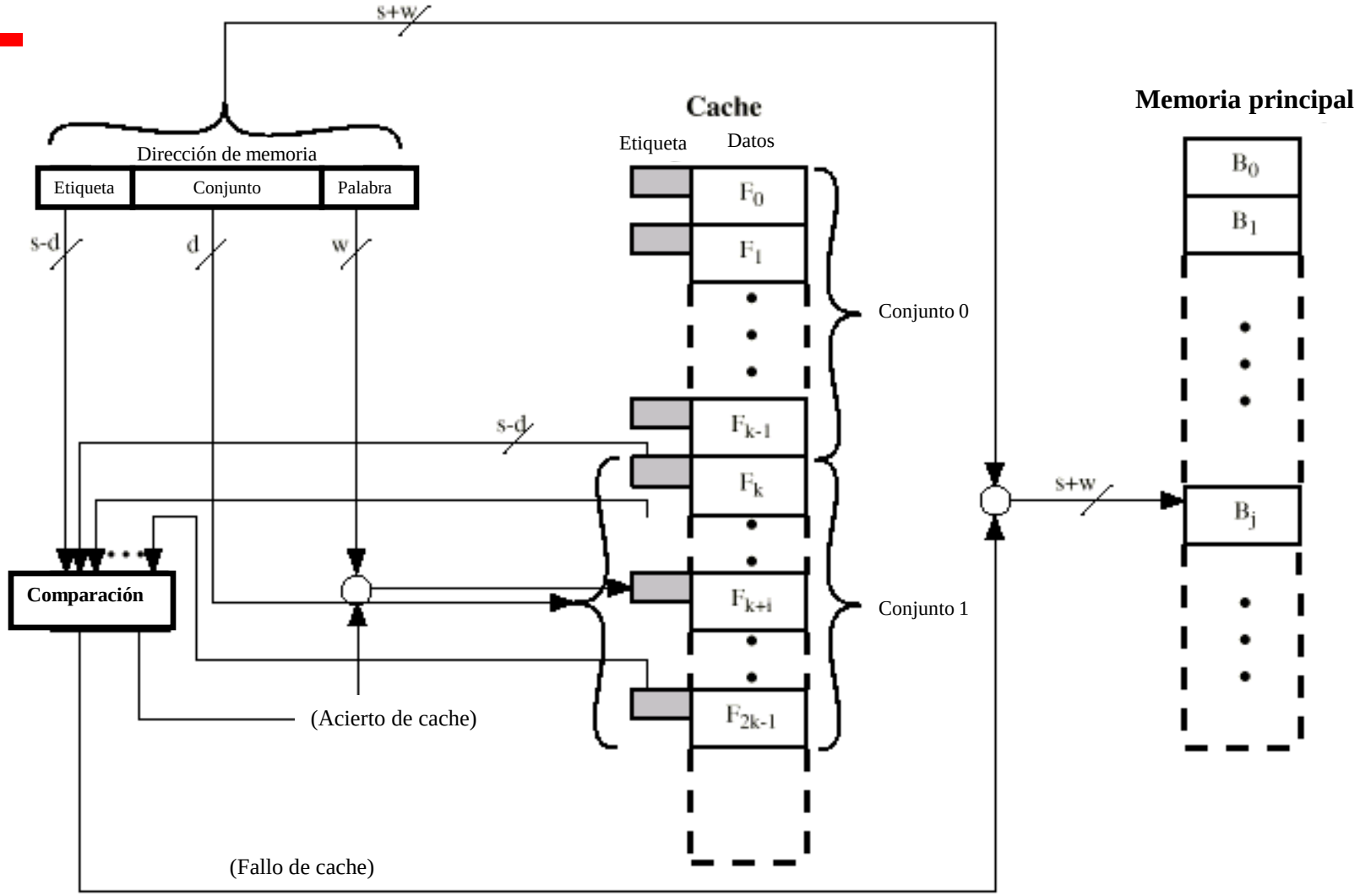
Ejemplo de correspon-dencia totalmente asociativa



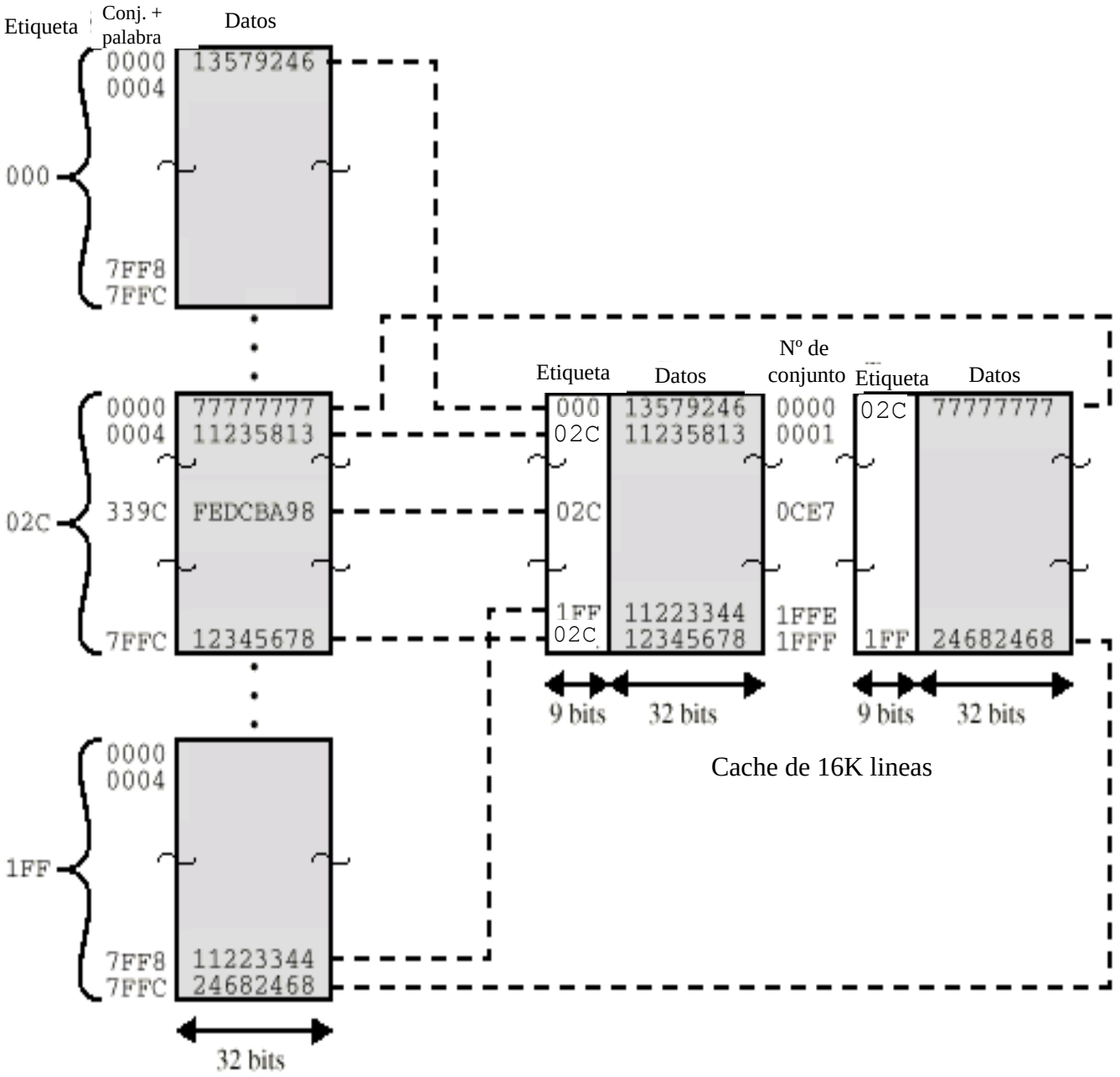
Correspondencia Asociativa: ventajas y desventajas

- Un bloque de memoria principal puede colocarse en cualquier línea de la cache.
- La etiqueta identifica unívocamente un bloque de memoria.
- Todas las etiquetas de las líneas se examinan para buscar una coincidencia.
- Búsqueda costosa (en tiempo principalmente).

Organización de cache asociativa por conjuntos



Ejemplo de correspon-dencia asociativa por conjuntos de 2 vías



Corres. asoc. por conjuntos: ventajas y desventajas

- Combina lo mejor de las otras correspondencias
- La cache se divide en un grupo de conjuntos.
  - Cada conjunto contiene un número de líneas
    - N vías, con N=2, 4, 8 ... etc.
- Un bloque determinado corresponderá a cualquier línea de un conjunto determinado.
  - El bloque B puede asignarse en cualquiera de las líneas del conjunto i.

Política de reemplazos

- Algoritmos de sustitución
  - En correspondencia directa:
    - el que ocupa el lugar del nuevo
  - En correspondencia asociativa:
    - LRU (menos recientemente usado)
    - FIFO (más antiguo)
    - LFU (menos frecuentemente usado)
    - Aleatoria

Algoritmos de sustitución

Correspondencia directa

- No hay elección.
- Sólo hay una posible línea para cada bloque.
- Se necesita una sustitución de esa línea (si o sí).

Algoritmos de sustitución (2)

Correspondencias asociativas

- Los algoritmos deben implementarse en hardware (para conseguir velocidad).

- Menos recientemente usado (LRU)
  - Requiere controles de tiempos
  - En correspondencias asociativas por conjuntos de 2 vías. ¿Cúal de las 2 líneas es la LRU?

Algoritmos de sustitución (3)

- Primero en entrar - primero en salir (FIFO).
  - Requiere controles de acceso.
  - Se sustituye aquella línea que ha estado más tiempo en la cache.
- Menos frecuentemente usado (LFU)
  - requiere controles de uso.
  - Se sustituye aquella línea que ha experimentado menos referencias.
- Aleatoria
  - Se sustituye una línea al azar.

Política de escritura

- Se debe evitar inconsistencia de memorias en el caso de escrituras.

Tener en cuenta:

- La CPU escribe sobre una línea de cache
  - El bloque de memoria principal correspondiente debe ser actualizado en algún momento.
- Un módulo E/S puede tener acceso directo a la memoria principal.
- En procesamiento paralelo, las múltiples CPU pueden tener caches individuales.

Política de escritura: en acierto

Write-through (*Escritura inmediata*).

- Se actualizan simultáneamente la posición de la caché y de la memoria principal.
  - con múltiples CPU, observar el tráfico a memoria principal para mantener actualizada cada cache local.
  - se genera mucho tráfico y retrasa la escritura.

Write-back (*Post-escritura*).

- La información sólo se actualiza en la caché.
  - Se marca como actualizada $\Rightarrow$ *bit de "sucio"*.
  - La memoria principal se actualiza en el reemplazo y puede contener información errónea en algún momento

Política de escritura: en fallo

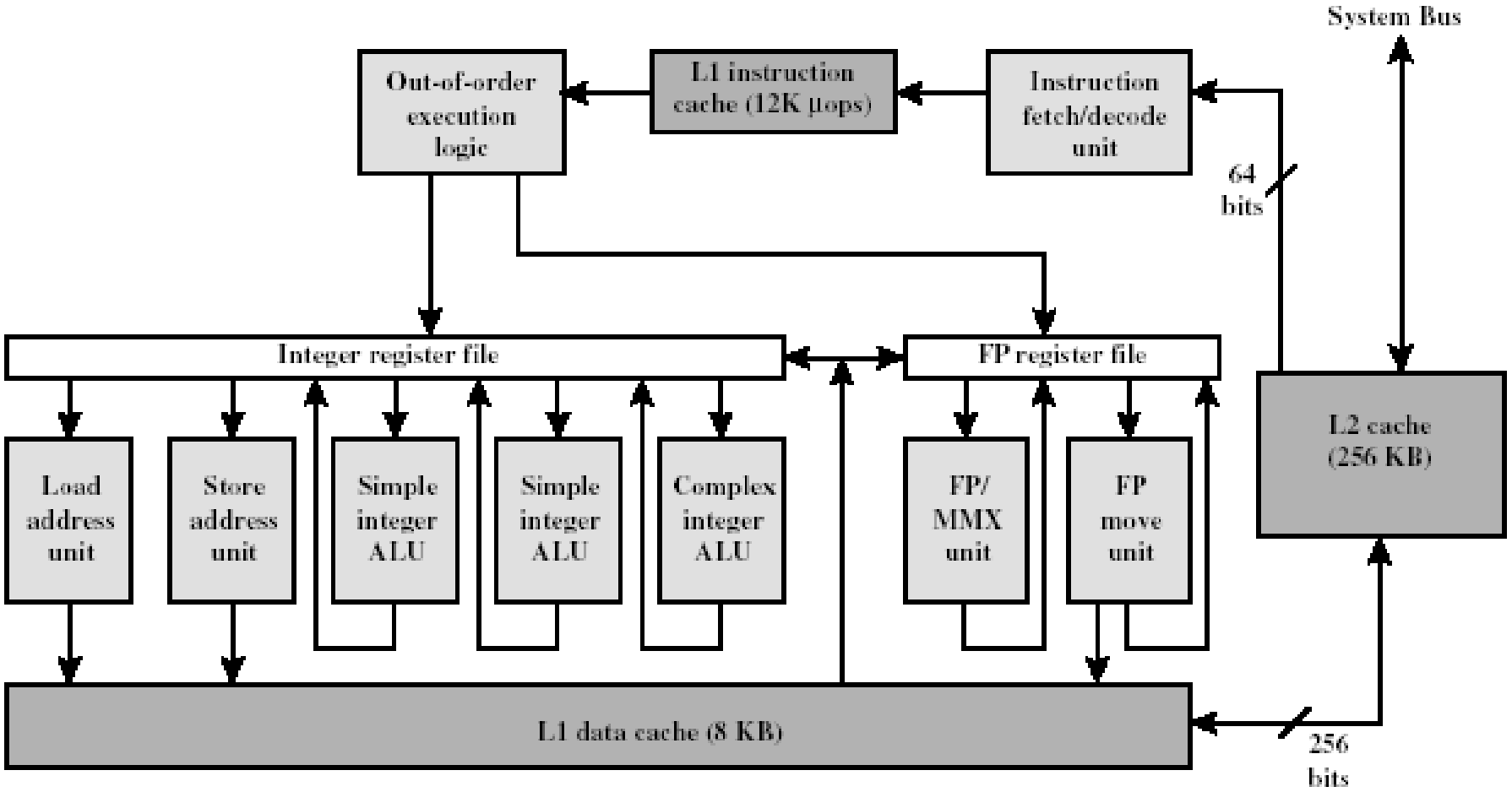
Write allocate

- La información se lleva de la memoria principal a la caché. Se sobreescribe en la caché
  - Habitual con write-back

No-write allocate

- El bloque no se lleva a la memoria caché. Se escribe directamente en la memoria principal.
  - Habitual con write-through

Pentium 4


System Bus
Out-of-order execution logic
L1 instruction cache (12K µops)
Instruction fetch/decode unit
64 bits
Integer register file
FP register file
L2 cache (256 KB)
Load address unit
Store address unit
Simple integer ALU
Simple integer ALU
Complex integer ALU
FP/ MMX unit
FP move unit
L1 data cache (8 KB)
256 bits

Pentium 4 (cont.)

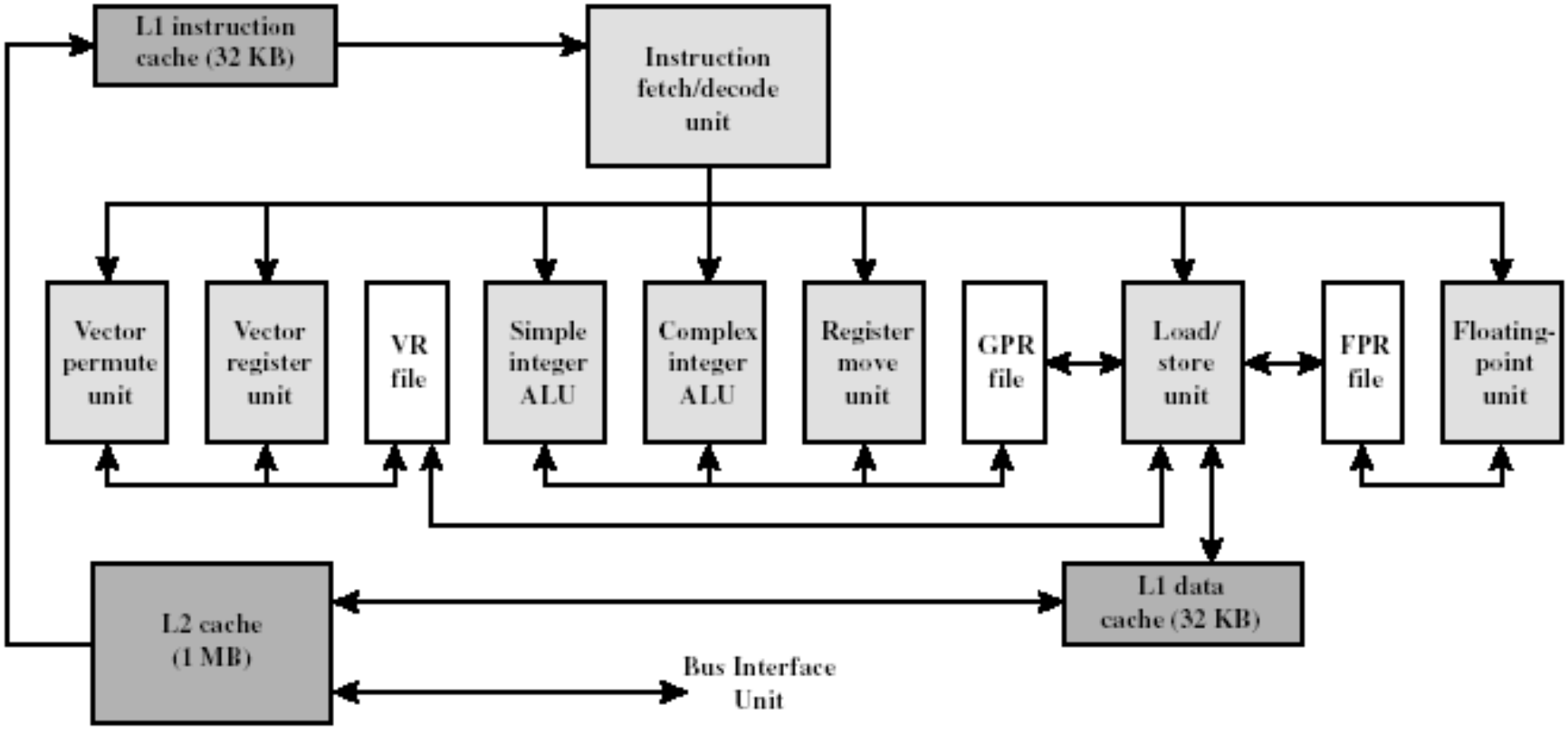
Puede tener hasta tres niveles de cache:

- Caches L1 separadas para datos e instrucciones
  - Cache de datos (de 8 KBytes). Asociación por conjuntos de 4 vías. Bloques de 64 bytes. Política de *Escritura inmediata.* Acceso a los datos enteros en dos ciclos de reloj.
  - La caché de instrucciones almacena segmentos de caminos de ejecución de instrucciones decodificadas (*trazas*).

Pentium 4 (cont.)

- Cache L2 (interna) unificada para datos e instrucciones
  - Capacidad de 256 KBytes. Organización asociativa por conjuntos de 8 vías. Bloques de 128 bytes. Política de *Post-escritura.* Latencia de acceso de 7 ciclos de reloj.
- Las dos caches (L1 y L2) en el chip del procesador. Ancho de banda de las transferencias entre L1 y L2 48 GBytes/s.
- La arquitectura admite un tercer nivel de caché (L3) en el mismo chip (servidores).

Power PC G3


L1 instruction cache (32 KB)
Instruction fetch/decode unit
Vector permute unit
Vector register unit
VR file
Simple integer ALU
Complex integer ALU
Register move unit
GPR file
Load/ store unit
FPR file
Floating-point unit
L2 cache (1 MB)
L1 data cache (32 KB)
Bus Interface Unit

Referencias

- Organización y Arquitectura de Computadoras, W. Stallings, Capítulo 4. 5º edición.
- *Diseño y evaluación de arquitecturas de computadores*, M. Pardo y A. Guzmán, Capítulo 2 (secciones 2.1 a 2.4). 1º edición.