Conceptos de Arquitectura de Computadoras

Clase 7

Memoria

Sistema de Memoria

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

Ya no es la memoria, sino es un sistema de memorias que para la CPU es una sola memoria pero en la práctica hay distintos niveles de almacenamiento construidos x distintas tecnologias que se gestionan de distintas maneras y que deben funcionar como una sola.

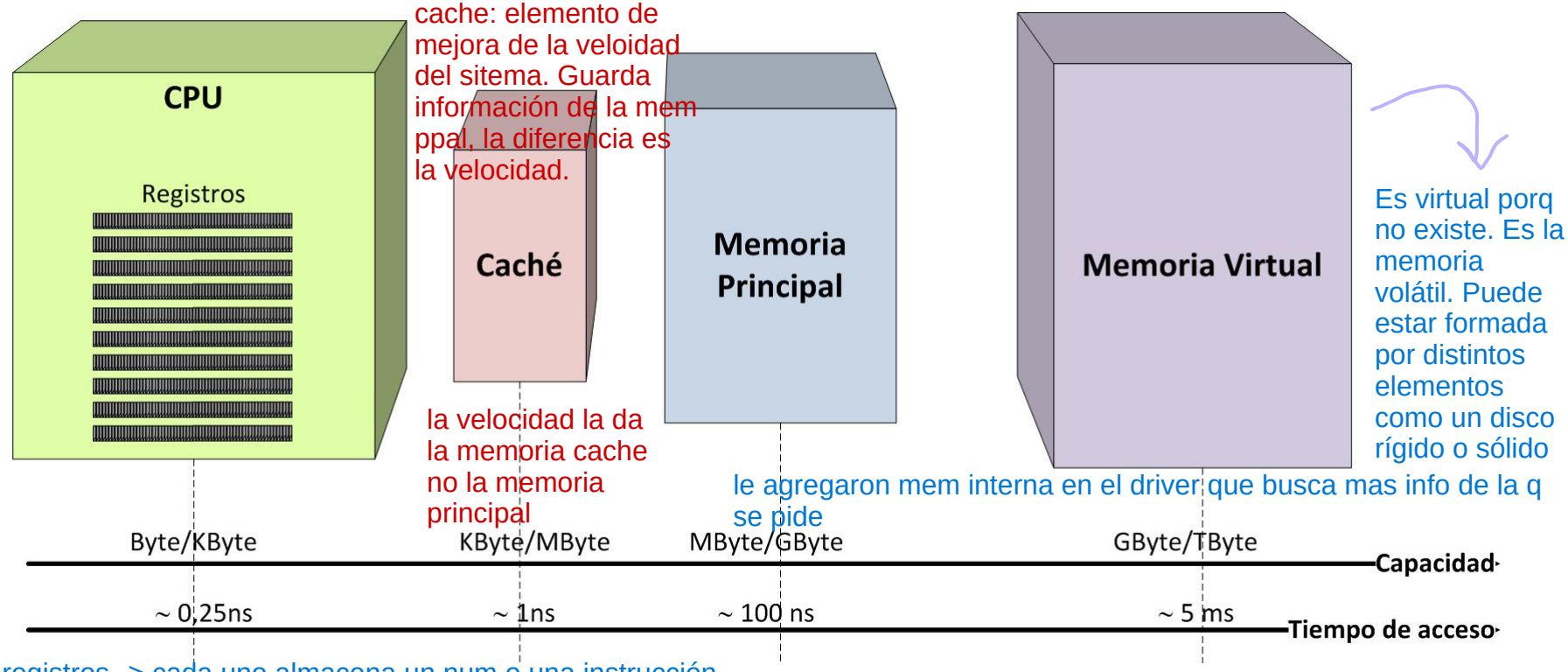
Jerarquía de memoria

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

Jerarquía de memoria

CPU: tiene una estructura de registros internos que sirve no solo para instrucciones sino que serven para la estructura de memoria que vamos a tener.

Memoria ahora es Memoria principal (RAM): sirve para todas las memorias que son de lectura y escritura.
Nuestra Mem Ppal es un nudo que representaria la memoria principal de vonNeumann. En esta hay que poner el código en binario del programa y los argumentos/variables que se necesitan para el código
Es mucho más lenta que el acceso a los registros. Por lo tanto se agregan mas registros (32). En medio ciclo de la velocidad de la CPU puedo acceder al registro y en el otro medio ciclo se hace otra tarea.



cache: elemento de mejora de la veloidad del sitema. Guarda información de la mem ppal, la diferencia es la velocidad.

la velocidad la da la memoria cache no la memoria principal

Es virtual porq no existe. Es la memoria volátil. Puede estar formada por distintos elementos como un disco rígido o sólido

le agregaron mem interna en el driver que busca mas info de la q se pide

32 registros--> cada uno almacena un num o una instrucción

depende cuantas palabras procese el procesador --> si tengo 8gb y tenemos el procesador winmips que es de 64bytes, tendria 1g por palabra CREO

Características: se necesitan 4gb o mas para el almacenamiento

La Mem. Virtual: cuanto cuesta en tiempo traer una palabra de la mem virtual --> 10-20milisegs para cualq cantidad de bytes que

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.

costo --> costo de un bit. Lo q se plantea de costo de un bit en los niveles el costo de un bit en registros NO tiene que ser el costo de un bit en mem ppal

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 lo que tengo en todos los lugares tiene el mismo valor
  - Las copias de la misma información en los distintos niveles deben contener los mismos valores.

PREGUNTA DE FINAL/PROMOCIÓN

¿Porqué funciona la jerarquía?

Por el principio que refleja las maneras en las q venimso programando desde siemrpe. LOCALIDAD DE REFERENCIIAS --> se tiene que tener un prog escrito en forma secuencial, es decir hay un orden en la ejecución de las instrucciones. Este orden es el mismo que la máquina las busca y las cumple. Esto lleva a que cuando estoy ejecutando un programa la CPU va a un valor y lo mas probable es q ese valor se va a volver a utilizar en un tiempo cercano

Principio de localidad de referencias

Lo que conviene es siempre hacer una copia de esa palabra en cada nivel de la jerarquia para tener más velocidad!!!!!!!

- **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 DMA

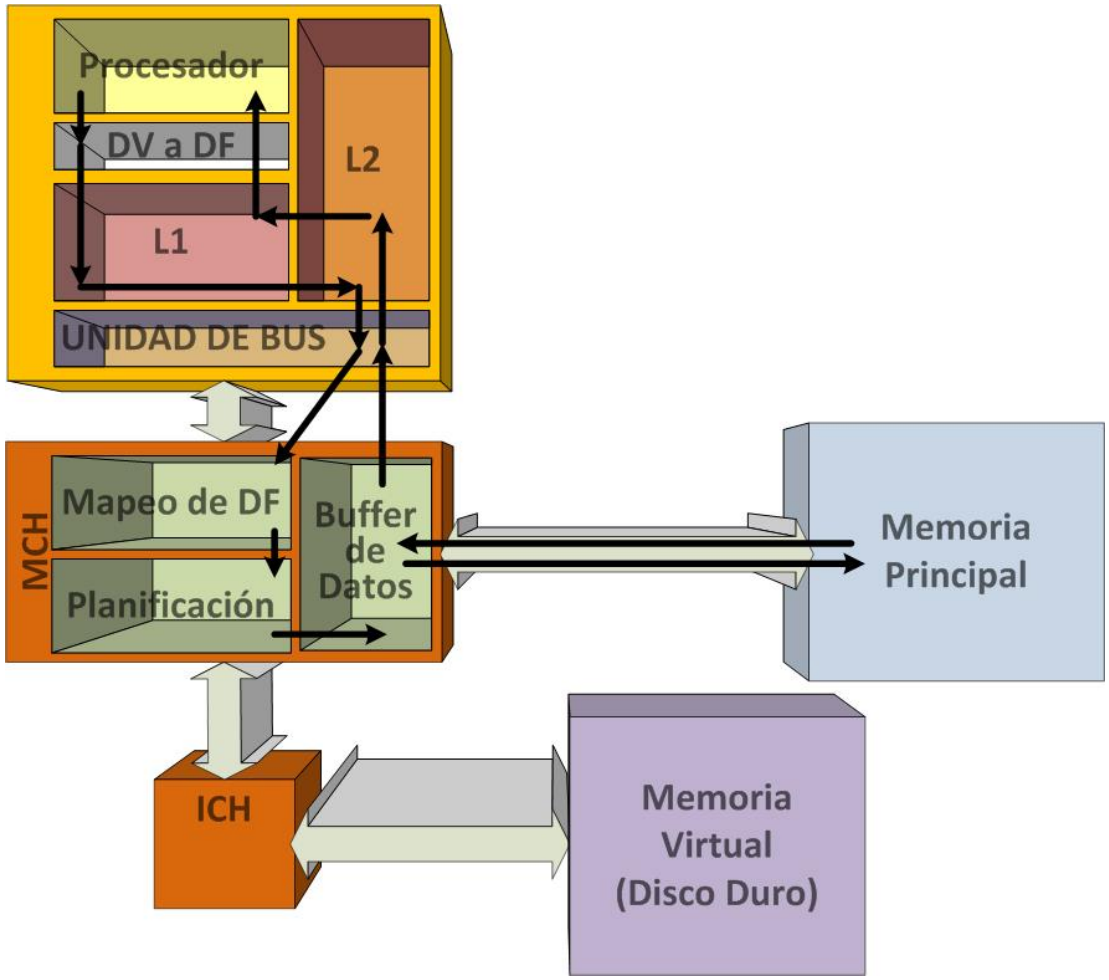
Basada en como escribimos los programas y la estructura de datos.
Copia un espacio de almacenamiento que contiene la palabra.
todas las máquinas tienen 4 canales de DMA trabajando cuando la compu se esta utilizando

Como se copia en la cache algo de mem ppal? Con un hardware específico sin intervención de la CPU

Como funcionan los procesadores actuales--> tiene dos niveles de cache, una mem ppal y los discos ridigos

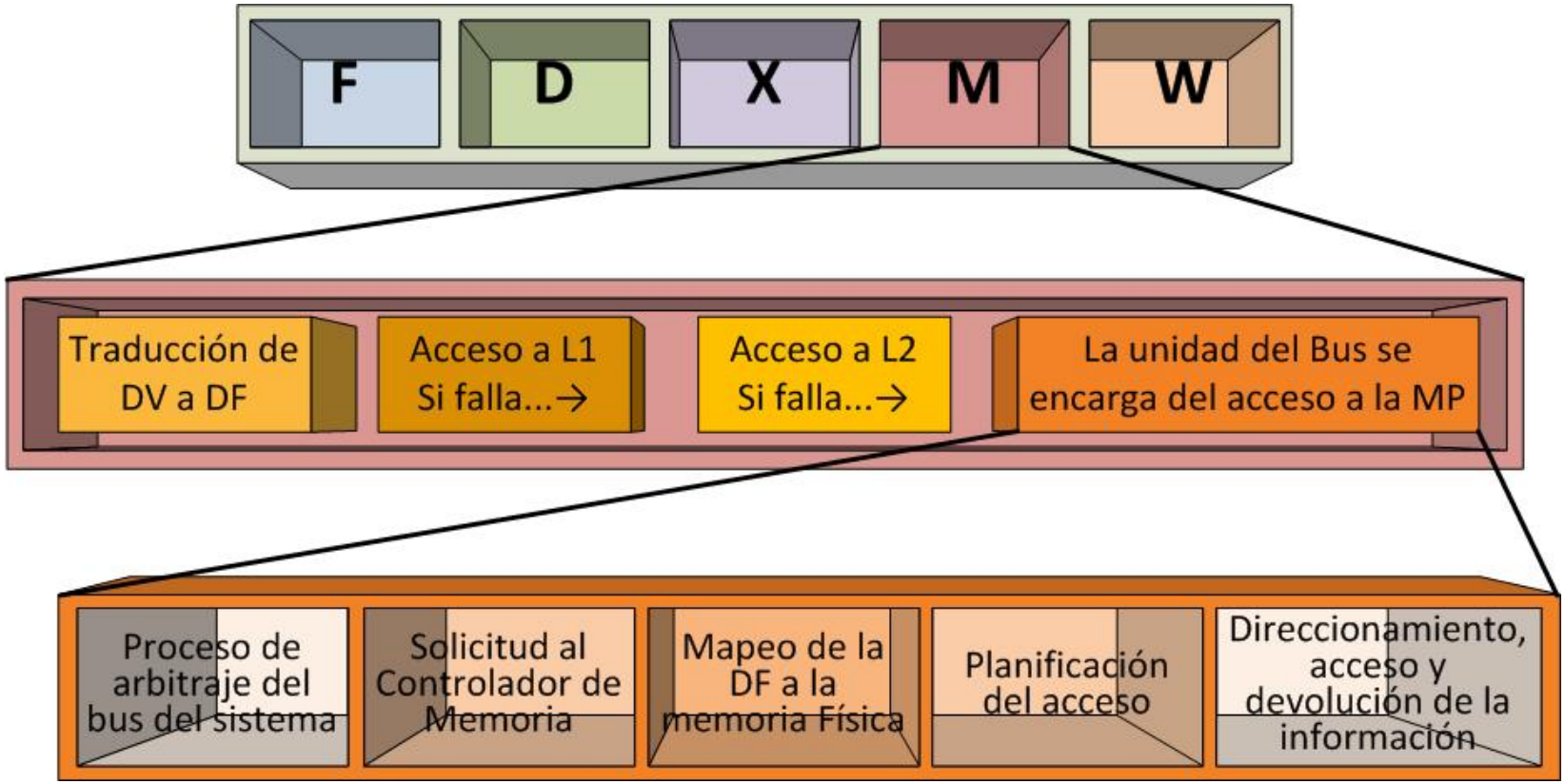
Mecanismo de acceso memoria

La CPU pide un dato y la mem virtual le envía una dirección VIRTUAL. Ahí es donde entra el hardware especifico en donde convierte esta dirección virtual en algo "real" para enviarlo a la CPU.
La CPU pone direc virtual, se convierte a direc fisica y espera rta.



concentrador--> concentra las interacciones con la mem ppal. Esta interconexión tiene muchas tareas. Por ej la direc fisica que busca el procesador puede estar modificada por el sistema operativo

Trabajo de etapa M del cauce



Memoria Cache

memoria pequeña en tamaño de almacenamiento con una velocidad de rta muy parecida al tiempo de rta que tienen los registros internos de la CPU

memorias RAM(lectura y) vs memorias ROM (solo lectura)

memorias dinámicas --> basadas en transistores. Almacenando en ellos la carga electrica que representa la info guardada. Se refrescan para cambiarle los valores almacenados esto es lo que las hace mas lentas

Sl proceso de refresco es generaciond e las direcciones de memoria de las celdas del dispositivo que estamos utilizando.

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

Se generó en las memorias un sistema automático que tiene un contador desde 0 al max valor posible que va generando direcciones adentro de la mem como si alguien las quisiera leer. Esto logro que la memoria no pueda ser utilizada por los elementos externos a la memoria.

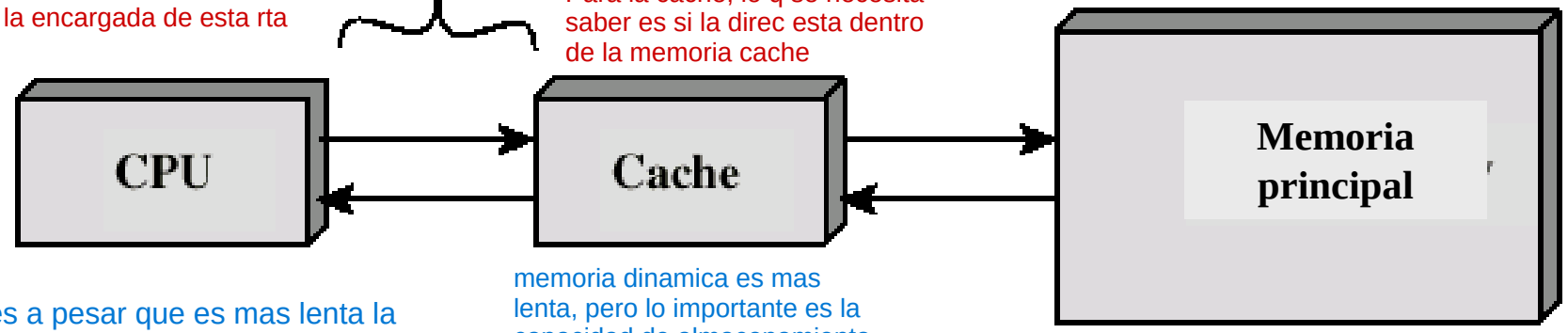
Nuevo sistema. Acceso por contenido.
En vez de enviarle la direc, la CPU pregunta si la direc esta almacenada en la memoria, la cache es la encargada de esta rta

Transferencia de bloques

Respoden a un modelo direccion-contenido. Definida esa direc se hace la operacion de lectura o escritura

Transferencia de palabras

Para la cache, lo q se necesita saber es si la direc esta dentro de la memoria cache



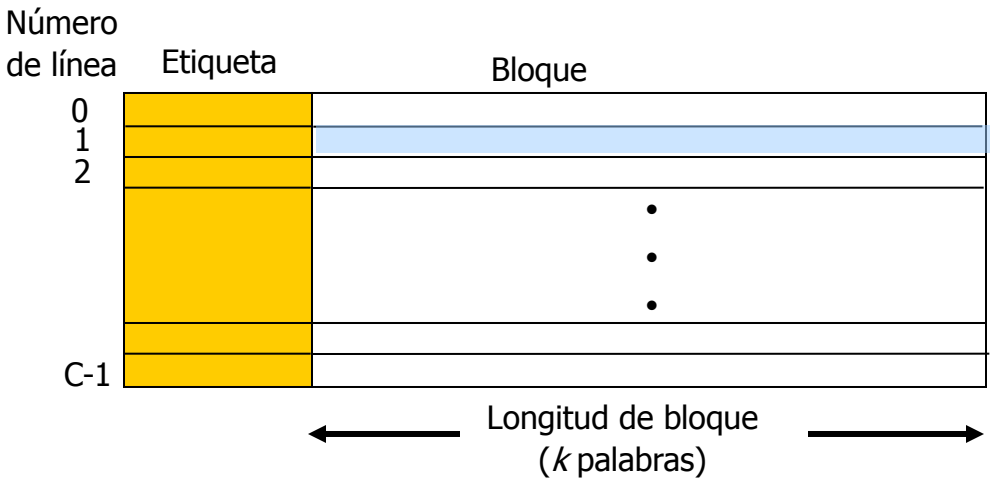
memoria dinamica es mas lenta, pero lo importante es la capacidad de almacenamiento

Entonces a pesar que es mas lenta la mem dinamica lo que buscamos es MAYOR cant de almacenamiento. Ademas el tiempo de ejecución mejoró.

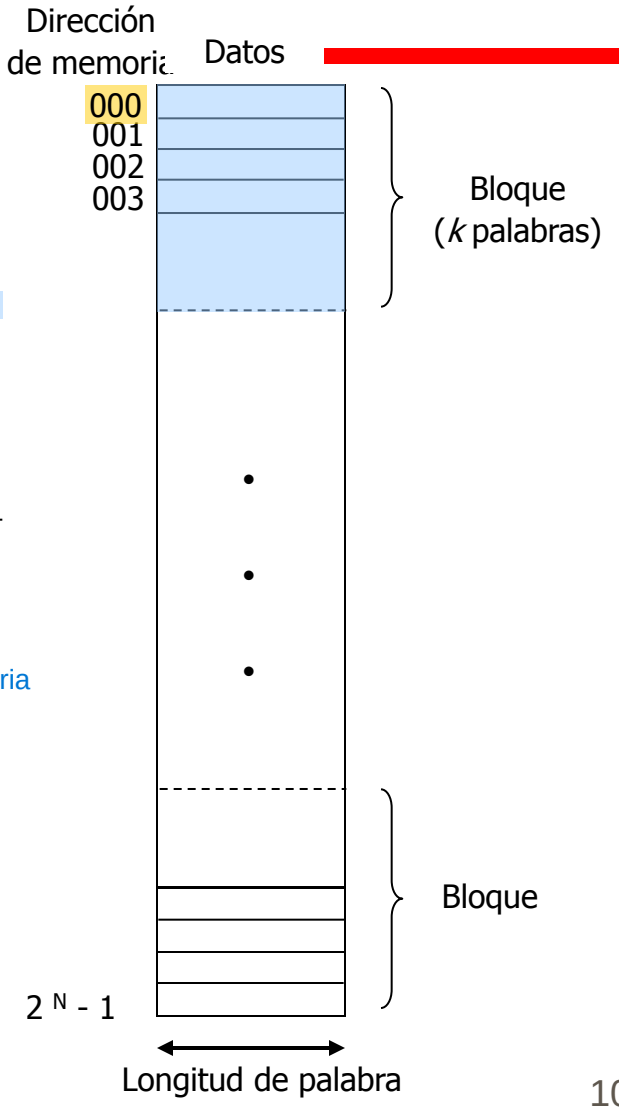
memoria estatica tiene la X velocidad

Memoria Cache y Principal

La cache pregunta si la direc que tiene la etiqueta esta en algun bloque de palabras de la memoria



cuando hablamos de linea es de la memoria cache--> etiqueta y bloque de memoria
cuando hablamos de bloque es de la memoria



Funcionamiento de la cache

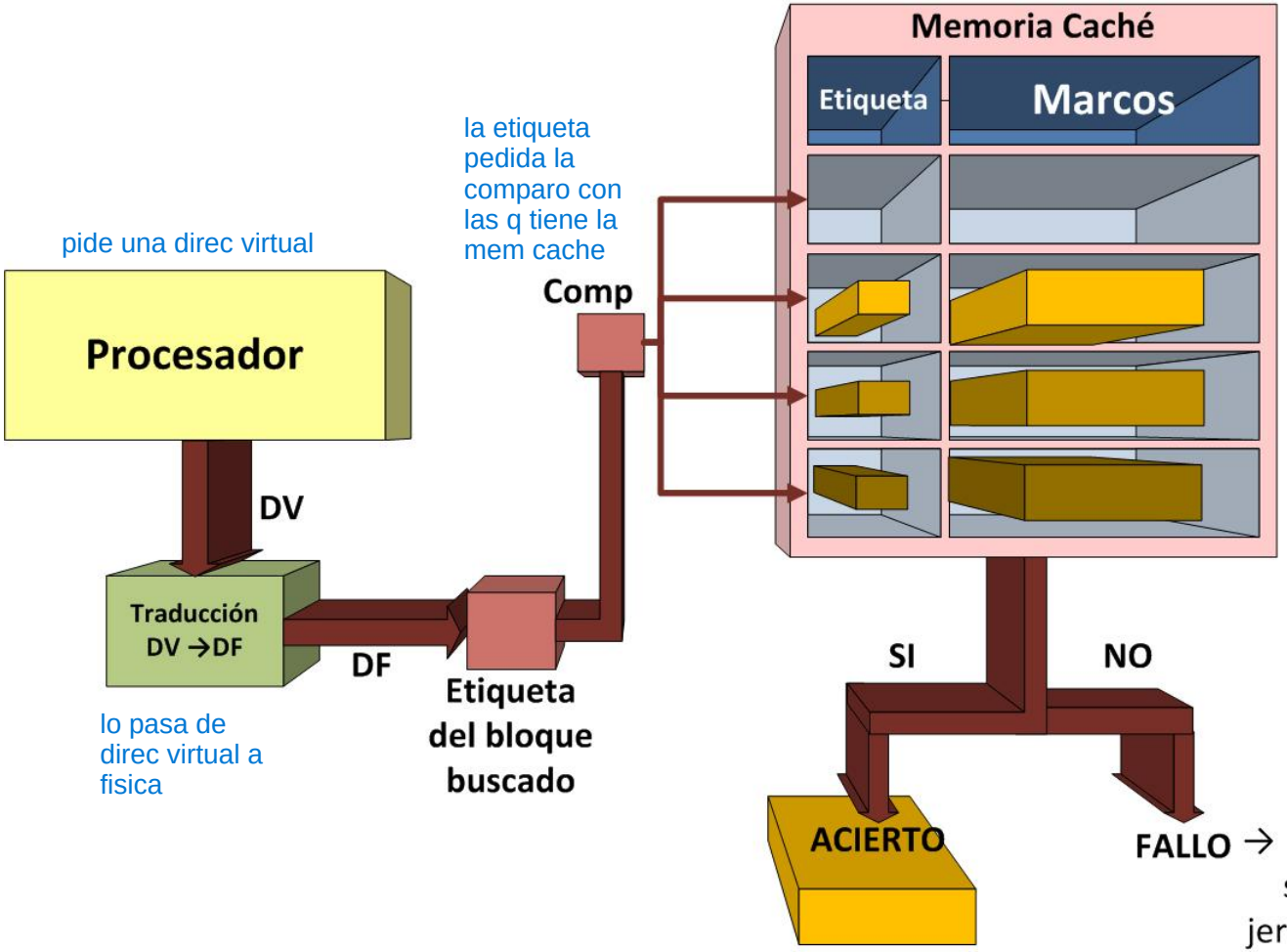
- La CPU solicita contenido de 1 dirección de memoria.
- La cache ¿tiene ese dato?
  - Si es así, la obtiene de la cache (rápidamente).
  - Si no está, se lee el bloque que contiene esa dirección desde la memoria principal y copia en la cache.
    - Después, la cache entrega el dato requerido a la CPU.

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

si la direc esta almacenada en la cache, devuelve el dato inmediatamente. La memoria practicamente no se entera

Si no esta, se dispara el proceso de actualizar los valores

Funcionamiento de la cache (2)



Conceptos básicos

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

el dato correspondiente a la direc que puso la CPU en el sistema esta dentro de la cache

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.

si no esta hay q acceder a mem ppal y copiar ese dato en un bloque. Esto nos lleva al tiempo para servir un fallo

- ***Tiempo para servir un fallo:*** depende de la latencia y ancho de banda de la memoria principal.

tiempo para copiar los valores

el tiempo necesario para suponer que tome un valor y dsp tome otro

tiempo calculado que maneja los accesos sucesivos

  - ***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.

para servir un fallo tengo q ir a memoria, copiar los datos en forma sucesiva

- Los **fallos de caché** se gestionan mediante hardware y causan que el procesador se detenga hasta que el dato esté disponible.

ancho de banda y latencia: nos condicionan el tiempo necesario para cargar la cache

Prestaciones de la jerarquía

Tiempo de acceso medio a memoria

tiempo de acierto + tiempos que fallo

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

valor que depende de la aplicacion que tengamos

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

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

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).

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

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

$$t = nI \,.\, CPI \,.\, T = 100 \,.\, 1 \,.\, (1/2G)$$

100 instrucciones; 1 ciclo de instruccion

$$t = 50ns$$

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.

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

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

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

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

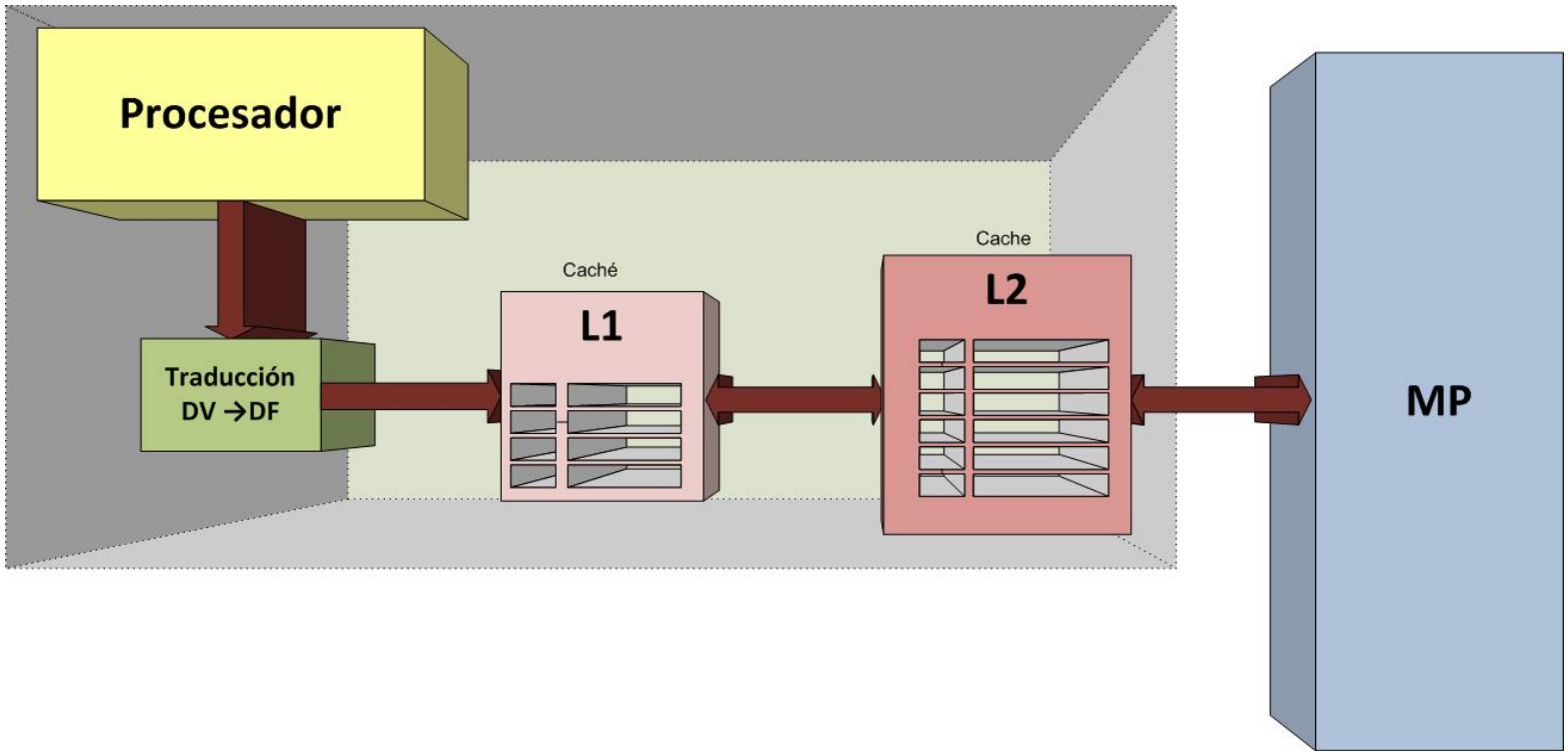
$$t = t_{cpu} + t_{mem} = 622{,}5ns$$

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

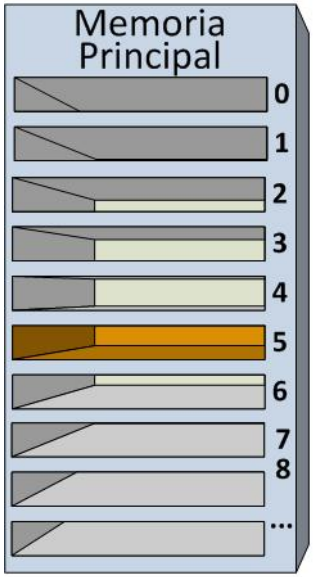
- Correspondencia **totalmente asociativa.** Un bloque puede almacenarse en cualquier lugar de la caché.
- Correspondencia **asociativa por conjuntos.** Un bloque puede almacenarse en un conjunto restringido de lugares en la caché.

  Un conjunto es un grupo de lineas de la caché.

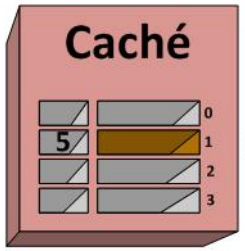
  Nº conjunto = Nº bloque ref. ***mod*** Nº conjuntos caché

Tipos de correspondencia

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

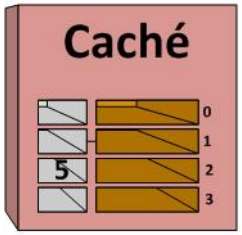


Directa

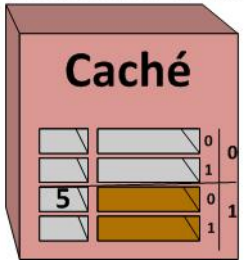


Bloque 5 mod 4 líneas = 1

Asociativa



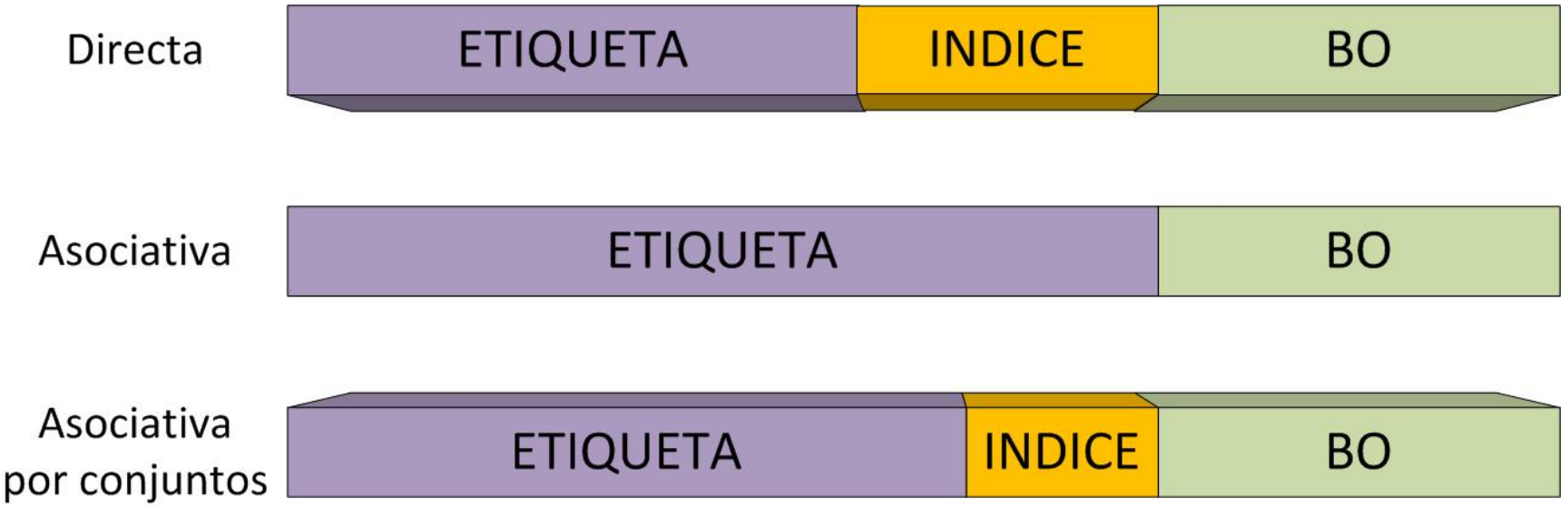
Asociativa 2 vías



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)

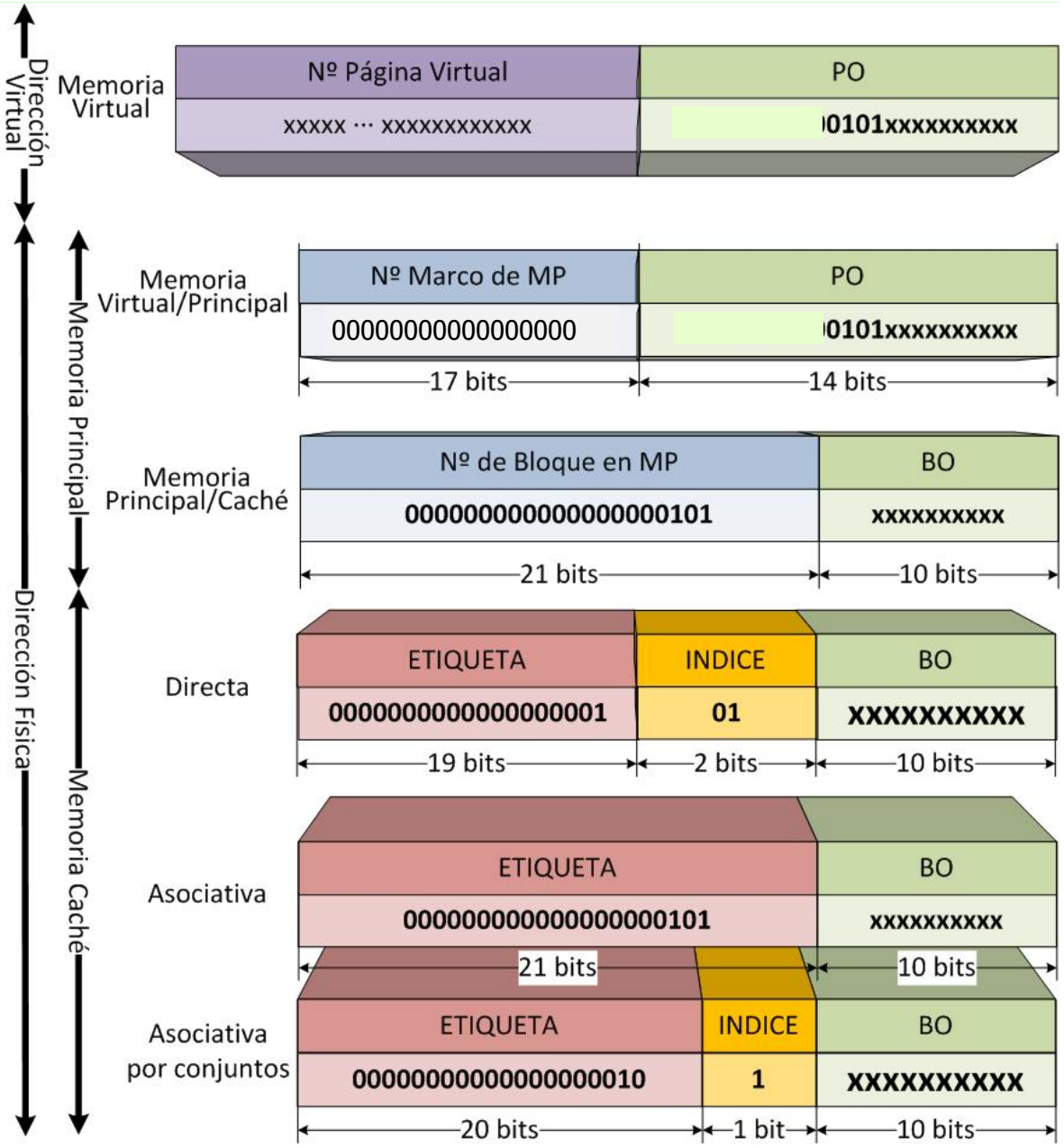
Supongamos que en el ejemplo anterior:

MP es de 2GB

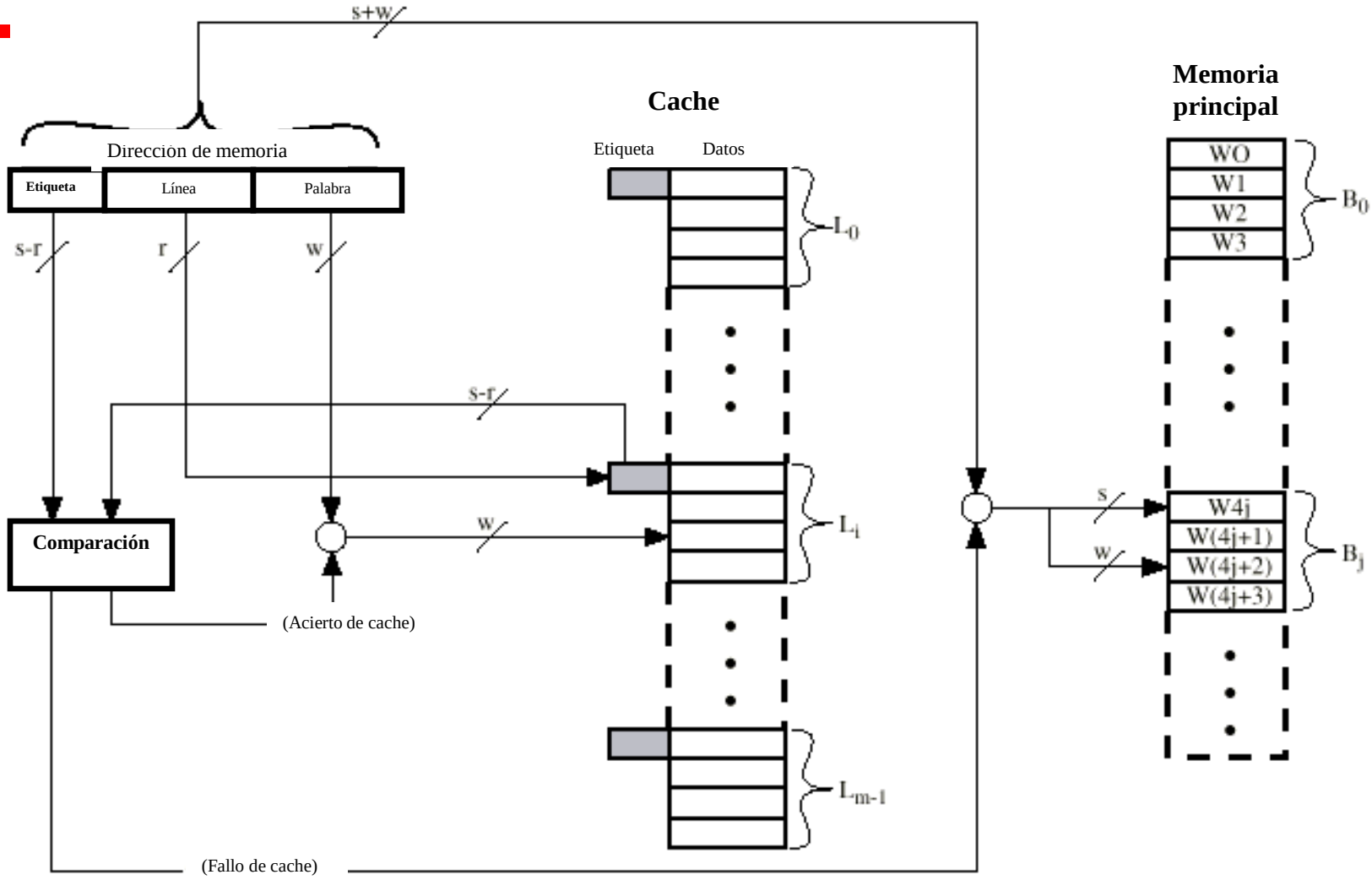
Páginas de 16KB

Bloques de 1KB

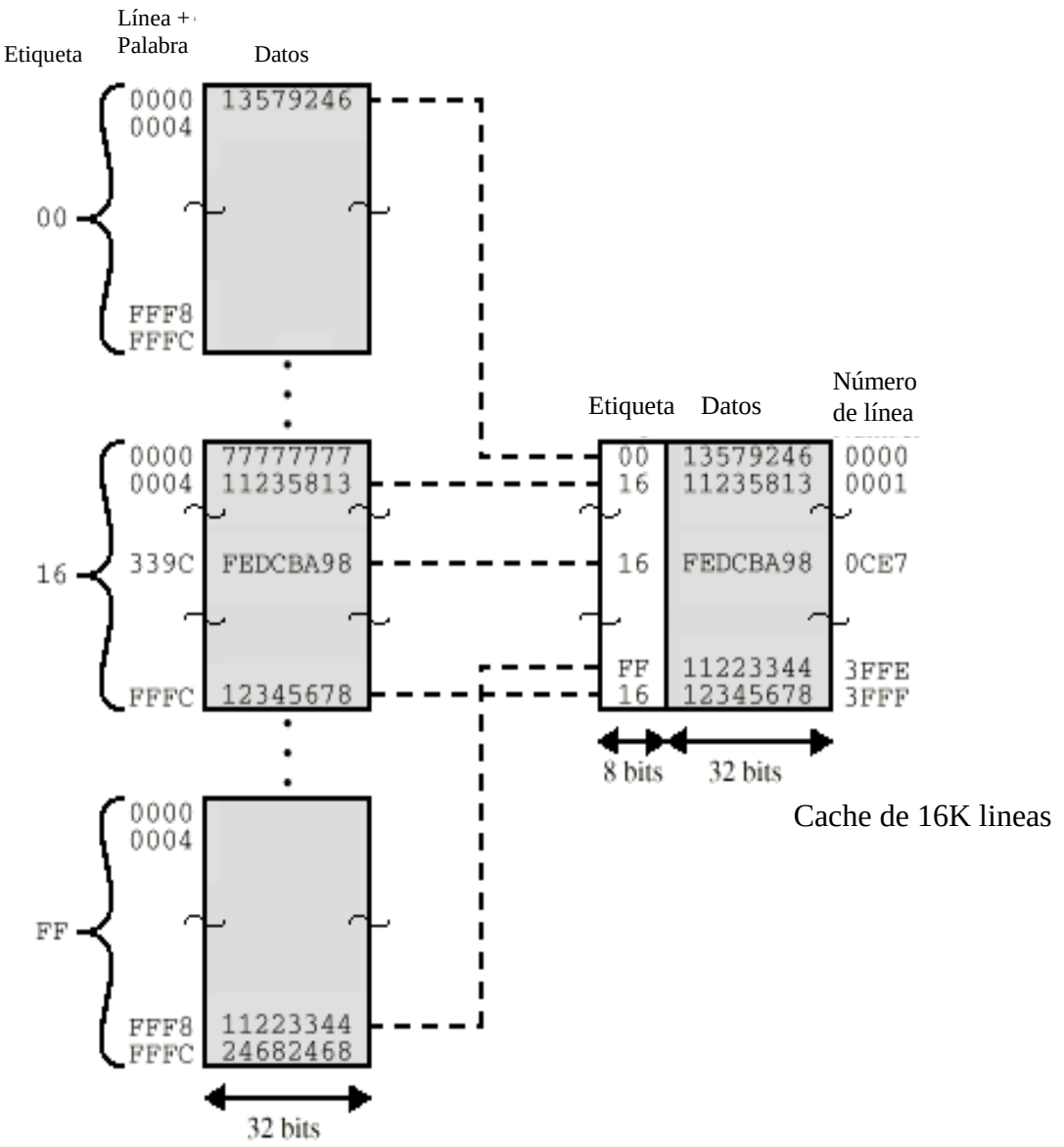
Cualquier dirección del bloque 5 es como en la figura



Correspondencia directa: Organización de cache



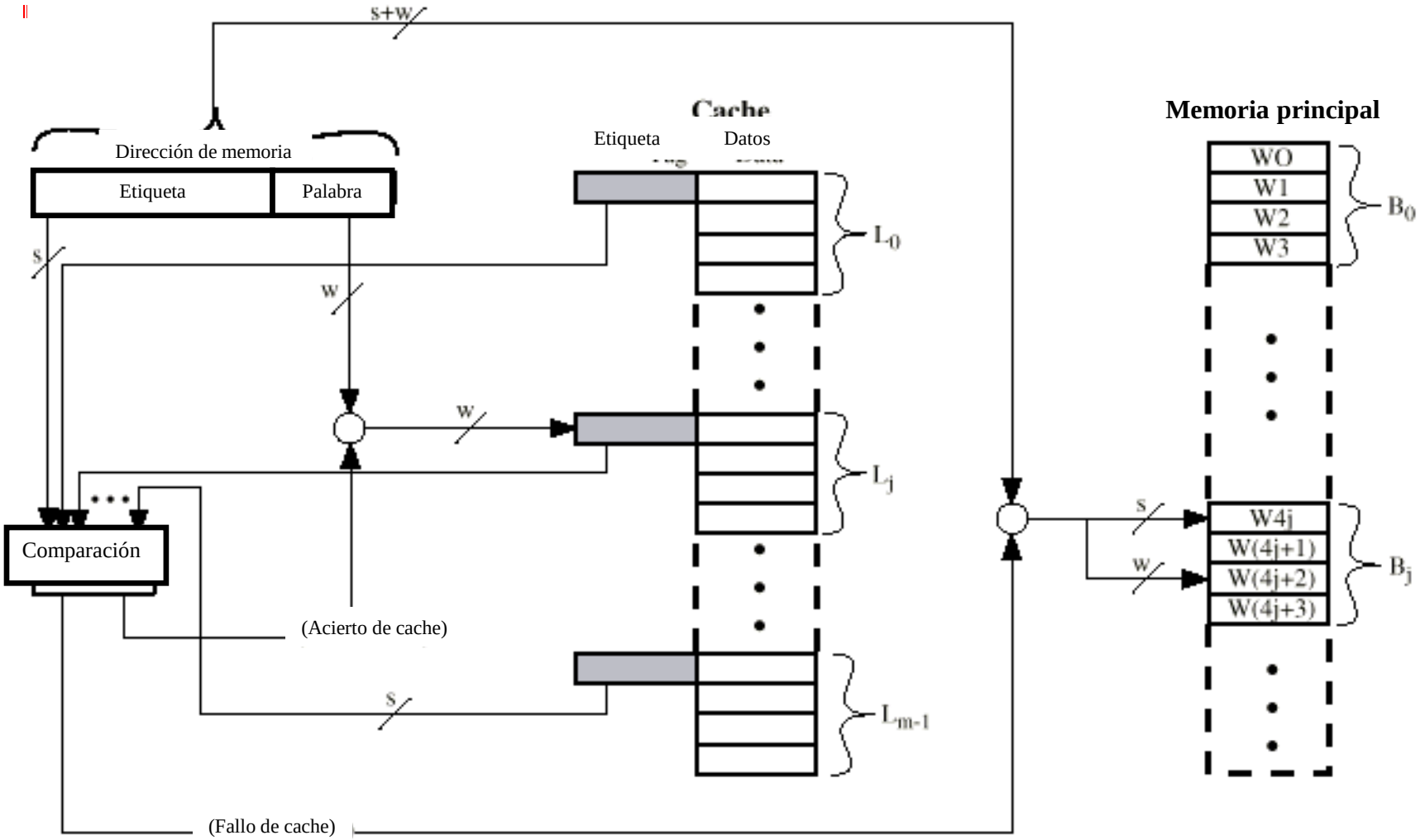
Ejemplo de correspon-dencia directa



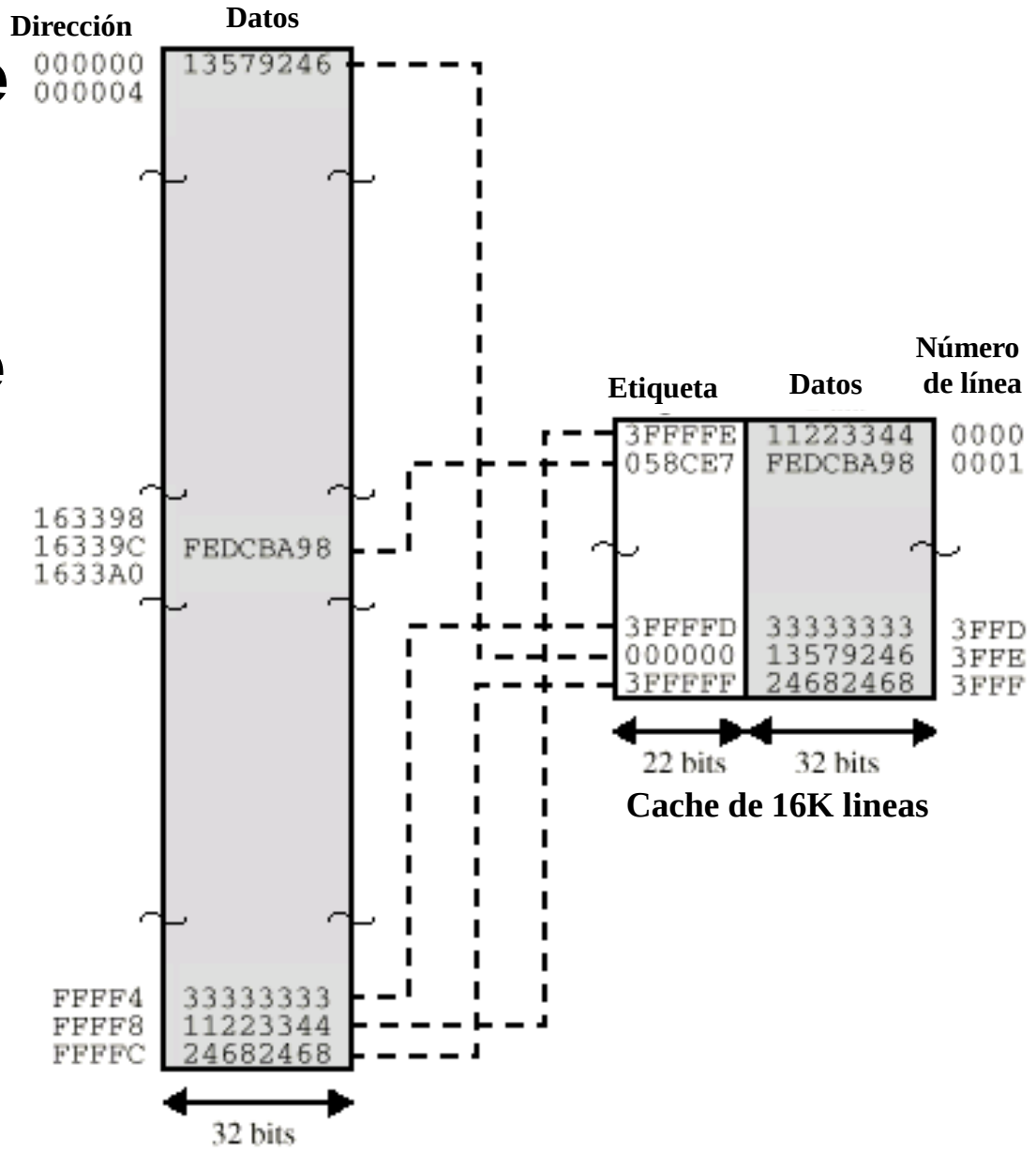
Correspondencia Directa: ventajas y desventajas

- 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



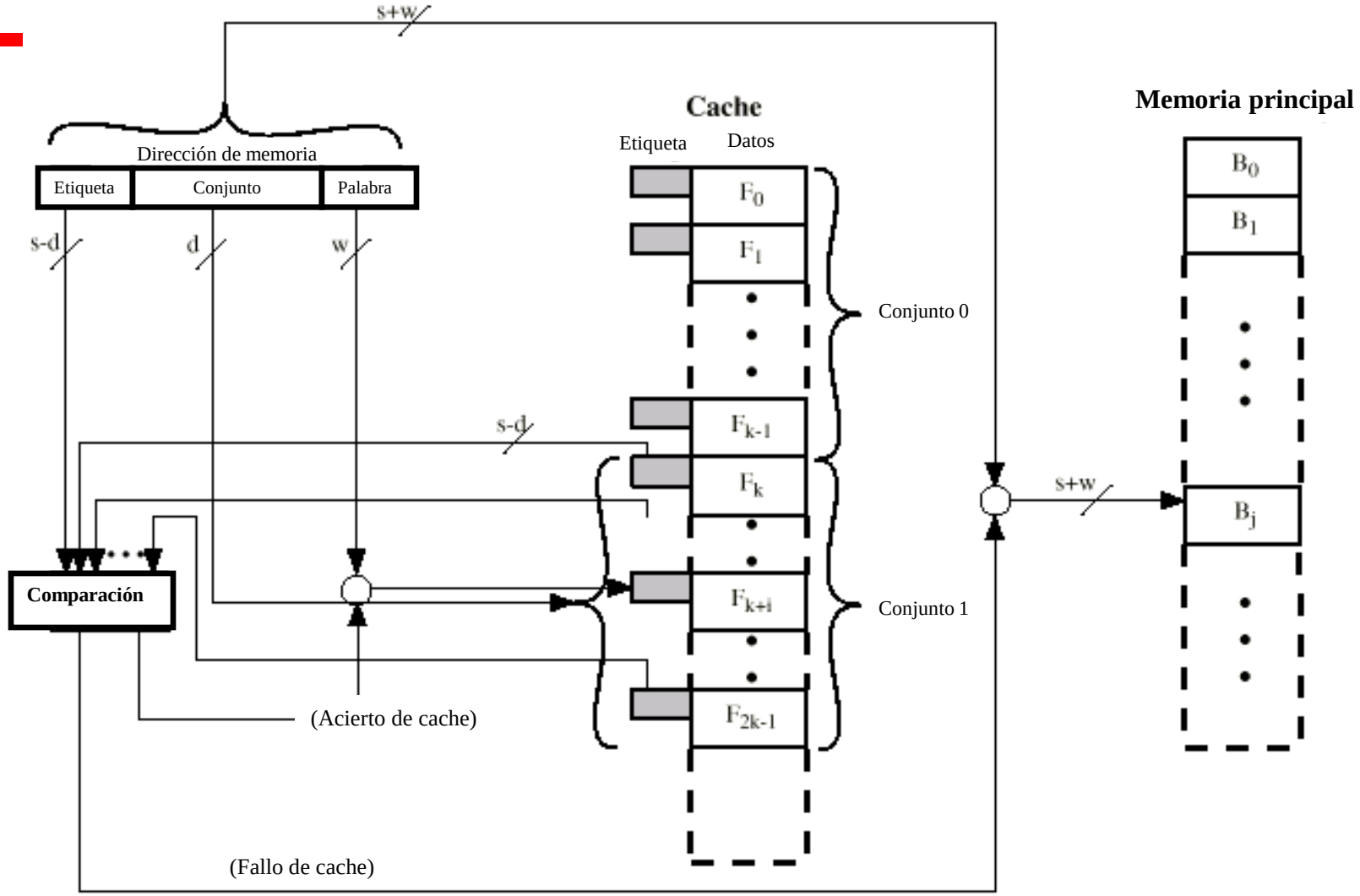
Ejemplo de correspon-dencia totalmente asociativa



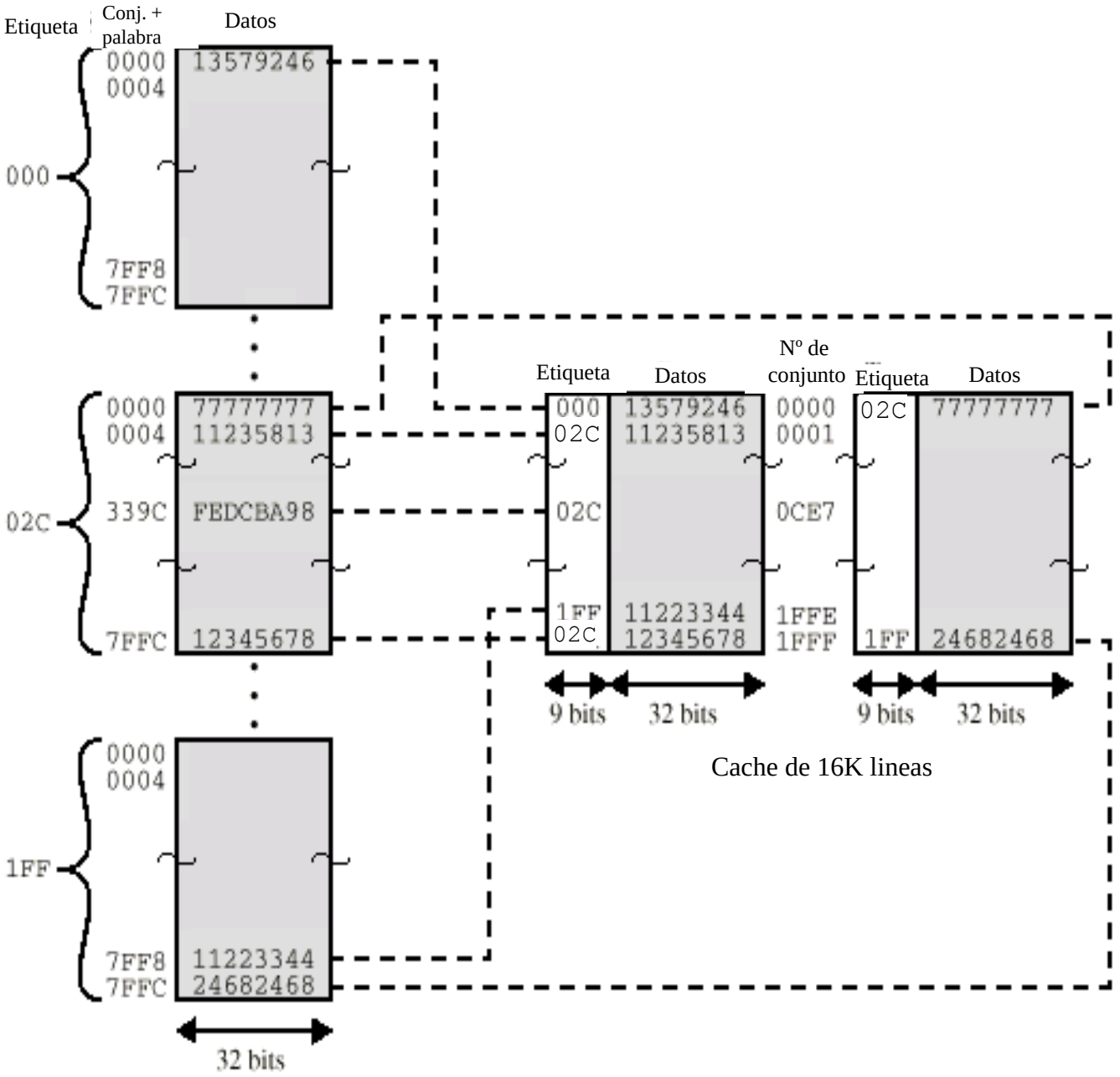
Correspondencia Asociativa: ventajas y desventajas

- 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?

cuando se tiene mas de una via se le pone el control de tiempo a uno y luego los demas se cuentan a partir del primero q se puso.

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.

cada uno q entra le pongo estampa de tiempo, el q mas tiempo esta es el primero en entrar

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*).

lo escribo en la cache y al mismo tiempo se escribe en la memoria principal

- 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*).

no se tiene coherencia de datos. Como sabemos que vamos a escribir dsp en memoria ppal.

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

Axctualizo datos en mem ppal cuando la linea que quiera sobreescribir fue modificada. Para eso la line de la cache sobre la q la CPU escribio un dato nuevo, la marca con un bit de "sucio".

Política de escritura: en fallo

Write allocate

lo traigo de cache xq no estaba y le pongo el bit de sucio para respetar lo q hace el writeback

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

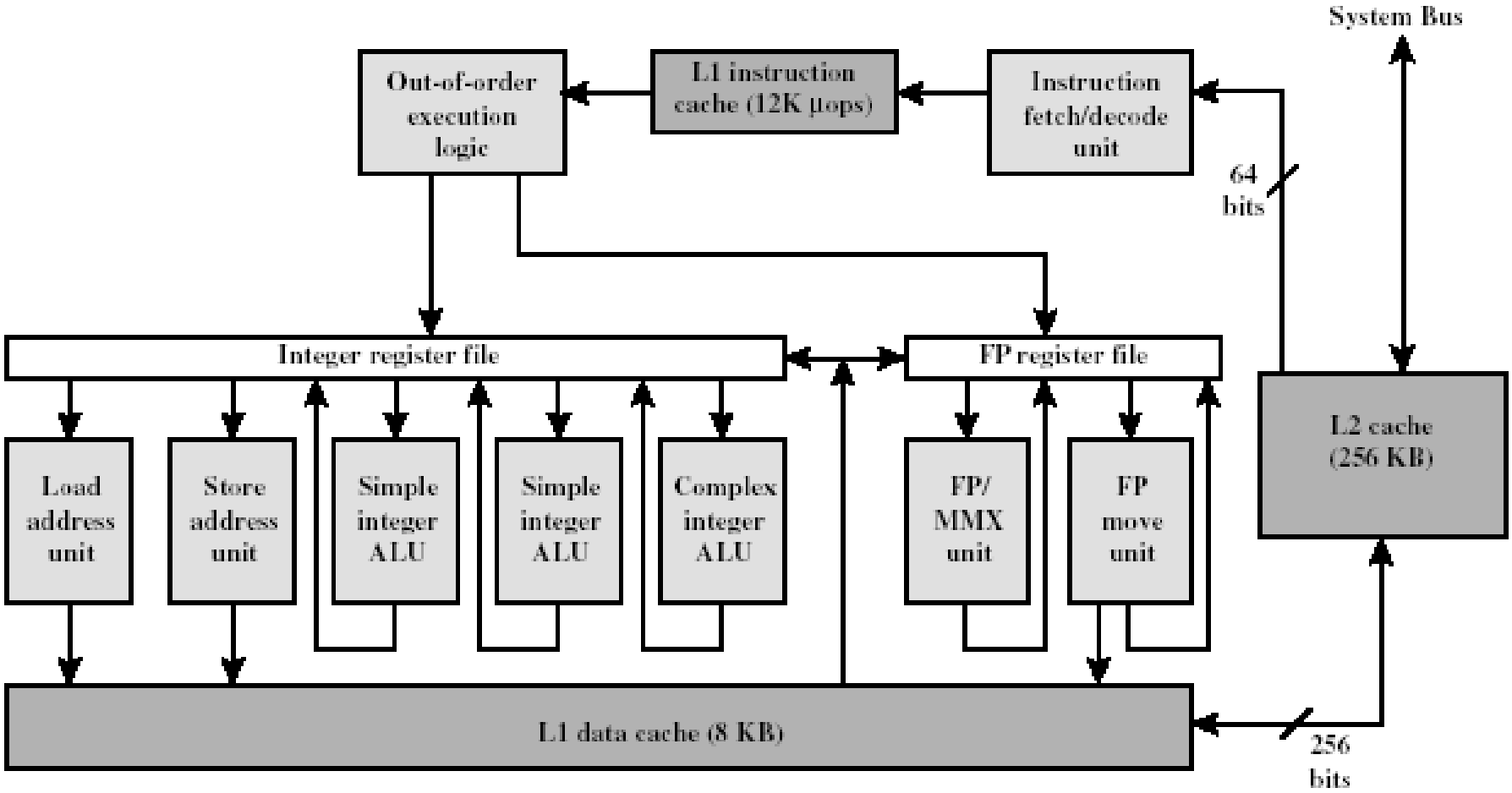
No-write allocate

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

No hay copia entre cache y mem ppal. Solo se actualiza la mem ppal

Pentium 4

procesador previo al CORE



Pentium 4 (cont.)

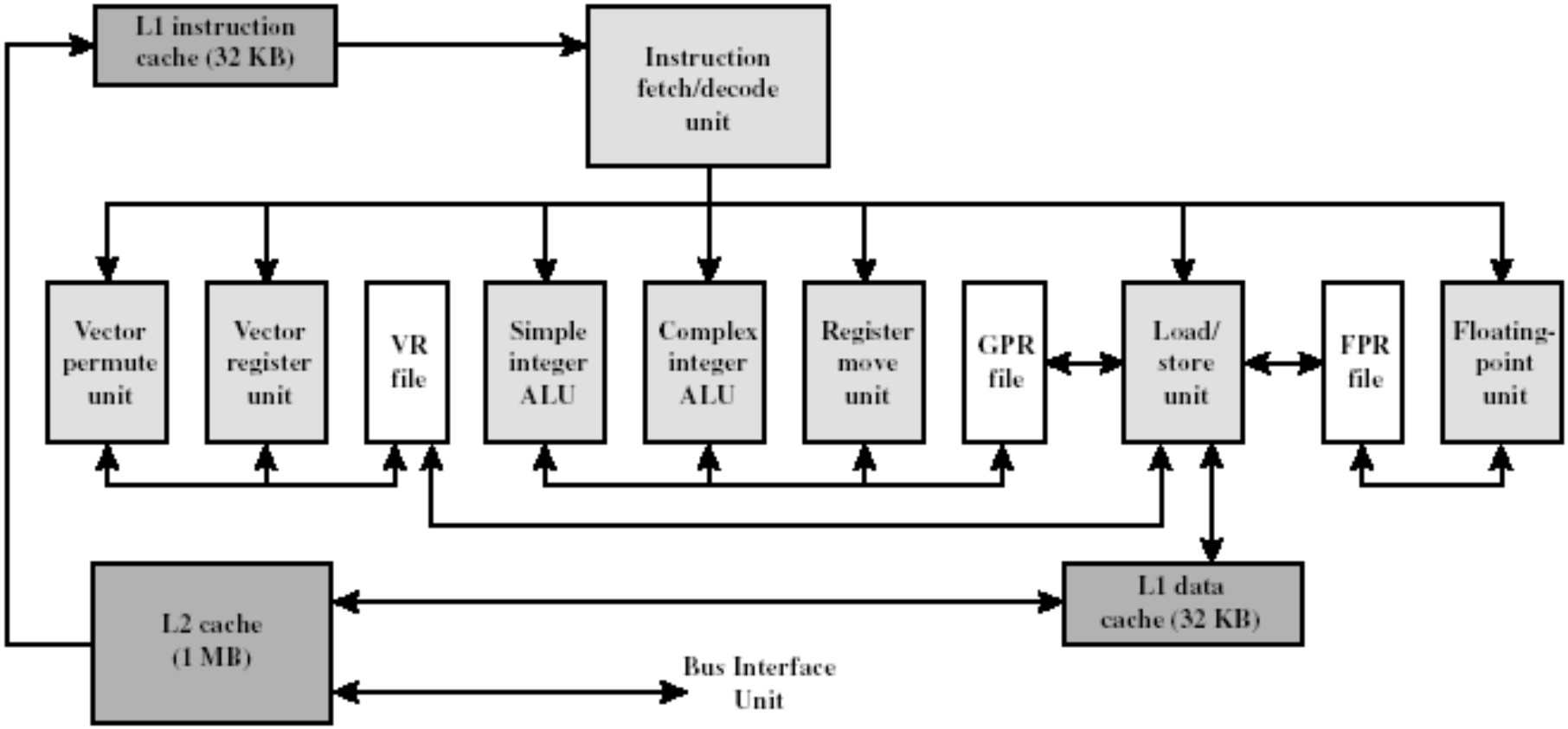
Puede tener hasta tres niveles de cache:

- 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.