Arquitectura y Tecnología de Computadores. Curso 09/10

Arquitecturas Paralelas. **Practica III**

ANÁLISIS DEL COMPORTAMIENTO DE LA MEMORIA CACHE

Estudio mediante Simulación

1. Objetivo:

En la presente práctica se pretende analizar el comportamiento de la memoria *cache* frente a la variación de algunos parámetros de diseño fundamentales en este tipo de memorias, como son el *tamaño de bloque* y el *grado de asociatividad,* y también analizar la alternativa que suponen las *caches* independientes de datos e instrucciones frente a las *caches* unificadas. El indicador elegido para valorar dicho comportamiento será la tasa de fallos a que da lugar un determinado diseño de la *cache* al trabajar sobre aplicaciones concretas, indicador que está relacionado de forma directa con el rendimiento proporcionado por la *cache*. En cuanto a la forma de realizar el análisis, se utilizara un programa de simulación, cuyos datos de entrada corresponden a todas las referencias a memoria efectuadas por un determinado procesador al ejecutar las aplicaciones.

2. Análisis a realizar:

Partiremos de ficheros que contienen las referencias a memoria (*trazas de memoria*) efectuadas por un procesador MIPS R3000 al ejecutar el conjunto de aplicaciones *(benchmarks)* que forman parte de SPEC CPU92. De todos ellos elegiremos dos representativos: ***sc*** (datos enteros) y ***nasa7*** (datos flotantes). A partir de estos ficheros, y con la ayuda del simulador de *cache*, realizaremos los siguientes análisis:

1. Análisis de la variación de la *tasa de fallos* en la *cache* en función del *tamaño de bloque* y el *tamaño total* de la *cache* para un *grado de asociatividad* fijo.
2. Análisis de la variación de la *tasa de fallos* en la *cache* en función del *grado de asociatividad* y el *tamaño total* de la *cache* para un *tamaño de bloque* fijo.
3. Análisis de la influencia del tipo de aplicación sobre el comportamiento de la *cache*
4. Análisis de la alternativa de *caches* i*ndependientes* de datos e instrucciones frente a *cache unificada*

3. Pasos a seguir en la práctica:

- Abrir una sesión en la estación de trabajo *centauro.*

- Leer el Anexo I. Copiar el simulador **acs**, pero **NO** las trazas.

- Invocar al simulador de *cache* con los parámetros adecuados para ir obteniendo cada uno de los puntos de las dos gráficas de análisis de la *cache* (ver Anexo II), todo ello para cada una de las dos aplicaciones representativas de SPEC CPU92 y considerando siempre *caches* de primer nivel (L1, integradas en el propio chip procesador), unificadas (datos + instrucciones) y sin sectores. Los resultados arrojados por el simulador son tasas de fallos en tanto por uno.

- Seleccionar un diseño adecuado de *cache* bajo la perspectiva de *rendimiento/coste* para cada una de las dos aplicaciones estudiadas (en función de las gráficas anteriores) y otro para una carga de trabajo formada por las dos aplicaciones. **Para la selección de la caché más adecuada debe encontrarse una solución de compromiso entre una menor tasa de fallos y la penalización que supone escoger una caché con mayor tamaño de bloque o mayor grado de asociatividad**.

- Con el diseño de *cache* correspondiente a la carga de trabajo formada por las dos aplicaciones representativas, realizar simulaciones con todas las aplicaciones de SPEC CPU92 y comparar los resultados. ¿Existen diferencias significativas entre los resultados obtenidos para las aplicaciones enteras? ¿y para las flotantes? ¿condiciona pues el tipo de aplicación el comportamiento de la *cache*?

- Muchos de los procesadores actuales tienen *caches L1* independientes de datos e instrucciones, por lo que se propone aquí realizar un estudio de la variación de rendimiento al utilizar dicha alternativa, partiendo del diseño inicial de *cache* unificada obtenido para el conjunto de las dos aplicaciones estudiadas. Para ello, repartir la capacidad de la *cache* unificada de forma equitativa entre *caches* independientes de datos e instrucciones y volver a simular con todas las aplicaciones SPEC CPU92. Hay que tener en cuenta que al trabajar con *caches* independientes, el simulador proporcionara dos tasas de fallo, una para cada cache. De este modo, si queremos comparar los resultados con los de la *cache* unificada habrá que obtener una tasa de fallos conjunta, a base de ponderar cada una de las tasas con el correspondiente peso de datos e instrucciones en cada aplicación:

tasa fallos[conjunta] = tasa fallos[cache datos] * peso_datos + tasa fallos[cache inst.] * peso_inst.

  ¿Qué variación de comportamiento se aprecia para las aplicaciones enteras? ¿y para las flotantes?
  ¿Qué alternativa da lugar a un mejor comportamiento de la cache para el conjunto de aplicaciones?

4. Trabajo a entregar:

- Confección y entrega de las 4 tablas de resultados y de las 4 gráficas correspondientes a los análisis efectuados sobre las dos aplicaciones representativas elegidas.

- Elección del diseño de *cache* mas adecuado para cada una de las aplicaciones estudiadas bajo el punto de vista *rendimiento/coste*, así como del diseño de *cache* mas adecuado para una carga de trabajo formada por las dos aplicaciones.

- Comparativa tabular y gráfica de los resultados de las simulaciones de *cache unificada* frente a *caches independientes* de datos e instrucciones para todas las aplicaciones de SPEC CPU92.

Todas las tablas, gráficas y diseños de cache seleccionados se incluirán en una nueva hoja de calculo del documento Excel de la asignatura, según el modelo mostrado en el Anexo III.

Anexo I: *El simulador de Cache y las trazas de memoria*

El simulador a utilizar se denomina **acs** y está en la máquina *centauro*, en el directorio:

`/profesores/fran/4atc/simcache/`

En dicho directorio se encuentran también los ficheros README y ACS.TXT, los cuales contienen información acerca del simulador, como por ejemplo el formato de llamada al simulador.

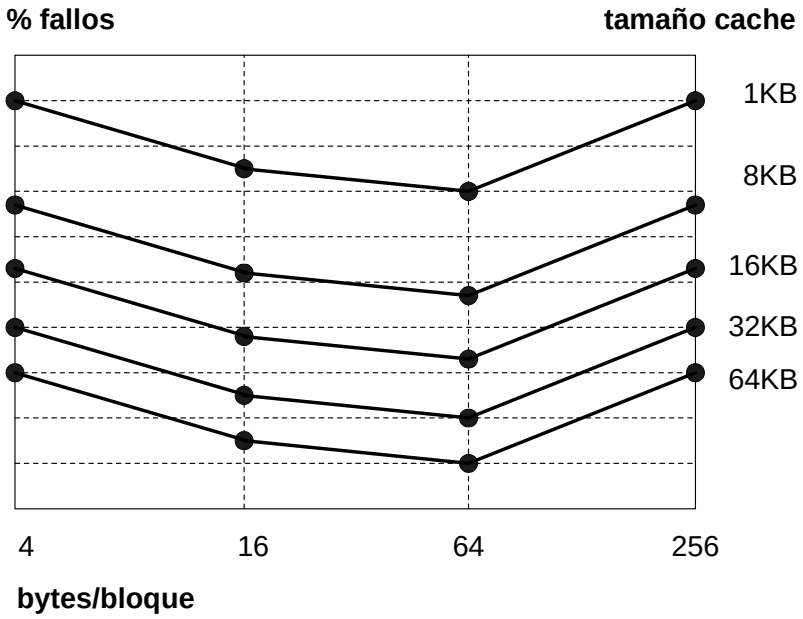
Las *trazas de memoria* de las aplicaciones que se utilizarán a la hora de invocar al simulador están en el directorio:

`/profesores/fran/4atc/simcache/r3000_cpu92_pdts/`

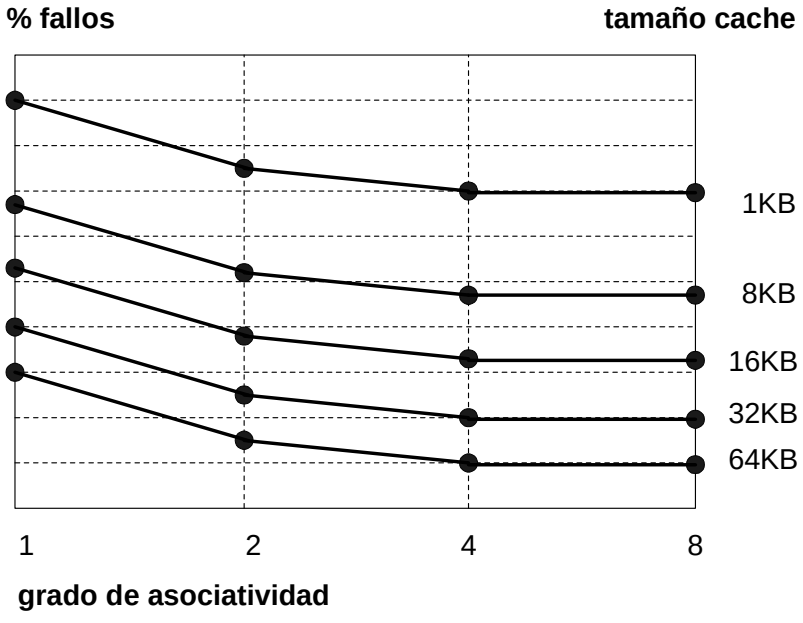
y todas ellas tienen extensión *.pdt.Z*, es decir, están comprimidas. A pesar de ello, las trazas ocupan bastante espacio en disco y por lo tanto no deben ser copiadas, sino referenciadas directamente al invocar al simulador. En los ficheros de ayuda del simulador comentados anteriormente, se indica la forma de invocar al simulador con trazas comprimidas (ver también el archivo de comandos *genera* dentro de la carpeta simcache, el cual contiene una invocación de ejemplo).

En el directorio `r3000_cpu92_pdts` están además los ficheros TRACE_LIST y SPEC_CPU92.TXT, los cuales contienen información acerca del conjunto de aplicaciones *(benchmarks)* SPEC CPU92 que será necesario conocer a la hora de realizar los análisis planteados.

Anexo II: *Gráficas de ejemplo*



Gráfica 1. *Ejemplo de comportamiento de una memoria cache de* ***correspondencia directa*** *ante cambios del tamaño de bloque*

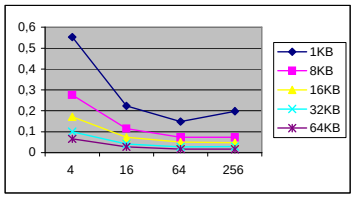


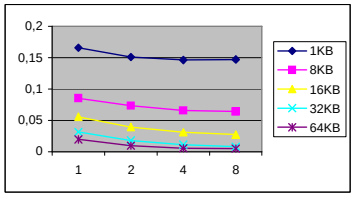
Gráfica 2. *Ejemplo de comportamiento de una memoria cache con* ***32 bytes de tamaño de bloque*** *ante cambios del grado de asociatividad*

Anexo III: *Tablas y Gráficas de Excel*

1) Curvas de Rendimiento de Caches

sc

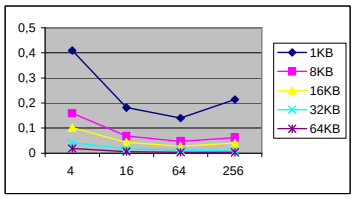


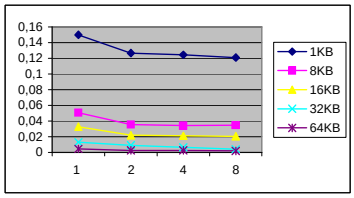


Caché adecuada:

Tamaño	
Bytes/Bloque	
Asociatividad	

nasa7





Caché adecuada:

Tamaño	
Bytes/Bloque	
Asociatividad	

Caché adecuada para las dos aplicaciones:

Tamaño	
Bytes/Bloque	
Asociatividad	

2) Caché Unificada frente a Caches Independientes

sc

unificada	independientes

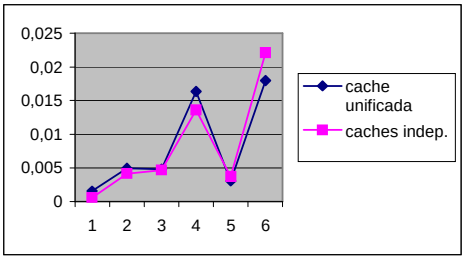
nasa7

unificada	independientes

spec_cpu92

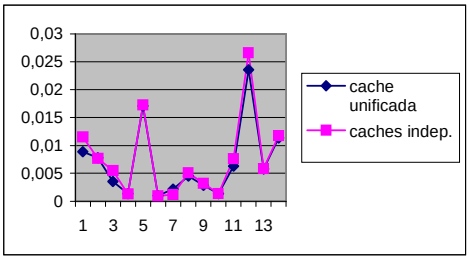
INT

unificada	independ.	FD	D	FI	I



FP

unificada	independ.	FD	D	FI	I



FD/FI: tanto por uno de fallos de datos/instrucciones
D/I: tanto por uno de datos/instrucciones

¿Condiciona el tipo de aplicación ejecutada el comportamiento de la cache? Sí/No

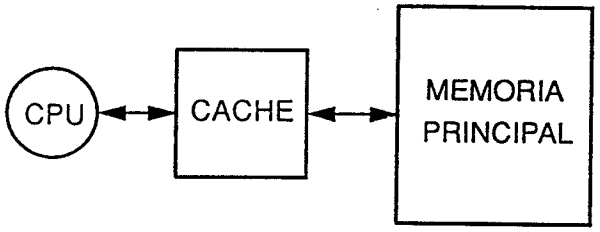
¿Qué alternativa da lugar a un mejor comportamiento de la cache para el conjunto de aplicaciones?
Unificada/Independientes

Anexo IV: *Documentación sobre Memoria Cache*

En las páginas que siguen se resumen las principales técnicas utilizadas en el diseño de los niveles de memoria *cache* dentro de la jerarquía de memoria de los computadores.

Principios Generales de las Cache

- Cache: memoria pequeña y rápida que almacena la parte del contenido de una memoria grande y lenta que se está utilizando en cada momento
- Objetivo: conseguir la capacidad de la memoria grande (y lenta) con la velocidad de la memoria rápida (y pequeña)
- Exito: posible debido a la Localidad (temporal y espacial) de los programas
  - ✓ retiene información recientemente usada
  - ✓ retiene información cercana a la recientemente usada



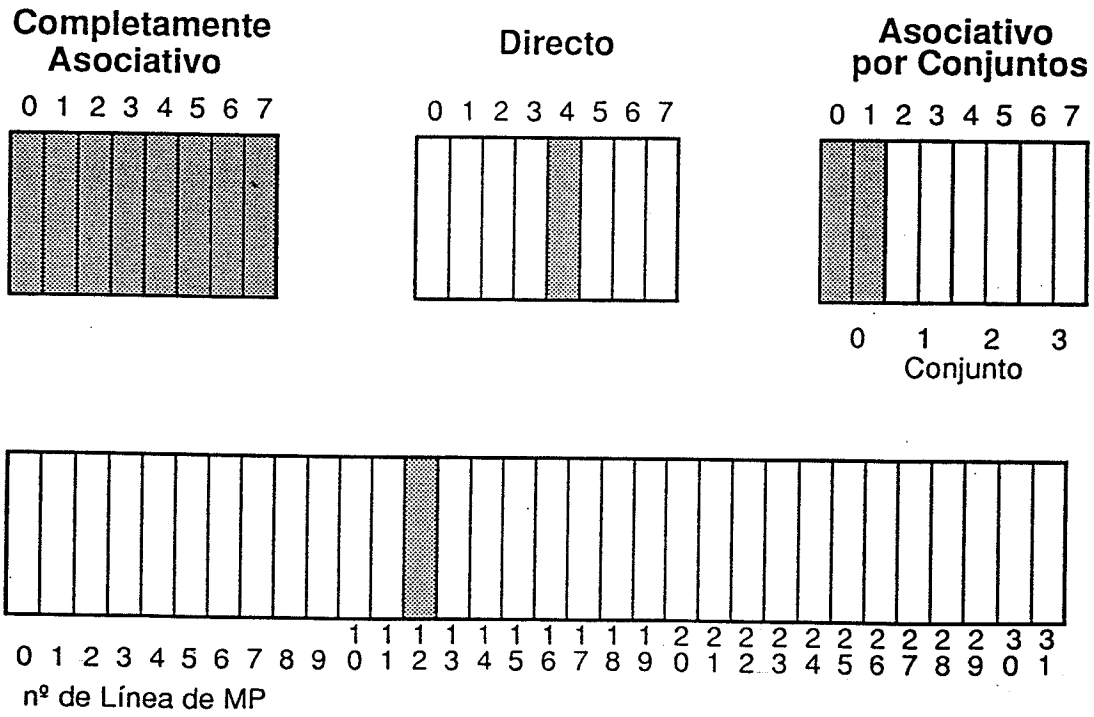
Acierto
Fallo
Línea

Emplazamiento
Reemplazo
Escritura

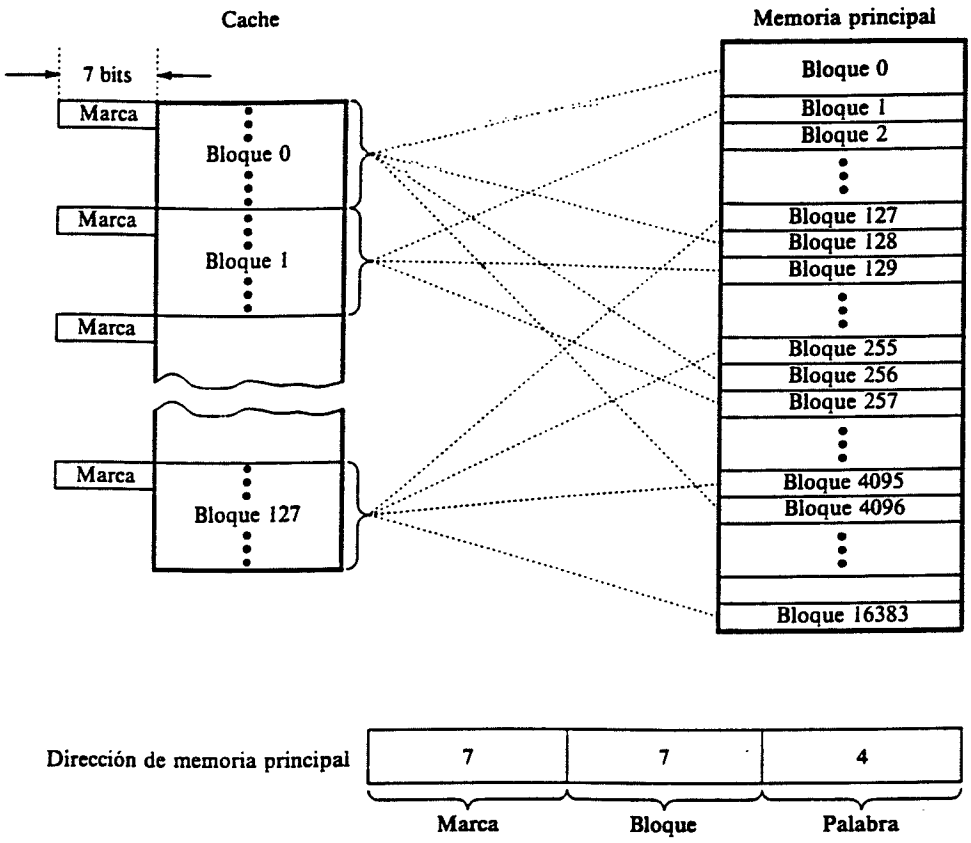
Capacidad	1 KByte - 256 KByte
Tamaño Línea	16 - 128 Byte
Tasa Fallos	1% - 20%
Tiempo Servicio *Acierto*	1 - 4 ciclos reloj (normalmente 1)
Tiempo Servicio *Fallo*	8 - 32 ciclos reloj

Algoritmos de Emplazamiento

- ¿Dónde se ubica una Línea de MP en la MC?
  - ✓ Completamente Asociativo
    Cada Línea de MP puede emplazarse en cualquier contenedor de Líneas de la MC
  - ✓ Directo
    Cada Línea de MP sólo puede emplazarse en un contenedor de Líneas concreto de la MC
  - ✓ Asociativo por Conjuntos
    MC se divide en Conjuntos disjuntos. Cada Línea de MP puede emplazarse en cualquier contenedor de un Conjunto concreto

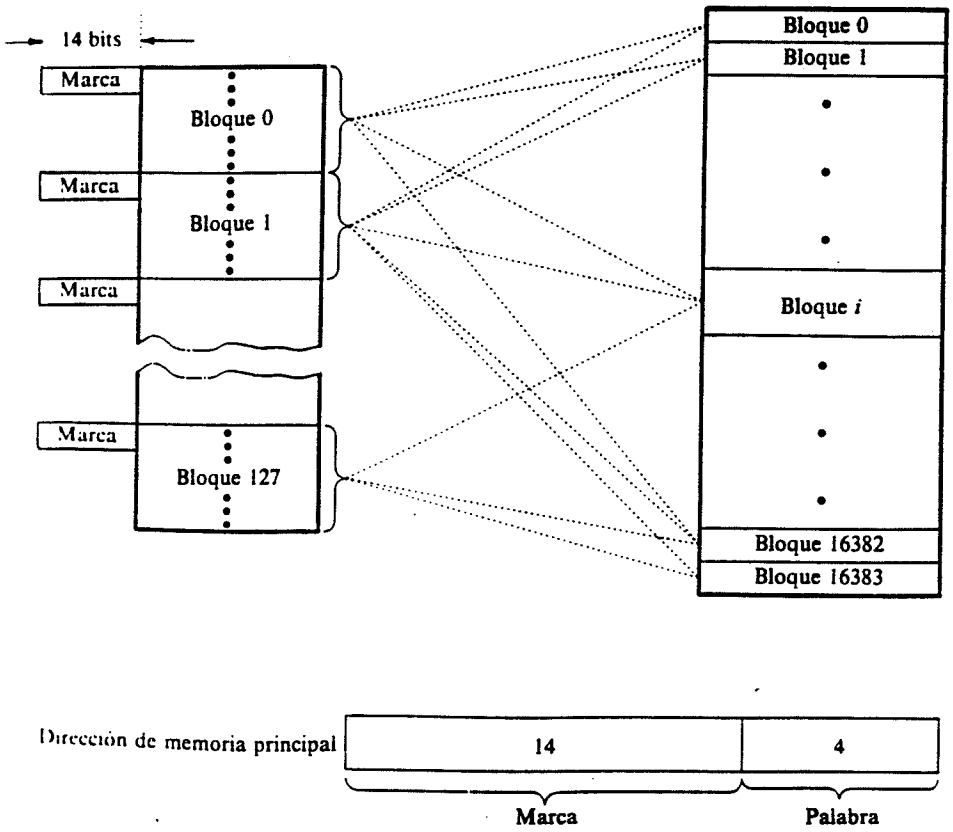


Correspondencia directa



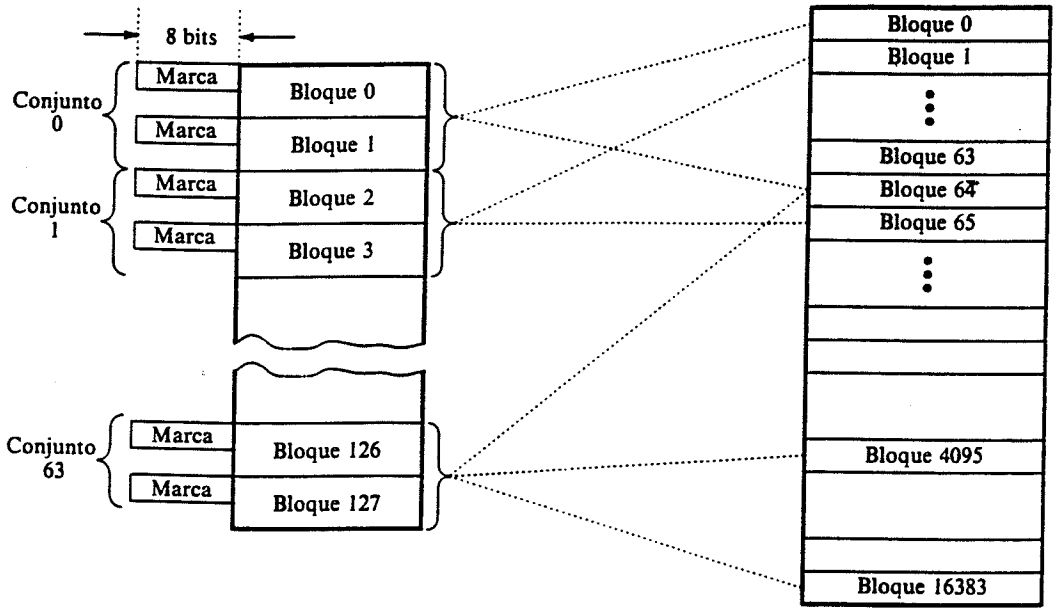
- Sencillez
- Comparación con una sola marca (Mínimo coste)

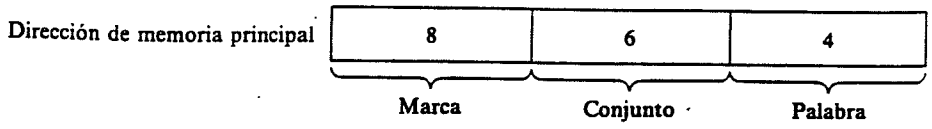
Correspondencia totalmente asociativa



- Máxima flexibilidad
- Comparación asociativa con todas las marcas (Máximo coste)

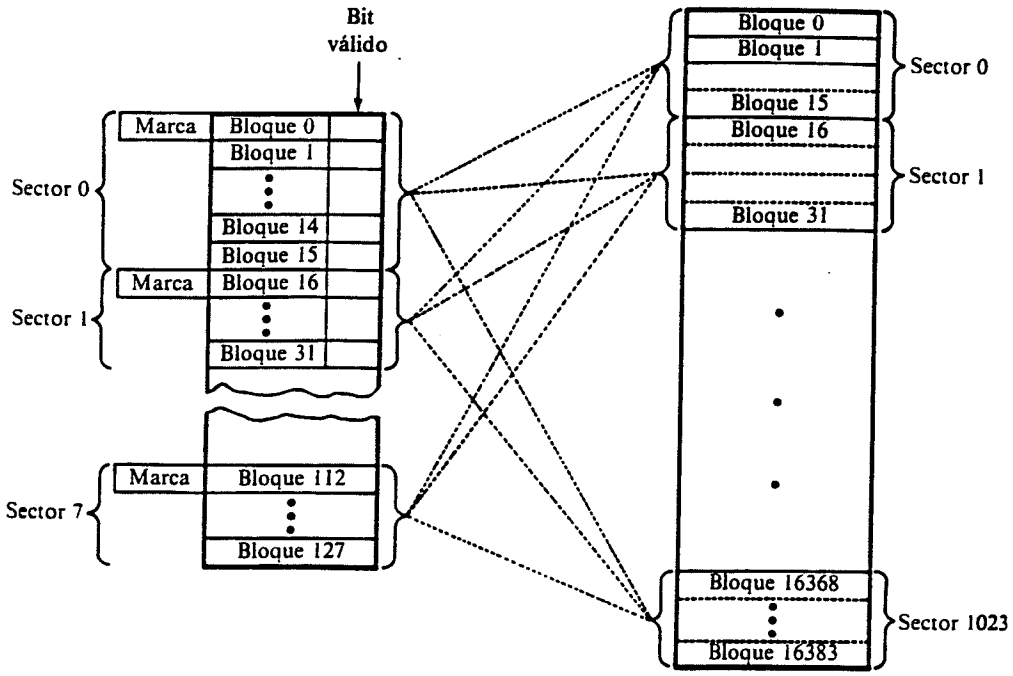
Correspondencia asociativa por conjuntos

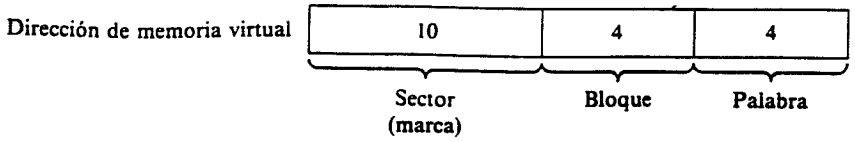




- Menor flexibilidad que la corresp. totalmente asociativa
- Menor coste
- Compromiso entre las anteriores organizaciones

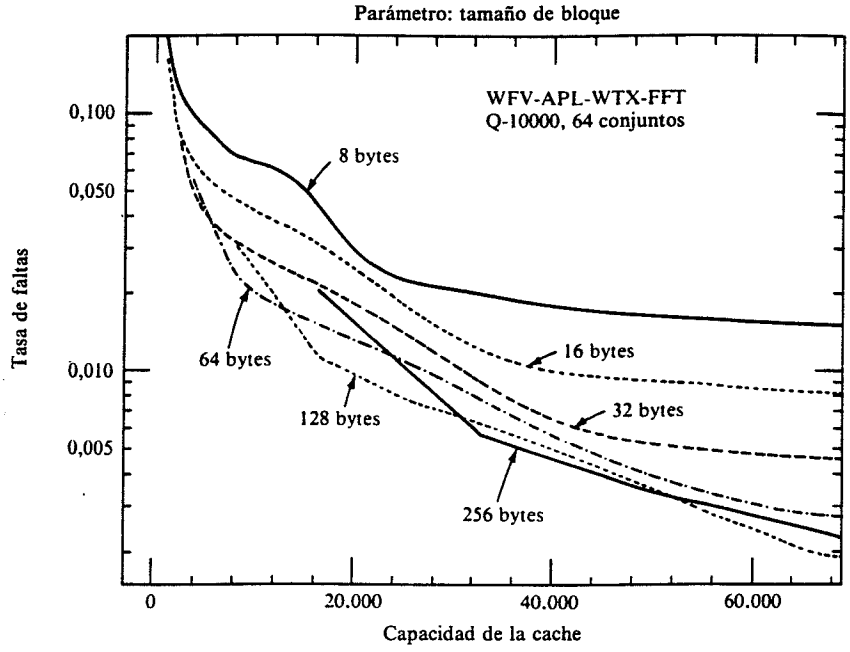
Correspondencia por sectores



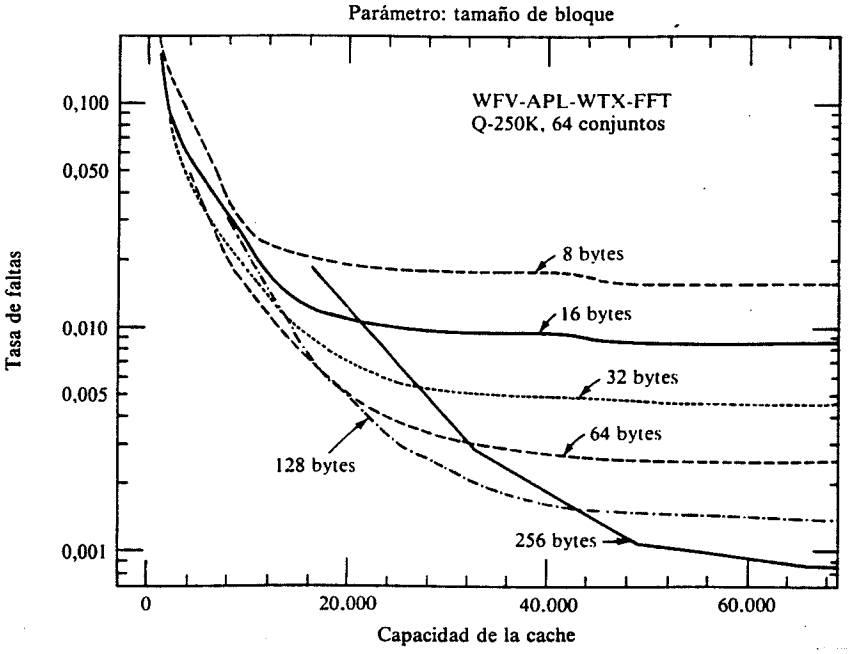


- Reducción del coste gracias a la reducción del numero de marcas

Tasa de faltas en función del tamaño de bloque y la capacidad de la Cache



(*a*) Tamaño de cuanto, $Q = 10.000$ referencias



(*b*) Tamaño de cuanto, $Q = 250K$ referencias

Algoritmos de Reemplazo

Si se produce fallo y el conjunto donde debe emplazarse el bloque esta lleno **¿qué bloque debe ser reemplazado?**

Emplazamiento Directo ➔ elección trivial

- ✓ **Aleatorio**
  Elección al azar
  Fácil de implementar

- ✓ **FIFO**
  Se reemplaza el bloque que lleva más tiempo en la MC

- ✓ **LRU**
  Se reemplaza el bloque que hace más tiempo que fue referenciado

Algoritmos de Escritura

¿Cuándo se actualiza la MP en caso de acierto en escritura?

(Importante: 10-30% de accesos = escrituras)

- ✓ **Escritura inmediata** *(Write Through)*
  Se escribe a la vez en MC y MP

- ✓ **Escritura cuando reemplazo** *(Copy Back)*
  MP solo se actualiza cuando se reemplaza el bloque (si fue modificado durante su permanencia en MC)

¿Qué ocurre en caso de fallo en escritura?

- ✓ Carga en escritura *(Write Allocate)*
- ✓ No carga en escritura *(No Write Allocate)*