Estructura de los Computadores

Tema 3. Unidad de Memoria

Memoria Cache

Concepto

La memoria principal es un recurso muy utilizado por la CPU ya que accede al menos una vez para buscar la instrucción a ejecutar y realiza frecuentemente múltiples accesos para leer operandos o almacenar resultados. La memoria asociada a la CPU, es decir, sus registros, es mucho más rápida que la memoria principal debido a la distinta tecnología con la que se fabrica. Lo ideal sería que la memoria principal fuera de la misma tecnología que los registros de la CPU, pero debido a su alto coste se tiende a soluciones intermedias. Una solución sería aprovecharnos del principio de localidad y colocar una memoria muy rápida entre la CPU y la memoria principal de tal forma que la CPU acceda más veces a esa memoria que a memoria principal. Esta memoria muy rápida deberá ser pequeña para que los costes no sean excesivos. A esta memoria se le denomina memoria cache.

El funcionamiento de la memoria cache se basa en coger partes (bloques) de la memoria principal y copiarlos a la memoria cache. La CPU, en primer lugar mira si el dato que necesita procesar está en la cache. Si ésta, lo coge muy rápido. Si no está, lo busca en la memoria principal y lo coge. Luego copia el bloque de memoria principal donde está el dato a la memoria cache. Debido al principio de localidad de referencia a memoria, es muy probable que cuando se transfiere un bloque de datos a la memoria cache, las futuras llamadas a memoria se hagan a otras palabras contenidas en el bloque transferido. En la figura 1 se muestra la configuración de un computador con memoria cache.

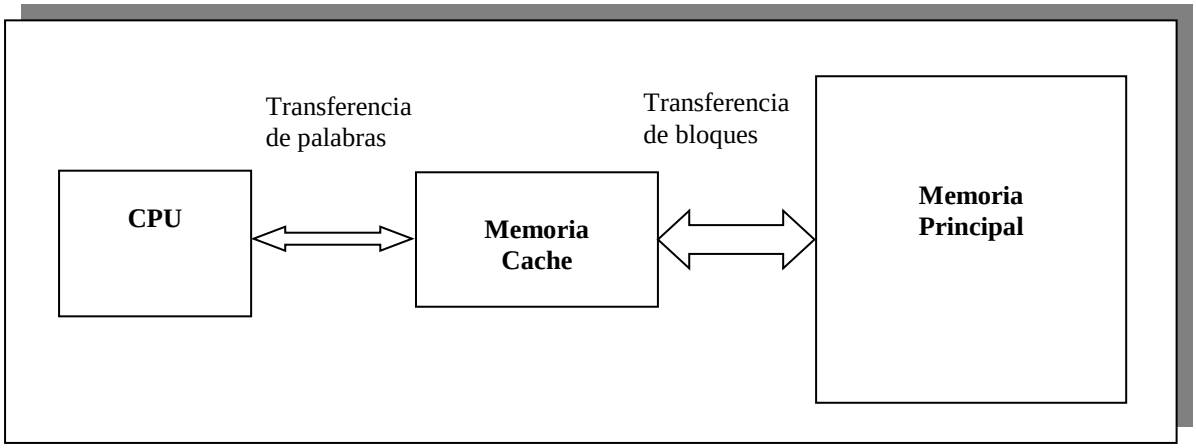


Figura 1. Configuración computador con memoria cache.

En un sistema con memoria cache, una memoria principal de 2^n palabras está organizada en M bloques de longitud fija (K palabras/bloque) donde $M=2^n/K$ bloques. La memoria cache está dividida en C líneas o particiones de K palabras cada una y el número de líneas de la memoria cache es mucho menor que el número de bloques que hay en la memoria principal, es decir, C<<M. En todo momento, un subconjunto de los bloques de memoria principal está almacenado en líneas de la cache. Ya que hay más bloques que líneas, una línea dada no puede dedicarse unívoca y permanentemente a un bloque. Por lo tanto, cada línea incluye una etiqueta que identifica qué bloque particular está siendo almacenado. En la figura 2 podemos ver la estructura de un sistema con memoria cache.

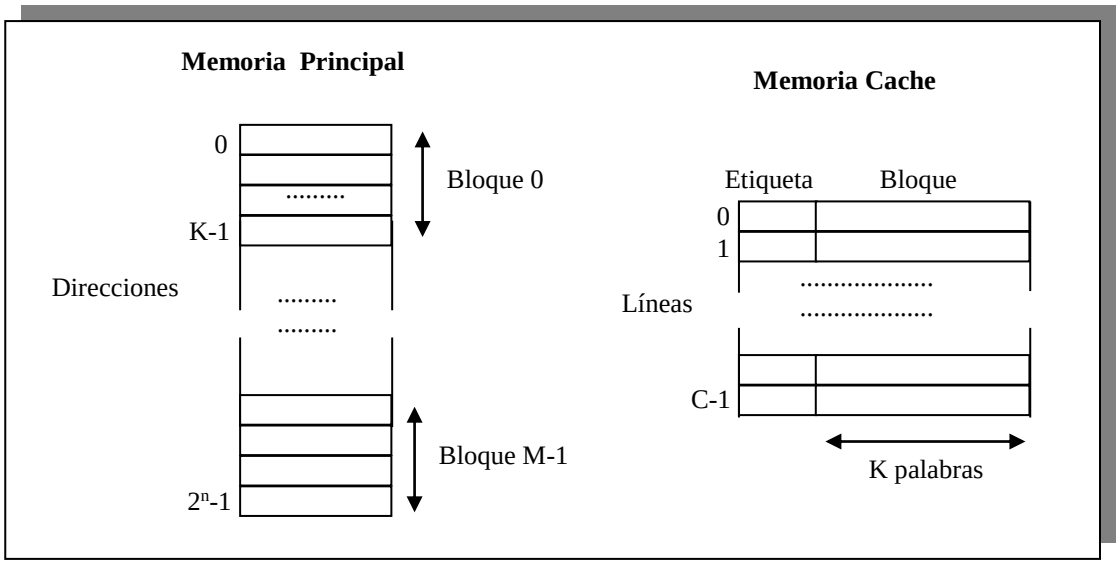


Figura 2. Estructura memoria principal y memoria cache.

El rendimiento de una memoria cache se mide frecuentemente en términos de un parámetro cantidad llamado tasa de acierto. Cuando la CPU necesita una palabra de memoria y la encuentra en la memoria

cache, se dice que se produce un acierto (hit). Si la palabra no está en la memoria cache se contabiliza un fallo (miss).

La relación entre el número de aciertos y el número total de referencias a memoria (aciertos+fallos), es la tasa de acierto. En sistemas bien diseñados se suelen conseguir tasas de aciertos de 0.9.

$$Tasa\quad de\quad acierto = \frac{N^{o}\quad aciertos}{N^{o}\quad referencia\quad s}$$

Parámetros de diseño

A la hora de realizar un diseño de un sistema con memoria cache es necesario tener presente los siguientes parámetro pues son muy influyentes en el rendimiento del mismo.

Tamaño de la memoria cache

El tamaño de la memoria cache plantea un cierto compromiso. Por un lado, debería ser lo suficientemente pequeña como para que el coste medio por bit de información almacenada en la memoria interna del computador estuviese próximo al de la memoria principal. Por otro, tendría que ser lo suficientemente grande como para que el tiempo de acceso medio total fuese lo más próximo posible al de la memoria cache.

Además, hay algunas otras motivaciones para tratar de minimizar el tamaño de una memoria cache. Cuanto más grande sea la memoria cache más compleja será la lógica asociada a su direccionamiento. El resultado es que las memorias cache de gran capacidad tienden a ser más lentas que las más pequeñas, incluso empleando en ambas la misma tecnología de fabricación. También su capacidad tiende a estar limitada por el espacio físico que se le asigna.

De acuerdo a estudios empíricos se sugiere que el tamaño de una cache esté situado entre 1Kb y 1Mb. No obstante, como las prestaciones de la cache son muy sensibles al tipo de tarea, es imposible predecir un tamaño óptimo.

Tamaño del bloque

Cuando se transfiere un bloque de la memoria principal a la memoria cache, con la palabra deseada se mueve un conjunto de palabras adyacentes. Por esta razón, cuando se va aumentando el tamaño del bloque, a partir de valores muy pequeños la tasa de acierto inicialmente aumentará a causa del principio de localidad, ya que se incrementa la probabilidad de que sean accedidos, en el futuro inmediato, datos próximos a la palabra referenciada. Sin embargo, a partir de un cierto tamaño del bloque la tasa de acierto comienza a disminuir, debido a que la probabilidad de utilizar la nueva información

captada se haga menor que la de reutilizar la información que tiene que reemplazarse. Surgen dos efectos:

Cuanto mayor sea el tamaño de los bloques, menor será el número de éstos que es posible tener en la memoria cache. Como cada búsqueda de un bloque en la memoria principal lleva consigo su escritura en la memoria cache, si el número de bloques es pequeño aumenta la frecuencia de utilización del algoritmo de sustitución de bloques.

Cuando crece el tamaño de un bloque, cada nueva palabra añadida a ese bloque estará a mayor distancia de la palabra requerida por la CPU, y por tanto es menos probable que sea necesitada a corto plazo.

La relación entre el tamaño de bloque y la tasa de acierto es compleja, y depende de las características de localidad de cada programa en particular. Por ese motivo no se ha encontrado un valor óptimo, sin embargo, una elección razonable está entre 4 y 8 unidades direccionables (palabras o bytes).

Número de caches

Cuando se surgió el concepto de memorias cache, se pensó en una memoria por sistema. Hoy en día, se ha visto que se aumenta mucho más el rendimiento del sistema si se utilizan varias memorias cache. En un sistema podemos encontrarnos varios tipos de caches.

Cache de uno o dos niveles.

Conforme la densidad de integración ha ido aumentando, se ha visto que es posible tener una cache en el mismo chip del procesador. Según dónde esté ubicada la memoria cache se distinguen dos tipos:

- Cache interna. Nivel 1. Físicamente está ubicada en el mismo chip que el procesador. Por tanto, reduce la actividad del bus externo del procesador y los tiempos de ejecución e incrementa las prestaciones globales del sistema. Debido a que los caminos de datos internos al procesador son muy cortos en comparación con la longitud de los buses, los accesos a esta cache se efectúan más rápido. Por otro lado, la capacidad de esta cache es muy pequeña ya que no se dispone de mucho espacio físico pues tiene que estar dentro del chip del procesador.

- Cache externa. Nivel 2. Físicamente se ubica fuera del procesador por lo que será más lenta que la cache de nivel 1 pero seguirá siendo más rápida que la memoria principal. Al estar fuera, la capacidad podrá ser mayor que la cache de nivel 1.

Contenido de la cache

En un principio, los diseños contenían una sola cache para almacenar las referencias tanto a datos como a instrucciones. Hoy en día, se ha visto que el hecho de separar la cache en dos, es decir, una dedicada a instrucciones y otra a datos, da mejores resultados.

Una cache que contiene tanto datos como instrucciones presenta las siguientes ventajas:

Tiene una tasa de aciertos mayor ya que nivela la carga, es decir, si un patrón de ejecución implica muchas captaciones de instrucción que de datos, la cache tenderá a llenarse con instrucciones, y viceversa.

Sólo se necesita implementar y diseñar una cache, por lo que el coste será más reducido.

En los computadores de hoy en día, donde se enfatiza la ejecución paralela de instrucciones y los diseños pipelining, el uso de dos caches da mejores prestaciones. La principal ventaja es que elimina la competición entre el procesador de instrucciones y la unidad de ejecución. De esta forma el procesador captará instrucciones anticipadamente y las almacenará en la cache, mientras que la unidad de ejecución tendrá los datos disponibles en su cache.

Estrategia de escritura

Antes de que pueda ser reemplazado un bloque que está en la cache, es necesario saber si se ha modificado o no, es decir, si es distinto al bloque de la memoria principal. Si no se ha modificado (bloque limpio), se puede sobreescribir sobre él. Si ha sido modificado (bloque modificado o sucio), se tendrá que actualizar la memoria principal antes de borrarlo.

Existen diversas estrategias de escritura con diferentes costes y rendimientos. A continuación vamos a describir las más importantes.

Escritura inmediata o directa (write through).

Todas las operaciones de escritura se hacen tanto en la memoria cache como en la memoria principal, lo que se asegura que los contenidos son siempre válidos. El principal inconveniente es que genera mucho tráfico con la memoria principal y puede originar un cuello de botella.

Post-escritura (write back)

Las escrituras se realizan sólo en la memoria cache. Asociada a cada línea de la cache existe un bit de modificación. Cuando se escribe en la cache, el bit de modificación se pone a 1. En el caso de reemplazar una línea de la cache, se mira el bit de modificación y si está a 1, se escribirá la línea en memoria principal, mientas que si está a 0 no. El

principal inconveniente de esta técnica es que obliga a los módulos de E/S que accedan a memoria principal a través de la memoria cache ya que si acceden directamente es posible que no lean los datos actualizados. Esto complica la circuitería y genera un cuello de botella. Como ventaja está que utiliza menos ancho de banda de memoria principal haciendo idóneo para su uso en multiprocesadores.

En un sistema donde hay diferentes dispositivos con memorias cache locales para acceder a memoria principal se agravia el problema de coherencia de los datos. Supongamos que el procesador lee el bloque L de memoria principal y lo almacena en su memoria cache. A continuación, otro dispositivo lee el bloque L de memoria principal y lo almacena en su cache. Si el procesador modifica la línea de la cache donde está el bloque L, la información que tiene el dispositivo en su cache ya no es coherente con la del procesador. Las dos estrategias que hemos visto anteriormente no solucionan este problema. Un sistema que evite este problema se dice que mantiene la coherencia de cache. Entre las posibles aproximaciones a la coherencia de cache se incluyen:

- Vigilancia del bus en escritura inmediata. Cada controlador de cache monitoriza las líneas de direcciones para detectar operaciones de escritura en memoria. Si se detecta que se escribe en una posición de memoria compartida, se puede invalidar la cache o bien recargar los datos actualizados.

- Transparencia hardware. Se utiliza hardware adicional para asegurar que todas las actualizaciones de memoria principal, vía cache, quedan reflejadas en todas las caches. Así, si un procesador modifica una dirección de su cache, esta actualización se escribe en memoria principal y en todas las caches que contengan esa dirección.

Función de correspondencia

Debido a que existen menos líneas que bloques, se necesita un algoritmo (función) que haga corresponder bloques de memoria principal a líneas de memoria cache. Esta función de correspondencia asigna a los bloques de la memoria principal posiciones definidas en la memoria cache. Vamos a estudiar tres técnicas: directa, asociativa y asociativa por conjuntos. La figura 3 muestra en qué línea de la cache se podrá almacenar el bloque 12 de memoria principal.

Número Línea	Directa	Asociativa	Asociativa por conjuntos	
0				Conjunto 0
1				
2				Conjunto 1
3				
4				Conjunto 2
5				

6				Conjunto 3
7				

Figura 3. Ejemplo de funciones de correspondencia.

En correspondencia directa, el bloque 12 de memoria principal solo podrá almacenarse en la línea 4 (=12 módulo 8). En correspondencia asociativa se puede almacenar en cualquier línea, mientras que en correspondencia asociativa por conjuntos se puede almacenar en cualquier línea del conjunto 0 (=12 módulo (8/2)).

A continuación vamos a describir las tres técnicas antes enumeradas. En cada caso veremos la estructura general y un ejemplo concreto. Para los tres casos, vamos a trabajar con un sistema con las siguientes características:

- El tamaño de la memoria cache es de 4Kb.
- Los datos se transfieren entre memoria principal y la memoria cache en bloques de 16 bytes. Esto nos indica que la cache está organizada en 256 (4096/16) líneas con 16 bytes por línea.
- La memoria principal consta de 64Kb, por lo que el bus de direcciones es de 16 bits (2^{16} = 64K). Esto nos indica que en la memoria principal está compuesta por 4096 bloques con 16 bytes por bloque.

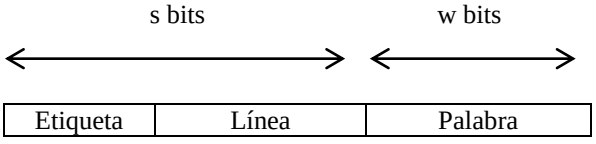
Correspondencia directa

Es la técnica más simple de todas. Consiste en hacer corresponder cada bloque de memoria principal a sólo una línea de memoria cache. La función de correspondencia se expresa mediante la siguiente función

$$i = j \text{ módulo } m$$

donde i = número de línea de cache
j = número de bloque de la memoria principal
m = número de líneas de la memoria cache

Cada dirección de la memoria principal puede verse como dividida en tres campos



donde w bits = Identifica cada palabra dentro de un bloque

s bits = Identifica el número de bloque

La lógica de la memoria cache interpreta los s bits como una etiqueta de (s-r) bits (parte más significativa) y un campo de línea de r bits. Este último campo identifica una de las 2^r línea de la memoria cache. El efecto es que se hacen corresponder bloques de memoria principal a líneas de cache de la siguiente forma:

Nº de líneas	**Nº del bloque que puede contener cada línea**
0	0, 2^r, ..., 2^s-2^r
1	2, 2^r+1, ..., 2^s-2^r+1
.....	....
.....	....
2^r-1	2^r-1, 2^{r+1}-1, ..., 2^s-1

Tabla 1. Correspondencia de bloques y líneas.

Así pues, el uso de una parte de la dirección como número de línea proporciona una asignación única de cada bloque de memoria principal en la cache. Cuando un bloque es realmente escrito en la línea que tiene asignada, es necesario etiquetarlo para distinguirlo del resto de los bloques que pueden introducirse en dicha línea. Para ello, se emplean los s-r bits más significativos.

En la figura 4 se muestra la organización de cache con correspondencia directa.

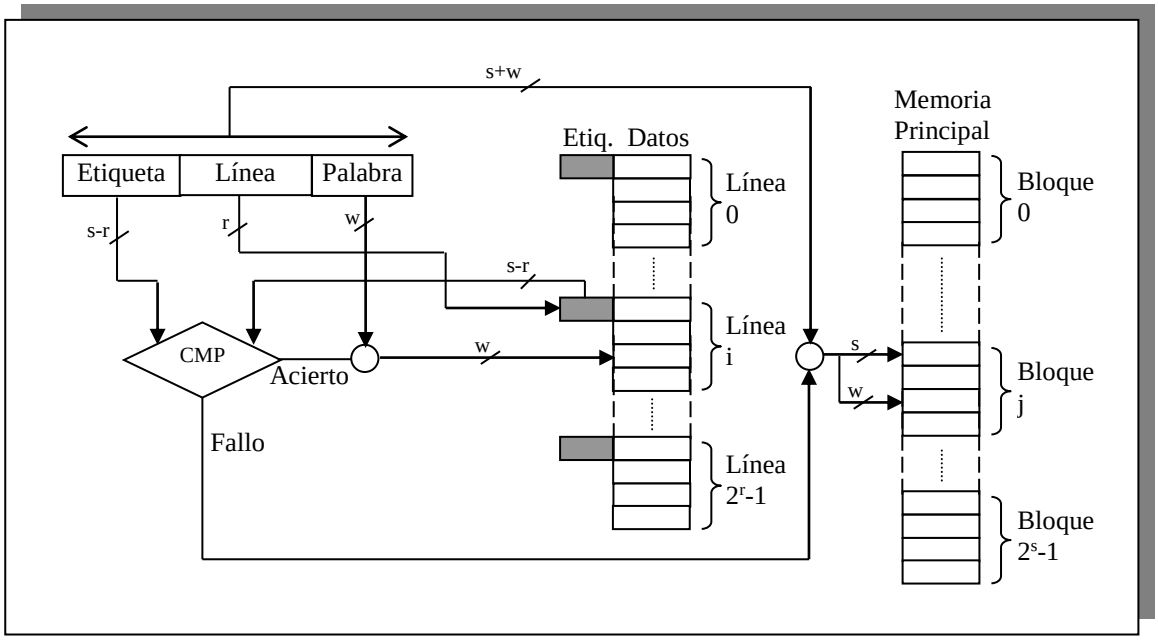


Figura 4. Correspondencia directa.

De acuerdo al ejemplo en el que nos estamos basando si utilizamos la correspondencia directa tendremos que la dirección de memoria se codificará como sigue

Etiqueta	Línea	Palabra
4 bits	8 bits	4 bits

En la figura 5 podemos ver un ejemplo gráfico de correspondencia directa.

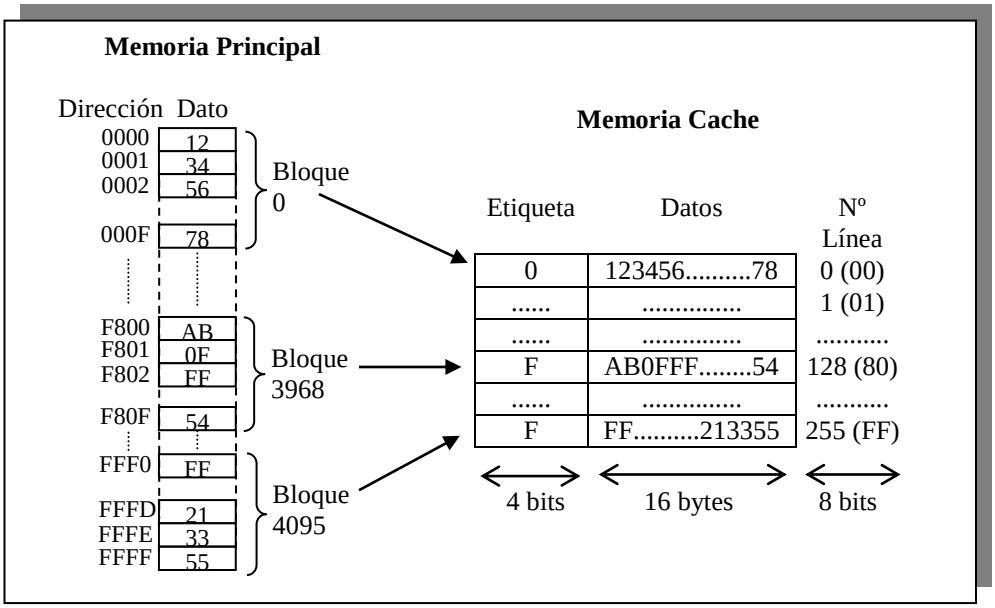


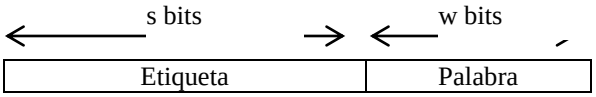
Figura 5. Ejemplo de correspondencia directa.

Al controlador de memoria cache se presenta una dirección de 16 bits. El número de línea, de 8 bits, se utiliza como índice para acceder a una línea particular dentro de la cache. Si el número de etiqueta, de 4 bits, coincide con el número de etiqueta almacenado actualmente en esa línea, el número de palabra de 4 bits se utilizará para seleccionar uno de los 16 bytes de esa línea. Si no, el campo de 12 bits de etiqueta-línea se empleará para coger un bloque de memoria principal. La dirección real que se utiliza para acceder a memoria principal serán los 12 bits de etiqueta-línea concatenados con 4 bits a 0, de manera que se cogerán 16 bytes a partir del comienzo del bloque.

La técnica de correspondencia directa es simple y poco costosa de implementar. Su principal inconveniente es que hay una posición concreta de cache para cada bloque dado. Por ello, si un programa referencia repetidas veces a palabras de dos bloques diferentes asignados en la misma línea, dichos bloques se estarían intercambiando continuamente en la cache, con lo que la tasa de aciertos disminuiría considerablemente.

Correspondencia Asociativa

Esta técnica soluciona el inconveniente que presenta la correspondencia directa, ya que permite que se cargue un bloque de memoria principal en cualquier línea de la memoria cache. En este caso, la lógica de control de la memoria cache interpreta una dirección de memoria como una etiqueta y un campo de palabra.



donde

Palabra = Palabra dentro de un bloque de memoria principal

Etiqueta = Identifica unívocamente un bloque que memoria principal

Para determinar si un bloque está en la memoria cache, se debe examinar simultáneamente todas las etiquetas de las líneas de la memoria cache.

En la figura 6 se muestra gráficamente la organización de la memoria cache con correspondencia asociativa.

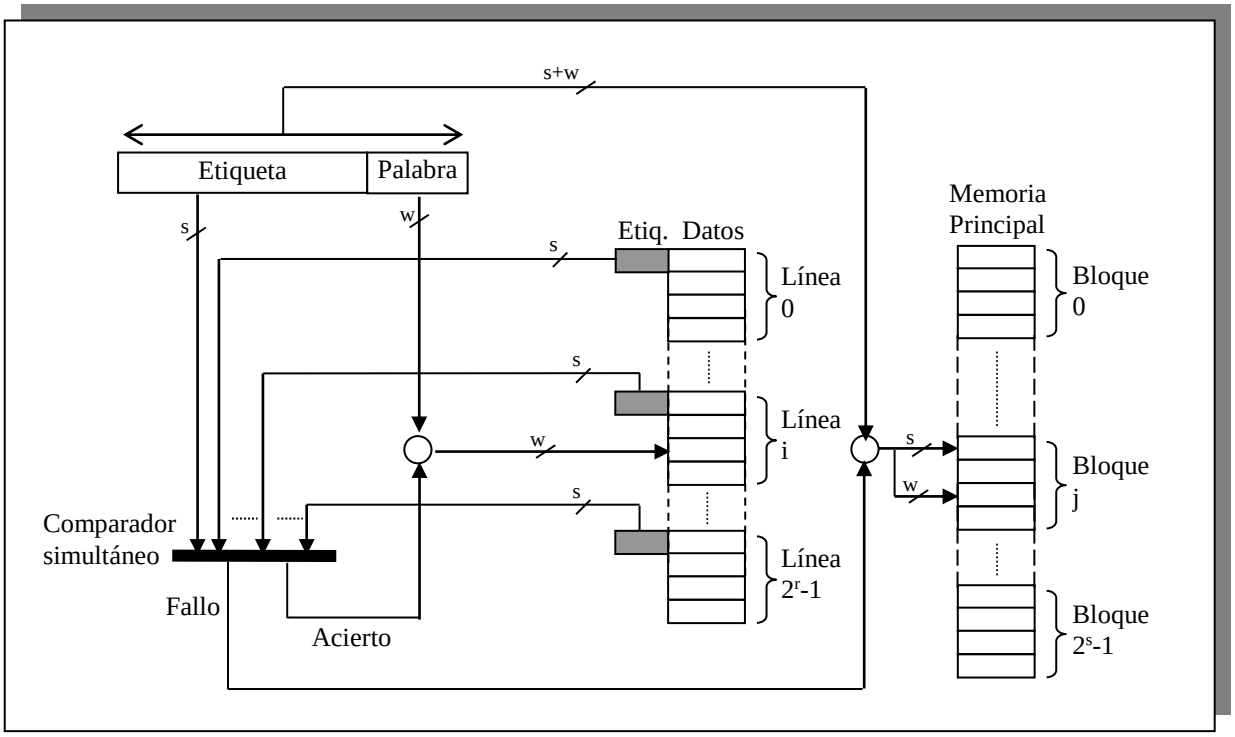
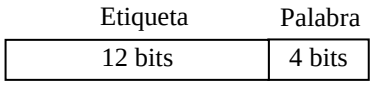


Figura 6. Correspondencia asociativa.

Siguiendo el ejemplo anteriormente descrito si utilizamos la correspondencia asociativa tendremos que la dirección de memoria se codificará como sigue



En la figura 7 podemos ver un ejemplo gráfico de correspondencia asociativa.

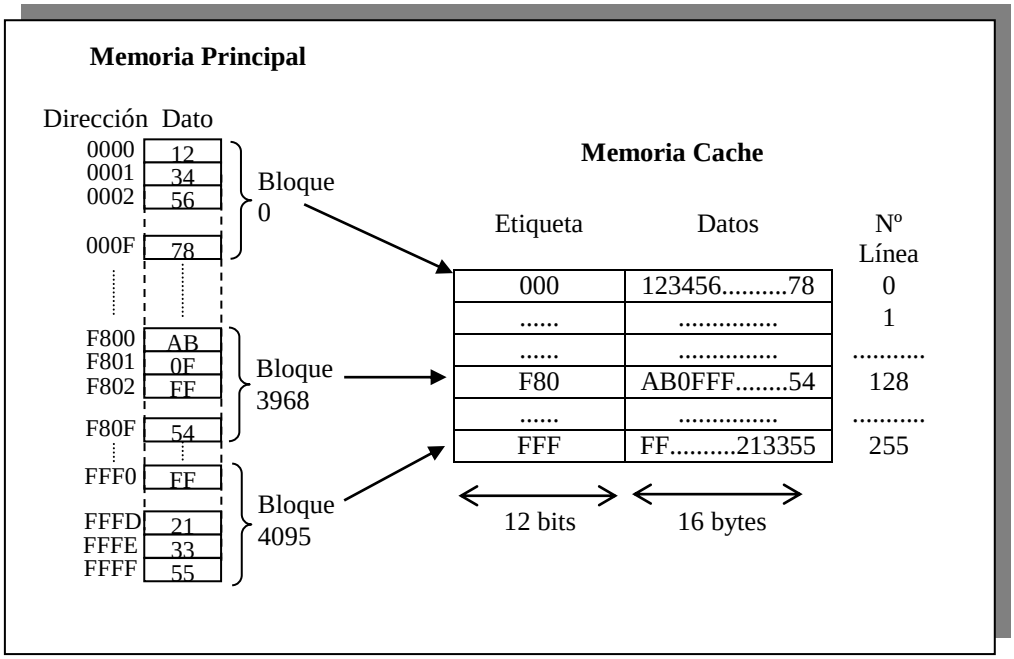


Figura 7. Ejemplo de correspondencia asociativa.

El principal inconveniente de esta técnica es la necesidad de una circuitería, bastante compleja, para examinar simultáneamente los campos de etiqueta de todas las líneas de la memoria cache.

Correspondencia asociativa por conjuntos.

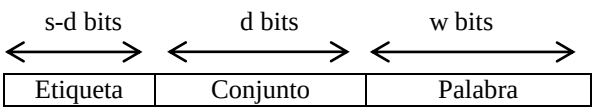
Este técnica es un compromiso que trata de aunar las ventajas de las dos técnicas vistas anteriormente. Para su aplicación, la memoria cache se divide en T conjuntos de L líneas cada uno. Así, las relaciones que se tienen son

$$C = T \times L \qquad i = j \text{ módulo } T$$

donde i = número de conjunto de la memoria cache

j = número de bloque de la memoria principal

Con esta correspondencia, el bloque B_j puede asociarse a cualquiera de las líneas del conjunto i. En este caso, la lógica de control de la cache interpreta una dirección de memoria con tres campos



donde

w bits = Palabra dentro de un bloque de memoria principal

s bits = Identifica un bloque de la memoria principal

d bits = Conjunto de la memoria cache

s-d bits = Etiqueta asociada a las líneas del conjunto d bits

Para saber si una dirección está o no en memoria cache, lo primero que se hace es localizar el conjunto donde estaría almacenada la dirección a buscar como si se tratara de una correspondencia directa, es decir, aplicando la función de correspondencia. Una vez que se sabe el conjunto, se tiene que mirar si la etiqueta de la dirección está o no en la líneas que forman el conjunto. Esta comparación se realiza simultáneamente como si se tratara de correspondencia asociativa. En el caso de que sí esté, se accede a la palabra en cuestión y sino se accederá a memoria principal. La organización de la memoria cache con correspondencia asociativa por conjuntos se muestra en la siguiente figura 8.

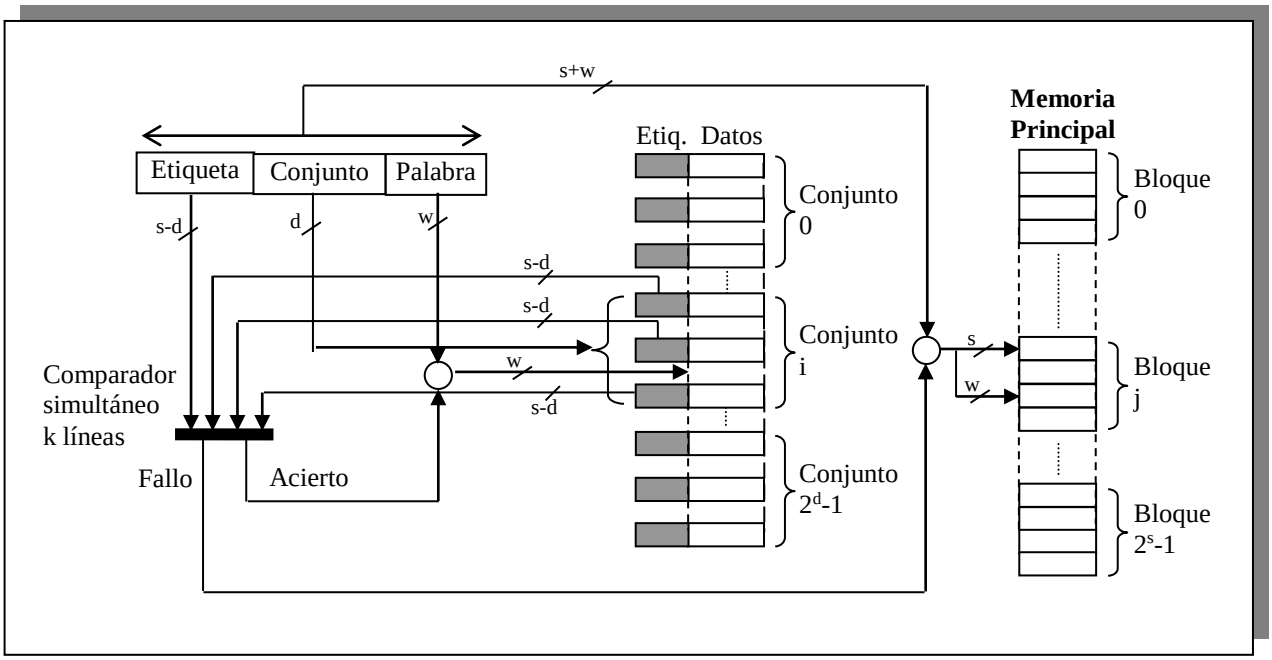


Figura 8. Correspondencia asociativa por conjuntos.

Según el diagrama, es fácil ver que cuando T=C y L=1, la técnica asociativa por conjuntos se reduce a la correspondencia directa, y cuando T=1 y L=C, la técnica asociativa por conjuntos se reduce a la correspondencia asociativa. El uso de dos líneas por conjunto (T=C/2, L=2) es el caso más común, mejorando significativamente la tasa de aciertos respecto de la correspondencia directa. La asociativa por conjuntos de cuatro vías produce una mejora adicional con un coste añadido relativamente pequeño. Un incremento adicional en el número de líneas por conjunto tiene poco efecto.

Siguiendo el ejemplo si utilizamos la correspondencia asociativa por conjuntos tendremos que la dirección de memoria se codificará como sigue

Etiqueta	Conjunto	Palabra
6 bits	6 bits	4 bits

En la figura 9 se muestra un ejemplo de correspondencia asociativa por conjuntos con cuatro líneas por cada conjunto, denominada asociativa por conjunto de 4 vías. En él, la memoria cache se divide en 64 conjuntos con 4 líneas por conjunto.

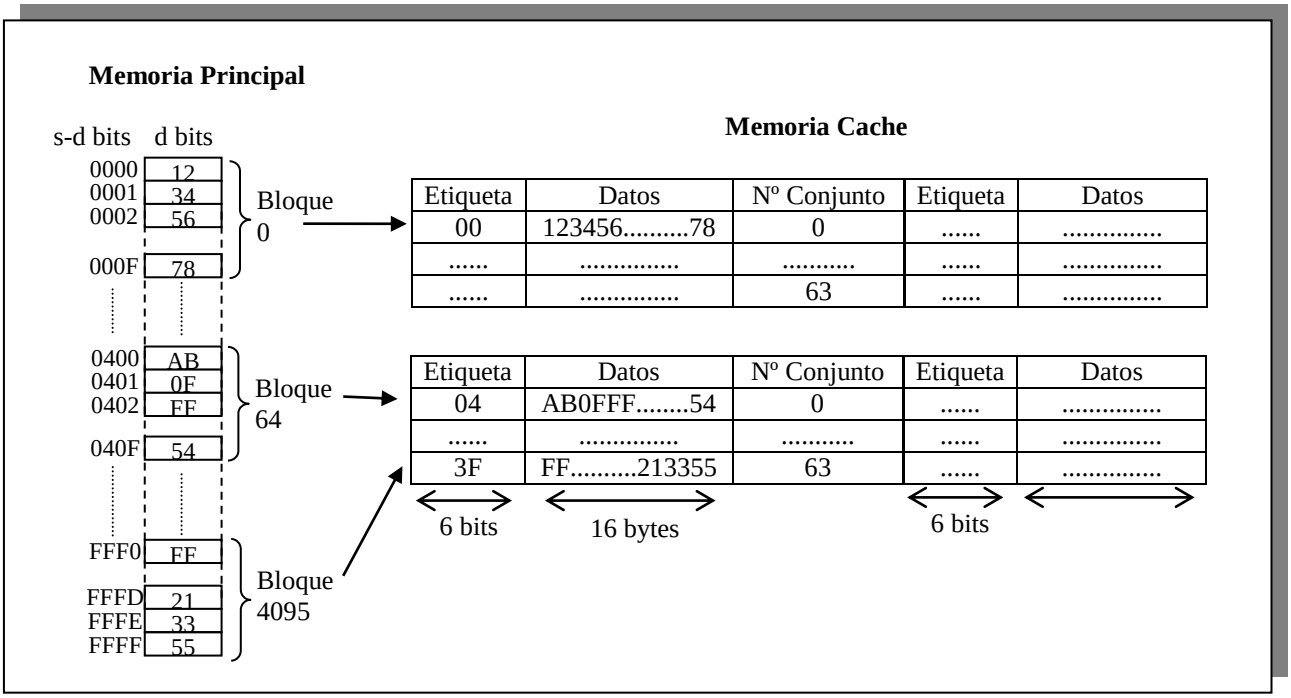


Figura 9. Ejemplo de correspondencia asociativa por conjuntos.

Algoritmos de sustitución

Cuando un nuevo bloque se transfiere a la memoria cache debe sustituir a uno de los ya existentes si está ocupada la línea que le puede corresponder. En el caso de la correspondencia directa la línea está determinada unívocamente y por lo tanto no se puede realizar ninguna elección de qué línea se puede eliminar. Por el contrario, en las técnicas de tipo asociativo es necesario un algoritmo de sustitución. Para obtener tasas de transferencia elevada, estos algoritmos deben realizarse por hardware. Entre los diferentes algoritmos que se han propuesto destacan los siguientes:

- FIFO (*First In, First Out* - Primero en entrar, primero en salir)

Elimina el bloque de la línea más antigua, según el parámetro tiempo de carga en memoria.

En la figura 10 podemos ver un ejemplo del algoritmo FIFO donde un programa accede a diferentes direcciones de memoria principal generando una secuencia de referencias de bloques. La memoria cache, con función de correspondencia asociativa, está formada por 5 líneas que inicialmente están vacías.

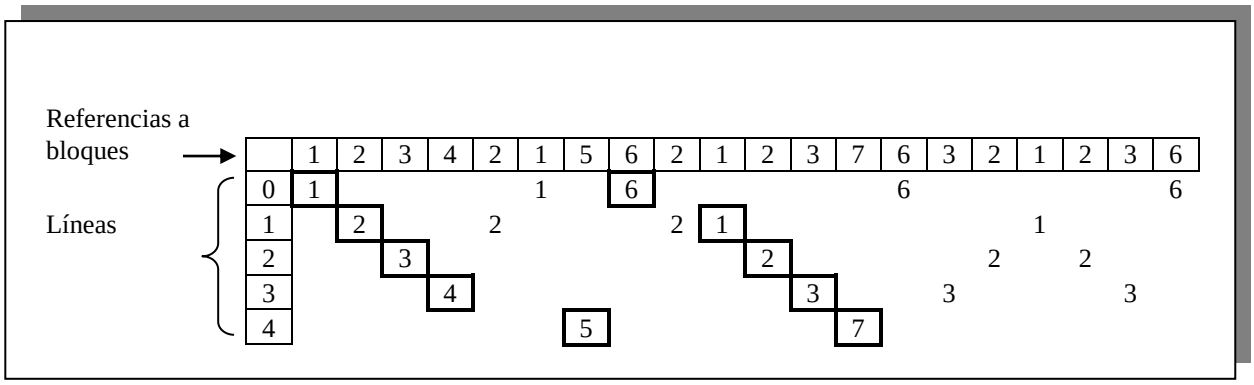


Figura 10. Ejemplo de algoritmo FIFO.

Como podemos ver en el ejemplo, se han producido 10 aciertos y 10 fallos en la memoria cache.

- LRU (*Least Recently Used* - Menos recientemente usado)

Elimina el bloque de la línea que hace más tiempo que no se ha utilizado.

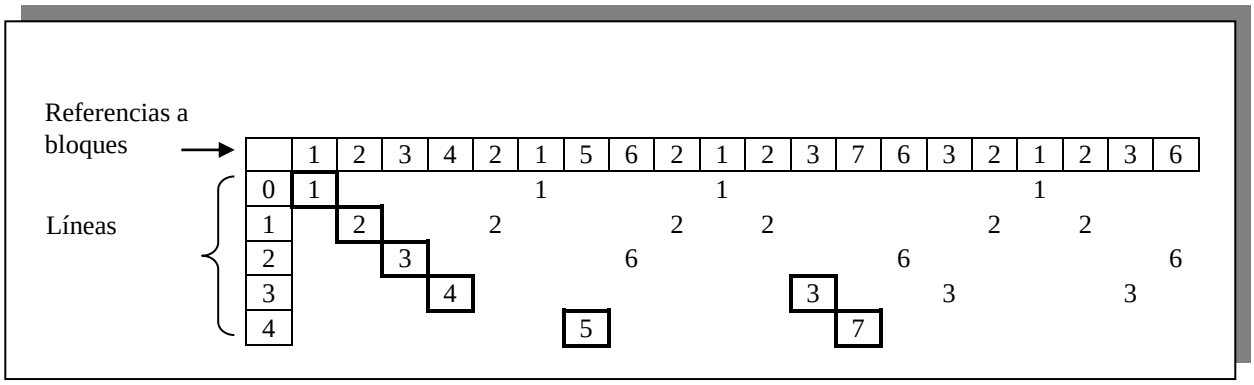


Figura 11. Ejemplo de algoritmo LRU.

Como podemos ver en la figura 11, se han producido 12 aciertos y 8 fallos en la memoria cache.

- LFU (*Least Frecuently Used* - Menos frecuentemente usado)

Elimina el bloque de la línea de menor índice de uso. Asociada a cada línea hay un contador de uso que se inicializa cada vez que se copia un bloque en memoria cache. En el caso de que hayan líneas con el mismo índice y se tenga que sustituir alguna se aplica el algoritmo FIFO.

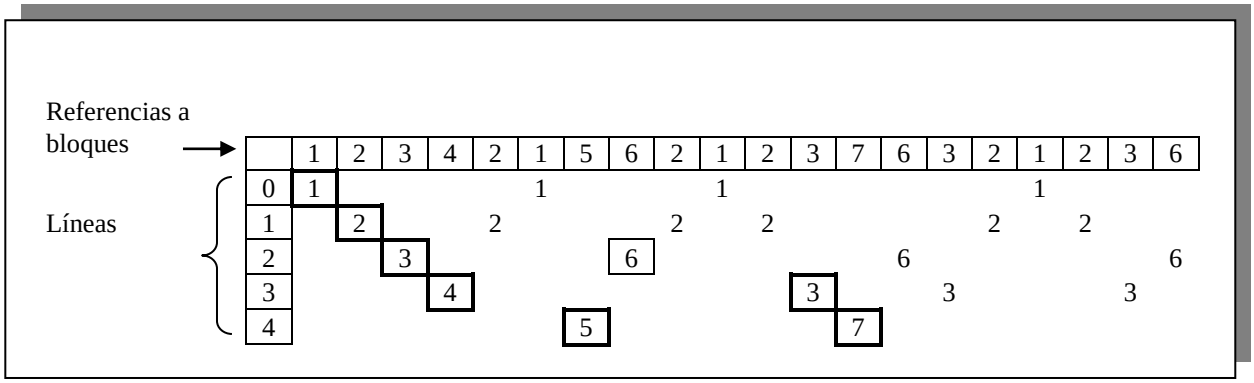


Figura 12. Ejemplo de algoritmo LFU.

Como podemos ver en la figura 12, se han producido 12 aciertos y 8 fallos en la memoria cache.

Estos algoritmos se estudian con mayor detalle en asignaturas de Sistemas Operativos.