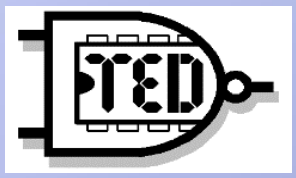
FACULTAD DE INGENIERIA
UNIVERSIDAD NACIONAL DE JUJUY

U.N.Ju. – F.I.

INGENIERÍA INFORMÁTICA

LICENCIATURA EN SISTEMAS

TÉCNICAS Y ESTRUCTURAS DIGITALES



MEMORIA CACHÉ

Necesidad de la Caché

La arquitectura de los últimos microprocesadores está orientada a mejorar el rendimiento, de manera que se ejecuten más instrucciones por unidad de tiempo.

Para llevar a cabo este objetivo los nuevos microprocesadores se apoyan en tres recursos fundamentales:

- Arquitectura superescalar, característica de la arquitectura RISC. Paralelismo explícito. Se compone de instrucciones sencillas.
- Supersegmentación, es decir, segmentación con elevado número de etapas. Esta técnica precisa de técnicas sofisticadas para eliminar riesgos, especialmente en las instrucciones que contengan saltos condicionales.
- Potenciamiento de la memoria caché, para aumentar la velocidad de la memoria.

Un procesador segmentado ejecuta cada instrucción en cinco etapas que son:

- Búsqueda de la instrucción (*Fetch*): se accede a la memoria para buscar la instrucción.
- Decodificación: es trabajo de la CPU y su ciclo es de 10 ns cuando trabaja a 100 MHz.
- Búsqueda de los operandos: se accede a la memoria (si es necesario).
- Ejecución de la instrucción: realizada por la CPU.
- Escritura del resultado: se accede de nuevo a la memoria.

Necesidad de la Caché

De estas cinco etapas hay tres que consisten en un acceso a la memoria principal (DRAM), donde están las instrucciones y los datos (búsqueda de la instrucción, búsqueda de los operandos, escritura del resultado) y las otras dos son propias del procesador. Las dos etapas que afectan al procesador se realizan en un ciclo cada una. Las otras tres etapas ocupan un tiempo equivalente al de acceso a la memoria principal.

Si suponemos que la memoria principal DRAM trabaja con un tiempo de acceso de 50 ns, los tres accesos a la memoria principal suman un total de 150 ns. Si además el microprocesador trabaja a 1 Ghz, su periodo es de 1 ns, por lo que el resultado anterior representa un desequilibrio notable entre etapas. Este hecho es la causa fundamental de un mal rendimiento.

Hay una laguna entre la tecnología de construcción de procesadores y la de la memoria.

Para que el rendimiento sea óptimo, el objetivo es que todas las etapas duren lo mismo. Debemos disponer de una memoria más rápida. Las cachés son ultrarápidas pero de poca capacidad y muy caras, por lo que no se pueden sustituir las DRAMs por cachés.

Necesidad de la Caché

Por lo que debemos emplear la jerarquía de memoria (Ver Fig. 1), la cual consiste en interponer entre la CPU y la memoria DRAM una memoria ultrarápida (caché).

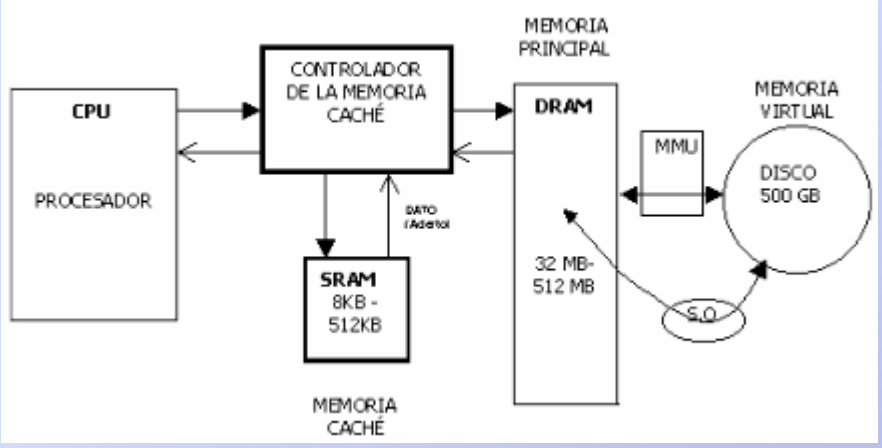


Figura 1. Jerarquía de memorias.

La memoria caché es una SRAM (RAM estática) que tiene un tamaño comprendido entre 8 KB y 512 KB mientras que la Memoria Principal puede alcanzar varios GB.

Como se estudió en el tema de la Memoria Virtual, cuando la CPU solicita una información no presente en la Memoria Principal, la MMU trae el bloque necesario de la Memoria Virtual. Este mismo procedimiento también es empleado para el comportamiento entre la Memoria Principal y la memoria Caché.

Necesidad de la Caché

La CPU se relaciona con la Memoria Caché y si ésta contiene lo solicitado se tardan unos pocos ns en el acceso. Si hay "fallo" se debe acceder a la Memoria Principal.

El movimiento de datos se genera de forma que produciéndose una ausencia, la caché recibe de la Memoria Principal el dato pedido y otros contiguos, que previsiblemente va a pedir la CPU. Esto es predecible dependiendo de la programación, ya que puede estar construida mediante las reglas clásicas de la contigüidad espacial y temporal. Es fácil prever la siguiente información que solicitará la CPU debido a la localidad espacial; la CPU tiende a requerir datos que estén en posiciones cercanas físicamente.

Por esto, existe un principio totalmente empírico, pero que la experiencia demuestra que se aplica ampliamente denominado Vecindad de las Referencias, y que consta, a su vez de dos subprincipios:

- Vecindad espacial.
- Vecindad temporal.

VECINDAD

Vecindad espacial: los programas solicitan datos o instrucciones cuyas direcciones en memoria están cercanas a los datos o instrucciones recientemente direccionados. Este principio se cumple, ya que los programas se escriben y ejecutan de forma secuencial y los segmentos de datos suelen tener las variables adyacentes.

Vecindad temporal: los programas tienden a usar datos más recientes. Cuando más antigua sea la información, más improbable es que un programa la solicite.

Si la caché tiene el dato, solo penaliza el tiempo de acceso a la misma, pero cuando la caché no dispone del dato solicitado, el tiempo empleado es el de acceso a la Memoria Caché más el de acceso a la Memoria Principal.

Almacenar en una caché una secuencia de instrucciones y datos usados ahorra una ingente cantidad de accesos a memoria y acelera significativamente la ejecución del programa. Hay que tener en cuenta que, de entrada, no se requiere ir 1000 veces a la memoria principal a buscar instrucciones del bucle, ya que se almacenan en la caché. Desde ésta se le envían al procesador cuando las requiere, a mayor velocidad posible.

TIEMPO DE ACCESO

Así por ejemplo, si empleamos una memoria caché con un tiempo de acceso *tc*, con una tasa de acierto del 90% y una memoria principal con un tiempo de acceso *tm*, la fórmula que refleja el tiempo medio de acceso al sistema de memoria es:

$$t = 0.9 * tc + 0.1 * (tc + tm)$$

Si suponemos que el tiempo de acceso a la memoria caché y el tiempo de acceso a la Memoria Principal es de 5 y 50 ns respectivamente, el resultado es:

$$t = 0.9 * 5ns + 0.1 * (5ns + 50ns) = 10\ ns$$

Un sistema con estas características tiene como tiempo de acceso medio 10 ns, por lo que se ve reflejada la importancia del algoritmo de intercambio de información, es decir, la capacidad que tiene el sistema de predecir las siguientes peticiones de información.

EFICIENCIA DE LA CACHÉ

Para conseguir un buen comportamiento de la caché hay que saber anticiparse a las necesidades de información de la CPU para almacenarla previamente.

Cuando la información requerida por el procesador no se halla en la caché hay una penalización de tiempo pues hay que localizarla en la Memoria Principal. Esto supone un importante decremento del rendimiento de la computadora. Si la información no está en la caché, el procesador debe permanecer inactivo muchos ciclos, por ello es muy importante reducir los fallos de la caché.

La eficiencia de la caché depende de los algoritmos que se utilizan para cargarla con la información que precisará la CPU próximamente. Dichos algoritmos implementan los "principios de localidad" mediante software.

Se denomina eficiencia de la caché a la relación entre su Tiempo de Acceso a la Caché (Tc) y el Tiempo de Acceso Medio (Tmedio) necesario para realizar un acceso en el sistema jerárquico. Así:

$$\text{Eficiencia} = \frac{T_C}{T_{medio}}$$

El Tmedio es función del porcentaje de aciertos que se produzcan en la caché. Existen dos parámetros que influyen en la eficiencia:

$$\text{Probabilidad de presencia } (h) = \frac{\text{Número de presencias en caché}}{\text{Número de accesos en caché}}$$

$$\text{Probabilidad de ausencia } (1\text{-}h) = \frac{\text{Número de ausencias en caché}}{\text{Número total de accesos en caché}}$$

$$T_{medio} = h \cdot T_C + (1-h) \cdot (T_C + T_{MP})$$

Se denomina Factor de Velocidad a la relación entre el Tiempo de Acceso de la Memoria Principal (TMP) y el Tc.

$$\text{Factor de Velocidad} = \frac{T_{MP}}{T_C}$$

Finalmente, se denomina Índice de Mejora a la relación entre el Tiempo de Acceso sin Caché y el Tiempo de Acceso con Caché, este índice nos indica si es conveniente el uso de la Caché y en qué medida.

$$\text{Índice de Mejora} = \frac{T_{MP}}{T_{medio}}$$

FUNCIONAMIENTO

PRINCIPIO DE FUNCIONAMIENTO DE LA CACHÉ

Una caché está estructurada en tres bloques importantes:

- **BLOQUE DE ETIQUETAS (RAM-CAM):** Es una memoria de acceso por contenido. No se accede por dirección de memoria, sino que, se compara el valor o dato con los que hay dentro de la memoria y así sabemos si se encuentra o no contenido en ella.

- **BLOQUE DE DATOS ASOCIADOS (SRAM):** Es un conjunto de datos de forma que a cada dato le corresponde una etiqueta. Por ejemplo: dato 0 → etiqueta 0.

  Si hay acierto, la etiqueta que coincide con el dato introducido, devuelve el dato asociado a la misma.

- **LÓGICA DE CONTROL:** Comparadores de “n" bits, tantos como tenga la etiqueta.

Véase Fig. 2.

FUNCIONAMIENTO

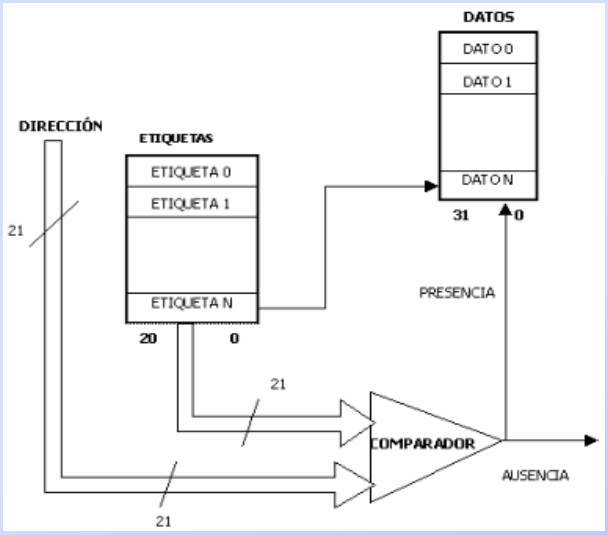


Figura 2. Principio de funcionamiento de la caché.

El bus de direcciones va a la caché, que tiene dos partes: la de las etiquetas y la de datos. Hay N posiciones de etiquetas y cada etiqueta tiene 21 bits. Los 21 bits de la etiqueta se corresponden con los 21 bits de más peso de la dirección, por eso, el comparador recoge las N etiquetas y las compara con los 21 bits de más peso de la dirección. Si alguna coincide, el comparador devuelve un uno, lo que implica presencia. En cambio, si no coincide ninguna, el comparador devuelve un cero lo que implica ausencia, lo que significa que esa dirección no está contenida en la caché.

Aunque hay múltiples organizaciones en la caché concretamente en el ejemplo de la figura 2 hay un dato por etiqueta, por lo tanto N datos, existiendo una correspondencia etiqueta-dato. Como por ejemplo: ETIQUETA1 → DATO1.

Si el comparador devuelve un uno (presencia), el dato correspondiente a la etiqueta es transferido a la CPU reflejando los bits restantes, la posición en la que se encuentra el dato. Este sistema es más rápido que el de localizar una dirección en memoria principal.

Si el comparador devuelve un cero (ausencia) se accede a la memoria principal.

CONEXIONADO

La Memoria Caché se puede conectar a la CPU en serie o en paralelo.

CONEXIÓN EN SERIE

La CPU sólo se conecta con la caché por lo que todas las peticiones que hace la CPU al bus del sistema son a través de la memoria caché.

Por lo tanto todo lo que necesita la CPU del sistema se lo proporciona la memoria caché y cómo es de tamaño y tiempo de acceso reducido cada vez que el dato está almacenado en la caché, el tiempo de acceso es muy reducido y evita manejar el bus del sistema.

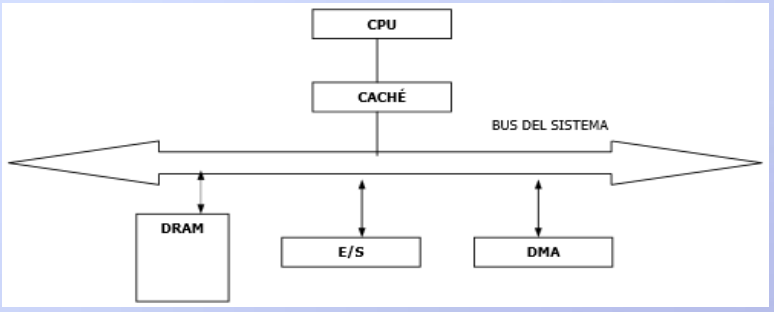


Figura 3. Conexión en serie.

Si la memoria caché contiene el dato solicitado por la CPU, la parte superior de la Fig. 3 funcionará de manera independiente y en pocos nanosegundos proporcionará la información, liberando de la búsqueda a la parte inferior de la figura.

De esta forma si el bus del sistema está desocupado, simultáneamente que la CPU ejecuta instrucciones, los módulos de entrada/salida pueden estar trabajando con la memoria principal. Se permite el paralelismo.

El inconveniente de la conexión en serie es la penalización de tiempo ya que cuando la información que necesita la CPU no está en la caché tiene que trasladar la petición al bus del sistema para poder acceder a la memoria principal.

Otro inconveniente es que la caché es de uso obligatorio; no se puede desconectar la caché y conectar la CPU al sistema.

Cuando el bus del sistema queda libre, puede ser utilizado por todos los elementos que dependan de él.

CONEXIONADO

CONEXIÓN EN PARALELO

En la conexión en paralelo, todo depende del bus del sistema:

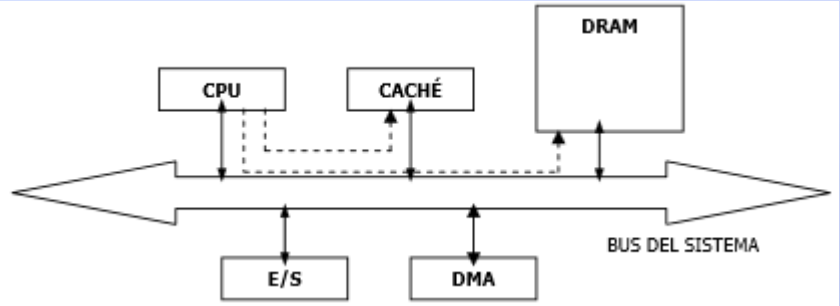


Figura 4. Conexionado en Paralelo.

Cada vez que la CPU realiza una petición, la envía simultáneamente a la caché y a la Memoria Principal. Si se encuentra la información contenida en la caché es entregada al bus en pocos ns avisando a la memoria principal para que no continúe con la búsqueda de la información. De esta forma se parará el ciclo. Si la memoria caché no tiene el dato pedido, la memoria principal sigue trabajando, con lo cual no hay penalización de tiempo, lo que es una gran ventaja.

Este sistema se utiliza mucho porque cuenta con otra ventaja, y es que la caché es opcional, es decir, se puede añadir o eliminar sin alterar el funcionamiento del sistema.

El inconveniente es que la CPU realiza todas las peticiones por el bus del sistema, quedando sobrecargado por el continuo flujo de información, lo cual deja muy poco espacio libre para E/S y DMA, ya que el bus del sistema se libera muy pocas veces (Ver Fig. 4).

ARQUITECTURA

ARQUITECTURA DEL SUBSISTEMA DE MEMORIA CACHÉ

Las características fundamentales que determinan un subsistema caché son las siguientes:

- Tamaño de la caché.
- Organización.
- Actualización de la caché.
- Actualización de la memoria principal.

EL TAMAÑO DE LA CACHÉ

El tamaño de la caché suele oscilar entre 8 Kb y 512 Kb. En muchas ocasiones, no se consigue aumentar el rendimiento mediante cachés de mayor capacidad. Lo que influye son los algoritmos de transferencia de la memoria caché.

Aumentos importantes del tamaño de la caché suponen variaciones pequeñas en el porcentaje de acierto, porque el porcentaje de aciertos se basa en el algoritmo. Según los algoritmos, la mejor capacidad de la caché está entre 32 Kb y 256 Kb.

Cuanto mayor sea la cantidad de RAM de un subsistema de caché, mayor será su costo. Hay que usar más chips de una memoria muy rápida, que es cara. Además, el espacio ocupado en placa es mucho mayor por la baja densidad de integración, aumentando la complejidad del diseño físico.

El precio sube de forma exponencial: aumentando el tamaño x8, la tasa de aciertos sólo aumenta un 4%. Se refleja en la Fig. 5:

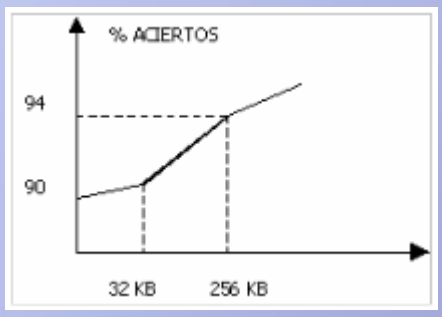


Figura 5. Gráfica número de aciertos respecto tamaño de la caché

ORGANIZACIÓN

Existen cuatro tipos de organización de las memorias caché:

Sectorizada: aunque puede resultar sencilla, no suele emplearse por ser más eficientes las organizaciones restantes.

Totalmente asociativa: Se caracteriza porque cualquier posición de la memoria principal se puede ubicar en cualquier posición de la memoria caché.

Se usan comparadores de 32 bits, muy caros, con buenos algoritmos.

Todas las direcciones de la memoria principal que quepan se pueden guardar sin ningún orden preestablecido, no hay normas, hay una flexibilidad total.

Esta flexibilidad es un problema grave, puesto que la caché necesita los 32 bits de la dirección para compartirla con la etiqueta.

Cada partición de la caché se puede ubicar en cualquier parte de la memoria principal. Como hay flexibilidad total, la etiqueta ha de contener TODOS los bits de la dirección de la memoria principal a los que corresponden los datos.

Luego el inconveniente de este tipo de organización de la memoria caché es que tiene que tener apuntadas todas las direcciones en la zona de etiquetas y por lo tanto éstas serán muy largas, lo cual implica gran cantidad de comparaciones simultáneas. Esto es complejo de realizar en un tiempo reducido. En consecuencia el comparador va a ser muy lento y caro ya que va a necesitar muchos bits.

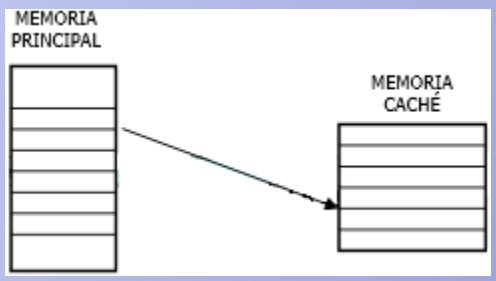


Figura 6. Organización de memoria caché asociativa.

ORGANIZACIÓN

Asociativa de una vía o Correspondencia Directa: Suponemos que la memoria principal tiene 256 KB y la caché 256 Bytes. La memoria principal se divide en bloques de 256 bytes. Habrá 1K bloques, es decir, 1024 bloques.

Cada posición de un bloque de la memoria principal sólo puede ir a la misma posición de la caché.

Solo es necesario precisar cuál es el bloque, porque sabiendo el bloque ya se conoce la posición de la caché debido a que todas las direcciones de un bloque están en la misma posición que en la caché.

La ventaja es que para conexionar la memoria principal necesitamos que el bus de direcciones tenga 18 líneas. La etiqueta, por tanto, sólo necesita 10 bits: 2^{10} = 1K es decir, lo que ocupa un bloque. Sólo se necesita los bits necesarios para definir el bloque y se ahorra así los 8 bits.

Como sólo cabe una dirección de cada bloque en una posición, se tiene que machacar la dirección anterior, cada vez que se quiere modificarla. La caché estará sufriendo continuamente modificaciones.

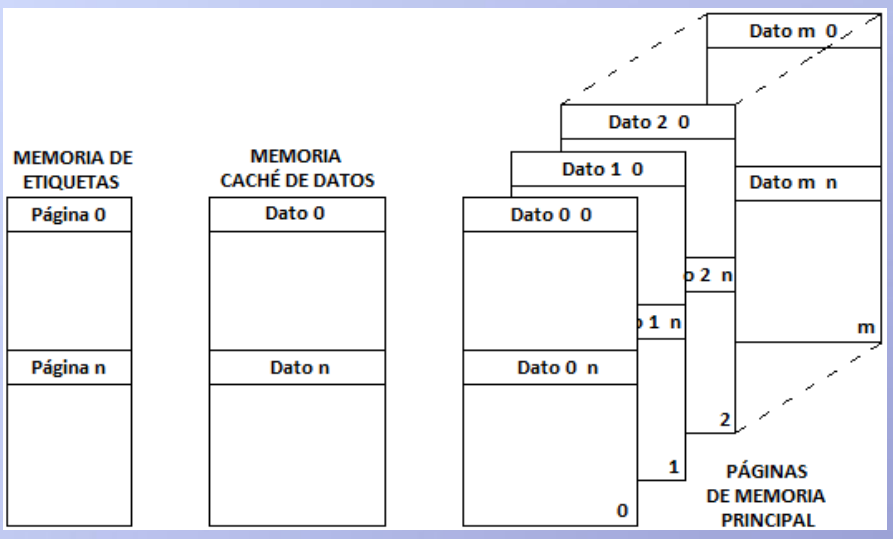


Figura 7. Organización asociativa de una sola vía de la memoria caché.

ORGANIZACIÓN

Asociativa de "n" vías: Su funcionamiento es similar al de una vía pero la caché se va a descomponer en varias vías, no en una sola.

La memoria principal se divide en fragmentos iguales a cada uno le corresponde una de las vías funcionando de la misma manera que en el caso anterior, incorporando la ventaja de que no es necesario machacar las posiciones de memoria inmediatamente que surja una modificación.

Con la caché asociativa de N vías, se ahorra gran cantidad de bits debido a que hay correspondencia entre las posiciones de las páginas de la memoria principal y las posiciones de la caché.

Esta organización de la memoria ofrece un mayor rendimiento pero las vías son de menor tamaño. Dependiendo de la aplicación será mejor utilizar la memoria asociativa de 1 vía o la de n vías.

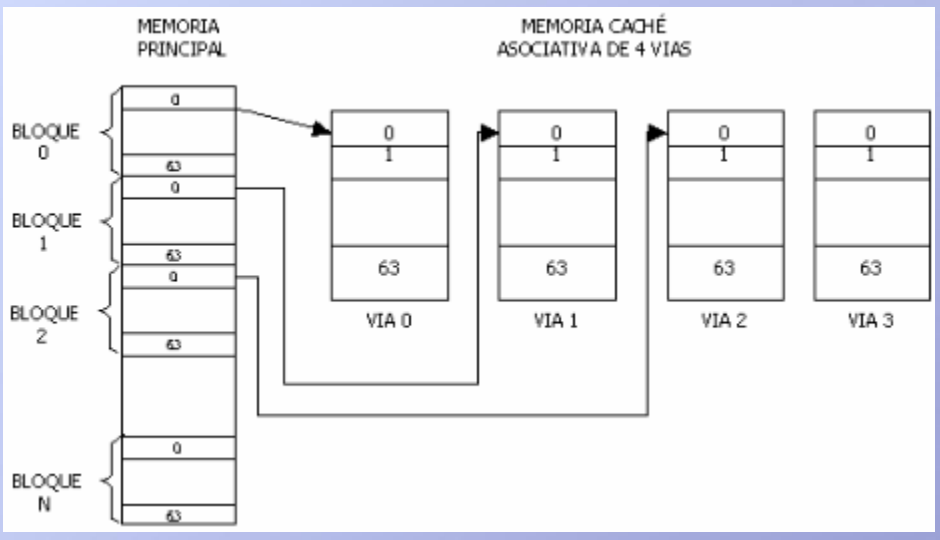


Figura 8. Organización asociativa de "N" vías de la memoria caché.

ACTUALIZACIÓN

ACTUALIZACIÓN DE LA MEMORIA CACHÉ

En el procesador, cuando se produce una ausencia en la caché se busca la información de una línea completa en la memoria principal y se carga en la caché, en una vía que esté libre.

Si fuera totalmente asociativa, cualquier línea se podría almacenar en cualquier posición de la caché. Si es asociativa de una vía sólo hay una posibilidad. Si es de varias vías, como el Pentium, hay varias posibilidades.

Si se necesita introducir una posición y hay alguna vía vacía, ésta se ocupa. Si hay una posición que las cuatro vías tengan ocupada, se aplica uno de varios algoritmos de reemplazo, para extraer una de ellas y machacarla.

Existen diversos algoritmos de reemplazo para una caché: LRU, FIFO, LIFO, RANDOM, etc. La elección del algoritmo de reemplazo junto con la organización de la caché, son dos factores que afectan, de manera muy importante, el rendimiento de la caché.

Los algoritmos más importantes son:

RANDOM: Aleatoriamente se elige y se machaca una de las posiciones ocupadas de una de las cuatro vías. El inconveniente es que quizá se machaque la información que a continuación se necesite. Se dice que es el algoritmo de las cachés de bajo coste.

LRU: Se elimina la posición de la vía que menos se haya empleado últimamente, ya que suponemos que es la que menos se va a seguir utilizando. Su funcionamiento se basa en dos bits que apuntan a la vía que menos se ha empleado.

La CPU siempre se dirige a la caché. Si va a leer un dato desde la caché no tiene ningún inconveniente, pero si la CPU necesita escribir en la caché y modificar una de sus posiciones, la CPU da por concluida su operación resultando que esa posición de la caché tiene una imagen, en memoria principal, que no tiene constancia de esta modificación, por lo que hay una discordancia. Esto puede dar lugar a errores graves. Por lo tanto se debe escribir lo que se haya modificado en la caché en la memoria principal.

ACTUALIZACIÓN

ACTUALIZACIÓN DE LA MEMORIA PRINCIPAL

La memoria principal se actualiza mediante uno de estos tres métodos (Fig. 9):

- Actualización por escritura inmediata: Cada vez que la CPU modifica la caché, ésta última manda una orden al bus del sistema y se transfiere la información a la CPU, consiguiendo que no haya errores de coherencia y actualizando así, inmediatamente, la memoria principal. Muchas veces la escritura de la CPU es repetitiva, ya que la escritura en la caché es continua, y bloquea el bus del sistema. Es normalmente seguro, pero baja el rendimiento por el tiempo empleado para escribir un dato en memoria principal. De esta forma se evitan muchos errores.
- Actualización por escritura diferida: La caché dispone de registros intermedios donde carga temporalmente las modificaciones que ha habido en la caché. Actualiza la memoria principal cuando el bus del sistema está libre, sólo hay que esperar a que el bus del sistema esté inactivo. Puede existir falta de coherencia mientras se espera y los periféricos pueden leer datos erróneos de la memoria principal. Es más rápido, pero hay un pequeño riesgo de fallo.
- Actualización por escritura obligada: La actualización de memoria principal se produce cuando no queda otro remedio, por lo que no hay nunca fallo.

  Hay que actualizar la memoria obligatoriamente cuando:

  - Se accede a una posición de la memoria principal modificada en la caché por la CPU. Antes de dar paso a esa lectura, hay que escribir el dato modificado.
  - Cuando hay que eliminar una línea en la caché porque está llena, en la cual hay un dato modificado. Antes de borrarlo hay que enviar dicho dato a la memoria principal.

ACTUALIZACIÓN

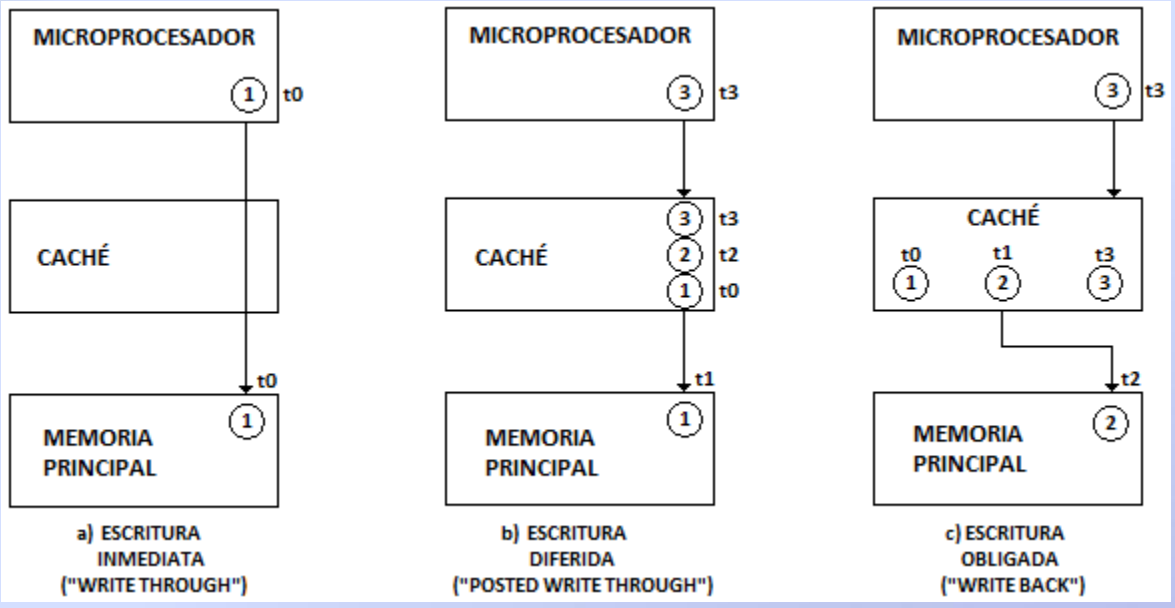


Figura 9. Tres métodos de actualización de la memoria principal de un subsistema caché.

PROTOCOLO MESI

En los sistemas multiprocesador puede haber varias cachés, por tanto, puede suceder que una misma posición de la memoria principal la están empleando dos CPU´s y como consecuencia, permanecer en las dos cachés. Cuando se plantea esta situación, surge la necesidad de asegurar que cualquier acceso a la memoria lea el dato más actualizado. Para evitar esta inconsistencia de la caché Intel desarrolló el protocolo MESI (**M**odified **E**xclusive **S**hared **I**nvalid).

El protocolo MESI asigna cuatro estados diferentes a cada línea (gestionados por los bits MESI), que definen si una línea es válida (es decir, si existe presencia o no), si está disponible para otras cachés, o ha sido modificada. Estos estados pueden ser modificados bien por el propio procesador, o bien por unidades lógicas externas tales como otros procesadores o el controlador caché L2.

Los posibles estados de la línea son:

M (Modificado): La línea que lleva dicha M está modificada por una escritura del procesador y a la espera de actualizar la memoria principal si es preciso.

E (Exclusiva): La línea sólo la tiene una caché, y está sin modificar. La memoria principal tiene una copia.

S (Simultáneo): Esta línea está repetida en otras cachés. Si se escribe en una de ellas, las demás se invalidan automáticamente.

I (Inválido): Esta línea está invalidada. La lectura de esta posición por parte del procesador genera una ausencia y por tanto un llenado con los datos que provienen de la memoria principal. Si el procesador escribe en esta posición, se procede a actualizar la memoria principal de inmediato.