Sistema de memoria

Miquel Albert Orenga
Gerard Enrique Manonellas

PID_00177073

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

Todo computador necesita un sistema de memoria para almacenar los programas que se ejecutan y los datos necesarios para ejecutar estos programas. Desde el punto de vista del programador, sería deseable disponer de cantidades ilimitadas de memoria y de velocidad ilimitada, si fuera posible, para almacenar el programa que se quiere ejecutar y los datos necesarios; eso permitiría al programador hacer la tarea de escribir programas sin tener que enfrentarse a ningún tipo de limitación. Lógicamente, este deseo no es factible y las cantidades de memoria de que dispone un computador tienen una limitación en capacidad y velocidad.

La cantidad de memoria que puede tener un computador responde básicamente a un factor de coste: cuanta más memoria instalada, más elevado es el coste. De manera parecida, la velocidad de la memoria también depende del coste. Las memorias más rápidas tienen un coste más elevado, pero no se puede conseguir toda la velocidad necesaria simplemente incrementando el coste; hay además un factor tecnológico que limita la velocidad de la memoria: no podemos adquirir memoria más rápida que la que está disponible en el mercado en un momento dado.

Existen diferentes tipos de memorias, con capacidades y tiempos de acceso diferentes. En general, cuanta más capacidad de almacenamiento tiene una memoria, mayor es el tiempo de acceso. Es decir, las memorias con gran capacidad son memorias lentas, mientras que las memorias rápidas (tiempo de acceso pequeño) suelen tener poca capacidad de almacenamiento. Las memorias rápidas son más caras que las memorias lentas. Por ello, los diseñadores de computadores deben llegar a un compromiso a la hora de decidir cuánta memoria ponen en sus diseños y de qué velocidad o tiempo de acceso.

En los últimos años, la evolución de la tecnología ha permitido reducir mucho el espacio necesario para almacenar un bit de información. Eso ha originado que el tamaño de las memorias aumente mucho con relación al espacio físico que ocupan y que se reduzca el precio que se ha de pagar por un bit de información. Así, los discos duros han pasado de los 20 Mbytes de capacidad a mediados de década de los ochenta a los 2.000 Gbytes a finales del 2010 (100.000 veces más), aunque ocupan el mismo espacio físico (incluso son un poco más pequeños) y cuestan casi lo mismo. Esto ha representado una reducción importante en el precio por bit. Este ha sido un factor muy importante para que los computadores actuales incorporen mucha más memoria que los computadores de hace treinta años. Por lo tanto, a la hora de diseñar un sistema de memoria, hay que tener presentes las características de capacidad, velocidad (y tiempo de acceso) y coste por bit.

Otras cuestiones también importantes que cabe considerar son la localización, la organización, el método de acceso o la tecnología de fabricación.

Objetivos

Con el estudio de este módulo se pretende que el estudiante alcance los objetivos siguientes:

1. Conocer las características básicas de una memoria.

2. Comprender los conceptos básicos sobre la organización de la jerarquía de memoria de un computador.

3. Analizar cómo se gestionan los datos entre los diferentes niveles de la jerarquía de memorias, especialmente en la memoria caché.

4. Conocer las diferentes tecnologías utilizadas para implementar los diferentes tipos de memorias utilizados en un computador.

1. Características de las memorias

Las características más importantes de los diferentes tipos de memoria son la localización, la capacidad, el método de acceso, la organización de los datos en una memoria, el tiempo de acceso y velocidad, y el coste. Las estudiaremos en este apartado.

1.1. Localización de la memoria

Podemos clasificar los tipos de memoria según su localización dentro del computador. Básicamente, se pueden distinguir: memoria dentro del chip del procesador, memoria interna (memoria en la placa base del computador) y memoria externa.

Dentro del chip del procesador habitualmente están los registros y uno o varios niveles de memoria caché.

La memoria interna corresponde a la memoria principal (memoria RAM del computador) y adicionalmente un nivel de memoria caché o varios.

La memoria externa corresponde a los dispositivos de almacenamiento secundario, como discos duros, unidades ópticas (CD-ROM, DVD, o Blu-ray), unidades de cinta, etc.

Ved también

En este módulo profundizaremos en el estudio del sistema de memoria formado por la memoria que se encuentra en el procesador y la memoria interna, y haremos una revisión más general de la memoria externa.

1.2. Capacidad de la memoria

La capacidad (o tamaño de la memoria) hace referencia a la cantidad de información que se puede almacenar. La unidad utilizada para especificar la capacidad de almacenamiento de información es el byte (1 byte = 8 bits), y a la hora de indicar la capacidad, se utilizan diferentes prefijos que representan múltiplos del byte.

En el sistema internacional de medidas (SI) se utilizan prefijos que representan múltiplos y submúltiplos de una unidad; estos prefijos SI corresponden siempre a potencias de 10.

Prefijos SI

Cada prefijo del sistema internacional recibe un nombre diferente y utiliza un símbolo para representarlo. Los prefijos que utilizaremos más habitualmente son:

10^{-12}	pico (p)
10^{-9}	nano (n)
10^{-6}	micro (μ)

10^{-3}	mili (m)
10^{3}	kilo (K)
10^{6}	mega (M)
10^{9}	giga (G)
10^{12}	tera (T)

Ejemplos de prefijos SI

10^3 bytes = 1.000 bytes = 1 Kilobyte (KB o Kbyte)

10^6 bytes = 10^3 KB = 1.000 KB = 1 Megabyte (MB o Mbyte)

10^9 bytes = 10^3 MB = 1.000 MB = 1 Gigabyte (GB o Gbyte)

10^{12} bytes = 10^3 GB = 1.000 GB = 1 Terabyte (TB o Tbyte)

Ahora bien, en informática, la capacidad de almacenamiento habitualmente se indica en múltiplos que sean potencias de 2; en este caso se utilizan los prefijos definidos por la International Electrotechnical Commission (IEC).

Prefijos IEC

Los prefijos IEC representan múltiplos para las unidades de información bit y byte. Los nombres se han creado añadiendo el término *binario* a los prefijos SI. Por ejemplo *kibi* sería la contracción de *kilo binario*. Los prefijos que utilizaremos más habitualmente son:

2^{10}	kibi (Ki)
2^{20}	mebi (Mi)
2^{30}	gibi (Gi)
2^{40}	tebi (Ti)

Ejemplos de prefijos IEC

2^{10} bytes = 1.024 bytes = 1 KiB (kibibyte)

2^{20} bytes = 1.024 KiB = 1 MiB (mebibyte)

2^{30} bytes = 1.024 MiB = 1 GiB (gibibyte)

2^{40} bytes = 1.024 GiB = 1 TiB (tebibyte)

La industria utiliza mayoritariamente las unidades SI. Por ejemplo, si nos fijamos en las características de un disco duro que se comercialice con 1 TB de capacidad, realmente la capacidad del disco será de 1.000 GB = 1.000.000 MB = 1.000.000.000 KB = 1.000.000.000.000 bytes.

En cambio, cuando conectamos este disco a un computador y mostramos las propiedades del dispositivo, veremos que en la mayoría de los sistemas operativos se nos mostrará la capacidad en unidades IEC; en este caso, 976.562.500 KiB = 953.674 MiB = 931 GiB = 0,91 TiB.

1.3. Métodos de acceso

Cada tipo de memoria utiliza un método a la hora de acceder a las posiciones de memoria. Hay métodos de acceso diferentes característicos de cada tipo de memoria:

1) Secuencial. Se accede desde la última posición a la que se ha accedido, leyendo en orden todas las posiciones de memoria hasta llegar a la posición deseada. El tiempo de acceso depende de la posición a la que se quiere acceder y de la posición a la que se ha accedido anteriormente.

Usos del acceso secuencial

El acceso secuencial se utiliza básicamente en dispositivos de cinta magnética.

2) Directo. La memoria se organiza en bloques y cada bloque de memoria tiene una dirección única, se accede directamente al principio de un bloque y dentro de este se hace un acceso secuencial hasta llegar a la posición de memoria deseada. El tiempo de acceso depende de la posición a la que se quiere acceder y de la última posición a la que se ha accedido.

Usos del acceso directo

El acceso directo es un método de acceso que se utiliza en discos magnéticos.

3) Aleatorio. La memoria se organiza como un vector, en el que cada elemento individual de memoria tiene una dirección única. Se accede a una posición determinada proporcionando la dirección. El tiempo de acceso es independiente de la posición a la que se ha accedido y es independiente de la última posición a la que se ha accedido.

Las operaciones básicas utilizadas cuando trabajamos con la memoria son:

a) Operación de lectura: en esta operación hay que proporcionar a la memoria la dirección donde se encuentra la información deseada. La acción que hace la memoria consiste en suministrar la información contenida en la dirección indicada.

b) Operación de escritura: en esta operación hay que suministrar a la memoria la información que se debe almacenar y la dirección de memoria donde se la quiere almacenar. La acción que se lleva a cabo consiste en registrar la información en la dirección especificada.

Usos del acceso aleatorio

El acceso aleatorio se suele utilizar en memorias RAM y ROM.

4) Asociativo. Se trata de un tipo de memoria de acceso aleatorio donde el acceso se hace basándose en el contenido y no en la dirección. Se especifica el valor que se quiere localizar y se compara este valor con una parte del contenido de cada posición de memoria; la comparación se lleva a cabo simultáneamente con todas las posiciones de la memoria.

Usos del acceso asociativo

Este método de acceso se suele utilizar en las memorias caché.

1.4. Organización de los datos de una memoria

En este subapartado solo nos referiremos a la manera de organizar los datos en memorias que se encuentren en el chip del procesador y en la memoria interna. La organización de la memoria externa se lleva a cabo de manera diferente.

Básicamente, los elementos que hemos de tener en cuenta son los siguientes:

1) Palabra de memoria. Es la unidad de organización de la memoria desde el punto de vista del procesador; el tamaño de la palabra de memoria se especifica en bytes o bits. Es el número de bytes máximo que se pueden leer o escribir en un solo ciclo de acceso a la memoria.

Ejemplo

Memoria de 2Kbytes con una palabra de memoria de 2 bytes. Por lo tanto, necesitaremos 10 bits para poder hacer referencia a las 1.024 (2^{10}) posiciones de memoria que almacenarán 2 bytes (16 bits) cada una.

Memoria interna

Dirección← (10 bits) →	Palabra← (16 bits) →
0000000000	**00000000 00000000**
0000000001	**00000000 00000000**
0000000010	**00000000 00000000**
0000000011	**00000000 00000000**
0000000100	**00000000 00000000**
...	**...**
↑ (1.024 direcciones) ↓	**contenido almacenado en la memoria**
...	**...**
1111111100	**00000000 00000000**
1111111101	**00000000 00000000**
1111111110	**00000000 00000000**
1111111111	**00000000 00000000**

2) Unidad de direccionamiento. La memoria interna se puede ver como un vector de elementos, una colección de datos contiguos, en la que cada dato es accesible indicando su posición o dirección dentro del vector.

La unidad de direccionamiento especifica cuál es el tamaño de cada elemento de este vector; habitualmente a la memoria se accede como un vector de bytes –cada byte tendrá su dirección–, aunque puede haber sistemas que accedan a la memoria como un vector de palabras, en los que cada dirección corresponda a una palabra.

El número de bits utilizados para especificar una dirección de memoria fija el límite máximo de elementos dirigibles, el tamaño del mapa de memoria; si tenemos n bits para las direcciones de memoria, el número máximo de elementos dirigibles será de 2^n.

3) Unidad de transferencia. En un acceso a memoria se puede acceder a un byte o a varios, con un máximo que vendrá determinado por el número de bytes de una palabra de memoria; es decir, en un solo acceso se leen o escriben uno o varios bytes.

Cuando se especifica la dirección de memoria a la que se quiere acceder, se accede a partir de esta dirección a tantos bytes como indique la operación de lectura o escritura.

En memoria externa, se accede habitualmente a un bloque de datos de tamaño muy superior a una palabra. En discos es habitual transferir bloques del orden de los Kbytes.

Ejemplo

En los procesadores x86 de 32 y 64 bits, la unidad de direccionamiento es de un byte, pero el tamaño de la palabra de memoria es de 4 bytes (32 bits).

Los registros del procesador (accesibles para el programador) habitualmente tienen un tamaño igual al tamaño de la palabra de memoria; por ejemplo, en un procesador de 32 bits (como el Intel 386) el tamaño de los registros era de 32 bits (4 bytes).

Los procesadores x86-64 son procesadores con registros de 64 bits, pero en cambio el tamaño de la palabra de memoria continúa siendo de 32 bits; eso es así para mantener la compatibilidad con procesadores anteriores. No hay que olvidar que la arquitectura x86-64 es una extensión de la arquitectura de 32 bits x86-32.

El tamaño de la palabra de memoria de los procesadores x86 de 32 y 64 bits es de 32 bits (4 bytes) y en un ciclo de memoria se puede acceder a 1, 2 o 4 bytes.

En la arquitectura CISCA el tamaño de palabra es también de 32 bits pero accedemos siempre a una palabra de 4 bytes.

1.4.1. Ordenación de los bytes en memoria

Aunque normalmente la unidad de direccionamiento de la memoria es el byte, es habitual que se puedan hacer accesos a memoria en múltiplos de byte, hasta el tamaño de la palabra (2, 4, e incluso 8 bytes). En este caso, solo se indica la dirección del primer byte de la palabra y se utilizan dos métodos a la hora de acceder a la palabra:

- ***Big-endian***: la dirección especificada corresponde al byte de más peso de la palabra.
- ***Little-endian***: la dirección especificada corresponde al byte de menos peso de la palabra.

Ejemplo

Supongamos una memoria de capacidad reducida (solo 256 bytes) en la que el tamaño de la palabra a la que se puede acceder es de 16 bits (2 bytes). Las direcciones de memoria serán de 8 bits; para acceder a una palabra de memoria se indica solo la dirección del primer byte.

Dirección		Valor
0	00000000	01100110
1	00000001	11100011
...	...	...
14	**00001110**	**00000000**
15	**00001111**	**11111111**
...	...	...
254	11111110	00001111
255	11111111	11001100

Si indicamos la dirección 00001110 (dirección 14), obtenemos los valores de las posiciones de memoria 14 y 15 así:

1) ***Little-endian***: la dirección 14 corresponde al byte de menos peso de la palabra de 16 bits. Obtenemos el valor:

1111111100000000 (direcciones 15 y 14)

2) ***Big-endian***: la dirección 14 corresponde al byte de más peso de la palabra de 16 bits. En este caso obtenemos el valor:

0000000011111111 (direcciones 14 y 15)

1.5. Tiempo de acceso y velocidad

En memorias de acceso aleatorio, memoria RAM, el **tiempo de acceso** (o **latencia**) es el tiempo que transcurre desde que una dirección de memoria es visible para los circuitos de la memoria hasta que el dato está almacenado (escritura) o está disponible para ser utilizado (lectura). En memorias de acceso no aleatorio (discos) es el tiempo necesario para que el mecanismo de lectura o escritura se sitúe en la posición necesaria para empezar la lectura o escritura.

En memorias de acceso aleatorio, el tiempo de un **ciclo de memoria** se considera el tiempo de acceso más el tiempo necesario antes de que se pueda empezar un segundo acceso a la memoria.

Finalmente, la **velocidad de transferencia** es la velocidad a la que se puede leer o escribir un dato de memoria. En las memorias de acceso aleatorio será el inverso del tiempo de ciclo.

Unidades de la velocidad de transferencia

La velocidad de transferencia se mide en bytes por segundo; es habitual indicar la velocidad de una memoria en MiB/segundo o GiB/segundo.

1.6. Coste

Consideramos el coste por unidad de almacenamiento (coste por bit). Podemos observar que existe una relación directamente proporcional entre la velocidad y el coste/bit: a medida que aumenta la velocidad aumenta también el coste/bit. Eso implica que, con un presupuesto fijado, podremos adquirir memorias muy rápidas pero relativamente pequeñas, o memorias más lentas, pero de mucha más capacidad.

Ejemplo de coste

En el año 2010 con 100 € se podía conseguir una memoria RAM de 4GB con un tiempo de acceso de 5ns o un disco magnético de 1 TB (1.000 GB) con un tiempo de acceso de 5 ms (5.000.000 ns), por lo tanto, al mismo coste podemos tenemos una memoria mil veces más grande, pero un millón de veces más lenta.

1.7. Características físicas

La memoria se puede clasificar según características físicas diferentes; básicamente podemos distinguir dos clasificaciones. La primera distingue entre:

- **Memoria volátil**: memoria que necesita una corriente eléctrica para mantener su estado; estas memorias incluyen registros, memoria caché y memoria principal.

- **Memoria no volátil**: mantiene el estado sin necesidad de corriente eléctrica, incluye memorias de solo lectura, memorias programables, memoria flash, dispositivos de almacenamiento magnético y óptico.

La segunda clasificación distingue entre:

- **Memoria de semiconductores**: es una memoria que utiliza elementos semiconductores, transistores, en su construcción; incluye: registros, memoria caché, memoria principal, memorias de solo lectura, memoria flash.

- **Memoria magnética**: utiliza superficies imantadas para guardar la información; dentro de esta categoría se incluyen básicamente discos y cintas magnéticas.

- **Memoria óptica**: utiliza elementos de almacenamiento que pueden ser leídos y escritos mediante luz láser; se incluyen dispositivos de CD, DVD, Blu-ray.

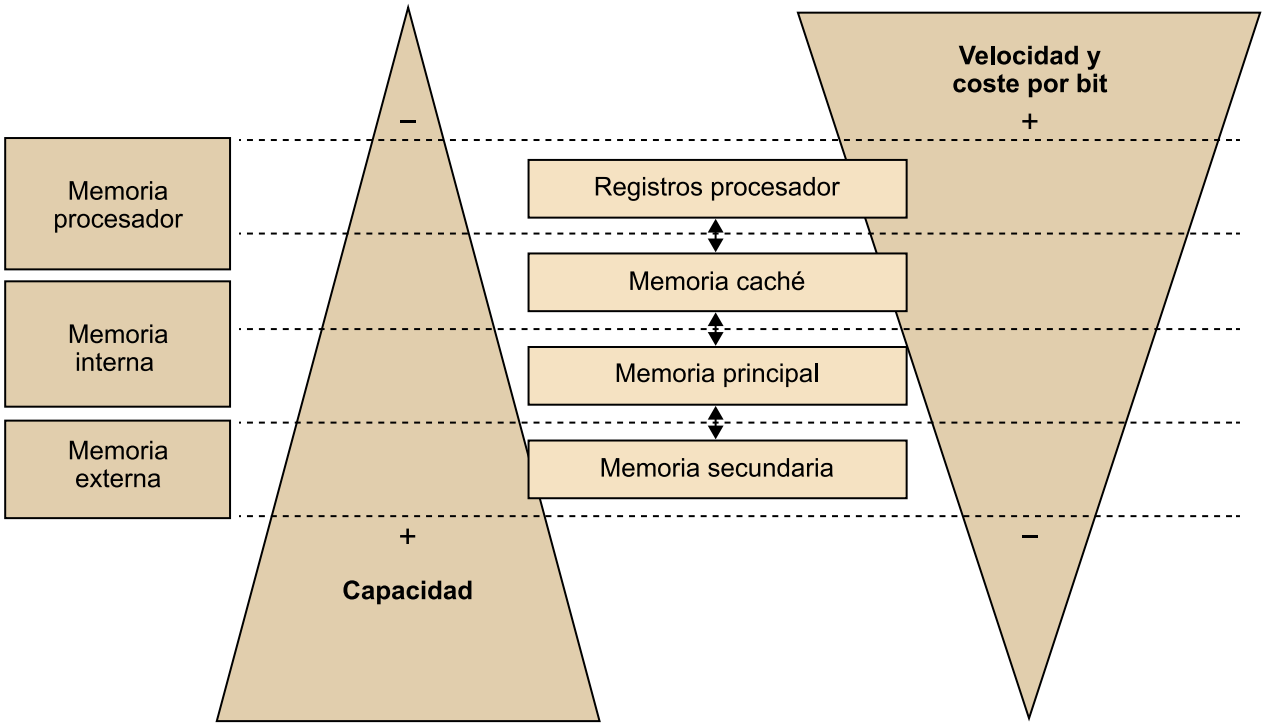
2. Jerarquía de memorias

El objetivo en el diseño del sistema de memoria de un computador es que tenga una gran capacidad y un tiempo de acceso reducido con el precio más bajo posible. Como no hay ninguna tecnología que cumpla simultáneamente estos requisitos, la memoria del computador se estructura en varios niveles con el objetivo de conseguir unas prestaciones mejores, y forma lo que denominamos **jerarquía de memorias**.

En una jerarquía de memorias se utilizan varios tipos de memoria con distintas características de capacidad, velocidad y coste, que dividiremos en niveles diferentes: memoria del procesador, memoria interna y memoria externa.

Cada nivel de la jerarquía se caracteriza también por la distancia a la que se encuentra del procesador. Los niveles más próximos al procesador son los primeros que se utilizan; eso es así porque también son los niveles con una velocidad más elevada.

A continuación se muestra cuál es la variación de la capacidad, velocidad y coste por bit para los niveles típicos de una jerarquía:



El objetivo final de la jerarquía de memorias es conseguir que, cuando el procesador acceda a un dato, este se encuentre en el nivel más rápido de la jerarquía. Obtenemos así una memoria a un coste moderado, con una velocidad próxima a la del nivel más rápido y la capacidad del nivel más alto.

Cada nivel de la jerarquía de la memoria se relaciona solo con los niveles superior e inferior, salvo casos excepcionales. El procesador tiene acceso solamente a los registros y obtiene los datos de memoria mediante la memoria caché.

Por ello, cuando el procesador necesita un dato y este no está disponible en la memoria caché, se tendrá que llevar a ella desde el nivel en el que esté disponible.

Por otra parte, si el procesador modifica un dato en un nivel de la jerarquía de memorias, hay que garantizar que la modificación se efectúe en el resto de los niveles en los que el dato se encuentre almacenado. Si esto no se hiciera así, la siguiente vez que se accediera a este dato, se podría tomar un valor incorrecto. Este problema se denomina *coherencia*.

Como los niveles de memoria más próximos al procesador no son muy grandes, se podría pensar que se pierde mucho tiempo trasladando los datos desde un nivel hasta otro, ya que este movimiento tiene que ser constante. En realidad, eso no es cierto: los datos se reutilizan con mucha frecuencia, por lo que resulta útil que estén en el nivel más próximo al procesador. Más adelante estudiaremos por qué se produce esta reutilización y, por lo tanto, por qué es efectiva la jerarquía de memorias.

A continuación se ofrece una descripción de las características principales de los diferentes niveles de una jerarquía de memoria.

2.1. Registros

El registro es el espacio de memoria que se encuentra dentro del procesador, integrado dentro del mismo chip de este. Se utilizan celdas de memoria de tipo estático, SRAM, para su implementación.

Es el espacio de memoria en el cual el procesador puede acceder más rápidamente a los datos. Este espacio de memoria es accesible al programador de lenguaje de ensamblador y, si se gestiona bien, permite minimizar el número de accesos a la memoria interna, que son bastante más lentos.

2.2. Memoria interna

La memoria interna en un computador moderno está formada típicamente por dos niveles fundamentales: memoria caché y memoria principal. En los computadores actuales es frecuente encontrar la memoria caché también dividida en niveles.

2.2.1. Memoria caché

Las **memorias caché** son memorias de capacidad reducida, pero más rápidas que la memoria principal, que utilizan un método de acceso asociativo. Se pueden encontrar dentro del chip del procesador o cerca de él y están diseñadas para reducir el tiempo de acceso a la memoria. En la memoria caché se almacenan los datos que se prevé que se utilizarán más habitualmente, de manera que sea posible reducir el número de accesos que debe hacer el procesador a la memoria principal (ya que el tiempo de acceso a la memoria principal siempre es superior al tiempo de acceso a la memoria caché).

No es accesible por parte del programador, es gestionada por el hardware y el sistema operativo y se implementa utilizando tecnología SRAM.

Los procesadores modernos utilizan diferentes niveles de memoria caché, lo que se conoce como **memoria caché de primer nivel**, **segundo nivel**, etc. Actualmente es habitual disponer de hasta tres niveles de memoria caché, referidos como L1, L2 y L3. Cada vez es más frecuente que algunos de estos niveles se implementen dentro del chip del procesador y que el nivel más próximo al procesador esté dividido en dos partes: una dedicada a las instrucciones y otra dedicada a los datos.

Memoria caché de los procesadores de Intel y AMD

Los últimos procesadores de Intel y AMD incluyen tres niveles de memoria caché: un primer nivel (L1) dividido en memoria caché de instrucciones y memoria caché de datos, y los otros niveles (L2 y L3), unificados. Los procesadores actuales tienen un diseño multinúcleo (*multicore*); un procesador integra en un solo chip varios núcleos completamente funcionales, cada núcleo dispone de una memoria caché de primer nivel (L1) y de segundo nivel (L2), y la memoria caché de tercer nivel (L3) es compartida por todos los núcleos del procesador. En estos procesadores toda la memoria caché se integra dentro del chip del microprocesador.

2.2.2. Memoria principal

En la memoria principal se almacenan los programas que se deben ejecutar y sus datos, es la memoria visible para el programador mediante su espacio de direcciones.

La memoria principal se implementa utilizando diferentes chips conectados a la placa principal del computador y tiene una capacidad mucho más elevada que la memoria caché (del orden de Gbytes o de Tbytes en supercomputadores).

Utiliza tecnología DRAM (Dynamic RAM), que es más lenta que la SRAM, pero con una capacidad de integración mucho más elevada, hecho que permite obtener más capacidad en menos espacio.

2.3. Memoria externa

La memoria externa corresponde a dispositivos de almacenamiento secundario: discos magnéticos, cintas magnéticas, discos ópticos, dispositivos de memoria flash, etc., y también se pueden considerar sistemas de almacenamiento en red.

Estos dispositivos son gestionados por el sistema de ficheros del sistema operativo mediante el sistema de entrada/salida.

Los dispositivos que forman la memoria externa se conectan al computador con algún tipo de bus (serie o paralelo). Estos dispositivos se pueden encontrar físicamente dentro del computador conectados por buses internos del computador (IDE, SATA, SCSI, etc.) o pueden estar fuera del computador conectados por buses externos (USB, Firewire, eSATA, Infiniband, etc.).

2.4. Memoria virtual

Decimos que un computador utiliza memoria virtual cuando las direcciones de memoria de los programas se refieren a un espacio de memoria superior al espacio de memoria físico, espacio de memoria principal.

La memoria virtual libera al programador de las restricciones de la memoria principal. En estos computadores diferenciamos entre el mapa de direcciones lógicas o virtuales (las direcciones que utilizan los programas) y el mapa de direcciones físicas o reales (las direcciones de la memoria principal). El espacio de memoria virtual utiliza como soporte un dispositivo de almacenamiento externo (habitualmente un disco magnético), mientras que el espacio de memoria físico se corresponde con la memoria principal del computador.

En la actualidad, prácticamente todos los computadores utilizan memoria virtual. La utilización de la memoria virtual implica resolver dos problemas: la traducción de direcciones lógicas a direcciones físicas y la asignación de espacio de memoria físico a los programas que se deben ejecutar. La gestión de la memoria virtual la efectúa el sistema operativo.

Nota

En este módulo nos centraremos en analizar cómo se gestiona el espacio de direcciones físicas del computador; la gestión del espacio de direcciones virtuales corresponde al ámbito de los sistemas operativos y no la analizaremos.

2.5. Funcionamiento de la jerarquía de memorias

Los dos factores básicos que provocan que el esquema de jerarquía de memorias funcione satisfactoriamente en un computador son los siguientes:

- El flujo de datos entre los niveles de la jerarquía de memorias se puede hacer en paralelo con el funcionamiento normal del procesador.
- El principio de **proximidad referencial** de los programas.

El código de los programas se organiza en subrutinas, tiene estructuras iterativas y trabaja con conjuntos de datos agrupados. Esto, unido al hecho de que la ejecución del código es secuencial, lleva a que durante un intervalo de tiempo determinado se utilice solo una pequeña parte de toda la información almacenada: este fenómeno se denomina **proximidad referencial**.

A causa de esta característica, se ha probado empíricamente que aproximadamente el 90% de todas las instrucciones ejecutadas corresponden al 10% del código de un programa.

Distinguimos dos tipos de proximidad referencial:

1) Proximidad temporal. Es cuando, en un intervalo de tiempo determinado, la probabilidad de que un programa acceda de manera repetida a las mismas posiciones de memoria es muy grande.

La proximidad temporal se debe principalmente a las estructuras iterativas; un bucle ejecuta las mismas instrucciones repetidamente, de la misma manera que las llamadas repetitivas a subrutinas.

2) Proximidad espacial. Es cuando, en un intervalo de tiempo determinado, la probabilidad de que un programa acceda a posiciones de memoria próximas es muy grande.

La proximidad espacial se debe principalmente al hecho de que la ejecución de los programas es secuencial –se ejecuta una instrucción detrás de la otra salvo las bifurcaciones– y también a la utilización de estructuras de datos que están almacenados en posiciones de memoria contiguas.

3. Memoria caché

La memoria caché se sitúa entre la memoria principal y el procesador, puede estar formada por uno o varios niveles. En este apartado explicaremos el funcionamiento de la memoria caché considerando un único nivel, pero el funcionamiento es parecido si tiene varios.

La memoria caché tiene un tiempo de acceso inferior al de la memoria principal con el objetivo de reducir el tiempo de acceso medio a los datos, pero también tiene un tamaño mucho más reducido que la memoria principal. Si un dato está en la memoria caché, es posible proporcionarlo al procesador sin acceder a la memoria principal, si no, primero se lleva el dato de la memoria principal a la memoria caché y después se proporciona el dato al procesador.

Si, en la mayoría de los accesos a memoria, el dato está en la memoria caché, el tiempo de acceso medio será próximo al tiempo de acceso a la memoria caché. Eso es factible gracias a la característica de proximidad referencial de los programas.

Para trabajar con memoria caché, la memoria principal se organiza en **bloques** de palabras, de manera que cuando hay que trasladar datos de la memoria principal a la memoria caché se lleva un bloque entero de palabras de memoria, no se trabaja con palabras individuales.

La memoria caché también se organiza en bloques que se denominan **líneas**. Cada línea está formada por un conjunto de palabras (el mismo número de palabras que tenga un bloque de memoria principal), más una etiqueta compuesta por unos cuantos bits. El contenido de la etiqueta permitirá saber qué bloque de la memoria principal se encuentra en cada línea de la memoria caché en un momento dado.

Dirección	Memoria principal	Bloque	Palabra
0		Bloque 0	0
1			1
...			...
$k-1$			$k-1$
k		Bloque 1	0
$k+1$			1
...			...
$2k-1$			$k-1$
...	...	...	...
2^n-k		Bloque $(2^n/k)-1$	0
			1
...			...
2^n-1			$k-1$

Memoria caché				
	Etiqueta del bloque x	Palabras de la línea		
Línea		0	...	$k-1$
0			...	
1			...	
...	...		...	
$m-1$			...	

A la izquierda, memoria principal de 2^n palabras, organizada en $(2^n)/k$ bloques de k palabras. Memoria caché con m líneas, con k palabras por línea.
A la derecha, memoria caché con m líneas, con k palabras por línea, la *etiqueta del bloque x* identifica qué bloque de la memoria principal tenemos almacenado en aquella línea de la memoria caché.

3.1. Aciertos y fallos

Cada vez que el procesador quiere acceder a una palabra de memoria, primero se accede a la memoria caché; si la palabra de memoria se encuentra almacenada en la memoria caché, se proporciona al procesador y diremos que se ha producido un **acierto**. En caso contrario, se lleva el bloque de datos de la memoria principal que contiene la palabra de memoria hacia la memoria caché y, cuando la palabra ya está en la memoria caché, se proporciona al procesador; en este caso diremos que se ha producido un **fallo**.

Cuando hay un fallo, el hardware de la memoria caché debe realizar la secuencia de tareas siguiente:

1) Solicitar a la memoria principal el bloque en el que está el dato que ha producido el fallo.
2) Llevar el bloque de datos solicitado a la memoria caché. Las operaciones realizadas en esta tarea dependerán de las políticas de asignación y algoritmos de reemplazo que veremos más adelante.
3) El procesador obtiene el dato de la memoria caché como si se hubiera producido un acierto.

Un acceso con fallo en la memoria caché puede ser bastante más costoso en tiempo que un acceso con acierto, por lo que es muy importante tener un número reducido de fallos.

3.2. Rendimiento de la memoria caché

A partir del concepto de acierto y fallo se definen los parámetros que utilizaremos para evaluar el rendimiento de una memoria caché: tasa de fallos y tiempo medio de acceso.

La tasa de fallos se define de la manera siguiente:

T_f = Número de fallos / Número de accesos a la memoria

Por otra parte se define la tasa de aciertos así:

T_e = Número de aciertos / Número de accesos a la memoria = $1 - T_f$

Uno de los objetivos del diseño del sistema de memoria es obtener una tasa de fallos tan baja como sea posible. Generalmente se espera que sea inferior al 10%.

Se puede calcular también el tiempo medio de acceso t_m a partir de la tasa de fallos y de la tasa de aciertos, conociendo el tiempo de acceso en caso de aciertos t_e y el tiempo de acceso en caso de fallo t_f, ya que el tiempo de fallo tiene en cuenta el tiempo necesario para llevar todo un bloque de la memoria principal a la memoria caché y el tiempo de acceso al dato.

$$t_m = T_f \text{ x } t_f + T_e \text{ x } t_e = T_f \text{ x } t_f + (1 - T_f) \text{ x } t_e = T_f \text{ x } (t_f - t_e) + t_e$$

Si la tasa de fallos es cero, el tiempo medio de acceso a memoria es igual al tiempo de acceso a la memoria caché.

3.3. Línea de memoria caché

Hemos visto que la memoria caché se organiza en líneas; una línea está formada básicamente por un conjunto de palabras más una etiqueta que identifica qué bloque de la memoria principal ocupa aquella línea de la memoria caché.

La línea de memoria caché es la unidad de transferencia entre la memoria caché y la memoria principal. El tamaño de la línea es uno de los parámetros fundamentales del diseño de la memoria caché. Hay que decidir cuántas palabras se almacenarán en una línea de memoria caché, es decir, cuál es el tamaño de una línea.

Hemos visto que los datos se trasladan de la memoria principal a la memoria caché cuando hay un fallo. Si se produce un fallo, se lleva a la memoria caché el dato que lo ha provocado y el resto de los datos del bloque de memoria donde se encuentra este dato. De esta manera, se espera que los accesos siguientes sean aciertos en la memoria caché.

El tamaño de la línea es de unos cuantos bytes de información (un tamaño habitual está entre los 32 bytes y 128 bytes). Aumentar el tamaño de la línea permite aprovechar la localidad espacial, pero hasta cierto punto. Cuando se produce un fallo, el tiempo necesario para trasladar una línea más grande aumenta; además, disminuye el número de líneas disponibles de la memoria caché (el tamaño de la memoria caché es fijo) y tendremos más competencia para conseguir un bloque, lo que hará que se saquen de la caché líneas que todavía no se han utilizado en su totalidad y se reducirá el efecto de la localidad espacial, y todo ello puede representar un aumento en la tasa de fallos.

3.4. Políticas de asignación

El número de líneas disponibles en la memoria caché es siempre mucho más pequeño que el número de bloques de memoria principal. En consecuencia, la memoria caché, además de la información almacenada, debe mantener alguna información que relacione cada posición de la memoria caché con su dirección en la memoria principal.

Para acceder a un dato se especifica la dirección en la memoria principal; a partir de esta dirección hay que verificar si el dato está en la memoria caché. Esta verificación la haremos a partir del campo *etiqueta* de la línea de la memoria caché que indica qué bloque de memoria principal se encuentra en cada una de las líneas de la memoria caché.

La política de asignación determina dónde podemos colocar un bloque de la memoria principal dentro de la memoria caché y condiciona cómo encontrar un dato dentro de la memoria caché.

Se definen tres políticas de asignación diferentes para almacenar datos dentro de una memoria caché:

1) Política de asignación directa: un bloque de la memoria principal solo puede estar en una única línea de la memoria caché. La memoria caché de asignación directa es la que tiene la tasa de fallos más alta, pero se utiliza mucho porque es la más barata y fácil de gestionar.

2) Política de asignación completamente asociativa: un bloque de la memoria principal puede estar en cualquier línea de la memoria caché. La memoria caché completamente asociativa es la que tiene la tasa de fallos más baja. No obstante, no se suele utilizar porque es la más cara y compleja de gestionar.

3) Política de asignación asociativa por conjuntos: un bloque de la memoria principal puede estar en un subconjunto de las líneas de la memoria caché, pero dentro del subconjunto puede encontrarse en cualquier posición.

La memoria caché asociativa por conjuntos es una combinación de la memoria caché de asignación completamente asociativa y la memoria caché de asignación directa. El número de elementos de cada subconjunto no suele ser muy grande, un número habitual de elementos es entre 4 y 64. Si el número de elementos del subconjunto es *n*, la memoria caché se denomina *n*-asociativa.

Líneas de la memoria caché donde podemos asignar el bloque x según las diferentes políticas de asignación

Dirección	Memoria principal	Bloque
0		Bloque 0
1		
...		
$k-1$		
k		Bloque 1
$k+1$		
...		
$2 \cdot k-1$		
...	...	...
$x \cdot k+0$		Bloque x
$x \cdot k+(k-1)$		
...	...	...
2^n-k		Bloque $(2^n / k)-1$
...		
2^n-1		

Memoria caché de acceso directo		
Línea	**Etiqueta**	**Palabras del bloque**
0		
...		...
Línea asignada al bloque x		
...		...
$m-1$		

Memoria caché de asignación completamente asociativa		
Línea		**Contenido de la caché**
0		
...	...	...
Bloque x asignado a cualquier línea		
...	...	...
$m-1$		

Memoria caché de asignación asociativa por conjuntos		
Línea		**Contenido de la caché**
0		
...	...	...
...	...	...
Conjunto de líneas asignado al bloque x		
		...
...	...	...
...	...	...
$m-1$		

3.4.1. Memoria caché de asignación directa

Para utilizar este tipo de memoria caché, se asigna cada bloque de la memoria principal a una única línea de la memoria caché.

Para relacionar una línea de la memoria caché con un bloque de la memoria principal a partir de la dirección especificada para acceder a una palabra de la memoria principal, hemos de determinar a qué bloque pertenece la dirección.

Se divide la dirección en dos partes: número de bloque, que corresponde a la parte más significativa de la dirección, y número de palabra, que corresponde a la parte menos significativa.

Si tenemos una memoria principal de 2^n palabras y una memoria caché de 2^m líneas de 2^k palabras por línea, la memoria principal se dividirá en bloques de 2^k palabras. Una dirección de memoria estará formada por n bits, utilizará los k bits menos significativos para el número de palabra y los $n - k$ bits restantes para el número de bloque.

Dirección de memoria	
Número de bloque	Número de palabra
$n - 1$ ... k	$k - 1$ ... 0
← ($n - k$ bits) →	← (k bits) →

Cálculo del número de bloque

A partir de una dirección de memoria a se puede calcular el número de bloque b realizando la operación siguiente: $b = a$ div 2^k, donde *div* es la división entera.

El número de palabra p se puede calcular mediante la operación siguiente: $p = a$ mod 2^k, donde *mod* es el residuo de la división entera.

Para determinar a qué línea de la memoria caché podemos asignar cada bloque, hay que dividir el número de bloque en dos partes: una etiqueta, que corresponde a la parte más significativa del número de bloque, y un número de línea, que corresponde a la parte menos significativa.

Si tenemos un número de bloque que utiliza $n - k$ bits, de estos $n - k$ bits utilizaremos m bits para especificar el número de línea y el resto de los bits ($n - k - m$) para especificar la etiqueta.

Dirección de memoria		
Número de bloque		Número de palabra
Etiqueta	Número de línea	
$n - 1$ ... $k + m$	$k + m - 1$ ... k	$k - 1$ ... 0
← ($n - k - m$ bits) →	← (m bits) →	← (k bits) →

Cálculo del número de etiqueta

A partir del número de bloque b se puede calcular la etiqueta e haciendo la operación siguiente: $e = b$ div 2^m, donde *div* es la división entera.

El número de línea l se puede calcular realizando la operación siguiente: $l = b$ mod 2^m, donde *mod* es el residuo de la división entera.

Tenemos 2^{n-k} bloques en la memoria principal y 2^m líneas en la memoria caché ($2^{n-k} > 2^m$), por lo tanto a cada línea de la memoria caché podemos asignar 2^{n-k-m} (= $2^{n-k} / 2^m$) bloques diferentes. Solo uno de estos 2^{n-k-m} puede estar en la memoria caché en cada momento.

El número de línea indicará en cuál de las 2^m líneas de la memoria caché se puede encontrar el bloque de datos al que queremos acceder de la memoria principal. La etiqueta nos permitirá saber si el bloque al que queremos acceder de la memoria principal es el bloque que en este momento está almacenado en aquella línea de la memoria caché.

Cuando se lleva un bloque de la memoria principal a la línea correspondiente de la memoria caché, el número de la etiqueta del bloque se almacena en el campo etiqueta de la línea, así podremos saber cuál de los 2^{n-k-m} bloques está almacenado en esta línea de la caché. El campo etiqueta es el que nos permite identificar de manera única cada uno de los bloques que podemos asignar a una misma línea de la memoria caché.

De manera general, se puede decir que si tenemos una memoria caché de 2^m líneas, los bloques de memoria principal que se pueden encontrar en cada una de las líneas de la memoria caché son los que se muestran en la tabla siguiente.

Número de línea	Bloques asignados
0	0, 2^m, 2x (2^m), ...
1	1, $2^m + 1$, 2x (2^m) + 1, ...
2	2, $2^m + 2$, 2x (2^m) + 2, ...
...	...
$2^m - 1$	$2^m - 1$, $2^m + (2^m - 1)$, 2x $2^m + (2^m - 1)$, ...

Para determinar si un acceso a una dirección de memoria produce un acierto en la memoria caché, hay que hacer lo siguiente: a partir de la dirección de memoria se determina cuál es su número de línea (bits $k + m - 1$ .. k), el cual se utiliza como índice para acceder a la caché y obtener la etiqueta que identifica el bloque almacenado en esta línea y que se compara con el campo etiqueta de la dirección (bits $n - 1$ .. $k + m$); si coinciden, se trata de un acierto, entonces se utiliza el número de palabra (bits $k - 1$ .. 0) para obtener la palabra solicitada y servirla al procesador.

Si la etiqueta de la dirección y la etiqueta de la línea no coinciden, se trata de un fallo y habrá que trasladar todo el bloque de memoria principal a la memoria caché, reemplazando el bloque que tenemos actualmente almacenado.

Ejemplo

Si tenemos una memoria principal de 2^{16} (64 K) palabras y una memoria caché de 2^{10} (1.024) palabras organizada en 2^4 (16) líneas de 2^6 (64) palabras por línea, la memoria principal se dividirá en bloques de 2^6 (64) palabras. Una dirección de memoria tendrá 16 bits, los 6 bits menos significativos para el número de palabra y los 16 – 6 = 10 bits restantes para el número de bloque; en total tendremos 2^{10} (1.024) bloques de 2^6 (64) palabras. Las direcciones se dividirán de la manera siguiente:

Dirección de memoria	
Número de bloque	Número de palabra
15 6	5 0

El número de bloque de 10 bits se divide en etiqueta y número de línea: 4 bits para la línea, ya que hay 2^4 líneas y 10 – 4 = 6 bits para la etiqueta. Así las direcciones de memoria principal se dividirán de la manera siguiente:

Dirección de memoria		
Número de bloque		Número de palabra
Etiqueta	Número de línea	
15 10	9 6	5 0

La asignación de bloques de la memoria principal a la memoria caché sería de la manera siguiente:

Número de línea	Bloques asignados
0	0, 16, 32, 48, 64, ..., 1008
1	1, 17, 33, 49, 65, ..., 1009
2	2, 18, 34, 50, 66, ..., 1010
...	...
15	15, 31, 63, 79, ..., 1023

A cada línea de la memoria caché le podemos asignar 2^6 = 64 bloques diferentes.

El mapa de direcciones de memoria principal quedaría de la manera siguiente:

	Dirección de memoria		
	Número de bloque		Número de palabra
	Etiqueta	Número de línea	
Bloque 0	0	0	0
			...
			63 ($2^6 - 1$)
Bloque 1	0	1	0
			...
			63 ($2^6 - 1$)
...			
Bloque 15	0	15	0
			...
			63 ($2^6 - 1$)
Bloque 16	1	0	0
			...
			63 ($2^6 - 1$)
...			
...			
...			
Bloque 1007	62	15	0
			...
			63 ($2^6 - 1$)
Bloque 1008	63	0	0
			...
			63 ($2^6 - 1$)
...			
Bloque 1023	63	15	0
			...
			63 ($2^6 - 1$)

Se puede observar que todos los bloques que pueden estar en una misma línea de la caché tienen un valor de etiqueta diferente; se podrá utilizar el valor de la etiqueta para saber qué bloque en concreto se encuentra en cada línea de la memoria caché. La tabla anterior muestra que todos los bloques sombreados están asignados a la línea 0 de la memoria caché y podremos saber cuál se encuentra en la memoria caché gracias al número de la etiqueta.

A continuación, se muestra un contenido posible de la memoria caché:

	Número de bloque	
	Etiqueta	Línea
Bloque 16	1	0
Bloque 1	0	1
	...	...
Bloque 1023	63	15

Memoria caché				
Línea	Etiqueta	Palabras de la línea		
		0	...	$2^k - 1$
0	1	M(64)	...	M(127)
1	0	M(1024)	...	M(1087)
...	...	...	...	...
15	63	M(65472)	...	M(65535)

En la línea 0 tenemos el bloque 16 (etiqueta: 1), en la línea 1 tenemos el bloque 1 (etiqueta: 0) y en la línea 15 tenemos el bloque 1023 (etiqueta: 63). M(x) indica que en esta palabra de la línea de la caché se ha almacenado la palabra de memoria con la dirección x.

A continuación, se muestra la descomposición de una de las direcciones del bloque 16 que se encuentra en la memoria caché.

	Dirección de memoria		
	64 = 000001 0000 000000		
	Número de bloque		Número de palabra
	Etiqueta	Número de línea	
Bloque 16	1 = 000001	0000	000000

3.4.2. Memoria caché completamente asociativa

A diferencia de la memoria caché directa, un bloque de memoria principal se puede encontrar en cualquier línea de la memoria caché.

Para relacionar una línea de la memoria caché con un bloque de la memoria principal a partir de la dirección especificada para acceder a una palabra de la memoria principal, hemos de determinar a qué bloque pertenece la dirección. Se divide la dirección en dos partes: número de bloque que corresponde a la parte más significativa de la dirección y número de palabra que corresponde a la parte menos significativa.

Si tenemos una memoria principal de 2^n palabras y una memoria caché de 2^m líneas de 2^k palabras por línea, la memoria principal se dividirá en bloques de 2^k palabras. Una dirección de memoria estará formada por n bits y utilizará los k bits menos significativos para el número de palabra y los $n - k$ bits restantes para el número de bloque.

Dirección de memoria	
Número de bloque	Número de palabra
$n - 1$ k	$k - 1$ 0
← ($n - k$ bits) →	← (k bits) →

Cálculo del número de bloque

A partir de una dirección de memoria a se puede calcular el número de bloque b haciendo la operación siguiente: $b = a \text{ div} 2^k$, donde *div* es la división entera.

El número de palabra p se puede calcular haciendo la operación siguiente: $p = a \bmod 2^k$, donde *mod* es el residuo de la división entera.

Cabe tener presente que a cada línea de la memoria caché le podemos asignar cualquier bloque de la memoria principal y debemos poder saber cuál se encuentra en cada momento en la memoria caché.

Cuando se traslada un bloque de memoria principal a la memoria caché, hay que decidir qué línea reemplazamos con el nuevo bloque de datos; para tomar esta decisión, se pueden utilizar diferentes algoritmos de reemplazo que

explicaremos más adelante. El número de bloque de la dirección de memoria se almacena en el campo etiqueta de la línea; así podremos saber qué bloque está almacenado en cada una de las líneas de la caché.

Para determinar si un acceso a una dirección de memoria produce un acierto en la memoria caché, hay que hacer lo siguiente:

A partir de la dirección de memoria se determina cuál es su número de bloque (bits $n - 1$ .. k) y se compara simultáneamente el número de bloque de esta dirección con el campo etiqueta de todas las líneas de la memoria caché; si se produce una coincidencia significa que hay un acierto, entonces se utiliza el número de palabra (bits $k - 1$ .. 0) para obtener la palabra solicitada y servirla al procesador.

Si el número de bloque de la dirección no coincide con ninguna etiqueta de la memoria caché, se trata de un fallo y habrá que llevar todo el bloque de memoria principal a la memoria caché reemplazando uno de los bloques que tenemos actualmente almacenados.

Ejemplo

Tenemos una memoria principal de 2^{16} (64 K) palabras y una memoria caché de 2^{10} (1.024) palabras organizada en 2^6 (64) líneas de 2^4 (16) palabras por línea.

La memoria principal se dividirá en bloques de 2^4 (16) palabras. Una dirección de memoria tendrá 16 bits, los 4 bits menos significativos para el número de palabra y los 16 – 4 = 12 bits restantes para el número de bloque; en total tendremos 2^{12} (4.096) bloques de 2^4 (16) palabras. Las direcciones se dividirán de la manera siguiente:

Dirección de memoria	
Número de bloque	Número de palabra
15 4	3 0

A continuación se muestra un posible contenido de la memoria caché:

	Número de bloque
	Etiqueta
Bloque 4095	4095
Bloque 1024	1024
	...
Bloque 1	1

Memoria caché				
Línea	Etiqueta	Palabras de la línea		
		0	...	$2^k - 1$
0	4095	M(65520)	...	M(65535)
1	1024	M(16384)	...	M(16400)
...	...	...	...	...
63	1	M(16)	...	M(31)

M(x) indica que en esta palabra de la línea de la caché está almacenada la palabra de memoria con la dirección x.

A continuación se muestra la descomposición de una de las direcciones del bloque 16 que se encuentra en la memoria caché.

	Dirección de memoria	
	16384 = 010000000000 0000	
	Número de bloque	Número de palabra
Bloque 1024	1024 = 010000000000	000000

3.4.3. Memoria caché asociativa por conjuntos

Un bloque de la memoria principal puede encontrarse en un único conjunto de líneas de la memoria caché, pero dentro del conjunto puede encontrarse en cualquier línea.

Para relacionar una línea de la memoria caché con un bloque de la memoria principal a partir de la dirección especificada para acceder a una palabra de la memoria principal, hemos de determinar a qué bloque pertenece la dirección. Se divide la dirección en dos partes: número de bloque que corresponde a la parte más significativa de la dirección y número de palabra que corresponde a la parte menos significativa.

Si tenemos una memoria principal de 2^n palabras y una memoria caché de 2^m líneas de 2^k palabras por línea, la memoria principal se dividirá en bloques de 2^k palabras. Una dirección de memoria estará formada por n bits, utilizará los k bits menos significativos para el número de palabra y los $n - k$ bits restantes para el número de bloque.

Dirección de memoria	
Número de bloque	Número de palabra
$n - 1$ k	$k - 1$ 0
← ($n - k$ bits) →	← (k bits) →

Cálculo del número de bloque

A partir de una dirección de memoria a se puede calcular el número de bloque b haciendo la operación siguiente: $b = a$ div2^k, donde *div* es la división entera.

El número de palabra p se puede calcular mediante la operación siguiente: $p = a \ mod \ 2^k$, donde *mod* es el residuo de la división entera.

En una memoria caché asociativa por conjuntos hay que organizar la memoria caché en conjuntos; se tiene que dividir las 2^m líneas de la memoria caché en 2^c conjuntos de $\omega = 2^{m-c} = (2^m/2^c)$ líneas cada uno, y de esta manera diremos que es una memoria caché ω -asociativa.

Si tenemos tantos conjuntos como líneas ($2^c = 2^m$) y cada conjunto tiene una sola línea ($\omega = 1$), estamos ante el mismo caso que una memoria caché de asignación directa; si tenemos un solo conjunto ($2^c = 1$) de 2^m líneas ($\omega = 2^m$), se trata de una memoria completamente asociativa.

Para determinar a qué conjunto de la memoria caché podemos asignar cada bloque de la memoria principal, hay que dividir el número de bloque en dos partes: una etiqueta que corresponde a la parte más significativa del número de bloque y un número de conjunto correspondiente a la parte menos significativa.

Si tenemos un número de bloque que utiliza $n - k$ bits, de estos $n - k$ bits utilizaremos c bits para especificar el número de conjunto y el resto de los bits ($n - k - c$), para especificar la etiqueta.

Dirección de memoria		
Número de bloque		Número de palabra
Etiqueta	Número de conjunto	
$n - 1$ $k + c$	$k + c - 1$ k	$k - 1$ 0
← ($n - k - c$ bits) →	← (c bits) →	← (k bits) →

Cálculo del número de etiqueta

A partir del número de bloque b se puede calcular la etiqueta e haciendo la operación siguiente: $e = b$ div 2^c, donde *div* es la división entera.

El número de línea l se puede calcular haciendo la operación siguiente: $l = b$ mod 2^c, donde *mod* es el residuo de la división entera.

Tenemos 2^{n-k} bloques de la memoria principal y 2^c conjuntos de la memoria caché ($2^{n-k} > 2^c$), por lo tanto a cada conjunto de la memoria caché podemos asignar 2^{n-k-c} ($= 2^{n-k}/2^c$) bloques diferentes. Como cada conjunto dispone de ω líneas, solo ω bloques de los 2^{n-k-c} pueden encontrarse en un conjunto de la memoria caché en cada momento.

El número de conjunto de la dirección indicará en cuál de los 2^c conjuntos de la memoria caché se puede encontrar el bloque al que queremos acceder de la memoria principal. La etiqueta nos permitirá saber, comparando simultáneamente todas las etiquetas de las líneas que forman el conjunto, si el bloque al que queremos acceder de la memoria principal es uno de los bloques que en este momento están almacenados en una línea de aquel conjunto de la memoria caché.

Cuando se traslada un bloque de memoria principal a la memoria caché, el campo etiqueta del bloque se almacena en el campo etiqueta de la línea seleccionada dentro del conjunto que le corresponda (según el número de conjunto que especifica la dirección), de esta manera podremos saber qué bloque está almacenado en cada una de las líneas de la caché.

De manera general se puede decir que, si tenemos una memoria caché de 2^m líneas, los bloques de memoria principal que se pueden encontrar en cada una de las líneas de la memoria caché son los siguientes:

Líneas	Número de conjunto	Bloques asignados
0, ..., ω – 1	0	0, 2^c, 2x (2^c), 3x (2^c) ...
ω , ..., 2ω – 1	1	1, 2^c + 1, 2x (2^c) + 1, 3x (2^c) + 1 ...
	...	...
(2^c – 1) ω , ..., $2^{c\,\omega}$ – 1	2^c – 1	2^c – 1, 2^c + (2^c – 1), 2x 2^c + (2^c – 1), ..., 3x 2^c + (2^c – 1)

Para determinar si un acceso a una dirección de memoria produce un acierto en la memoria caché, hay que hacer lo siguiente: a partir de la dirección de memoria se determina cuál es su número de conjunto (bits $k + c - 1$ .. k). Este número de conjunto se utiliza como índice para acceder a las etiquetas de las líneas que identifican los bloques que están almacenados en este conjunto y que se comparan simultáneamente con el campo etiqueta de la dirección (bits $n - 1$ .. $k + c$). Si hay una coincidencia significa que se ha producido un acierto y entonces se utiliza el número de palabra (bits $k - 1$ .. 0) para obtener la palabra solicitada y servirla al procesador.

Si la etiqueta de la dirección no coincide con ninguna etiqueta del conjunto, se trata de un fallo y habrá que llevar todo el bloque de memoria principal a una de las líneas de este conjunto de la memoria caché, reemplazando uno de los bloques que tenemos actualmente almacenados. Como un bloque de memoria principal puede ir a cualquier línea del conjunto, hay que decidir qué línea reemplazaremos con el nuevo bloque de datos; para tomar esta decisión se pueden utilizar diferentes algoritmos de reemplazo que explicaremos más adelante.

Ejemplo

Si tenemos una memoria principal de 2^{16} (64 K) palabras y una memoria caché de 2^{10} (1.024) palabras organizada en 2^6 (64) líneas de 2^4 (16) palabras por línea, dividimos las líneas de la caché en 2^4 (16) conjuntos de 2^2 (4) líneas; por lo tanto, tendremos ω = 4 y diremos que es una memoria caché 4-asociativa.

La memoria principal se dividirá en bloques de 2^4 (16) palabras. Una dirección de memoria tendrá 16 bits, los 4 bits menos significativos para el número de palabra y los 16

– 4 = 12 bits restantes para el número de bloque, en total tendremos 2^{12} (4.096) bloques de 2^4 (16) palabras. Las direcciones se dividirán de la manera siguiente:

Dirección de memoria	
Número de bloque	Número de palabra
15 4	3 0

El número de bloque de 12 bits se divide en etiqueta y número de conjunto: 4 bits para el número de conjunto, ya que hay 2^4 conjuntos, y 12 – 4 = 8 bits para la etiqueta. Así las direcciones de memoria principal se dividirán de la manera siguiente:

Dirección de memoria		
Número de bloque		Número de palabra
Etiqueta	Número de conjunto	
15 8	7 4	3 0

La asignación de bloques de la memoria principal a la memoria caché sería de la manera siguiente:

Líneas	Número de conjunto	Bloques asignados
0, 1, 2, 3	0	0, 16, 32, 48, 64, ..., 4080
4, 5, 6, 7	1	1, 17, 33, 49, 65, ..., 4081
...	...	...
60, 61,62, 63	15	15, 31, 47, 63, 79, ..., 4095

El mapa de direcciones de la memoria principal quedaría de la manera siguiente:

	Dirección de memoria		
	Número de bloque		Número de palabra
	Etiqueta	Número de conjunto	
Bloque 0	0	0	0
			...
			15 ($2^4 - 1$)
Bloque 1	0	1	0
			...
			15 ($2^4 - 1$)
...			
Bloque 15	0	15	0
			...
			15 ($2^4 - 1$)
Bloque 16	1	0	0
			...
			15 ($2^4 - 1$)
...			
...			
...			
Bloque 4079	254	15	0
			...
			15 ($2^4 - 1$)
Bloque 4080	255	0	0
			...
			15 ($2^4 - 1$)
...			
Bloque 4095	255	15	0
			...
			15 ($2^4 - 1$)

Se puede observar que todos los bloques que se pueden encontrar en un mismo conjunto de la caché tienen un valor de etiqueta diferente; se puede utilizar el valor de la etiqueta para saber qué bloque en concreto se encuentra en cada línea de la memoria caché. La tabla anterior muestra que los bloques sombreados están asignados todos al conjunto 0 de la memoria caché y podremos saber cuál se encuentra en la memoria caché gracias al número de la etiqueta.

A continuación se muestra un posible contenido de la memoria caché:

	Número de bloque		Conjunto	Memoria caché				
				Línea	Etiqueta	Palabras de la línea		
	Etiqueta	Conjunto				0	...	$2^k - 1$
Bloque 16	1	0	0	0	1	M(256)	...	M(271)
Bloque 0	0	0		1	0	M(0)	...	M(15)
Bloque 4080	255	0		2	255	M(65280)	...	M(65295)
Bloque 1008	63	0		3	63	M(16128)	...	M(16143)
Bloque 1	0	1	1	4	0	M(16)	...	M(31)
	...	...		...	...	...	...	...
	...	...	...	...	...	...	...	...
	...	...	15	...	...	...	...	...
Bloque 4095	255	15		63	255	M(65520)	...	M(65535)

M(x) indica que en esta palabra de la línea de la caché se ha almacenado la palabra de memoria con la dirección x.

A continuación se muestra la descomposición de una de las direcciones del bloque 1008 que se encuentra en la memoria caché.

	Dirección de memoria		
	16128 = 00111111 0000 0000		
	Número de bloque		Número de palabra
	Etiqueta	Número de conjunto	
Bloque 1008	63 = 00111111	0000	0000

3.5. Algoritmos de reemplazo

Cuando se produce un fallo y se tiene que llevar a la memoria caché un bloque de memoria principal determinado, si este bloque de memoria se puede almacenar en más de una línea de la memoria caché, hay que decidir en qué línea de todas las posibles se pone, y sobreescribir los datos que se encuentran en aquella línea. El algoritmo de reemplazo se encarga de esta tarea.

En una memoria caché directa no es necesario ningún algoritmo de reemplazo, ya que un bloque solo puede ocupar una única línea dentro de la memoria caché. En una memoria caché completamente asociativa, solo se aplica el algoritmo de reemplazo para seleccionar una de las líneas de la memoria caché. En una memoria caché asociativa por conjuntos, solo se aplica el algoritmo de reemplazo para seleccionar una línea dentro de un conjunto concreto.

Para que estos algoritmos de reemplazo no penalicen el tiempo medio de acceso a memoria, se deben implementar en hardware y, por lo tanto, no deberían ser muy complejos.

A continuación se describen de manera general los algoritmos de reemplazo más comunes, pero se pueden encontrar otros algoritmos o variantes de estos.

1) FIFO (*first in first out*). Para elegir la línea se utiliza una cola, de manera que la línea que hace más tiempo que está almacenada en la memoria caché será la reemplazada. Este algoritmo puede reducir el rendimiento de la memoria caché porque la línea que se encuentra almacenada en la memoria caché desde hace más tiempo no tiene que ser necesariamente la que se utilice menos.

Se puede implementar fácilmente utilizando técnicas de *buffers* circulares (o *round-robin*): cada vez que se debe sustituir una línea se utiliza la línea del *buffer* siguiente, y cuando se llega a la última, se vuelve a empezar desde el principio.

2) LFU (*least frequently used*). En este algoritmo se elige la línea que hemos utilizado menos veces.

Se puede implementar añadiendo un contador del número de accesos a cada línea de la memoria caché.

3) LRU (*least recently used)*. Este algoritmo elige la línea que hace más tiempo que no se utiliza. Es el algoritmo más eficiente, pero el más difícil de implementar, especialmente si hay que elegir entre muchas líneas. Se utiliza habitualmente en memorias caché asociativas por conjuntos, con conjuntos pequeños de 2 o 4 líneas.

Para memorias cachés 2-asociativas, se puede implementar añadiendo un bit en cada línea; cuando se hace referencia a una de las dos líneas, este bit se pone a 1 y el otro se pone a 0 para indicar cuál de las dos líneas ha sido la última que se ha utilizado.

4) Aleatorio. Los algoritmos anteriores se basan en factores relacionados con la utilización de las líneas de la caché; en cambio, este algoritmo elige la línea que se debe reemplazar al azar. Este algoritmo es muy simple y se ha demostrado que tiene un rendimiento solo ligeramente inferior a los algoritmos que tienen en cuenta factores de utilización de las líneas.

3.6. Comparativa entre diferentes sistemas de memoria caché

Utilizaremos un ejemplo para ver cómo los accesos a memoria de un programa pueden producir aciertos y fallos en la memoria caché y cómo modifican el contenido de la memoria caché.

Supongamos una memoria principal de 2^{10} (1.024) palabras, en la que cada dirección de memoria corresponde a una palabra, y una memoria caché de 2^4 (4) líneas de 2^2 (4) palabras; por lo tanto, la memoria principal también estará organizada en bloques de tamaño de 4 palabras.

Para determinar el número de bloque que corresponde a una dirección de memoria, dividimos la dirección *d* entre el número de palabras de un bloque:

$$b = d \text{ div } 2^k = d \text{ div } 4$$

Este número de bloque se aplica a todas las organizaciones de la memoria caché.

3.6.1. Memoria caché de asignación directa

En una memoria caché directa un bloque de memoria solo se puede encontrar en una única línea de la memoria caché. A un bloque de memoria *b* le corresponderá la etiqueta *e* y lo asignaremos a la línea *l* de la memoria caché, para determinar la etiqueta y la línea, dividimos el número de bloque *b* entre el número de líneas de la memoria caché:

$$e = b \text{ div } 2^m = b \text{ div } 4$$
$$l = b \text{ mod } 2^m = b \text{ mod } 4$$

Por lo tanto, los bloques de memoria principal que se pueden asignar a cada línea de la memoria caché son los siguientes:

l: número de línea	Bloques	Bloque: etiqueta (6 bits) línea (2 bits)
0 (00)	0, 4, 8, 12, ..., 252	0(000000) 0(00), 1(000001) 0(00), 2(000010) 0(00), ..., 63(111111) 0(00)
1 (01)	1, 5, 9, 13, ..., 253	0(000000) 1(01), 1(000001) 1(01), 2(000010) 1(01), ..., 63(111111) 1(01)
2 (10)	2, 6, 10, 14, ..., 254	0(000000) 2(10), 1(000001) 2(10), 2(000010) 2(10), ..., 63(111111) 2(10)
3 (11)	3, 7, 11, 15, ..., 255	0(000000) 3(11), 1(000001) 3(11), 2(000010) 3(11), ..., 63(111111) 3(11)

Mostramos a qué líneas de la memoria caché se asignan los primeros 16 bloques de memoria principal con las direcciones de memoria que contiene el bloque:

l: número de línea	b:e (a_0 ,a_1 ,a_2 ,a_3): bloque asignado : etiqueta (direcciones del bloque)			
0	0:0 (0,1,2,3)	4:1 (16,17,18,19)	8:2 (32,33,34,35)	12:3 (48,49,50,51)
1	1:0 (4,5,6,7)	5:1 (20,21,22,23)	9:2 (36,37,38,39)	13:3 (52,53,54,55)
2	2:0 (8,9,10,11)	6:1 (24,25,26,27)	10:2 (40,41,42,43)	14:3 (56,57,58,59)

l: número de línea	b:e (a_0 ,a_1 ,a_2 ,a_3): bloque asignado : etiqueta (direcciones del bloque)			
3	3:0 (12,13,14,15)	7:1 (28,29,30,31)	11:2 (44,45,46,47)	15:3 (60,61,62,63)

Hay que remarcar que especificamos el número de bloque y el número de etiqueta, pero que el valor que tendremos realmente almacenado en la caché es tan solo la etiqueta asociada al bloque.

Supongamos que durante la ejecución de un programa se accede a las direcciones de memoria siguientes: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La tabla siguiente muestra la evolución del contenido de la memoria caché e indica el número de bloque, la etiqueta del bloque y las direcciones de memoria del bloque que hay en cada una de las 4 líneas de la memoria caché. Inicialmente, la memoria caché está vacía. Cuando se produce un acierto, se indica con una *E* la línea donde se ha producido el acierto. Cada vez que hay un fallo, se indica con una letra *F* qué línea de la memoria caché se reemplazará y se actualiza el contenido llevando el nuevo bloque de memoria principal a esta línea de la memoria caché.

	Estado inicial	1	Fallo	2	4	Fallo	10	Fallo	15	Fallo
Línea 0		F	0:0 (0, 1, 2, 3)	E		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)
Línea 1					F	1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)
Línea 2							F	2:0 (8, 9, 10, 11)		2:0 (8, 9, 10, 11)
Línea 3									F	3:0 (12, 13, 14, 15)

	1	26	Fallo	27	28	Fallo	29	36	Fallo
Línea 0	E		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)
Línea 1			1:0 (4, 5, 6, 7)			1:0 (4, 5, 6, 7)		F	9:2 (36, 37, 38, 39)
Línea 2		F	6:1 (24, 25, 26, 27)	E		6:1 (24, 25, 26, 27)			6:1 (24, 25, 26, 27)
Línea 3			3:0 (12, 13, 14, 15)		F	7:1 (28, 29, 30, 31)	E		7:1 (28, 29, 30, 31)

	37	38	40	Fallo	10	Fallo	11	12	Fallo	13	9
Línea 0				0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)		
Línea 1	E	E		9:2 (36, 37, 38, 39)		9:2 (36, 37, 38, 39)			9:2 (36, 37, 38, 39)		
Línea 2			F	10:2 (40, 41, 42, 43)	F	2:0 (8, 9, 10, 11)	E		2:0 (8, 9, 10, 11)		E
Línea 3				7:1 (28, 29, 30, 31)		7:1 (28, 29, 30, 31)		F	3:0 (12, 13, 14, 15)	E	

	30	Fallo	8	12	Fallo	40	Fallo	17	Fallo	40

Línea 0		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)	F	4:1 (16, 17, 18, 19)	
Línea 1		9:2 (36, 37, 38, 39)			9:2 (36, 37, 38, 39)		9:2 (36, 37, 38, 39)		9:2 (36, 37, 38, 39)	
Línea 2		2:0 (8, 9, 10, 11)	E		2:0 (8, 9, 10, 11)	F	10:2 (40, 41, 42, 43)		10:2 (40, 41, 42, 43)	E
Línea 3	F	7:1 (28,29,30,31)		F	3:0 (12, 13, 14, 15)		3:0 (12, 13, 14, 15)		3:0 (12, 13, 14, 15)	

3.6.2. Memoria caché completamente asociativa

Utilizamos ahora una memoria caché completamente asociativa. En una memoria caché completamente asociativa, un bloque de memoria principal se puede encontrar en cualquier línea de la caché. Las direcciones de memoria se dividen en número de bloque y número de palabra.

Mostramos los primeros 16 bloques de la memoria principal con las direcciones de memoria que contiene el bloque:

b (a_0,a_1 ,a_2 ,a_3): bloque (direcciones del bloque)			
0 (0,1,2,3)	4 (16,17,18,19)	8 (32,33,34,35)	12 (48,49,50,51)
1 (4,5,6,7)	5 (20,21,22,23)	9 (36,37,38,39)	13 (52,53,54,55)
2 (8,9,10,11)	6 (24,25,26,27)	10 (40,41,42,43)	14 (56,57,58,59)
3 (12,13,14,15)	7 (28,29,30,31)	11 (44,45,46,47)	15 (60,61,62,63)

Cabe remarcar que el número de bloque es el número de etiqueta que tendremos almacenado en la memoria caché.

1) FIFO. Utilizamos este algoritmo de reemplazo y la misma secuencia de direcciones de memoria que en el caso anterior: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La tabla siguiente muestra la evolución del contenido de la memoria caché e indica el número de bloque y las direcciones de memoria del bloque que hay en cada una de las 4 líneas de la memoria caché. Inicialmente la memoria caché está vacía. Cuando se produce un acierto, se indica con una E la línea donde se ha producido el acierto. Cada vez que hay un fallo, se indica con una letra F qué línea de la memoria caché se reemplazará y se actualiza el contenido llevando el nuevo bloque de memoria principal a esta línea de la memoria caché.

	Estado inicial	1	Fallo	2	4	Fallo	10	Fallo	15	Fallo
Línea 0		F	0 (0, 1, 2, 3)	E		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)
Línea 1					F	1 (4, 5, 6, 7)		1 (4, 5, 6, 7)		1 (4, 5, 6, 7)
Línea 2							F	2 (8, 9, 10, 11)		2 (8, 9, 10, 11)

Línea 3									F	3 (12, 13, 14, 15)

	26	Fallo	27	28	Fallo	29	36	Fallo	37	38
Línea 0	F	6 (24, 25, 26, 27)	E		6 (24, 25, 26, 27)			6 (24, 25, 26, 27)		
Línea 1		1 (4, 5, 6, 7)		F	7 (28, 29, 30, 31)	E		7 (28, 29, 30, 31)		
Línea 2		2 (8, 9, 10, 11)			2 (8, 9, 10, 11)		F	9 (36, 37, 38, 39)	E	E
Línea 3		3 (12, 13, 14, 15)			3 (12, 13, 14, 15)			3 (12, 13, 14, 15)		

	40	Fallo	10	Fallo	11	12	Fallo	13	9
Línea 0		6 (24, 25, 26, 27)	F	2 (8, 9, 10, 11)	E		2 (8, 9, 10, 11)		E
Línea 1		7 (28, 29, 30, 31)		7 (28, 29, 30, 31)		F	3 (12, 13, 14, 15)	E	
Línea 2		9 (36, 37, 38, 39)		9 (36, 37, 38, 39)			9 (36, 37, 38, 39)		
Línea 3	F	10 (40, 41, 42, 43)		10 (40, 41, 42, 43)			10 (40, 41, 42, 43)		

	30	Fallo	8	12	40	17	Fallo	40	Fallo
Línea 0		2 (8, 9, 10, 11)	E				2 (8, 9, 10, 11)	F	10 (40, 41, 42, 43)
Línea 1		3 (12, 13, 14, 15)		E			3 (12, 13, 14, 15)		3 (12, 13, 14, 15)
Línea 2	F	7 (28, 29, 30, 31)					7 (28, 29, 30, 31)		7 (28, 29, 30, 31)
Línea 3		10 (40, 41, 42, 43)			E	F	4 (16, 17, 18, 19)		4 (16, 17, 18, 19)

2) LRU. Utilizamos ahora el algoritmo de reemplazo LRU y la misma secuencia que en los casos anteriores: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La tabla siguiente muestra la evolución del contenido de la memoria caché e indica el número de bloque y las direcciones de memoria del bloque que hay en cada una de las 4 líneas de la memoria caché. Inicialmente la memoria caché está vacía. Cuando se produce un acierto, se indica con una E la línea donde se ha producido el acierto. Cada vez que hay un fallo, se indica con una letra F qué línea de la memoria caché se reemplazará y se actualiza el contenido llevando el nuevo bloque de memoria principal a esta línea de la memoria caché.

	Estado inicial	1	Fallo	2	4	Fallo	10	Fallo	15	Fallo
Línea 0		F	0 (0, 1, 2, 3)	E		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)
Línea 1					F	1 (4, 5, 6, 7)		1 (4, 5, 6, 7)		1 (4, 5, 6, 7)
Línea 2							F	2 (8, 9, 10, 11)		2 (8, 9, 10, 11)

Línea 3										F	3 (12, 13, 14, 15)

	1	26	Fallo	27	28	Fallo	29	36	Fallo	37	38
Línea 0	E		0 (0, 1, 2, 3)			0 (0, 1, 2, 3)			0 (0, 1, 2, 3)		
Línea 1		F	6 (24, 25, 26, 27)	E		6 (24, 25, 26, 27)			6 (24, 25, 26, 27)		
Línea 2			2 (8, 9, 10, 11)		F	7 (28, 29, 30, 31)	E		7 (28, 29, 30, 31)		
Línea 3			3 (12, 13, 14, 15)			3 (12, 13, 14, 15)		F	9 (36, 37, 38, 39)	E	E

	40	Fallo	10	Fallo	11	12	Fallo	13	9
Línea 0	F	10 (40, 41, 42, 43)		10 (40, 41, 42, 43)			10 (40, 41, 42, 43)		
Línea 1		6 (24, 25, 26, 27)	F	2 (8, 9, 10, 11)	E		2 (8, 9, 10, 11)		E
Línea 2		7 (28, 29, 30, 31)		7 (28, 29, 30, 31)		F	3 (12, 13, 14, 15)	E	
Línea 3		9 (36, 37, 38, 39)		9 (36, 37, 38, 39)			9 (36, 37, 38, 39)		

	30	Fallo	8	12	40	17	Fallo	40
Línea 0		10 (40, 41, 42, 43)			E		10 (40, 41, 42, 43)	E
Línea 1		2 (8, 9, 10, 11)	E				2 (8, 9, 10, 11)	
Línea 2		3 (12, 13, 14, 15)		E			3 (12, 13, 14, 15)	
Línea 3	F	7 (28, 29, 30, 31)				F	4 (16, 17, 18, 19)	

3.6.3. Memoria caché asociativa por conjuntos

En una memoria caché asociativa por conjuntos con dos conjuntos de dos líneas, un bloque de memoria se puede encontrar en un único conjunto y dentro del conjunto en cualquier línea. A un bloque de memoria b le corresponderá la etiqueta e y lo asignaremos al conjunto j de la memoria caché, para determinar la etiqueta y el conjunto, dividimos el número de bloque b entre el número de líneas de cada conjunto:

$$e = b \text{ div } 2^c = \text{b div } 2$$
$$j = b \bmod 2^c = b \bmod 2$$

Por lo tanto, los bloques de memoria principal que se pueden asignar a cada conjunto de la memoria caché son los siguientes:

j: número de conjunto	Línea	Bloques	Bloque: etiqueta (7 bits) conjunto de 1 bit
0	0	0, 2, 4, 6, 8, ..., 254	0:0(0000000) 0, 2:1(0000001) 0, 4:2(0000010) 0, ..., 254:127(1111111) 0
	1		

j: número de conjunto	Línea	Bloques	Bloque: etiqueta (7 bits) conjunto de 1 bit
1	2	1, 3, 5, 7, 9, ..., 255	1:0(0000000) 1, 3:1(0000001) 1, 5:2(0000010) 1, ..., 255:127(1111111) 1
	3		

Mostramos a qué conjuntos de la memoria caché se asignan los primeros 16 bloques de memoria principal con las direcciones de memoria que contiene el bloque:

j: número de conjunto	b:e (a_0 ,a_1 ,a_2 ,a_3): bloque asignado : etiqueta (direcciones del bloque)			
0	0:0 (0,1,2,3) 2:1 (8,9,10,11)	4:2 (16,17,18,19) 6:3 (24,25,26,27)	8:4 (32,33,34,35) 10:5 (40,41,42,43)	12:6 (48,49,50,51) 14:7 (56,57,58,59)
1	1:0 (4,5,6,7) 3:1 (12,13,14,15)	5:2 (20,21,22,23) 7:3 (28,29,30,31)	9:4 (36,37,38,39) 11:5 (44,45,46,47)	13:6 (52,53,54,55) 15:7 (60,61,62,63)

Hay que remarcar que especificamos el número de bloque y el número de etiqueta, pero que el valor que tendremos realmente almacenado en la caché es tan solo la etiqueta asociada al bloque.

1) LRU. Utilizamos el algoritmo de reemplazo LRU y la misma secuencia que en los casos anteriores: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La tabla siguiente muestra la evolución del contenido de la memoria caché e indica el número de bloque, la etiqueta del bloque y las direcciones de memoria del bloque que hay en cada una de las 4 líneas de la memoria caché. Inicialmente la memoria caché está vacía. Cuando se produce un acierto, se indica con una E la línea donde se ha producido el acierto. Cada vez que hay un fallo, se indica con una letra F qué línea de la memoria caché se reemplazará y se actualiza el contenido llevando el nuevo bloque de memoria principal a esta línea de la memoria caché.

	Estado inicial	1	Fallo	2	4	Fallo	10	Fallo	15	Fallo
Línea 0		F	0:0 (0, 1, 2, 3)	E		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)
Línea 1							F	2:1 (8, 9, 10, 11)		2:1 (8, 9, 10, 11)
Línea 2					F	1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)
Línea 3									F	3:1 (12, 13, 14, 15)

	1	26	Fallo	27	28	Fallo	29	36	Fallo
Línea 0	E		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)
Línea 1		F	6:3 (24,25,26,27)	E		6:3 (24,25,26,27)			6:3 (24,25,26,27)
Línea 2			1:0 (4, 5, 6, 7)		F	7:3 (28,29,30,31)	E		7:3 (28,29,30,31)

Línea 3			3:1 (12, 13, 14, 15)			3:1 (12, 13, 14, 15)		F	9:4 (36,37,38,39)

	37	38	40	Fallo	10	Fallo	11	12	Fallo	13	9
Línea 0			F	10:5 (40,41,42,43)		10:5 (40,41,42,43)			10:5 (40,41,42,43)		
Línea 1				6:3 (24,25,26,27)	F	2:1 (8,9,10,11)	E		2:1 (8,9,10,11)		E
Línea 2				7:3 (28,29,30,31)		7:3 (28,29,30,31)		F	3:1 (12,13,14,15)	E	
Línea 3	E	E		9:4 (36,37,38,39)		9:4 (36,37,38,39)			9:4 (36,37,38,39)		

	30	Fallo	8	12	40	17	Fallo	40
Línea 0		10:5 (40,41,42,43)			E		10:5 (40,41,42,43)	E
Línea 1		2:1 (8,9,10,11)	E			F	4:2 (16,17,18,19)	
Línea 2		3:1 (12,13,14,15)		E			3:1 (12,13,14,15)	
Línea 3	F	7:3 (28,29,30,31)					7:3 (28,29,30,31)	

Podemos comparar las tasas de fallos de los casos anteriores y observar que, en esta secuencia de accesos, en el algoritmo LRU es donde se obtienen menos fallos, mientras que la memoria caché de asignación directa es la que obtiene más fallos.

La tasa de fallos en cada caso es:

Asignación directa:	$T_f = 14/25 = 0{,}56$
Completamente asociativa con FIFO:	$T_f = 13/25 = 0{,}52$
Completamente asociativa con LRU:	$T_f = 12/25 = 0{,}48$
Asociativa por conjuntos con LRU:	$T_f = 12/25 = 0{,}48$

3.7. Políticas de escritura

Cuando accedemos a la memoria caché, podemos hacer lecturas o escrituras; hasta ahora hemos visto la problemática de acceder a la memoria caché para leer un dato. Cuando se debe realizar una operación de escritura, aparecen nuevos problemas porque los datos que tenemos en la memoria caché son una copia de los datos que tenemos en la memoria principal y hay que garantizar la coherencia de los datos.

Analizaremos el caso con un único procesador y un único nivel de memoria caché entre el procesador y la memoria principal. Si tenemos más de un procesador con una memoria caché local para cada procesador, la modificación de un dato en una de estas memorias caché invalida el valor del dato en la memoria principal, pero también invalida el valor del dato si se encuentra en otra memoria caché. De manera parecida, si tenemos otros dispositivos que

puedan modificar directamente un dato de la memoria principal, el valor de este dato queda invalidado en las memorias caché donde se pueda encontrar. La gestión de la coherencia en estos sistemas es más compleja y no se analizará aquí.

A continuación comentaremos diferentes políticas para gestionar las escrituras y mantener la coherencia entre los datos de la memoria caché y la memoria principal:

1) Escritura inmediata (*write trough*): cuando se escribe en la memoria caché, también se escribe en la memoria principal transfiriendo todo el bloque que contiene el dato modificado; de esta manera en todo momento la copia que tenemos en la caché es idéntica a la que tenemos en la memoria principal. La política de escritura inmediata es la más fácil de implementar, pero su inconveniente es que produce un gran flujo de información entre la memoria caché y la memoria principal.

2) Escritura aplazada (*write back*): las escrituras se efectúan solo sobre la memoria caché. La memoria principal se actualiza cuando se elimina una línea de la memoria caché que ha sido modificada. Eso implica añadir algunos bits a cada línea de la memoria caché para saber si la línea se ha modificado o no.

Si hay que reemplazar una línea que ha sido modificada, primero es necesario copiar la línea modificada a la memoria principal y a continuación llevar el nuevo bloque, lo que aumenta significativamente el tiempo para acceder al dato.

Fallo en la escritura

Cuando se quiere hacer una escritura de una dirección que no está en la memoria caché, se producirá un fallo; este fallo se puede tratar de diferentes maneras:

a) Escribir directamente en memoria principal y no llevar el dato a la memoria caché. Esta técnica se puede utilizar en la escritura inmediata. Evitamos transferir el bloque a la caché pero no se tiene en cuenta la proximidad referencial, ya que es muy probable que haya nuevos accesos al mismo bloque de datos.

b) Llevar el bloque a la memoria caché y escribir simultáneamente en la caché y en la memoria principal. Esta técnica es la que se utiliza habitualmente en la escritura inmediata.

c) Llevar el bloque a la memoria caché y escribir solo en la caché. Esta técnica se utiliza habitualmente en escritura aplazada.

En máquinas reales se utilizan cada vez más políticas de escritura aplazada, pero el tratamiento de los fallos en caso de escritura es diferente, principalmente porque se consideran diferentes niveles de memoria caché y porque múltiples dispositivos (procesadores, DMA, canales de E/S) pueden acceder a la memoria.

4. Memoria interna

Todos los tipos de memoria interna se implementan utilizando tecnología de semiconductores y tienen el transistor como elemento básico de su construcción.

El elemento básico en toda memoria es la celda. Una celda permite almacenar un bit, un valor 0 o 1 definido por una diferencia de potencial eléctrico. La manera de construir una celda de memoria varía según la tecnología utilizada.

La memoria interna es una memoria de acceso aleatorio; se puede acceder a cualquier palabra de memoria especificando una dirección de memoria.

Una manera de clasificar la memoria interna según la perdurabilidad es la siguiente:

- Memoria volátil
  - SRAM (*static random access memory*)
  - RAM (*dynamic random access memory*)

- Memoria no volátil
  - ROM (*read only memory*)
  - PROM (*programmable read only memory*)
  - EPROM (*erasable programmable read only memory*)
  - EEPROM (*electrically erasable programmable read only memory*)

- Memoria flash

4.1. Memoria volátil

La memoria volátil es la memoria que necesita una corriente eléctrica para mantener su estado, de manera genérica denominada *RAM*.

Las memorias volátiles pueden ser de dos tipos:

1) SRAM. La memoria estática de acceso aleatorio (SRAM) implementa cada celda de memoria utilizando un flip-flop básico para almacenar un bit de información, y mantiene la información mientras el circuito de memoria recibe alimentación eléctrica.

Para implementar cada celda de memoria son necesarios varios transistores, típicamente seis, por lo que la memoria tiene una capacidad de integración limitada y su coste es elevado en relación con otros tipos de memoria RAM, como la DRAM; sin embargo, es el tipo de memoria RAM más rápido.

Usos de la memoria SRAM

La memoria SRAM se utiliza en la construcción de los registros del procesador y en la memoria caché.

2) DRAM. La memoria dinámica implementa cada celda de memoria utilizando la carga de un condensador. A diferencia de los flip-flops, los condensadores con el tiempo pierden la carga almacenada y necesitan un circuito de refresco para mantener la carga y mantener, por lo tanto, el valor de cada bit almacenado. Eso provoca que tenga un tiempo de acceso mayor que la SRAM.

Cada celda de memoria está formada por solo un transistor y un condensador; por lo tanto, las celdas de memoria son mucho más pequeñas que las celdas de memoria SRAM, lo que garantiza una gran escala de integración y al mismo tiempo permite hacer memorias más grandes en menos espacio.

Usos de la memoria DRAM

La memoria DRAM se utiliza en la construcción de la memoria principal del computador.

4.2. Memoria no volátil

La memoria no volátil mantiene el estado sin necesidad de corriente eléctrica.

Las memorias no volátiles pueden ser de diferentes tipos:

1) Memoria de solo lectura o ROM (*read only memory*). Tal como indica su nombre, se trata de memorias de solo lectura que no permiten operaciones de escritura y, por lo tanto, la información que contienen no se puede borrar ni modificar.

Este tipo de memorias se pueden utilizar para almacenar los microprogramas en una unidad de control microprogramada; también se pueden utilizar en dispositivos que necesitan trabajar siempre con la misma información.

La grabación de la información en este tipo de memorias forma parte del proceso de fabricación del chip de memoria. Estos procesos implican la fabricación de un gran volumen de memorias ROM con la misma información; es un proceso costoso y no es rentable para un número reducido de unidades.

2) Memoria programable de solo lectura o PROM (*programmable read only memory*). Cuando hay que fabricar un número reducido de memorias ROM con la misma información grabada, se recurre a otro tipo de memorias ROM: las memorias ROM programables (PROM).

A diferencia de las anteriores, la grabación no forma parte del proceso de fabricación de los chips de memoria, sino que se efectúa posteriormente con un proceso eléctrico utilizando un hardware especializado para la grabación de memorias de este tipo.

Como el proceso de programación no forma parte del proceso de fabricación, el usuario final de este tipo de memorias puede grabar el contenido según sus necesidades.

Cabe destacar que, tal como sucede con las memorias ROM, el proceso de grabación o programación solo se puede realizar una vez.

Este tipo de memoria tiene unas aplicaciones parecidas a las de las memorias ROM.

3) Memoria reprogramable mayoritariamente de lectura. Esta puede ser de tres tipos:

a) EPROM (*erasable programmable read only memory*). Se trata de memorias en las que habitualmente se hacen operaciones de lectura, pero cuyo contenido puede ser borrado y grabado de nuevo.

Hay que destacar que el proceso de borrar es un proceso que borra completamente todo el contenido de la memoria; no se puede borrar solo una parte. Para borrar, se aplica luz ultravioleta sobre el chip de memoria EPROM; para permitir este proceso, el chip dispone de una pequeña ventana sobre la cual se aplica la luz ultravioleta.

La grabación de la memoria se hace mediante un proceso eléctrico utilizando un hardware específico.

Tanto para el proceso de borrar como para el proceso de grabar hay que sacar el chip de memoria de su localización de uso habitual, ya que la realización de estas dos tareas implica la utilización de hardware específico.

b) EEPROM (*electrically erasable programmable read only memory*). Tiene un funcionamiento parecido a la EPROM, permite borrar el contenido y grabar información nueva; sin embargo, a diferencia de las memorias EPROM, todas las operaciones son realizadas eléctricamente.

Para grabar datos no hay que borrarlos previamente; se permite modificar directamente solo uno o varios bytes sin modificar el resto de la información.

Son memorias mayoritariamente de lectura, ya que el proceso de escritura es considerablemente más lento que el proceso de lectura.

c) Memoria flash. La memoria flash es un tipo de memoria parecida a las memorias EEPROM, en las que el borrado es eléctrico, con la ventaja de que el proceso de borrar y grabar es muy rápido. La velocidad de lectura es superior a la velocidad de escritura, pero las dos son del mismo orden de magnitud.

Este tipo de memoria no permite borrar la información byte a byte, sino que se deben borrar bloques de datos enteros; eso lleva a que todo el contenido de la memoria se pueda borrar en pocos segundos, pero también modera el proceso de escritura, ya que para escribir un dato nuevo hay que borrar previamente todo el bloque.

Tiene una capacidad de integración muy elevada y por este motivo también se utiliza para dispositivos de almacenamiento externo.

5. Memoria externa

La memoria externa está formada por dispositivos de almacenamiento secundario (discos magnéticos, CD, DVD, Blu-ray, etc.). Estos dispositivos se pueden encontrar físicamente dentro o fuera del computador.

La memoria externa es de tipo no volátil; por lo tanto, los datos que se quieran mantener durante un tiempo indefinido o de manera permanente se pueden almacenar en dispositivos de memoria externa.

El método de acceso varía según el dispositivo: generalmente los dispositivos basados en disco utilizan un método de acceso directo, mientras que otros dispositivos, como las cintas magnéticas, pueden utilizar acceso secuencial.

Los datos almacenados en la memoria externa son visibles para el programador en forma de bloques de datos, no como datos individuales (bytes), normalmente en forma de registros o ficheros. El acceso a estos dispositivos se lleva a cabo mediante el sistema de E/S del computador y es gestionado por el sistema operativo.

Ved también

La manera de acceder a los dispositivos de almacenamiento secundario se verá con más detalle cuando se analice el sistema de E/S.

5.1. Discos magnéticos

Los discos magnéticos son dispositivos formados por un conjunto de platos con superficies magnéticas y un conjunto de cabezales de lectura y escritura. La información se graba en estas superficies. Un solo dispositivo integra varios platos, que habitualmente utilizan las dos caras para almacenar la información.

Los platos y los cabezales son accionados por motores eléctricos. Los platos hacen un movimiento de rotación continuo y los cabezales se pueden mover de la parte más externa del disco a la parte más interna, lo que permite un acceso directo a cualquier posición del disco.

Son los dispositivos de almacenamiento secundario más importantes en cualquier computador y constituyen la base de cualquier sistema de memoria externa.

Son los dispositivos de memoria externa que proporcionan más capacidad de almacenamiento y los que tienen las prestaciones más elevadas. La capacidad de los discos magnéticos es del orden de Tbytes, el tiempo de acceso medio es de pocos milisegundos y pueden llegar a velocidades de transferencia del orden de un Gbyte por segundo.

5.1.1. RAID

Un sistema RAID[1] consiste en utilizar una colección de discos que trabajan en paralelo con el objetivo de mejorar el rendimiento y la fiabilidad del sistema de almacenamiento.

(1)RAID son las siglas de *redundant array of independent discs*, en español: matriz redundante de discos independientes.

Un conjunto de operaciones de E/S puede ser tratado en paralelo si los datos a los que se ha de acceder en cada operación se encuentran en diferentes discos; también una sola operación de E/S puede ser tratada en paralelo si el bloque de datos al cual hay que acceder se encuentra distribuido entre varios discos.

Un RAID está formado por un conjunto de discos que el sistema operativo ve como un solo disco lógico.

Los datos se pueden distribuir entre los discos físicos según configuraciones diferentes. Se utiliza información redundante para proporcionar capacidad de recuperación en el caso de fallo en algún disco.

La clasificación del tipo de RAID original incluye 7 niveles, del RAID 0 al RAID 6, en los que cada uno necesita un número diferente de discos y utiliza diferentes sistemas de control de la paridad y de detección y corrección de errores.

El control de un sistema RAID se puede llevar a cabo mediante software o hardware, con un controlador específico.

5.2. Cinta magnética

La cinta magnética es un dispositivo que utiliza una tecnología de almacenamiento parecida a la de los discos magnéticos; la diferencia básica es que la superficie magnética en la que se guarda la información se encuentra sobre una cinta de poliéster. Ya que es una cinta, se utiliza un método de acceso secuencial.

Son dispositivos lentos y se utilizan para hacer copias de seguridad de grandes volúmenes de datos o para almacenar datos a los que se accede con poca frecuencia.

5.3. Memoria flash

Las tendencias actuales incluyen dispositivos de almacenamiento construidos a partir de circuitos de memoria flash. El objetivo es sustituir los discos magnéticos ofreciendo características parecidas en cuanto el tiempo de acceso, tasa de transferencia de datos y capacidad de almacenamiento.

Como las memorias flash no tienen partes mecánicas ni superficies magnéticas, son más tolerantes a fallos y más adecuadas para entornos en los que la fiabilidad es muy importante. También hay dispositivos de este tipo para el gran público, pero que actualmente no rivalizan con los discos magnéticos, especialmente con respecto a la capacidad.

Usos de la memoria flash

La memoria flash se utiliza habitualmente en diferentes tipos de dispositivos de almacenamiento externo: tarjetas de memoria (Compact Flash, Secure Digital, etc.), memorias USB (*pendrive*) y unidades de estado sólido (SSD).

5.4. Disco óptico

Los discos ópticos son unidades de almacenamiento que utilizan luz láser para realizar operaciones de lectura y escritura sobre un soporte extraíble. Estas unidades pueden ser internas (conectadas a un bus interno del computador) o externas (conectadas por un bus externo).

Básicamente, se distinguen tres tipos de soportes: CD, DVD y Blu-ray (BD). Su capacidad máxima varía según el tipo de soporte; es del orden de los centenares de Mbytes en el caso del CD, del orden Gbytes en el caso de los DVD y de decenas de Gbytes en el caso de los Blu-ray.

El tipo de operaciones que se pueden realizar sobre el disco depende de su tipo: hay discos de solo lectura (CD-ROM, DVD-ROM, BD-ROM), discos que se pueden escribir solo una vez (CD-R, DVD+R, DVD-R, BD-R) y discos que se pueden escribir varias veces (CD-RW, DVD+RW, DVD-RW, BD-RE).

Habitualmente, una misma unidad es capaz de trabajar con soportes de diferentes tipos. Por ejemplo, una grabadora de DVD es capaz de leer CD-ROM, DVD-ROM y de leer y escribir CD-R, DVD+R, DVD-R, CD-RW, DVD+RW y DVD-RW.

La velocidad es inferior a la de los discos magnéticos. Están diseñados básicamente para hacer operaciones de lectura, ya que escribir implica un proceso de grabación relativamente lento, del orden de minutos, dependiendo de la cantidad de datos que se quiere almacenar.

5.5. Red

Utilizando los recursos de redes LAN o Internet, se puede disponer de almacenamiento de gran capacidad. Existen tipos diferentes, como SMB, NFS o SAN. Se puede acceder a cualquier dato almacenado en un ordenador conectado a la Red, sin límite de capacidad, y para ampliar la capacidad solo hay que añadir ordenadores nuevos a la red con capacidad de almacenamiento.

Es habitual medir la velocidad de transferencia en bits por segundo y está limitada por el ancho de la banda de la red.

Es adecuado utilizar este tipo de sistema cuando queremos gestionar grandes volúmenes de datos, los soportes físicos que se utilizan para almacenar los datos son discos magnéticos; si el objetivo es simplemente hacer copias de seguridad, se suelen utilizar cintas magnéticas.

Resumen

En este módulo, primero se ha ofrecido una introducción al sistema de memoria de un computador y se han explicado las características principales de los diferentes tipos de memorias:

- localización,
- capacidad,
- métodos de acceso,
- organización de los datos de una memoria,
- tiempo de acceso y velocidad,
- coste y
- características físicas.

Se ha introducido el concepto de jerarquía de memorias con el objetivo de conseguir que el procesador, cuando accede a un dato, este se encuentre en el nivel más rápido y así conseguir tener una memoria a un coste moderado, con una velocidad próxima al nivel más rápido y la capacidad del nivel mayor.

La jerarquía de memorias de un computador se organiza en niveles diferentes:

- Registros.
- Memoria interna.
  - Memoria caché.
  - Memoria principal.

- Memoria externa.

Hemos visto el concepto de la proximidad referencial, gracias al cual la jerarquía de memorias funciona, y hemos distinguido dos tipos:

- proximidad temporal y
- proximidad espacial.

Se han explicado detalladamente las características y el funcionamiento de la memoria caché:

- La organización de la memoria caché.
- El concepto de acierto y fallo y los índices para medir el rendimiento.
- Las diferentes políticas de asignación que hay a la hora de llevar un bloque de memoria principal a la memoria caché:
  - asignación directa,
  - asignación completamente asociativa y

  - asignación asociativa por conjuntos.

- Los diferentes algoritmos de reemplazo que se pueden utilizar cuando hay que sustituir el contenido de una línea de la memoria caché:
  - FIFO,
  - LFU,
  - LRU y
  - aleatorio.

- La comparativa entre los diferentes sistemas de memoria caché.
- Cómo gestionar las escrituras en una memoria caché.

A continuación se han explicado los diferentes tipos de memoria interna y cómo se pueden clasificar según su perdurabilidad en memorias volátiles y no volátiles, y los diferentes tipos que podemos encontrar dentro de cada categoría.

Finalmente, en el último apartado, se han descrito los dispositivos más habituales que conforman la memoria externa de un computador y que son gestionados por el sistema operativo mediante el sistema de E/S.