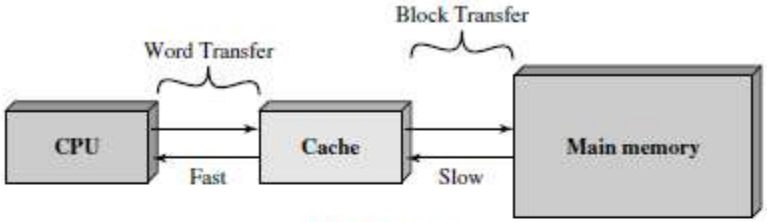
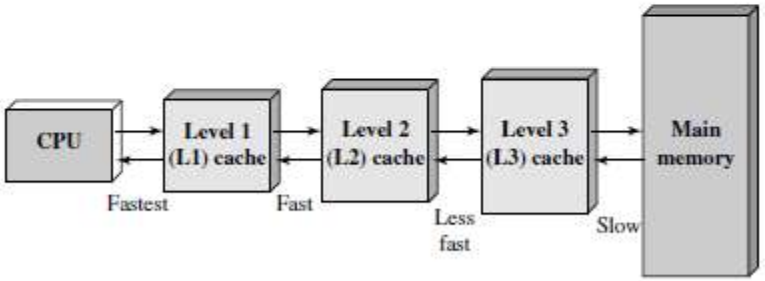


(a) Single cache



(b) Three-level cache organization

Cache and Main Memory

Memoria Caché

Algorítmo de Mapeo Directo

Antecedentes

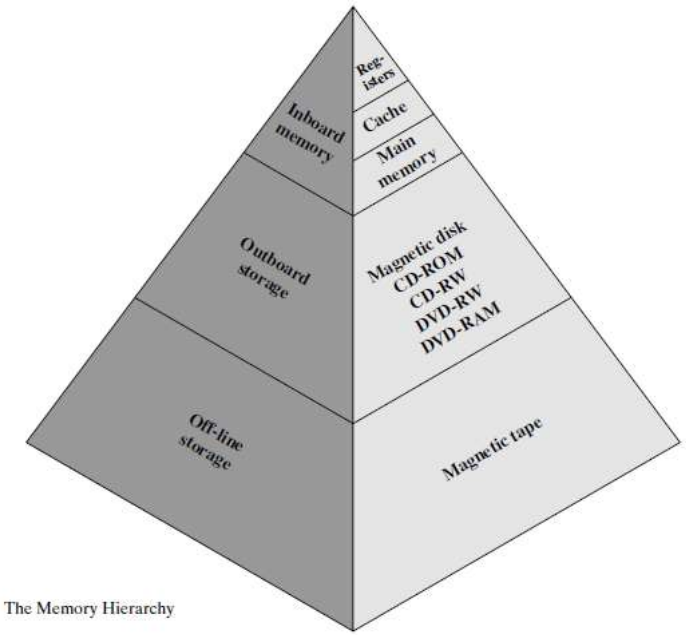
Uno de los principales problemas con la arquitectura de computadoras Von Neumann es que la memoria principal generalmente es considerablemente más lenta que el procesador; además, el proceso de transferencia de información entre el procesador y la memoria es complejo y añade retardos considerables por tiempos de espera en el uso de buses, decodificaciones de direcciones, y transmisión de información binaria entre chips separados. A este problema se le conoce como **Cuello de Botella Von Neumann**.

La memoria de una computadora es la parte que exhibe mayor variedad de tipo, tecnología, organización, desempeño y costo que cualquier otro componente del sistema, por lo que en el diseño de los sistemas de memoria a utilizar se deben considerar varios factores: capacidad de almacenamiento, tiempos de acceso, costo, etc. Para visualizar todos estos factores, se suele organizar los distintos tipos de memoria en una jerarquía de memoria, en la que la memoria más cercana al procesador tiene la más alta jerarquía, y la memoria más lejana al procesador tiene la jerarquía más baja.

También, conforme se desciende en la jerarquía, se presentan las siguientes condiciones:

a) Decrece el costo de almacenamiento por bit.
b) Incrementa la capacidad de almacenamiento.
c) Incrementa el tiempo de acceso (el acceso se torna más lento).
d) Decrece la frecuencia de acceso a la memoria por parte del procesador (el volumen de transferencia de bits es mayor, por lo que se requiere menor frecuencia de acceso).

Entonces, las memorias más pequeñas, caras y rápidas son suplementadas por memorias más grandes, baratas y lentas. La clave para el éxito de esta organización jerárquica es el punto **d): disminuir la frecuencia de acceso**.



The Memory Hierarchy

Principios de Funcionamiento de una Memoria Caché

Supóngase que el procesador tiene acceso a dos niveles de memoria:

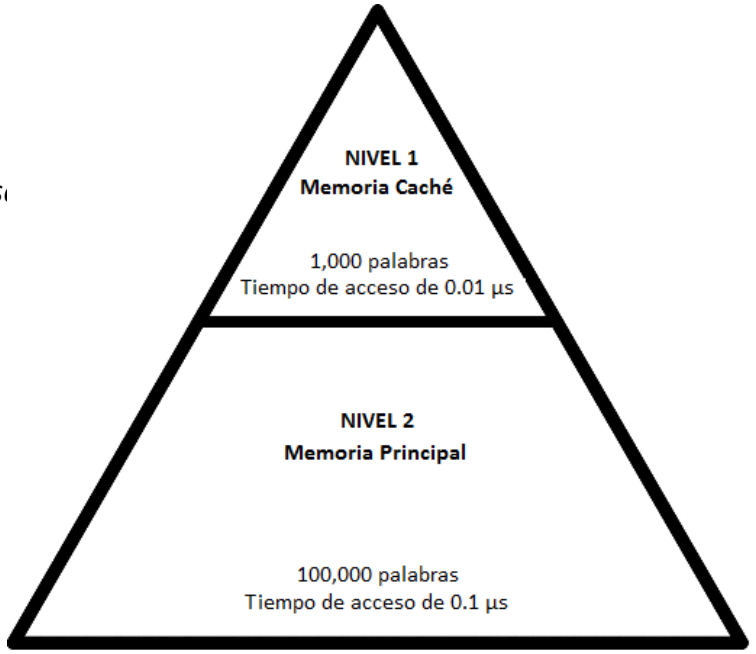
- El nivel 1 contiene 1,000 palabras y tiene un tiempo de acceso de 0.01 µs.
- el nivel 2 contiene 100,000 palabras y un tiempo de acceso de 0.1 µs.
- Asúmase que si una palabra que se requiere se localiza en el **nivel 1** (**Memoria Caché**), entonces el procesador tiene acceso a ella directamente. Si se encuentra en el **nivel 2** (**Memoria Principal**), entonces la palabra es primeramente transferida al nivel 1 y después el procesador tiene acceso a ella. Por simplicidad, se ignorará el tiempo que el procesador requiere para determinar si una palabra está en el nivel 1 o el nivel 2.

Considérense los siguientes términos:

H = ***Hit Ratio***: *Fracción (porcentaje) de todos los acces a memoria en que los datos requeridos se encuentran en la memoria más rápida (memoria caché).*

T_1 = *Tiempo de acceso correspondiente al nivel 1.*

T_2 = *Tiempo de acceso correspondiente al nivel 2.*

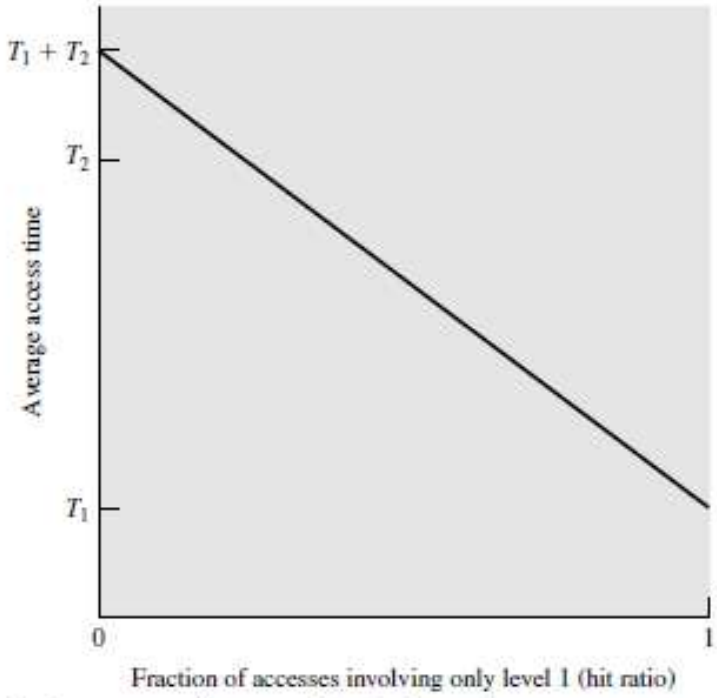


La figura muestra el tiempo de acceso promedio a una memoria de dos niveles como función de la razón *H*, el porcentaje de éxito de encontrar la palabra deseada en el nivel 1. Como se puede apreciar, para niveles altos de accesos a memoria nivel 1, el tiempo total de acceso promedio es mucho más cercano a el nivel 1 que al nivel 2.

Usando el mismo ejemplo, supóngase que el 95% de los accesos a memoria se hacen en la memoria caché. Entonces el tiempo promedio de acceso a una palabra se puede expresar como:

$$(0.95)(0.01\ \mu s) + (0.05)(0.01\ \mu s + 0.1\ \mu s) = 0.0095 + 0.0055 = 0.015\ \mu s$$

- En este ejemplo, el tiempo de acceso promedio es mucho más cercano a 0.01 µs que a 0.1 µs, como se desea.
- El uso de los dos niveles de memoria para reducir el tiempo de acceso promedio funciona en principio, pero solo si las condiciones a) a la d) aplican.
- Mediante el empleo de una variedad de tecnologías, se puede encontrar un amplio espectro de sistemas de memoria que satisfacen las condiciones a) a c). Afortunadamente, la condición d) es válida generalmente.

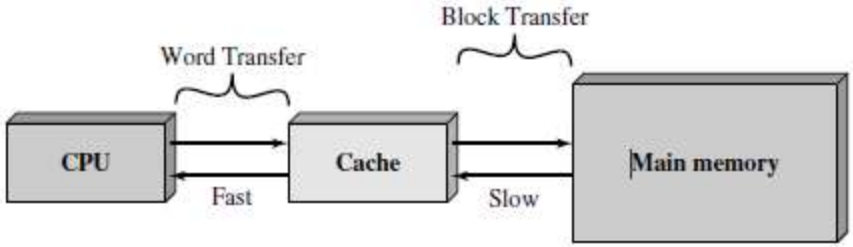


Performance of accesses involving only level 1 (hit ratio)

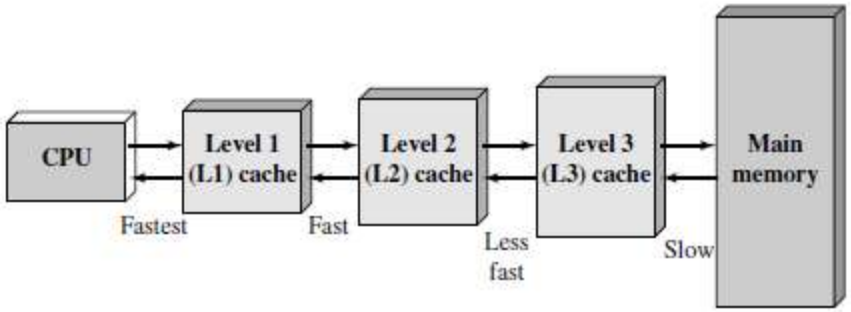
Memoria Caché

La ***memoria caché*** tiene como propósito proveer una memoria de alta velocidad, lo más próxima posible a la velocidad más alta disponible, y al mismo tiempo proveer una memoria con gran capacidad al precio más económico de las memorias tipo semiconductor.

Este concepto se aprecia a continuación, en la que se tiene una memoria principal relativamente grande y lenta, y una memoria caché más pequeña pero más rápida.



(a) Single cache



(b) Three-level cache organization

Cache and Main Memory

La memoria caché contiene una copia de algunas porciones de la memoria principal.

- Cuando el procesador intenta leer o escribir una palabra (instrucción o data) en memoria, se realiza una revisión para determinar si la palabra está en la memoria caché.

- Si la palabra está en la memoria caché (**caché hit**), la palabra es entregada al procesador en caso de lectura o, en caso de escritura, se escribe la información enviada por el procesador en el espacio que le corresponde a la palabra con la dirección indicada.

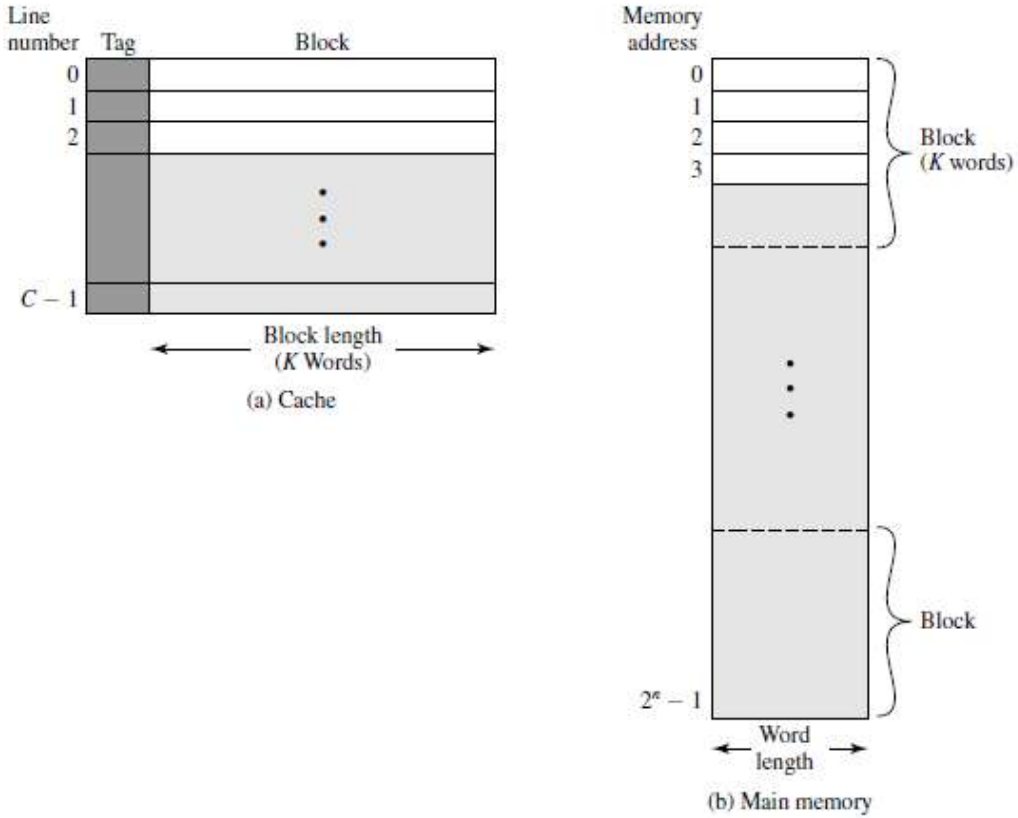
- Si la palabra no se encuentra en la caché (**caché miss**), un **bloque de la memoria principal**, consistente en un número fijo de palabras, es leído y copiado en la memoria caché, y entonces la palabra correspondiente es entregada al procesador en caso de tratarse de una operación de lectura o la información enviada por el procesador se escribe en el espacio que le corresponde a la palabra con la dirección indicada.

- Como consecuencia del ***fenómeno de localidad de referencia***, cuando un bloque de datos es extraído y copiado en la caché para satisfacer una referencia de memoria simple, es muy probable que se presentarán futuras referencias a memoria a la misma localidad o a otras palabras contenidas en el mismo bloque que fue copiado en la caché.

Estructura de la Memoria Principal y la Memoria Caché

La figura muestra la estructura de un sistema de memoria principal/memoria caché.

- La memoria principal consiste en hasta 2^n palabras direccionables (podrían ser palabras de un byte, como es el caso de las PCs), con cada palabra direccionada mediante una dirección única de ***n*** bits.



Cache/Main Memory Structure

- Para propósitos de ***mapeo***, se considera que esta memoria consiste en un número de **bloques** con longitud fija de ***P* palabras** cada uno. Esto es, hay $B = 2^n/P$ bloques en la memoria principal.

- La memoria caché consiste en ***m* líneas** de *P* palabras cada una (puede almacenar un solo bloque), y el número de líneas es considerablemente menor a el número de bloques en la memoria principal ($m << B$).

- En un momento dado, la copia de algún subconjunto de los bloques de memoria reside en las líneas de la memoria caché. Si se requiere leer o escribir una palabra en un bloque de memoria, ese bloque es transferido a una de las líneas de la caché en caso de no encontrarse ya en la caché.

- Dado que existen más bloques que líneas, una línea individual no puede estar dedicada de manera única y permanente a un bloque en particular. Entonces, cada línea incluye una ***etiqueta*** ***(tag)*** que identifica qué bloque en particular está almacenado en ese momento en la línea. La etiqueta es usualmente una porción de la dirección de memoria, como se verá posteriormente.

Considere un sistema en el que la memoria principal tiene una tamaño de M bytes y consiste en un total de N palabras o localidades de T_p bytes cada una, esto es, $M = N * T_p$, entonces:

- Las direcciones de memoria principal consisten en n bits:

$$n = log_2 N = log_2 M / T_p$$

- Si la memoria principal es organizada en un total de B bloques, cada uno con un tamaño fijo de P palabras, entonces:

  el número de bits para especificar el número de palabra dentro del bloque es:

$$p = log_2 P$$

  el tamaño de cada bloque en bytes (o, cuando aplique, bytes por bloque) es:

$$T_b = P * T_p$$

  el número total de bloques en la memoria principal es:

$$B = M / T_b = N / P$$

  y el número de bits para especificar el número de bloques es:

$$b = log_2 B$$

- Si la memoria caché tiene un tamaño de ***C*** bytes, entonces:

  el número total de líneas en la caché es:

$$m = C / Tb$$

  y se requieren ***l*** bits para representar el número de línea:

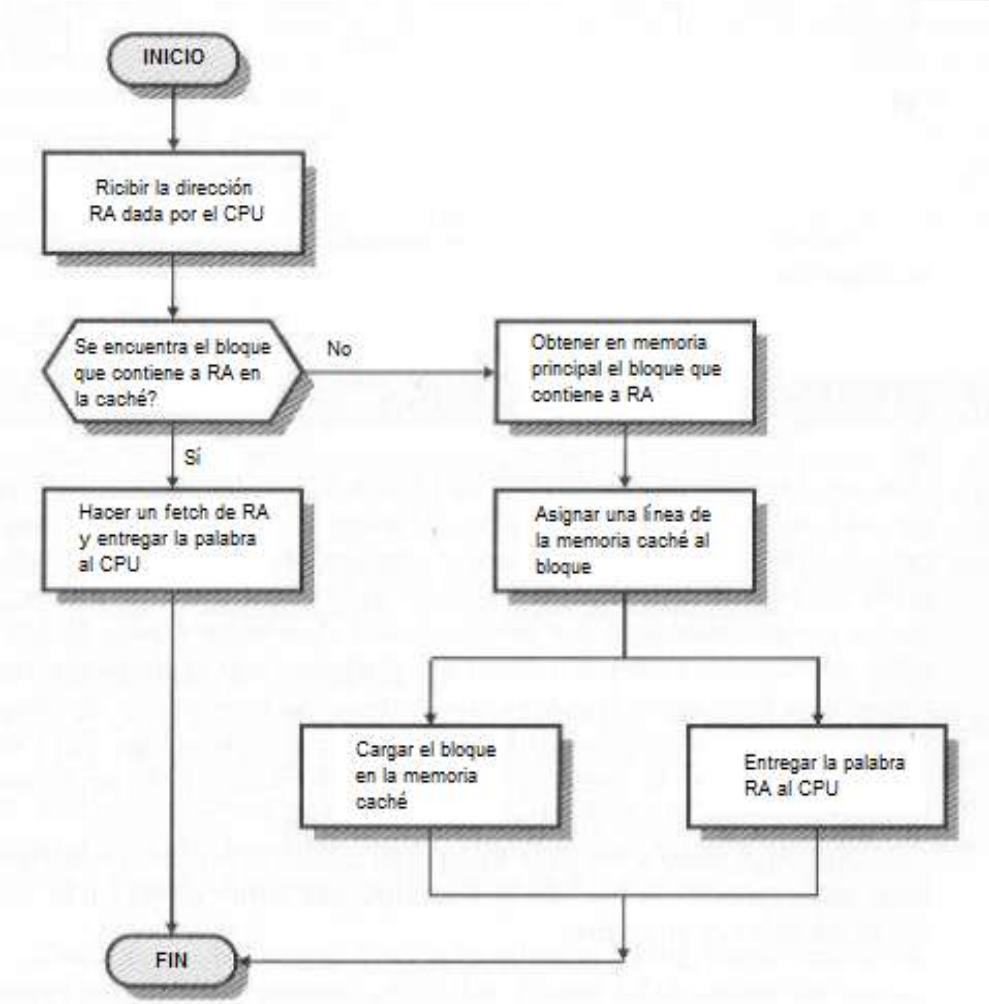
$$l = log_2 m$$

- El número de bits, ***e***, requerido para especificar la etiqueta de un bloque es:

$$e = b - l$$

Política de lectura

- El procesador genera la dirección *RA* que se desea leer en la memoria principal.
- Si la palabra deseada (contenida en la localidad de memoria con dirección *RA*) se encuentra en la memoria caché, esta es entregada al procesador.
- Opuestamente, el bloque que contiene la palabra es cargado en la memoria caché, y la palabra es entregada al procesador. En la figura se puede apreciar que estas dos operaciones ocurren en paralelo.



Política de escritura

Antes de que un bloque que reside en caché pueda ser sustituido en la misma, es necesario considerar si éste ha sido alterado (cambiado) en la caché pero no en la memoria principal.

- Si no ha sido alterado, entonces el viejo bloque en caché puede ser sobrepuesto o sustituido por otro bloque.

- Si ha sido alterado, entonces esto significa que por lo menos una operación de escritura se efectuó en una palabra en esa línea de la caché, y la memoria principal debe ser actualizada correspondientemente para reflejar dichos cambios.

Existen una variedad de políticas de escritura, con sus ventajas y desventajas en desempeño y economía. Hay dos problemas a considerar:

1. Más de un dispositivo puede tener acceso a la memoria principal. Por ejemplo, un módulo de E/S (I/O) podría tener la capacidad de leer/escribir directamente en la memoria. Si una palabra ha sido alterada solo en la caché, entonces la palabra correspondiente en memoria principal es inválida. Más aún, si el dispositivo I/O ha alterado la memoria principal, entonces la palabra correspondiente en caché es inválida.

1. Un problema más complejo se presenta cuando múltiples procesadores están conectados al mismo bus y cada procesador tiene su propia caché. Entonces, si una palabra es alterada en una de las cachés, puede invalidar la palabra correspondiente en otras cachés.

Función de Mapeo y Mapeo Directo

Dado que hay menos líneas de caché que bloques de memoria principal, se requiere de un algoritmo que implemente una ***función de mapeo*** de bloques de memoria principal a líneas de memoria caché. Además, se requiere de un mecanismo para determinar qué bloque de memoria principal se encuentra en una línea de memoria caché.

La selección de la función de mapeo dicta la forma en que la caché es organizada. Existen tres técnicas que pueden ser usadas: ***mapeo directo*** (***direct mapping***), ***mapeo asociativo*** (***associative mapping***) y ***mapeo asociativo de conjunto*** (***set associative mapping***).

Nota: En esta clase examinaremos únicamente la *función de mapeo directo*, dejando al alumno es estudio de los otros dos algoritmos.

Mapeo Directo

El ***mapeo directo*** es la técnica más simple, mapea cada bloque de memoria principal en solo una línea de caché posible. El mapeo se expresa como:

$$i = j \text{ módulo } M$$

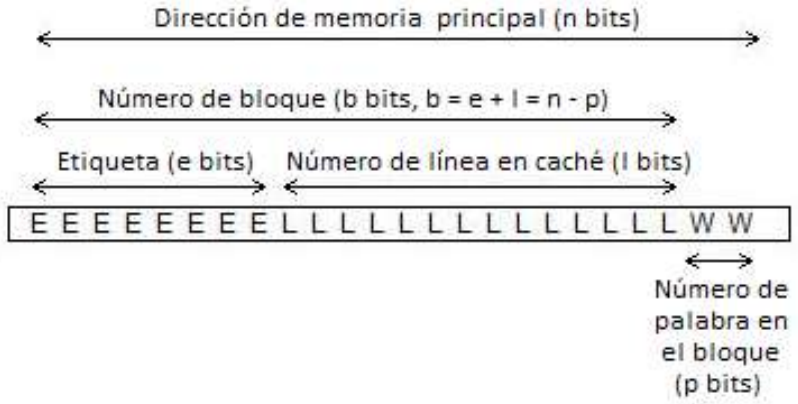
Donde

i = Número de línea de memoria caché
j = Número de bloque de memoria principal
M = Número de líneas en la memoria caché

El efecto del mapeo directo es que los bloques de memoria principal son asignados a líneas de memoria caché como sigue:

Línea de memoria caché	Bloques de memoria principal asignado
0	0, m, 2m, …, 2^s - m
1	1, m + 1, 2m + 1, …, 2^s – m + 1
. . .	. . .
m - 1	m – 1, 2m – 1, 3m – 1, …, 2^s - 1

Entonces, el uso de una porción de la dirección como número de línea provee un mapeo único de cada bloque de memoria principal en una línea de memoria caché. Cuando un bloque es efectivamente leído y copiado en la línea asignada, es necesario etiquetar los datos para distinguirlo de otros bloques que puedan corresponder a esa misma línea. Los bits más significativos que sirven para este propósito representan la **etiqueta** o **tag**.



Ejemplo de Mapeo Directo:

Considere el siguiente sistema:

La memoria principal consiste en 16 MB, con localidades o palabras de memoria de un byte, es decir, la memoria es direccionable por byte (una dirección única por cada localidad de un byte). Los bloques de memoria principal son de 4 palabras = 4 bytes. La caché tiene una capacidad de 64 KB.

- Las direcciones de memoria principal son de

  $n = \log_2 N = \log_2 16M = \log_2 2^{24} = 24$ *bits*

- Número total de bloques en memoria principal

  $B = M / Tb = 2^{24}$ Bytes / (2^2 Bytes /bloque) = 4 M bloques = 2^{22} Bloques

- Los bits requeridos para representar el número de bloque son:

  $b = \log_2 B = \log_2 4M = \log_2 2^{22} = 22$ *bits*

- Número total de líneas en memoria caché:

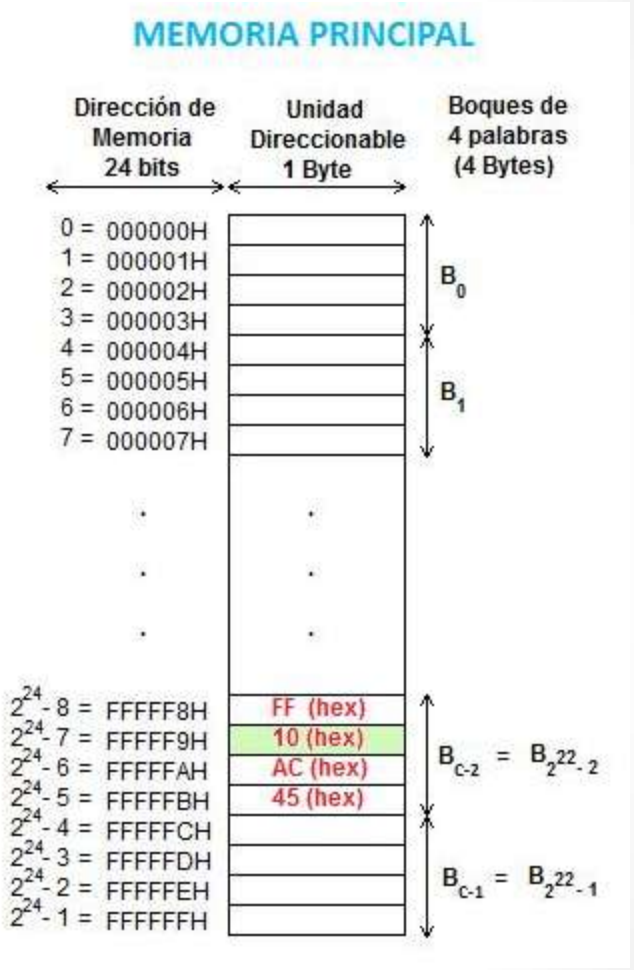
  $m = C / Tb$ = 64 KB / 4 B = 2^{16} Bytes / 2^2 Bytes = 2^{14} líneas = 16 K líneas

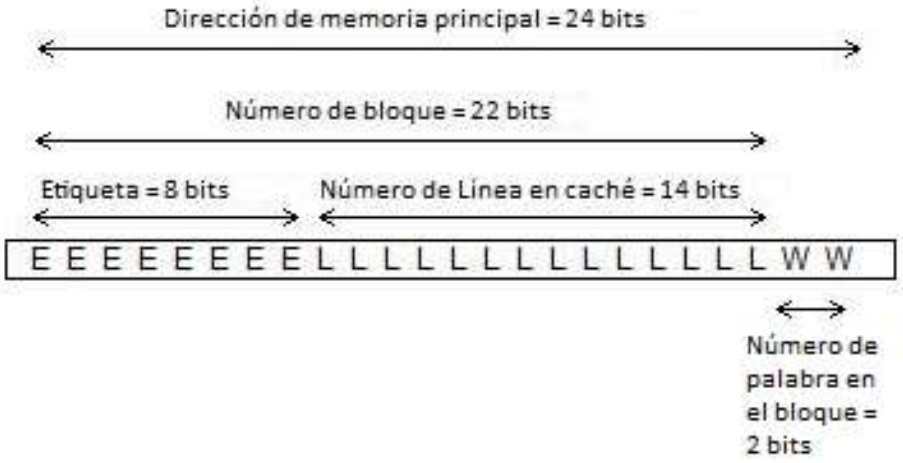
- Los bits requeridos para representar el número de línea son:

  $l = \log_2 m = \log_2 2^{14} = 14$ *bits*

- Los bits requeridos para representar la etiqueta del bloue son:

  $e = b - l$ = 22 -14 = 8 bits





Línea	Bloques de memoria principal asignados	Dirección inicial de memoria del bloque
0	0, m, 2m, ..., 2^s - m	000000, 010000, ..., FF0000
1	1, m + 1, 2m + 1, ..., 2^s – m + 1	000004, 010004, ..., FF0004
.	.	.
.	.	.
.	.	.
m - 1	m – 1, 2m – 1, 3m – 1, ..., 2^s - 1	00FFFC, 01FFFC, ..., FFFFFC

EJEMPLO

Asuma que el procesador requiere leer el dato contenido en la siguiente dirección:

FFFFF9 (hex) = 111111111111111111111001 (binario)

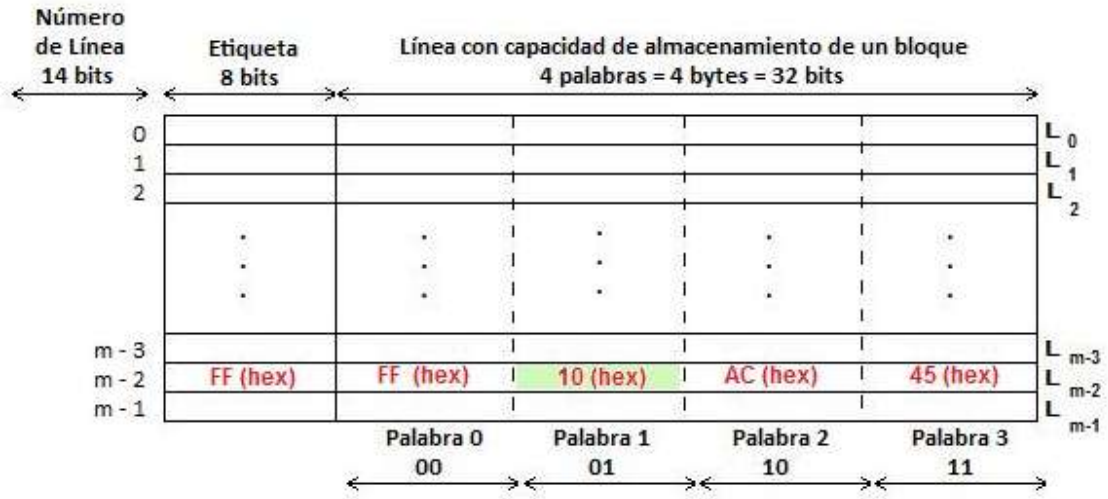
De acuerdo a los 2 bits menos significativos (01), se trata de la segunda palabra dentro del bloque. El número del bloque en el que se encuentra el dato está dado por los restantes 22 bits de la dirección (los 22 bits más significativos):

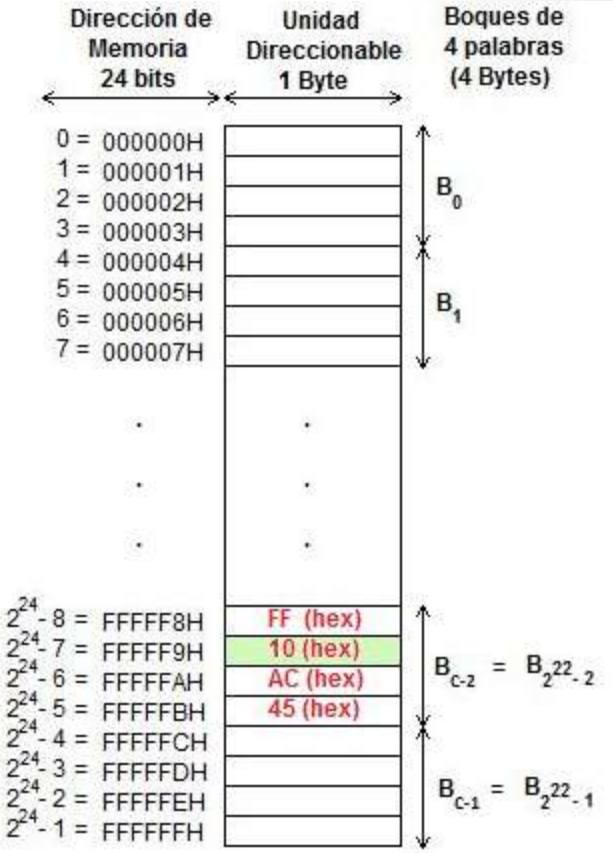
j = 1111111111111111111110 (binario) = 4,194,302 (dec)

La línea en memoria caché que corresponde a este bloque es:

i = 4,194,302 mod 16,384 = 16,382

Entonces, si el bloque se encuentra en caché hay que buscarlo en la línea número 16,382. Si este bloque ya está en caché, la etiqueta de 8 bits correspondiente a esa línea en la memoria caché debe ser igual a la etiqueta del bloque antes mencionado. Es decir, la etiqueta en caché debe ser igual a los 8 bits más significativos de la dirección FFFFF9 (hex), en otras palabras, la etiqueta debe ser igual a 11111111(bin) o. lo que es lo mismo, FF (hex). Si este es el caso, entonces el bloque ya se encuentra en memoria caché, por lo tanto el dato buscado se encuentra en la segunda palabra o byte del contenido de la línea 16,382. En caso de que no coincidan las etiquetas, el bloque no se encuentra en la caché, por lo que hay que copiarlo en la línea correspondiente.



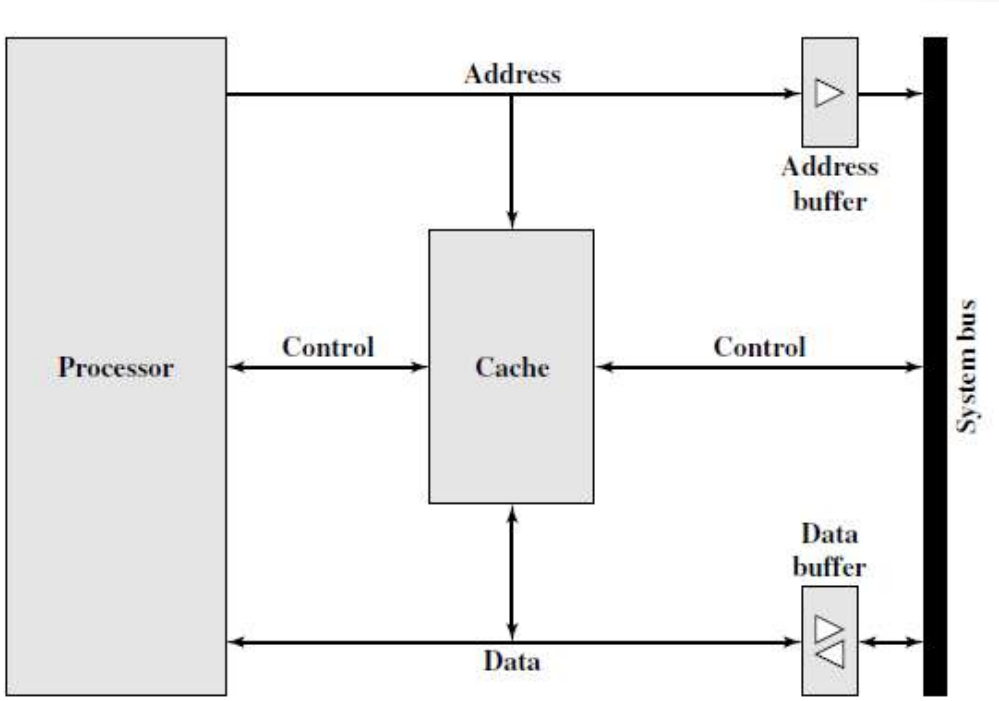


En color rojo se muestra el caso en el que el bloque del ejemplo anterior se encuentra en la memoria caché y no se ha cambiado el contenido del mismo.

Organización Típica de una Memoria Caché

- La memoria caché está conectada al procesador por medio líneas de datos, control y direcciones.
- Las líneas de datos y direcciones también se conectan a buffers de datos y direcciones, los que a la vez se conectan a los buses del sistema a través de los cuales se tiene acceso a la memoria.
- En caso de un caché hit, los buffers de datos y direcciones se inhabilitan y la comunicación ocurre solo entre el procesador y la caché, sin tráfico de bus.
- En caso de un caché miss, la dirección deseada se carga en el bus del sistema y el dato se regresa a través del buffer de datos tanto a la caché como al procesador.

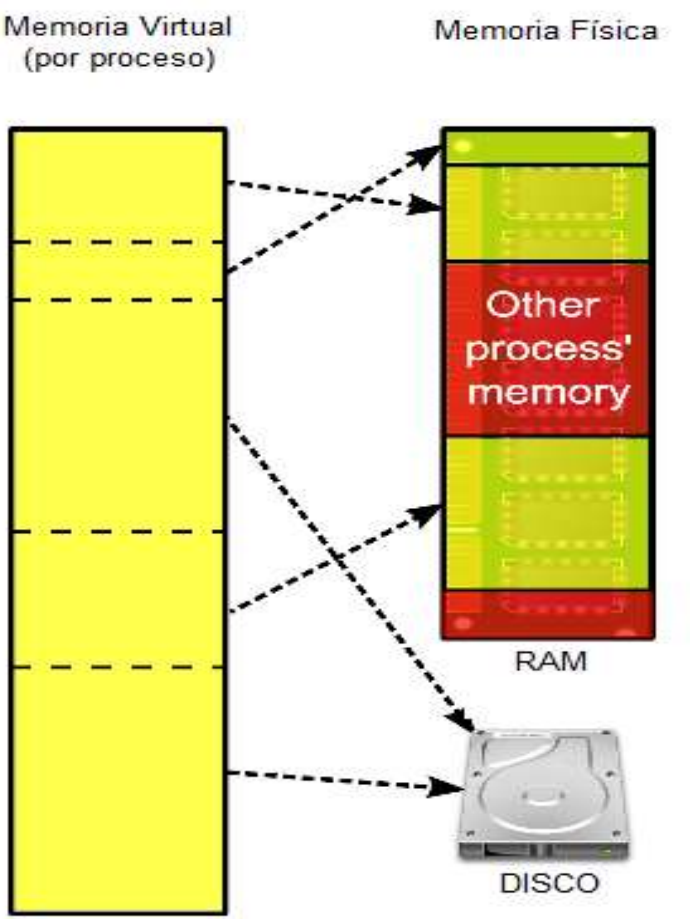
En otras organizaciones, la caché es interpuesta físicamente entre el procesador y la memoria principal para todas la líneas de datos, direcciones y control. En esta forma, para un caché miss, la palabra deseada se pone primeramente en caché y después se transfiere al procesador.



Typical Cache Organization

Direcciones de Caché

Muchos procesadores soportan **memoria virtual**, un concepto que se verá en la próxima clase. En esencia, la memoria virtual es una facilidad que permite a los programas direccionar la memoria desde un punto de vista lógico, sin preocuparse de la cantidad de memoria principal físicamente disponible ni la forma en que la memoria disponible está distribuida. Cuando se usa memoria virtual, los campos correspondientes a direcciones en el código de las instrucciones contienen **direcciones virtuales**. Para escrituras y lecturas en memoria principal, una **unidad de administración de memoria** (**MMU**) en hardware traduce cada dirección virtual en una **dirección física** de memoria principal.



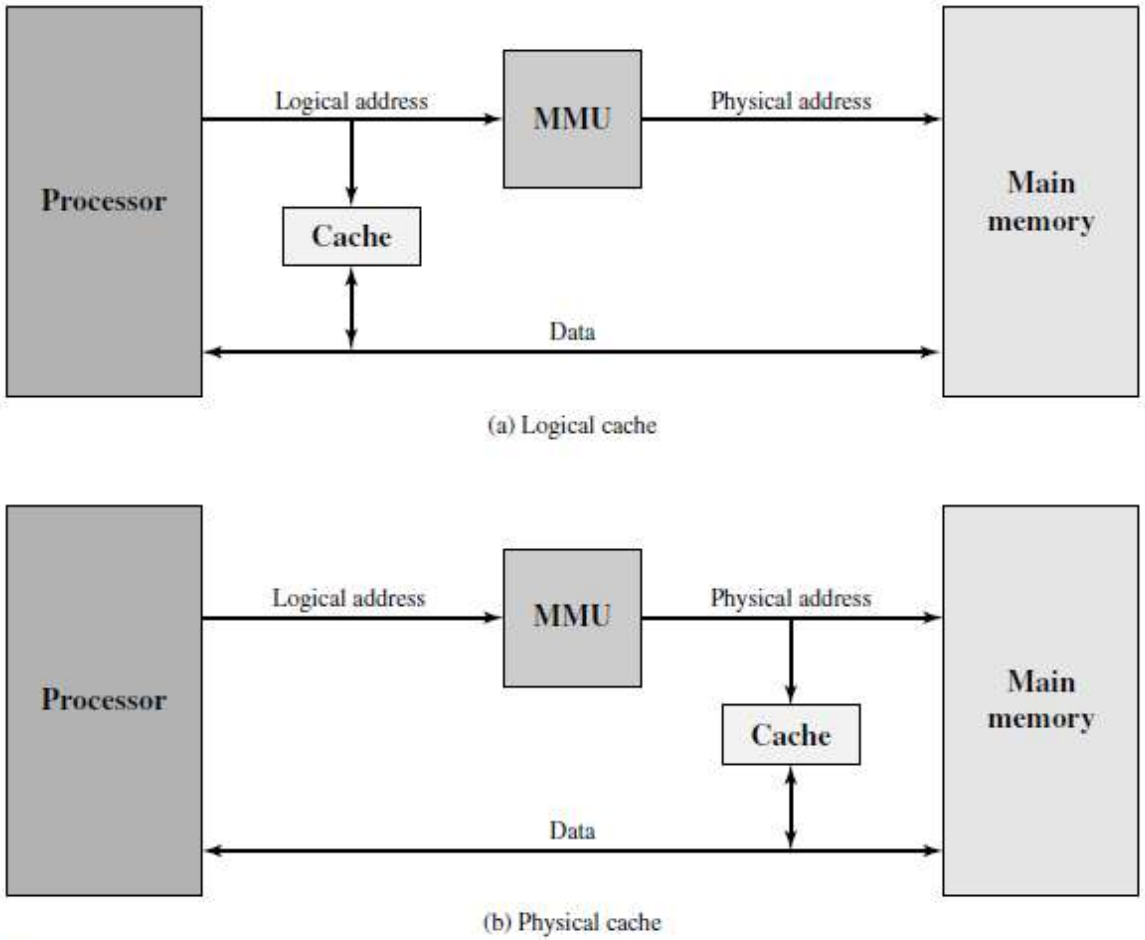
Una **caché lógica**, también conocida como **caché virtual**, almacena datos usando direcciones virtuales. El procesador tiene acceso a la caché directamente, sin pasar por la MMU.

Una **caché física** almacena los datos usando direcciones físicas de memoria principal.

- Una ventaja obvia de la caché lógica es que la velocidad de acceso de la caché es más rápida que la de una caché física, porque la caché puede responder antes de que la MMU haga la traducción o mapeo de direcciones.

- La desventaja tiene que ver con el hecho de que la mayoría de los sistemas de memoria virtual provee a cada aplicación el mismo espacio de direcciones de memoria virtual. Esto es, cada aplicación ve una memoria virtual que inicia en la dirección 0. Entonces, la misma dirección virtual en dos diferentes aplicaciones se refieren a dos diferentes direcciones físicas. La memoria caché debe, por lo tanto, ser limpiada cuando se presenta un **cambio de contexto** (**context switch**) de proceso o aplicación, o se deben añadir bits extras a cada línea de caché para identificar a cuál espacio de direcciones virtuales se refiere esta dirección.

Cuando se usan direcciones virtuales, el diseñador de sistemas puede decidir colocar la caché entre el procesador y la MMU o entre la MMU y la memoria principal.



Logical and Physical Caches

Sistemas de Caché Multinivel

Al incrementarse la densidad de los circuitos integrados, se ha hecho posible incluir memoria caché en el mismo chip que el procesador. Comparada con una **caché accesible mediante un bus externo**, la **cache en el chip** (***on-chip cache***), reduce la actividad en el bus externo y, por lo tanto, acelera los tiempos de ejecución e incrementa el desempeño del sistema. Cuando el dato o la instrucción deseada se encuentra en la caché interna, el acceso a bus es eliminado. Dado que las **rutas de datos internas al procesador** son más cortas que las longitudes del bus, los accesos a una caché interna al chip serán más rápidos, aún con ciclos de bus con **estados de cero espera**. Además, durante este periodo, el bus queda libre para dar soporte a otras transferencias.

La organización de memoria ***caché multinivel*** consiste en tener varios niveles de caché para reducir la frecuencia de acceso a la memoria principal y acelerar los tiempos de acceso. Por ejemplo, una memoria caché de tres niveles consiste en memorias caché de niveles ***nivel 1 (L1)***, ***nivel 2 (L2)*** y ***nivel 3 (L3)***, lo cual consiste en conectar tres cachés en cascada entre el procesador y la memoria principal. En los actuales diseños, todos los niveles de caché están dentro del procesador, pero en algunos casos se puede tener que la caché más externa (L3, en el ejemplo) se encuentre fuera del chip. Los tiempos de acceso son más cortos (accesos más rápidos) entre más cercano este un determinado nivel de caché y la capacidad de almacenamiento es mayor para niveles de caché menos cercanos al procesador.

Además, en los diseños más actuales, podemos encontrar que los niveles más internos de caché pueden dividirse en dos módulos cachés: uno para almacenar **bloques de datos** y otro para almacenar **bloques de instrucciones**. Esto permite acceder a datos e instrucciones de manera paralela, acelerando el procesamiento de programas.

Ejemplo: Sistema Caché del Intel Core i7 (año 2014) – Tres niveles de caché (todos internos al procesador)

Level 1 cache size ?	4 x 32 KB instruction caches 4 x 32 KB data caches
Level 2 cache size ?	**4 x 256 KB**
Level 3 cache size	**8 MB shared cache**
Cache latency	4 (L1 cache) 11 (L2 cache) 25 (L3 cache)

Tamaño de la Memoria Caché

Cache Sizes of Some Processors

Processor	Type	Year of Introduction	L1 Cache[a]	L2 Cache	L3 Cache
IBM 360/85	Mainframe	1968	16 to 32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128 to 256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256 to 512 KB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 kB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 KB	4 MB
SGI Origin 2001	High-end server	2001	32 kB/32 kB	4 MB	—
Itanium 2	PC/server	2002	32 kB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1 MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24–48 MB

[a] Two values separated by a slash refer to instruction and data caches.
[b] Both caches are instruction only; no data caches.