MEMORIA CACHE

La memoria de un computador tiene una organización jerárquica.
En el nivel superior (el más próximo al procesador) están los registros del procesador.

A continuación se encuentran uno o más niveles de caché, denominados Ll, L2, etc.

Posteriormente la memoria principal, normalmente construida con memorias dinámicas de acceso aleatorio (DRAM).

Todas ellas se consideran memorias internas del computador.
La jerarquía prosigue con la memoria externa, siendo el siguiente nivel usualmente un disco fijo, y uno o más niveles de soportes extraíbles tales como discos ópticos y cintas magnéticas, pendrives , etc.

A medida que se baja en la jerarquía de memoria disminuye el costo por bit, aumenta la capacidad y crece el tiempo de acceso.
Sería deseable poder utilizar solo la memoria más rápida, pero al ser la más costosa se llega a un compromiso entre tiempo de acceso y costo, utilizando más cantidad de memoria más lenta.
La estrategia a seguir consiste en organizar los datos y los programas en memoria de manera que las palabras de memoria necesarias estén normalmente en la memoria más rápida.

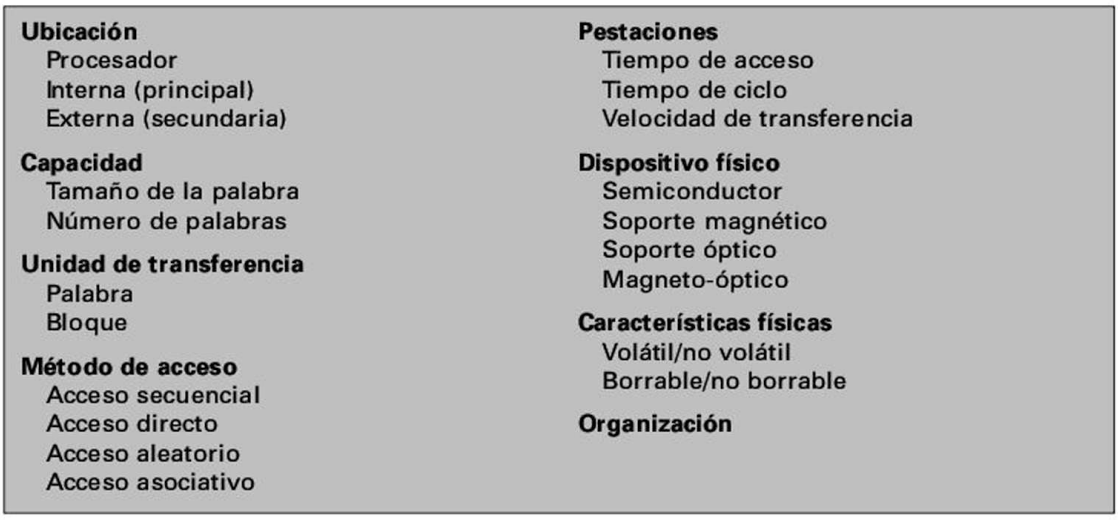
En general, es probable que la mayoría de los accesos futuros a la memoria principal, por parte del procesador, sean a posiciones accedidas recientemente.
Por eso la memoria caché automáticamente retiene una copia de algunas de las palabras de la DRAM utilizadas recientemente.
Si el caché se diseña adecuadamente, la mayor parte del tiempo el procesador solicitará palabras de memoria que están ya en la caché.

Las memorias de los computadores, aunque parezcan conceptual mente sencillas, presentan una vez la más amplia diversidad de tipos, tecnología, estructura, prestaciones y costo, de entre todos los componentes de un computador.

Casi ninguna tecnología ofrece una única solución óptima para satisfacer todas las necesidades de memoria de un computador.

En consecuencia, un computador convencional está equipado con una jerarquía de subsistemas de memoria, algunos internos (directamente accesibles por el procesador), y otros extremos (accesibles por el procesador mediante módulos de entrada/salida) .

CARACTERÍSTICAS DE LOS SISTEMAS DE MEMORIA



El término *ubicación indica si la memoria es interna o extrema al* computador.
La memoria interna suele identificarse con la memoria principal / real.

Sin embargo hay además otras formas de memoria interna.

El procesador necesita su propia memoria local en forma de registros.
Además, como veremos, la unidad de control del procesador también puede necesitar su propia memoria interna.

La memoria *caché* es también otro tipo de memoria interna.

La memoria *externa* consta de dispositivos periféricos de almacenamiento,
tales como discos y cintas, que son accesibles por el procesador a través de controladores de E/S.

Una característica obvia de las memorias es su *capacidad.*
Para memorias internas se expresa normalmente en términos de bytes (1 byte = 8 bits) o de palabras. Longitudes de palabra comunes son 8, 16, 32 y 64 bits.

Un concepto relacionado es la *unidad de trasferencia.*
Para memorias internas, la unidad de transferencia es igual al número de líneas de entrada/salida de datos del módulo de memoria. Puede ser igual a la longitud de palabra, pero suele ser mayor, por ejemplo 64, 128, o 256 bits.

Consideremos tres conceptos relacionados con la memoria interna:

Palabra: es la unidad natural de organización de la memoria.
El tamaño de la palabra suele coincidir con el número de bits utilizados para representar números y con la longitud de las instrucciones.
Pero hay muchas excepciones, hay máquinas que tienen una longitud de palabra de 64 bits, pero utiliza una representación de números enteros de 46 bits.

Unidades direccionables: en algunos sistemas la unidad direccionable es la palabra.
Sin embargo muchos de ellos permiten direccionar a nivel de bytes.
En cualquier caso, la relación entre la longitud *A* de una dirección y el número *N* de unidades direccionables, es $2^A = N$.

*Unidad de transferencia***:** para la memoria principal es el número de bits que se leen o escriben en memoria a la vez. La unidad de transferencia no tiene por qué coincidir con una palabra o con una unidad direccionable.
Para la memoria externa, los datos se transfieren normalmente en unidades más grandes que la palabra denominadas *bloques*.

Otro distintivo entre tipos de memorias es el *método de acceso,* que incluye las siguientes variantes:

Acceso secuencial: la memoria se organiza en unidades de datos llamadas registros.
El acceso debe realizarse con una secuencia lineal específica.
Se hace uso de información almacenada de direccionamiento que permite separar los registros y ayudar en el proceso de recuperación de datos.
Se utiliza un mecanismo de lectura/escritura compartida que debe ir trasladándose desde su posición actual a la deseada, pasando y obviando cada registro intermedio.
El tiempo necesario para acceder a un registro dado es muy variable.

Acceso directo: como en el caso de acceso secuencial, el directo tiene asociado un mecanismo de lectura/escritura. Sin embargo, los bloques individuales o registros tienen una dirección única basada en su dirección física.

Acceso aleatorio (random): cada posición direccionable de memoria tiene un único mecanismo de acceso cableado físicamente.
El tiempo para acceder a una posición dada es constante e independiente de la secuencia de accesos previos. Por tanto, cualquier posición puede seleccionarse y ser direccionada y accedida directamente. La memoria principal y algunos sistemas de caché son de acceso aleatorio.

Asociativa: es una memoria del tipo de acceso aleatorio que permite hacer una comparación de ciertas posiciones de bits dentro de una palabra buscando que coincidan con unos valores dados, y hacer esto para todas las palabras simultáneamente. Una palabra es por tanto recuperada basándose en una porción de su contenido en lugar de su dirección.
El tiempo de recuperación de un dato es constante independiente de la posición o de los patrones de acceso anteriores.
Las memorias caché pueden emplear acceso asociativo.

Desde el punto de vista del usuario, dos características más importantes de una memoria son su capacidad y sus *prestaciones*.
Se utilizan tres parámetros de medida de prestaciones:

Tiempo de acceso (latencia): para memorias de acceso aleatorio es el tiempo que tarda en realizarse una operación de escritura o de lectura, es decir, el tiempo que transcurre desde el instante en el que se presenta una dirección a la memoria hasta que el dato, ha sido memorizado, o está disponible para su uso.
Para memorias de otro tipo, el tiempo de acceso es el que se tarda en situar el mecanismo de lectura/escritura en la posición deseada.

Tiempo de ciclo de memoria: este concepto se aplica principalmente a las memorias de acceso aleatorio y consiste en el tiempo de acceso y algún tiempo más que se requiere antes de que pueda iniciarse un segundo acceso a memoria.
Este tiempo adicional puede que sea necesario para que finalicen las transiciones en las líneas de señal o para regenerar los datos en el caso de lecturas destructivas.
El tiempo de ciclo de memoria depende de las características del bus del sistema y no del procesador.

Velocidad de transferencia: es la velocidad a la que se pueden transferir datos a, o desde, una unidad de memoria. Para memorias de acceso aleatorio coincide con el inverso del tiempo de ciclo.

Para otras memorias se utiliza la siguiente relación: $T_N = T_A + N/R$

Tn = Tiempo medio de escritura o de lectura de N bits
Ta = Tiempo de acceso medio
N = Número de bits
R = Velocidad de transferencia, en bits por segundo (bps)

Del almacenamiento de datos son importantes varias *características físicas.*
En memorias *volátiles* la información se va perdiendo o desaparece cuando se desconecta la alimentación.

En las memorias *no volátiles* la información, una vez grabada, permanece sin deteriorarse hasta que se modifique intencionadamente; no se necesita la fuente de alimentación para retener la información.

Las memorias de *superficie magnética* son no volátiles, las semiconductoras pueden ser volátiles o no volátiles.

Las memorias no borrables no pueden modificarse, salvo que se destruya la unidad de almacenamiento.
Las memorias semiconductoras de este tipo se conocen por el nombre de *memorias de solo lectura* (ROM, *Read Only Memory).*

Una memoria no borrable es necesariamente no volátil.

En memorias de acceso aleatorio, su *organización* es un aspecto clave de diseño.
Por *organización* se entiende su disposición o estructura física en bits para formar palabras.
La estructura más obvia no es siempre la utilizada en la práctica.

JERARQUÍA DE MEMORIA

Las restricciones de diseño de la memoria de un computador se podría resumir en tres cuestiones: capacidad , velocidad y costo.

El tamaño es un tema siempre presente.
Si se consigue hasta una cierta capacidad, probablemente se desarrollarán aplicaciones que la utilicen.

La cuestión de la velocidad es, en cierto sentido,es fácil de responder.
Para conseguir las prestaciones óptimas, la memoria debe seguir al procesador.
Es decir, cuando el procesador ejecuta instrucciones, no es aceptable que tenga que detenerse a la espera de instrucciones o de operandos.

Y en la práctica, el costo de la memoria debe ser razonable con relación a los otros componentes.

Existe un compromiso entre las tres características clave de costo, capacidad, y tiempo de acceso.
Se emplean diversas tecnologías para realizar los sistemas de memoria y dentro de las posibles tecnologías se cumplen las siguientes relaciones:

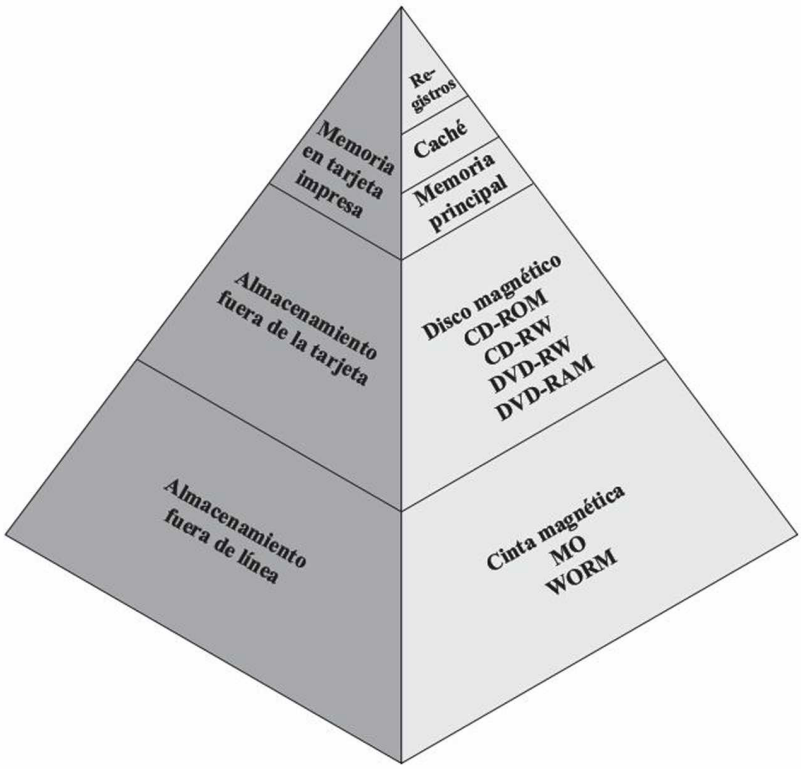
A menor tiempo de acceso, mayor costo por bit.

A mayor capacidad, menor costo por bit.

A mayor capacidad, mayor tiempo de acceso.

El dilema con que se enfrenta el diseñador está claro.
Desearía utilizar tecnologías de memoria que proporcionen gran capacidad, tanto porque esta es necesaria como porque el costo por bit es bajo.
Pero para satisfacer las prestaciones requeridas, el diseñador necesita utilizar memorias costosas, de capacidad relativamente baja y con tiempos de acceso reducidos.

La respuesta es no contar con un solo componente de memoria, sino emplear una *jerarquía de memoria*



Cuando se desciende en la jerarquía ocurre:

a) Disminuye el coste por bit.
b) Aumenta la capacidad.
c) Aumenta el tiempo de acceso.
d) Disminuye la frecuencia de accesos a la memoria por parte del procesador.

Las memorias más pequeñas, más costosas y más rápidas, se complementan con otras más grandes, más económicas y más lentas.
La clave de esta organización está en el último *item* (d):
la disminución de la frecuencia de acceso

Ejemplo .

Suponer que el procesador tiene que acceder a dos niveles de la memoria.

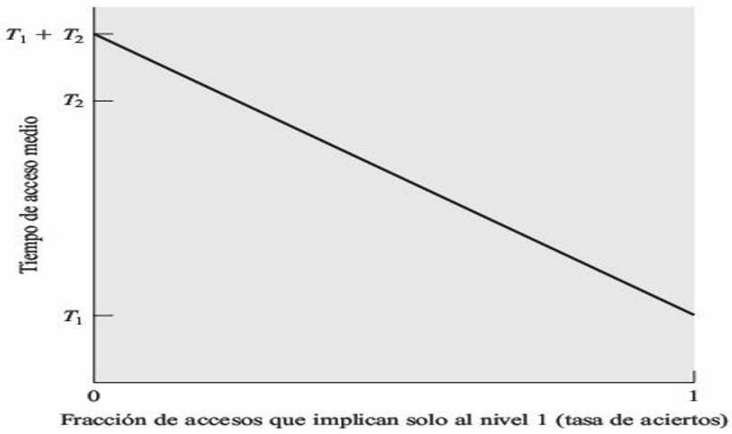
El nivel 1 contiene 1000 palabras y tiene un tiempo de acceso de 0,01 ns.
El nivel 2 contiene 100000 palabras y tiene un tiempo de acceso de 0,1ns.

Su poniendo que si la palabra a la que se va a acceder está en el nivel 1,
el procesador accede a ella directamente.

Pero si está en el nivel 2, entonces primero es transferida al nivel 1 y recién
después accedida por el procesador.

Por simplicidad se ignora el tiempo necesario para que el procesador determine
si la palabra está en un nivel u otro.

La figura muestra el tiempo de a acceso medio a una memoria de dos niveles, en función de la tasa de acierto *H, donde H* se define como la fracción del total de accesos a memoria encontrados en la memoria más rápida por ejemplo en el cache.



T_1 es el tiempo de acceso al nivel 1 y T_2 el tiempo de acceso al nivel 2.

Como puede verse, para porcentajes altos de accesos al nivel 1, el tiempo de acceso total promedio es mucho más próximo al del nivel 1 que al del nivel 2.

Si el 95 por ciento de los acceso a memoria se encuentran con éxito en el caché, entonces el tiempo medio para acceder a una palabra puede expresarse en la forma:

(0,95) (0,01 /ns) + (0,05) (0,01 ns + 0,1 ns) = 0,0095ns + 0,0055 ns = 0,015 ns

Como era deseable, el tiempo de acceso medio está mucho más próximo a 0,01ns que a 0,1 ns.

En principio, el uso de dos niveles de memoria para reducir el tiempo de acceso medio funciona, pero solo si se aplican las condiciones (a) a (d) anteriores.
Empleando diversas tecnologías se tiene todo un espectro de sistemas de memoria que satisfacen las condiciones (a) a (c).
Afortunadamente, la condición (d) es también generalmente valida.

La base para la validez de la condición (d) es el principio conocido como *localidad de las referencias*

En el curso de la ejecución de un programa, las referencias a memoria por parte del procesador, tanto para instrucciones como para datos, tienden a estar agrupadas.

Los programas normalmente contienen un número de bucles iterativos y subrutinas.
Cada vez que se entra en un bucle o una subrutina, hay repetidas referencias a un pequeño conjunto de instrucciones. De manera similar, operaciones con tablas o con matrices implican accesos a un conjunto de palabras de datos agrupadas.

En periodos de tiempo largos, las agrupaciones en uso cambian, pero en periodos de tiempo cortos, el procesador trabaja principalmente con agrupaciones fijas de referencias a memoria.

De acuerdo con lo anterior, es posible organizar los datos a través de la jerarquía de tal manera que el porcentaje de accesos a cada nivel siguiente más bajo sea sustancialmente menor que al nivel anterior.

Suponiendo que en el caso presentado de dos niveles la memoria del nivel 2 contiene todos los datos e instrucciones de programa.
Las agrupaciones actuales pueden ubicarse temporalmente en el nivel 1.

De vez en cuando, una de las agrupaciones del nivel 1 tendrá que ser devuelta al nivel 2 para dar lugar a que entre otra nueva agrupación al nivel 1.

En general, sin embargo, la mayoría de las referencias serán a instrucciones y datos contenidos en el nivel 1.

Este principio puede aplicarse a través de más de dos niveles de memoria, como sugiere la jerarquía mostrada .

El tipo de memoria más rápida, pequeña y costosa, lo constituyen los registros internos al procesador.

Un procesador suele contener unas cuantas docenas de tales registros,aunque algunas máquinas contienen cientos de ellos.

Descendiendo dos niveles, la memoria principal es el principal sistema de memoria interna del computador.
Cada posición de memoria principal tiene una única dirección.
La memoria principal es normalmente ampliada con una caché, que es más pequeña y rápida.

El caché no suele estar visible al programador, y realmente tampoco al procesador.

Es un dispositivo para escalonar las transferencias de datos entre memoria principal y los registros del procesador a fin de mejorar las prestaciones.

Las tres formas de memoria que acabamos de describir son, normalmente, volátiles y de tecnología semiconductora.

El uso de tres niveles aprovecha la variedad existente de tipos de memorias semiconductoras,que difieren en velocidad y costo.
El almacenamiento de datos de forma permanente se hace en dispositivos de memoria masiva, como discos y los dispositivos extraíbles, tales como discos extraíbles, cintas , dispositivos ópticos de almacenamiento,,,etc

Las memorias externas no volátiles o permanentes se denominan también memorias secundarias o auxiliares. Se utilizan para almacenar programas y archivos de datos, y suelen estar visibles al programador solo en términos de archivos y registros, en lugar de bytes aislados o de palabras.

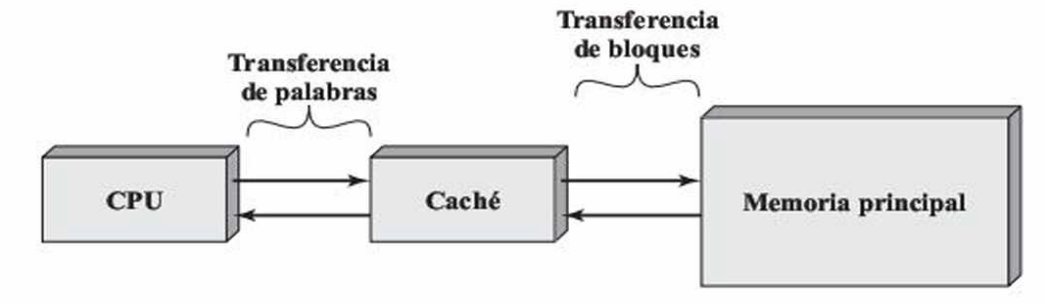
Mediante software se pueden añadir más niveles a la jerarquía.
Una parte de la memoria principal puede utilizarse como área intermedio (*buffer*) para guardar temporalmente datos que van a ser volcados en disco.
Esta técnica, a veces denominada caché de disco mejora las prestaciones de dos maneras:

Las escrituras en disco se hacen por grupos.
En lugar de muchas transferencias cortas de datos, tenemos pocas transferencias largas.
Esto mejora las prestaciones del disco y minimiza la participación del procesador.

Algunos datos destinados a ser escritos como salidas pueden ser referenciados por un programa antes de que sean volcados en disco. En ese caso, los datos se recuperan rápidamente desde la caché software en lugar de hacerlo lentamente de disco.

FUNCIONAMIENTO.

El objetivo de la memoria caché es lograr que la velocidad de la memoria sea lo más rápida posible, consiguiendo al mismo tiempo una capacidad grande al precio de memorias semiconductoras menos costosas.



Hay una memoria principal relativamente grande y más lenta, junto con una memoria caché más pequeña y rápida.
La memoria caché contiene una copia de partes de la memoria principal.
Cuando el procesador intenta leer una palabra de memoria, se hace una comprobación para determinar si la palabra está en el caché.
Si es así, se entrega dicha palabra al procesador.

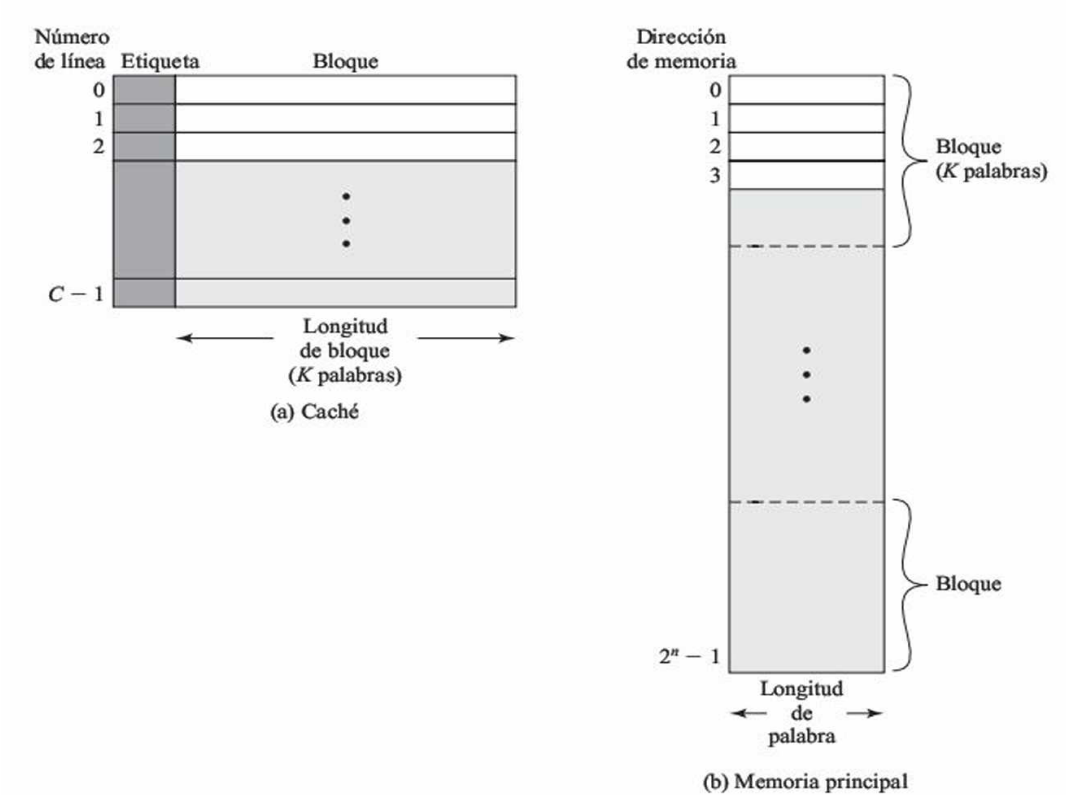
Caso contrario un bloque de memoria principal, consistente en un cierto número de palabras, se transfiere al caché y después la palabra es entregada al procesador.

Debido al fenómeno de localidad de las referencias, cuando un bloque de datos es capturado por el caché para satisfacer una referencia a memoria simple, es probable que se hagan referencias futuras a la misma posición de memoria o a otras palabras del mismo bloque.

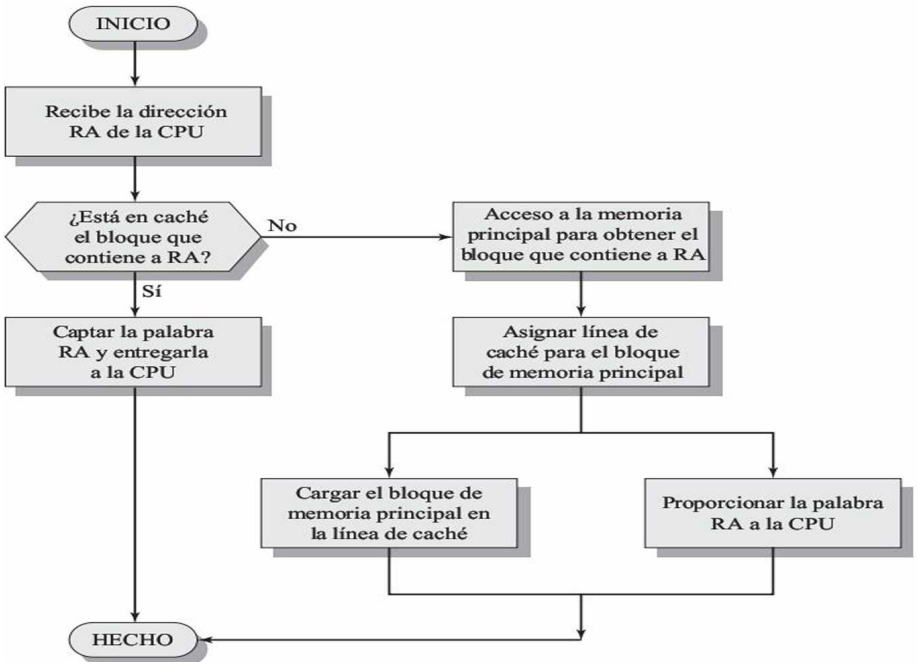
La memoria principal consta de hasta 2^n palabra direccionables, teniendo cada palabra una única dirección de n bits.
Esta memoria la consideramos dividida en un número de bloques de longitud fija, de K palabras por bloque. Es decir, hay $M = 2^n/K$ bloques.

El caché consta de C *líneas* y cada línea contiene K palabras, más una etiqueta denominándose tamaño de línea al número de palabras que hay en la línea.

El número de líneas es considerablemente menor que el número de bloques de memoria principal ($C < M$). Y en todo momento, un subconjunto de los bloques de memoria reside en líneas de caché.
Si se lee una palabra de un bloque de memoria, dicho bloque es transferido a una de las líneas del caché.
Ya que hay más bloques que líneas, una línea dada no puede dedicarse unívoca y permanentemente a un bloque. Por consiguiente, cada línea incluye una *etiqueta* que identifica qué bloque particular de memoria almacena.
La etiqueta es usualmente una porción de la dirección de memoria principal.

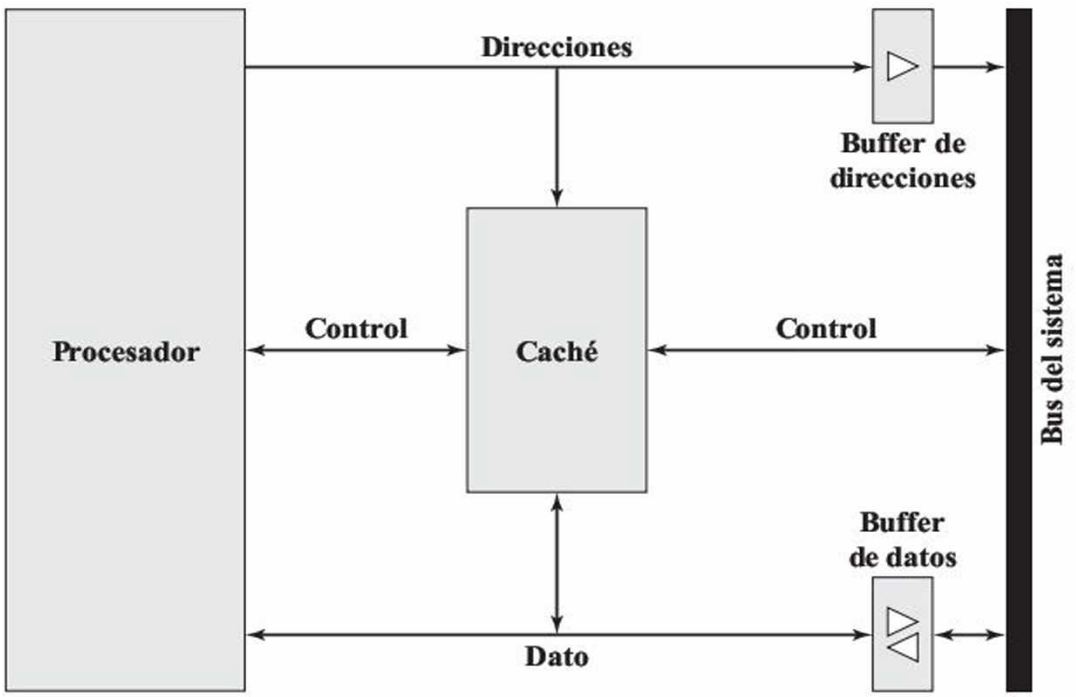


El procesador genera la dirección, RA, de una palabra a leer.
Si la palabra está en caché, es entregada al procesador.
Caso contrario el bloque que contiene dicha palabra se carga en el caché,
y la palabra después es llevada al procesador.
Estas dos últimas operaciones se realizan en paralelo.



El caché conecta con el procesador mediante líneas de datos, de control y de direcciones. Las líneas de datos y de direcciones conectan también con buffers de datos y de direcciones que las comunican con un bus del sistema a través del cual se accede a la memoria principal.
Cuando ocurre un acierto de caché, los buffers de datos y de direcciones se inhabilitan, y la comunicación tiene lugar solo entre procesador y caché, sin tráfico en el bus.

Cuando ocurre un fallo de caché, la dirección deseada se carga en el bus del sistema y el dato es llevado, a través del buffer de datos, tanto a la caché como al procesador.



En algunas otras formas de organización, el caché se interpone físicamente entre el procesador y la memoria principal para todas las líneas de datos, direcciones y control. En este caso, frente a un fallo de caché, la palabra deseada es primero leída por el caché y después transferida desde esta al procesador.

ELEMENTOS DE DISEÑO DE LA CACHÉ.

Tamaño de caché	**Política de escritura**
Función de correspondencia	Escritura inmediata
Directa	Postescritura
Asociativa	Escritura única
Asociativa por conjuntos	**Tamaño de línea**
Algoritmo de sustitución	**Número de cachés**
Utilizado menos recientemente (LRU)	Uno o dos niveles
Primero en entrar-primero en salir (FIFO)	Unificada o partida
Utilizado menos frecuentemente (LFU)	
Aleatorio	

TAMAÑO DE CACHÉ.

Sería ideal que el tamaño fuera lo suficientemente pequeño como para que el costo total medio por bit se aproxime al de la memoria principal sola, y que fuera lo suficientemente grande como para que el tiempo de acceso medio total sea próximo al de caché solo.

Hay otras muchas motivaciones para minimizar el tamaño del caché.
Cuanto más grande es, mayor es el número de puertas implicadas en direccionar el caché. El resultado es que un caché grande tiende a ser ligeramente más lento que los pequeñas (incluso estando fabricadas con la misma tecnología de circuito integrado y con la misma ubicación en el chip o en la tarjeta de circuito impreso).

El tamaño de caché está también limitado por las superficies disponibles de chip y de tarjeta. Como las prestaciones del caché son muy sensibles al tipo de tarea, es difícil predecir un tamaño «óptimo».

FUNCIÓN DE CORRESPONDENCIA.

Ya que hay menos líneas de caché que bloques de memoria principal, se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de caché.

Además, se requiere algún medio para determinar qué bloque de memoria principal ocupa actualmente una línea dada de caché.

La elección de la función de correspondencia determina cómo se organiza el caché. Pueden utilizarse tres técnicas: *directa, asociativa, y asociativa por conjuntos.*

Correspondencia directa.

La técnica más sencilla, denominada correspondencia directa,consiste en hacer corresponder cada bloque de memoria principal a solo una línea posible de caché.

La función de correspondencia se implementa fácilmente utilizando la dirección.
Esta técnica de correspondencia directa es sencilla y poco costosa de implementar.

Su principal desventaja es que hay una posición concreta de caché para cada bloque dado. Por ello, si un programa referencia repetidas veces a palabras de dos bloques diferentes asignados en la misma línea, dichos bloques se estarían intercambiando continuamente en el caché, y la tasa de aciertos sería baja (*thrashing*).

Correspondencia asociativa.

La correspondencia asociativa supera la desventaja de la directa,permitiendo que cada bloque de memoria principal pueda cargarse en cualquier línea del caché.

En este caso, la lógica de control de la caché interpreta una dirección de memoria simplemente como una etiqueta y un campo de palabra.

El campo de etiqueta identifica unívoca mente un bloque de memoria principal.
Para determinar si un bloque está en la caché, su lógica de control debe examinar simultáneamente todas las etiquetas de líneas para buscar una coincidencia.

Con la correspondencia asociativa hay flexibilidad para que cualquier bloque sea reemplazado cuando se va a escribir uno nuevo en caché.
La principal desventaja de la correspondencia asociativa es la compleja circuitería necesaria para examinar en paralelo las etiquetas de todas las líneas de caché.

Correspondencia asociativa por conjuntos.

La correspondencia asociativa por conjuntos es una solución de compromiso que toma lo positivo de las correspondencias directa y asociativa, sin presentar sus desventajas.
En este caso, la caché se divide en c conjuntos, cada uno de l líneas.

ALGORITMOS DE SUSTITUCIÓN

Una vez que se ha llenado la caché, para introducir un nuevo bloque debe sustituirse uno de los bloques existentes.

Para el caso de correspondencia directa, solo hay una posible línea para cada bloque particular y no hay elección posible.

Para las técnicas asociativas se requieren algoritmos de sustitución.

Para conseguir alta velocidad, tales algoritmos se implementarse en hardware.
Se han probado diversos algoritmos; algunos de los más comunes :

El más efectivo es probablemente el denominado *"utilizado menos recientemente"* (LRU, *least-recently used*): se sustituye el bloque que se ha mantenido en la caché por más tiempo sin haber sido referenciado. Esto es fácil de implementar para la asociativa por conjuntos de dos vías, cada línea incluye un bit USO.
Cuando una línea es referenciada se pone en 1 su bit USO y a 0 el de la otra línea del mismo conjunto.
Cuando va a transferirse un bloque al conjunto, se utiliza la línea cuyo bit USO es 0.
Ya que estamos suponiendo que son más probables de referenciar las posiciones de memoria utilizadas más recientemente, el LRU debiera dar la mejor tasa de aciertos.

Otra posibilidad es el *primero en entrar primero en salir* (FIFO, *First-In-First-Out*): se sustituye aquel bloque del conjunto que ha estado más tiempo en caché.

El algoritmo FIFO puede implementarse fácilmente mediante una técnica cíclica (*round-robin) o* buffer circular.

Otra posibilidad más es la del *utilizado menos frecuentemente*
(LFU, *Least-Frequently Used):*
se sustituye aquel bloque del conjunto que ha experimentado menos referencias.
LFU podría implementarse asociando un contador a cada línea.

Una técnica no basada en el grado de utilización consiste simplemente en tomar una línea al azar (aleatoria) entre las posibles candidatas. La sustitución aleatoria proporciona unas prestaciones solo ligeramente inferiores a un algoritmo basado en la utilización.

POLÍTICA DE ESCRITURA

Hay dos casos a considerar cuando se ha de reemplazar un bloque del caché.

Si el bloque antiguo del caché no fue modificado, puede sobrescribirse con el nuevo bloque sin necesidad de actualizar el antiguo en memoria,

Si se ha realizado al menos una operación de escritura sobre una palabra de la línea correspondiente del caché, entonces la memoria principal debe actualizarse, reescribiendo la línea de caché en el bloque de memoria antes de transferir el nuevo bloque.

Son posibles varias políticas de escritura con distintos compromisos entre prestaciones y costo económico.

Hay dos problemas a considerar :

En primer lugar, más de un dispositivo puede tener acceso a la memoria principal.
Por ejemplo, un módulo de E/S puede escribir/leer directamente en/de memoria.

Si una palabra ha sido modificada solo en caché, la correspondiente palabra de memoria no es válida.

Además, si el dispositivo de E/S ha alterado la memoria principal, entonces la palabra de caché no es válida

Un problema más complejo ocurre cuando varios procesadores se conectan al mismo bus y cada uno de ellos tiene su propio caché local.
En tal caso , si se modifica una palabra en una de los caché, podría presumible mente invalidar una palabra de otras caché.

La técnica más sencilla se denomina **escritura inmediata.**
Utilizando esta técnica, todas las operaciones de escritura se hacen tanto en caché como en memoria principal, asegurando que el contenido de la memoria principal siempre es válido.
Cualquier otro módulo procesador-caché puede monitorizar el tráfico a memoria principal para mantener la coherencia en su propia caché.

La principal desventaja de esta técnica es que genera un tráfico sustancial con la memoria que puede originar un cuello de botella.

Una técnica alternativa, conocida como **post escritura,** minimiza las escrituras en memoria.
Con la post escritura, las actualizaciones se hacen solo en la caché. Cuando tiene lugar una actualización, se activa un bit ACTUALIZAR asociado a la línea.
Después, cuando el bloque es sustituido, es (post) escrito en memoria principal si y solo si el bit ACTUALIZAR está activo.

El problema de este esquema es que se tienen porciones de memoria principal que no son válidas, y los accesos por parte de los módulos de E/S tendrán que hacerse solo a través de la caché. Esto complica la circuitería y genera un cuello de botella potencial.

En una estructura de bus en la que más de un dispositivo (normalmente un procesador) tiene su caché y la memoria principal es compartida, se presenta un nuevo problema.

Si se modifican los datos de una caché, se invalida no solamente la palabra correspondiente de memoria principal, sino también la misma palabra en otras caché (si coincide que otras caché tengan la misma palabra).

Incluso si se utiliza una política de escritura inmediata, los otras caché pueden contener datos no válidos.
Un sistema que evite este problema se dice que mantiene *la coherencia de caché.*
Entre las posibles aproximaciones a *la coherencia de caché* se incluyen:

Vigilancia del bus con escritura inmediata: cada controlador de caché monitoriza las líneas de direcciones para detectar operaciones de escritura en memoria por parte de otros maestros del bus.
Si otro maestro escribe en una posición de memoria compartida que también reside en la memoria caché, el controlador de caché invalida el elemento de la caché.
Esta estrategia depende del uso de una política de escritura inmediata por parte de todos los controladores de caché.

Transparencia hardware: se utiliza hardware adicional para asegurar que todas las actualizaciones de memoria principal, vía caché, quedan reflejadas en todos los caché.
Así, si un procesador modifica una palabra de su caché, esta actualización se escribe en memoria principal.
Además, de manera similar se actualizan todas las palabras coincidentes de otras caché.

Memoria excluida de caché: solo una porción de memoria principal se comparte por más de un procesador, y esta se diseña como no transferible a caché.
En un sistema de este tipo, todos los accesos a la memoria compartida son fallos de caché, porque la memoria compartida nunca se copia en la caché.
La memoria excluida de caché puede ser identificada utilizando lógica de selección de chip o los bits más significativos de la dirección.

TAMAÑO DE LÍNEA

Otro elemento de diseño es el tamaño de línea.

Cuando se recupera y ubica en caché un bloque de datos, se recuperan no sólo la palabra deseada sino además algunas palabras adyacentes.
A medida que aumenta el tamaño de bloque, la tasa de aciertos primero aumenta debido al principio de localidad, el cual establece que es probable que los datos en la vecindad de una palabra referenciada sean referenciados en un futuro próximo.
Al aumentar el tamaño de bloque, más datos útiles son llevados a la caché.

Sin embargo, la tasa de aciertos comenzará a decrecer cuando el tamaño de bloque se haga aún mayor y la probabilidad de utilizar la nueva información captada se haga menor que la de reutilizar la información que tiene que reemplazarse.

Dos efectos concretos entran en consideración :

• Bloques más grandes reducen el número de bloques que caben en la caché.
Dado que cada bloque captado se escribe sobre contenidos anteriores de la caché, un número reducido de bloques da lugar a que se sobrescriban datos poco después de haber sido captados.

• A medida que un bloque se hace más grande, cada palabra adicional está más lejos de la requerida y por tanto es más improbable que sea necesaria a corto plazo.
La relación entre tamaño de bloque y tasa de aciertos es compleja, dependiendo de las características de localidad de cada programa particular, no habiéndose encontrado un valor óptimo definitivo.

NÚMERO DE CACHÉ.

Cuando se introdujeron originalmente los caché, un sistema tenía normalmente solo un caché. Luego , se ha convertido en una norma el uso de múltiples caché.

Hay dos aspectos de diseño relacionados con este tema que son el número de niveles de caché, y el uso de caché unificada frente al de caché separadas.

Caché multinivel.

Con el aumento de densidad de integración, ha sido posible tener una caché en el mismo chip del procesador: caché *on-chip.*

Comparada con la accesible a través de un bus externo, la caché *on-chip* reduce la actividad del bus externo del procesador y por tanto reduce los tiempos de ejecución e incrementa las prestaciones globales del sistema.

Cuando la instrucción o el dato requeridos se encuentran en la caché *on-chip,* se elimina el acceso al bus. Debido a que los caminos de datos internos al procesador son muy cortos en comparación con la longitud de los buses, los accesos a la caché *on-chip* se efectúan más rápidos que los ciclos de bus, incluso en ausencia de estados de espera.

Además, durante este periodo el bus está libre para realizar otras transferencias.
La inclusión de una caché *on-chip* deja abierta la cuestión de si es además deseable un caché externa u *off-chip.* Normalmente la respuesta es afirmativa, y los diseños incluyen tanto caché *on-chip* como externa.

La estructura más sencilla de este tipo se denomina caché de dos niveles, siendo la caché interna el nivel 1 (Ll), y la externa el nivel 2 (L2).

La razón por la que se incluye una caché L2 es la siguiente:

Si no hay caché L2 y el procesador hace una petición de acceso a una posición de memoria que no está en la caché Ll, entonces el procesador debe acceder a la DRAM o la ROM a través del bus. Y debido a la lentitud usual del bus y a los tiempos de acceso de las memorias, se obtienen bajas prestaciones.

Por otra parte, si se utiliza una caché L2 SRAM (RAM estática), entonces con frecuencia la información que falta puede recuperarse fácilmente. Si la SRAM es suficientemente rápida para adecuarse a la velocidad del bus, los datos pueden accederse con cero estados de espera, el tipo más rápido de transferencia de bus.

En el caso de una caché L2 externa, muchos diseños no usan el bus del sistema como camino para las transferencias entre la caché L2 y el procesador, sino que se emplea un camino de datos aparte para reducir el tráfico en el bus del sistema

También es fácil encontrar procesadores que incorporan la caché L2, L3 en el propio chip, con la consiguiente mejora de prestaciones.

El uso del caché multinivel mejora las prestaciones y complica todos los aspectos de diseño incluyendo el tamaño, el algoritmo de sustitución y la política de escritura.

Caché unificada frente a caché separados.

Cuando hicieron su aparición las caché *onchip,* muchos de los diseños contenían solo un caché para almacenar las referencias tanto a datos como a instrucciones.

Luego se ha hecho normal separar la caché en dos: uno dedicado a instrucciones y otro a datos.
Un caché unificado tiene varias ventajas potenciales:

- Para un tamaño dado de caché, el unificado tiene una tasa de aciertos mayor que un caché separado, ya que nivela automáticamente la carga entre captación de instrucciones y de datos.

Es decir, si un patrón de ejecución implica muchas más captaciones de instrucciones que de datos, tenderá a llenarse con instrucciones, y si el patrón de ejecución involucra relativamente más captaciones de datos, ocurrirá lo contrario.

- Solo se necesita diseñar e implementar un solo caché.

A pesar de estas ventajas, la tendencia es hacia caché separadas, particularmente para máquinas super-escalares en las que se enfatiza la ejecución paralela de instrucciones y la pre captación de instrucciones futuras previstas.
La ventaja clave del diseño del caché separado es que elimina la competición por la caché entre el procesador de instrucciones y la unidad de ejecución.
Esto es importante en diseños que cuentan con segmentación de cauce *(pipelining)* de instrucciones.

Normalmente el procesador captará instrucciones anticipadamente y llenará un buffer con las instrucciones que van a ejecutarse.

Supongamos ahora que se tiene una caché unificada de instrucciones/datos.

Cuando la unidad de ejecución realiza un acceso a memoria para cargar y almacenar datos,se envía la petición a la caché unificada. Si, al mismo tiempo, el pre captador de instrucciones emite una petición de lectura de una instrucción al caché, dicha petición será bloqueada temporalmente para que la caché pueda servir primero a la unidad de ejecución permitiéndose completar la ejecución de la instrucción en curso.

Esta disputa por la caché puede degradar las prestaciones, interfiriendo con el uso eficiente del cauce segmentado de instrucciones.
El caché separado supera esta dificultad.

Problema	Solución
Memoria externa más lenta que el bus del sistema.	Añadir caché externa usando tecnología de memoria más rápida.
El aumento de velocidad de los procesadores hace que el bus se convierta en un cuello de botella para el acceso a caché.	Trasladar la caché externa al mismo chip (caché *on-chip*), funcionando a la misma velocidad que el procesador.
La caché es pequeña debido a la disponibilidad de superficie *on-chip* limitada.	Añadir una caché L2 externa con tecnología más rápida que la empleada en la memoria principal.
Se produce contención cuando las unidades de precaptación de instrucciones y de ejecución necesitan acceder simultáneamente a la caché. En este caso, la unidad de precaptación se bloquea mientras la unidad de ejecución accede a los datos.	Crear cachés separadas de datos y de instrucciones.
El incremento de velocidad del procesador hace que el bus externo sea un cuello de botella para el acceso a la caché L2.	Crear un bus externo específico o «puerta trasera» (BSB, *back-side bus*) que funcione a más velocidad que el bus principal. El BSB se dedica a la caché L2.
	Llevar la caché L2 al chip del procesador.
Algunas aplicaciones que trabajan con grandes bases de datos deben tener acceso rápido a cantidades ingentes de datos. Las cachés *on-chip* son demasiado pequeñas.	Añadir una caché L3 externa.
	Integrar la caché L3 *on-chip*.