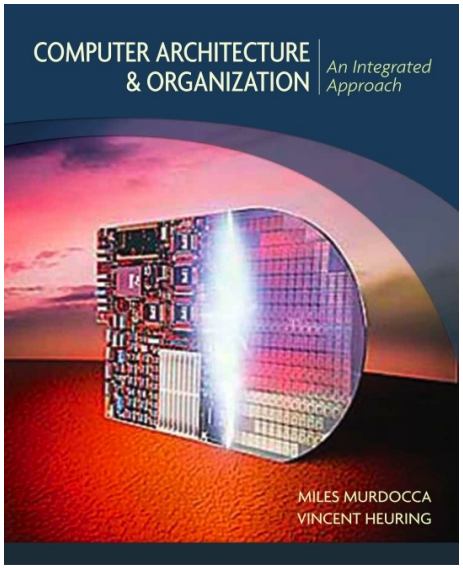
Organización de las Computadoras

Leonardo Giovanini



Organización de la memoria

Memoria local

Contenidos

7.1 Ejecución de programas

7.2 Memoria scratchpad

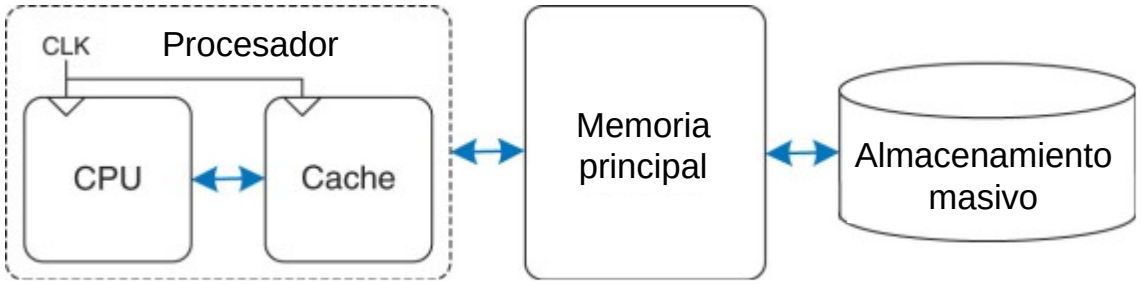
7.3 Memoria fuertemente acoplada

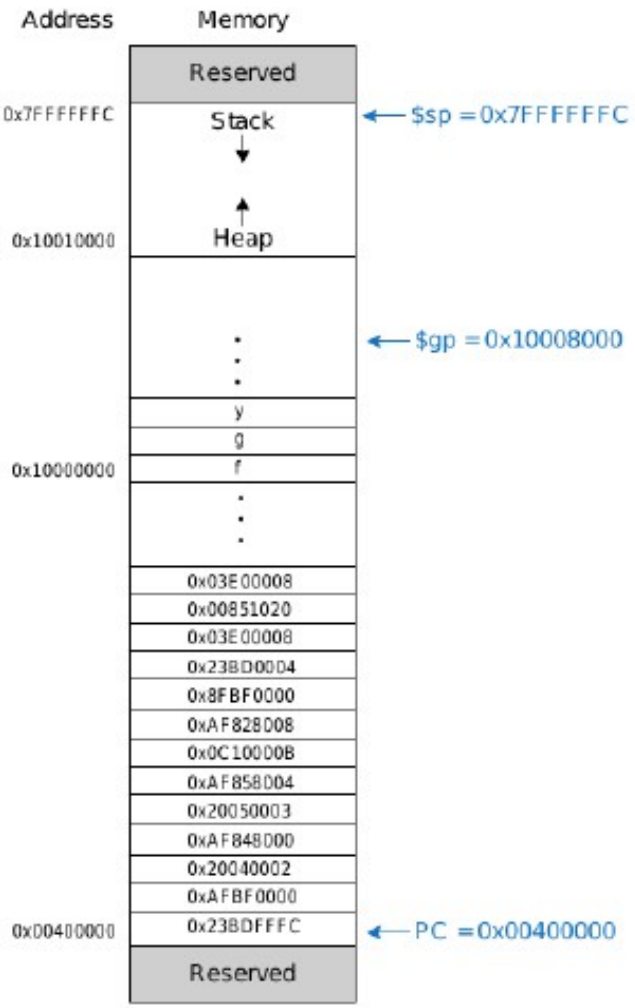
7.3 Memoria cache

Ejecución de programas

Ejecución de un programa

El sistema operativo lee el archivo ejecutable del dispositivo de almacenamiento y lo carga en la memoria de la computadora para su ejecución.





Dependiendo del entorno utilizado,

- En ***procesadores avanzados*** debe ubicarse en el contexto de un **proceso**, y la gestión esta a cargo de la unidad de manejo de memoria,
- En ***procesadores convencionales*** se ubica en la memoria disponible y la gestión esta a cargo del programador.

La ubicación de cada segmento del programa (código, datos, variables locales y pila) depende de las características técnicas del procesador definidas en el ISA.

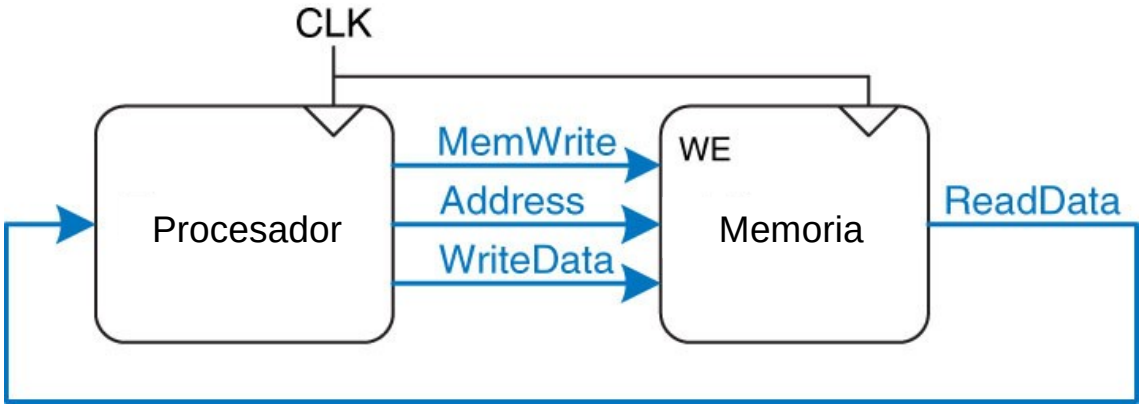
Ejecución de un programa

Los ***registros y la memoria son los únicas formas de almacenamiento*** a la que el procesador puede acceder directamente para ***ejecutar un programa***.

Sin embargo, puede almacenar datos y programas en perifericos.

La unidad de memoria sólo ve desde el procesador:

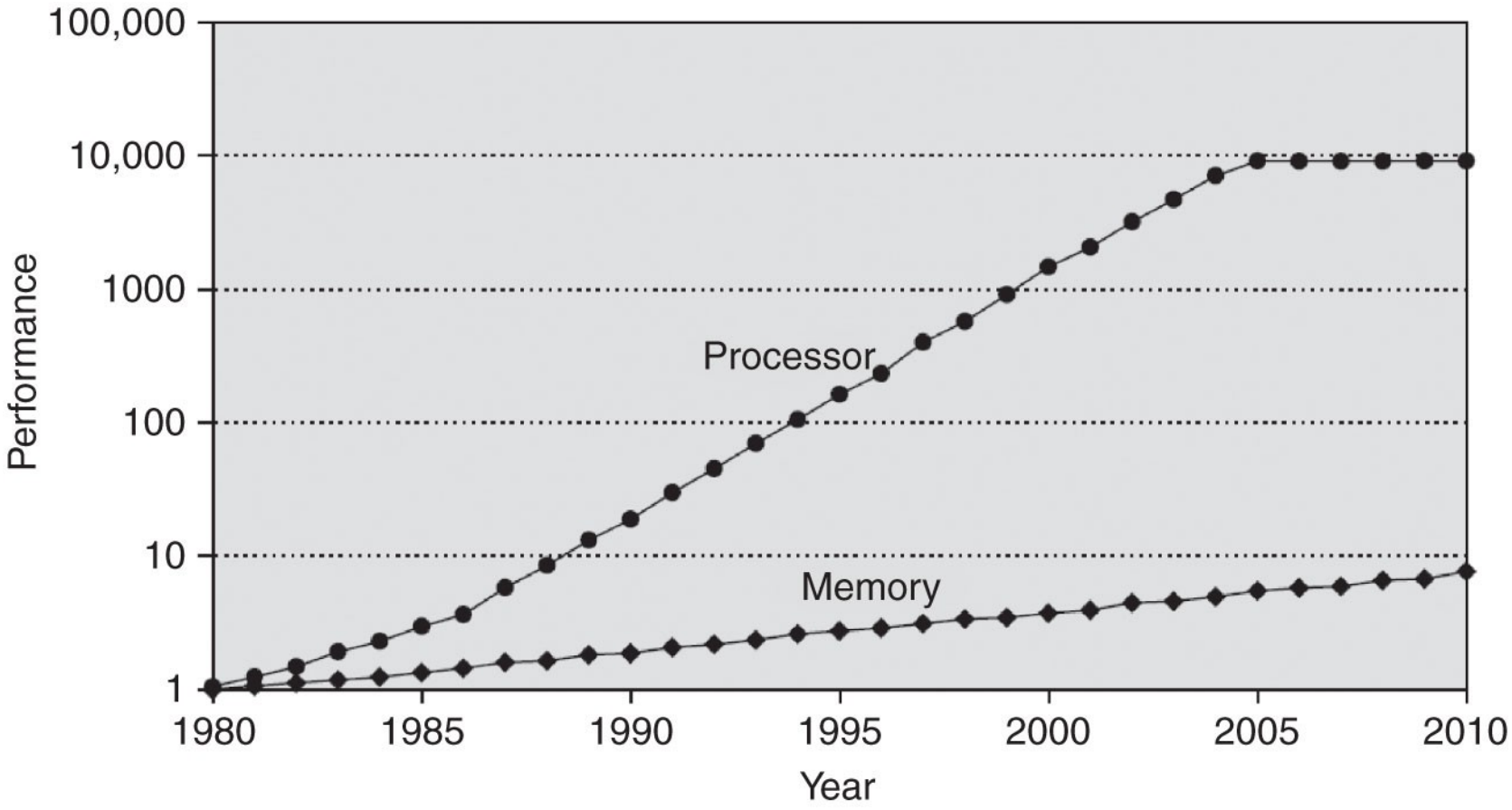
- Requerimentos de lectura mas las correspondientes direcciones;
- Requerimientos de escritura mas las correspondientes direcciones y datos.



Por lo tanto, el ***desempeño de una computadora*** depende de los ***desempeños de su procesador y su sistema de memoria***

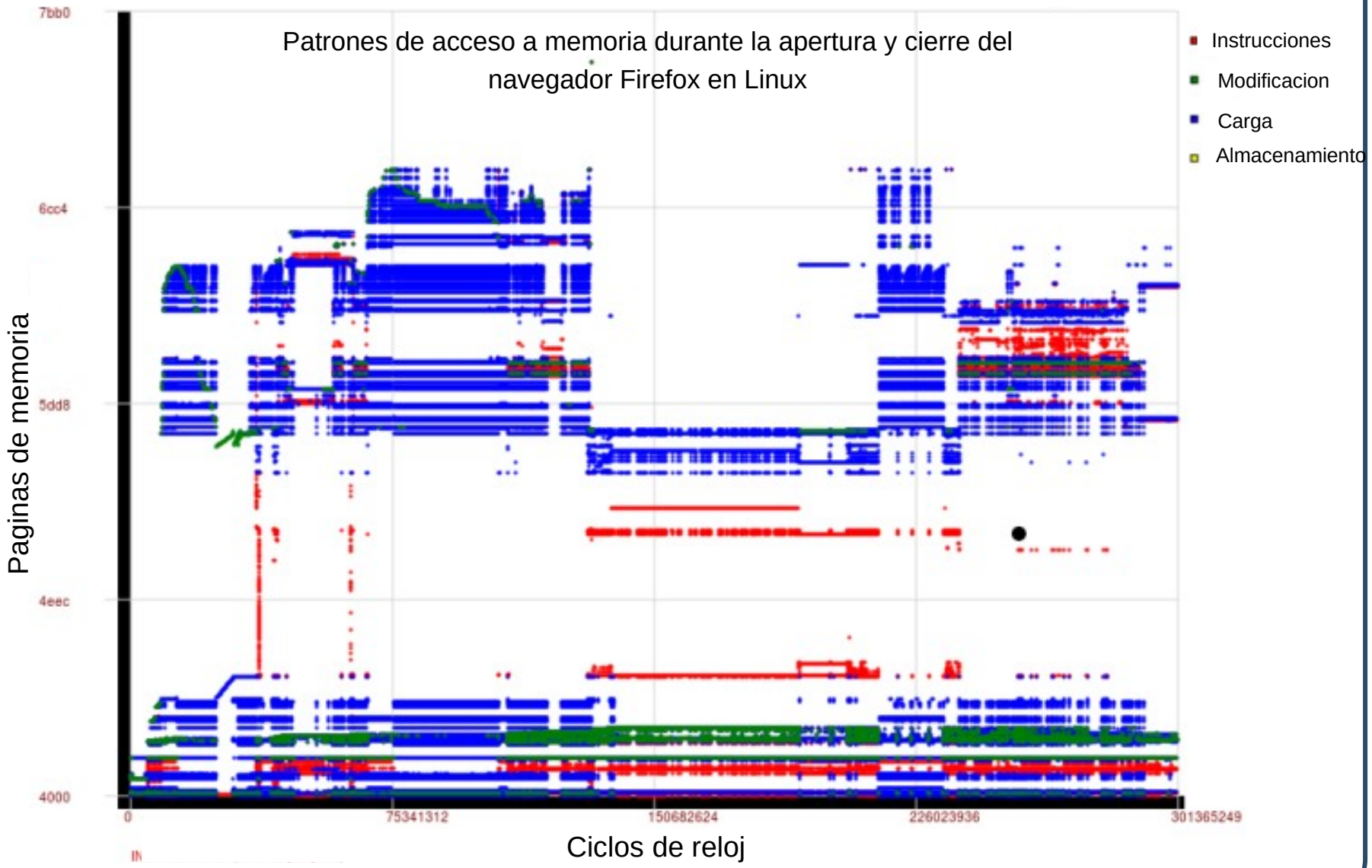
Instruction time = (CPU clock cycles + Memory-stall clock cycles) * Cycle time

Ejecución de un programa



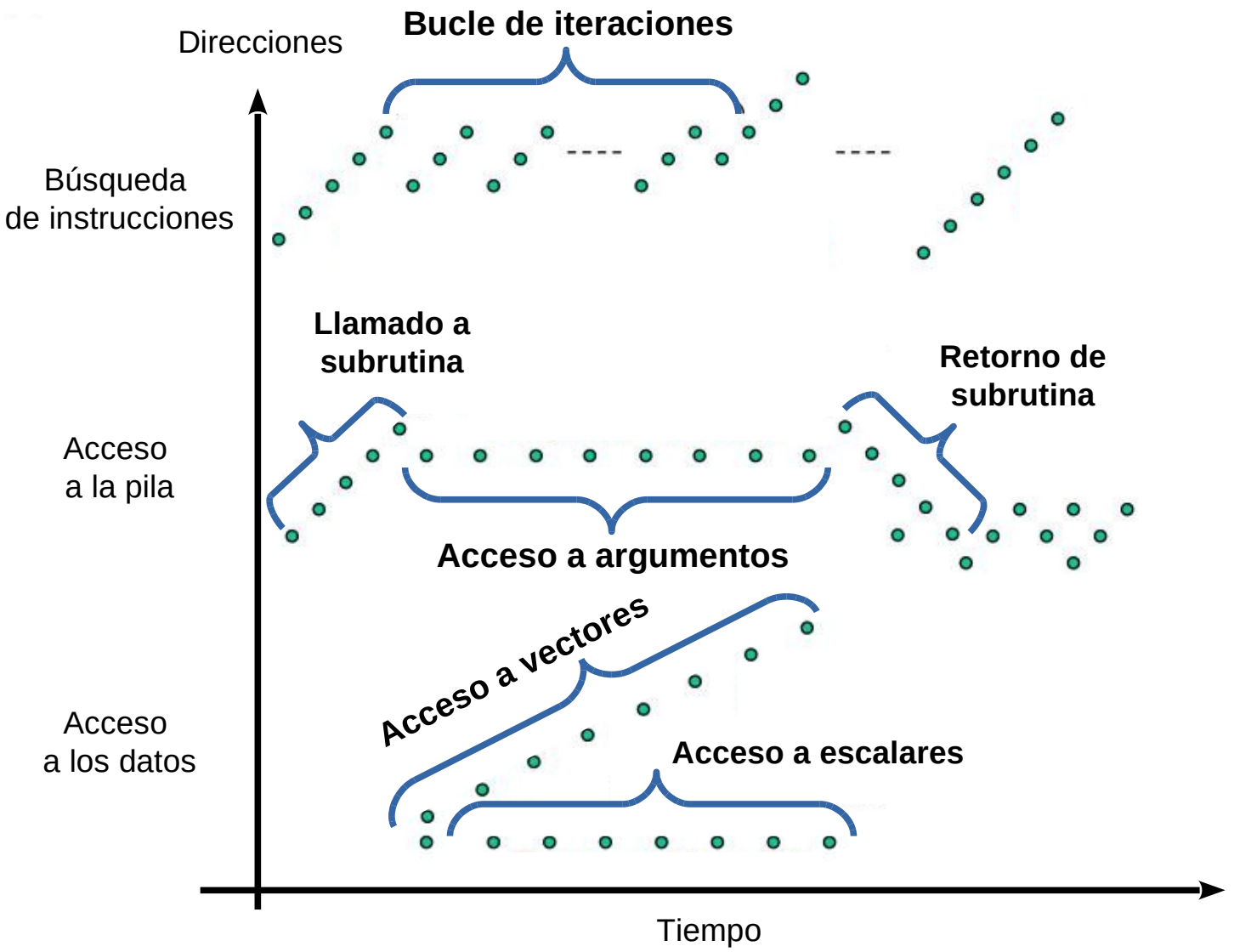
Ejecución de un programa

Patrones de acceso a memoria



Ejecución de un programa

Patrones de acceso de referencia



Ejecución de un programa

Para acelerar el acceso a la memoria por parte del procesador, se aprovecha la ***localidad*** de la información del programa (datos y código).

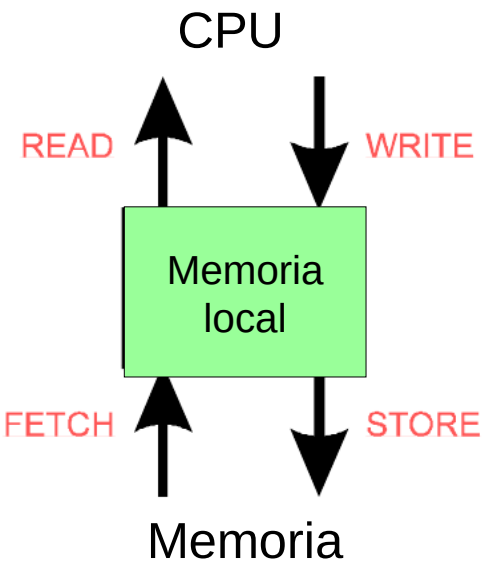
La localidad de la información se clasifica en

- **Localidad Temporal:** Si los datos han sido utilizados recientemente, es muy probable que vayan a ser utilizados de nuevo.

  Como se utiliza: Se mantienen los datos recientemer accedidos en las memorias más rápidas**.**

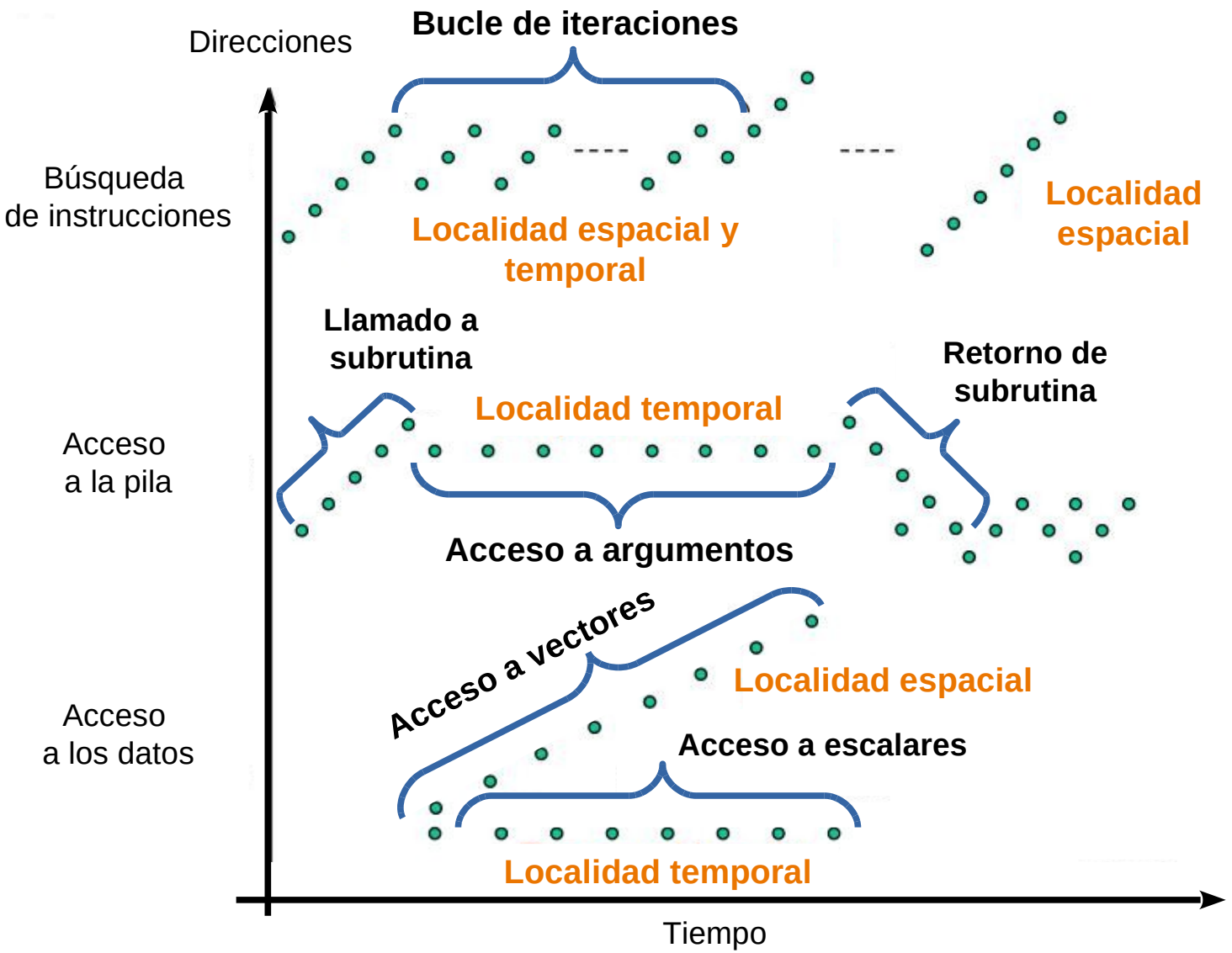
- **Localidad Espacial:** Si los datos han sido utilizados recientemente, es muy probable que se utilicen datos que se hallan en las direcciones próximas.

  Como se utiliza: Cuando se accede a un dato, cargue los datos próximos en la memoria más rápida**.**



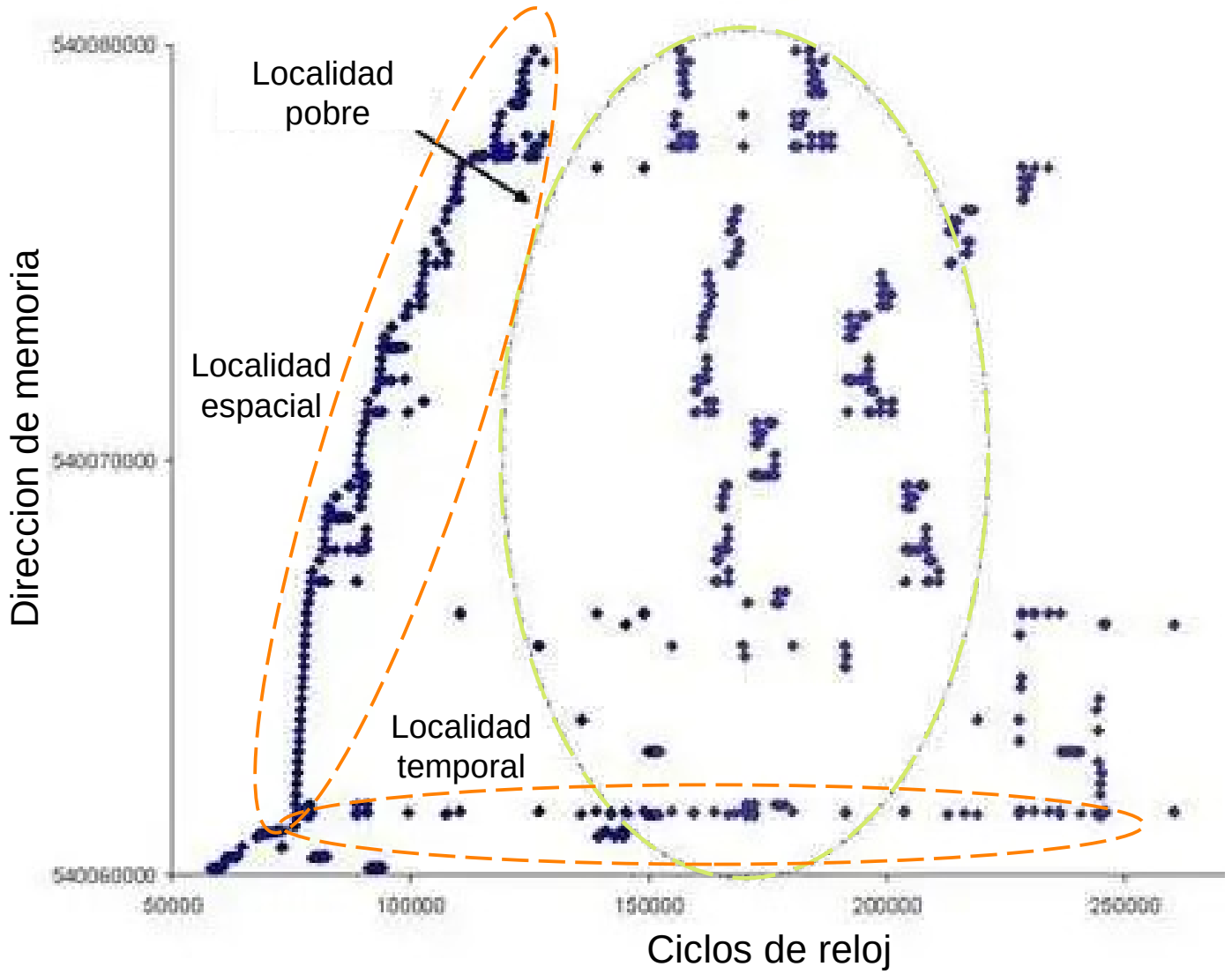
Ejecución de un programa

Patrones de acceso de referencia

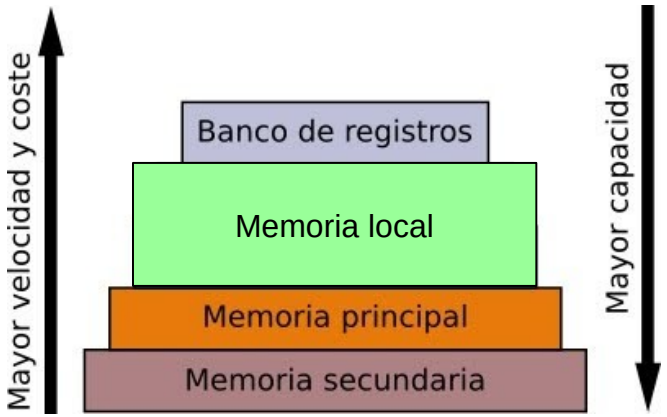


Ejecución de un programa

Localidad



Ejecución de un programa



Para mejorar el desempeño de una computadora y mantener su costo bajo el sistema de almacenamiento esta organizado en una jerarquia determinada por

- **Velocidad**
- **Costo**
- **Volatilidad**

Hay cuatro niveles de almacenamiento

- ***Interno*** – Esta constituido por los registros del procesador y memoria local, consiste en una pequeña cantidad de almacenamiento rápido, aunque algunos tienen funciones de hardware específicas. Este nivel se trata con mecanismos de direccionamiento distintos a la memoria principal;
- **Principal** – Esta constituida por la memoria del sistema y memorias locales. Consiste en circuitos integrados computadora que almacenan información para uso inmediato. Esta organizadaen palabras de longitud de palabra fija que agrupan celdas de memoria que almacena un bit;
- **Almacenamiento masivo on-line (*secundario*) –** Esta constituido por las unidades de disco duro, estado sólido y discos flash USB. No es accesible directamente por el procesador, el cual utiliza canales de entrada/salida para accederlos y transferir los datos deseados al almacenamiento primario. El almacenamiento masivo on-line retiene datos cuando se apaga la alimentación.
- **Almacenamiento masivo off-line (*terciario*) –** Esta constituida por unidades de almacenamiento masivos extraibles (cintas magneticas, discos ópticos). Se utiliza para archivar información a la que rara vez se accede, de modo que se utiliza para almacenar y/o transportar cantidades extraordinariamente grandes de datos.

Ejecución de un programa

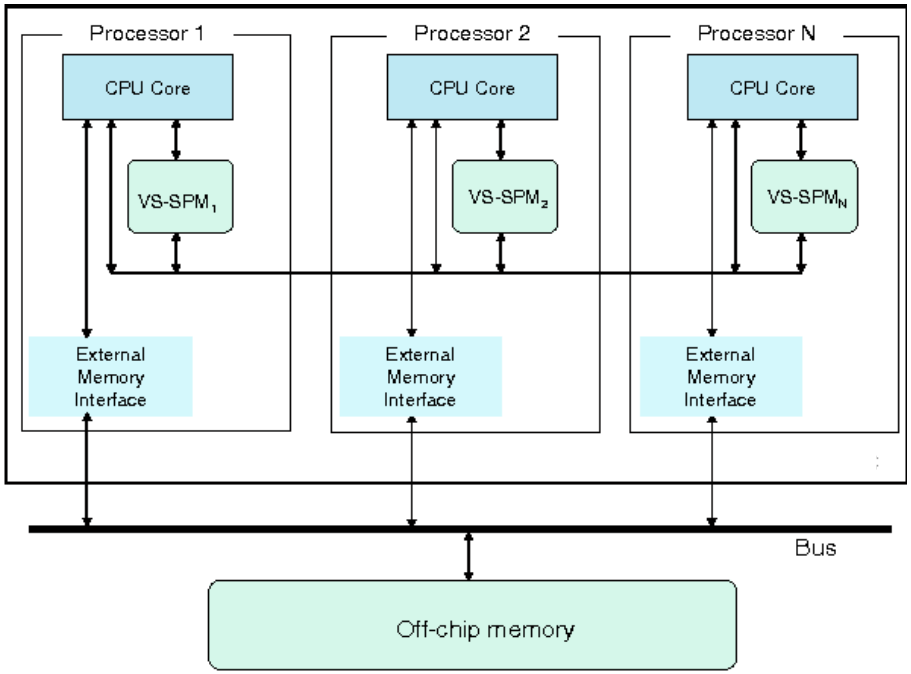
Memorias locales

La idea de localidad de la información se aprovecha, para mejorar la ejecución de los programas, a través de las siguientes estructuras de hardware

Memorias fuertemente acopadas – TCM – es una pequeña región de memoria dedicada que está muy cerca del procesador. Su principal ventja es que el procesador puede acceder a la TCM en cada ciclo, alcanzando el máximo desempeño posible y todos los accesos de memoria son predecibles.

Memorias scraptchpad – SPM – es una memoria interna de alta velocidad utilizada para el almacenar temporalmente datos que normalmente no necesitarían estar siempre comprometidos con la memoria principal.

Se emplea para simplificar la lógica de almacenamiento en caché y garantizar que un procesador pueda funcionar sin la contención de la memoria principal en sistemas multiprocesador.



Memorias cache – es un tipo de memoria extremadamente rápido que actúa como un búfer entre la RAM y la CPU. Contiene los datos e instrucciones solicitados con frecuencia para que estén inmediatamente disponibles para la CPU cuando sea necesario.

Memorias scraptchpad

Memoria Scratchpad

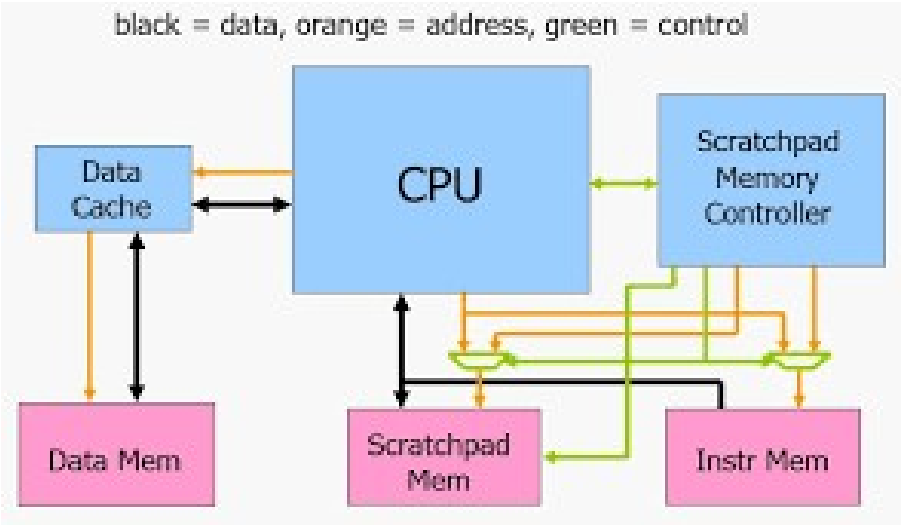
La ***memoria scratchpad*** es una memoria de alta velocidad utilizada para almacenar pequeños elementos de datos para una rápida recuperación. Se emplea para garantizar que un procesador pueda funcionar sin la contención de la memoria principal en sistemas multiprocesadores.

Son adecuados para almacenar resultados temporales que normalmente no necesitarían estar siempre comprometidos con la memoria principal. Sin embargo, cuando son alimentadas por DMA, pueden usarse en lugar de un caché.

Tiene los mismos problemas de localidad de referencia que las cache, aunque el uso de DMA para actualizarla permite el acceso simultaneo a conjuntos de datos. Otra diferencia es que las memorias scratchpad son manipulados explícitamente por las aplicaciones.

La SPM es un sistema con latencias de acceso de memoria no uniformes ya que las latencias de acceso de memoria a los diferentes scractchpad y la memoria principal varían.

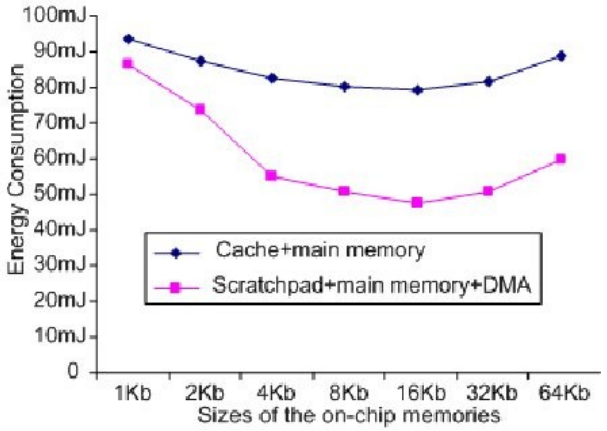
Otra diferencia con respecto a un sistema que emplea caches es que, en general, un scratchpad no contiene una copia de los datos que se almacenan en la memoria principal.

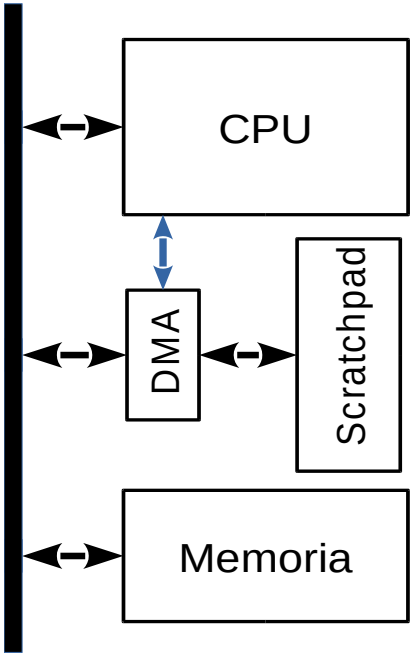


Memoria Scratchpad

La SPM se utilizan para acelerar la ejecución de los programas, acotar los tiempos de ejecución y reducir el consumo de energía, factores importantes en los dispositivos móviles.

Las instrucciones de carga y almacenamiento generan problemas si se utiliza memoria caché, porque los tiempos de acceso a la memoria (latencias) dependen del contenido de la memoria caché, el cualdepende de las referencias anteriores (***cadena de referencia***).





Las memorias scratchpad no se utilizan en los procesadores de escritorio, donde se requiere ***portabilidad*** (*legacy*) para que el software heredado se ejecute de generación en generación de computadoras, en el que el tamaño de la memoria en chip disponible puede cambiar.

Se implementa en sistemas integrados, procesadores especiales y consolas de juegos, donde los chips a menudo se fabrican como MPSoC, y donde el software a menudo se ajusta a una configuración de hardware.

Memoria fuertemente acoplada

Memoria fuertemente acoplada

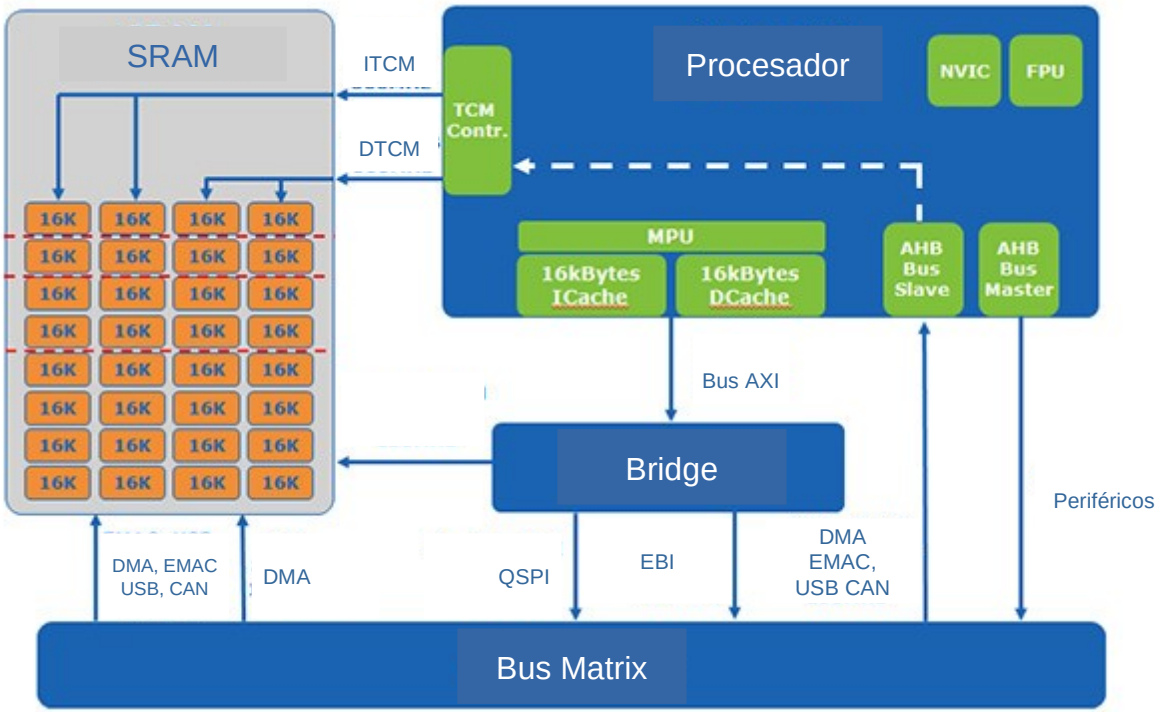
Una memoria fuertemente acoplada (TCM) es una pequeña memoria RAM (por lo general unos pocos kilo bytes) dentro del procesador administrado por el usuario a través del compilador o aplicacion.

Debido a que está embebido dentro del procesador, la TCM tiene una arquitectura de Harvard por lo que hay una TCM de instrucción (ITCM) y una TCM de datos (DTCM).

La DTCM no puede contener ninguna instrucción, pero la ITCM puede contener datos.

Los programas acceden a la TCM utilizando las instrucciones de carga y almacenamiento regulares.

Desde la perspectiva del software, no hay diferencia al acceder a una memoria estrechamente acoplada en comparación con otra memoria.



Memoria fuertemente acoplada

Los procesadores tienen registros especiales para leer el estado, la ubicación física y el tamaño de las memorias TCM.

Para gestionar la TCM, hay un registro de región que modifica el tamaño y ubicación en tiempo de ejecución. Además, los registros de la región también admiten la división de la TCM en bancos separados con sus propios registros de control.

La idea es poder bloquear y ocultar uno de los bancos para uso seguro (TrustZone).

Debe tenerse en cuenta que este registro no es una tabla de MMU, en realidad mueve la ubicación física del TCM. El lugar donde se la coloque la TCM, enmascarará cualquier memoria subyacente, por lo que generalmente es aconsejable no superponer ninguna memoria física con la TCM.

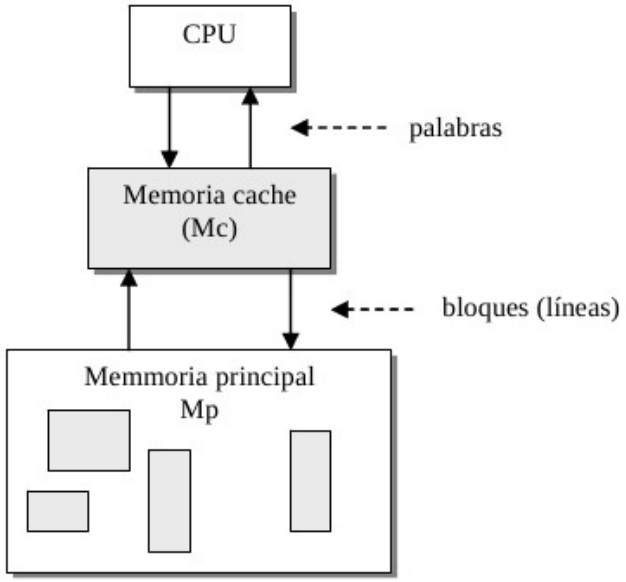
La memoria TCM puede también asignarse a otra dirección utilizando la MMU, pero debe observarse que la TCM se usa en situaciones en las que la MMU está desactivada.

La TCM se utiliza cuando:

- ***Rutinas de interrupciones*** que necesitan ***tiempos de ejecución determinista*** y no pueden esperar a los errores de cache;
- ***Accesos a memorias RAM externas*** configuradas para en actualización automática; y
- Operaciones que impliquen ***apagar o reconfigurar el controlador*** de memoria externa.

Memoria cache

Memoria cache



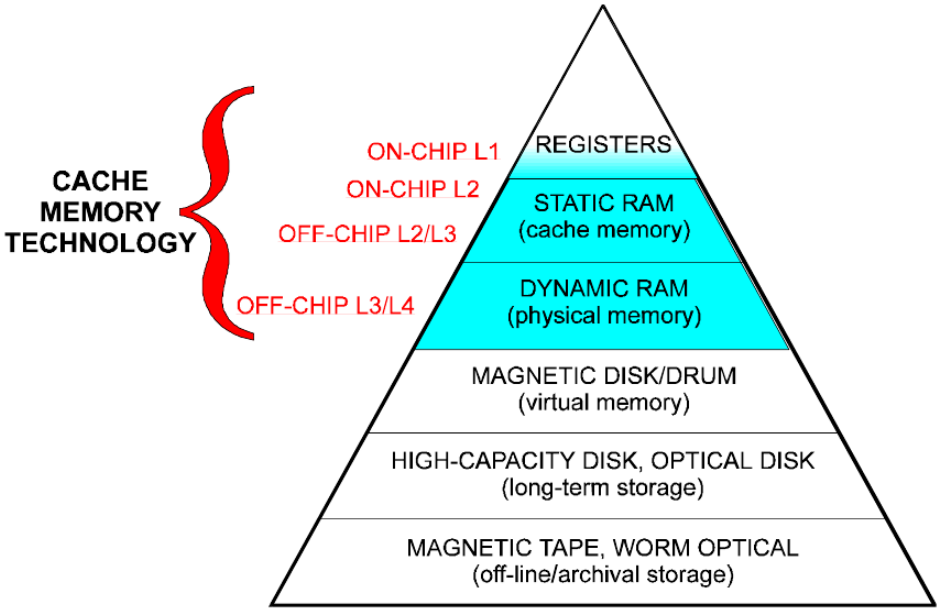
La memoria cache es el nivel mas alto en la jerarquia de la memoria, utilizada reducir el costo promedio, en tiempo y/o energía, para acceder a la información de la memoria principal.

Una caché es una memoria más pequeña y más rápida, cercana al núcleo del procesador, que almacena copias de los datos de las ubicaciones de la memoria principal de uso frecuente.

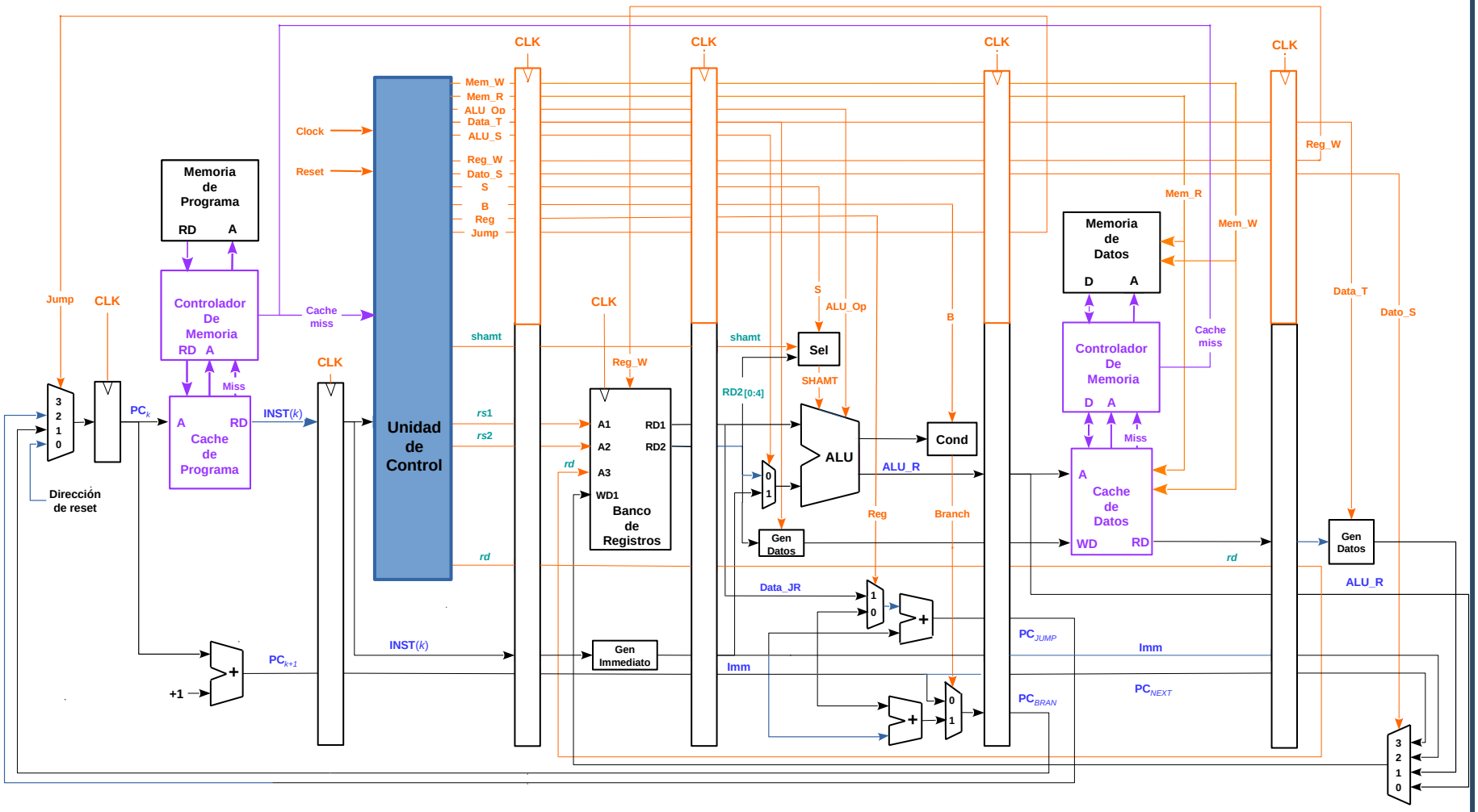
El rendimiento de la caché aumenta el desempeño global de la computador al reducir la brecha de velocidad entre el procesador y la memoria a partir del aprovechamiento de la localidad espacial y temporal de la información.

Estas transferencias necesitan definir:

- Ubicación de los bloques;
- Función de mapeo;
- Reemplazo de bloques;
- Politicas de escritura.



Memoria cache



Memoria cache

Los procesadores modernos tienen múltiples niveles de caches.

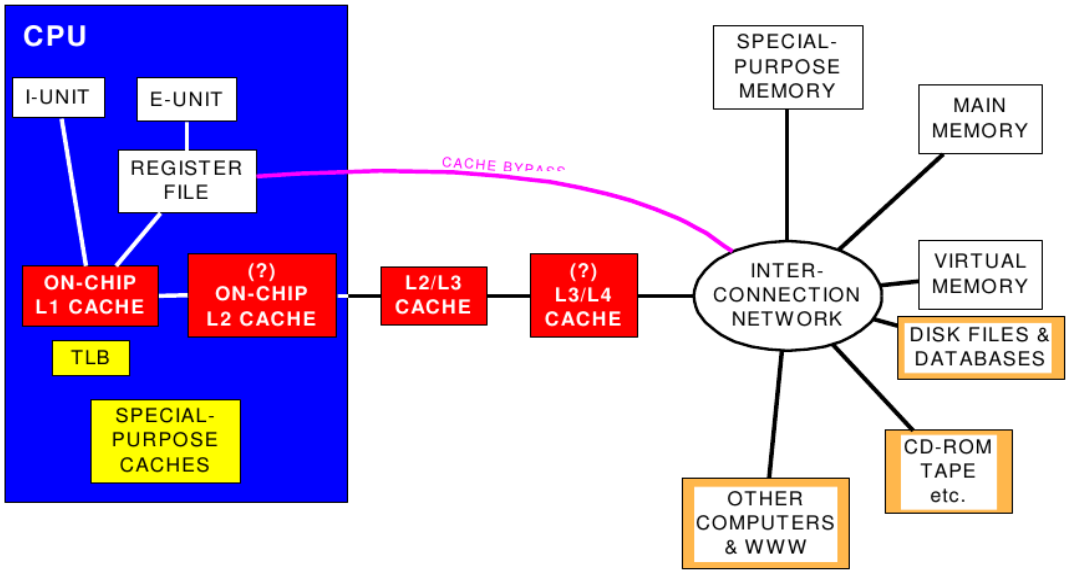
El ***primer nivel*** (L1) de caché se ubica ***cerca del núcleo y está dividida*** en un bloque para datos y otro para instrucciones. Es el más pequeño de todos (unos pocos kilo bytes)

El ***segundo nivel*** (L2) de caché se ubica ***dentro del procesador***, ***no se divide*** y actúa como un repositorio común para el primer nivel de caché. Cada núcleo de un procesador de multinúcleo tiene un caché de este tipo que no se comparte entre los núcleos.

El ***tercer nivel*** (L3) de caché se ubica ***dentro del procesador***, ***no se divide*** y ***se comparte*** entre los núcleos. Es la más grande (varios mega bytes).

El ***cuarto nivel*** (L4) de caché se ubica fuera del procesador y en la memoria principal.

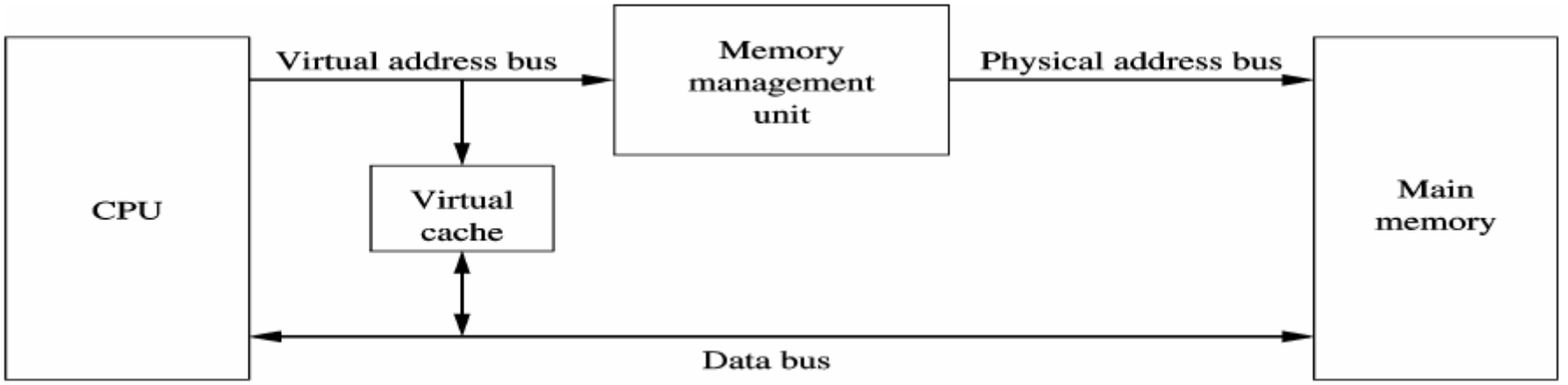
Cada nivel adicional de caché tiende a ser más grande y se optimiza de manera diferente.



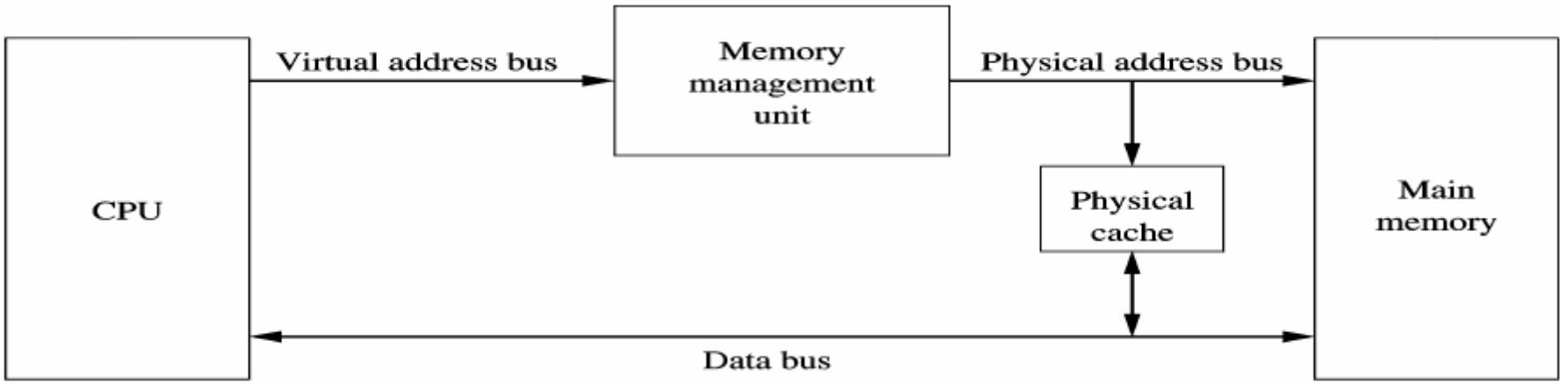
Existen otros tipos de cachés, como el búfer de traducción (TLB) que forma parte de la unidad de administración de memoria.

Memoria cache

Formas de implementación



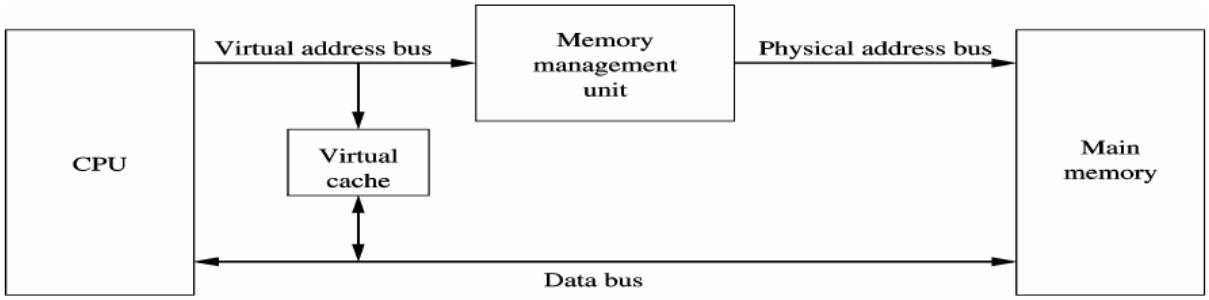
Cache virtual



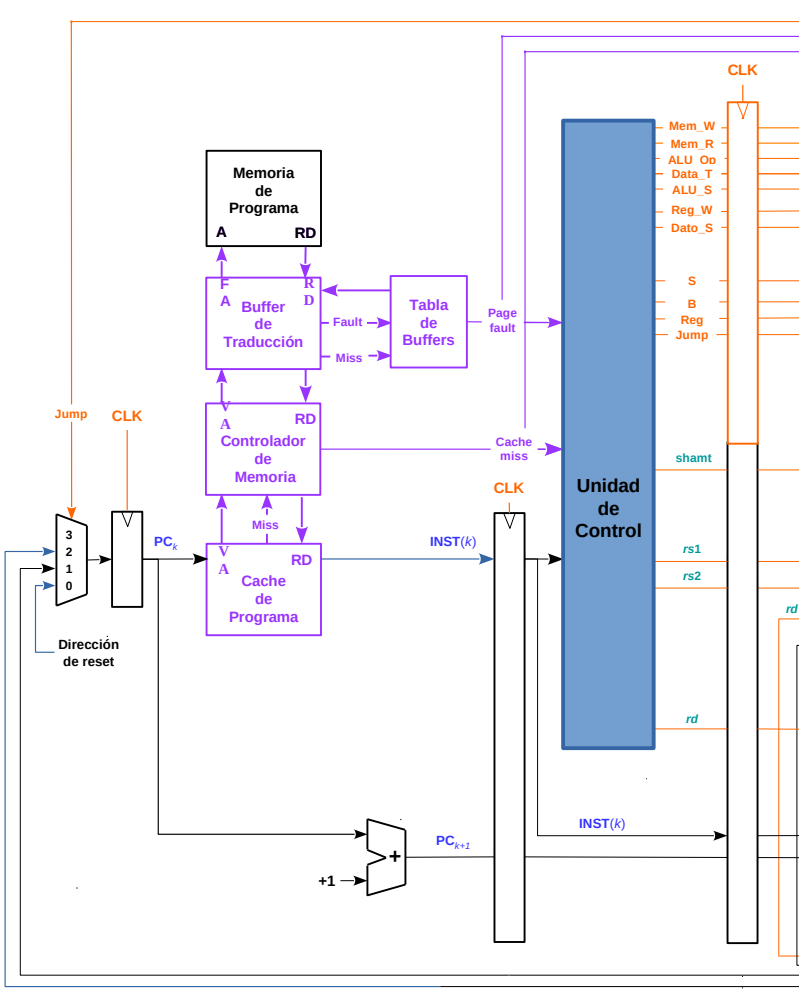
Cache física

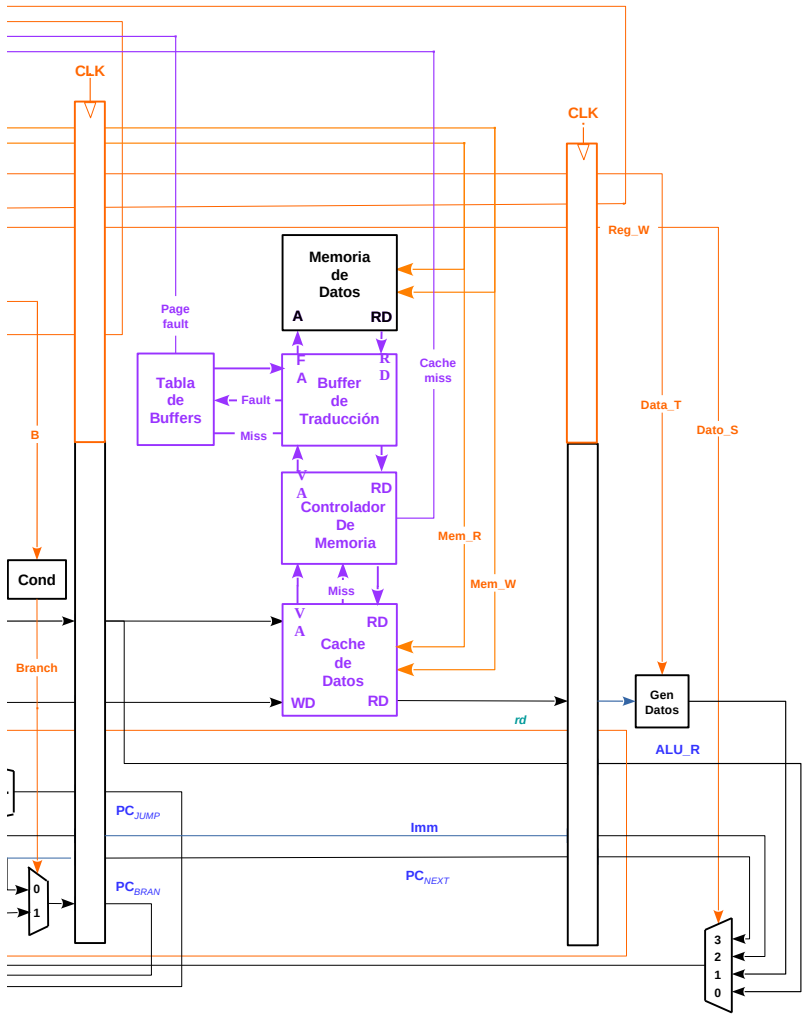
Memoria cache

Implementación con direcciones virtuales



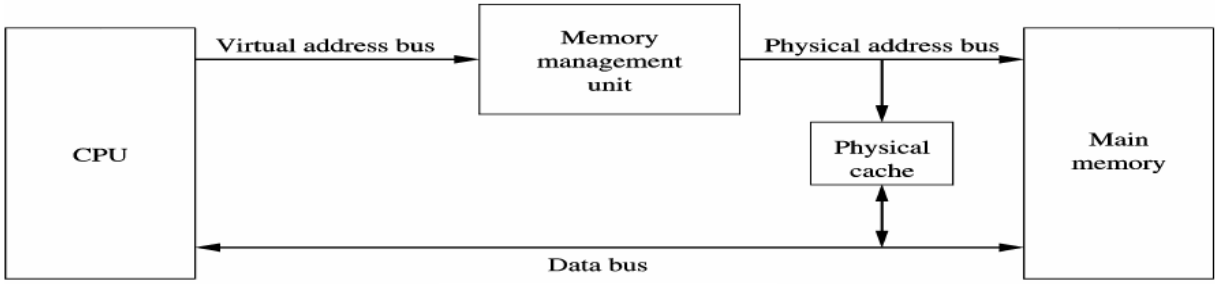
Cache virtual



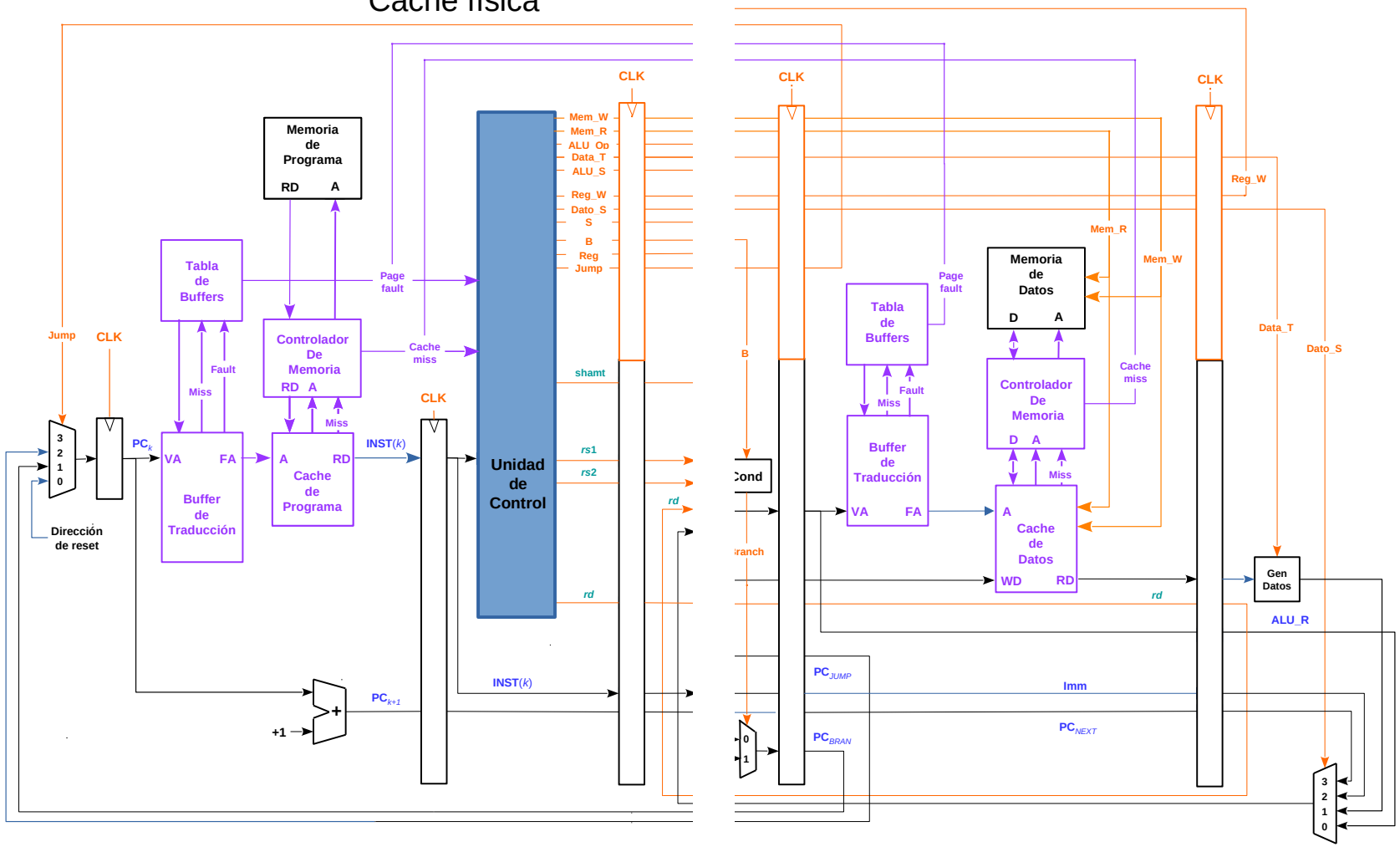


Memoria cache

Implementación con direcciones físicas



Cache física



Memoria cache

Los procesadores de ejecución segmentada acceden a la memoria desde diferentes etapas, ***utilizando diferentes caches*** para ***cada una de las etapas***, de modo que no se deba planificar el uso de los recursos compartidos para dar servicios a la microarquitectura.

Por lo tanto, la microarquitectura incluye cachés especializadas para cada parte del ciclo de instrucción

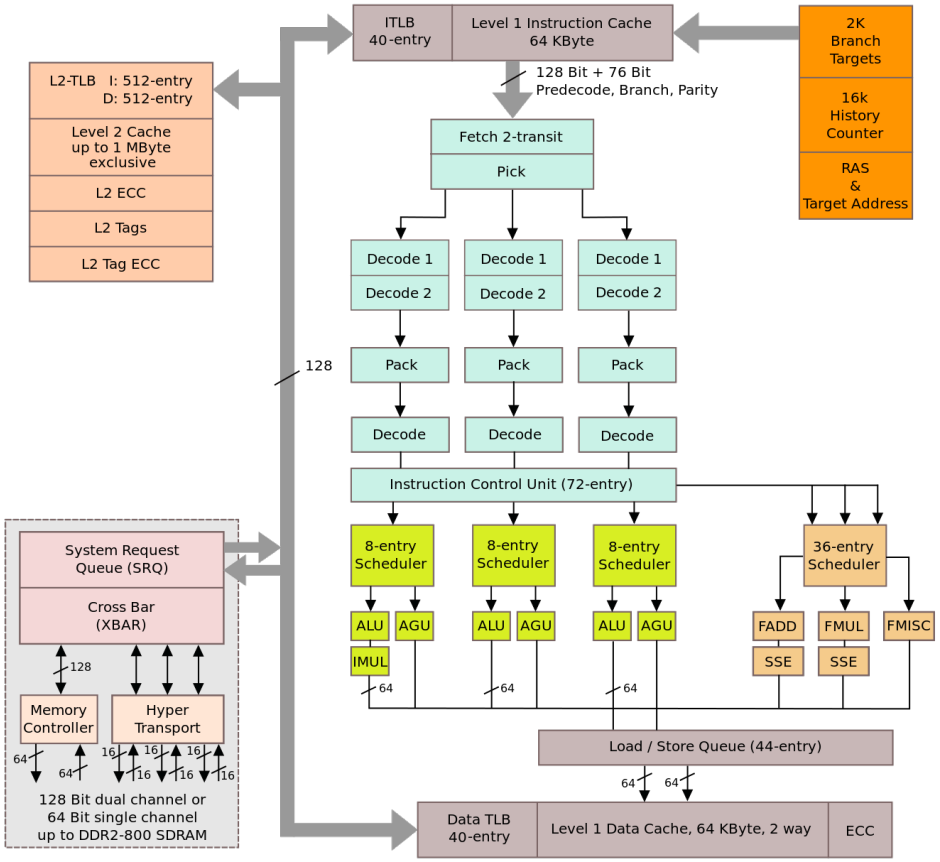
Victim cache – retiene los bloques de datos desalojados de la cache principal cuando se los reemplaza. Se encuentra entre el cache principal y su ruta de llenado, está destinado a reducir la cantidad de fallos por conflicto.

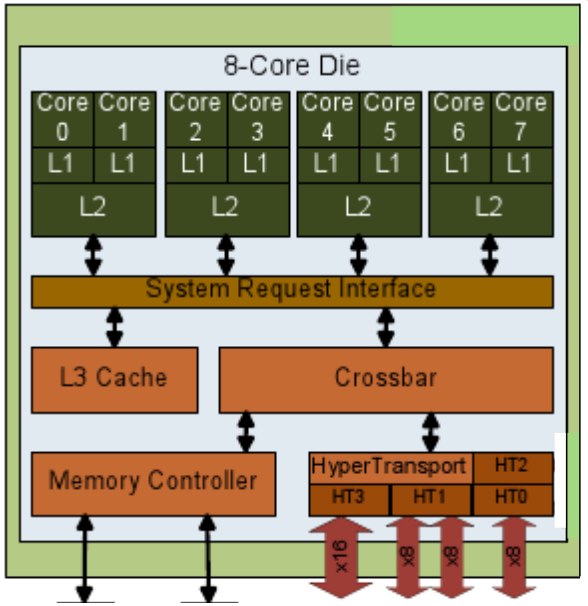
Trace cache - almacena las instrucciones después de que se han descodificado o retirado. Las instrucciones se agregan a la trace cache en grupos que representan bloques. Almacenando esta información, la próxima vez que se necesite una instrucción no tiene que ser descodificada nuevamente. Se utiliza para aumentar el ancho de banda de la etapa de fetch y disminuir el consumo de energía.

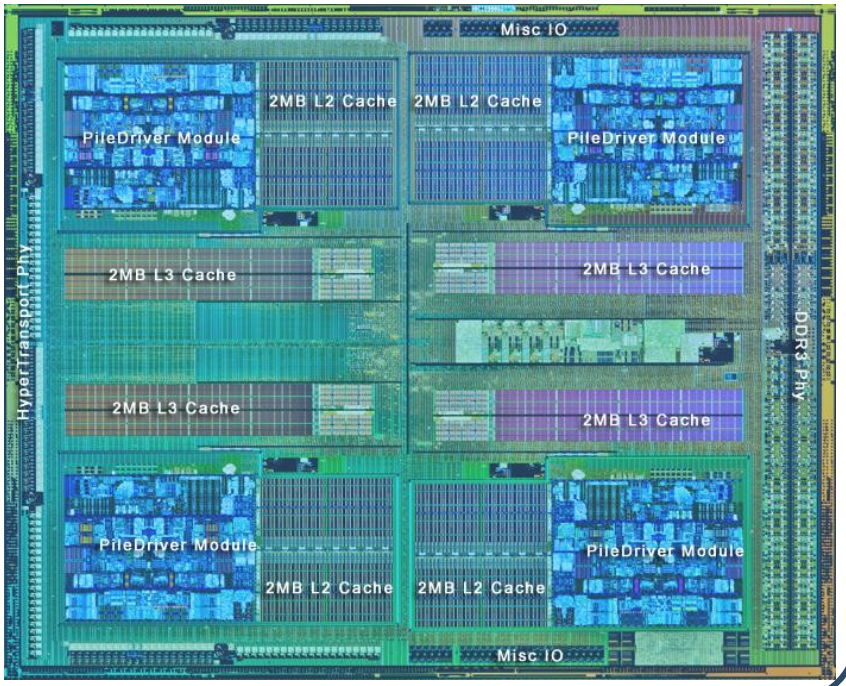
Micro-operation cache - almacena las microoperaciones de las instrucciones decodificadas, tal como se reciben directamente del decodificador de instrucciones. Cuando una instrucción necesita ser descodificada, esta caché comprueba la forma decodificada de la instrucción y la reutiliza si está almacena.

Branch target cache - almacena las primeras instrucciones del destino de los saltos ejecutados. La usan los procesadores de baja potencia que no necesitan una cache. Esto permite la operación a toda velocidad con un caché mucho más pequeño que un caché de instrucciones a tiempo completo tradicional.

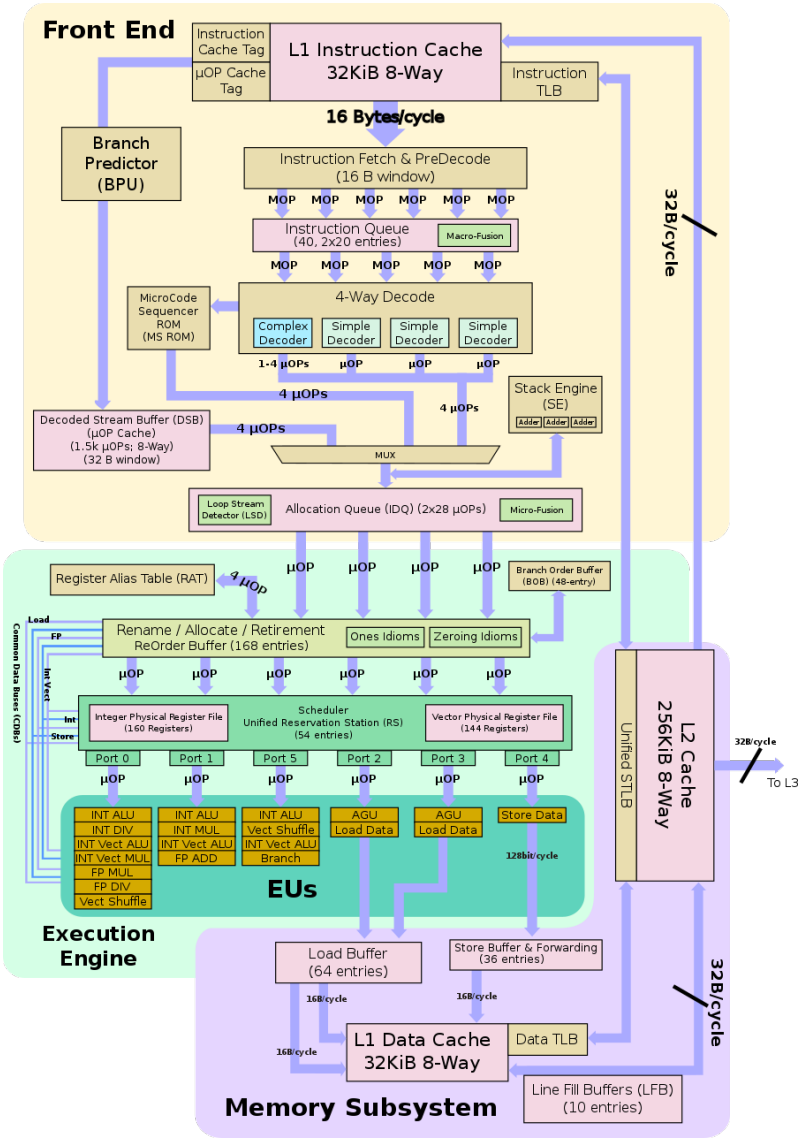
Memoria cache - *AMD Opteron*

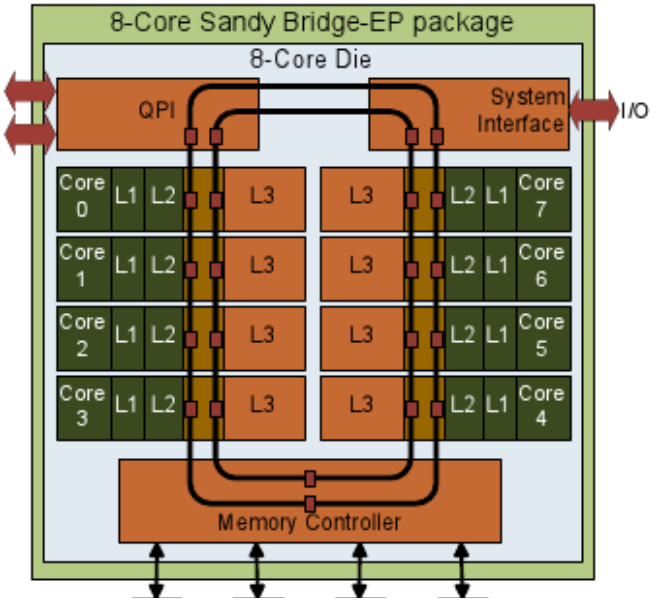


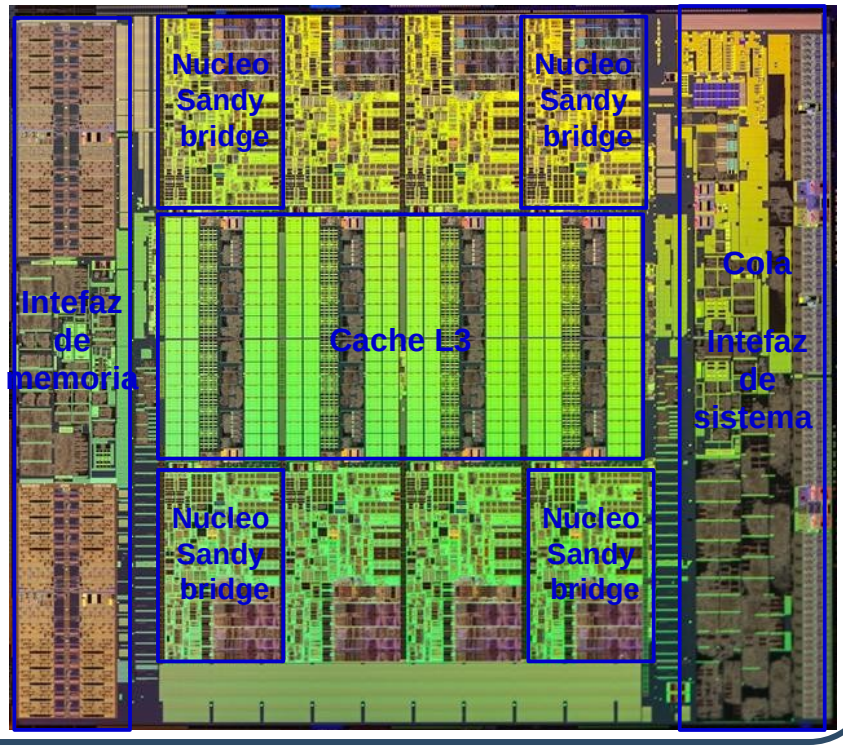




Memoria cache - *Intel Xeon*







Memoria cache

Terminologia

Algunas definiciones asociadas a las memorias cache:

- **Hit:** es un acceso a la memoria a cache en el que **se encuentra** el dato buscado;
- **Tiempo de acceso de Hit:** es el numero de ciclos de reloj necesarios para que la memoria cache devuelva un dato;
- **Miss:** es un acceso a la memoria a cache en el que **no se encuentra** el dato buscado, frzando el acceso al proximo nivel de la jerarquia de la memoria;
- **Porcentaje de Miss:** es el numero de accesos fallidos a la memoria cache comparado con el numero total de accesos;
- **Penalidad de Miss:** es el numero de ciclos de reloj necesarios para procesar un cache miss;
- **Relacion de trafico:** es la cantidad de informacion transferida a traves el bus a la cache comparado con la cantidad de informacion transferida desde la cache a la CPU.

Memoria cache

Organización

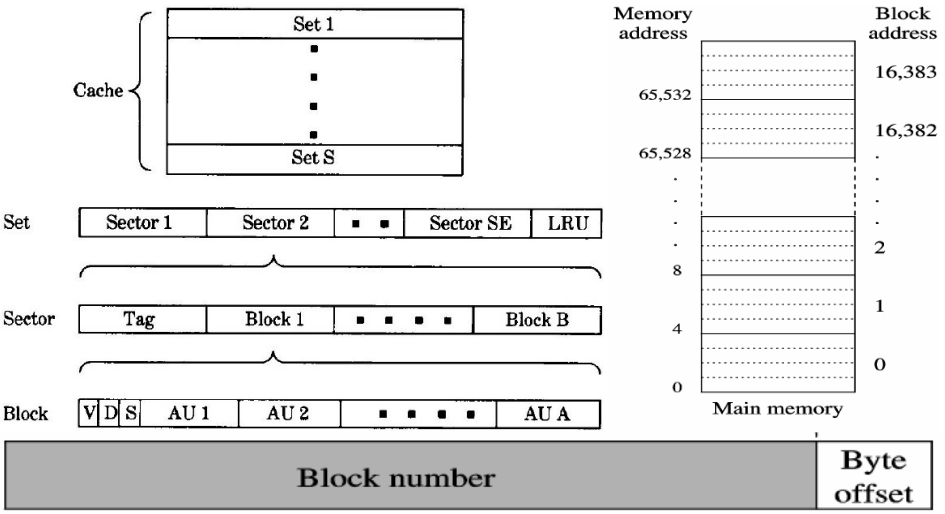
Desde el punto de vista del hardware, la cache se implementa como un bloque de memoria para el almacenamiento temporal de datos organizada como un conjunto de entradas que tienen datos asociados a través de una etiqueta.

La memoria se organiza en *S* **conjuntos**, los cuales mapean a uno o mas bloques de memoria;

El **tamaño de bloque** *b* es la cantidad de bytes que se pueden transferir a la cache en una operación;

El **numero de bloques** *B* (*cache lines*) en que esta dividida la memoria cache y esta dado por

$$B = C/b;$$



El **grado de asociatividad** *N* es el numero de bloques que hay en un conjunto

$$N = B/S.$$

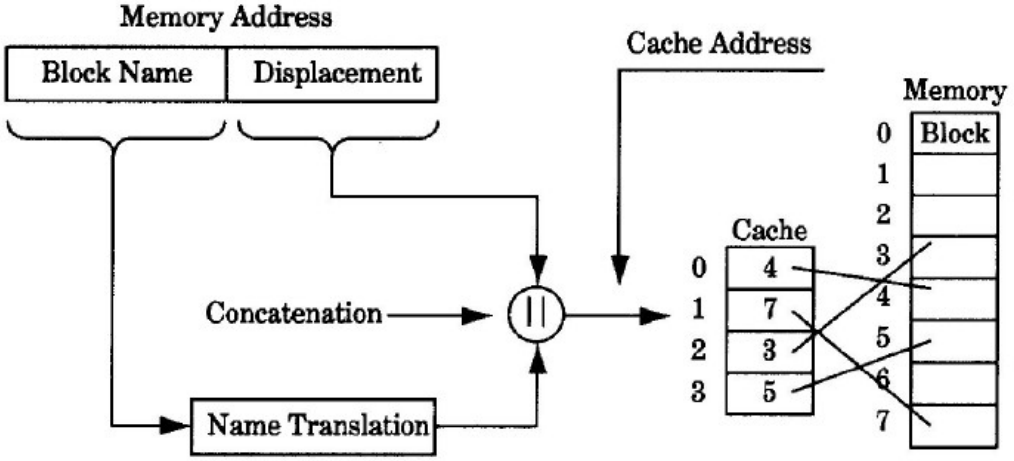
Los bits de control de cada bloque son:

- **Valido** (V): indica que el bloque contiene datos validos. Puede ser modificado por un programa o el protocolo de coherencia.
- **Corrompido** (D): indica que los datos fueron modificados.
- **Compartidos** (S): indica que los datos son compartidos en un esquema multiprocesador.

Memoria cache

Operación

La cache es direccionada utilizando direcciones parciales de memoria: los bits superiores determinan la correspondencia entre cache y bloques de memoria



El calculo de la dirección (address translation) puede incluir el calculo de la direccion fisica a partir de la virtual.

Hay tres características importantes de la traducción de direcciones:

Latencia - la dirección física está disponible desde la MMU en algún momento, quizás algunos ciclos, después de que la dirección virtual esté disponible desde el generador de direcciones.

Aliasing - varias direcciones virtuales se pueden asignar a una sola dirección física. La mayoría de los procesadores garantizan que todas las actualizaciones de esa dirección física única se realizarán en el orden del programa. Para cumplir con esa garantía, el procesador debe garantizar que solo una copia de una dirección física se encuentre en el caché en un momento dado.

Granularidad - el espacio de direcciones virtuales se divide en páginas.

Memoria cache

Operación

Las caches se pueden dividir en función del el índice o la etiqueta utilizados:

Indexado físico - etiquetado físico (PIPT) - usa direcciones físicas tanto para el índice como para la etiqueta. Si bien este esquema es simple resulta en una respuesta lenta, ya que se busca la dirección física antes que la dirección en la cache;

Indexado virtual – etiquetado virtual (VIVT) - usa direcciones virtuales tanto para el índice como para la etiqueta. Este esquema de almacenamiento puede resultados de búsquedas más rápidas, ya que no es necesario consultar a la MMU. Sin embargo, sufre de ***problemas de aliasing***, lo que causará problemas de coherencia, y ***problemas de homónimos***, la misma dirección virtual se asigna a varias direcciones físicas diferentes;

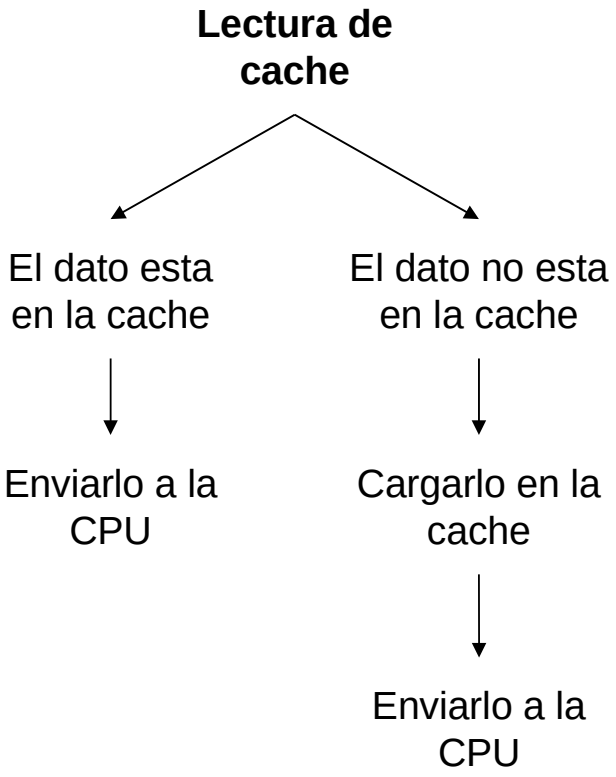
Indexado virtual – etiquetado físico (VIPT) - usa la dirección virtual para el índice y la dirección física en la etiqueta. Su ventaja sobre PIPT es una menor latencia, ya que la caché se puede buscar en paralelo con la traducción de TLB, sin embargo la etiqueta no se puede comparar hasta que la dirección física esté disponible. Su ventaja sobre VIVT es que, dado que la etiqueta tiene la dirección física, la memoria caché puede detectar homónimos.

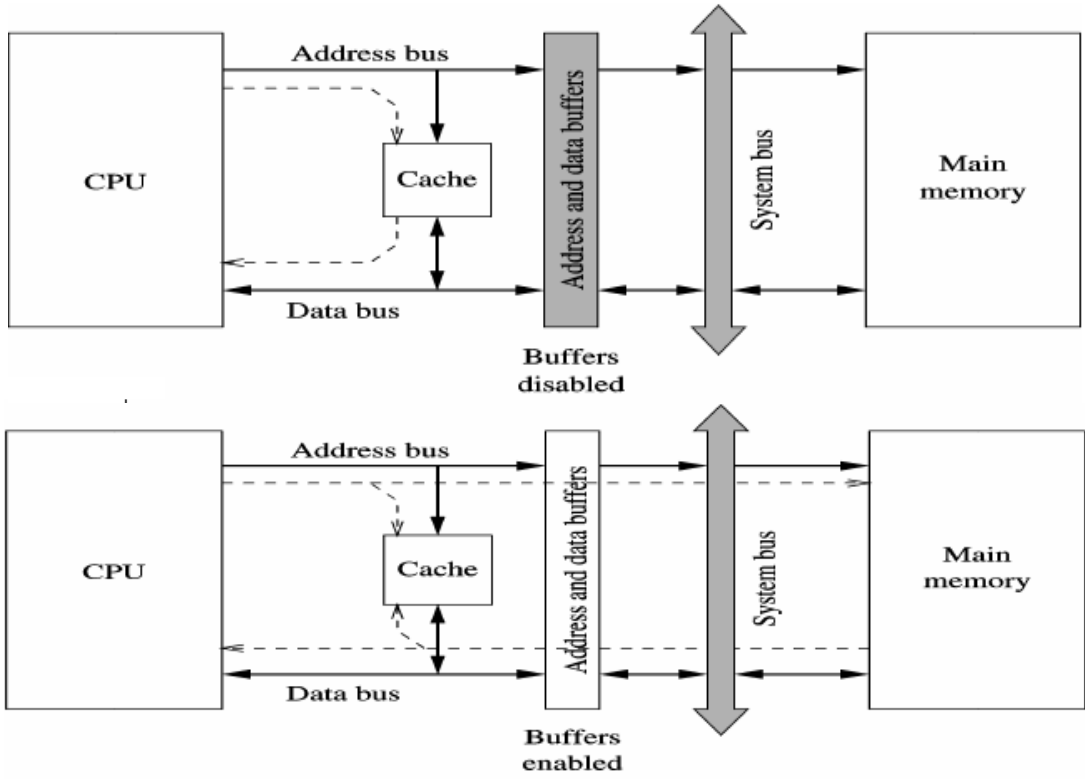
Indexo físico – etiquetado virtual (PIVT) - la cache se indexa por la dirección física obtenida del segmento TLB. Como el segmento TLB solo traduce los bits de dirección virtual que son necesarios para indexar la memoria caché y no utiliza ninguna etiqueta, pueden producirse falsos intentos de memoria caché.

Memoria cache

Operaciones - Lectura

La operación de lectura de un resultado en memoria requiere es la más simple de las operaciones ya que no imorta el origen del dato.





Memoria cache

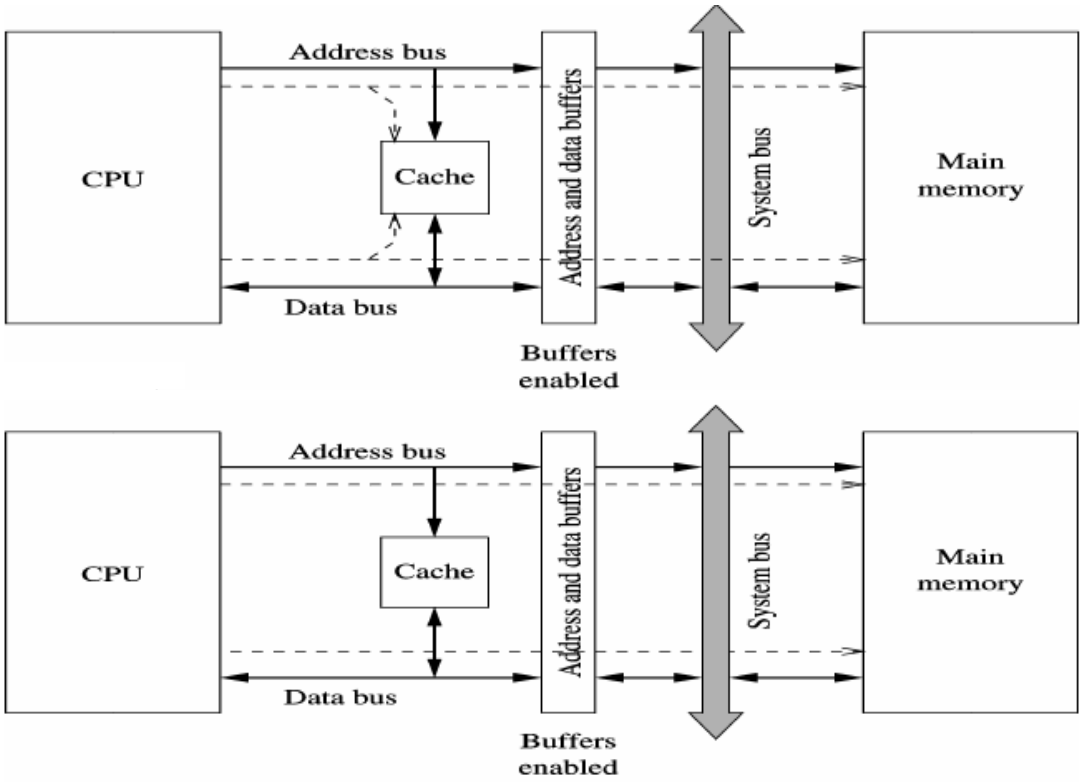
Operaciones - Escritura

La operación de escritura de un resultado en memoria requiere de una especial atención porque se deben realizar dos copias: *i*) una en la memoria y ii) otra en la cache.

La politica de escritura determina como se realiza la operación de escritura.

Escritura de cache

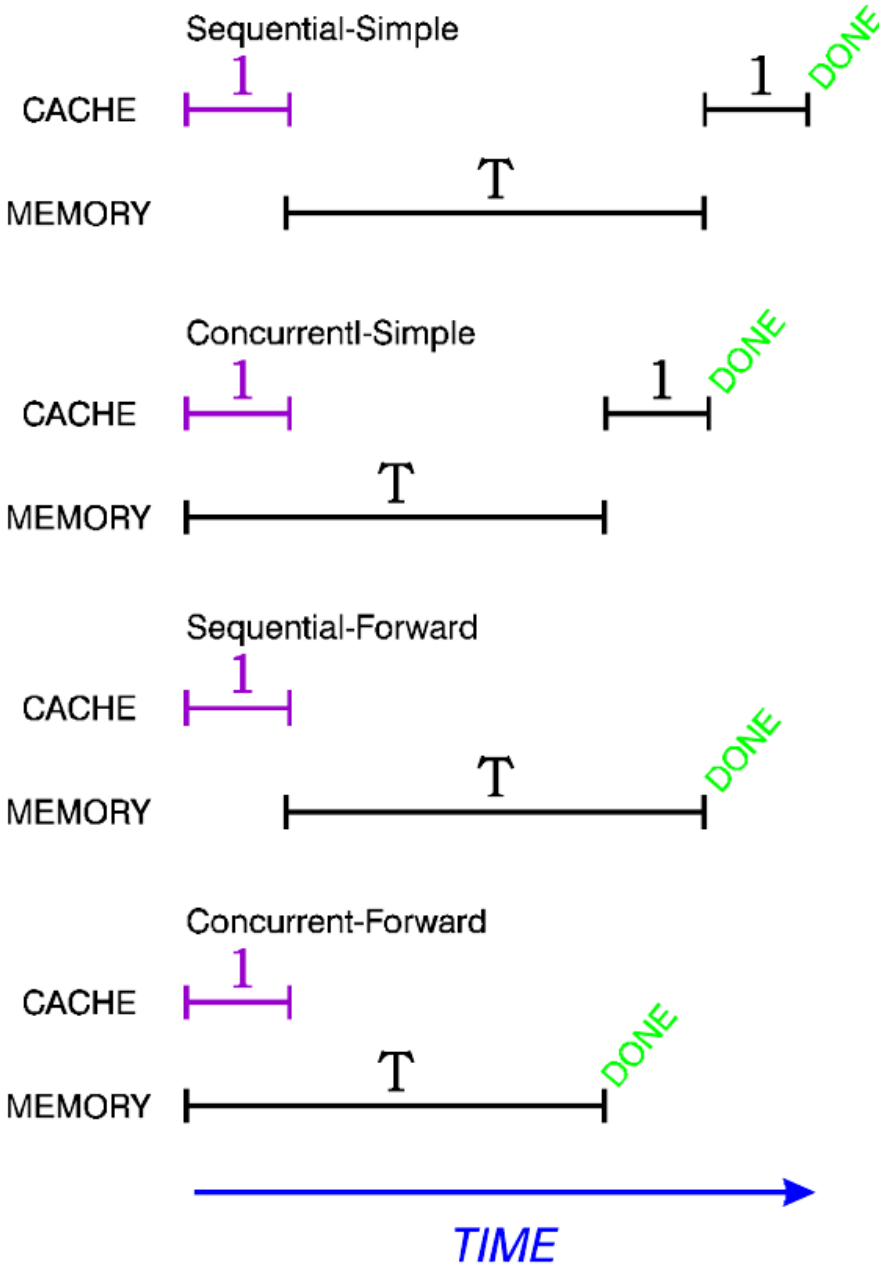
El dato esta en la cache	El dato no esta en la cache
Escribir cache y memoria principal	Cargarlo en la cache y actualizarlo
o	o
Escribir sólo cache	Escribir sólo memoria principal



Memoria cache

Operaciones - Estrategias de acceso

Para implementar estas operaciones, se utilizan las siguientes estrategias de acceso



Memoria cache

Organización – Mapeo de la memoria

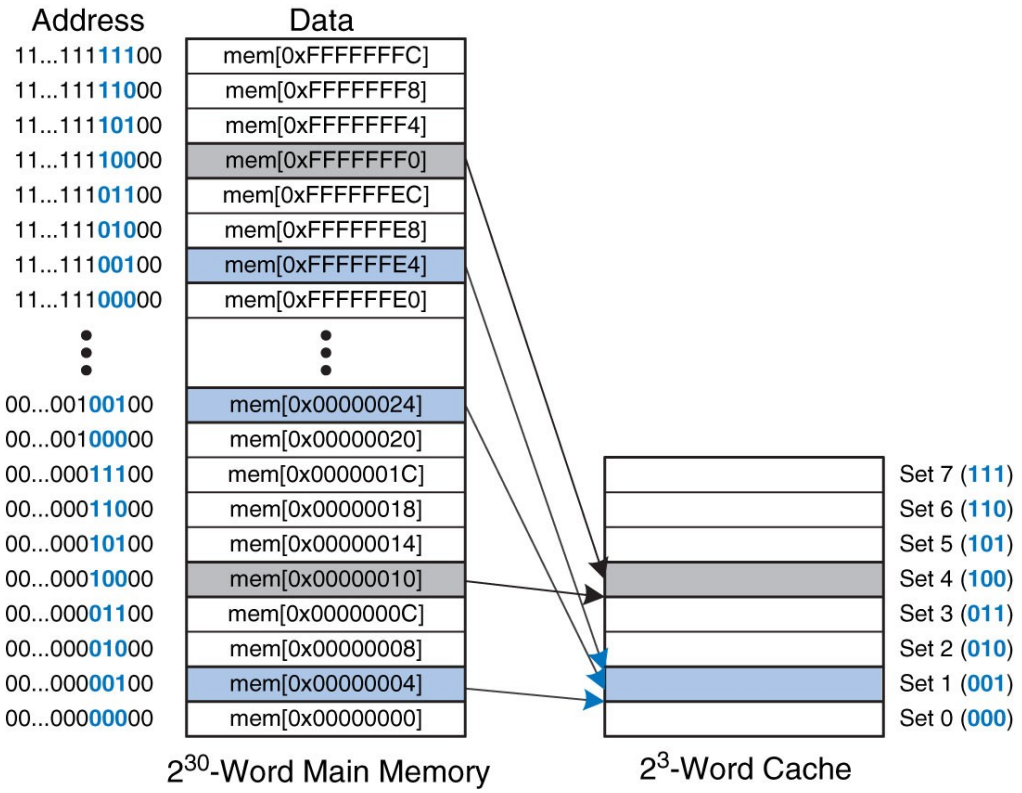
La función de mapeo determina cómo se asignan los bloques de la memoria principal a las líneas de caché.

Existen tres tipos de mapeos:

Mapeo directo - especifica una sola línea de caché para cada bloque de memoria;

Mapeo asociativo - especifica un conjunto de líneas de caché para cada bloque de memoria;

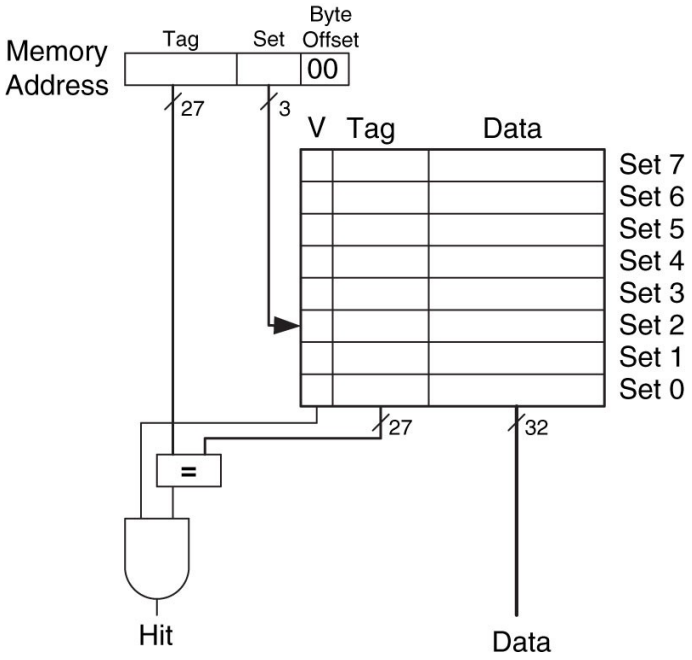
Mapeo asociativo completo - todos los bloques de la cache estan en un solo conjunto.



Memoria cache

Operación – Mapeo directo

La memoria cache por mapeo directo contienen un bloque de datos por cada conjunto.



La implementación del mapeo directo es la más sencilla de todas.

Solo necesita mantener tres datos en memoria

Datos de cache – la información actual

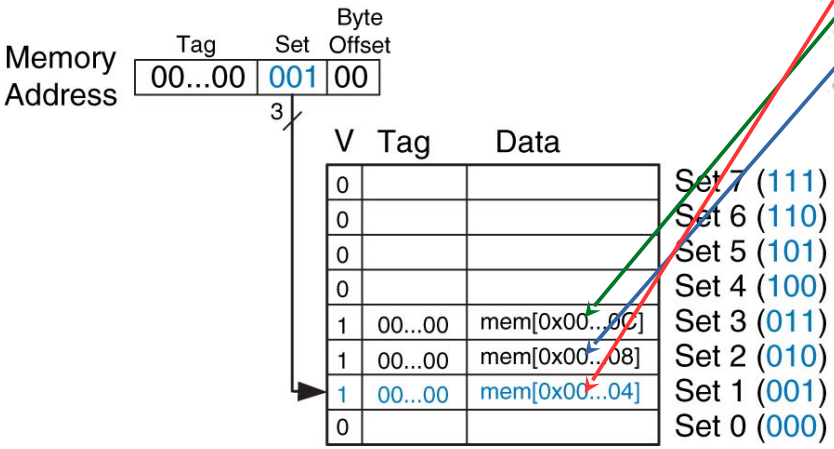
Etiqueta de cache – el bloque de datos de memoria;

Estado del bloque – indica si la línea de caché contiene un bloque válido.

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        t0, $t0, -1
        oop
```

El problema de este enfoque es que hay ***más bloques de memoria que líneas de cache***.

Una de las ventajas de un caché mapeado directo es que permite una especulación simple y rápida.



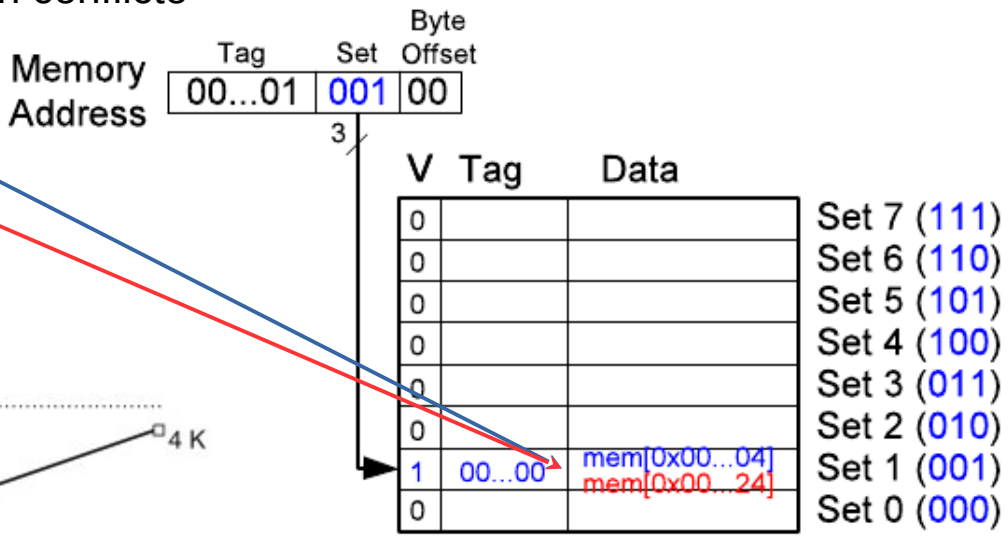
Memoria cache

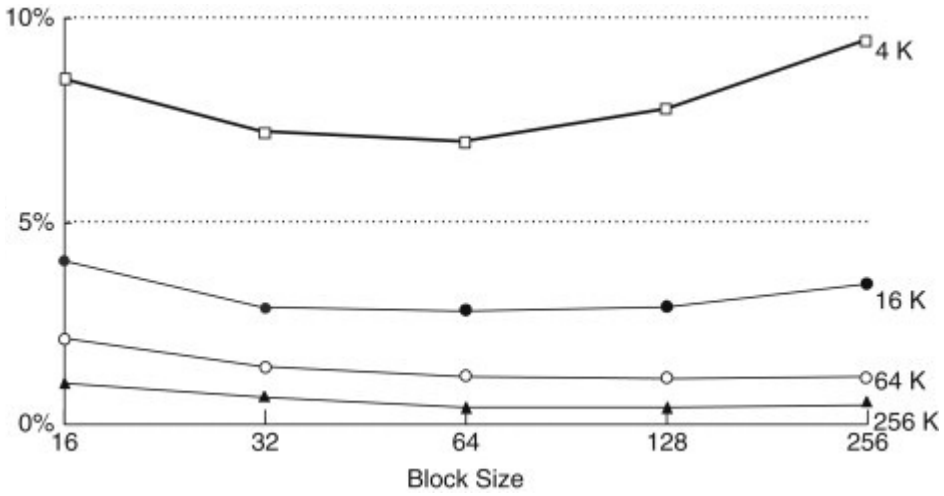
Operación – Mapeo directo

Cuando ***dos o más datos comparten*** el mismo conjunto, entonces la etiqueta que almacena la dirección del bloque de memoria en la línea de caché.

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```

Si el bloque no tiene el ***tamaño adecuado*** se produce un conflicto

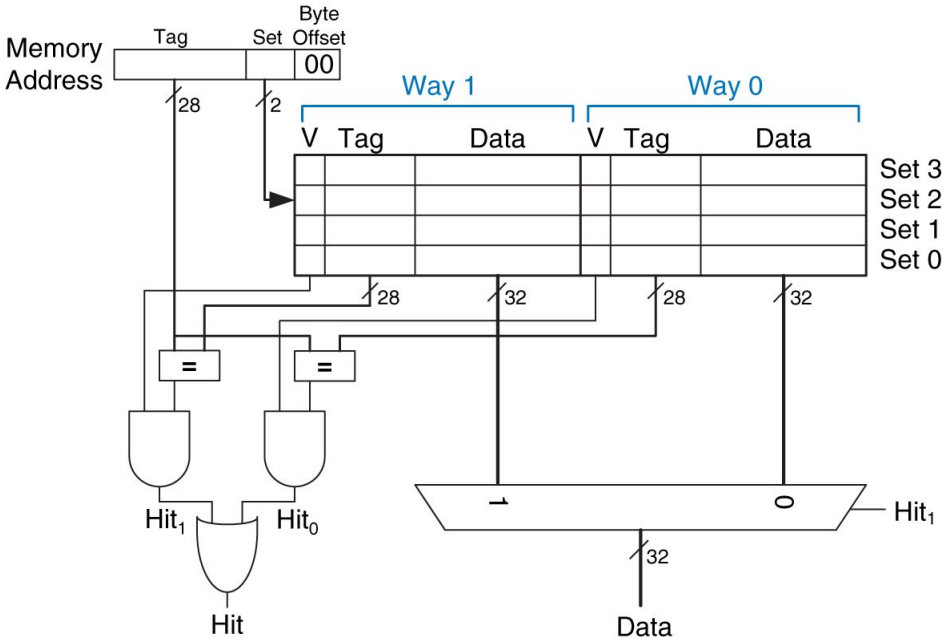


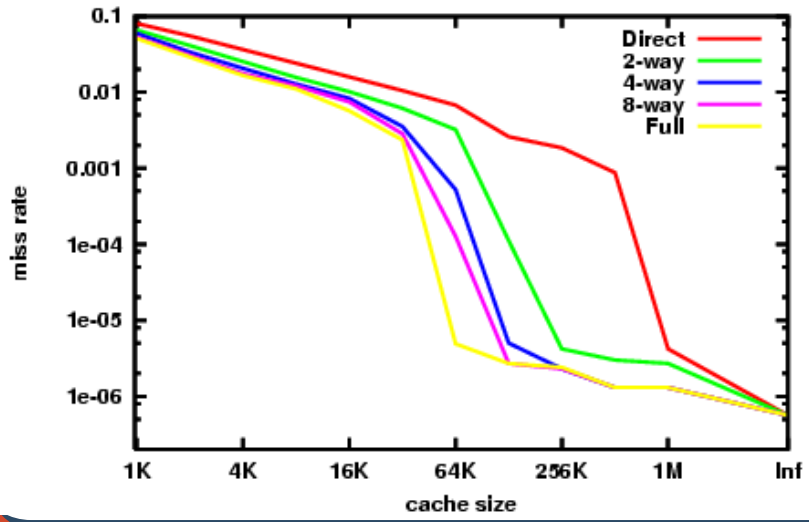


Las memorias caches por mapeo directo tienen problemas con ***la localidad temporal*** y las ***fallas obligadas***, las cuales se producen cuando el sistema empieza a funcionar y no hay información del mismo

Memoria cache

Operación – Mapeo asociativo

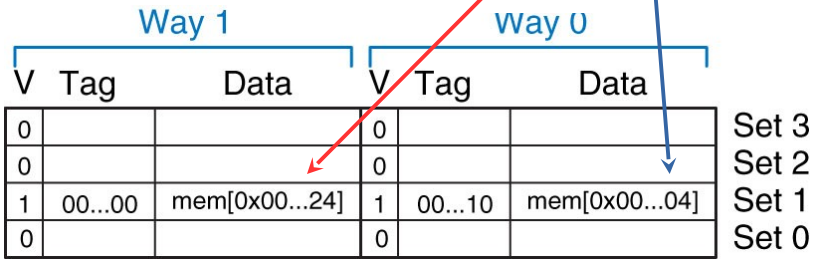




La memoria cache por mapeo asociativo contienen una entrada (o vía) por cada bloque de datos en cada conjunto, permitendo tener ***localidad espacial***.

Las etiquetas almacenadas en el cache no tienen que incluir esa parte de la dirección de memoria principal que está implícita en el índice de la memoria.

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Un memoría cache asociativa ***prueba todas las vias simultáneamente***, utilizando algo una arquitectura de hardware similar a una memoria de contenido direccionable.

Memoria cache

Operación - Desempeño

Un error de caché es un intento fallido de leer o escribir una parte de los datos en el caché, lo que resulta en un acceso a la memoria principal con una latencia mucho más larga. Las causas de los fallos permiten clasificarlos como

Compulsory - en el primer acceso a un bloque, este debe ser llevado a la cache. A este fallo se lo llama ***fallo de arranque en frío*** o ***fallo de primera referencia***;

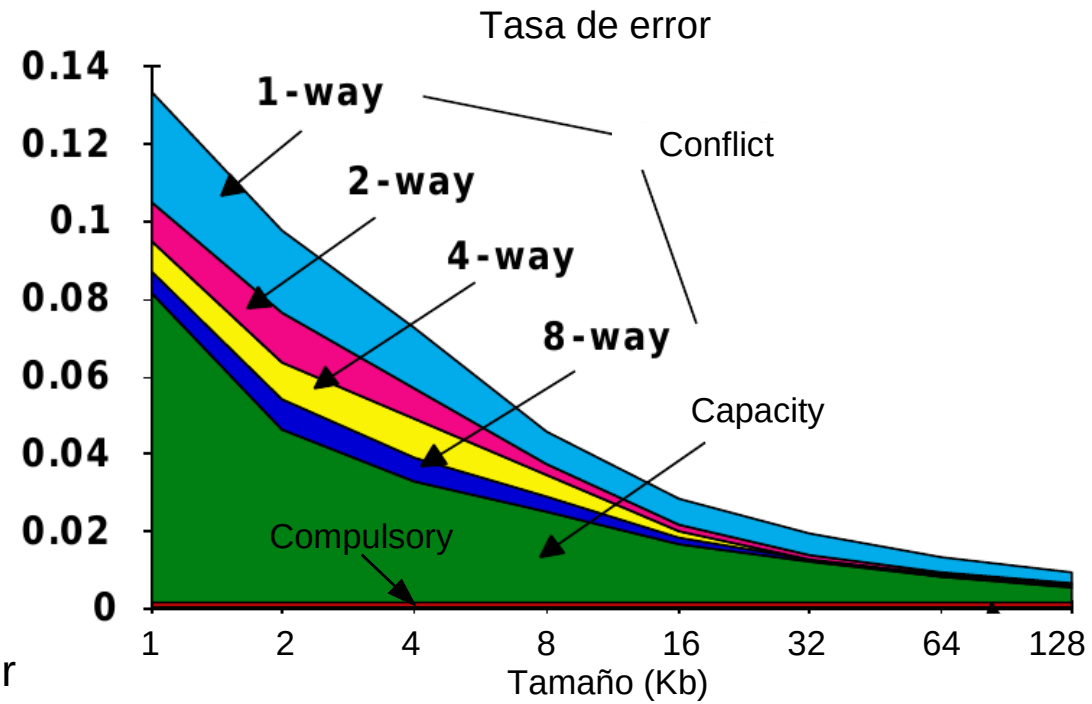
Capacity - ocurre porque los bloques se descartan de la memoria caché debido a que no puede contener todos los bloques necesarios para la ejecución del programa (el conjunto de trabajo del programa es mucho mayor que la capacidad de la memoria cache);

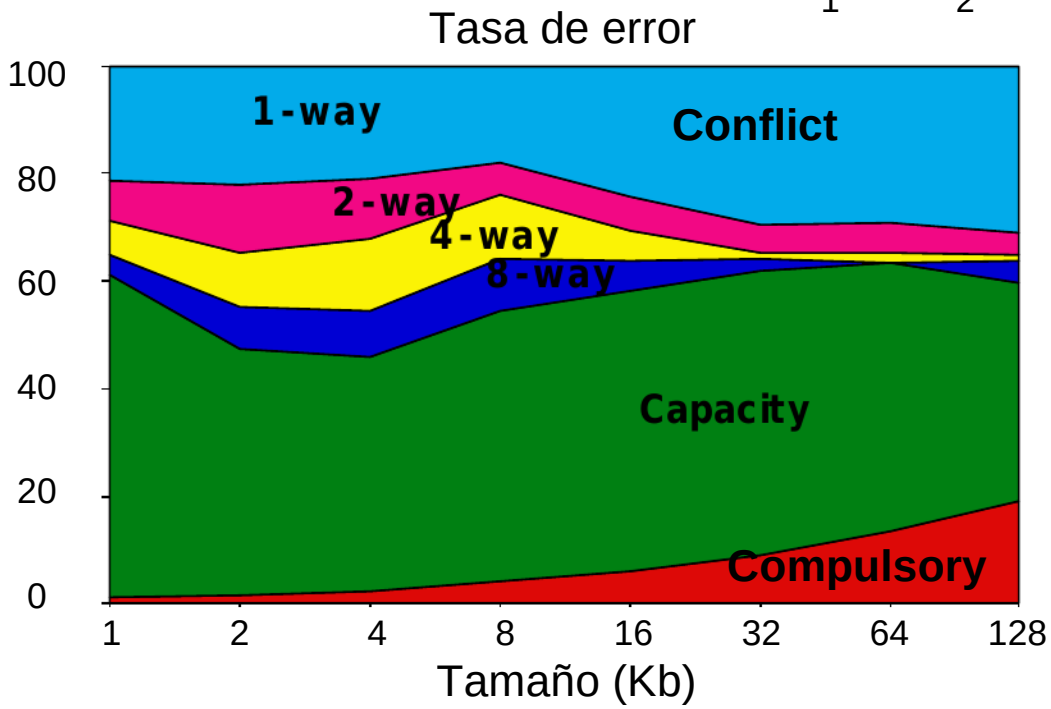
Conflict - en el caso de mapeos asociativas o mapeos directos, los conflictos se producen cuando varios bloques se asignan al mismo conjunto de bloques. También se lo conoce como ***fallo de colisión*** o ***fallo de interferencia***.

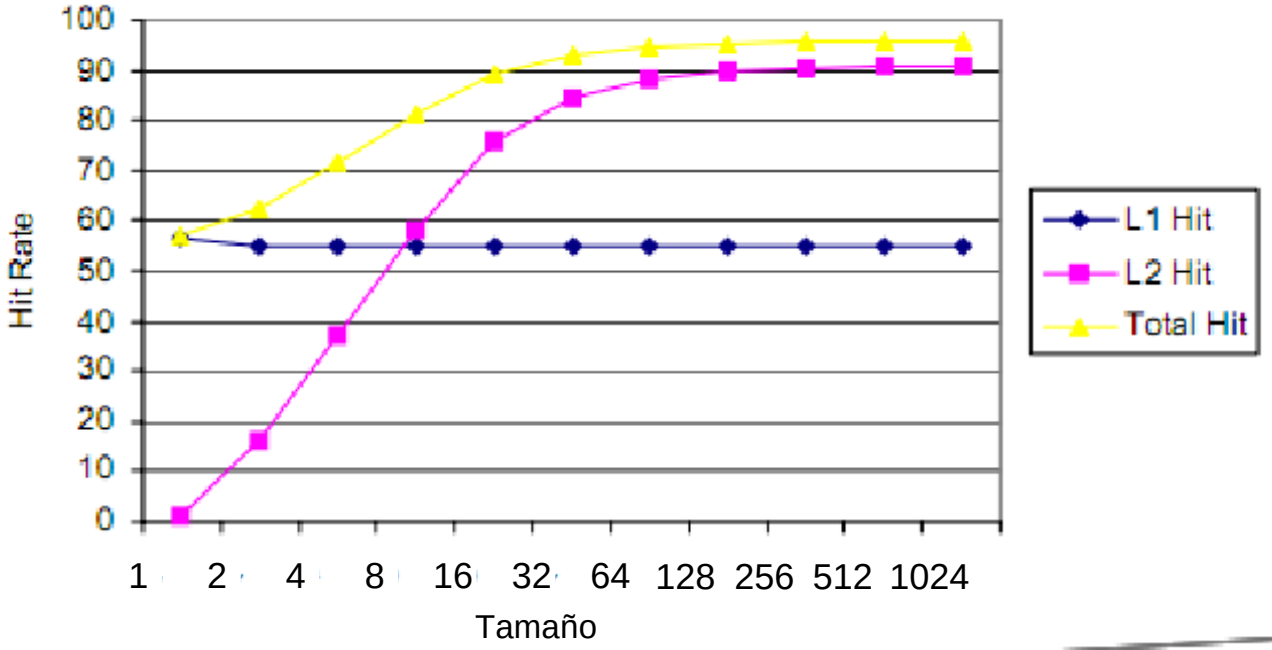
Cambiar los parámetros de la memoria caché puede afectar a uno o más tipos de fallas. Por ejemplo, aumentar la capacidad de la memoria cache puede reducir conflictos y fallas de capacidad, pero no afecta faltas obligatorias. Por otro lado, aumentar el tamaño del bloque podría reducir las fallas obligatorias (debido a la ubicación espacial), pero en realidad podría aumentar las fallas de conflicto (porque más direcciones se mapearían al mismo conjunto y podrían entrar en conflicto).

Memoria cache

Operación - Desempeño

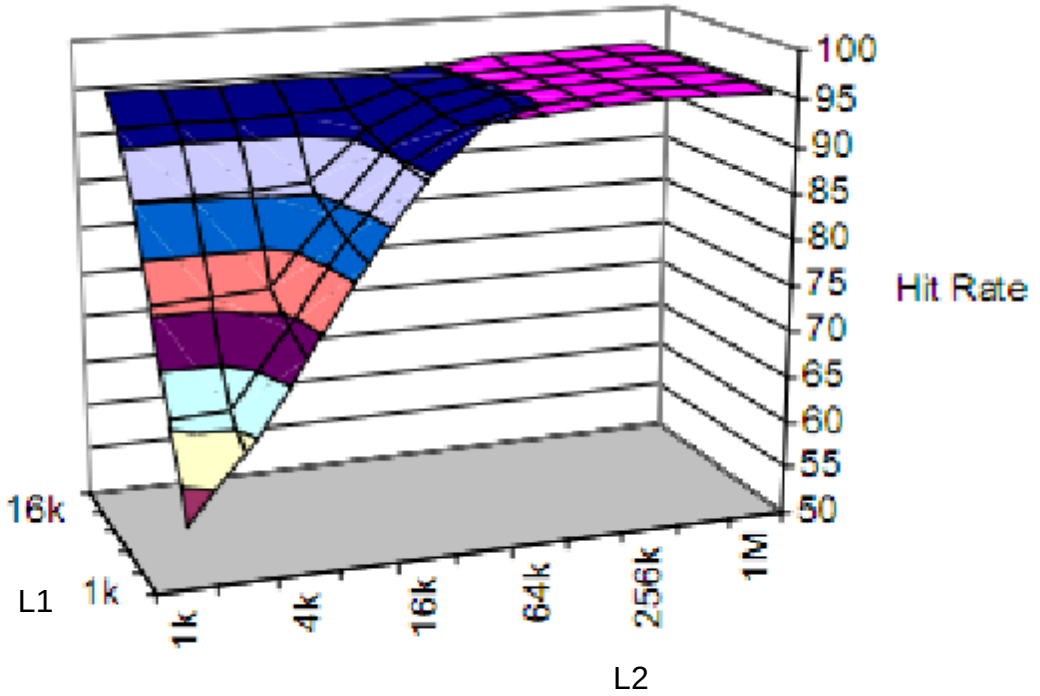






Memoria cache

Operación - Desempeño



Memoria cache

Estrategias de reemplazo

Las politicas de reemplazo son invocadas cuando no hay lugar libre disponible en la memoria cache. Depende de la política de mapeo utilizada:

- **Mapeo directo** – no necesita de una política especial de reemplazo, sólo reemplaza la línea de cache mapeada;
- **Mapeos asociativos** – necesitan de políticas especiales de reemplazo.

Cada estrategia de reemplazo tiene en cuenta el compromiso entre la tasa de aciertos y la latencia.

La tasa de aciertos es la medida de referencia más utilizada para evaluar el desempeño del sistema. La tasa de aciertos varía de una aplicación a otra. En particular, las aplicaciones de transmisión de video y audio a menudo tienen una proporción de aciertos cercana a cero, debido a que cada bit de datos en la transmisión se lee una vez, se usa, y luego nunca usa más.

Otros aspectos a considerar en una política de reemplazo de elementos:

- **Diferentes costos** - mantener los artículos que son costosos de obtener;
- **Ocupación de memoria** - si los artículos tienen tamaños diferentes, la memoria caché puede querer descartar un artículo grande para almacenar varios más pequeños;
- **Caducidad** - la computadora puede descartar artículos porque están vencidos.

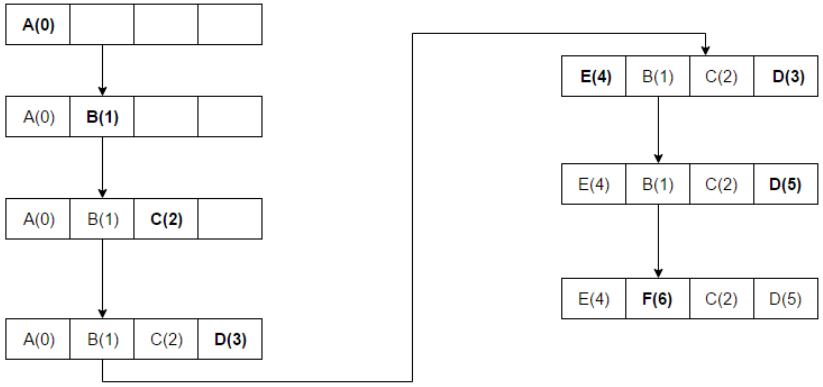
Memoria cache

Estrategias de reemplazo

First in - first out (FIFO) - al usar este algoritmo, la cache se comporta como una cola FIFO. La memoria caché desaloja el bloque al que se accede primero, sin tener en cuenta la frecuencia que se accedió;

Last in - first out (LIFO) – al usar este algoritmo, la cache se comporta exactamente de la manera opuesta a una cola LIFO. La cache desaloja el bloque al que se accedió más recientemente primero, sin tener en cuenta la frecuencia o el número de veces que se accedió anteriormente;

Least recently used (LRU) - descarta los artículos menos usados recientemente primero. Este algoritmo requiere realizar un seguimiento de cuando se utilizó cada bloque, lo cual es costoso. Las implementaciones de esta técnica requieren mantener "*bits de edad*" para cada línea de la caché y, en base a un seguimiento de los mismos, determinar cual bloque se descarta.



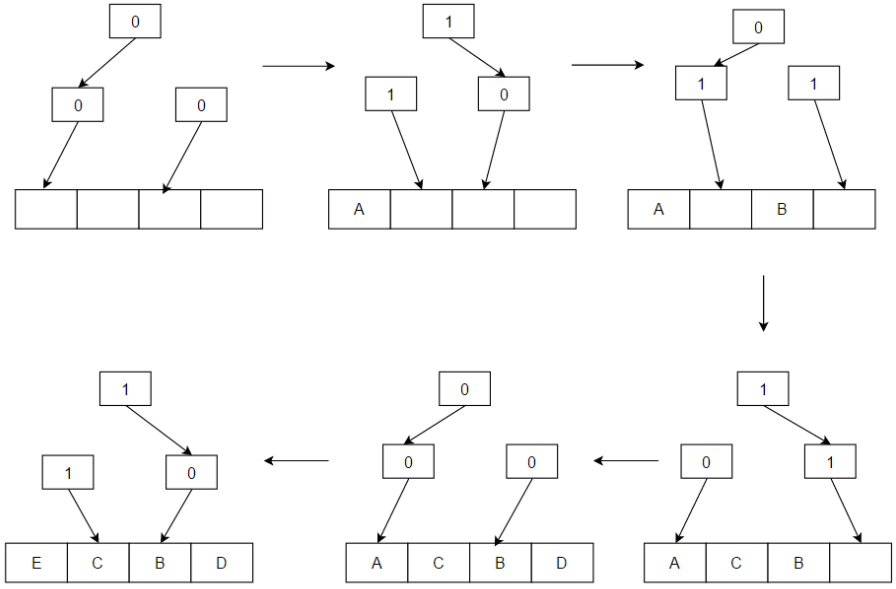
Time aware least recently used (TLRU) - es una variante de LRU diseñada para la situación en la que los contenidos almacenados tienen un tiempo de vida determinado. Introduce el concepto de "*tiempo de uso*" (TTU), el cual es una marca de tiempo-de-un-contenido/página que estipula el tiempo de uso del contenido según la localidad del contenido. El TLRU garantiza que el contenido menos popular y de menor tamaño debe reemplazarse con el contenido entrante.

Memoria cache

Estrategias de reemplazo

Most recently used (MRU) - descarta, en contraste con LRU, los artículos usados más recientemente primero. Los algoritmos MRU son más útiles en situaciones en las que cuanto más antiguo es un elemento, es más probable que se acceda a él.

Pseudo-LRU (PLRU) - para caches de CPU con asociatividad grande (>4) el costo de implementación de LRU se vuelve prohibitivo. En muchos procesadores, el uso de estrategias de reemplazo que ***casi siempre*** descartan uno de los elementos menos usados recientemente (LRU) es suficiente. Por este motivo, muchos diseñadores eligen un algoritmo de PLRU que solo necesita un bit (puntero) por elemento de caché. Por lo general, el PLRU tiene una proporción de fallas ligeramente peor, una latencia ligeramente mejor y emplea menos de potencia que la LRU.



Random replacement (RR) - selecciona aleatoriamente un elemento candidato y lo descarta para hacer espacio cuando sea necesario. Este algoritmo no requiere mantener ninguna información sobre el historial de acceso. Admite simulación estocástica eficiente.

Memoria cache

Estrategias de reemplazo

Segmented LRU (SLRU) - la cache se divide en dos segmentos, uno de prueba y otro protegido. Las líneas en cada segmento se ordenan desde la más hasta la menos recientemente visitada. Los datos de las fallas se agregan a la cache en el extremo del segmento de prueba al que se accedió más recientemente. Los aciertos se eliminan de donde residen actualmente y se agregan al segmento protegido al que se accedió más recientemente. El segmento protegido es finito, por lo que la migración de una línea del segmento probatorio al segmento protegido puede forzar la migración de la línea LRU en el segmento protegido al extremo más recientemente utilizado (MRU), lo que brinda a esta línea otra oportunidad. El límite de tamaño en el segmento protegido es un parámetro del algoritmo.

Least-frequently used (LFU) - cuenta con qué frecuencia se necesita una línea de cache. Aquellas que se usan menos a menudo se descartan primero. Esto funciona de manera muy similar a LRU, excepto que, en lugar de almacenar el valor de la cantidad de acceso a un bloque, almacenamos el valor de la cantidad de veces que se accedió. Así que mientras ejecutamos una secuencia de acceso, reemplazaremos un bloque que se usó la menor cantidad de veces de

Least frequent recently used (LFRU) - este esquema de reemplazo combina los beneficios de los esquemas LFU y LRU. LFRU es adecuado para aplicaciones de ***cache en red***. En el LFRU, la cache se divide en dos particiones llamadas particiones ***privilegiadas*** y ***no privilegiadas***. Si el contenido es ***muy popular***, se ***inserta en la partición privilegiada***. El reemplazo de la partición privilegiada se realiza de la siguiente manera: LFRU desaloja el contenido de la partición no privilegiada, empuja el contenido de la partición privilegiada a la partición no privilegiada y finalmente inserta contenido nuevo en la partición privilegiada. La idea básica es filtrar los contenidos populares localmente con el esquema ALFU y enviar los contenidos populares a una de las particiones privilegiadas.