Arquitectura de Computadoras

CURSO 2022

Turno:

Clase 7

Resumen clase 7

Memoria

- Subsistema Memoria
- Organización jerárquica de la memoria
- Memoria caché , concepto.
- Organización de la caché.
- Consideraciones de la caché
- Análisis cuantitativo
- Políticas de asignación
- Políticas de reemplazo y escritura
- Ejemplos prácticos de procesadores con caché

Subsistema de Memoria

En el diseño de la Memoria existe un compromiso entre:

- Capacidad
- Velocidad
- Costo

Una Memoria ideal sería aquella que:

- Es infinitamente grande
- El tiempo de acceso sumamente pequeño (casi 0)
- El costo relativamente bajo

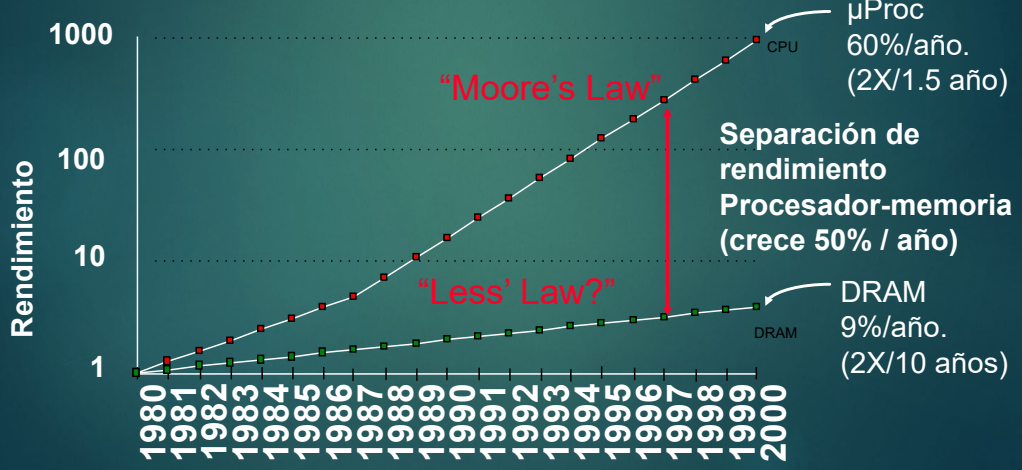
Subsistema de Memoria

Durante años se ha cumplido que:

- La velocidad del procesador aproximadamente se ha duplicado cada 18 meses (casi sin variar su precio), medido en la cantidad de instrucciones ejecutadas por segundo.
- Se ha cuadruplicado el tamaño de la memoria cada 36 meses (al mismo precio). Pero la velocidad aumenta a razón de un 10% anual.
- El desbalance entre la velocidad del procesador y la de la memoria ha generado una brecha que ha crecido a lo largo del tiempo.

Subsistema de Memoria

Esta brecha se puede apreciar en el siguiente diagrama temporal de las velocidades de CPU y memoria, para el periodo 1980-2000.



Subsistema de Memoria

En la figura anterior se pueden ver las curvas de performance de la CPU y la memoria:

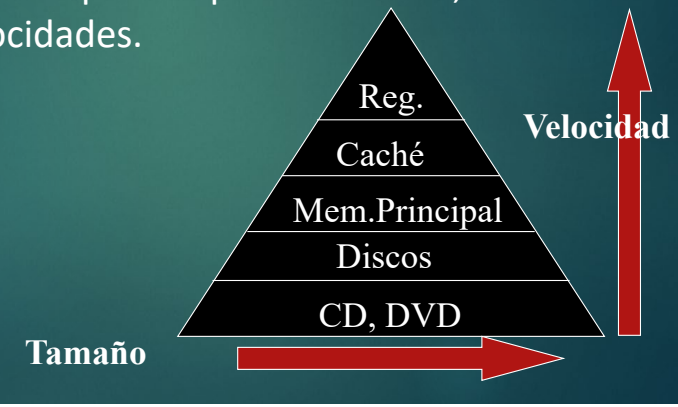
- La velocidad del procesador creció según una curva (conocida como ley de Moore).
- La velocidad de la memoria creció más lentamente.
- Estas diferencias crean un desbalance entre ambos subsistemas.
- Para compensar este desbalance sin impactar fuertemente en el costo del subsistema, se implementa la Memoria como una organización jerárquica compuesta por varios tipos de dispositivos.

Subsistema de Memoria

- En una computadora típica hay distintos tipos de memorias, desde las rápidas y caras (Ej.: registros) hasta las lentas y baratas (Ej.: discos).
- En las computadoras actuales los diferentes tipos de memorias actúan coordinadamente y no separadas.
- Esa interacción permite un comportamiento global equivalente al que tendría con una memoria única, grande y rápida.

Jerarquía de Memoria

- La forma en que se organizan coordinadamente los distintos tipos de memoria se conoce con el nombre de **Jerarquía de Memoria**
- La Jerarquía de memoria se puede pensar como una pirámide de múltiples capas o niveles, de diferentes tamaños y velocidades.



Jerarquía de Memoria

Jerarquía de Memoria es:

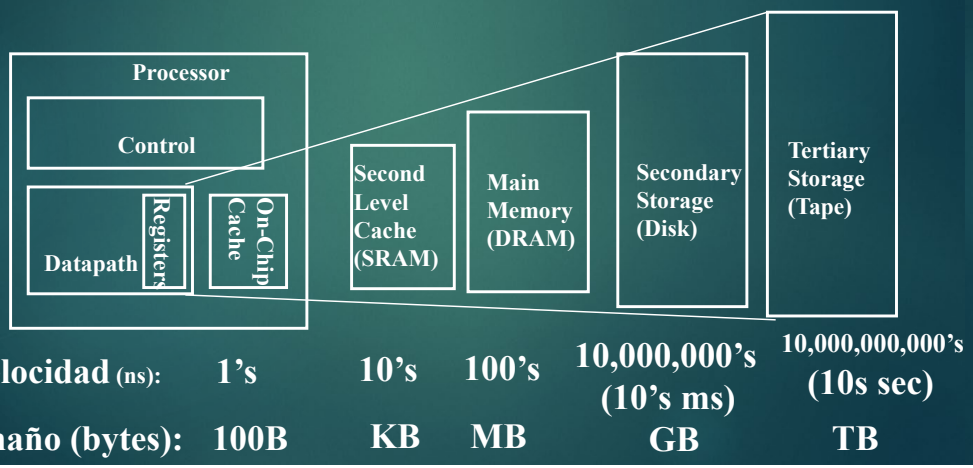
- Un método de administración del almacenamiento de la información (Memoria) estructurado en varios niveles ubicados físicamente en distintos lugares, con tecnologías, costos, tamaños y velocidades distintas.
- De esta manera los programadores "creen" disponer de cantidades casi "ilimitadas" de memoria a un costo accesible, y con velocidades cercanas a las de una memoria "ultrarápida".

Jerarquía de Memoria

- Niveles principales de la Jerarquía de Memoria:
  - Registros
  - Memoria cache (RAM de muy alta velocidad)
  - Memoria principal (RAM de alta velocidad)
  - Memoria virtual o secundaria (medios de almacenamiento magnético/óptico)
- A medida que nos alejamos de la CPU, los niveles son más grandes, más lentos y más baratos que los niveles previos (o superiores) en la jerarquía, de ahí la forma de pirámide.

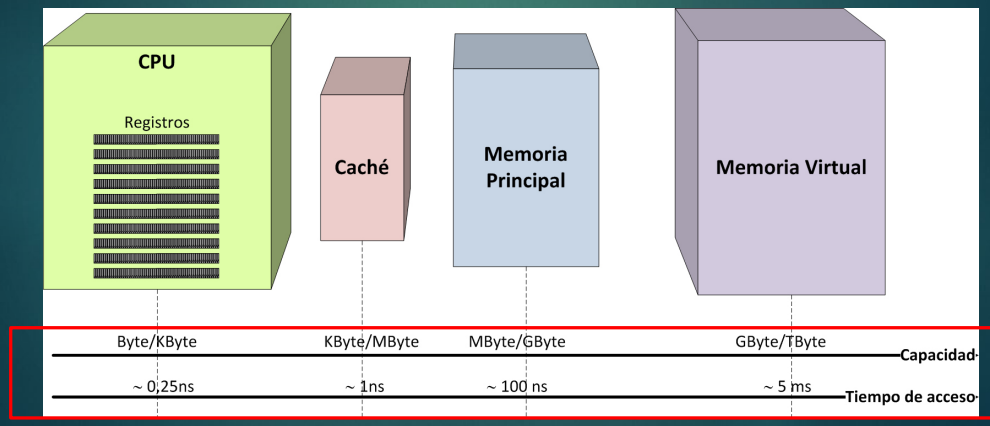
Jerarquía de Memoria

Comparación de ubicación física, tamaños y velocidades de los niveles jerárquicos de la memoria.



Jerarquía de Memoria

Comparación de ubicación física, tamaños y velocidades de los niveles jerárquicos de la memoria.



Jerarquía de Memoria

Los principales objetivos de una Jerarquía de memoria son:

- Maximizar tamaño: idealmente disponer de una "capacidad ilimitada", equiparada al tamaño del nivel más grande.
- Optimizar velocidad: simular que se dispone de un banco de memoria "ultrarápida", próximo a la velocidad del nivel más rápido
- Minimizar el costo total: implementar una memoria a un costo cercano al del nivel más lento.

Jerarquía de Memoria

Para que la memoria se comporte como una jerarquía (integrada) debe cumplir las siguientes propiedades:

- Inclusión: los datos almacenados en un nivel han de estar también almacenados en los niveles inferiores a él.
- Coherencia: las copias de la misma información en los distintos niveles deben contener los mismos valores.

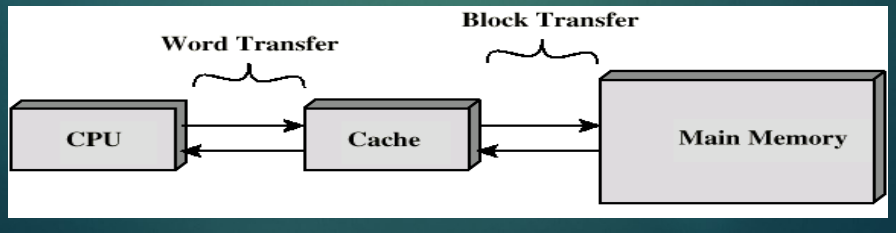
Jerarquía de Memoria

El manejo de la jerarquía de memoria es administrado por:

- Nivel registros: el compilador. Se puede decir que el programador no interviene en la administración de este recurso porque en los lenguajes de programación no son visibles (con algunas excepciones)
- Cache: la administración se hace por hardware
- Memoria principal: la administración la pueden hacer:
  - Hardware
  - Sistema operativo
  - Programador (archivos)

Memoria Cache

- En la Jerarquía de memoria ya se vieron anteriormente los niveles de Registros y Memoria Principal. Ahora se va a analizar el nivel de la Memoria cache.
- La memoria Caché es una memoria pequeña y muy rápida, que se ubica entre la memoria principal y la CPU.
- Puede localizarse en un chip separado o dentro de la CPU, o en ambos lugares.
- Contiene algunos sectores (bloques) de la MP.

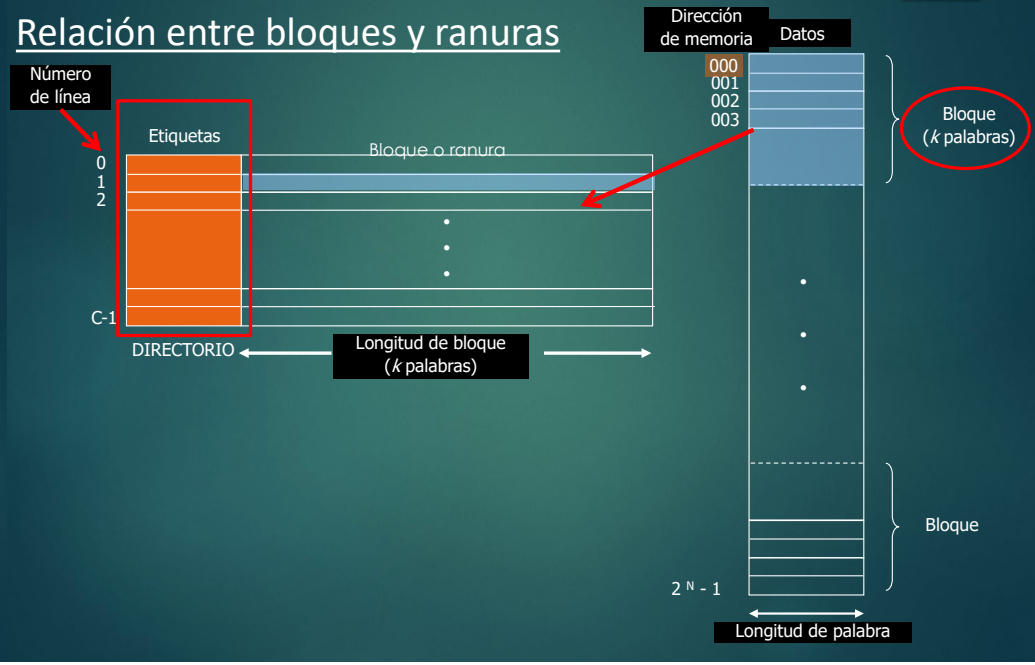


Memoria Cache

- La información contenida en la Caché se organiza en bloques (también llamadas ranuras) de longitud fija (Ej.: 8 bytes, 16 bytes, 32 bytes, etc.).
- En los bloques de la Caché se copian algunos bloques (de idéntico tamaño) de la Memoria principal.
- La cantidad de bloques copiados depende del tamaño de la memoria Caché y del bloque.
- Cada ranura tiene asociada una etiqueta para identificar el bloque de la Memoria que tiene copiado.
- El conjunto de etiquetas forma el directorio de la cache.

Memoria Cache

Relación entre bloques y ranuras



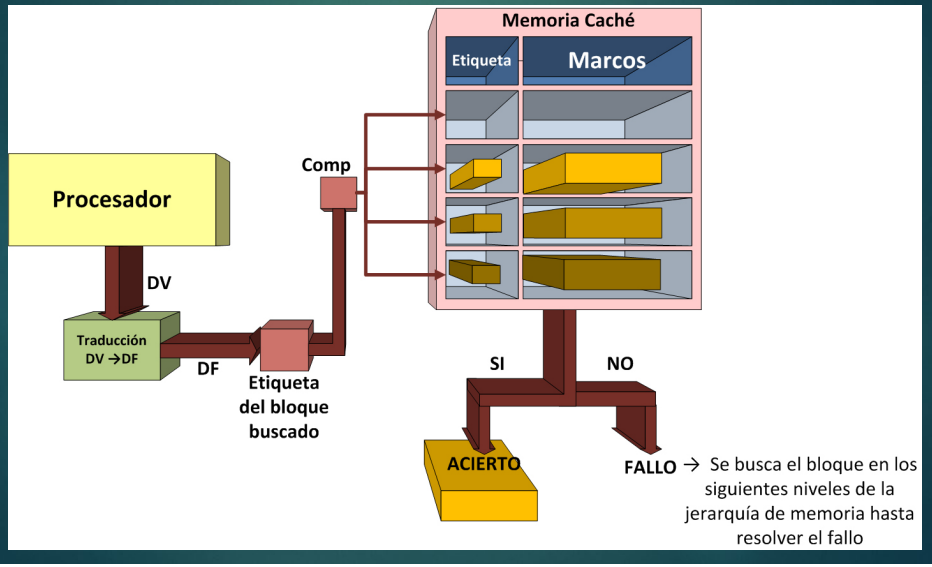
Memoria Cache

Funcionamiento:

- Cuando la CPU necesita un dato, genera 1 dirección de memoria.
- La cache "intercepta" esa dirección y determina si tiene ese dato. Pueden ocurrir 2 situaciones:
  - ACIERTO: si lo tiene, se lo envía a la CPU (a la velocidad de la cache)
  - FALLO: Si no lo tiene, se trae el bloque que contiene esa dirección desde la memoria principal, y la cache entrega el dato requerido a la CPU.

Memoria Cache

Gráficamente, el proceso completo de búsqueda de información cuando hay una memoria caché es el siguiente.



Memoria Cache

- Si el dato que busca la CPU está en la cache, la velocidad del acceso depende del tiempo de acceso de la cache (relativamente muy corto).
- Si el dato no está en la cache, la velocidad del acceso depende del tiempo de acceso de la memoria principal (relativamente largo).
- Así, la eficiencia del uso de la cache depende de la cantidad de veces que "acierta" a la cache.
- La cantidad de veces que se acierta (la "tasa de aciertos") no necesariamente tiene que ser proporcional al tamaño de la caché, que es miles de veces mas chico que el de la Memoria principal. La razón de esto tiene que ver con el comportamiento de los programas.

Memoria Cache

Principios en los que se basa la memoria caché

Los programas tiene un comportamiento basado en 2 principios (que son de carácter empírico).

- Principio de localidad temporal de las referencias:

  Es altamente probable que los elementos de memoria referenciados recientemente (datos o instrucciones) vuelvan a ser referenciados en el corto tiempo.

- Principio de localidad espacial de las referencias:

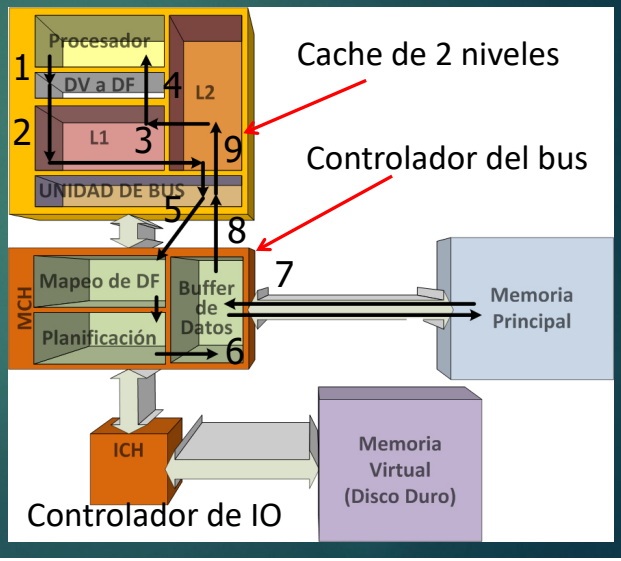
  Es altamente probable que los próximos elementos de memoria referenciados estén en las proximidades de los últimos referenciados.

Memoria Cache

Desde el punto de vista físico, el proceso de acceso a los datos de parte de la CPU se muestra en la figura siguiente.

El camino 1-2-3-4 corresponde a un acierto.

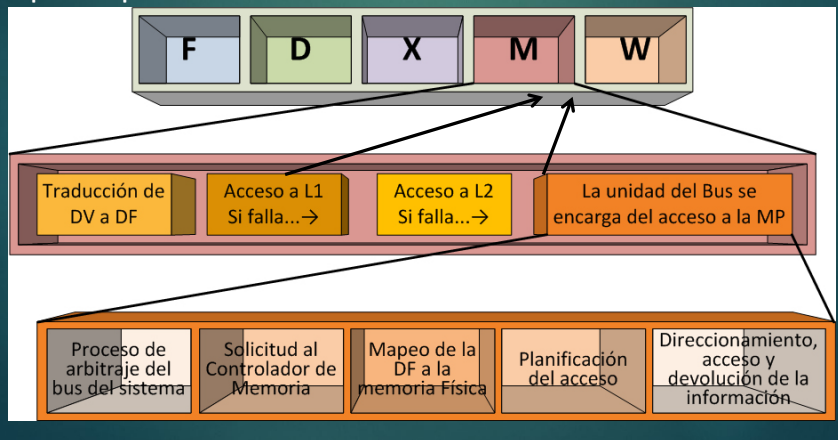
El camino 1-2-5-6-7-8-9-3-4 corresponde a un fallo.



Memoria Cache

Acceso a memoria en un procesador segmentado

En un procesador con segmentación del cauce, se dispone de un ciclo de reloj para el acceso a memoria. Ese tiempo debe ser suficiente para el acceso a la memoria caché. En caso de fallo el acceso a la memoria principal requiere de varios ciclos extra.



Memoria Cache

Tal como se vio, cuando la CPU busca un dato se pueden dar 2 situaciones:

- Acierto (se conoce como "hit"): se encuentra en la caché el dato solicitado
- Fallo (se conoce como "miss"): no se encuentra en la caché el dato solicitado

- Cuando ocurre un fallo, el bloque que contiene la palabra accedida se copia de la memoria principal a una línea de caché.
- Los fallos de caché se gestionan mediante hardware y causan que el procesador se detenga hasta que el dato esté disponible.
- Esta acción requiere un tiempo determinado.

Memoria Cache

- El tiempo para servir un fallo depende de la latencia y ancho de banda de la memoria principal.
  - La latencia es el tiempo necesario para completar un acceso a memoria (depende de la memoria).
  - El ancho de banda es la velocidad a la cual se puede transferir el dato, es decir, es la cantidad de información por unidad de tiempo que puede transferirse desde/hacia la memoria (depende de la velocidad del bus).

Memoria Cache

El tiempo de acceso promedio de la CPU es el promedio del tiempo que tarda en obtener los datos buscados en memoria, compuestos por accesos a la caché y accesos a la memoria principal.

$$t_{CPU} = (1\text{-}TF)\, t_{accesoMC} + TF \times PF$$

TF es la "tasa de fallos", que se obtiene como:

TF=Tasa de Fallos = número de fallos / número total de accesos

PF es la "penalización por fallo", es decir el tiempo "gastado" en acceder a la Memoria principal ($t_{accesoMP}$)

- Para mejorar las prestaciones hay que reducir el $t_{accesoMC}$, TF y PF.

Organización de la cache

Consideraciones sobre la cache

Para el diseño de la caché hay que tener en cuenta varias consideraciones.

1. Tamaño de la memoria caché

- Debe ser suficientemente grande para contener la mayor cantidad posible de información.
- Pero no demasiado grande porque el tamaño tiene impacto en la velocidad (es decir, en el $t_{accesoMC}$) y en el costo.

Organización de la cache

Consideraciones sobre la cache

2. Tamaño del bloque o ranura

- El tamaño del bloque es muy importante en la tasa de aciertos.
- El bloque o ranura debe ser suficientemente grande para aprovechar al máximo las referencias cercanas. Al aumentar el tamaño mejora la tasa de aciertos, hasta un cierto tamaño del bloque. Después de eso, aumentar el tamaño no mejora la tasa de aciertos.
- Por otra parte, al aumentar el tamaño del bloque hay menos bloques de la memoria principal en la caché, lo que tiende a aumentar la tasa de fallos, y la penalización por fallos porque son más palabras a transferir entre la memoria principal y la caché.
- Existe un valor óptimo para el tamaño del bloque.

Organización de la cache

Consideraciones sobre la cache

3. Costo

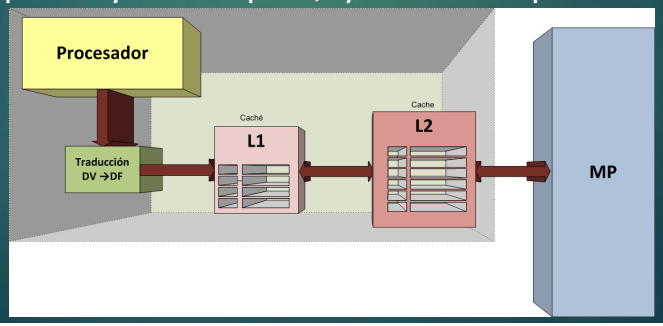
- El costo crece fuertemente con el tamaño de la memoria.
- Y este costo es significativamente grande cuando la caché está incluida en el chip que contiene el procesador.

Organización de la cache

Consideraciones sobre la cache

4. Niveles de la caché

- La memoria caché pude ser una sola (1 nivel) o estar dividida en varias unidades (multiples niveles). Los múltiples niveles de caché por lo general tienen distintos tamaños.
- En la figura siguiente, la caché está dividida en 2 niveles, el L1 muy pequeño y ultrarápido, y el L2 un poco más grande y lento.



Organización de la cache

Consideraciones sobre la cache

5. Separación de la caché de instrucciones y operandos

- El mecanismo de acceso a las instrucciones es distinto al del acceso de los datos. Es por ello que las estrategias para obtener una alta tasa de aciertos es distinta en un caso que en el otro.
- Teniendo en cuenta esto, es posible mejorar la tasa de aciertos general si la caché se divide en una caché de instrucciones y una de datos, con distintas caracteríticas.
- Por ejemplo, ambas caches pueden tener distintos tamaños y políticas de acceso y reemplazo.

Organización de la cache

Análisis cuantitativo de los fallos en la caché

- Para entender el impacto en la velocidad de ejecución de un programa, de los fallos en los accesos a la Caché, supongamos el siguiente problema.
- Se tiene un procesador con:
  - Frecuencia de reloj de 2 GHz
  - CPI=1 (ciclos de reloj por instrucción)
  - Caches de instrucciones y datos separadas,
  - Cuánto tarda en ejecutar 100 instrucciones?

Organización de la cache

Análisis cuantitativo de los fallos en la caché

Si F= 2 GHz

Entonces el ciclo (período) del reloj es:

$T = 1/f = 1 / 2 \times 10^9 \text{ seg} = 0{,}5 \times 10^{-9} \text{ seg} = 0{,}5\text{nseg}$

es decir:

1 Instrucción = 1 ciclo de reloj = 0,5 nseg

Organización de la cache

Análisis cuantitativo de los fallos en la caché - Caso 1

Se tienen 2 memorias caches ideales: CI (caché de instrucciones) y CD (caché de datos):

- Los tiempos de accesos de ambas caché son igual a 0.

$$t_{accesoCI} = t_{accesoCD} = 0$$

- Las tasas de fallos son 0 (la CPU siempre "acierta" a las caché).

$$TF_{CD} = TF_{CI} = 0$$

- El tiempo total requerido por la CPU para ejecutar 100 instrucciones es:

$t_{total} = t_{cpu} + t_{mem}$ donde:

- t_{cpu} es el tiempo de ejecución de la instrucción
- t_{mem} es el tiempo de acceso a las instrucciones y datos

Organización de la cache

Análisis cuantitativo de los fallos en la caché - Caso 1

Los tiempos de accesos a la memoria se pueden descomponer en accesos a instrucciones y accesos a datos:

$t_{mem} = t_{mem\_inst} + t_{mem\_datos}$ por lo tanto:

$t_{total} = t_{cpu} + t_{mem\_inst} + t_{mem\_datos}$

En general, se puede considerar que:

$t_{mem\_inst} = N° \text{ accesosIns}(t_{accesoCI} + TF_{CI} . PF_{MI})$

$t_{mem\_datos} = N° \text{ accesosDat}(t_{accesoCD} + TF_{CD} . PF_{MD})$

donde PF_{MI} y PF_{MD} son las penalizaciones por falla de acceso a las memorias caché de datos e instrucciones.

En el caso ideal que estamos considerando:

$t_{accesoCI} = t_{accesoCD} = TF_{CI} = TF_{CD} = 0$

Organización de la cache

Análisis cuantitativo de los fallos en la caché - Caso 1

es decir:

$$t_{mem} = t_{mem\_inst} + t_{mem\_datos} = 0$$

y, por lo tanto:

$$t_{total} = t_{cpu} + t_{mem\_inst} + t_{mem\_datos} = t_{cpu}$$

Finalmente:

$$t_{total} = t_{cpu} = nI \times CPI \times T = 100 \times 1 \times 0{,}5\ ns = 50ns$$

El tiempo total gastado en ejecutar 100 instrucciones corresponde únicamente al tiempo neto ocupado por la CPU, dado que no hay pérdidas de tiempo por accesos a las memorias caché.

Organización de la cache

Análisis cuantitativo de los fallos en la caché - Caso 2

Ahora se tienen 2 memorias caches con tiempos de accesos despreciables (es decir, 0) pero con tasas de fallas mas "reales":

- Cache de instrucciones con:
  - $t_{accesoCI} = 0$
  - TF = 4% (la tasa de fallos a la caché de instrucciones es de 0,04)
  - PF = 100ns (cuando no accede a la caché de instrucciones se requieren 100nseg para resolver el acceso a la memoria principal)
- Cache de datos con:
  - $t_{accesoCD} = 0$
  - TF = 6%
  - PF = 115ns

Organización de la cache

Análisis cuantitativo de los fallos en la caché - Caso 2

- Suponemos que el 25% de las instrucciones acceden a datos.

Se puede calcular nuevamente el tiempo requerido para resolver 100 instrucciones.

$$t_{total} = t_{cpu} + t_{mem\_inst} + t_{mem\_datos}$$

Los tiempos requeridos para los accesos a las instrucciones y a los datos son:

t_{mem_inst} = N° accesosIns($t_{accesoCI} + TF_{CI} . PF_{MI}$) = 100 (0+0,04x100nseg)

t_{mem_datos} = N° accesosDat($t_{accesoCD} + TF_{CD} . PF_{MD}$) = 25(0+0,06x115nseg)

Por lo tanto:

$$t_{total} = t_{cpu} + t_{mem\_inst} + t_{mem\_datos} = 50nseg + 400nseg + 172nseg = 622{,}5nseg$$

Organización de la cache

Análisis cuantitativo de los fallos en la caché - Conclusiones

- En el análisis anterior se ha considerado que:
  - Tasas de fallo típicos de instrucciones y datos
  - Los tiempos de accesos a las memorias caché se han despreciado respecto de las penalizaciones por fallo
- Las penalizaciones por fallo en los accesos a las cache de instrucciones y datos producen un tiempo de ejecución más de 10 veces superior al requerido por la CPU. Es decir, el tiempo gastado por accesos a la memoria principal (caso de fallos a accesos a las caché) es mucho mayor del propio de la CPU.

Diseño de la caché

En el diseño de la caché se deben definir:

- Organización: tamaño de la caché, cantidad y tamaño de de bloques
- Política de asignación: es decir, el tipo de función de correspondencia entre los bloques de la Memoria Principal y los bloques o ranuras de la Caché.
- Política de reemplazo: se refiere a los algoritmos para reemplazar bloques en la memoria Caché.
- Política de escritura: son los mecanismos de escritura en Memoria Principal.

Políticas de asignación

Las políticas de asignación son las funciones (de mapeo) que definen la forma en que se van a asignar los bloques de la Memoria Principal en la Memoria Caché.

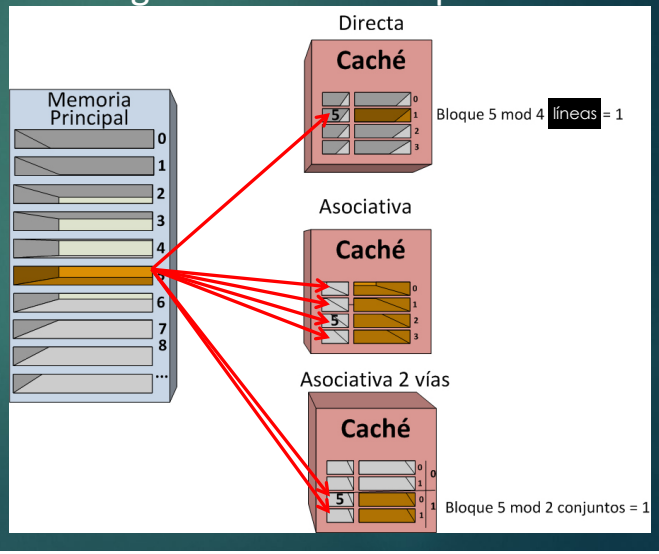
Las políticas más empleadas son:

- Correspondencia totalmente asociativa: Un bloque puede almacenarse libremente en cualquier lugar de la caché.
- Correspondencia directa: Un bloque sólo puede estar almacenado en un lugar fijo de la caché.
- Correspondencia asociativa por conjuntos: Un bloque puede almacenarse en un conjunto restringido de lugares en la caché.

Políticas de asignación

Políticas de asignación típicas

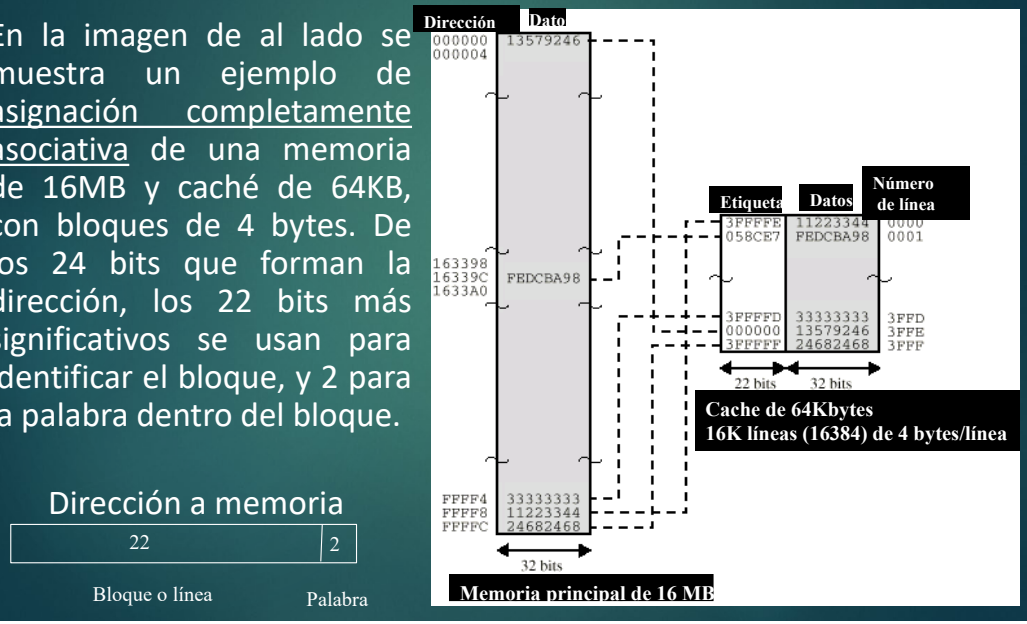
Posibles alternativas de asignación del bloque 5 de la Memoria Principal



Políticas de asignación

Política de asignación Asociativa

En la imagen de al lado se muestra un ejemplo de asignación completamente asociativa de una memoria de 16MB y caché de 64KB, con bloques de 4 bytes. De los 24 bits que forman la dirección, los 22 bits más significativos se usan para identificar el bloque, y 2 para la palabra dentro del bloque.



Dirección a memoria

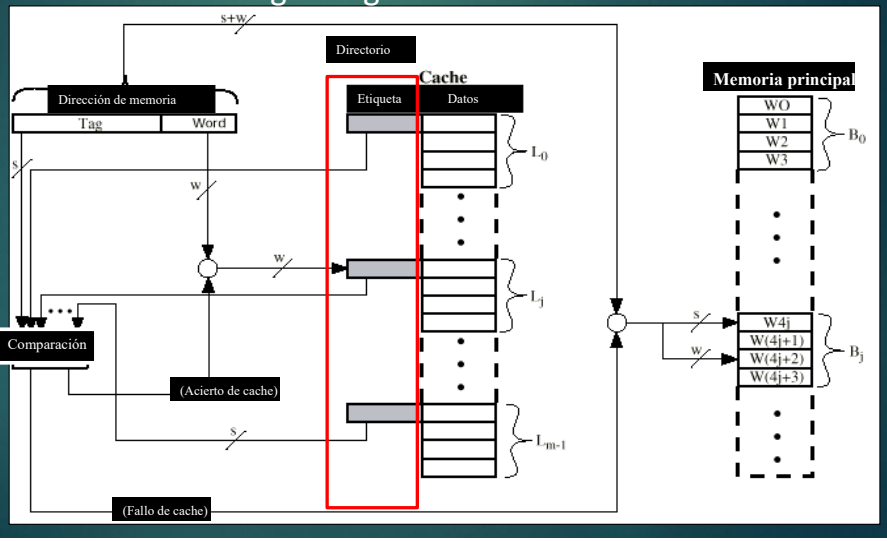
22	2

Bloque o línea Palabra

Políticas de asignación

Política de asignación Asociativa

En general, el esquema de operación en un esquema de asignación asociativa, se muestra en la figura siguiente.



Políticas de asignación

Política de asignación Asociativa

En la figura anterior, se puede apreciar que:

- La dirección de memoria está compuesta de 2 campos
  - Tag: es el número de bloque en la memoria principal
  - Word: es el número de palabra dentro del bloque
- El tag es comparado simultáneamente con todas las etiquetas de la caché, que identifican qué bloques de la MP están asignados a la caché. Las etiquetas de la memoria caché forman el "directorio" de la caché.
- Si la comparación da un acierto, el dato se busca en la caché. Si la comparación da una falla, el dato se trae de la memoria principal.

Políticas de asignación

Política de asignación Asociativa - Conclusiones

- Un bloque de memoria principal puede colocarse en cualquier línea de la cache.
- La etiqueta identifica unívocamente un bloque de memoria.
- Todas las etiquetas de las líneas se examinan para buscar una coincidencia. Para esa búsqueda se require una memoria del tipo asociativa (es decir, una memoria de acceso por contenido CAM) para implementar el directorio.
- La implementación del directorio de la caché es compleja y costosa.
- Esta política permite una libertad absoluta para la asignación y reemplazo de los bloques.

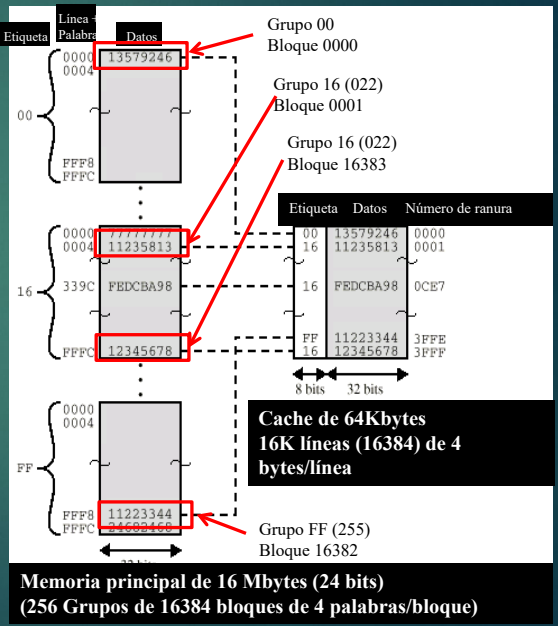
Políticas de asignación

Política de asignación Directa

En la imagen de al lado se muestra un ejemplo de asignación directa de una memoria de 16MB y caché de 64KB, con bloques de 4 bytes. De los 24 bits que forman la dirección, los 8 más significativos son el N° de grupo, los siguientes 14 el bloque dentro del grupo y los 2 restantes la palabra dentro del bloque.

Dirección a memoria

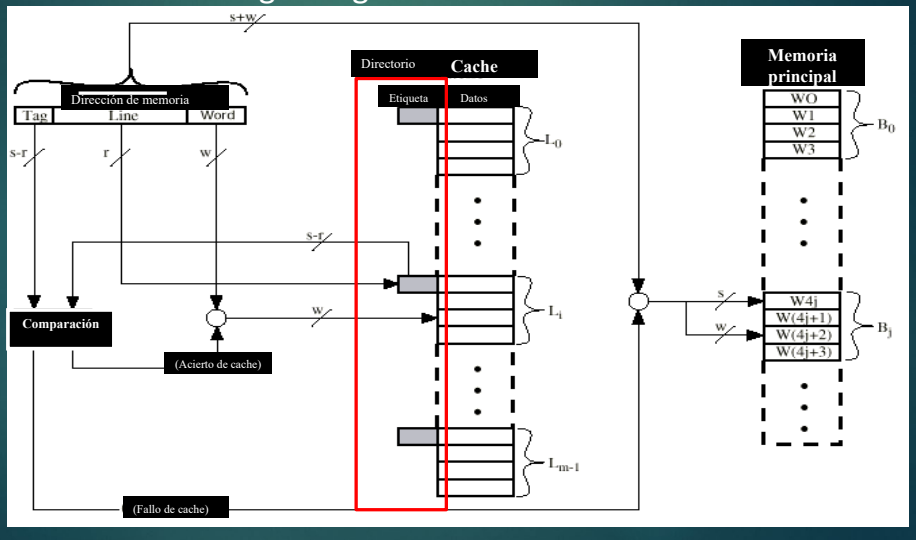
8	14	2
Grupo	Bloque o línea	Palabra



Políticas de asignación

Política de asignación Directa

En general, el esquema de operación en un esquema de asignación directa, se muestra en la figura siguiente.



Políticas de asignación

Política de asignación Directa

En la figura anterior, se puede apreciar que:

- La dirección de memoria está compuesta de 3 campos:
  - Tag: es el número de grupo en la memoria principal
  - Line: en el número de bloque dentro del grupo
  - Word: es el número de palabra dentro del bloque
- El tag es comparado con la etiqueta de la caché correspondiente al número de línea. Hay tantas líneas en la caché como bloques por grupo en la memoria principal. Las etiquetas de la memoria caché forman el "directorio" de la caché.
- Si la comparación da un acierto, el dato se busca en la caché. Si la comparación da una falla, el dato se trae de la memoria principal.

Políticas de asignación

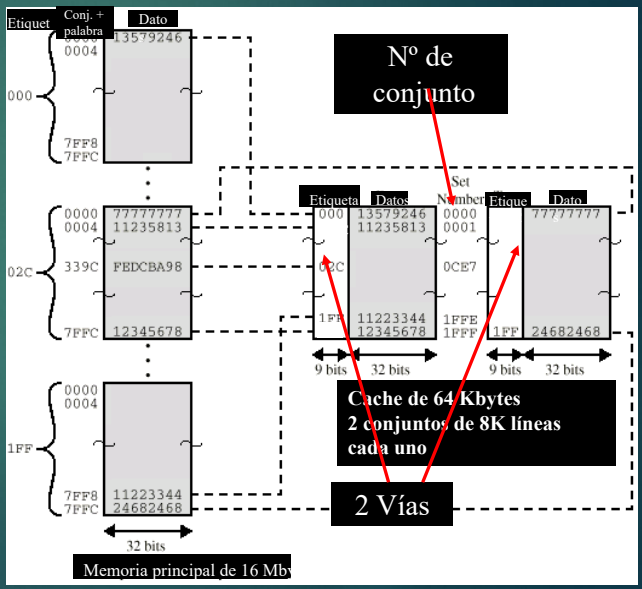
Política de asignación Directa - Conclusiones

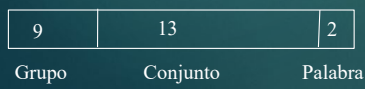
- Un bloque de memoria principal puede colocarse en una única línea de la cache.
  - Nº línea caché = Nº bloque ref. "en módulo" Nº líneas caché
  - La etiqueta solo contiene el número de grupo de la memoria principal asignado a esa línea de la caché.
- Esta política de asignación es muy simple de implementar y poco costosa.
- Tiene un rendimiento acceptable, aunque a veces puede ser malo. Por ejemplo, si un programa accede a dos bloques que se corresponden a la misma línea (diferentes bloques de memoria principal) de forma repetida, las pérdidas de cache (desaciertos) serán muy grandes.

Políticas de asignación

Política de asignación Asociativa por conjuntos

En la imagen de al lado se muestra un ejemplo de asignación asociativa por conjuntos de 2 vías de una memoria de 16MB y caché de 64KB, con bloques de 4 bytes. De los 24 bits que forman la dirección, los 9 más significativos son el N° de grupo, los siguientes 13 el conjunto dentro del grupo, y los 2 restantes la palabra dentro del bloque.

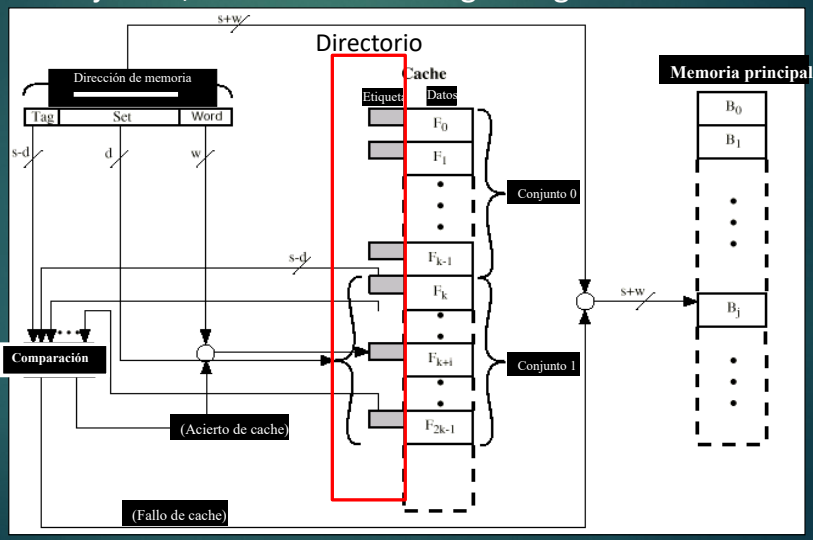




Políticas de asignación

Política de asignación Asociativa por conjuntos

En general, el esquema de operación en un esquema de asignación asociativa por conjuntos, se muestra en la figura siguiente.



Diseño de la caché

Política de asignación Asociativa por conjuntos - Conclusiones

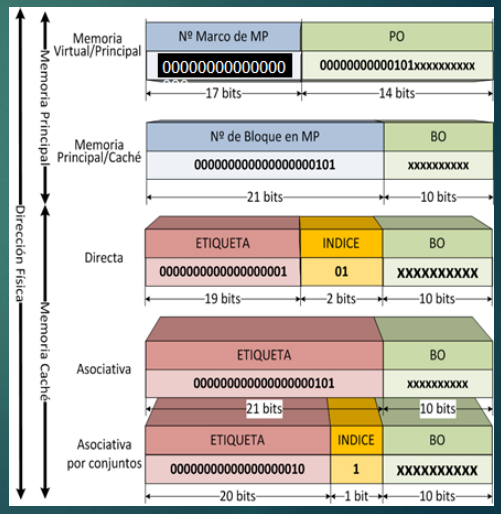
- Un bloque de memoria principal puede colocarse en bloques determinados de la cache. La cache se divide en un número de conjuntos N (N vías, con N=2, 4, 8 ... etc.)
- Cada conjunto contiene un número de líneas o ranuras.
- Un bloque determinado corresponderá a alguna línea o ranura de un conjunto determinado.
- En general, la función asociativa por conjuntos combina lo mejor de las otras correspondencias (asociativa y directa).

Políticas de asignación

Políticas de asignación – Comparación

El siguiente ejemplo compara los resultados de implementar las 3 políticas de asignación.

- MP de 2GB (31 bits)
- Bloques de 1KB (10 bits)
- MC de 4KB
- MC de 4 bloques
- Directa: 2 bits para N° de bloque/grupo + 19 bits N° grupo
- Asociativa: 21 bits N° de bloque
- Asociativa de 2 conjuntos: 1 bit N° de bloque/grupo + 20 bits N° grupo



Políticas de reemplazo

Políticas de reemplazo

- Cuando hay un fallo de acceso a la memoria cache, se debe traer un bloque desde la Memoria principal y almacenar un bloque existente en la memoria cache.
- El lugar a donde va a ser ubicado el bloque a traer desde la memoria principal requiere reemplazar un bloque existente.
- Las diferentes estrategias de reemplazo de los bloques se conocen como Políticas de reemplazo de bloque.

Políticas de reemplazo

Políticas de reemplazo - Correspondencia directa

- En la función de mapeado directo, la asignación de bloques de la memoria principal a bloques de la cache es fija, no hay elección.
- Sólo hay una posible línea para cada bloque.
- Por lo tanto si se requiere traer un nuevo bloque desde la memoria principal, indefectiblemente reeemplazará el que está usando esa ranura actualmente.

Políticas de reemplazo

Políticas de reemplazo - Correspondencia asociativa y asociativa por conjuntos

- Para las asignaciones asociativa y asociativa por conjuntos, hay varios algoritmos de reemplazo:
  1. LRU
  2. FIFO
  3. LFU
  4. Aleatorio
- Todos los algoritmos deben implementarse en hardware (por razones de velocidad).

Políticas de reemplazo

1.- Menos usado recientemente (LRU)

- Se reemplaza el bloque que lleva más tiempo sin utilizarse.
- Requiere controles de tiempos.
- Aprovecha la localidad temporal.
- Válido en Asociativas por conjunto, donde por cada ranura se agrega un bit de USE para identificar el usado recientemente.

2.- Primero en entrar - primero en salir (FIFO).

- Se reemplaza el bloque que entró antes en la caché.
- Requiere controles de acceso para identificar el orden en que ingresaron.

Políticas de reemplazo

3.- Menos frecuentemente usado (LFU)

- Se sustituye aquella línea que ha experimentado menos referencias.
- Requiere controles de uso.

4.- Aleatoria

- Se sustituye una línea al azar.

Políticas de escritura

Políticas de escritura

- Cuando la CPU tiene que almacenar (“escribir”) un resultado en la memoria puede hacerlo tanto en un acierto (la dirección donde se va a guardar el dato está en un bloque de la cache) como en un fallo (la dirección donde se va a guardar el dato NO está en un bloque de la cache).
- En cualquiera de las 2 situaciones, se debe evitar inconsistencia de información entre las memorias principal y cache, durante los procesos de escrituras. Es decir que, aún escribiendose el dato en la cache, el correspondiente bloque de la memoria principal debe ser actualizado en algún momento.
- Además, a veces un módulo E/S puede tener acceso directo a la memoria principal y requerir información que fué modificada en la cache. En arquitecturas complejas (procesadores paralelo) múltiples CPU pueden tener caches individuales.

Políticas de escritura

Políticas de escritura

- Las políticas de escritura son distintas en aciertos que en fallos.
  - Cuando la dirección de memoria a donde tiene que escribir un dato está en la memoria caché, se usan las políticas de escritura en acierto.
  - Cuando la dirección de memoria a donde tiene que escribir un dato no está en la memoria caché, se usan las políticas de escritura en fallo.

Políticas de escritura

Políticas de escritura en aciertos

Hay 2 estrategias:

- Write-through (Escritura inmediata)
  - Se actualizan simultáneamente la posición de la caché y de la memoria principal.
  - Aumenta el tráfico con la memoria principal.
  - Pueden haber retrasos durante múltiples escrituras.
- Write-back (Post-escritura).
  - La información sólo se actualiza en la caché y se escribe la memoria cuando se reemplaza el bloque. El bloque requiere de un bit de “sucio” para indicar cuando se lo escribió.
  - Como la memoria principal se actualiza en el reemplazo, puede contener información errónea en algún momento.

Políticas de escritura

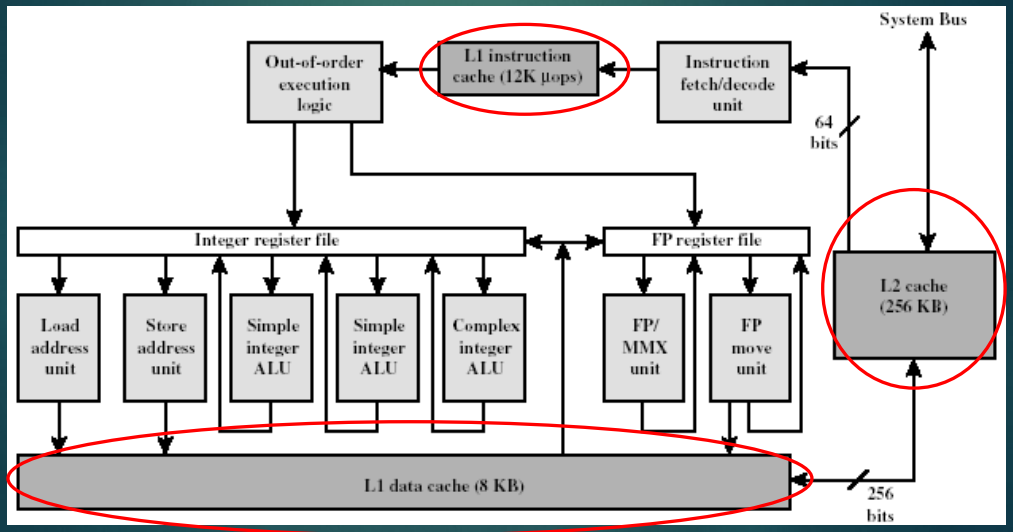
Políticas de escritura en fallos

Hay 2 estrategias:

- No-write allocate
  - Se escribe directamente en la memoria principal. La caché se usa solo en las lecturas. El bloque no se lleva a la memoria caché mientras no se lo tenga que leer.
  - Habitual con write-through.
- Write allocate
  - El bloque requerido primero se copia en la cache, y luego se escribe (en la cache).
  - Habitual con write-back.

Caché de Pentium 4

Ejemplo práctico de memoria caché - Pentium 4



Caché de Pentium 4

Caché de Pentium 4

- 2 niveles de caché: L1 y L2
- Ambas caché integradas en el chip del procesador.
- Ancho de banda de las transferencias: 48 GBytes/s.
- La arquitectura admite un tercer nivel de caché (L3) en el mismo chip (para aplicaciones en servidores).

Caché de Pentium 4

Caché de Pentium 4

- Cache L1: integrada por 2 cache, una para datos y otra para instrucciones
  - Cache de datos
    - Tamaño: 8 KB
    - 128 Bloques de 64 bytes
    - Organización: Asociativa por conjuntos de 4 vías.
    - Política de Escritura: Write-through (inmediata)
    - Velocidad: acceso a los datos enteros en dos ciclos de reloj.
  - Caché de instrucciones:
    - Tamaño: 12 kB
    - Almacena segmentos de caminos de ejecución de instrucciones decodificadas (trazas).

Caché de Pentium 4

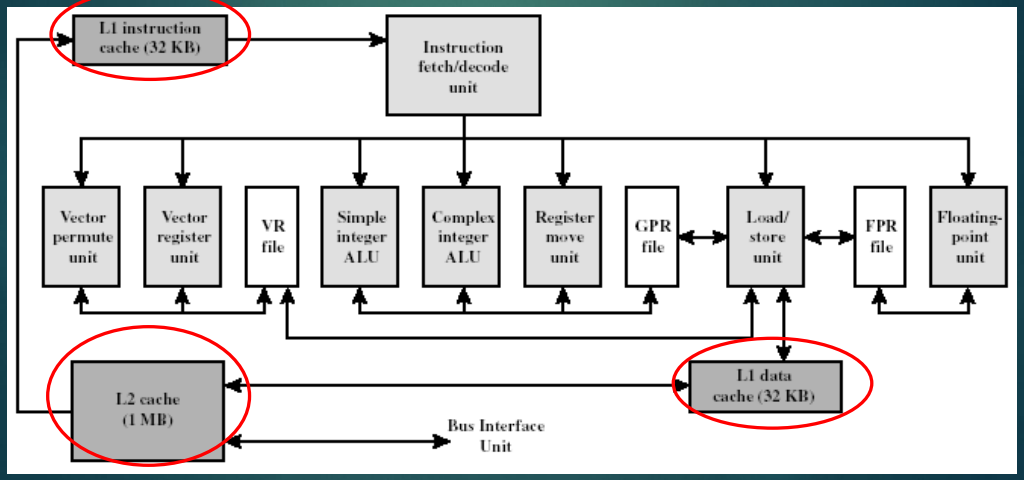
Caché de Pentium 4

- Cache L2 (interna) unificada para datos e instrucciones
  - Capacidad: 256 KBytes.
  - Bloques de 128 bytes.
  - Función de mapeo: Asociativa por conjuntos de 8 vías.
  - Política de escritura: Post-escritura.
  - Latencia de acceso de 7 ciclos de reloj.

Power PC G3

Ejemplo práctico de memoria caché - Power PC G3

El Power PC tiene también 2 niveles de caché L1 y L2 (1MB). La L1 dividida en cache de instrucciones (32Kb) y de datos (32Kb).



Referencias

- Organización y Arquitectura de Computadoras, William Stallings, Capítulo 4, 5ta ed.
- Diseño y evaluación de arquitecturas de computadoras, M. Beltrán y A. Guzmán, Capítulo 2 Apartados 2.1 a 2.4, 1er ed.