Memorias

MSc. Stefano Romero

Tipos de memoria

- En la actualidad, se habla de muchos tipos de memoria. Esto es debido a que las nuevas tecnologías siempre buscan introducir mejoras en los CPUs considerando las limitaciones de velocidad o cuellos de botella.

- Principalmente, se pueden distinguir dos tipos de memoria:

    1. Read-Only Memory (ROM).

    2. Random Access Memory (RAM).

ROM

1. ROM
2. Programmable ROM (PROM)
3. Erasable PROM (EPROM)
4. Electrically erasable PROM (EEPROM)
5. Flash.

Random Access Memory (RAM).

1. **Dynamic RAM (DRAM):** Las memorias DRAM operan básicamente de la misma forma; sin embargo, se presentan varios enfoques:

    - Multibank DRAM (MDRAM)
    - Fast-Page Mode (FPM) DRAM.
    - Extended Data OUT (EDO) DRAM.
    - Burst EDO DRAM (BEDO DRAM).
    - **Synchronous Dynamic Random Access Memory (SDRAM).**
    - Synchronous-Link (SL) DRAM.
    - Double Data Rate **(DDR) SDRAM**.
    - Rambus DRAM **(RDRAM)**.
    - Direct Rambus (DR) RAM.

Static RAM (SRAM)

Static RAM (SRAM): Las memorias DRAM operan básicamente de la misma forma; sin embargo, se presentan varios enfoques:

- Asynchronous SRAM
- Synchronous SRAM
- Pipeline burst SRAM

Características de la memoria principal - Método de acceso

- Acceso secuencial: La información accede a los datos en un orden secuencial. El acceso secuencial puede ser:

  a) Directo: Los bloques tienen una dirección única basada en su dirección física. Para leer/escribir se accede de forma directa al bloque o a una vecindad dada seguido de una búsqueda secuencial.

  b) Indirecto: Se accede por medio de un registro base y un registro de desplazamiento para acceder a un espacio de memoria en particular.

Características de la memoria principal - Método de acceso

- Acceso aleatorio: Es el método más común de la RAM. Los datos se pueden acceder directamente a cualquier ubicación de memoria sin atravesar otros datos en una secuencia en particular. Permite acceder a una dirección de memoria en cualquier momento lo que genera un acceso rápido y eficiente.

Características de los accesos a memoria

a) Tiempo de acceso:

- También conocido como latencia.

- Para memorias de acceso aleatorio es el tiempo que tarda en realizarse una operación de lectura/escritura. Es decir, el tiempo transcurrido desde que se presenta una dirección a la memoria hasta que el dato esté disponible para su uso.

- Para la memoria de acceso no aleatorio, el tiempo de acceso es el tiempo que se tarda en colocar el mecanismo de lectura y escritura en la ubicación deseada.

b) Velocidad de transferencia:

- Velocidad de transferencia de datos hacia o desde una unidad de memoria

Jerarquía de memoria

- No todas las memorias han sido creadas de la misma forma por lo que algunas terminan siendo menos eficientes que otras. Esto afecta el rendimiento del procesador.

- Jerarquía de memorias: Realizar una combinación de diferentes tecnologías de memorias para mejorar el rendimiento al mejor costo.

- Los tipos de base que normalmente constituyen el sistema de memoria jerárquica incluye registros, caché, memoria principal, memoria secundaria y memoria masiva off-line.

- La memoria se puede clasificar según la “distancia” desde el procesador con la distancia medida con el número de ciclos de máquina requerida por acceso. A menor distancia del procesador más rápido debe ser el acceso.

Jerarquía de memoria

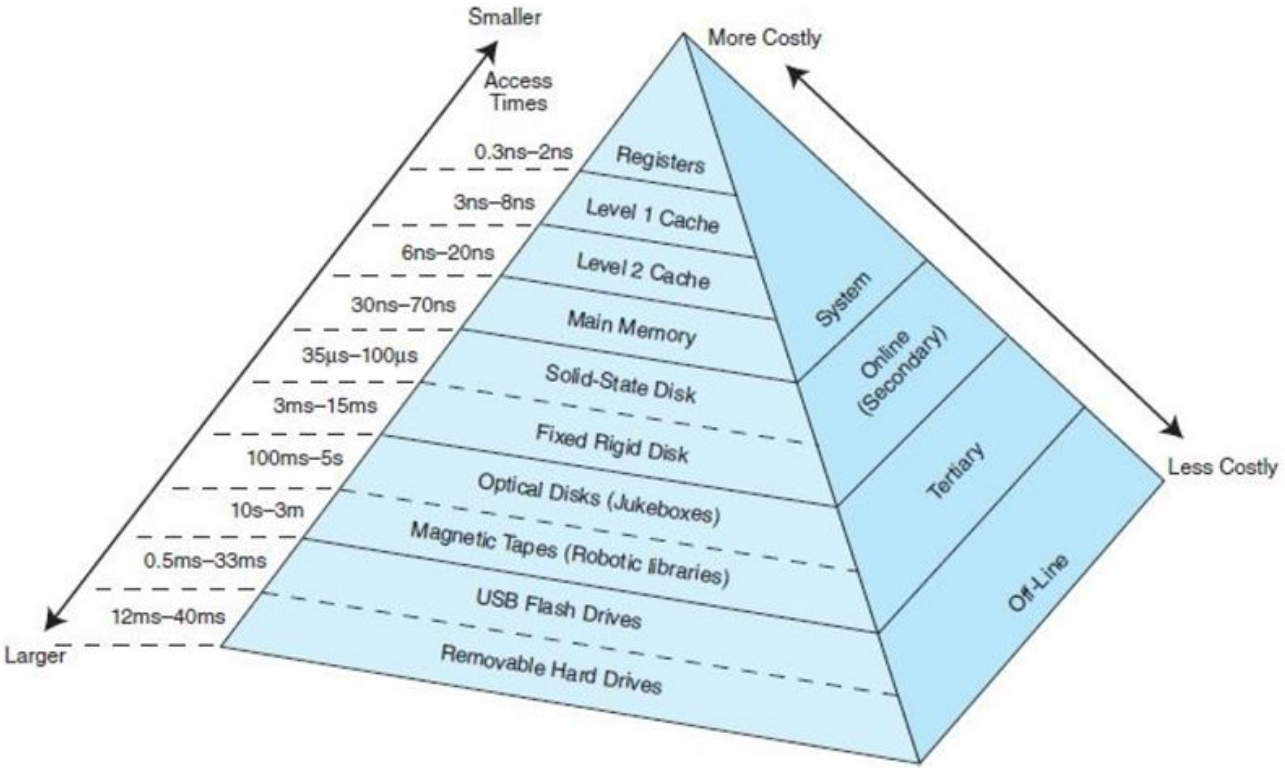


Figura 3.1 Jerarquía de memoria [3].

Jerarquía de memoria

- Para cada dato requerido, el procesador envía un requerimiento a la partición más pequeña y rápida (típicamente la caché).

- Si la data se encuentra en la caché, es posible cargarla en el CPU; por el contrario, si no está el requerimiento es reenviado al nivel inmediato inferior de la jerarquía y el proceso de búsqueda comienza de nuevo.

- En el caso de que se encuentre la data en el siguiente nivel, la información de dichos datos y los vecinos que conforman un bloque son enviados a la caché. De lo contrario, se sigue al siguiente nivel y así hasta concluir.

Jerarquía de memoria

- Hablar de la jerarquía de memoria implica familiarizarse con los siguientes términos:

a) **Hit:** Cuando la data requerida se encuentra en algún nivel de la memoria.

b) **Miss:** Cuando la data requerida no está en ningún nivel de la memoria.

c) **Hit rate:** Porcentaje de acceso a memoria encontrado en un nivel de memoria.

d) **Miss rate:** Porcentaje de acceso a memoria no encontrado en un nivel de memoria (Miss rate = 1-Hit rate).

e) **Hit time:** Tiempo requerido para acceder a la información en un nivel de memoria.

f) **Miss penalty:** Tiempo requerido para procesar una falla (incluye reemplazar un bloque en un nivel superior de memoria, más el tiempo adicional para entregar los datos solicitados al procesador).

Ejemplo

La arquitectura de un procesador contempla 2 niveles en memoria (memoria caché y memoria principal). Se sabe que el tiempo de acceso al nivel 1 es de 0.01 us. Así mismo, el nivel 2 tiene un tiempo de acceso de 0.1 us.

Si se sabe que el Hit Ratio del primer nivel es de 95%:

1) ¿Cuánto tiempo tomará buscar un dato en el nivel 1?
2) ¿Cuánto tiempo tomará buscar un dato en el nivel 2? ¿En qué circunstancias lo debe buscar?
3) ¿Cuál será el tiempo promedio para acceder a un dato dentro del sistema?

Jerarquía de memoria

La idea principal es que cuando el nivel más bajo de la jerarquía responde al requerimiento de altos niveles de contenido en la ubicación X, al mismo tiempo, localiza los valores cercanos a la ubicación X y regresa todo el bloque (que contiene a X) a los niveles superiores.

Esto termina siendo funcional por las propiedades de localización de referencia.

La localización por referencia se divide en 2:

- **Localidad temporal:** Es probable que un recurso que fue empleado recientemente se vuelva a emplear en un futuro cercano.

- **Localidad espacial:** Es probable que un recurso aún no requerido sea accesado si fue algún recurso cercado ha sido requerido.

Memoria cache

- Es una memoria de acceso aleatorio (se lee/escribe en cualquier orden).

- En computadoras personales o de altas prestaciones, su tecnología está basada en la SRAM.

- Su estructura depende de un arreglo de transistores. Cada bits se almacena en 4 transistores y tiene 2 transistores adicionales que permiten el control de acceso para lecturas y escrituras.

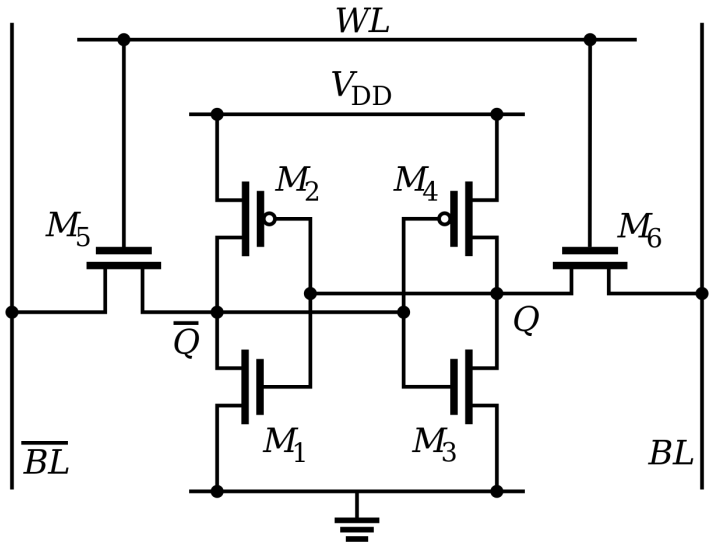


Figura 3.2 Estructura de la memoria caché.

Memoria caché

- La memoria caché (MC) trabaja copiando frecuentemente la data utilizada en la caché en lugar de requerir acceso a la memoria principal para recuperar los datos.

- No hay forma que el CPU sepa a priori a qué datos es más fácil acceder. Siendo así, la MC utiliza el principio de localidad y transfiere un bloque entero de memoria principal a la caché cada vez que tiene que hacer que la MP acceda.

- El tamaño de la memoria caché puede variar considerablemente:
  L1: 8KB - 64KB
  L2: 256KB - 512KB
  L3: MB - GB

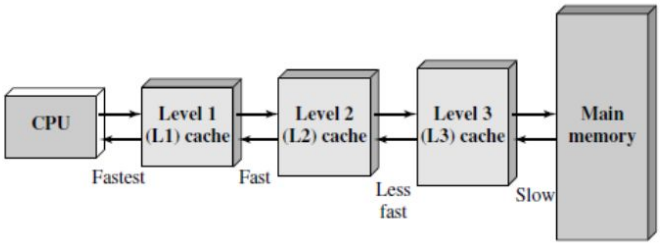


Figura 3.3 Relación entre CPU, memoria caché y memoria principal [1].

Memoria caché

- Una regla general es que la caché sea lo suficientemente pequeña para que el costo promedio general por bit sea similar al de la MP pero lo suficientemente grande como para ser beneficioso.

- Debido a que esta es una memoria rápida y bastante costosa, no es factible utilizar la tecnología que se encuentra en la memoria caché para construir toda la MP.

- La ubicación de la caché para el nuevo bloque depende de dos cosas: la política de mapeo de caché y el tamaño de la caché.

Memoria caché

Para entender claramente el funcionamiento de la MC debemos realizar las siguientes preguntas desde el punto de vista del procesador:

- ¿Está el elemento en la caché?

- ¿Cómo encontrar un elemento en la caché?

La respuesta a estas preguntas está en definir primero qué tipo de mapeo de memoria ha sido utilizado.

1. Mapeo Directo

2. Mapeo Asociativo

3. Mapeo Asociativo por N-vías

Mapeo Directo

- La forma más sencilla de asignar una posición de la caché a cada valor de palabra de la memoria principal (MP) consiste en asignar una posición de la caché basándose en la dirección de dicha palabra en la MP.

- Este método se denomina Mapeo directo, puesto que cada posición de MP se mapea “directamente” con una posición en la memoria caché.

- Procedimiento: Determinar el formato que seguirá la dirección de MP hallando:
  1. # Bits de la MP
  2. # Bits para representar la etiqueta
  3. # Bits para representar el # de bloques de la caché
  4. # Bits para representar el offset dentro del bloque
- Finalmente, el formato se establece de la siguiente manera:

DF: |Etiqueta|Bloque|Offset|

Mapeo Directo - Ejemplo

- Se tiene una memoria principal direccionable por byte con 4 bloques y la memoria caché con 2 bloques de 4 bytes cada una. Se le pide determinar el formato de mapeo.
  Solución:
- # Bits de la MP:

- # Bits para etiqueta:

- # Bits para bloque de la caché:

- # Bits para el offset:

Mapeo Directo - Ejemplo

- En el ejemplo anterior, se partirá del hecho de que la memoria caché está vacía. ¿Qué sucede si un programa hace referencia a la dirección 0x3.

- Dado que ya se sabe el formato, se descompone el número en bits y se lleva el bloque de datos a la caché:

- ¿Qué sucede si se quiere acceder a la dirección 0x2?

- ¿Qué sucede si se quiere acceder a la dirección 0x7?

- ¿Qué sucede si se quiere acceder a la dirección 0xA?

Mapeo Asociativo

- Permite mapear un bloque de MP a cualquier bloque de la MC. Una vez que un bloque. La computadora solo hace uso de la etiqueta para guardar todo el bloque.

- El formato que requiere este tipo de mapeo es bastante sencillo. Basta con saber cuál es el # bits que emplea el desplazamiento y el resto corresponde a la etiqueta que llevará el bloque.

  DF: |Etiqueta|Offset|

- Si la etiqueta que se busca no coincide con las etiquetas de la caché, se producirá un Miss y se reemplazará el bloque con algún algoritmo de reemplazo (FIFO, LRU, LFU, random, entre otros).

- ¿Cuáles son las ventajas y desventajas con respecto al Mapeo Directo?

Mapeo Asociativa por conjunto de N-vías

- Mapeo de alta velocidad y complejidad.

- Es un esquema que junta los enfoques del mapeo directo y asociativo.

- Así como los mapeos previos, el entendimiento de este mapeo se basa en hallar el formato que seguirá la MP, la diferencia es que en lugar de decidir qué bloque se utilizará se decide el conjunto.

    1. # Bits de la MP
    2. # Bits para representar la etiqueta
    3. # Bits para representar el # de conjuntos de la caché
    4. # Bits para representar el offset dentro del bloque

Mapeo Asociativa por conjuntos de N vías

- Al encontrar el conjunto deseado, la caché se trata como un mapeo asociativo ya que las etiquetas de la dirección de la MP se debe comparar con las etiquetas de cada bloque en conjunto.

- Solo se necesita un comparador para el conjunto y no para cada bloque de la caché.

DF: |Etiqueta|Conjunto|Offset|

Mapeo Asociativa por conjuntos

- Por ejemplo, un conjunto de 2 bloques se puede relacionar como una caché de dos dimensiones.

- Sin embargo, se debe tener presente que la memoria es un espacio lineal.

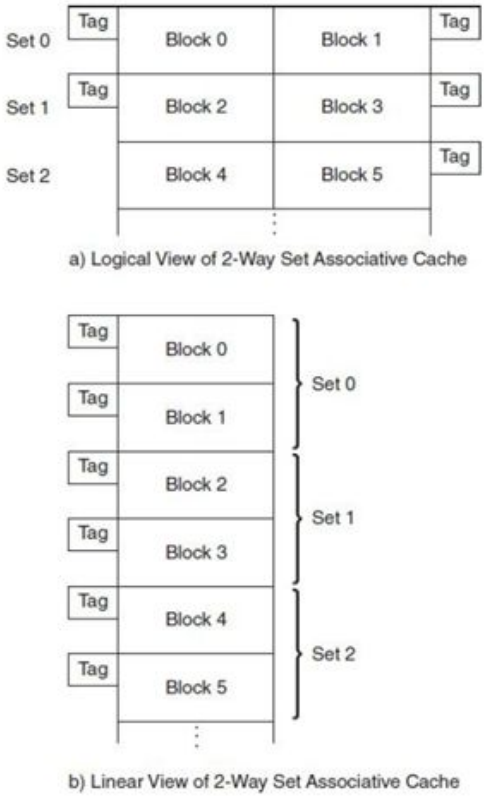


Figura 3.10 Mapeo asociativo por conjunto de N-vías [2].

Mapeo Asociativa por conjuntos - Ejemplo

- Se requiere realizar un mapeo asociativo por conjunto de 2 vías con una MP direccionable por byte de 16 KiB en una MC con 16 bloques de 8 bytes cada uno. Se le pide lo siguiente:

- ¿Cuántos conjuntos tiene la MC y cuántos bits necesita para ser representado?

- ¿Cuál es el formato para este mapeo?
  # Bits de la MP:
  # Bits para representar la etiqueta:
  # Bits para representar el # de conjuntos de la caché:
  # Bits para representar el offset dentro del bloque:

¿Preguntas?

- ¿Qué tipo de mapeo se usará en nuestras computadoras?
- ¿Qué mapeo termina siendo más eficiente?
- ¿Importa el tamaño de la caché para determinar la eficiencia del mapeo utilizado?
- ¿Cómo el tipo de mapeo se relaciona con el Hit Ratio?

Memoria Cache

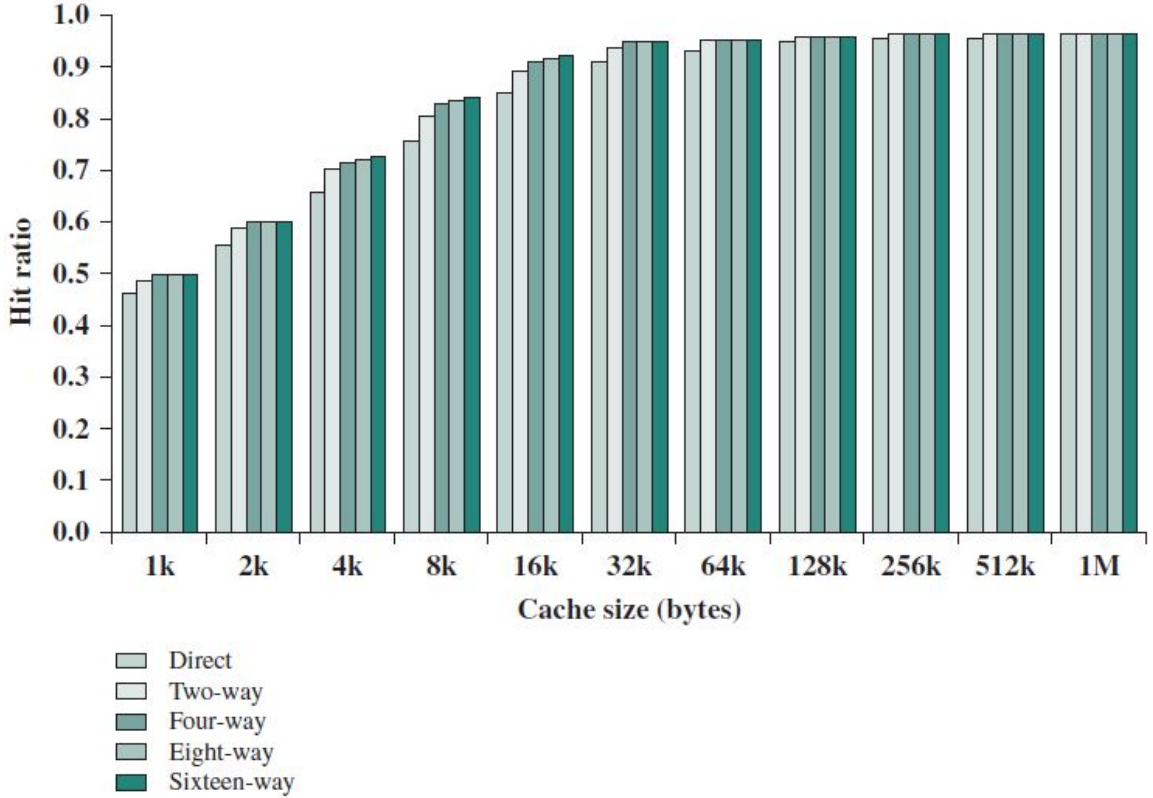


Figura 3.11 Comparación entre Hit - Ratio y tamaño de caché con diferentes tipos de mapeo [2].

Ejemplo: Mapeo

- Un sistema que es direccionable por byte tiene una memoria principal de 1 MiB capacidad y una memoria caché de 32 bloques de 16 bytes cada uno.

- Determinar la ubicación de la dirección de MP 0x326A0 en la memoria caché si se utiliza:

    1. Mapeo directo

    2. Mapeo asociativo

    3. Mapeo asociativo por 4 vías

Ejemplo: Mapeo

- Una cosa que los 3 tipos de mapeo es el tamaño de memoria y el # de bits del offset.
  #bits de MP: 1 MiB->
  #bits del offset: 16 Bytes ->

- Mapeo directo:

- Mapeo Asociativo por N-vías:

- Mapeo Asociativo:

Ejemplo: Mapeo

- Una computadora direccionable por byte tiene una Memoria Caché que consta de 8 bloques de 4 bytes cada uno.

- Si cada dirección tiene 8 bits y la MC está originalmente vacía. Se le pide lo siguiente:

  a) Formato de mapeo directo, asociativo y asociativo por 2 vías.

  b) Realizar un bosquejo de la MC cuando la memoria accede a las siguientes direcciones: 0x01, 0x04, 0x09,0x05, 0x14, 0x21, 0x01. Utilizando los mapeos previos. En caso necesite reemplazar algún bloque en los mapeos asociativos se debe utilizar FIFO.

Políticas de escritura

- Adicionalmente a determinar qué bloque se reemplazará, los diseñadores deben decidir qué se hará con los bloques que han sido modificados.

- Cuando el procesador escribe en la memoria principal, la data puede ser escrita en la memoria caché asumiendo que el procesador utilizará dicha data en una instrucción siguiente.

- Si el bloque de caché es modificado, la política de escritura en caché determina cuándo el bloque correspondiente de memoria se actualiza para que se corresponda con el bloque en memoria caché.

Políticas de escritura

- **Write-through:** Se actualiza tanto la memoria caché como la memoria principal, simultáneamente, en cada escritura. Si bien asegura ser consistente con el sistema de memoria principal, su principal desventaja es que aumenta el tráfico en el bus de la memoria y crea cuellos de botella.

- **Write-back:** También llamada copyback. Solo actualiza los bloques de la memoria principal cuando el bloque de la caché debe ser removido de la memoria caché. Esta forma presenta mayor rapidez que write-through ya que no invierte tiempo escribiendo información en la caché en cada escritura. Utiliza un bit de uso **(dirty bit).**

Políticas de escritura

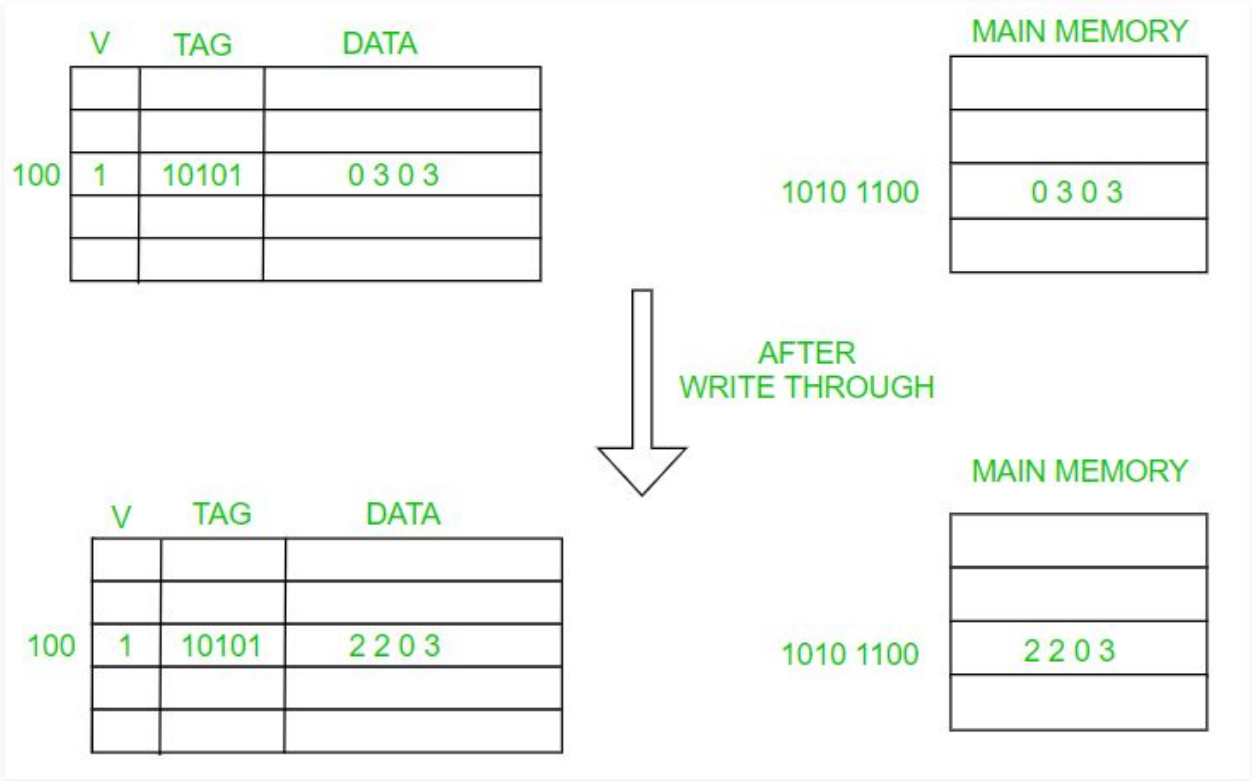
- **Dirty bit:** Cada bloque en la memoria caché necesita un bit para indicar si los datos presentes en la memoria caché se modificaron (dirty) o no se modificaron. Si están limpios, no es necesario escribirlos en la memoria. Está diseñado para reducir la operación de escritura en una memoria. Si la caché falla o si el sistema falla o se corta la energía, los datos modificados se perderán. Porque es casi imposible restaurar los datos de la memoria caché si se pierde.

- **Write Miss:** se da cuando una palabra no se encuentra en el bloque en memoria caché (guardar resultado de operación).

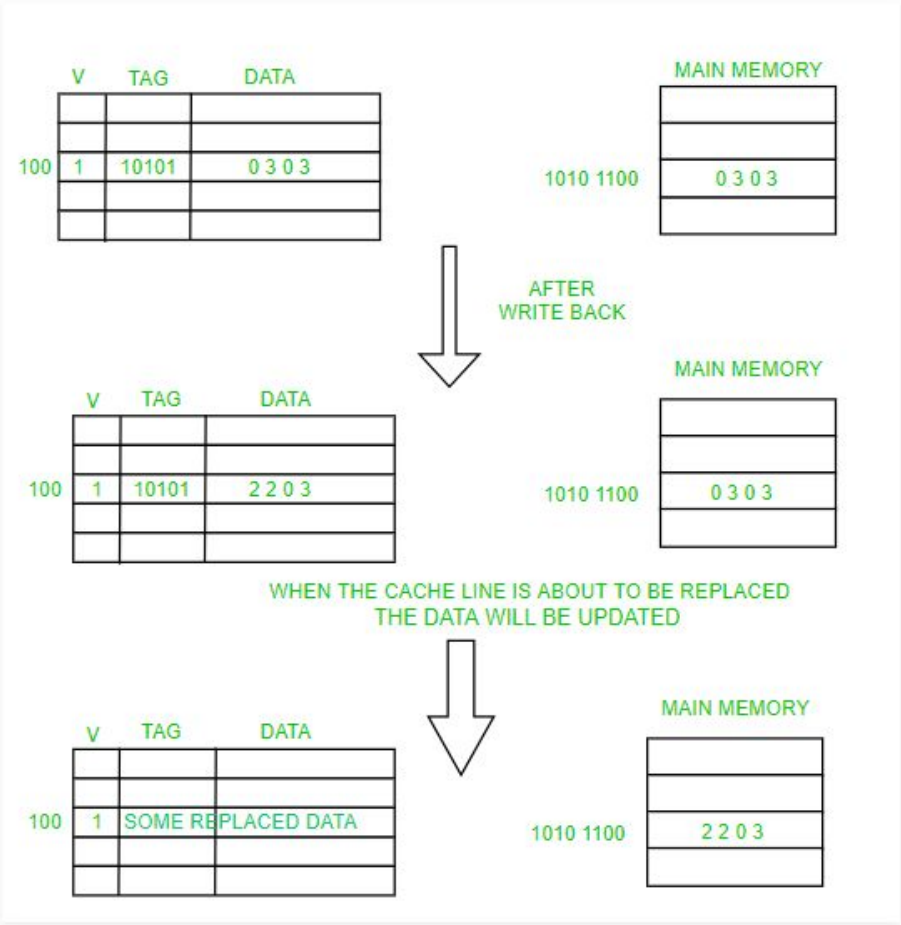
  Write Allocate: Los datos se cargan desde la memoria en la memoria caché y luego se actualizan. Write Allocate trabaja con Write through y Write back. Usualmente, se utiliza con Write Back

  Write Around: los datos se escriben / actualizan directamente en la memoria principal sin alterar la memoria caché. Es mejor usar esto cuando los datos no se vuelven a usar de inmediato.

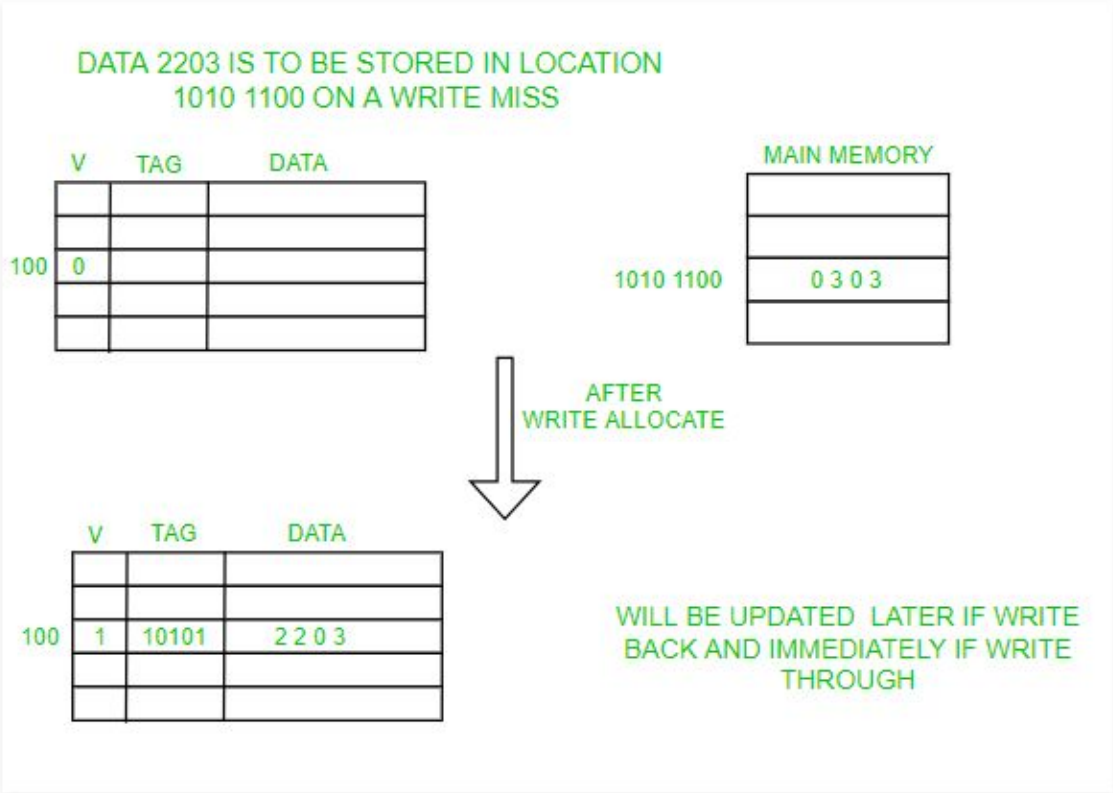
Políticas de escritura - Write Through [4]



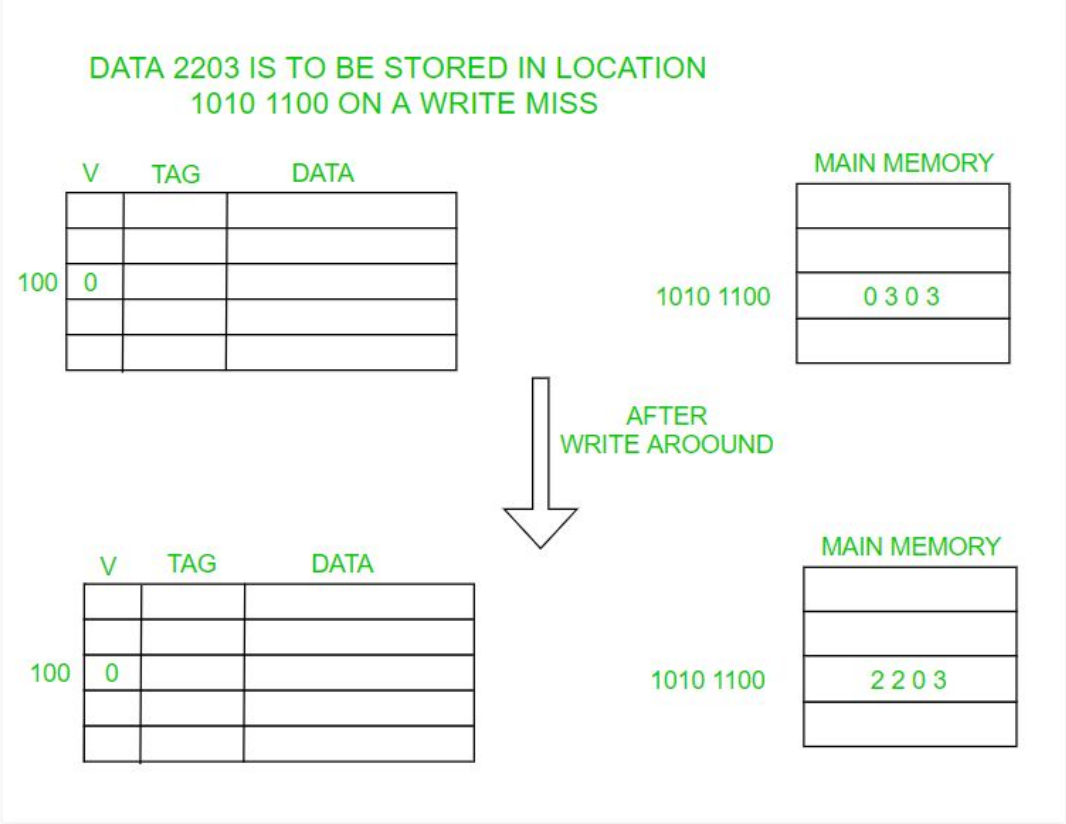
Políticas de escritura - Write Back [4]



Políticas de escritura - Write Allocate (Write miss) [4]



Políticas de escritura - Write Around (Write miss) [4]



Referencias

[1] Stallings, William. Computer organization and architecture: designing for performance. Pearson Education India, 2003.

[2] Null, L., et al. The essentials of computer organization and architecture. Jones & Bartlett Publishers, 2014.

[3] Patterson, D. A., & Hennessy, J. L. Computer Organization and Design The Hardware/Software Interface, 2013.

[4] https://www.geeksforgeeks.org/write-through-and-write-back-in-cache/

Q&A