Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI-0120 Arquitectura de computadoras

Reporte Examen 1

Profesor:

Francisco Arroyo Mora

Estudiante:

Josef Ruzicka González B87095

8 / 10 / 2021

Memoria cache

¿Qué es una memoria cache?

La memoria cache es una memoria SRAM (Static RAM) temporal utilizada por el CPU para accesar de manera rápida a los datos más frecuentemente utilizados, o aquellos que se estén utilizando actualmente. El CPU utiliza esta memoria porque es la más rápida de la computadora por varias razones. Una razón por la que esta memoria es la más rápida para el CPU es porque es la más cercana físicamente, en algunas ocaciones se encuentra integrada al chip del CPU, en otras se encuentra en otro chip conectado al CPU por un bus (Un bus es una conexión interna de la computadora que permite enviar y recibir señales y datos del procesador a otros componentes o viceversa. Fuente: https://www.bbc.co.uk/bitesize/guides/zhppfcw/revision/2).

Existen distintos tipos de memoria cache.

- **L1 Cache (Level 1 cache):** Es muy rápida y generalmente se encuentra embebida en el chip del CPU es la primer memoria en la que el CPU busca datos. Si el procesador tiene varios núcleos, existirá una L1 cache para cada uno de ellos y la latencia en la comunicación es tan solo 0.9 ns (Para tener un punto de comparacion: La latencia en la memoria RAM suele estar cerca de los 15 ns y la de un disco duro puede estar entre 1 y 20 ms dependiendo si es de estado sólido o no, entre otras características. Este nivel de memoria cache tiene su almacenamiento separado en dos categorías: datos e instrucciones. Fuentes: https://louwrentius.com/understanding-storage-performance-iops-and-latency.html, https://www.tvtechnology.com/miscellaneous/seeking-hard-disk-drive-latency, https://www.crucial.com/articles/about-memory/difference-between-speed-

and-latency). El promedio de espacio en esta memoria suele rondar los 256 KB en total, divididos entre los núcleos.

- **L2 Cache (Level 2 Cache):** Esta memoria suele ser más amplia en tamaño que la L1 cache (suele ir entre los 256 KB y los 18 MB) pero es más lenta, su latencia suele rondar lo 2.8 ns. También puede presentarse embebida al chip del procesador, o estar en un chip diferente conectado al CPU con un bus. Esta es la segunda memoria que el CPU revisa si no encuentra los datos que necesitaba en la L1 Cache. Si el procesador presenta múltiples núcleos, suele haber una L2 Cache por cada núcleo.

- **L3 Cache:** Esta es una memoria que es compartida entre los núcleos del procesador y puede tener entre 4 MB hasta 64 MB de memoria generalmente aunque es más lenta que las memorias de tipo L1 y L2 ya que presenta una latencia de 11 ns.

(Fuentes: https://www.britannica.com/technology/cache-memory, https://searchstorage.techtarget.com/definition/cache-memory, https://www.hp.com/us-en/shop/tech-takes/what-is-cache-memory, https://www.profesionalreview.com/2019/05/02/memoria-cache-l1-l2-y-l3/#Memoria_cache_L1)

Uso de la memoria cache

Los datos e instrucciones que se encuentran en la memoria cache son cargados desde la memoria RAM y son los que el CPU ha utilizado más reciente y frecuentemente. Cuando el CPU busca algún dato en la memoria cache (La información se busca en el orden jerárquico, primero L1, luego L2 y finalmente L3), se le denomina un “Hit” si encuentra la información deseada, o un “miss” si la información no está presente en la memoria cache, en estos casos se debe acceder a la memoria RAM, buscar los datos ahí y almacenarlos en la memoria cache, la manera en la que

este proceso se realiza está determinado por la política de búsqueda, esta puede ser por demanda, es decir, que cuando se requiere un bloque de la memoria principal que no se encuentra en la memoria cache, se busca este bloque y se almacena en una linea de la memoria cache, o la política también podría ser anticipativa, también conocida como prebúsqueda, en la que cuando se almacena un bloque de la memoria principal en la memoria cache también se almacena el siguiente bloque de la memoria principal preventivamente por si se fuera a dar el caso de que también se va a utilizar. Los misses se clasifican por tipos dependiendo de la razón por la cuál se presentaron:

- **Compulsory miss:** Ocurre la primera vez que se accede a la dirección solicitada.
- **Capacity miss:** Se presenta cuando el working set es muy grande (Working set se refiere a las direcciones activas actualmente en la memoria cache)
- **Conflict miss:** Sucede cuando hay varios bloques de la memoria principal mapeados en un solo set de de la memoria cache (En la sección de mapeo se explicará qué es un set de la memoria cache).
- **Coherence miss:** Se da cuando la memoria es modificada por un procesador externo.

  (Fuentes:
  https://www.sciencedirect.com/topics/computer-science/direct-mapped-cache,
  https://www.geeksforgeeks.org/types-of-cache-misses/)

Para medir el rendimiento de la memoria cache se mide utilizando un “Hit ratio” que se calcula con la siguiente fórmula: Hit ratio = hit / (hit + miss) = hits / total de accesos.

En el momento que se desea escribir datos en la memoria cache se debe tomar la decisión de cuál método o técnica utilizar para dicha operación, existen dos más comúnmente elegidas:

- **Write-through:** Con este método los datos se escriben en la memoria cache, pero también se escriben simultáneamente en la memoria principal por lo que siempre se ejecutan dos escrituras.

- **Write-back:** A diferencia del método Write-through, al utilizar esta técnica las escrituras se realizan primeramente solo en la memoria cache por lo que es más eficiente al no tener que realizar más de una escritura. Esto genera inconsistencias en los datos que existen en la memoria cache y los que están presentes en la memoria principal, por lo que se debe analizar el bit conocido como “dirty bit”, este es un bit que se utiliza como una bandera presente en cada bloque de memoria para marcar si dicho bloque ha sido modificado, cuando se encuentra una inconsistencia de este tipo entonces sí se procede a escribir los nuevos datos en memoria principal.

Para almacenar datos de la memoria principal en la memoria cache existen diversos métodos de mapeo:

- **Direct cache mapping:** Este mètodo es el màs simple, le asigna a cada línea de la memoria cache un bloque de la memoria principal con la siguiente fórmula: “linea = direccionDelBloque MODULO lineasDeCache”, cuando se agrega el bloque a su línea designada, reemplaza cualquier bloque que se encuentre en esa línea. En la linea de cache, el bloque queda guardado como una estructura de 3 campos:
  - El bloque de memoria.
  - El tag (dirección del bloque en la memoria principal).
  - Bit que identifica la linea de cache.

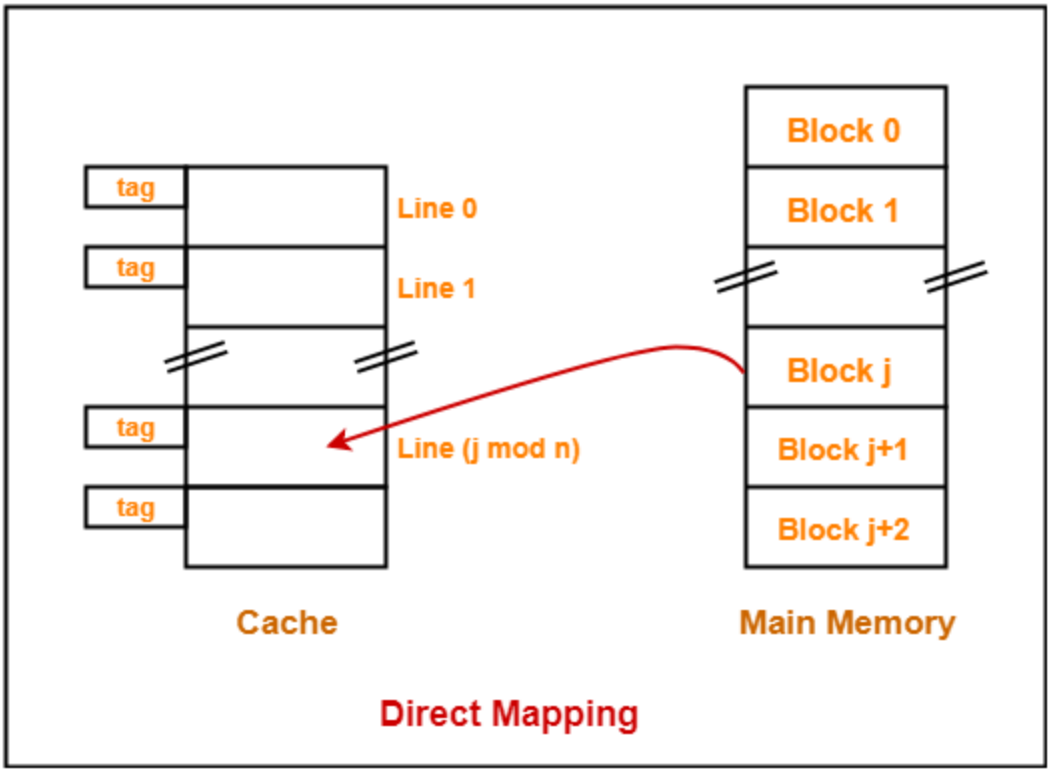

Imagen tomada de https://www.gatevidyalay.com/cache-mapping-cache-mapping-techniques/#:~:text=Cache%20mapping%20is%20a%20technique%20that%20defines%20how%20contents%20of,K%2Dway%20Set%20Associative%20Mapping.

- **Fully associative cache mapping:** Este tipo de mapeo es muy parecido al método “Direct mapped cache” pero es màs flexible con respecto a la posiciòn de los bloques de memoria en las lineas de cache ya que permite que cada bloque cargado de la memoria principal sea guardado en cualquier linea de la memoria de cache. Este método es considerado el más rápido de los tipos de mapeo, sin embargo, cuando la memoria cache está llena y se desea guardar en ella un bloque de la memoria principal, para decidir cuál bloque debe ser reemplazado se debe utilizar un algoritmo de reemplazo, este tipo de algoritmos se encuentra más detallado en la siguiente sección del escrito.

- **Set associative cache mapping**: También conocido como “N-Way set associative mapping” es un método de mapeo comúnmente considerado como un híbrido entre los dos métodos anteriores ya que no es totalmente flexible en cuanto a la ubicación de los nuevos bloques de memoria principal que se desean guardar en la memoria cache, pero tampoco se propone una única línea para esta operación, si no que este mapeo asigna N posibles lineas en la memoria cache donde cada bloque de la memoria principal podría ser almacenado en el momento que se desee, A este grupo de N lineas se les denomina “Set” y más especificamente “N-Set” donde N es la cantidad de lineas que conforman dicho set. La fórmula que indica a qué set pertenece un bloque de memoria es la siguiente: “Set = direcciónDelBloque MODULO SetsEnCache”. Si todas las lineas del set se encuentran ocupadas, se determina cuál debe ser reemplazada con un algoritmo de reemplazamiento (Explicados con más detalle en la siguiente sección del escrito).

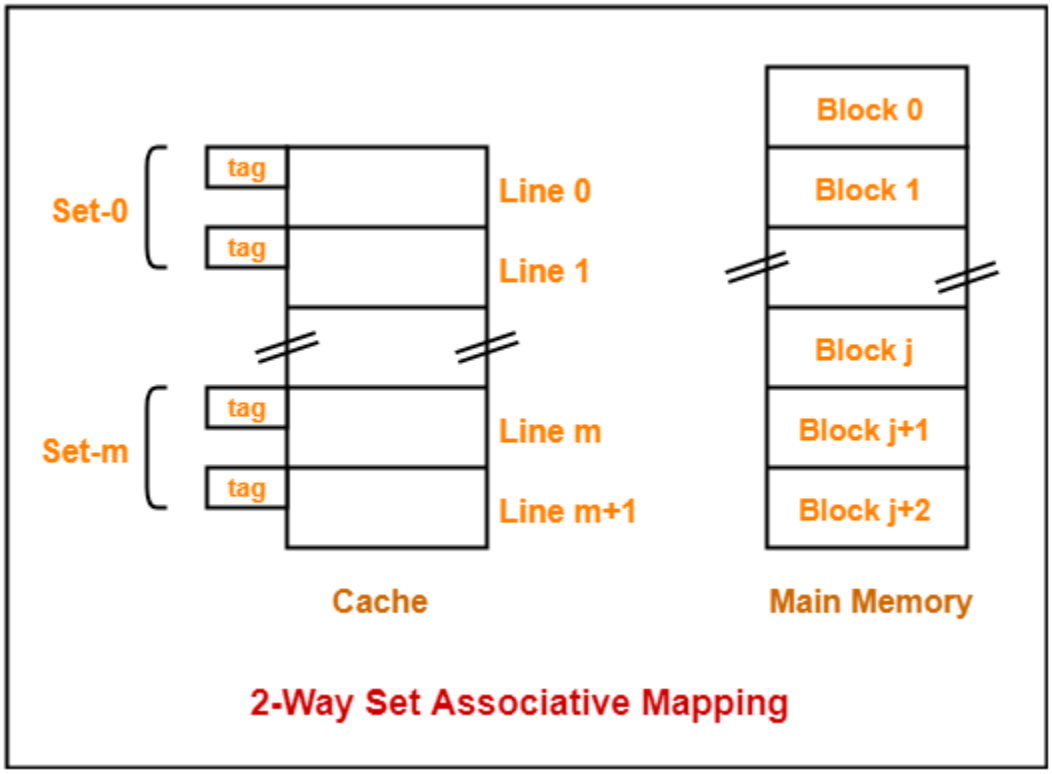

Imagen tomada de https://www.gatevidyalay.com/cache-mapping-cache-mapping-techniques/

#:~:text=Cache%20mapping%20is%20a%20technique%20that%20defines%20how%20contents%20of,K%2Dway%20Set%20Associative%20Mapping.

En la descripción de los diferentes mapeos se mencionan en algunos casos los llamados algoritmos de reemplazo (también conocidos como algoritmos de cache) en inglés “Cache replacement policies”, estos algoritmos se utilizan cuando hay un miss, la memoria cache está llena y se desea almacenar en ella un bloque de la memoria principal para determinar en cúal linea de la memoria debe almacenarse este bloque, reemplazando al bloque que se encontraba en esa línea. Los algoritmos de reemplazo más comúnmente utilizados son los siguientes:

- **FIFO (First In, First Out):** Este algoritmo es sencillo de programar, ya que solo consiste en que el nuevo bloque reemplaza la línea que lleva más tiempo en la memoria cache, como si se tratara de una estructura de datos de tipo cola, donde el primer bloque que llegó, será el primero en ser reemplazado. Este algoritmo no es eficiente ya que podría darse el caso de que el primer bloque en ser guardado en la memoria cache sea utilizado con mucha frecuencia, y reemplazarlo significa que muy pronto habría otro miss y habría que buscarlo de nuevo para traerlo de vuelta reemplazando otra línea que podría ser muy frecuentemente utilizada también.
- **LFU (Least Frequently Used):** Este algoritmo requiere llevar un control de que tanto han sido utilizadas las líneas de la memoria cache, y cuando llegue el momento de realizar un reemplazo por un miss, se elige la línea que ha sido menos utilizada.
- **LRU (Less Recently Used):** Muy similar al algoritmo LFU, también debe llevar un control no de cuánto han sido utilizadas las líneas de la memoria, si no de qué tan recientemente, así cuando se desea reemplazar una línea, se coloca en lugar de la línea que lleva más tiempo sin haber sido utilizada.

- **Random:** Como lo expresa su nombre, este algoritmo elige una linea aleatoria cuando se presenta un miss, y esta es la que se ve reemplazada por el nuevo bloque de la memoria principal, según Smith A.J., Goodman J.R. Instruction Cache Replacement Policies and Organizations. IEEE Transactions on Computers vol C-34 pp 234-241 (march 1985) en simulaciones experimentales, este algoritmo ha dado mejores resultados que FIFO y que LRU.

Fuentes: https://searchstorage.techtarget.com/definition/cache-memory,
Estructura de Computadores, Facultad de Informática, UCM, Curso 11-12 , https://sites.google.com/site/copolinformaticabi2015/tema-2/arquitectura-de-computadores/2-1-3,
https://www.gatevidyalay.com/cache-mapping-cache-mapping-techniques/#:~:text=Cache%20mapping%20is%20a%20technique%20that%20defines%20how%20contents%20of,K%2Dway%20Set%20Associative%20Mapping.
https://www.geeksforgeeks.org/cache-memory-in-computer-organization/,
https://www.sciencedirect.com/topics/computer-science/direct-mapped-cache,
https://studylib.es/doc/4440976/algoritmos-de-reemplazo-en-memorias-cache,
https://codifica.me/memorias-cache-y-algoritmos-de-reemplazo/

Algunos ejemplos donde se puede visualizar la memoria cache.

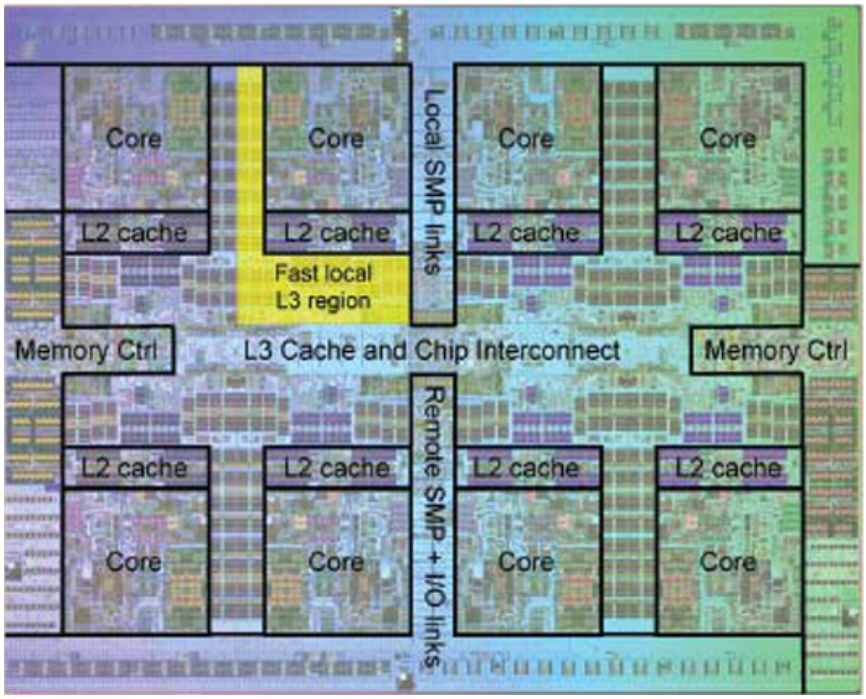

Imagen #1: IBM Power 7. L1 64 KB (32 KB para instrucciones y 32 KB para datos) Embebido a cada núcleo. L2 256 KB separado del núcleo pero hay una para cada núcleo. L3 32 MB compartida por los distintos núcleos.

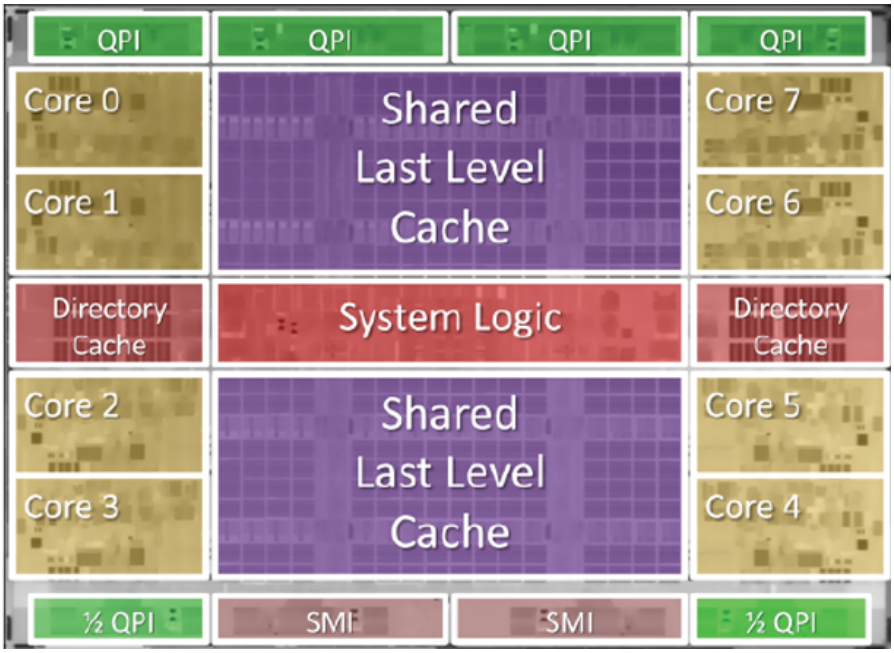

Imagen #2 Intel Poulson, L1 32 KB (16KB para instrucciones, 16 KB para datos) embebida al núcleo. L2 768 KB (512 KB para instrucciones, 256 KB para datos) embebida al núcleo. L3 32 MB compartida por los núcleos.

Referencias

https://www.britannica.com/technology/cache-memory
https://searchstorage.techtarget.com/definition/cache-memory
https://www.hp.com/us-en/shop/tech-takes/what-is-cache-memory
https://www.profesionalreview.com/2019/05/02/memoria-cache-l1-l2-y-l3/#Memoria_cache_L1
https://louwrentius.com/understanding-storage-performance-iops-and-latency.html
https://www.tvtechnology.com/miscellaneous/seeking-hard-disk-drive-latency
https://www.crucial.com/articles/about-memory/difference-between-speed-and-latency
https://www.bbc.co.uk/bitesize/guides/zhppfcw/revision/2
https://sites.google.com/site/copolinformaticabi2015/tema-2/arquitectura-de-computadores/2-1-3
https://www.gatevidyalay.com/cache-mapping-cache-mapping-techniques/#:~:text=Cache%20mapping%20is%20a%20technique%20that%20defines%20how%20contents%20of,K%2Dway%20Set%20Associative%20Mapping
https://www.geeksforgeeks.org/cache-memory-in-computer-organization/
https://www.sciencedirect.com/topics/computer-science/direct-mapped-cache
https://studylib.es/doc/4440976/algoritmos-de-reemplazo-en-memorias-cache
https://codifica.me/memorias-cache-y-algoritmos-de-reemplazo/
https://www.geeksforgeeks.org/types-of-cache-misses/