Arquitectura de computadores: caché

La memoria caché permite aumentar la velocidad de nuestros programas, pues la memoria RAM es muy lenta. En esencia, la caché es una memoria intermedia, más cercana a la CPU, que permite guardar temporalmente porciones de la memoria principal (bloques) mientras son usadas y así disminuir los accesos a la memoria principal.

1 Mapeo de cache

Se refiere a las formas que existen de interpretar las direcciones en memoria para situar sus bloques en la cache.

En una cache de N lineas y un tamaño de bloque S donde N y S son potencias de 2 se tienen los siguientes tipos de mapeo:

1.1 Mapeo directo

name	tag	line	word
example	FF	6543	21
size	remaining	$\log_2 N$	$\log_2 S$

En este mapeo, **Word** define la posición de un dato dentro del bloque. **Line** la linea de cache fija donde la dirección de memoria debería situarse, y **Tag** sirve como un diferenciador. Este mapeo no permite un mejor uso de la cache con políticas de reemplazo.

1.2 Mapeo totalmente asociativo

name	tag	word
example	FF6543	21
size	remaining	$\log_2 S$

En este mapeo solo encontramos a **Word** y **Tag**, lo cual permite una asignación dinámica de lineas con políticas de reemplazo, aunque es muy costoso de fabricar.

1.3 Mapeo asociativo por conjuntos

Considere a E como el número de conjuntos donde E es potencia de 2.

name	tag	set	word
example	FF654	3	21
size	remaining	$\log_2 \frac{N}{E}$	$\log_2 S$

Este mapeo ofrece un balance entre las alternativas anteriores. Se particiona la cache en E conjuntos. Cada bloque de memoria puede pertenecer solo a un conjunto fijo, pero la posición dentro del mismo es dinámica con la **Tag**. El numero de lineas por conjunto $\frac{N}{E}$ denota las **Vías** asociativas.

Más información sobre mapeos en Geeksforgeeks

2 Políticas de reemplazo

2.1 FIFO o Round Robin

Asigna lineas de cache con un contador circular (Wraps on overflow). Es fácil de implementar, pero no tiene en cuanta el uso de cada linea.

2.2 LRU o Least Recently Used

Descarta lineas de cache basándose en una prioridad que denota qué tan usadas fueron. Es eficiente, aunque necesita muchos bits de control y por ende es muy costosa.

2.3 Aleatoria

Su efectividad depende de la suerte, aunque entre más lineas de cache, es menos probable descartar las más importantes.

3 Políticas de escritura

3.1 [Hit] Write through

Escribe en la cache y memoria principal en paralelo. Esto aumenta el tráfico en el BUS de datos.

3.2 [Hit] Write back

Se escribe todo en la cache y solo se actualizan los datos en memoria principal cuando el bloque va a ser descartado. Requiere un bit de control **dirty**.

3.3 [Miss] Write allocate

Trae a la cache el bloque solicitado y escribe.

3.4 [Miss] No write allocate

Se escribe en memoria principal sin traer el bloque a cache.

4 Políticas de captación

4.1 Demand

Cuando hay un miss en la lectura, trae el bloque solicitado a la cache.

4.2 Prefetch Always

Cuando haya una lectura traerá los bloques cercanos en hits y fallos.

4.3 Prefetch miss

Solo trae los bloques cercanos en fallos