Memoria Cache

Che cos'è la cache?

La cache è una piccola memoria veloce posta tra la CPU e la RAM e permette di ottenere tempi di accesso alla memoria ridotti senza rinunciare alla possibilità di avere una memoria di grandi dimensioni. L'obiettivo della cache è ridurre i tempi d'accesso alla memoria introducendo uno o più livelli di cache tra la CPU e la RAM.

I processori moderni effettuano una quantità travolgente di richieste al sistema di memoria, sia in termini di latenza (il ritardo nel fornire un operando) sia in termini di banda (la quantità di dati fornita per unità di tempo). Questi due aspetti che caratterizzano un sistema di memoria sono purtroppo in larga parte in conflitto. Una delle tecniche più efficaci per migliorare sia la larghezza di banda sia la latenza è l'uso della cache.

Prima tecnica: cache separata

Una tecnica elementare che funziona in modo efficace consiste nell'introdurre 2 cache separate, una per le istruzioni e una per i dati. Questo sistema viene chiamato **a cache separata** e fornisce dei vantaggi. In primo luogo è possibile far partire le operazioni di memoria per ciascuna cache, raddoppiando effettivamente la larghezza di banda del sistema di memoria. Ed è per questo motivo che ha senso fornire 2 porte distinte per la memoria come abbiamo fatto per il MIC-1; infatti ogni porta ha la propria cache. Da notare che le 2 cache hanno accessi indipendenti alla memoria centrale.

Esistono però sistemi di memoria ancora più complicati.

Seconda tecnica: cache di secondo livello

Spesso tra la memoria centrale e le cache di istruzioni e dei dati è presente un'altra cache chiamata cache di secondo livello. In realtà si possono avere cache anche a 3 o più livelli di cache.

Principio località dei riferimenti

L'introduzione delle cache è efficace solo se i dati o le istruzioni necessari per proseguire nell'esecuzione vengono molto spesso trovati in cache. Questa condizione è vera grazie a 2 proprietà che caratterizzano l'esecuzione dei programmi:

la località spaziale che afferma se al tempo 't' si è fatto accesso ad un dato o ad una istruzione che si trova in una cella di memoria 'x' allora è molto probabile che al tempo 't+D' si faccia accesso ad una cella di memoria vicina a 'x'.

la località temporale che afferma se al tempo 't' si è fatto accesso ad un dato o ad una istruzione che si trova in una cella di memoria 'x' allora è molto probabile che al tempo 't+D' si faccia accesso di nuovo alla cella di memoria 'x'.

Prestazione memoria cache

Consideriamo per semplicità il caso di una sola cache. Ogni volta che il processore esegue un accesso alla memoria, la cache:

1)intercetta l'indirizzo;

2)controlla se la parola in quel blocco della memoria centrale è presente nella cache;

Se è presente estrae la parola dal blocco e la fornisce alla CPU al posto della memoria principale (HITCACHE), altrimenti (MISSCACHE) ci sono due casi:

1)si accede alla memoria e si carica l'intero blocco mancante nella cache e poi si fornisce la parola richiesta.

2)si accede alla memoria e si fornisce subito la parola richiesta, e poi so carica il blocco della cache.

Tempo medio di accesso in memoria per la CPU " t_m =hC*(1-h)*M."

M=tempo d'accesso alla memoria centrale quando il dato non è presente in cache

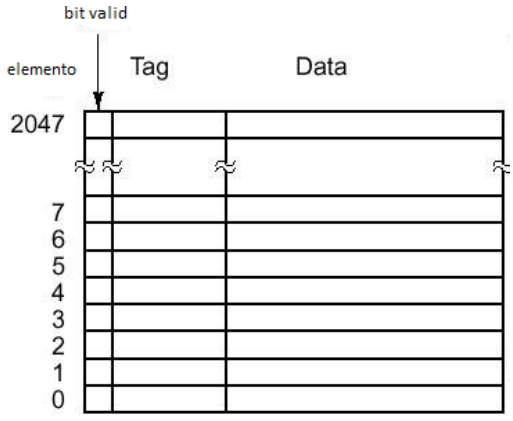
C=tempo d'accesso in cache; h= hit ratio della cache.

Modello struttura cache

La memoria è divisa in blocchi di dimensione fissa chiamati **linee di cache**. Una linea di cache è composta da 4 a 64 byte consecutivi. Le linee sono numerate consecutivamente a partire da 0.

Cache a corrispondenza diretta

La cache più semplice è la cache a corrispondenza diretta.



Ciascun elemento(riga) della cache può memorizzare esattamente una linea di cache della memoria centrale. Se ipotizziamo che la linea di cache abbia dimensione di 32 byte, la cache può memorizzare 2048 elementi di 32 byte, per un totale di 64 KB. Gli elementi della cache sono composti da 3 parti: il bit valid, campo Tag e il campo Data.

Questo è un esempio di cache che contiene 2048 elementi.

1)il bit valid che indica se il dato nell'elemento è valido oppure no. All'avvio del sistema tutti gli elementi della cache sono marcati come non validi.

2)il campo Tag che è un valore univoco a 16 bit, corrispondente alla linea di memoria da cui provengono i dati.

3)il campo Data che contiene una copia del dato della memoria. Questo campo memorizza una linea di cache di 32 byte.

Nella cache a corrispondenza diretta una data parola di memoria può essere memorizzata in un'unica posizione della cache. Per ogni indirizzo di memoria esiste un solo posto all'interno della cache in cui cercare. Per memorizzare e prelevare dati dalla cache l'indirizzo è diviso in 4 campi:

1)**il campo TAG** corrisponde ai bit Tag memorizzati in un elemento della cache;(16 bit)

2)**il campo LINE** indica l'elemento della cache contenente i dati corrispondenti, se sono presenti; (11 bit)

3)**il campo WORD** indica a quale parola si fa riferimento all'interno della linea; (3 bit)

4)**il campo BYTE** non viene generalmente usato, ma se si richiede un solo byte esso indica quale byte è richiesto all'interno della parola. (2 bit)

Quando la CPU genera un indirizzo di memoria, l'hardware estrae dall'indirizzo gli 11 bit del campo LINE e li usa come indice all'interno della cache per cercare uno dei 2048 elementi. Se questo elemento è valido si effettua un confronto tra il campo TAG dell'indirizzo di memoria e il campo Tag dell'elemento della cache. Se sono uguali, l'elemento della cache contiene la parola richiesta; questa situazione è chiamata **hit cache**. In tal caso la parola che si vuole leggere può essere presa direttamente dalla cache, evitando di dover andare fino alla memoria.

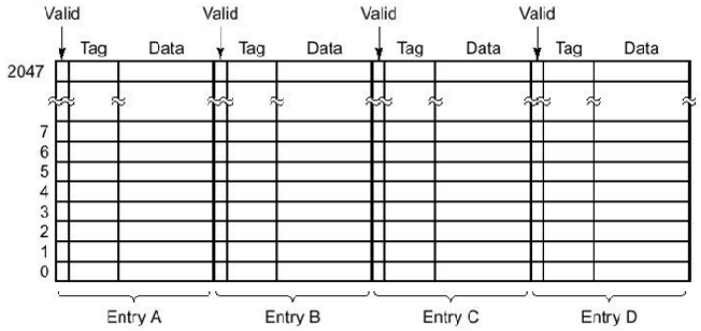
Se invece l'elemento della cache non è valido oppure i due campi ''tag'' non sono uguali, allora il dato richiesto non è presente nella cache e questa situazione è chiamata **cache miss**. In questo caso la linea della cache a 32 byte viene prelevata dalla memoria e memorizzata nell'elemento appropriato della cache, sostituendo quello che era presente precedentemente. In ogni caso se l'elemento della cache era stato modificato dopo averlo caricato, occorre riscriverlo in memoria prima di poterlo sovrascrivere.

Tuttavia all'interno della cache non è possibile memorizzare allo stesso tempo 2 linee i cui indirizzi differiscono di 64KB (65536 byte). Per esempio, se un programma accede a dati che si trovano nella locazione X e successivamente esegue un'istruzione che richiede dati presenti alla locazione X+65536(o in qualsiasi altra locazione della stessa linea), la seconda istruzione obbligherà a ricaricare l'elemento della cache, sovrascrivendo quello precedente. E se ciò avviene con una certa frequenza la cache potrebbe addirittura peggiorare le prestazioni peggio di quando non ci fosse la cache.

Cache set-associative a n-vie

Com'è stato detto precedentemente, molte linee della memoria sono in competizione per l'utilizzo di uno stesso elemento della cache. Se un programma usa la cache a corrispondenza disegnata sopra che usa intensamente le parole comprese tra l'indirizzo 0 e l'indirizzo 65536, si verificheranno costantemente dei conflitti e ogni riferimento alla memoria potrebbe potenzialmente espellere un elemento dalla cache. Una soluzione a questo problema è quello di consentire che in ciascun elemento della cache ci possano essere 2 o più linee. Una cache con n possibili elementi per ciascun indirizzo è chiamata cache set associativa a n vie.

Anche se l'insieme di elementi della cache da esaminare viene calcolato a partire dall'indirizzo di memoria, è necessario tuttavia controllare un insieme di n elementi per vedere se la linea richiesta è presente nella cache. Questa verifica deve essere effettuata molto velocemente. Le simulazioni hanno mostrato tuttavia che cache a 2 o a 4 vie garantiscono prestazioni sufficientemente buone, da rendere accettabile l'aumento dei componenti digitali richiesti. Ricorda è raro incontrare cache con più di 4 vie.



L'uso di una cache a set-associativa pone il progettista di fronte a una scelta: quando viene portata all'interno della cache una nuova linea, quale delle presenti deve essere scartata? In questi casi si usa un algoritmo chiamato LRU.

LRU(Least recently used)

Questo algoritmo mantiene un ordinamento temporale di ciascun insieme di elementi ai quali si può far riferimento in base a una certa locazione di memoria. Quando si accede a una delle linee presenti nella cache l'algoritmo aggiorna la lista marcando l'elemento utilizzato più di recente. Quando giunge il momento di sostituire un elemento, l'algoritmo scarta quello che si trova alla fine della lista, cioè quello alla quale è stato effettuato un accesso meno di recente.

Cosa accade quando si fa una scrittura in memoria?

Quando un processore scrive una parola, e questa parola è presente nella cache, ovviamente esso deve aggiornare la parola. Ma che cosa possiamo dire riguardo l'aggiornamento della copia nella memoria centrale? Questa operazione può essere rinviata finché la linea della cache non sia pronta a essere sostituita dall'algoritmo LRU.

Write-through→ è l'aggiornamento immediato nella memoria centrale dell'elemento della cache.

Questo approccio è in genere il più semplice, e il più affidabile, dato che il contenuto della memoria rimane sempre aggiornato. Purtroppo però questa tecnica richiede molto traffico verso la memoria; per questo motivo si tende a usare un'implementazione più sofisticata, la **write-back**.

Riguardo alle operazioni di scrittura occorre affrontare un ulteriore problema ed è cosa fare se si verifica una scrittura in una locazione che in quel momento non è memorizzata nella cache? Bisogna portare il dato nella cache o basta scriverlo in memoria? Anche in questo caso nessuna delle risposte è migliore dell'altra. Si tende ad usare la **write-allocation** che porta i dati nella cache quando si verifica un fallimento in scrittura. Al contrario quando si impiega la scrittura diretta, si tende generalmente a non allocare un elemento della cache a ogni operazione di scrittura, dato che ciò complicherebbe un'organizzazione altrimenti semplice. Le prestazioni della cache influenzano largamente le prestazioni generali del sistema.