Cache Teoria

GERARCHIA DI MEMORIA

Un programma non accede a tutte le istruzioni e tutti i dati con la stessa probabilità.
La gerarchia di memoria consiste in un insieme di livelli di memoria, ciascuno caratterizzato da una diversa velocità e dimensione.

+ veloce → + costo

Dopo vedremo il discorso meglio, nella struttura gerarchica.

PRINCIPIO DI LOCALITÀ

Un programma in un certo momento accede soltanto ad una piccola porzione del suo spazio di indirizzamento: **principio di località**. 2 tipi:

- **temporale**: dopo aver fatto riferimento a un elemento c'è tendenza a riferirsi allo stesso (cache con dimensioni maggiori la utilizzano per diminuire la frequenza miss)
- **spaziale**: dopo aver fatto riferimento a un elemento c'è la tendenza a riferirsi agli elementi vicini.

La cache utilizza più il primo del secondo.

STRUTTURA GERARCHICA

+ veloce → + vicino al processore → + costosa → + piccola
+ grande → + lontana → + economica → + lenta

HIT/MISS

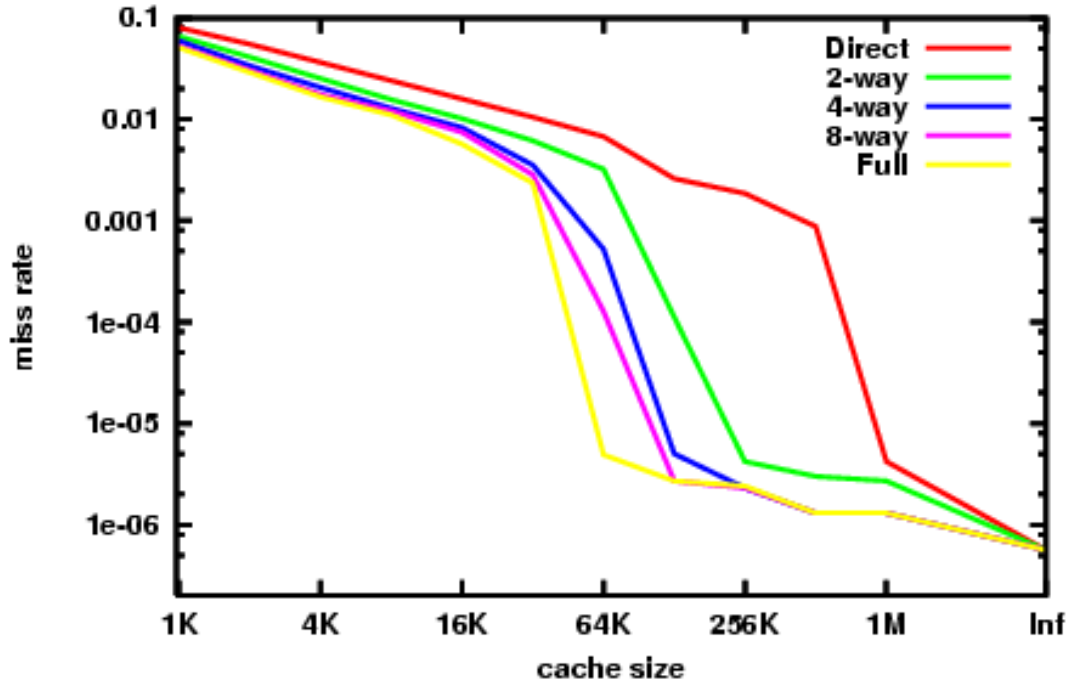
Quando il processore necessita di leggere o scrivere in una data collocazione in memoria principale, inizialmente controlla se il contenuto di questa posizione è caricato in cache.

Questa operazione viene effettuata confrontando l'indirizzo della posizione di memoria con tutte le etichette nella cache che potrebbero contenere il dato a quell'indirizzo.

Se il processore trova che la posizione di memoria è in cache, si parla di **hit** altrimenti di **miss**.

Il rapporto tra cache hit e accessi totali è chiamato anche **hit rate** ed è una misura indiretta dell'efficacia dell'algoritmo di cache. Un altra misura di efficienza è l'**hit time**, ovvero quanto ci mette la cache ad hittare.

Nel caso di un cache miss, ne consegue la creazione di una nuova entità,che comprende l'etichetta richiesta dal processore e una copia del dato nella memoria principale. Ciò comporta perdita di velocità, e quindi di efficienza.



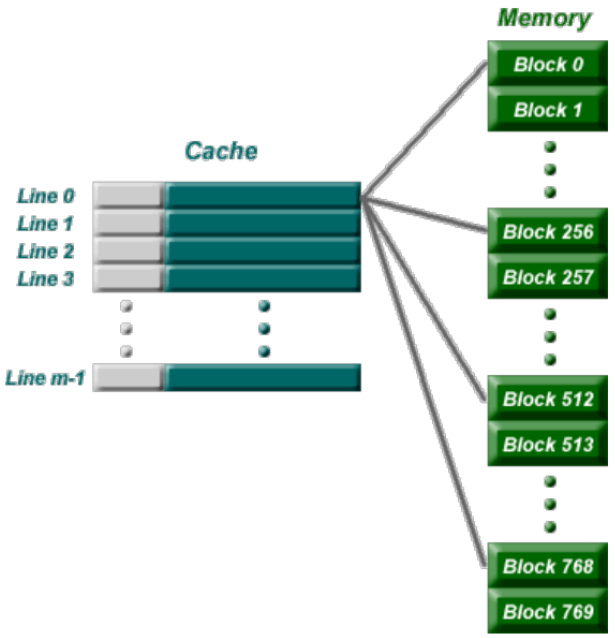
TIPI DI CACHE

Cache è un'area di memoria tra processore e memoria principale(Ram). Ci sono 3 tipi:

- **Mappaggio diretto**: a ciascun blocco della memoria corrisponde una locazione della cache.
- **Set Associative**:La cache associativa è un compromesso tra la cache a mappaggio diretto e la cache fully associative. Una cache set-associativa può essere immaginata come una matrice (n*m). La cache è divisa in 'n' set e ogni set contiene 'm' righe di cache. Un blocco di memoria viene prima mappato su un set e dopo inserito in qualsiasi riga della cache del set.
- **Fully associative**: Ogni blocco può essere ovunque, per cercalo bisogna cercare in tutta la cache (ricerca sequenziale sarebbe lenta; si effettua una ricerca in parallelo che però è costosa)

MAPPAGGIO DIRETTO

DIMOSTRAZIONE



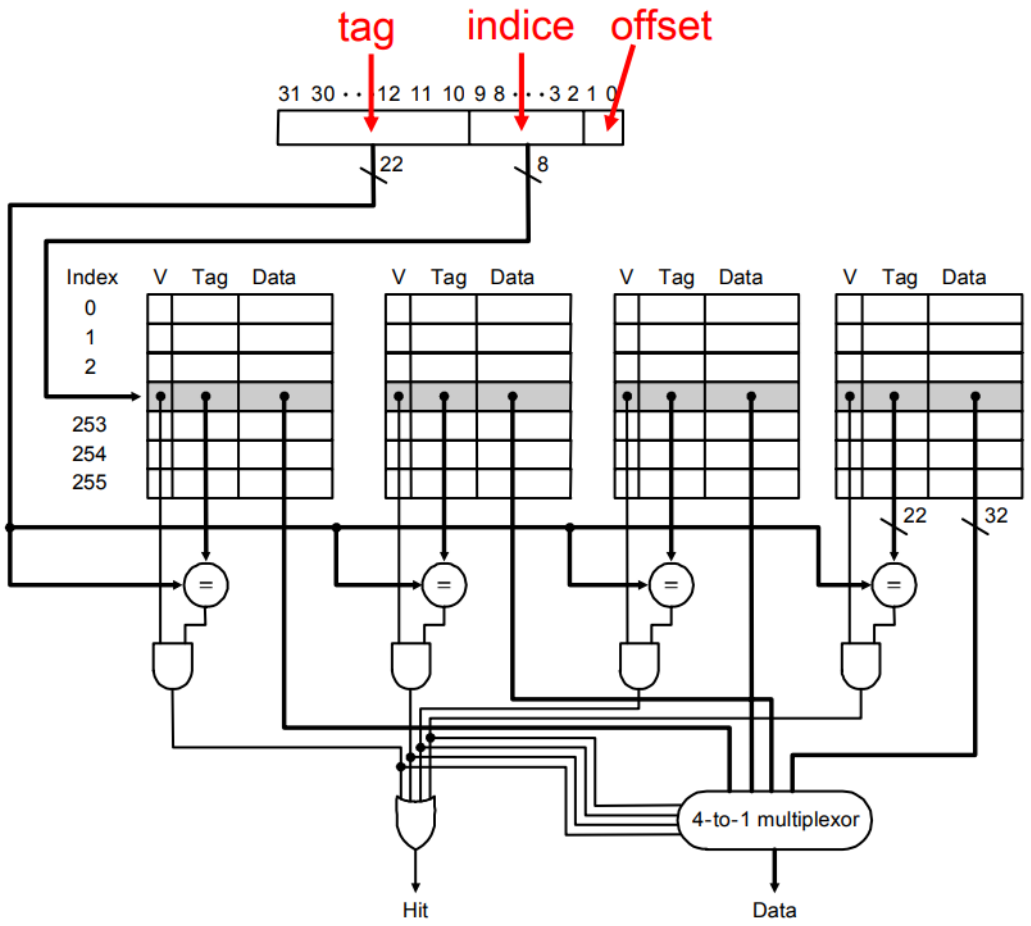
In questo caso, la cache ha 256 linee, quindi per trovare un blocco bisogna prendere l'indirizzo in memoria % numero blocchi=256. Tutti i multipli di 256, infatti , vanno nella linea 0; tutti i multipli di 256+1 vanno nella linea 1 ecc.

TECNICA DI INDIRIZZAMENTO

- **Trovare un blocco**: indirizzo blocco % numero blocchi
- **Tag**: è un'etichetta, verifica se una word della cache corrisponde a una word cercata.
- **Indice**: seleziona blocco cache
- **Offset**: Ha dimensione 2. Infatti 2 bit ci bastano per separare una word (4 byte). Così facendo, attraverso l'offset, possiamo ricavare un byte della word.
- **Dimensione campo tag** : 32 - bit indice - 2 (bit offset)

SET ASSOCIATIVE

DIMOSTRAZIONE



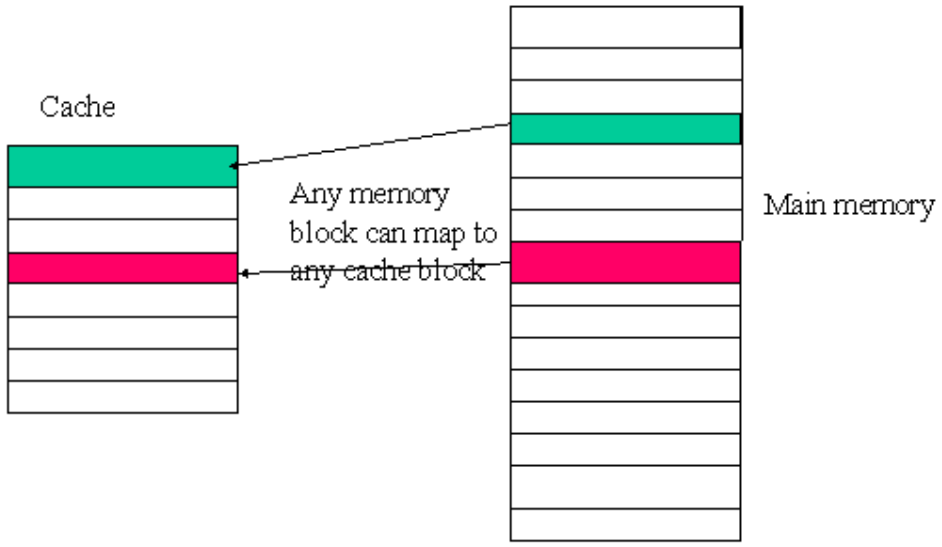
Questo è un esempio di Set associative, in questo caso 4 way set associative.
La cache è divisa in 4 set. Con lo stesso principio del Direct Mapped, per accedere a una linea della cache, bisogna fare il modulo. In questo caso però, abbiamo 4 sezioni per ogni indice, di conseguenza, ogni indice può essere linkato contemporaneamente a 4 indirizzi di memoria differenti; invece di 1 della Direct Mapped. Questo comporta una minor probabilità di Miss.

TECNICA DI INDIRIZZAMENTO

Blocchi della cache raggruppati in set.
Ogni indirizzo appartiene a un set. Per sapere se è presente bisogna confrontare in parallelo tutti i tag di tutti i blocchi del set
+associatività → + comparatori → + bit tag → - miss rate → + costo → + hit time (svantaggio)

FULLY ASSOCIATIVE:

DIMOSTRAZIONE



Ciascun blocco può essere posto in qualsiasi locazione della cache. Occorre cercarlo in tutte le celle, poiché esso può risiedere in una qualunque di esse. Per rendere conveniente la ricerca, essa avviene in parallelo (aumento costi, quindi va bene solo per cache con pochi blocchi).

POLITICHE SOSTITUZIONE

Per sostituire un blocco, dobbiamo spesso “affidarci” al nostro passato, con i principi di località, in quanto non possiamo sapere in futuro che blocchi ci serviranno.

- Random: quando bisogna sostituire si sceglie un blocco a caso. Non è sempre la scelta migliore, ma non essendoci politiche è quantomeno il più veloce.
- LRU: least recently used: sfruttando la località temporale si sostituisce il blocco che non si usa da più tempo.
- FIFO: si sostituisce il blocco più vecchio.

VERIFICA MISS

Quando si verifica un miss delle istruzioni nella cache si eseguono i passi:

1. Inviare Pc-4 alla memoria
2. Lettura memoria
3. Scrittura nella cache
4. Riavvio esecuzione dall’istruzione che ha causato miss

SCRITTURA DATO 3 TECNICHE

Per scrivere un dato bisogna aggiornare i livelli inferiori della gerarchia di memoria, il che implica stallo della cpu. 3 tecniche risolutive:

WRITE-THROUGH

Write-through: si scrivono i dati nel blocco + nel blocco inferiore.
Vantaggi: semplice, si mantiene coerenza delle informazioni
Svantaggi: operazioni di scrittura alla velocità del livello inferiore (- prestazione), + traffico

WRITE-BACK

Write-back: dati solo nel blocco in cui vogliamo scrivere. Il blocco viene spostato al livello inferiore solo quando deve essere sostituito.
Vantaggi: scritture veloci, solo le scritture successive alterano la cache
Svantaggi: le sostituzioni del blocco possono provocare trasferimento in memoria; incongruenza tra memoria corrente e memoria di livello inferiore. (finché non verrà riscritto il dato, nella ram rimane una ‘versione vecchia’ del dato).

WRITE THROUGH + WRITE BUFFER

Write-through+write buffer: scrittura su blocco e in write buffer, il quale viene gestito in fifo. 4 elementi nel buffer. Questa modalità risulta efficiente se la frequenza di aggiornamento del livello inferiore è più grande della velocità del buffer con cui aggiorna i dati.

WRITE-MISS

Le scritture possono indurre write-miss; 2 soluzioni:

1. Write allocate: blocco caricato in cache e si effettua scrittura (write back lo usa di solito)
2. No write allocate: blocco scritto nella memoria di liv. inferiore (write through)