

Memoria Cache

Introduzione

La velocità di un microprocessore è molto più alta di un qualsiasi sistema di memoria disponibile a costi ragionevoli. Ogni singolo accesso alla memoria principale di un microprocessore richiede più cicli macchina e quindi ne rallenta notevolmente le operazioni. Per risolvere questo problema si usano sistemi gerarchici di memoria (Figura 1).

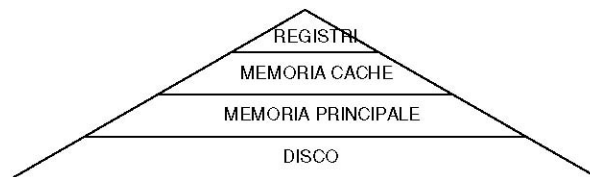
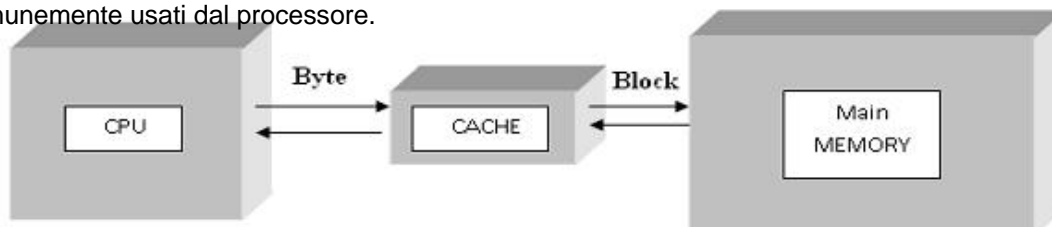


Figura 1 – Gerarchia di memoria.

Un sistema di memoria a livelli opera sul principio della **località dei riferimenti alla memoria**: un programma tende a accedere solo una piccola parte dello spazio di indirizzamento in un dato momento dell'esecuzione. Questo principio si basa su tre assunti:

- **località spaziale**: si assume che se un dato è stato recentemente referenziato è probabile che dati ad esso vicini siano presto referenziati.
- **località temporale**: si assume che se un dato è stato recentemente referenziato è probabile che sia referenziato ancora e presto.
- **sequenzialità**: si assume che gli accessi in memoria vengono fatti in sequenza rispetto all'accesso corrente.

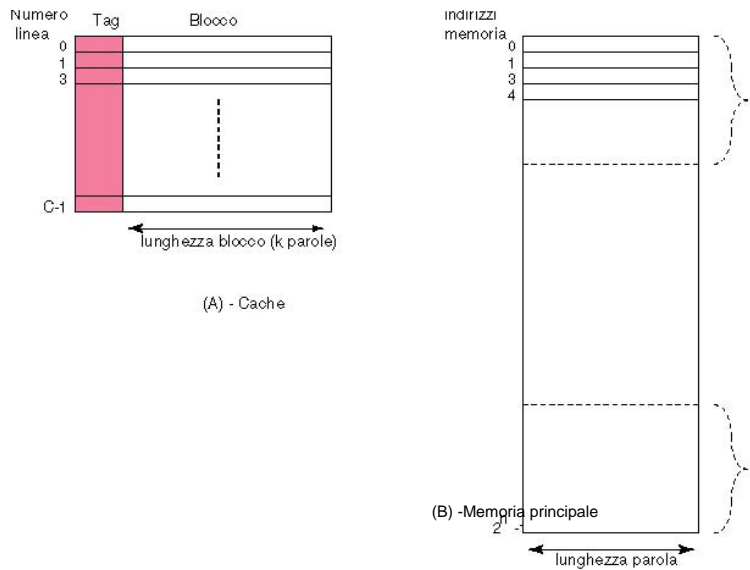
Sulla base di questo principio si opera in modo da usare memorie piccole ma che usano dati cui il processore accede più frequentemente. Un esempio di memoria piccola e veloce è dato dai registri del processore, essi sono situati all'interno del processore si possono accedere in modo molto veloce e gestiscono i dati che il processore usa più frequentemente. La gestione dei dati nei registri è uno dei problemi che vengono risolti durante la compilazione e la loro assegnazione rappresenta una grossa parte del lavoro del compilatore. Subito dopo i registri nella gerarchia della memoria si ha la memoria cache. La memoria cache è una memoria situata tra il processore e la memoria principale. È una memoria molto veloce che opera alla stessa velocità del processore ma che è molto più piccola della memoria principale. La funzione della memoria cache è di contenere le istruzioni e i dati più comunemente usati dal processore.



Principio di funzionamento

La memoria cache è composta di un certo numero di blocchi dette linee di k parole ciascuna. La memoria principale viene suddivisa in blocchi di k parole ciascuno, contenendo quindi un multiplo delle linee della cache. In ogni momento soltanto un sottoinsieme di tutti i blocchi della memoria risiede nelle linee della cache. Se viene richiesta la lettura di una parola in un blocco, il blocco viene trasferito in una delle linee della cache. Dal momento che vi sono più blocchi che linee, una singola linea non può essere unicamente associata

a un blocco particolare. Quindi ogni linea contiene un tag che identifica quale blocco è al momento contenuto in quella linea della cache. Il tag è una porzione dell'indirizzo di memoria.



– Struttura in blocchi della cache e della memoria

Il rendimento della cache è dato dal numero di volte che essa fornisce al processore il dato richiesto. Con **hit** si denota quando il dato cercato è contenuto nella cache, con **miss** quando il dato non è contenuto nella cache. I miss vengono a loro volta distinti in:

- Compulsory misses: la prima volta che si accede a un blocco che non è nella cache, quindi il blocco deve essere caricato nella cache, questi miss sono anche chiamati miss per partenza a freddo o **miss per primo accesso**.
- Capacity misses: questi miss avvengono quando la cache non può contenere tutti i blocchi che sarebbero necessari per l'esecuzione del programma corrente. Questi miss avvengono **per blocchi che sono stati scaricati** e successivamente caricati di nuovo.

Il tempo necessario per accedere alla cache quando si ha un hit viene chiamato hit time. Se il dato non è contenuto nella cache allora si deve trasferire dalla memoria principale e quindi si deve accedere ancora alla cache. Il tempo richiesto per effettuare queste operazioni è chiamato **miss penalty**. La percentuale delle volte che si ha un hit rispetto alle volte che si accede alla memoria è detto **hit ratio**. **Miss ratio** invece è dato da 1-hit ratio. Le prestazioni della cache sono misurate in hit ratio.

3 Dimensioni della cache

Il numero di hit dipende direttamente dalle dimensioni della cache, più è grande la cache più informazioni contiene e quindi vi è più probabilità che il processore trovi nella cache le informazioni che necessita in quanto meno locazioni della memoria principale avranno in comune le stesse linee della cache. L'aumento delle dimensioni della cache permette l'aumento delle prestazioni, ma non all'infinito, si raggiunge un punto in cui anche se si aumentano le dimensioni le prestazioni non aumentano. Questo è dovuto a uno di due possibili fattori: la cache è grande abbastanza da contenere le informazioni necessarie al programma che il processore sta eseguendo, il programma richiede dei dati i cui indirizzi non sono fra loro correlati. Quest'ultima possibilità avviene molto raramente. In generale la cache deve essere piccola abbastanza in modo che il suo costo non influenzi in modo sensibile il costo totale della memoria. Inoltre le dimensioni della cache influenzano il tempo di accesso e quindi si rischia di non avere i benefici dati dall'uso della cache. **Dimensioni tipiche della cache variano da 1k parole a alcune Mega parole.** Le prestazioni della cache sono molto sensibili al tipo di applicazioni che si hanno rendendo impossibile il progetto della cache ottima per qualsiasi applicazione.

4 Funzione di corrispondenza

La funzione di corrispondenza fornisce la **corrispondenza tra i blocchi della memoria principale e le linee della cache**. Dal momento che molti blocchi condividono una linea della cache, quando un blocco è trasferito dentro la cache un altro deve essere rimosso. La funzione di corrispondenza minimizza la probabilità che il blocco rimosso sia richiesto ancora molto presto. Vi sono tre tipi di funzioni di corrispondenza: **diretta, completamente associativa, associativa a n-vie o a gruppi.**

4.1 Corrispondenza diretta

Con la corrispondenza diretta **ogni blocco della memoria principale è assegnato a una specifica linea** della cache secondo la relazione:

$$i = j \bmod C$$

dove i è il numero della linea della cache assegnata al blocco j , e C è il numero di linee della cache. La funzione di corrispondenza diretta divide l'indirizzo di una parola in tre campi (Figura 3):

- Tag (gli s bit più significativi): un identificatore unico per ogni blocco. È memorizzato nella cache insieme con le parole dei dati della linea e specifica uno dei 2^s blocchi.
- Numero di linea (r bit) specifica quale linea contiene l'indirizzo referenziato. Il numero di linea identifica una delle $C = \text{linee}_2$ della cache.
- Parola (w bit): indica quale parola si sta indirizzando all'interno della linea composta da 2^w parole.

Per ogni accesso in memoria fatto dal processore, il controllore della memoria cache seleziona la linea della cache associata al blocco che contiene l'indirizzo del dato da accedere. Quindi verifica se il tag della linea corrisponde al blocco che si vuol accedere.

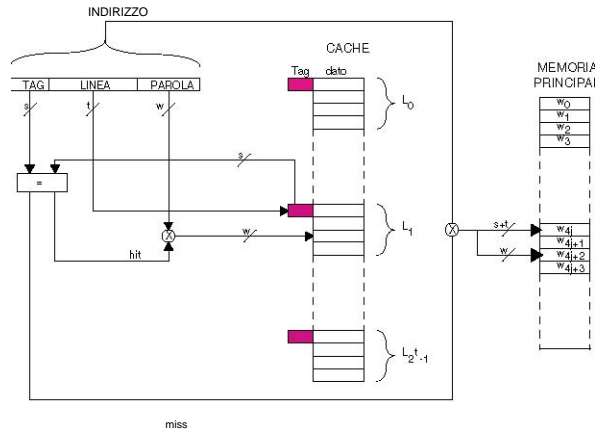


Figura 3 Struttura della cache a corrispondenza diretta.

La cache a corrispondenza diretta è la forma più semplice di cache. La sua realizzazione è semplice e relativamente poco costosa. Inoltre essendo molto semplice è anche molto veloce. Il maggiore svantaggio risiede nel fatto che ogni blocco di memoria corrisponde a una e solo una linea della cache. Per il principio di località è possibile che si faccia riferimento a blocchi che condividono la stessa linea della cache e questo risulterebbe in un continuo trasferimento di blocchi da e verso la cache, causando un serio degrado delle prestazioni. Questo tipo di fenomeno è raro in sistemi a singolo task ma può accadere spesso in sistemi multitasking. Questo tipo di cache è quello con minor prestazioni ma anche con minor costo di realizzazione.

4.2 Corrispondenza completamente associativa.

La funzione di corrispondenza completamente associativa permette a qualsiasi blocco della memoria principale di essere memorizzato in ogni linea della cache. L'indirizzo di un dato da accedere viene diviso in due campi: tag formato da s bit e offset formato da w bit (Figura 4).

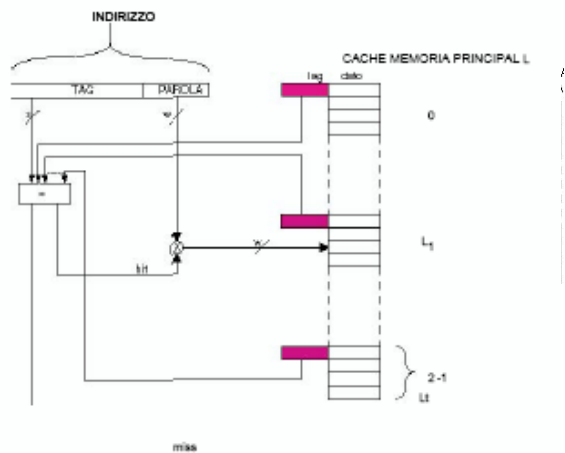


Figura 4 – Struttura della cache a corrispondenza completamente associativa.

Il controllore della cache deve verificare il tag di ogni linea per determinare se un dato è contenuto nella cache. Per limitare lo hit time è necessario fare questa verifica in parallelo per tutte le linee. A questo proposito si usa una struttura di memoria chiamata **associativa** (CAM, Content Addressable Memory) in quanto essa viene indirizzata per contenuto e non per indirizzo. In una memoria CAM ogni linea è divisa in due campi la chiave e il contenuto associato alla chiave. Quando si vuol accedere alla memoria si invia a essa la chiave e essa risponde con il contenuto. Per poter operare

in questo modo la memoria CAM contiene un comparatore per ogni chiave memorizzata in modo che quando arriva una chiave dall'esterno tutte le chiavi contenute nella CAM sono confrontate in parallelo con essa e la chiave che risulta uguale a quella in ingresso abilita la lettura del contenuto a essa associato. Nel caso della cache i tag sono le chiavi associate a ciascuna linea della cache. Il tag dell'indirizzo viene inviato alla cache, questa confronta in parallelo i tag di tutte le linee con quello in ingresso e se una linea risulta avere il tag uguale allora la lettura della parola all'interno della linea viene abilitata. L'indirizzo della parola all'interno della linea è dato dal campo offset dell'indirizzo. La funzione di corrispondenza completamente associativa è molto flessibile e elimina le principali debolezze della corrispondenza diretta. La memoria cache completamente associativa ha il miglior hit ratio dal momento che ogni linea della cache può memorizzare ogni indirizzo che necessita di essere posto nella cache. La cache completamente associativa ha dei problemi legati alla sua realizzazione che richiede un hardware specializzato e costoso. Inoltre è necessaria dell'altra logica per decidere quale blocco rimuovere se la memoria è piena. Quindi, sebbene la cache completamente associativa dia le migliori prestazioni essa è raramente utilizzata per il suo costo molto elevato.

4.3 Corrispondenza parzialmente associativa.

La funzione di corrispondenza parzialmente associativa prende i vantaggi dei primi due tipi di funzione. Una memoria cache con questo tipo di funzione unisce un algoritmo semplice di associazione paragonabile alla corrispondenza diretta, e una buona flessibilità della memoria derivante dalla memoria completamente associativa. La memoria cache parzialmente associativa è divisa in aree (v), chiamate sets, che funzionano in corrispondenza diretta, ogni set contiene un certo numero di linee (k). La cache è quindi denotata dal numero di linee contenute in ogni set. Se, per esempio, i set contengono X linee ciascuno, allora la cache è chiamata cache associativa a X vie. Un blocco di memoria j può venir memorizzato in ognuna delle k linee di un set n se

$$n = j \text{ mod } v$$

La memoria parzialmente associativa organizza l'indirizzo in tre campi: tag, set e parola. Essa può memorizzare contemporaneamente v diversi blocchi di memoria che hanno lo stesso tag (Figura 5).

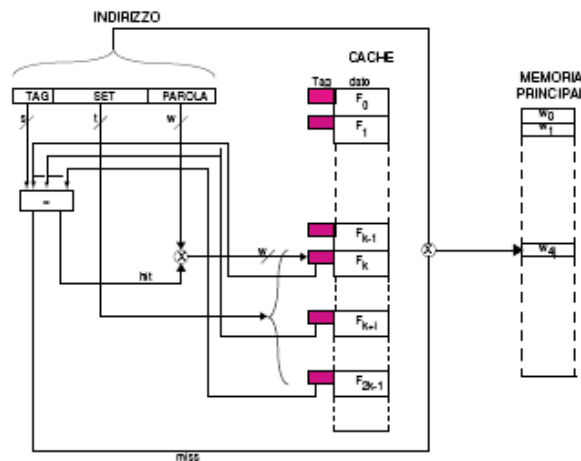


Figura 5 – Struttura della cache a corrispondenza parzialmente associativa.

Ognivolta che viene richiesto un accesso alla memoria, il controllore della cache deve confrontare il tag con quello di k linee della cache. Se una delle linee ha lo stesso tag allora si ha un hit. Altrimenti si ha un miss e una delle linee deve essere scelta per memorizzare il blocco che verrà trasferito dalla memoria.

In generale una cache parzialmente associativa ha un hit ratio migliore di una cache a corrispondenza diretta e è più veloce di una pienamente associativa.

5 Algoritmi per la scelta della linea da sostituire

Quando un blocco non viene trovato nella cache, esso deve essere trasferito dalla memoria principale e memorizzato in una delle linee della cache, è necessario quindi disporre di un algoritmo che scelga la linea dove memorizzare il nuovo blocco. In una memoria a corrispondenza diretta la scelta della linea è fissata, mentre nella cache parzialmente o pienamente associativa si può scegliere fra un insieme o fra tutti i blocchi. Vi sono diversi algoritmi che si possono adottare, fra questi i più usati sono:

- **LRU (Least Recently Used)**: si sceglie la linea della cache che memorizza il blocco che contiene dati che sono stati usati meno di recente. - **FIFO (First In First Out)**: si sceglie la linea che è stata scelta meno di recente.

- **LFU (Least Frequently Used)**: si sceglie la linea che contiene un blocco i cui dati sono stati meno usati.

- **Casuale**: si sceglie una qualunque tra le linee.

6 Politiche di scrittura della cache

Prima che una linea della cache possa essere sostituita, è necessario determinare se la linea è stata modificata. Il contenuto del blocco nella memoria centrale e la linea della cache che corrisponde a quel blocco sono copie uno dell'altra e quindi dovrebbero contenere gli stessi dati. Se la linea X della cache non è stata modificata da quando è stata memorizzata nella cache, allora non è necessario aggiornare il blocco di memoria principale corrispondente prima della sostituzione della linea. La linea della cache che arriva dalla memoria principale può sovrascrivere la linea della cache. D'altra parte, se la linea della cache è stata modificata, almeno un'operazione di scrittura è stata fatta sulla linea della cache allora il blocco corrispondente in memoria principale deve essere aggiornato. Vi sono le due differenti politiche che possono essere impiegate per accertarsi che il contenuto della memoria principale e della cache siano gli stessi: writethrough e writeback.

6.1 Politica Write-through

Supponendo di voler scrivere un dato e che si abbia un hit, cioè il dato è contenuto nella cache, le informazioni sono scritte immediatamente sia nella linea della cache che nel blocco nella memoria principale (con i relativi cicli di attesa). I vantaggi di questa tecnica sono che il contenuto della memoria centrale e della cache sono sempre consistenti, esso è semplice da realizzare e miss nella lettura non danno luogo a scritture nella memoria principale. D'altra parte, questa politica di scrittura ha uno svantaggio significativo, essa necessita di un accesso alla memoria principale, che è più lento e necessita di larghezza di banda di memoria superiore.

6.2 Politica Write-back

La politica writeback (a volte denominato posted write o copy back) ogni volta che vi è un hit di scrittura scrive soltanto nella cache. Ciò permette che il processore non abbia cicli di attesa dovuti alla scrittura nella memoria principale e possa immediatamente procedere nell'esecuzione del programma. La linea modificata della cache è scritta nella memoria principale soltanto quando è sostituita. Per evitare di riscrivere in memoria tutti i blocchi che sono sostituiti anche quando non sono stati scritti, si usa un bit chiamato dirty bit. Questo bit indica se il blocco è sporco (dirty) cioè è stato modificato mentre era nella cache, o pulito (non modificato). Se è pulito il blocco nel caso che debba essere sostituito, non è scritto in memoria principale. I vantaggi della politica write

back sono che le scritture si fanno alla velocità della cache, scritture multiple all'interno di un blocco richiedono soltanto una scrittura nella memoria principale, che provoca meno uso di larghezza di banda di memoria. Il writeback è un'alternativa più veloce alla politica di writethrough ma presenta uno svantaggio principale, i contenuti della cache e della memoria principale non sono consistenti. Questo problema viene chiamato problema della coerenza della cache ed è soggetto a un'ampia attività di ricerca. Un esempio di come la coerenza della cache influisce su un sistema di calcolo lo si ha quando viene un disco rigido e i dati sono trasferite nella memoria principale attraverso il sistema di DMA (accesso di memoria diretta), che non coinvolge il processore. Il controllore della cache deve costantemente controllare i cambiamenti fatti nella memoria principale e accertarsi che il contenuto della cache ricopi correttamente questi cambiamenti nella memoria principale. Ci sono molte tecniche, che sono state impiegate per permetter al controllore della cache di controllare la memoria principale, ma ancora una volta, queste aggiungono complessità e costo nella realizzazione della cache

6.3 Politiche del write on miss.

La scrittura in memoria presenta ancora un problema che non è presente nelle letture. Se viene richiesta una scrittura di un dato che non è presente nella cache (si ha un miss) allora si può decidere di seguire due politiche:

-Politica writeallocate: il blocco, cui appartiene il dato, viene portato nella cache e quindi il dato viene scritto all'interno della cache. -Politica writenoallocate: il dato viene aggiornato direttamente in memoria principale.

Normalmente le cache che usano il writethrough utilizzano il writenoallocate, mentre le cache che usano il writeback utilizzano il writeallocate.

6.4 Write Buffer

La scrittura in memoria può essere migliorata utilizzando la tecnica del writebuffer (Figura 6). Il processore scrive i dati sia nella cache che nel writebuffer (una piccola memoria) quindi il controllore della cache scrive il contenuto del writebuffer in memoria. Se usato insieme al writethrough, questo schema permette di non avere tempi di attese per scritture successive, inoltre si può accedere alla cache in parallelo mentre questa sta scrivendo i dati in memoria.

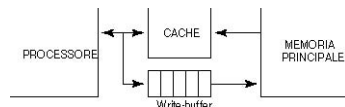


Figura 6 – Principio del write-buffer.

Dimensioni dei blocchi e delle linee

La dimensione in bytes delle linee della cache sono un altro fattore determinante per le prestazioni. Quando si ha una miss per la lettura di un dato, un intero blocco viene trasferito dalla memoria principale alla cache, quindi un certo numero di dati adiacenti a quello richiesto vengono anch'essi trasferiti. Se partiamo da un blocco con dimensioni piccole e via via ne aumentiamo la dimensione, vedremo che inizialmente il numero di hit aumenterà grazie al principio di località, ma poi, da quando il blocco raggiunge una certa dimensione, le hit cominceranno a diminuire. Questo fenomeno è dovuto al fatto che la probabilità di usare i dati all'interno di un blocco appena caricato è minore della probabilità di usare dati in blocchi abbiamo sostituito per far posto al nuovo blocco. Nel determinare la dimensione di blocchi nella cache si devono considerare due fattori:

-Se i blocchi sono di grande dimensione. pochi potranno essere contenuti nella cache e quindi la sostituzione dei blocchi aumenterà di frequenza, aumentando il numero delle miss. -Se un blocco ha grande dimensione, i dati in esso contenuti non sono vicini violando il principio di località.

La relazione tra dimensione del blocco e hitratio è complessa e dipende molto sulle caratteristiche di località di un particolare programma. Al momento non è stata trovata una dimensione ottima, nella pratica dimensioni tra 2 e 8 parole sembrano funzionare molto bene.

8 Uso di cache multiple

L'uso di più cache in un sistema di calcolo aiuta a migliorare le prestazioni. Le cache multiple possono essere usate in due modi a livelli e a parti.

8.1 Livelli di cache.

L'uso di più livelli di cache è usato per avere memorie cache molto vicine alla frequenza di lavoro del processore. Si ha quindi una gerarchia di cache dove la cache di primo livello (L1 cache) detta anche cache primaria è quella più veloce e più piccola. Essa è accessibile dal processore senza l'uso di bus esterni diminuendo in questo modo il traffico sul bus del processore e aumentando la velocità. La cache L1 opera alla stessa frequenza del processore. La cache di livello 2 (L2) è spesso chiamata secondaria. La cache L2 immagazzina una quantità maggiore di dati che spesso vengono trasferiti dalla cache L1 in multipli della dimensione della L1. La velocità di L2 è usualmente fra 4 e 16 volte più lenta di L1. Al contrario delle cache L1 e L2 la cache di terzo livello (L3) opera esternamente al processore e la sua velocità è paragonabile a quella della memoria principale. Cache divisa
L'organizzazione della cache può essere di tipo unificato o unito. Nel primo caso si utilizza una cache per memorizzare dati e istruzioni, nel secondo caso si utilizzano due cache diverse, una per i dati e una per le istruzioni. Una cache unica permette un hardware più semplice e meno costoso, essa inoltre permette un bilanciamento fra dati e istruzioni adattandosi maggiormente alle caratteristiche di un programma. Infatti, se un programma richiede più accessi ai dati che alle istruzioni (nel caso di un loop) la cache unica può memorizzare più dati che istruzioni, e il contrario se accade il viceversa. D'altra parte una cache divisa in cache per istruzioni e cache per dati permette di avere maggiori prestazioni in quanto il processore può accedere contemporaneamente alle memorie, questa situazione è molto importante per processori a alte prestazioni. Si possono anche usare organizzazioni miste. Per esempio si può usare un'organizzazione divisa per una cache L1 e un'organizzazione unita per la cache L2.