Miglioramento delle prestazioni

Aniello Minutolo

Memoria cache

Sommario

1. Memoria cache.
2. Cache a corrispondenza diretta.
3. Cache set-associative.

Sfida storica nella progettazione dei calcolatori

La progettazione dei calcolatori ha sempre affrontato la sfida di creare un sistema di memoria in grado di fornire operandi alla velocità di elaborazione della CPU

- uno dei problemi principali è il divario crescente tra la velocità dei processori e quella delle memorie;
- l'incapacità delle memorie di tenere il passo con la CPU ha sempre stimolato la ricerca di soluzioni per migliorare le prestazioni.

Conflitto tra latenza e banda

I sistemi di memoria devono gestire due aspetti critici:

- la **latenza**, ovvero il ritardo nel fornire un operando;
- la **banda**, cioè la quantità di dati forniti per unità di tempo.

Latenza e banda sono spesso in conflitto

- molte tecniche che permettono di aumentare la larghezza di banda incrementano allo stesso tempo la latenza;
- ad esempio, l'uso delle pipeline permette di gestire efficientemente la sovrapposizione di più richieste alla memoria, ma in cambio si ottiene anche una latenza maggiore per le singole operazioni di memoria.

Soluzione tramite cache

Una delle tecniche più efficaci per migliorare sia la larghezza di banda sia la latenza è l’uso di più memorie cache

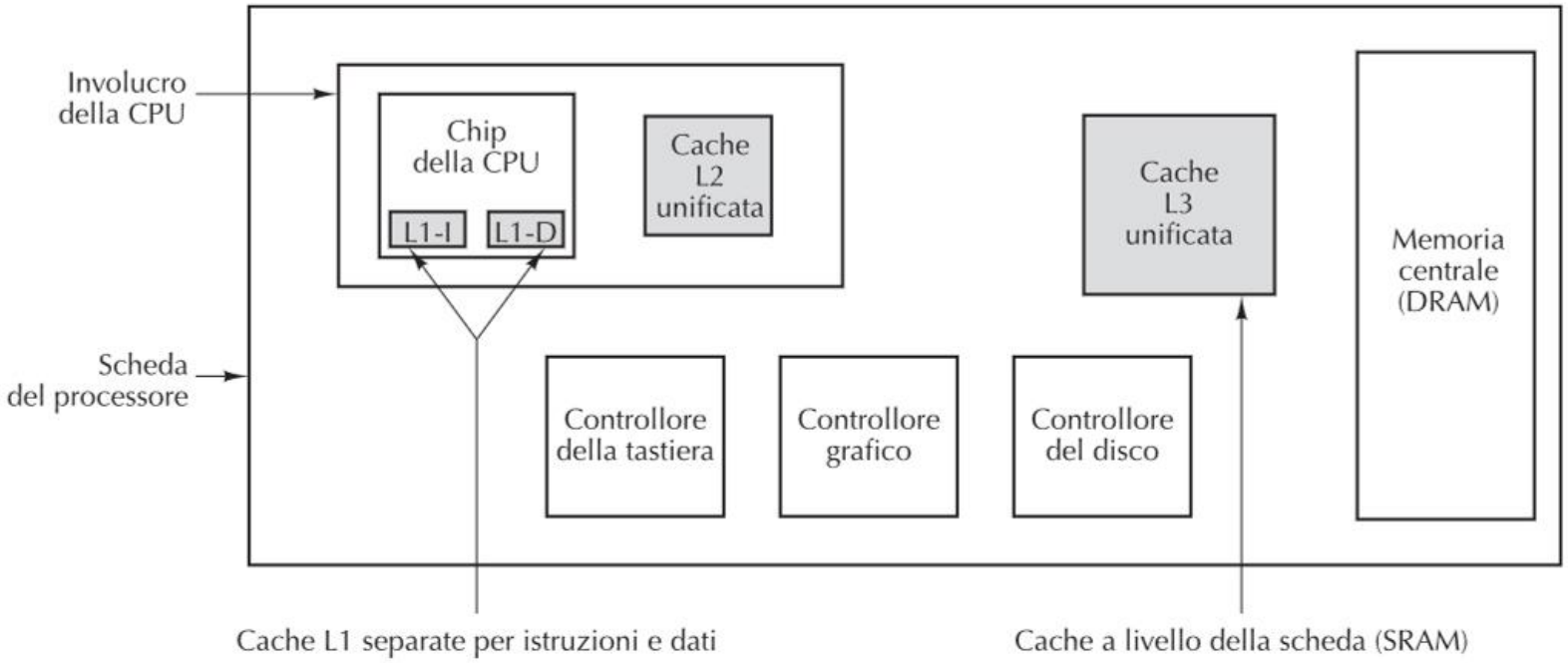
- se la cache contiene una percentuale sufficiente delle parole necessarie, l'effettiva latenza della memoria può essere ridotta in modo significativo.

Cache separate per istruzioni e dati

Un sistema con due cache distinte, una per le istruzioni e una per i dati, offre vantaggi significativi in termini di prestazioni:

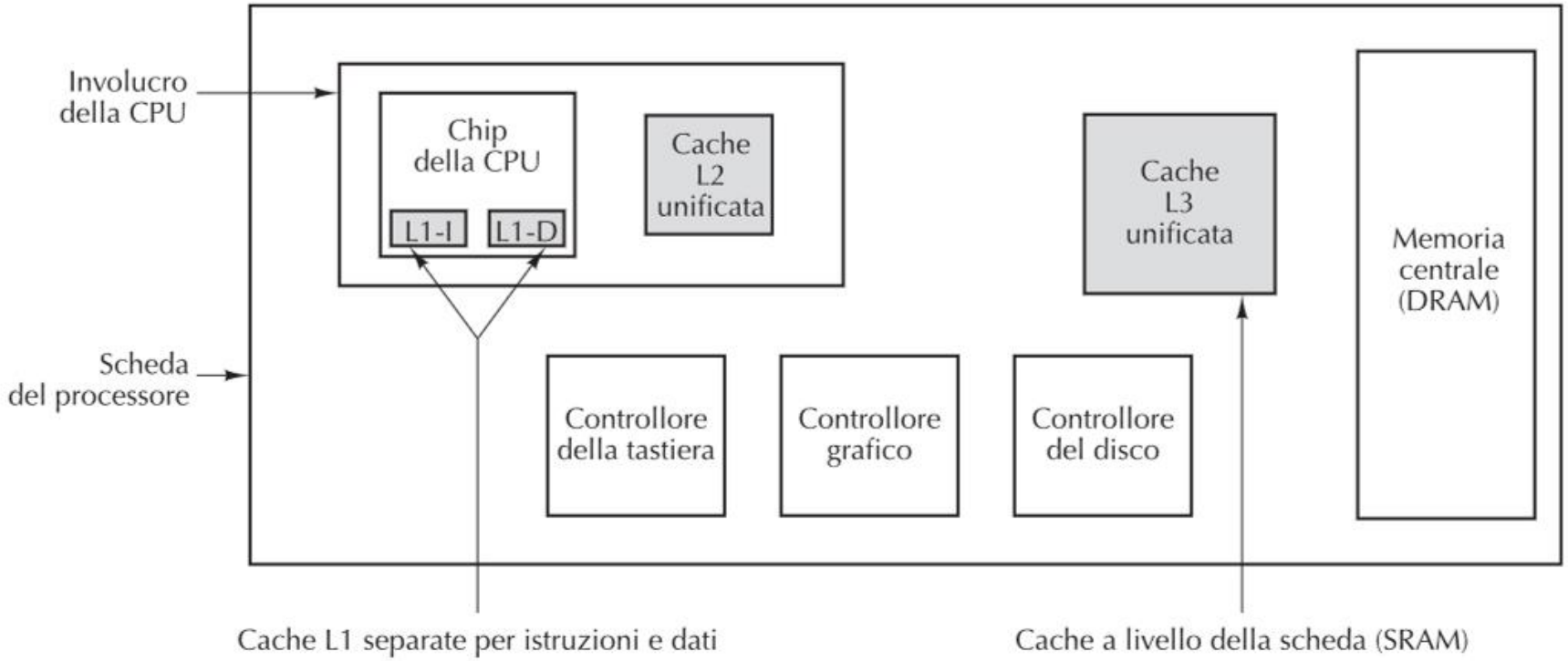
- le cache separate permettono operazioni di memoria indipendenti, raddoppiando effettivamente la larghezza di banda del sistema;
- questa architettura a **cache separata** è stata implementata in sistemi come Mic-1 con due porte distinte per la memoria;
- ogni porta ha la propria cache;
- le le due cache hanno accessi indipendenti alla memoria centrale.

Gerarchia di cache



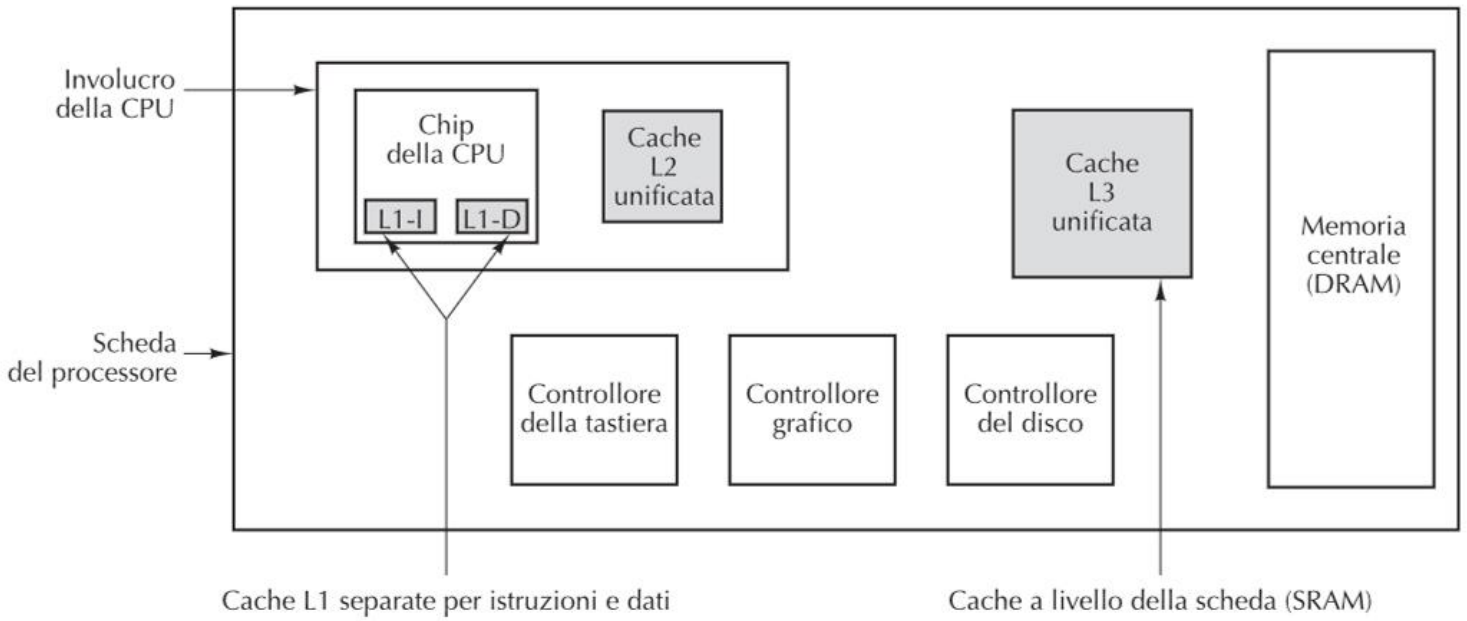
- I sistemi moderni spesso includono più livelli di cache, come cache di primo, secondo e terzo livello, per migliorare le prestazioni;
- la cache di primo livello L1 è integrata nel chip della CPU e ha dimensioni ridotte, in genere comprese tra 16 e 64 KB;
- la cache L1 è separata per istruzioni e dati.

- La cache di secondo livello L2 non si trova nel chip della CPU, ma può essere inclusa all'interno del suo involucro, vicina al chip e a esso collegata tramite circuiti ad alta velocità;
- la cache L2 ha dimensioni maggiori, generalmente tra 512 KB e 1 MB;
- la cache L2 è in genere unificata e contiene sia dati che istruzioni.



Di solito le cache sono annidate, nel senso che l'intero contenuto della cache L1 è compreso nella cache L2, che a sua volta è contenuta nella cache L3

Gerarchia di cache



- La cache di terzo livello L3 si trova sulla scheda del processore ed è costituita da alcuni megabyte di SRAM, un tipo di memoria molto più veloce rispetto alla memoria DRAM centrale;
- la cache L3 è in genere unificata e contiene sia dati che istruzioni.

Località spaziale e temporale

Per raggiungere il proprio scopo le cache sfruttano due tipi di località degli indirizzi:

- **località spaziale**, ovvero la tendenza ad accedere a locazioni di memoria vicine, cioè con indirizzi simili a quelli recentemente usati;
- **località temporale**, ovvero l'accesso ripetuto nel tempo alle stesse locazioni di memoria, come nel caso delle istruzioni di un ciclo.

Linee di cache

Tutte le cache utilizzano lo stesso modello

- la memoria centrale è divisa in blocchi di dimensione fissa chiamati **linee di cache**, generalmente composti da 4 a 64 byte consecutivi;
- le linee di cache sono numerate consecutivamente e la cache contiene un sottoinsieme di queste linee in ogni momento;
- se si verifica un **cache miss**, una linea viene sostituita con quella richiesta, prelevata dalla memoria centrale o da una cache di livello superiore.

L'idea base è di tenere il più a lungo possibile nella cache le linee usate più frequentemente, in modo da massimizzare il numero di riferimenti alla memoria che la cache riesce a soddisfare

Cache a corrispondenza diretta

Struttura della cache a corrispondenza diretta

Una **cache a corrispondenza diretta** è la forma più semplice di cache, dove ciascun elemento (riga) della cache può memorizzare esattamente una linea di cache della memoria centrale

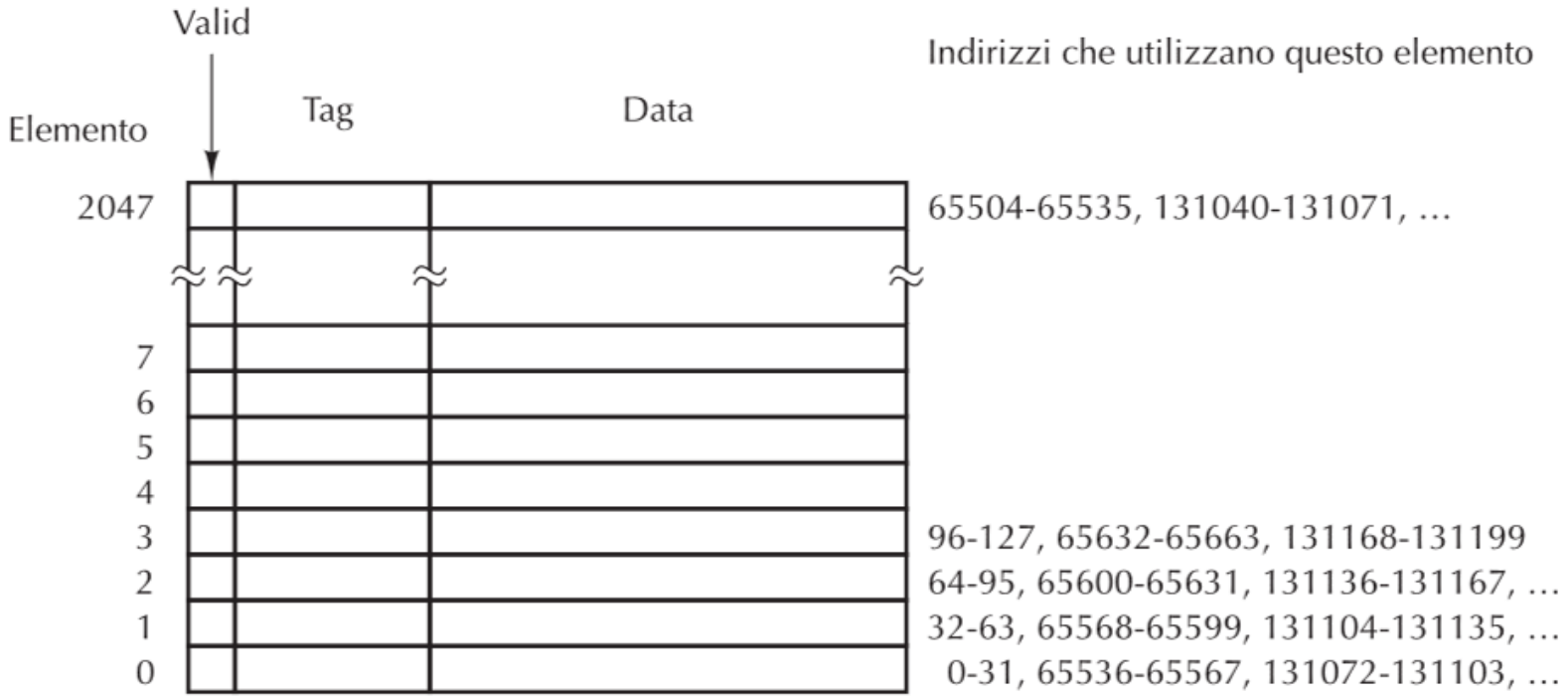
- una data parola di memoria può essere memorizzata in un'unica posizione della cache;
- per ogni indirizzo di memoria esiste un solo posto all'interno della cache in cui cercare il dato.

Se ipotizziamo che la cache contenga 2048 elementi, e che ogni linea di cache abbia dimensione di 32 byte, la cache può memorizzare un totale di 64 KB.

Struttura della cache a corrispondenza diretta

Gli elementi della cache sono composti da tre parti:

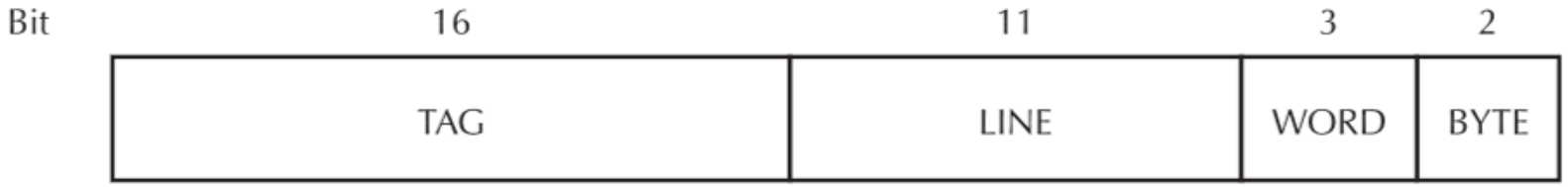
- il bit **Valid** che indica se il dato nell'elemento è valido oppure no. All'avvio del sistema tutti gli elementi della cache sono marcati come non validi.
- Il campo **Tag** che è un valore univoco a 16 bit, corrispondente alla linea di memoria da cui provengono i dati
- Il campo **Data** che contiene una copia del dato della memoria. Questo campo memorizza una linea di cache di 32 byte.



Divisione dell'indirizzo di memoria

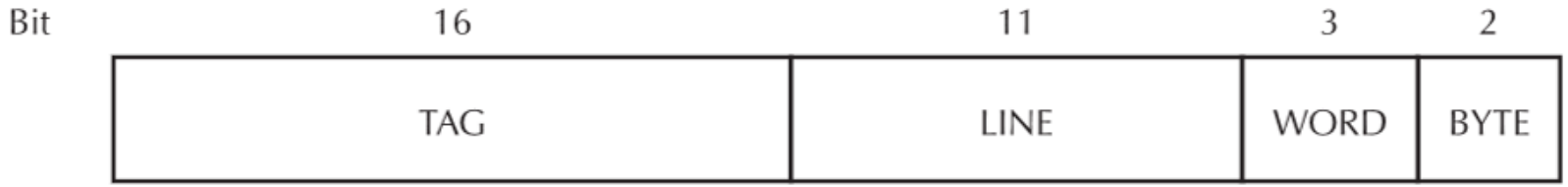
L'indirizzo di memoria è diviso in quattro campi: **TAG**, **LINE**, **WORD** e **BYTE**, ognuno con una funzione specifica nella gestione della cache:

- **TAG** corrisponde ai bit Tag memorizzati in un elemento della cache.
- **LINE** indica l'elemento della cache contenente i dati corrispondenti, se sono presenti.
- **WORD** indica a quale parola si fa riferimento all'interno della linea; 4. il campo BYTE non viene generalmente utilizzato, ma se si richiede un solo byte esso indica quale byte è richiesto all'interno della parola.



Cache hit e cache miss

- Quando la CPU genera un indirizzo di memoria
  - gli 11 bit del campo **LINE** sono usati come **indice** all'interno della cache per cercare uno dei 2048 elementi.
- Se l'elemento della cache è **valido** e se il campo **TAG** dell'indirizzo di memoria è **uguale** al campo **Tag** dell'elemento della cache
  - l'elemento della cache contiene la parola richiesta (**cache hit**);
  - dalla linea di cache viene estratta solamente la parola effettivamente richiesta, mentre non viene utilizzato il resto della linea.



Cache hit e cache miss

Se l'elemento della cache **non è valido** oppure se il campo **TAG** dell'indirizzo di memoria **è diverso** dal **Tag** dell'elemento della cache:

- il dato richiesto non è presente nella cache (**cache miss**);
- la linea della cache a 32 byte viene prelevata dalla memoria e memorizzata nell'elemento appropriato della cache, sostituendo quello che era presente precedentemente;
- se l'elemento della cache da sostituire era stato modificato dopo averlo caricato, occorre riscriverlo in memoria prima di poterlo sovrascrivere

Gestione delle collisioni

Nelle cache a corrispondenza diretta, due linee con indirizzi che differiscono di 64 KB o multipli possono collidere nello stesso elemento della cache (dato che il loro campo LINE è uguale).

Le collisioni possono degradare le prestazioni se si verificano frequentemente, costringendo a leggere continuamente dalla memoria un’intera linea di cache e non semplicemente di una parola:

- 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 + 65.536

L'uso di cache separate per istruzioni e dati può ridurre le collisioni, poiché le richieste vengono gestite da cache diverse.

Vantaggi delle cache a corrispondenza diretta

Le cache a corrispondenza diretta sono semplici ed efficienti, con collisioni che si verificano raramente nella maggior parte delle applicazioni:

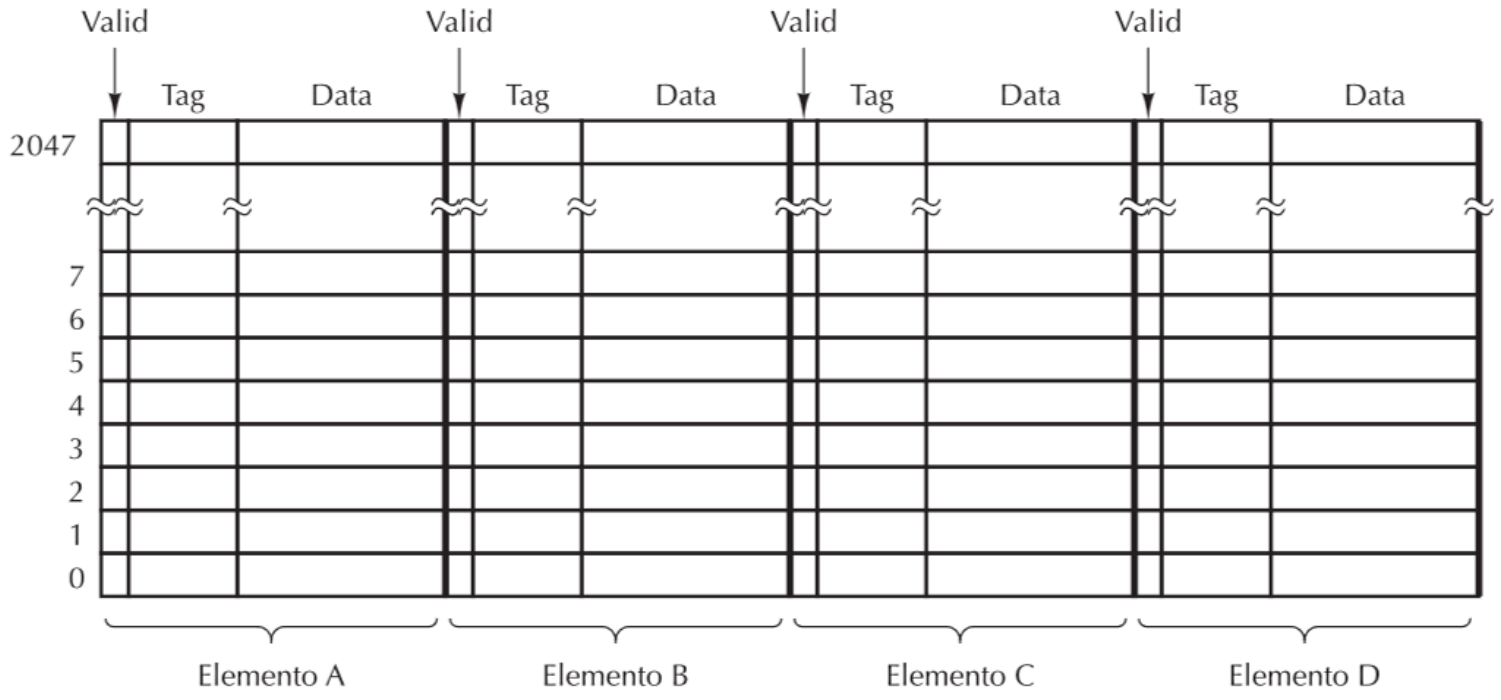
- un compilatore ottimizzato può posizionare dati e istruzioni in memoria per minimizzare le collisioni, migliorando ulteriormente le prestazioni.

L'accesso alla cache è estremamente veloce, poiché la posizione del dato è determinata immediatamente dall'indirizzo.

Cache set-associative

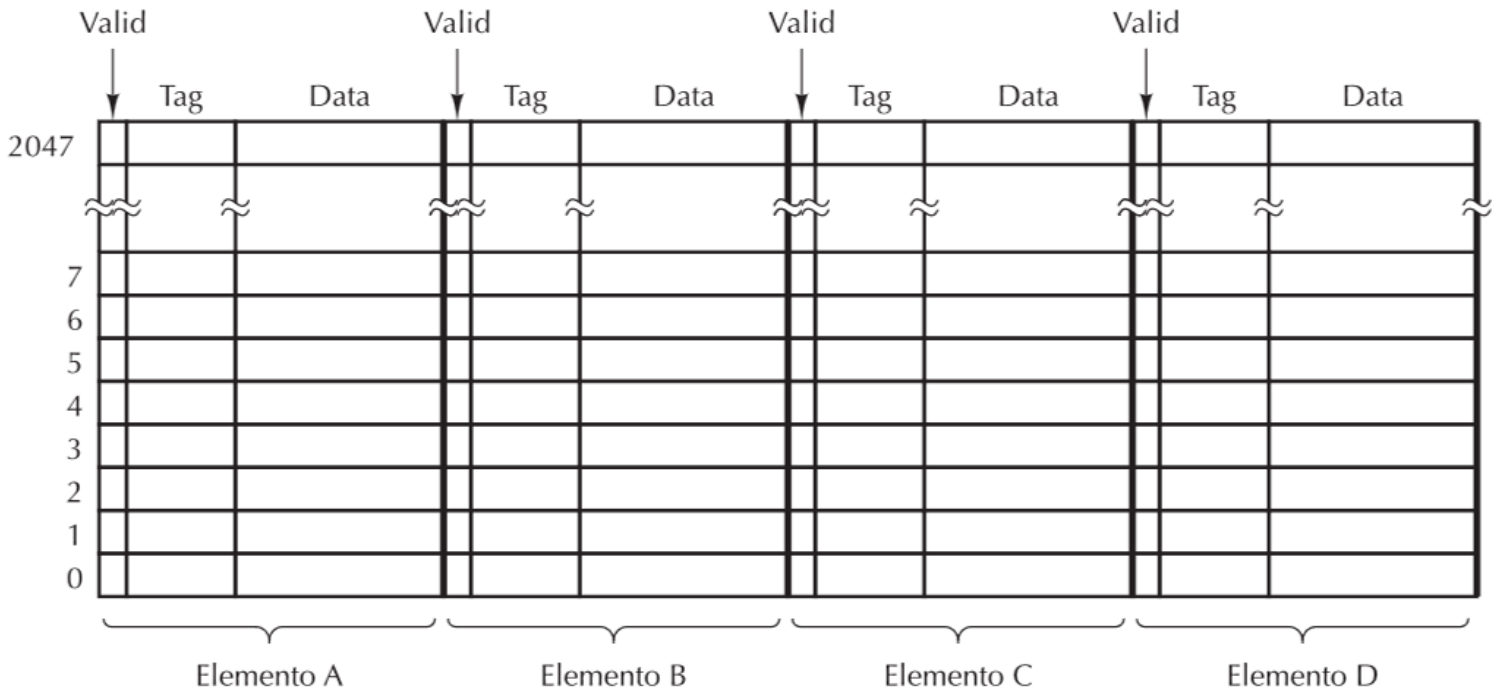
Introduzione alle cache set-associative

Una **cache set-associative** a n vie permette che per ciascun indirizzo di memoria esistano n possibili elementi



Introduzione alle cache set-associative

- Questa architettura risolve il problema delle collisioni frequenti nelle cache a corrispondenza diretta, migliorando le prestazioni;
- sono più complesse ma offrono prestazioni superiori in scenari con accessi ripetuti a indirizzi conflittuali;
- l'insieme di elementi della cache da esaminare viene calcolato a partire dall'indirizzo di memoria
    - è necessario controllare un insieme di n elementi per vedere se la linea richiesta è presente nella cache.



Algoritmo LRU

L'algoritmo **LRU** (Least Recently Used) è comunemente utilizzato per decidere quale linea sostituire in una cache set-associative:

- LRU mantiene un ordinamento temporale degli elementi, scartando quelli meno recentemente utilizzati quando si verifica un **cache miss;**
- l'implementazione di LRU diventa complessa in cache con molte vie, limitando l'uso di cache con più di quattro vie.

Cache completamente associative

Una **cache completamente associative** è un caso estremo dove tutte le linee competono per l'intero spazio della cache:

- in questa configurazione, ogni accesso richiede il confronto con tutti i tag presenti nella cache, aumentando la complessità hardware;
- le cache completamente associative sono raramente utilizzate a causa della loro complessità e del limitato miglioramento delle prestazioni.

Gestione delle operazioni di scrittura

Le operazioni di scrittura pongono sfide aggiuntive nelle cache, richiedendo strategie come **write-through** o **write-back**:

- il **write-through** aggiorna immediatamente la memoria centrale, garantendo coerenza ma aumentando il traffico verso la memoria;
- il **write-back** ritarda l'aggiornamento della memoria centrale fino alla sostituzione della linea, riducendo il traffico ma aumentando la complessità.

Write allocation

- La tecnica di write allocation prevede il caricamento del dato nella cache in caso di cache miss durante un'operazione di scrittura.
- Questa tecnica è vantaggiosa quando si verificano ripetute scritture sulla stessa linea, migliorando le prestazioni complessive.
- L'**write allocation** è spesso associata a politiche di **write-back**, mentre le politiche di **write-through** tendono a evitarla.

Importanza delle prestazioni della cache

- Le prestazioni della cache hanno un impatto significativo sulle prestazioni complessive del sistema, data la differenza di velocità tra CPU e memoria.
- La ricerca di strategie ottimali per la gestione delle cache è un campo di studio attivo, con continui miglioramenti nelle architetture.
- L'ottimizzazione delle cache è cruciale per sfruttare appieno le potenzialità dei moderni processori ad alta velocità.

Bibliografia

Libro di testo

- Andrew S. Tanenbaum e Todd Austin. Architettura dei calcolatori. Un approccio strutturale. 6/ED. Anno 2013. Pearson Italia spa. (Disponibile nella sezione “Biblioteca”)

Fonte argomenti e immagini

- Capitolo 4, Paragrafo 4.5.1, pp. 314-321