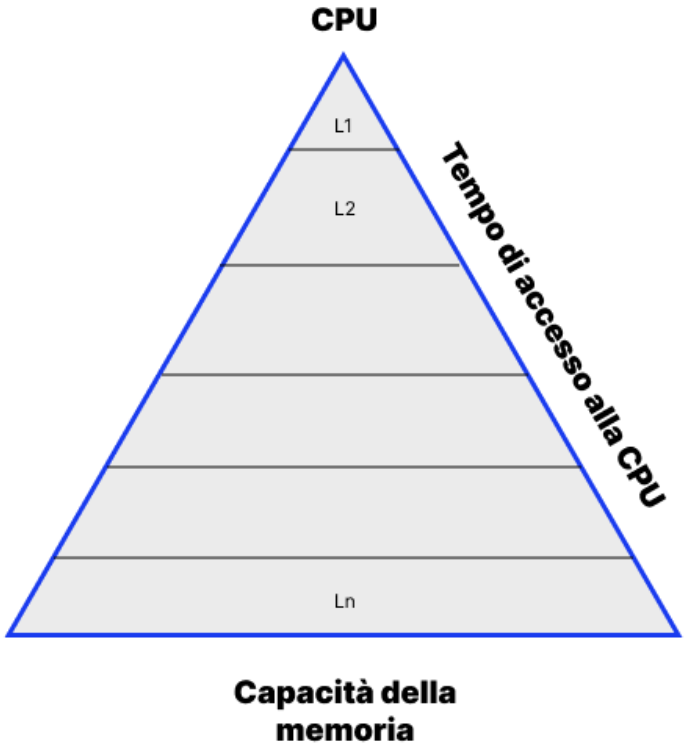
Gerarchie di memoria

Ogni livello di memoria ha delle caratteristiche diverse per quanto riguarda la **velocità** e la **capacità di storage**.

Per **velocità** si intende il tempo impiegato per accedere alla risorse.

Per **capacità** si intende il numero di risorse che è possibile salvare in memoira.

Piramide gerarchica



▼ **Memoria interna**

Registri interni alla CPU: **alta velocità**, **bassa capacità**

▼ **Memoria cache**

Memoria che si trova tra la memoria interna e la memoria centrale.

▼ **Memoria centrale**

Memoria principale del pc, **velocità** più **bassa** e **capacità** più **alta**

▼ **Memoria secondaria**

Memoria di massa, **moltissima capacità** ma **velcoità ridotta**.

💡 **NB**

- I programmi in esecuzione non vedono la gerarchia ma interpretano ogni memoria come centrale.
- ogni livello più vicino alla CPU contiene un sottoinsieme dei dati contenuti dal livello sottostante.

Principio di località

Un programma in un verto istante ha accesso solo ad una porzione della memoria, tale porzione è gestita secondo il **principio di località.**

▼ Località temporale

L’elemento di riferimento **NON cambia** dopo poco tempo.

▼ Località spaziale

L’elemento di riferimento **cambia** dopo poco tempo.

Alcune definizioni

- **Linea o blocco**

  Il più piccolo dato gestibile in memoria

- **Hit**

  Successo nell'accesso al dato *(il dato si trova in quel livello di memoria)*

- **Miss**

  Insuccesso nell'accesso al dato *(il dato si potrebbe trovare al livello superiore)*

- **Hit rate**

  Frequanza di hit

- **Miss rate**

  Frequenza di miss

- **Tempo di hit**

  Tempo necessario per accedere al livello di memoria superiore

- **Tempo di miss**

  Tempo necessario pre trasferire un dato da un livello ad un altro

- **Frequenza di hit**

  numero di hit fratto numero di richieste

- **Frequenza di miss**

  numero di miss fratto numero di richieste

Cache

Si tratta di una memoria che mantiene i dati richiesti recentemente vicini alla CPU e si occupa anche di spostare i dati vicini a quello richiesto.

Tipi di cache

Direct Mapped

Ogni blocco della memoria corrisponde ad una specifica locazione nella cache.

Per stabilire la poszione degli elementi :

- *Se la cache ha 2^n numero di elementi: guardo i $\log_2(n^\circ elementi)$ LSB dell'indirizzo originale.*
- *Se la cache non ha 2^n numero di elementi: (indirizzo originale) MOD (n° elementi)*

▼ Fully Associative

Ogni blocco della memoria può essere salvato in un qualsiasi punto della cache.

Per stabilire la posizione posso utilizzare:

- *Ricerca sequenziale per tutta la cache.*
- *Ricerca parallela per tutta la cache.*

▼ Set Associative

Ogni blocco della memoria dispone di un certo numero di locazioni possibili nella cache.

Direct Mapped

Campi della cache

▼ TAG

Contiene le informazioni per controllare se la parola corrisponde a quella cercata.

▼ INDICE

Utilizzato per indicizzare la cache e quidni selezionare il blocco corretto.

▼ Bit di validità

Bit di controllo per stabilire se il blocco contiene o meno valori.

💡 **NB**
Il numero di bit contenuti in una cache D.M. è:
$2^n(dim\_blocco + dim\_tag + bit\_validità) \rightarrow$
$2^n(2^m * 32 + (32 - (n + m + 2)) + 1)$

Es.

Indirizzo: **00101**

- **TAG: 00**
- **INDICE: 101**

Come avviene la richiesta di un dato

1. Leggo l’indirizzo del dato
2. i bit meno significativi vengono usati come indice
3. i bit più significativi sono usati come tag

Dimensioni del blocco

La dimesione del blocco in una cache M.D. influisce in modo negativo sul miss rate e sul tempo di miss ma in modo positivo sulla possibilità di sfruttare la località spaziale.

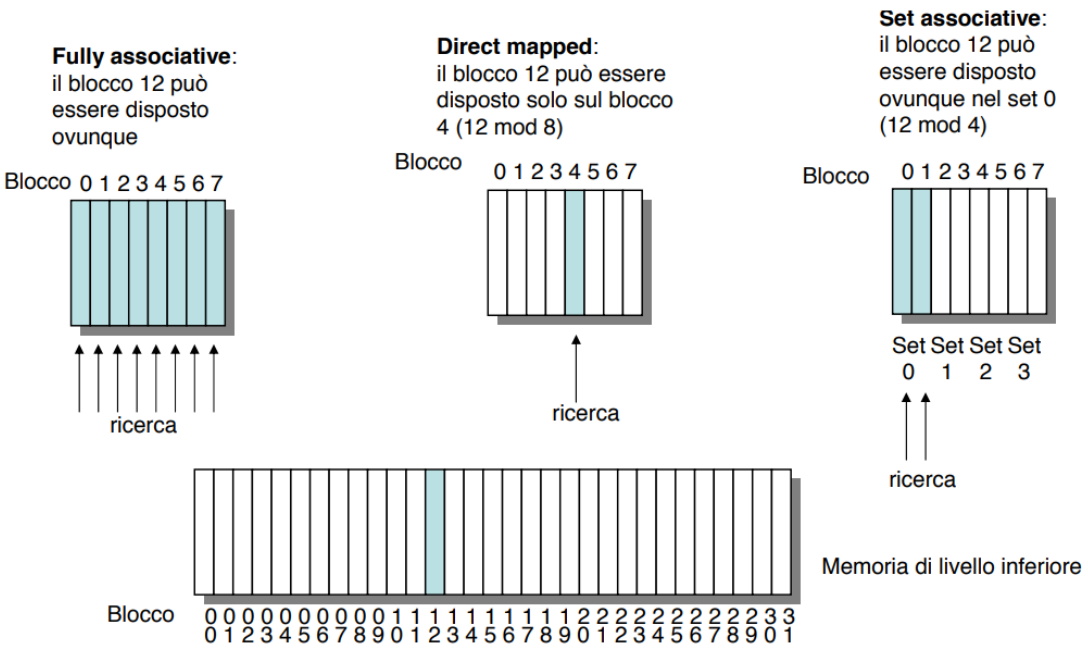
Fully Associative

Tutti i tag vengono controllati in parallelo per identificare il dato richiesto.

Set Associative

- I blocchi sono raggruppati in **set**.
  - Data una cache S.A. ad **N vie** ogni **set** conterrà **N blocchi**
- Stabilito il set di appartenenza di un indirizzo per determinare la sua presenza **controllo i tag di quel set** in parallelo.
- La cache S.A. presneta alcuni svantaggi:
  - **N comparatori** invece di 1
  - Presenza di un **MUX** che aumenta i tempi

Riassunto



Gestione della cache

Dove posizionare un blocco e deve reperirlo sono compiti svolti dall'indirizzamento *vedere siassunto sopra.*

Quale blocco sostituire dopo un miss è un compito dell'algoritmo di sostituzione e come gestire la scrittua in cache è gestito a seconda della stategia di aggiornamento utilizzata.

Algoritmo di sostituzione

Esistono diverse politiche di sostituzione:

- Cache **A.D.** → il blocco viene sovrascritto a quello precedente

- Cache **F.S.** → il blocco viene sovrascritto ad un blocco secondo una certa politica.
- Cache **S.S.** → il blocco viene sovrascritto ad un blocco del set secondo una certa politica.
  - Secondo la politica **FIFO** → il blocco più vecchio viene eliminato
  - Secondo la politica **LRU** → il blocco inutilizzato da più tempo viene eliminato
  - Secondo la politica **Random**

Gestione dei miss di lettura

Se si verifica un miss la cpu viene messa in stallo dopo di che:

- Invio PC-4 alla memoria
- Leggo i dati presenti in memoria
- Riscrivo la cache
- Riavvio l’esecuzione che ha causato il miss

Accesso in scrittura

Per scrivere dei dati nella cache è necessario non creare incoerenza con gli stati sottostani e quindi anche quelli devono essere aggiornati.

▼ **Write through**

Scrivo nella cache e nel livello sottostante.

▼ **Write back**

Scrivo solo nella cache, il dato verrà aggiornato nel livello sottostante solo quando esso verrà sostituito.

▼ **Write buffer**

La CPU scrive il dato nella cache e in un buffer, dopo di che un controller sposterà il dato dal buffer alla memoria sottostante

Write Miss

▼ **No write allocate**

in caso di miss nella cache la scrittura può avvenire solo nel livello sottostante.

▼ **Write allocate**

In caso di miss il blocco in cui scrivere viene caricato in cache e poi scritto.