Architettura degli Elaboratori I - B

Le Memorie Cache

Introduzione

Daniel Riccio

Università di Napoli, Federico II

21 aprile 2018

Fonti: lucidi degli anni precedenti

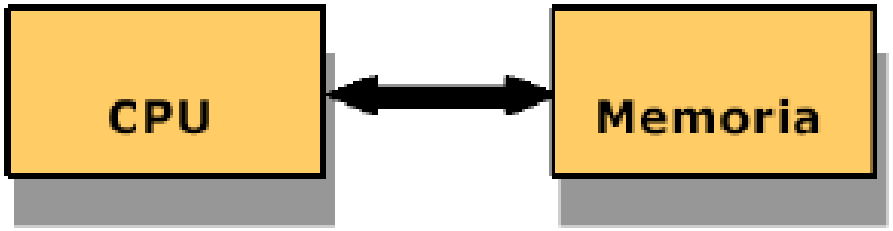
Memorie “Cache”

Sommario

1. Principio di funzionamento.
2. Tecniche di gestione.
3. Algoritmi di posizionamento (mapping).
   a) Indirizzamento diretto.
   b) Indirizzamento associativo.
   c) Indirizzamento set-associativo.
4. Algoritmi di sostituzione.
5. Considerazioni sulle prestazioni di un computer.
6. Interallacciamento della memoria principale.
7. Efficacia di una cache memory: Frequenza di successo e penalità di fallimento.
8. Considerazioni sul posizionamento della cache memory.

Memorie vs CPU

Nell'architettura VonNeuman il canale di comunicazione tra la CPU e la memoria è il punto critico (collo di bottiglia) del sistema.



La tecnologia consente di realizzare CPU sempre più veloci e memorie sempre più grandi MA la velocità di accesso delle memorie non cresce così rapidamente come la velocità della CPU.

Memorie vs CPU

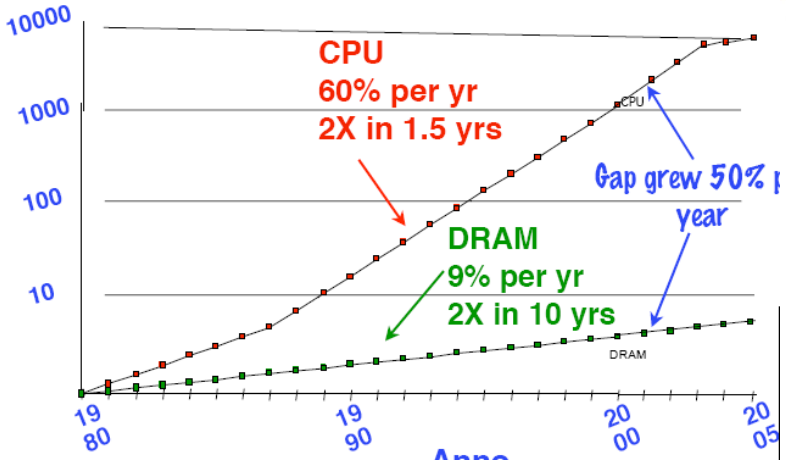
Storicamente le **CPU** sono sempre state più veloci delle **memorie**.

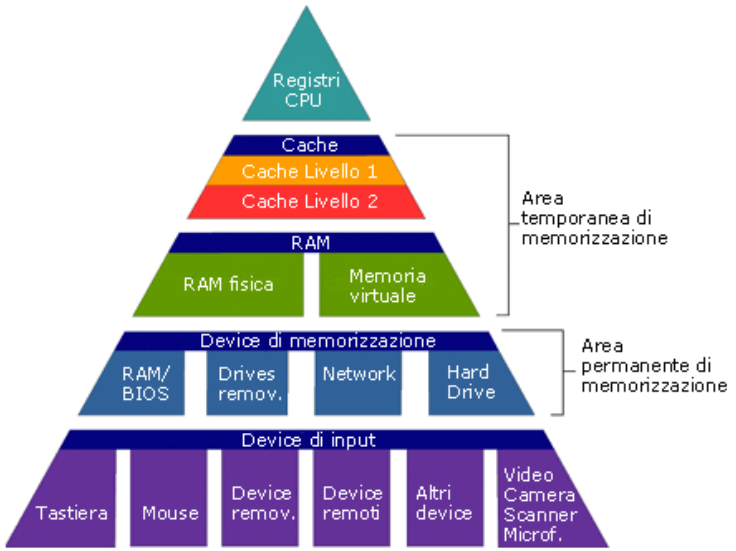
Al giorno d'oggi siamo in grado di produrre delle memorie veloci quanto una CPU moderna, il reale problema è dato da:

- queste memorie hanno un costo elevatissimo.
- le memorie dovrebbero essere piazzate in gran parte sugli stessi chip delle CPU, il che non è possibile.

Per ovviare a questo problema, gli ingegneri ricorrono ad uno schema chiamato "**a gerarchia di memoria**" in cui si combinano:

- una quantità molto **piccola** di memoria estremamente **veloce** (la cache)
- una quantità molto **grande** di memoria **lenta**.





Considerazioni sulla tecnologia

I vari tipi di memoria sono realizzati con tecnologie diverse e si differenziano per

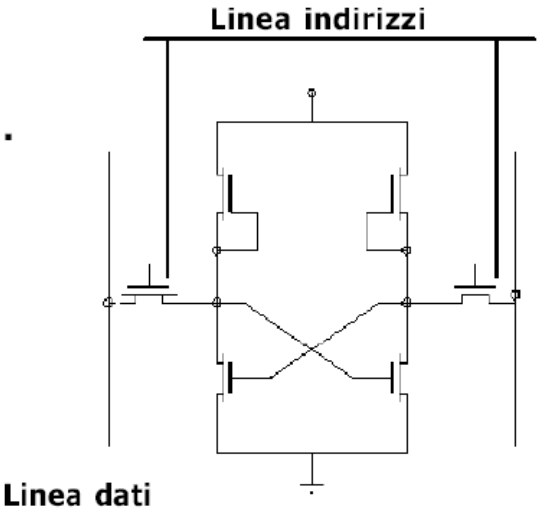
Costo per bit immagazzinato

-Tempo di accesso (o latenza): ritardo fra l'istante in cui avviene la richiesta e l'istante in cui il dato è disponibile

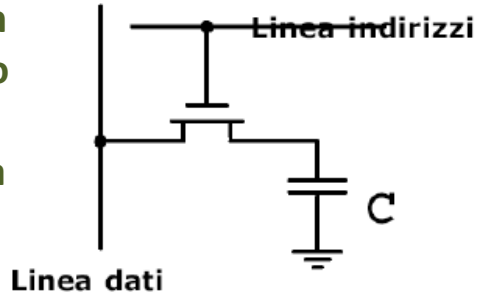
- Modo di accesso (seriale o casuale).

Tecnologie

- Memorie a semiconduttore con tecnologia VLSI (memoria cache e DRAM)
- Memorie magnetiche (memoria secondaria).
- Memorie ottiche (memoria secondaria).



La cella elementare è costituira da un condensatore che viene caricato (1) o scaricato (0).
La tensione sul condensatore tende a diminuire (millisecondi) e quindi deve essere ripristinata o rinfrescata



La cella elementare è costituita da 6 transistori che formano un FLIPFLOP. L'informazione permane stabile in presenza della tensione di alimentazione.

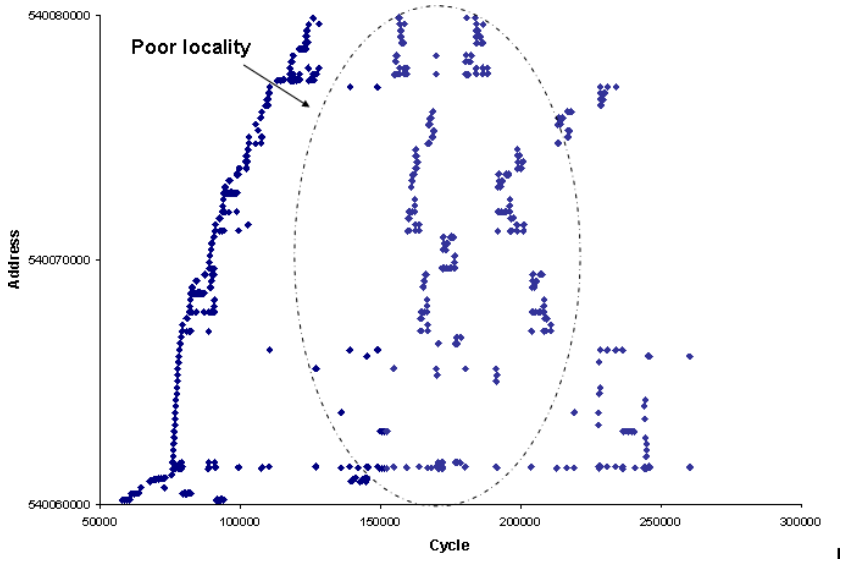
Inquadramento del Problema

La maggior parte del tempo di esecuzione di una CPU è, di solito, impegnato da procedure **in cui vengono eseguite ripetutamente le stesse istruzioni**.

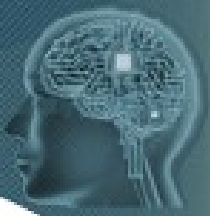
In realtà, non è importante conoscere lo schema dettagliato della sequenza di istruzioni, il punto chiave è che molte istruzioni in aree ben localizzate del programma vengono eseguite ripetutamente in un determinato periodo, e si accede al resto del programma relativamente di rado.

Questa proprietà viene chiamata ***località dei riferimenti***. Si manifesta in due modi: ***località temporale*** e ***località spaziale***.

- La ***località temporale*** rappresenta la probabilità che un'istruzione eseguita di recente venga eseguita nuovamente, entro breve tempo.
- La ***località spaziale*** rappresenta invece la probabilità che istruzioni vicine ad un'istruzione eseguita di recente (dove la vicinanza è espressa in termini di indirizzi delle istruzioni) siano anch'esse eseguite nel prossimo futuro.



Utilità di avere una cache memory



La velocità con cui la memoria risponde alle richieste di istruzioni e dati della CPU ha un peso determinante sulle prestazioni di un sistema.

Se tra la memoria principale e la CPU si potesse **interporre una memoria molto veloce, contenente le parti di programma e di dati che, volta per volta, interessano l'elaborazione,** il tempo totale di esecuzione verrebbe ridotto in modo significativo.

Questa è esattamente la funzione della "***cache memory***" che in inglese significa letteralmente "***schermo della memoria***".

Si tratta quindi di un elemento che inserito tra CPU e memoria principale impedisce alla prima di "***vedere***" i tempi di risposta reali della memoria.

La struttura della CPU, infatti, non viene influenzata dalla presenza o meno di una ***cache memory***. L'interfaccia verso la memoria applica il "***protocollo***" di trasferimento ***dalla*** memoria o ***verso*** la memoria ignorando la presenza di una struttura intermedia.

Agli albori di questa tecnica, infatti, la cache memory era solo un accessorio a pagamento.

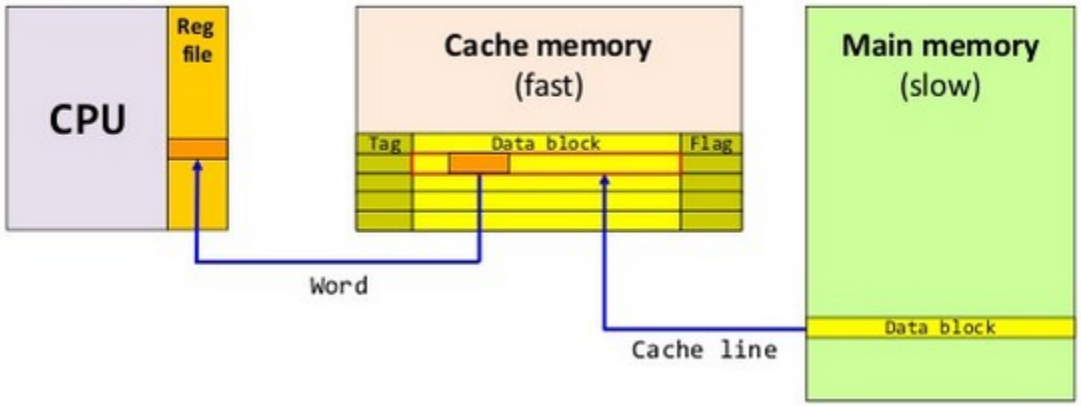
Una traduzione possibile in italiano del concetto è "***memoria tampone***".

Principi di funzionamento

Concettualmente, le operazioni svolte da una memoria cache sono molto semplici. I circuiti di controllo della memoria cache sono progettati per avvantaggiarsi della proprietà della località dei riferimenti.

- L'aspetto temporale di questa proprietà suggerisce di portare un elemento (istruzioni o dati) nella cache quando viene richiesto per la prima volta, in modo tale che rimanga a disposizione nel caso di una nuova richiesta.

- L'aspetto spaziale suggerisce che, invece di portare dalla memoria principale alla cache un elemento alla volta, è conveniente portare un insieme di elementi che risiedono in indirizzi adiacenti.



Si farà riferimento al termine ***blocco*** per indicare un insieme di indirizzi contigui di una qualche dimensione.

Un altro termine utilizzato di frequente per indicare un blocco della cache è ***linea di cache***.

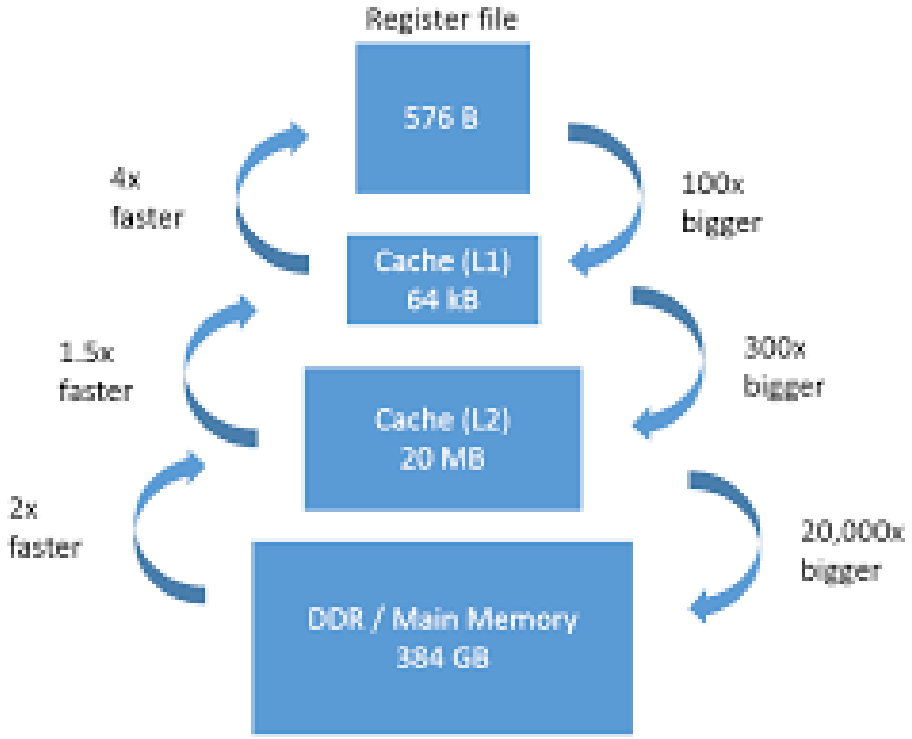
Dimensione della cache

Quando si riceve una richiesta di lettura fatta dalla CPU, il contenuto di un blocco di parole di memoria contenenti la locazione specificata viene trasferito nella cache, una o più parole alla volta.

In seguito, ogniqualvolta il programma fa riferimento a una di queste locazioni del blocco, i valori desiderati vengono letti direttamente dalla cache.

E' chiaro che perché la cache svolga efficacemente il suo compito deve avere tempi di accesso molto più brevi di quelli della memoria principale e ciò come vedremo impone che sia piccola.

Se è piccola non potrà che contenere una frazione ridotta delle istruzioni ed i dati della memoria principale.

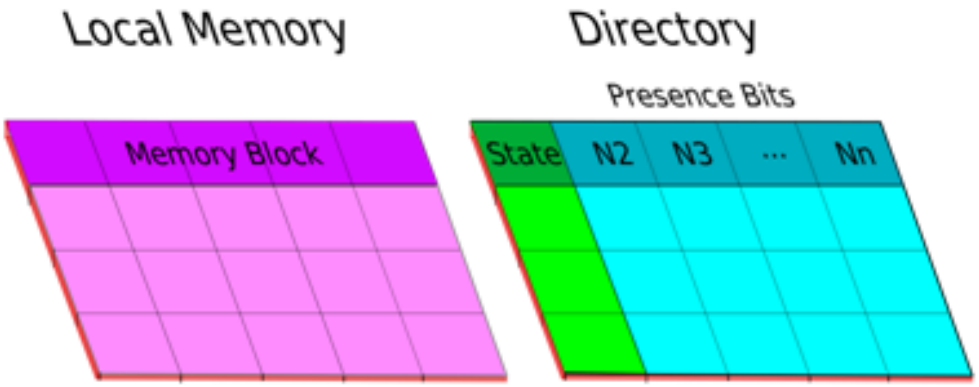


Tecniche di gesione posizionamento e sostituzione

La corrispondenza tra i blocchi della memoria principale e quelli della cache è specificata dalla funzione di ***posizionamento*** (***mapping***).

Quando la cache è piena e si fa riferimento a una parola di memoria (istruzione o dato) che non è presente nella cache, l'hardware di controllo della cache deve decidere quale blocco della cache debba essere rimosso per far spazio al nuovo blocco, che contiene la parola a cui si fa riferimento.

L'insieme di regole in base alle quali viene fatta questa scelta costituisce ***l'algoritmo di sostituzione***.



Tecniche di gesione

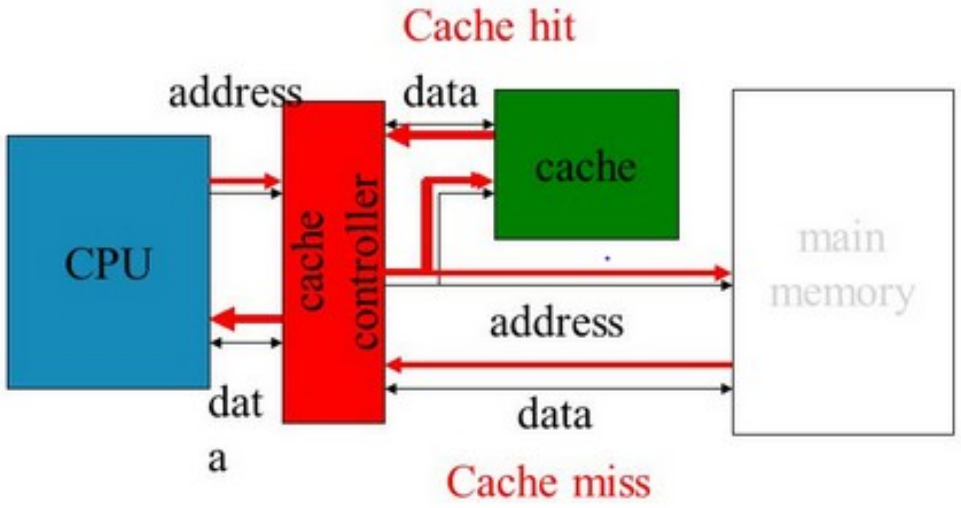
L'indirizzo è quello della memoria

La struttura dei sistemi è, di solito, organizzata in modo che la presenza di una cache non influenzi il funzionamento della CPU.

Quest'ultima, infatti, nell'esecuzione del programma **effettua le richieste di lettura e scrittura utilizzando gli indirizzi delle locazioni nella memoria principale**.

E' la logica di controllo della cache che si fa carico di determinare se la parola richiesta è presente o meno nella cache.

Se è presente, viene effettuata l'operazione di lettura o scrittura della locazione di memoria appropriata. In questo caso si dice che **l'accesso in lettura o in scrittura ha avuto successo** (***read hit*** o ***write hit***).



Possibili gestioni di un “cache hit”

Se l’accesso alla cache ha avuto successo bisogna distinguere due tipi di situazioni:

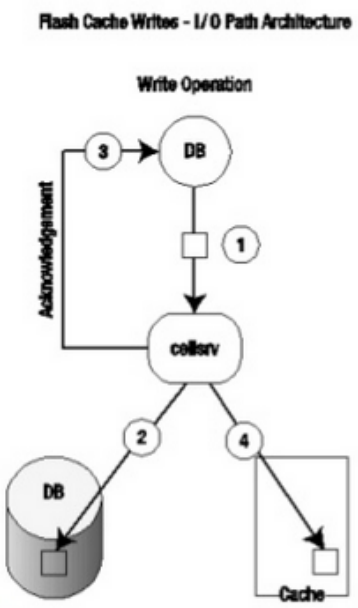
Operazione di lettura, la memoria principale non viene coinvolta

Operazione di scrittura, il sistema può procedere in due modi

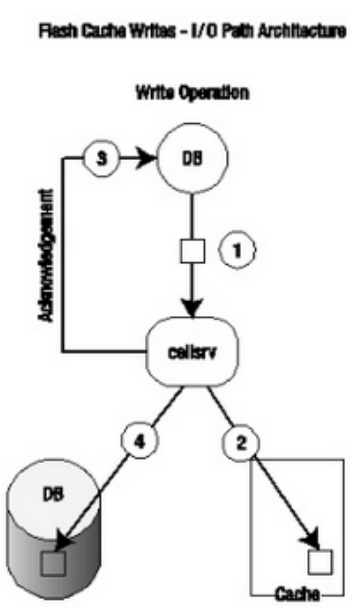
Nel primo (***write-through***), la locazione della cache e quella della memoria principale **vengono entrambe aggiornate**.

Nel secondo si aggiorna **soltanto la locazione della memoria cache**, segnandola come aggiornata con un **bit di modifica** o ***dirty***.

Write-Through

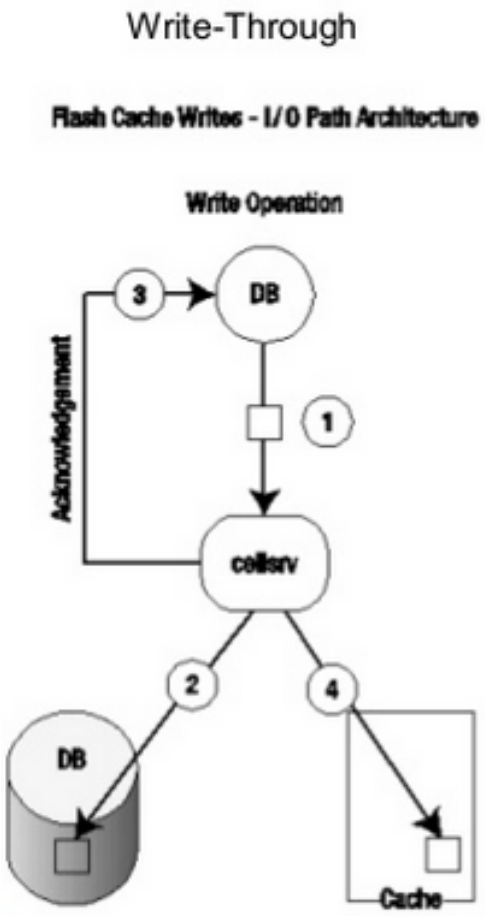


Write-Back



Nell’ultimo caso, la locazione della memoria principale viene aggiornata in seguito, quando il blocco contenente la parola marcata deve essere rimosso dalla memoria cache per far posto a un nuovo blocco.

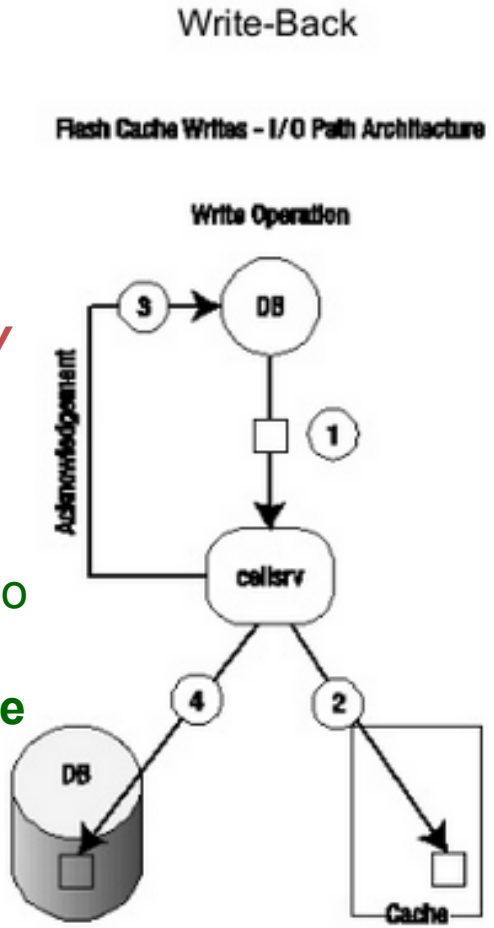
Possibili gestioni di un “cache hit”



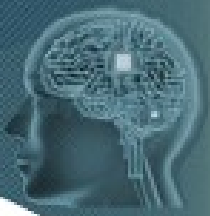
Il ***write-through*** è più semplice, ma implica inutili operazioni di scrittura nella memoria principale, se una parola viene aggiornata più volte durante il periodo in cui risiede nella cache.

La tecnica di aggiornamento differito della memoria principale viene chiamata spesso ***write-back***, o ***copy back***.

Ma anche il protocollo ***write-back*** può causare inutili scritture nella memoria principale, visto che, quando si procede con la scrittura nella memoria principale di un blocco, **tutte le parole del blocco vengono scritte**, anche se solo una delle parole del blocco della cache è stata modificata.



Gestione di un “read miss”



Quando la parola indirizzata durante un'operazione di lettura non è presente nella cache si dice che **l'accesso in lettura è fallito** (***read miss***).
Il blocco di parole contenente la parola richiesta deve essere copiato dalla memoria principale nella memoria cache.

Due sono le possibilità:

- Dopo aver caricato l'intero blocco nella memoria cache, la parola richiesta viene inviata alla CPU.
- In alternativa, è possibile inviare immediatamente la parola alla CPU, non appena la si legge dalla memoria principale.

Quest'ultimo approccio, chiamato ***load-through***, o anche ***early restart***, riduce in qualche modo il tempo di attesa della CPU, a discapito di una maggiore complessità del circuito di controllo della cache.

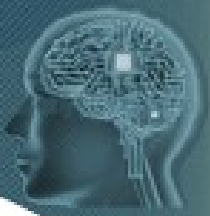
Gestione di un “write miss”

Durante un'operazione di scrittura, se la parola indirizzata non è nella cache si dice che **l'accesso in scrittura è fallito** (***write miss***).

Anche in questo caso ci sono le due alternative,

- se si utilizza un protocollo ***write-through***, le informazioni vengono scritte direttamente nella memoria principale,
- nel caso invece di un protocollo ***write-back***, il blocco, contenente la parola indirizzata, viene prima caricato nella cache, poi viene soprascritto con le nuove informazioni.

Prestazioni



tasso di **hit** (h) : rapporto tra il numero di hit nel livello superiore e il numero totale dei riferimenti;

tasso di **miss** (m) : rapporto tra il numero di miss nel livello superiore e il numero totale dei riferimenti.

$$m = 1 - h$$

tc : tempo di accesso alla cache

tp : tempo di penalizzazione (tempo per accedere alla memoria centrale e per portare la linea in cache)

t : tempo medio di accesso alla cache

$$t = h\ tc + m\ tp$$