La memoria cache

A cura di:
Luca Breveglieri Giuseppe Pozzi Donatella Sciuto

DEI, PoliMI, Milano

luca.breveglieri,giuseppe.pozzi,donatella.sciuto@polimi.it

- versione dell'11 aprile 2003 -

La memoria cache

Obiettivo

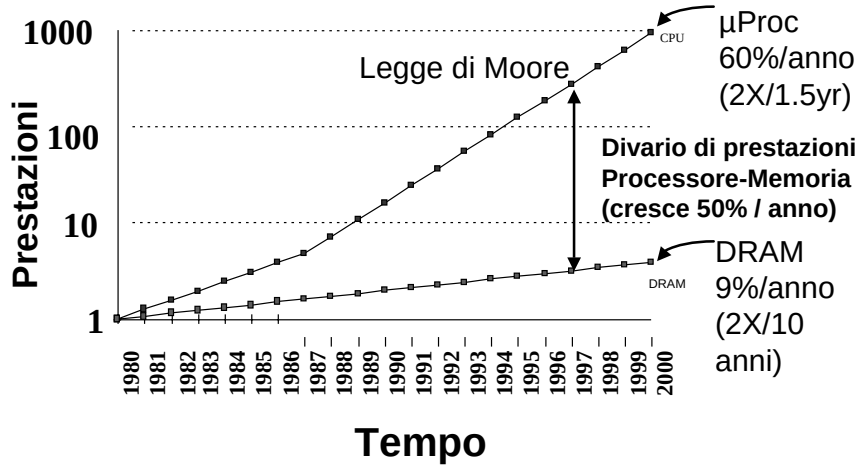
- Come migliorare le prestazioni attraverso il sistema di memoria:
  - La gerarchia di memoria
  - Le memorie cache: architetture
  - Analisi delle prestazioni della memoria

Il problema della memoria

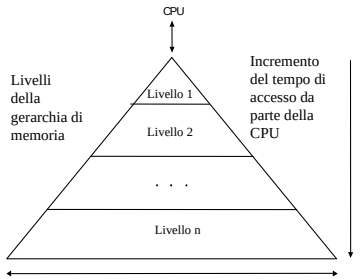
- Obiettivo:
  - fornire agli utenti una memoria grande e veloce
  - fornire al processore i dati alla velocità con cui è in grado di elaborarli
- Problema: Il tasso di crescita nella velocità dei processori non è stato seguito da quello delle memorie

  Tempo di accesso alle SRAM: 2 - 25ns al costo di $100 - $250 per Mbyte.
  Tempo di accesso alle DRAM: 60-120ns al costo di $5 - $10 per Mbyte.
  Tempo di accesso al disco: da 10 a 20 million ns al costo di $.10 - $.20 per Mbyte.

Prestazioni di processori e cache

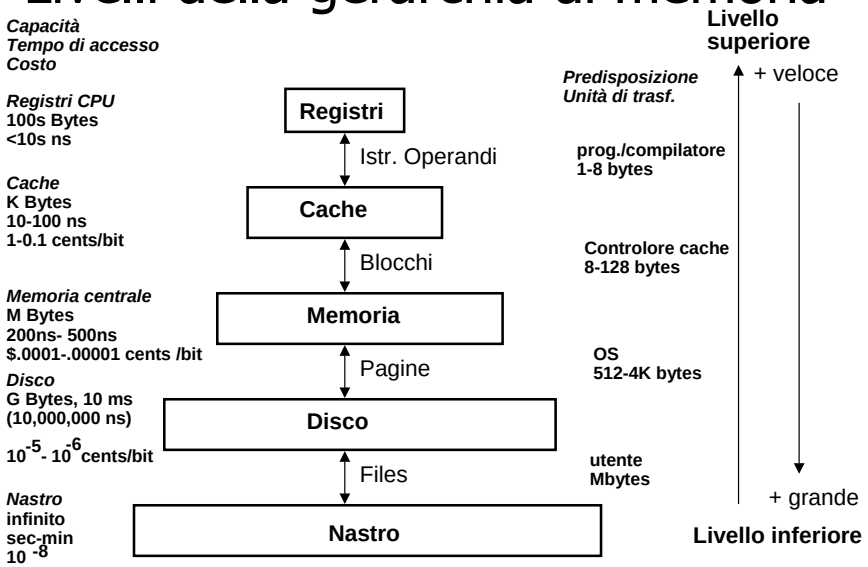


Soluzione: gerarchia di memoria



- Utilizzare diversi livelli di memoria, con tecnologie diverse in modo da ottenere un buon compromesso costo/prestazioni

Livelli della gerarchia di memoria



Località

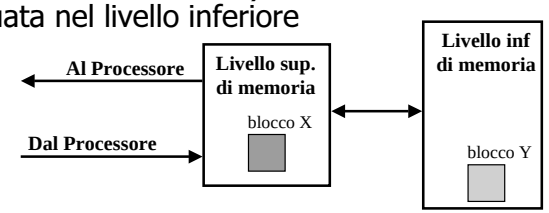
- Il principio che rende la gerarchia di memoria una buona idea per incrementare le prestazioni del sistema di memoria
- Località: in ogni istante di tempo un programma accede a una parte relativamente piccola del suo spazio di indirizzamento
- Esistono due diversi tipi di località: *temporale* e *spaziale*

Il principio di località

- Località temporale: se un dato viene referenziato in un dato istante, è probabile che lo stesso dato venga nuovamente richiesto entro breve
- Località Spaziale: Se un dato viene utilizzato in un dato istante, è probabile che dati posizionati in celle di memoria adiacenti vengano anch'essi richiesti entro breve
- Negli ultimi 15 anni, le tecniche di miglioramento delle prestazioni nell'hardware si sono basate sul principio di località

Gerarchia di memoria

- Si considerino solo due livelli di gerarchia:
- Il processore richiede un dato al sistema di memoria:
  - La richiesta viene prima inviata al livello di memoria superiore (più vicino al processore)
  - Se il dato non è presente nel livello superiore (fallimento della richiesta) la ricerca viene effettuata nel livello inferiore



Gerarchia di memoria: definizioni

- Hit (successo): dati presenti in un blocco del livello superiore (esempio: Blocco X)
  - Hit Rate (tasso di successo): numero di accessi a memoria che trovano il dato nel livello superiore sul numero totale di accessi
  - Hit Time (tempo di successo): tempo per accedere al dato nel livello superiore della gerarchia:

    Tempo di accesso alla RAM + tempo per determinare successo/fallimento della richiesta

Gerarchia di memoria: definizioni

- Miss (fallimento): i dati devono essere recuperati dal livello inferiore della memoria (Blocco Y)
  - Miss Rate (tasso di fallimento) = 1 - (Hit Rate)
  - Miss Penalty (tempo di fallimento): tempo necessario a sosituire un blocco nel livello superiore + tempo per trasferire il blocco al processore
  - Hit Time << Miss Penalty

Cache

- Memoria al livello superiore della gerarchia
- Sfruttare il principio di località dei programmi e tenere in memoria cache i dati utilizzati più di recente
- Obiettivo: fornire dati al processore in uno o due cicli di clock
- Memoria cache: veloce nei tempi di accesso ma di dimensioni ridotte

Cache e principio di località

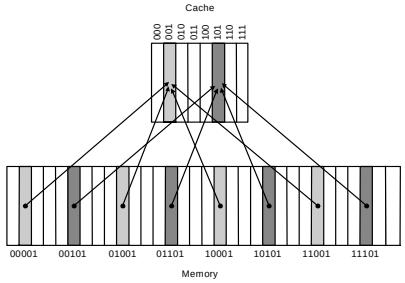
- Le memorie cache sfruttano il principio di località spaziale trasferendo dal livello inferiore della gerarchia più dati di quanti non ne siano stati strettamente richiesti (blocco o linea di cache)
- La località temporale viene sfruttata nella scelta del blocco da sostituire nella gestione di un fallimento (es: sostituire il blocco a cui si è fatto accesso meno di recente)

Memoria cache: organizzazione

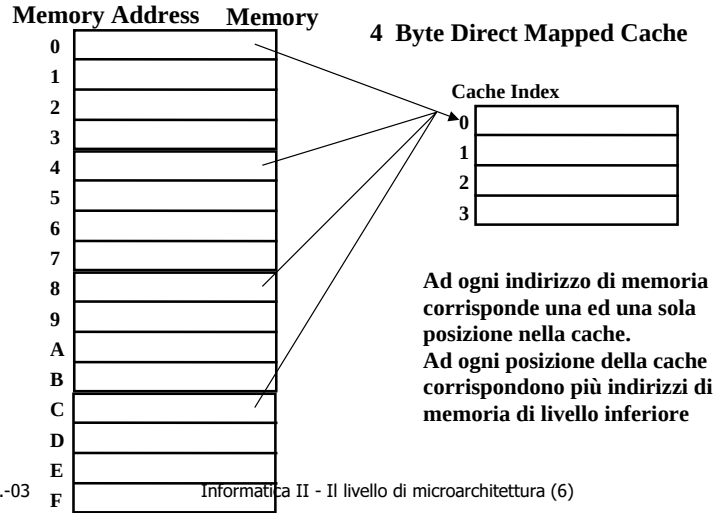
- Due problemi:
  - Come verifico se un dato è presente in cache?
  - Se lo è, dove lo trovo?
- Primo esempio:
  - dimensione della linea di cache: un dato (una parola di memoria)
  - **Indirizzamento diretto**

Cache a indirizzamento diretto

- Mapping: l'indirizzo del dato in cache corrisponde all'indirizzo in memoria modulo il numero di blocchi



Indirizzamento nella cache a indirizzamento diretto



Cache a indirizzamento diretto

- Posizione 0 può essere occupata da dati provenienti da:
  - Indirizzi di memoria 0, 4, 8, ... etc.
  - In generale: ogni indirizzo di memoria i cui 2 bit meno significativi dell'indirizzo sono 0
  - Indirizzo<1:0> => posizione in cache
- Quale dato va posto in cache?
- Come identificare univocamente il dato in cache ?

Cache a indirizzamento diretto di 1KByte, 32Byte linee di cache

- Per una cache di 2^N byte:
  - I (32 - N) bit più significativi corrispondono sempre all'etichetta
  - Gli M bit meno significativi permettono la selezione del singolo Byte (dimensione della linea di cache = 2^M byte)
- Indirizzi di memoria da 32 bit, 1KByte cache con linee di cache da 32 Byte: 22 bit di etichetta, 5 bit per linea di cache e 5 bit meno significativi per indirizzare il singolo byte

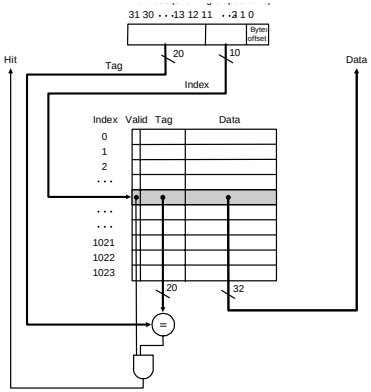
Indirizzamento nelle cache a indirizzamento diretto

- Indirizzo di memoria di N bit diviso in 4 campi:
  1. B bit meno significativi permettono di individuare il singolo byte della parola nella linea di cache
     - Se la parola non è indirizzabile per byte B= 0
  2. K bit per identificare la parola all'interno della linea di cache
     - Se la linea contiene una sola parola K=0
  3. M bit per individuare la posizione della linea di cache
  4. N-M-K bit di etichetta per verificare che la linea di cache contenga esattamente l'indirizzo cercato

Esempio

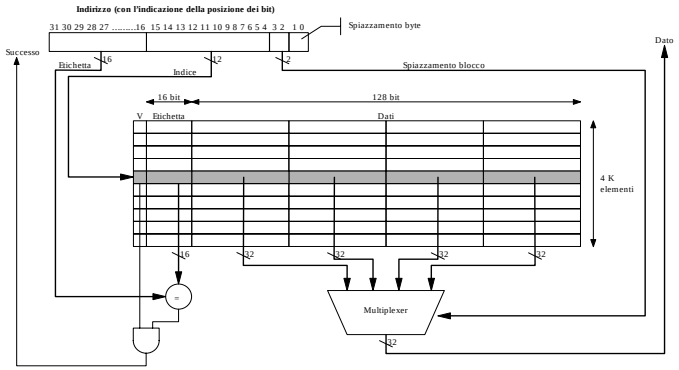
- Indirizzi di memoria a 32 bit
- Cache a indirizzamento diretto con una parola di 4 byte per linea di cache e 1024 linee di cache (2^{10})
- Struttura dell'indirizzo di memoria:
- Bit 0 e 1 per individuare il singolo byte
- Bit 2-11 per individuare la linea di cache
- Bit 12-31 come etichetta

Cache a indirizzamento diretto



Cache a indirizzamento diretto

- Per sfruttare la località spaziale: linea di cache di dimensioni maggiori (es. 4 parole)



Struttura delle cache a indirizzamento diretto

- Ogni posizione della cache include:
  1. **Valid bit** che indica se questa posizione contiene o meno dati validi. Quando il calcolatore viene acceso tutte le posizioni della cache sono segnalate come NON valide
  2. **Campo etichetta** che contiene il valore che identifica univocamente l'indirizzo di memoria corrispondente ai dati memorizzati
  3. **Campo dati** che contiene una copia dei dati

Hit vs. Miss in lettura

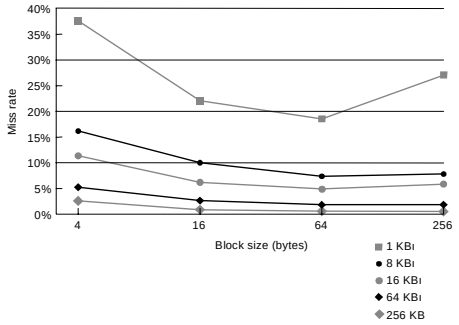
- Interazione tra processore e cache: lettura o scrittura di un dato
- Successo in lettura di un dato
  - Obiettivo da raggiungere!
- Fallimento nella lettura di un dato
  - *stallo* della CPU, richiesta del blocco contenente il dato cercato alla memoria, copia in cache, ripetizione dell'operazione di lettura in cache

Hit vs Miss in scrittura

- Successo nella scrittura:
  - Sostituzione del dato sia in cache sia in memoria (write-through)
  - Scrittura del dato solo nella cache (write-back) : la copia in memoria avviene in un secondo momento
- Fallimento nella scrittura:
  - *stallo* della CPU, richiesta del blocco contenente il dato cercato alla memoria, copia in cache, ripetizione dell'operazione di scrittura

Prestazioni

- Aumento delle dimensioni della linea di cache (blocco) tende a ridurre il numero dei miss:



Miglioramento delle prestazioni

- Migliorare sia larghezza di banda (velocità di esecuzione) sia latenza (tempo necessario per svolgere l'operazione): uso di cache multiple
- Introdurre una cache separata per istruzioni e dati (**split cache**)
  - Le operazioni di lettura/scrittura possono essere svolte in modo indipendente in ogni cache ⇨ raddoppia larghezza di banda della memoria
  - Processore necessita di due porte di collegamento alla memoria

Utilizzo di due cache: risultati sperimentali

Programma	Dim. Blocco (n. Parole)	Miss rate istruzioni	Miss rate dati	Miss rate globale effettivo
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

Analisi delle prestazioni

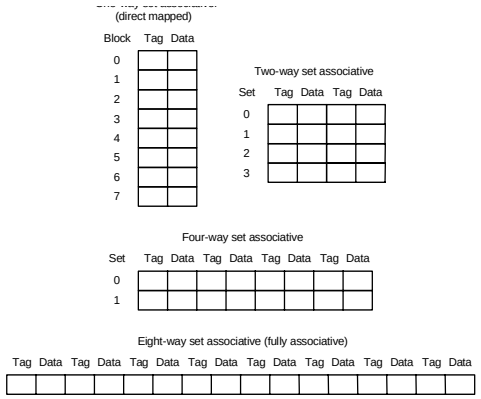
- Modello semplificato:

  Tempo di esecuzione = (cicli di esecuzione + cicli di stallo) $\times$ periodo del ciclo

  Cicli di stallo = # istruzioni $\times$ miss rate $\times$ miss penalty

- Due modi per migliorare le prestazioni:
  - ridurre miss rate
  - ridurre il miss penalty

Ridurre il miss rate mediante associatività



Cache completamente associative

- Un dato può essere memorizzato in qualunque posizione della cache.
- Non esiste una relazione tra indirizzo di memoria del dato e posizione in cache
- Struttura dell'indirizzo di memoria di N bit, con linee di cache di 2^M byte:
  - M bit meno significativi dell'indirizzo individuano il byte nella cache
  - N-M bit più significativi: etichetta

Cache associative

- Ricerca di un dato nella cache richiede il confronto di tutte le etichette presenti in cache con l'etichetta dell'indirizzo di memoria richiesto
- Per aumentare le prestazioni la ricerca avviene in parallelo
- In caso di fallimento della ricerca è necessario copiare il dato dalla memoria centrale
- Se la cache è piena: necessario sostituire un dato. Quale?
  - Scelta casuale
  - Scelta del dato meno utilizzato di recente (LRU)

Cache set-associative

- Ogni blocco può essere messo in un numero prefissato di posizioni (almeno due);
- Una cache set-associativa in cui un blocco può andare in *n* posizioni viene definita set-associativa a *n* vie.
- Una cache set-associativa a *n* vie è costituita da numerosi insiemi, ognuno dei quali comprende *n* blocchi

Cache set associative

- Ogni blocco della memoria corrisponde ad un unico *insieme* della cache ed il blocco può essere messo in uno *qualsiasi* degli elementi di questo insieme
- Combina la modalità a indirizzamento diretto per gli insiemi della cache, e la modalità completamente associativa per i blocchi all'interno dell'insieme.

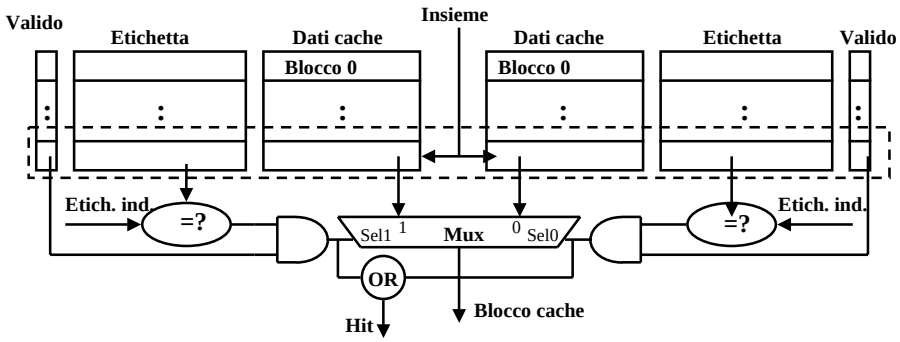
Indirizzamento nelle cache set associative

- Un indirizzo di memoria di N bit è suddiviso in 4 campi:
  1. B bit meno significativi per individuare il byte all'interno della parola
  2. K bit per individuare la parola all'interno del blocco
  3. M bit per individuare l'insieme
  4. N-(M+K+B) come etichetta

Cache set associativa

- Cache a due vie: insiemi di 2 blocchi
- Equivale ad avere due cache a indirizzamento diretto che operano in parallelo
- La parte di indirizzo che individua l'insieme seleziona i due blocchi della cache
- Le due etichette vengono confrontate in parallelo con quella dell'indirizzo cercato
- Il dato viene selezionato in base al risultato dei due confronti

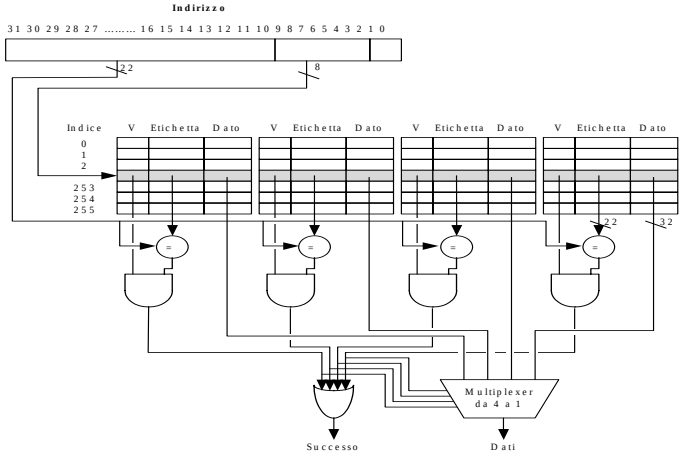
Cache Set Associativa a due vie



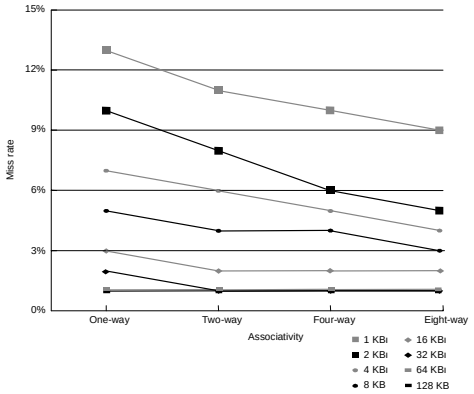
Cache set associativa a 4 vie

- Indirizzo di memoria: 32 bit
- Memoria cache 1KByte indirizzabile per byte, 1 parola da 4 Byte per blocco
- Organizzazione dell'indirizzo:
  - Bit 0 e 1 per indirizzare i byte
  - Numero blocchi nella cache = dimensioni della cache/dimensioni del blocco= $2^{10}/1 = 2^{10}$
  - Numero di insiemi nella cache= numero di blocchi/ dimensioni dell'insieme = $2^{10}/2^2 = 2^8$
  - Bit 2-10 indirizzo dell'insieme nella cache
  - Bit 31-11 etichetta

Cache set associativa a 4 vie



Prestazioni



Confronto tra diverse organizzazioni di cache

- Cache set-associativa a N vie v. Cache a indirizzamento diretto:
  - N comparatori vs. 1
  - Un ritardo dovuto al MUX aggiuntivo per i dati
  - Dati sono disponibili solo DOPO il segnale di Hit/Miss
- In una cache a indirizzamento diretto, il blocco di cache richiesto è disponibile PRIMA del segnale di Hit/Miss:
  - Possibile ipotizzare un successo e quindi proseguire. Si recupera successivamente se si trattava in realtà di un fallimento.

Conclusioni: 4 domande su gerarchia di memoria

- Q1: Dove portare un blocco nel livello di memoria superiore? *(Posizionamento del blocco)*
- Q2: Come si identifica un blocco se si trova nel livello superiore? *(Identificazione del blocco)*
- Q3: Quale blocco deve essere sostituito nel caso di un fallimento? *(Sostituzione del blocco)*
- Q4: Cosa succede durante una scrittura? *(Strategia di scrittura)*

Posizionamento del blocco

- Indirizzamento diretto:
  - Posizione univoca: indirizzo di memoria modulo numero dei blocchi in cache
- Completamente associativa:
  - Posizione qualunque all'interno della cache
- Set associativa
  - Posizione libera all'interno dell'insieme
  - Insieme = (indirizzo di memoria/numero dei blocchi) modulo numero degli insiemi

Identificazione del blocco

- Indirizzamento diretto:
  - Calcolo posizione
  - Verifica etichetta e verifica bit valido
- Completamente associativo:
  - Confronta etichetta in ogni blocco e verifica bit valido
- Set-associativo
  - Identifica insieme
  - Confronta etichette dell'insieme e verifica bit valido

Sostituzione del blocco

- Definito dall'indirizzo nelle cache a indirizzamento diretto
- Cache set associative or completamente associative:
  - Casuale
  - LRU (Least Recently Used)

Associatività:	2-way		4-way		8-way	
Dim	LRU	Casuale	LRU	Casuale	LRU	Casuale
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Strategie di scrittura di un blocco

- *Write through* —L'informazione viene scritta sia nel blocco del livello superiore sia nel blocco di livello inferiore della memoria
- *Write back* —L'informazione viene scritta solo nel blocco di livello superiore Il livello inferiore viene aggiornato solo quando avviene la sostituzione del blocco di livello superiore.

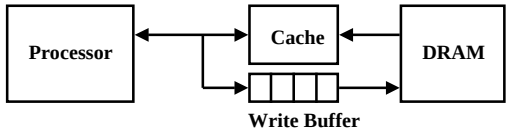
Strategie di scrittura

- Write back
  - Per ogni blocco di cache è necessario mantenere l'informazione sulla scrittura:
  - Ad ogni blocco è associato un bit MODIFICA che indica se il blocco in cache è stato modificato o meno e va quindi copiato in memoria in caso di sostituzione

Confronto tra strategie di scrittura

- Write Through: fallimenti in letture non si tramutano in scritture in memoria (penalità di fallimento più lunga)
- Write Back: non si hanno aggiornamenti ripetuti delle stesse celle di memoria
  - L'aggiornamento avviene una volta sola
- Write Through viene realizzato con buffer di scrittura per non aumentare troppo i tempi di scrittura dovuti alle inferiori prestazioni della memoria di livello inferiore

Buffer di scrittura per Write Through



- Necessario un buffer di scrittura tra Cache e Memoria
  - Processore: scrive i dati in cache e nel buffer di scrittura
  - Controllore di memoria: scrive i contenuti del buffer in memoria

Tempo di accesso alla memoria

Tempo medio di accesso alla memoria =
Tasso di successo x tempo di accesso a cache +
Tasso di fallimento x Penalità di fallimento
(ns o cicli di clock)

- Tasso di fallimento = 1 – Tasso di successo
- Penalità di fallimento:
  - Tempo di accesso al livello inferiore: f(latenza livello inferiore)
  - Tempo di trasferimento di un blocco dal livello inferiore: f(larghezza di banda tra i due livelli)

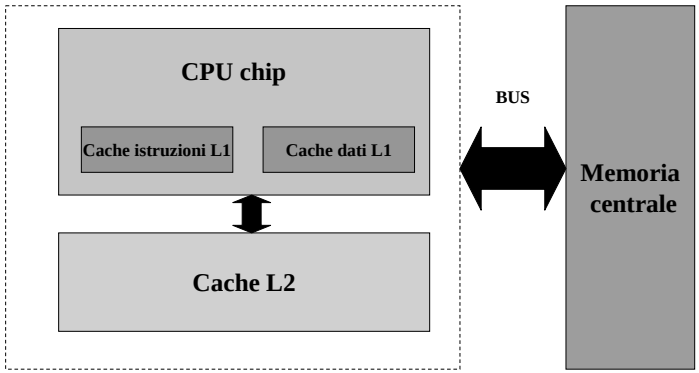
Riduzione della penalità di fallimento: cache multilivello

- Aggiunta di un secondo livello di cache:
  - Spesso la cache primaria è posizionata sullo stesso chip del processore (dimensioni ridotte)
  - Si possono utilizzare SRAM per aggiungere una cache prima della memoria centrale (DRAM)
  - Penalità di fallimento si riduce se il dato è disponibile nel secondo livello di cache (tempi di accesso inferiori)

Utilizzo di cache multilivello:

- Cercare di ottimizzare il tempo di successo della cache di primo livello
- Cercare di ottimizzare il tasso di fallimento del secondo livello di cache

Organizzazione cache a due livelli



Prestazioni con due livelli di cache

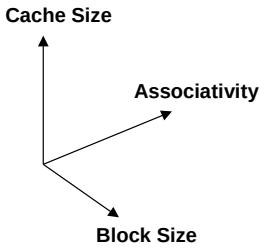
Tempo medio di accesso alla memoria =

Tasso di successo L1 x tempo di accesso a cache L1 +

(1- tasso di successo L1) x tasso di successo L2 x tempo di accesso a cache L2 +

(1- tasso di successo L1) x (1- tasso di successo L2) x penalità di fallimento L2

Lo spazio di progetto delle cache



- Diverse dimensioni interagenti
  - Dimensione delle cache
  - Dimensione dei blocchi
  - Associatività
  - Politica di sostituzione
  - Politica di scrittura (write-through vs write-back)
- La scelta ottima è sempre un compromesso
  - Dipende dalle caratteristiche di accesso
    - carico di lavoro, uso (I-cache, D-cache)
  - Dipende da tecnologie/costi
- La scelta migliore è spesso la più semplice