Gerarchie di Memoria

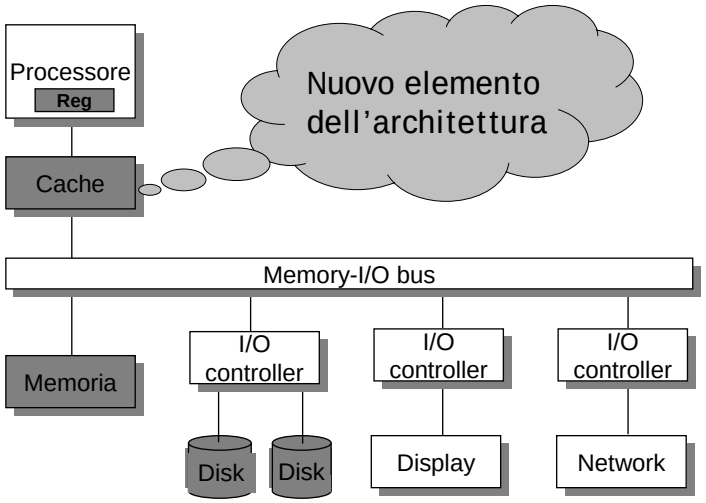
Sistemi Operativi
AA. 2008-09

Prof. O Mejri

Sommario

- Le esigenze di memoria
- Il principio di località dei programmi
- La gerarchia di memoria
- Principio di funzionamento della memoria cache
- I problemi di progetto delle memorie cache
- Prestazioni e ottimizzazioni (cenni)

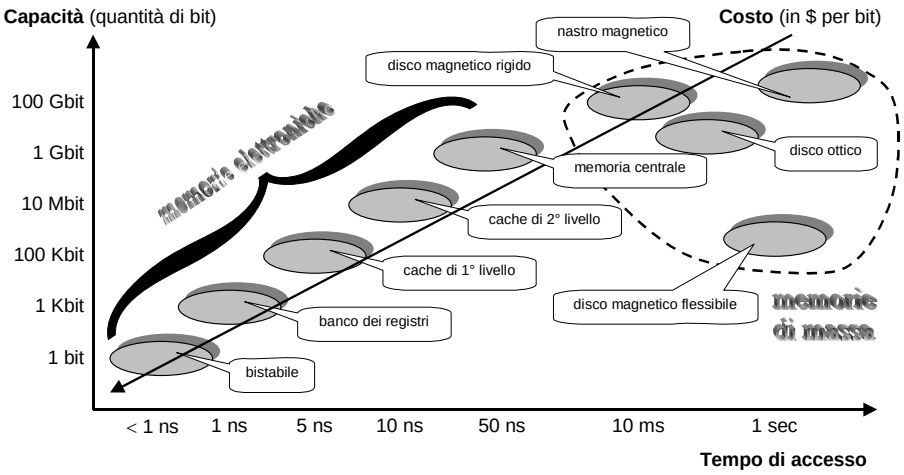
Elementi di memoria di un elaboratore



Le memoria di un sistema d'elaborazione

- Esigenze
  - Elaborazione
  - Archiviazione
- Caratteristiche importanti
  - Velocità, Capacità, Persistenza, Costo
- Tecnologie più comuni
  - Elettronica: veloce, costosa, volatile
  - Magnetica: lenta, economica, permanente
  - Ottica: lenta, economica, difficilmente modificabile
- Obiettivi
  - Unire le tecnologie per giungere a una memoria, veloce, capace e persistente a costi di mercato

Tecnologie di memoria



Il principio di località dei programmi

- Località **spaziale**
  - se l'istruzione di indirizzo *i* entra in esecuzione, con probabilità ≈ 1 anche l'istruzione di indirizzo *i* + *di* entrerà in esecuzione (*di* è un intero piccolo)
- Motivazione
  - di solito le istruzioni sono eseguite in sequenza
  - i salti sono relativamente rari o comunque spesso sono *polarizzati* verso un ramo

Il principio di località dei programmi

- Località **temporale**
  - se un'istruzione entra in esecuzione al tempo t, con probabilità ≈ 1 la stessa istruzione sarà rieseguita al tempo $t + dt$ (dove dt è piccolo rispetto a t)
- Motivazione
  - spesso le istruzioni rieseguite fanno parte di un ciclo, la cui presenza è giustificata solo se esso viene reiterato molte volte
  - se un'istruzione appartenente a un ciclo entra in esecuzione, è molto probabile che, entro il tempo di un'iterazione del ciclo, essa venga rieseguita
  - i cicli brevi generalmente sono molto più numerosi di quelli lunghi

Il principio di località dei programmi

- Località spaziale e temporale sono indipendenti
- Se valgono entrambi, sono riassunti nel principio di località **spazio-temporale**
  - se l'istruzione di indirizzo i entra in esecuzione al tempo t, con probabilità ≈ 1 l'istruzione di indirizzo $i + di$ entrerà in esecuzione al tempo $t + dt$, dove di e dt sono piccoli rispetto a i e t, rispettivamente
- La località spazio-temporale è statisticamente ben verificata dalla maggior parte dei programmi
  - la maggioranza delle istruzioni appartiene a cicli interni, con corpo sequenziale, brevi, iterati numerose volte e operanti su dati contigui
  - legge empirica: 90% del tempo è speso sul 10% codice

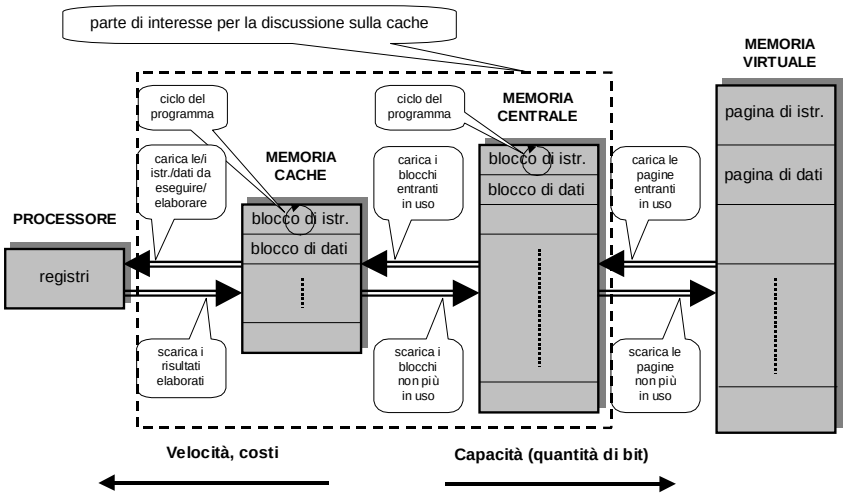
Interpretazione del principio di località

- Le istruzioni del programma si possono raggruppare in "blocchi" di istruzioni consecutive
  - se un'istruzione (qualsiasi) di un blocco entra in esecuzione, allora l'intero blocco di istruzioni verrà eseguito
  - se un blocco (qualsiasi) entra in esecuzione, allora entro breve tempo lo stesso blocco verrà rieseguito
- I blocchi sono per esempio i "corpi" dei cicli più interni al programma
- Anche i dati (oltre alle istruzioni) possono soddisfare al principio di località spazio-temporale

La gerarchia di memoria

- La memoria viene organizzata in livelli caratterizzati da velocità, dimensioni e costi diversi
- I blocchi possono essere trasferiti da un livello inferiore a uno superiore
- Cerco di tenere i *blocchi* di informazione usati più di frequente vicino alla CPU, per ottimizzare i tempi
- Il dimensionamento del sistema e le politiche di gestione derivano da analisi statistico/quantitative delle applicazioni
- L'obiettivo è fornire la sensazione di una memoria con la velocità del primo livello e la capacità del (dei) successivo(i)

La gerarchia di memoria: il livello cache



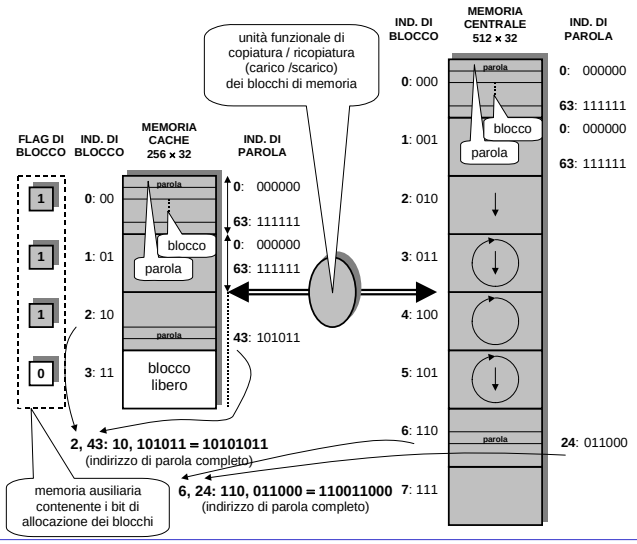
Terminologia

- **Hit**: tentativo di accesso(lett./scritt.) con successo a un determinato livello della gerarchia
- **Miss**: tentativo di accesso(lett./scritt.) andato a vuoto
- **Hit time**: tempo di accesso a livello superiore della gerarchia (incluso rilevazione ev. fallimento)
- **Miss penality**: tempo per sostituire un blocco nel livello superiore con uno del livello inferiore, più il tempo di lettura del dato cercato
- **Hit rate**, **Miss rate**: percentuale dei tentativi di accesso che hanno successo o falliscono (Mr=1-Hr)

Organizzazione di una memoria cache

- Memoria centrale e cache sono organizzate a blocchi di parole, di uguale dimensione
- La memoria cache contiene copie di blocchi della memoria centrale, oppure blocchi liberi
- Il sistema di gestione della cache è in grado di copiare (caricare) blocchi dalla memoria centrale alla memoria cache, oppure di ricopiare (scaricare) blocchi dalla memoria cache alla memoria centrale, tramite un'apposita unità funzionale
- Normalmente il processore accede solo alla memoria cache (tranne casi particolari)
- La capacità del sistema è pari a quella della sola memoria centrale (la cache contiene solo *copie*)

Struttura della memoria cache



Istruzioni: funzionamento base

- Il processore preleva istruzione dalla memoria cache (non dalla memoria centrale)
- Se il blocco contenente l'istruzione da prelevare **si trova** nella memoria cache, l'istruzione viene letta e il processore prosegue l'esecuzione
- Se l'istruzione da prelevare **non si trova** nella cache
  - il processore **sospende** l'esecuzione
  - il blocco contenente l'istruzione da prelevare (ed eseguire) viene **caricato** dalla memoria centrale in un blocco libero della memoria cache
  - il processore preleva l'istruzione dalla memoria cache e riprende l'esecuzione

Dati: funzionamento base

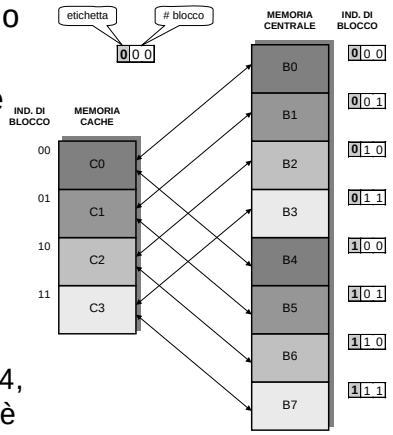
- Il processore deve leggere il dato dalla memoria cache, oppure deve scrivere il dato nella memoria cache
- Se il dato da leggere/scrivere non si trova nella memoria cache, si procede in modo simile alle istruzioni (con qualche differenza, soprattutto nel caso della scrittura)
- Casi particolari
  - in determinate situazioni o sistemi è richiesto che il processore possa accedere direttamente alla memoria centrale, bypassando la cache

I principali problemi di progetto

- Metodo di **indirizzamento**
  - come scegliere il blocco della cache in cui copiare un dato blocco di memoria centrale (*mapping*)
- Metodo di **identificazione**
  - come localizzare un dato blocco di istruzioni o dati all'interno della cache
- Metodo di **lettura**
  - come comportarsi quando si deve leggere una parola contenuta in un blocco della memoria cache
- Metodo di **scrittura** (problema della *coerenza*)
  - come comportarsi quando si deve leggere una parola contenuta in un blocco della memoria cache
- Metodo di **sostituzione**
  - Come si sostituiscono i blocchi della cache al fine di liberare spazio per caricarne altri

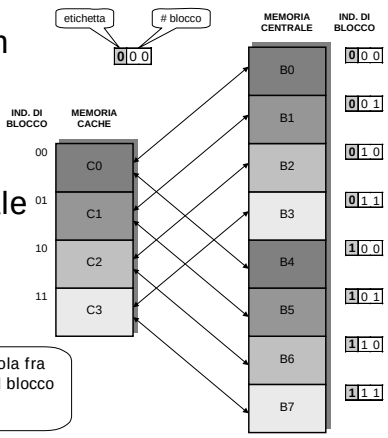
Indirizzamento diretto

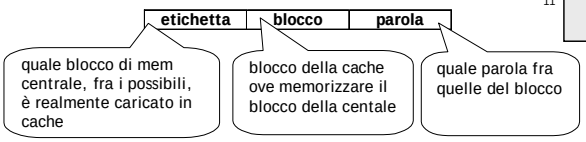
- Ogni blocco della memoria centrale è caricabile in un solo blocco della cache
- Il blocco `j` in mem principale è tradotto in blocco `j MOD #blocchi` in cache
- Alta velocità, basso costo, numerosi conflitti di allocazione
  - Es. se debbo accedere a informazioni in blocco B0 e B4, anche se il resto della cache è vuota, ho continui conflitti



Indirizzamento diretto e identificazione

- Per identificare un blocco in cache occorre memorizzare in memoria associativa l'etichetta dell'indirizzo di memoria centrale
- L'indirizzo di memoria centrale e suddiviso in tre sezioni



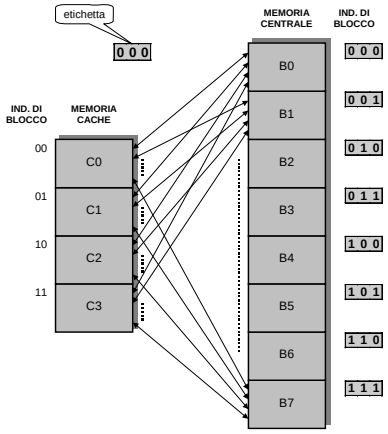


Indirizzamento diretto: esecuzione

- La CPU genera indirizzo e subito analizza la cache
- I bit del campo blocco puntano a blocco particolare della cache
- I bit più significativi, quelli dell'etichetta, sono confrontati con l'etichetta associata al blocco cache
  - se sono uguali
    - OK, la parola è in quel blocco di cache
  - se sono diversi
    - debbo leggere e caricare blocco contenente la parola desiderata dalla memoria centrale

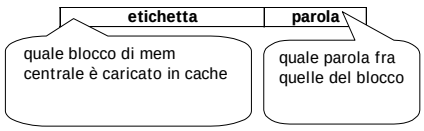
Indirizzamento associativo

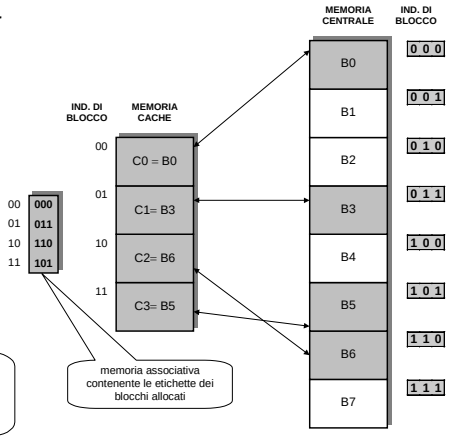
- Ogni blocco delle memoria centrale è caricabile in un blocco qualunque della cache
- Ho pochi conflitti di allocazione ma costo elevato
- Riduco al minimo la presenza di blocchi liberi non sfruttati in cache



Indirizz. associativo e identificazione

- In una memoria associativa viene memorizzata l'etichetta dell'indirizzo di memoria centrale
- L'indirizzo di memoria centrale e suddiviso in due sole sezioni





Indirizzamento associativo: esecuzione

- La CPU genera un indirizzo e subito viene analizzata la cache
- I bit di etichetta dell'indirizzo di memoria centrale sono confrontati in modo associativo con tutti quelli della cache per vedere se il blocco è presente in cache
- Etichette lunghe
  - mem associativa più grossa, costosa e lenta del diretto
- Uso lo spazio più efficientemente
  - un blocco esce dalla cache solo se è piena
  - riduco in numero di miss (penalità)

Cache hit: metodi di scrittura

- Problema della coerenza
  - il blocco è in cache, ma scrivendo la parola solo in cache la memoria centrale non è più *allineata*
- Politiche di scrittura della parola
  - **Write-through (immediata)**
    - il processore scrive la parola sia in memoria centrale, sia nella cache e prosegue la sua attività
  - **Write-back (differita)**
    - il processore scrive solo in cache e prosegue attività
    - il blocco viene marcato come modificato e in seguito (tipicamente all'atto della rimozione per fare spazio) verrà ricopiato in cache

Cache hit: metodi di lettura

- Nessun problema!
- Il blocco è nella cache
- la parola viene letta dalla copia residente in cache
- Il processore continua subito la propria attività

Cache miss: lettura

- Il processore si sospende e il blocco deve essere caricato da mem centrale in mem cache
- Politiche di lettura della parola
  - **Read-through** (immediata)
    - il processore legge la parola non appena viene caricata in cache e riprende subito l'attività
  - **Read-back** (differita)
    - il processore attende che l'intero blocco sia stato caricato in cache, poi legge la parola e riprende attività

Cache miss: scrittura

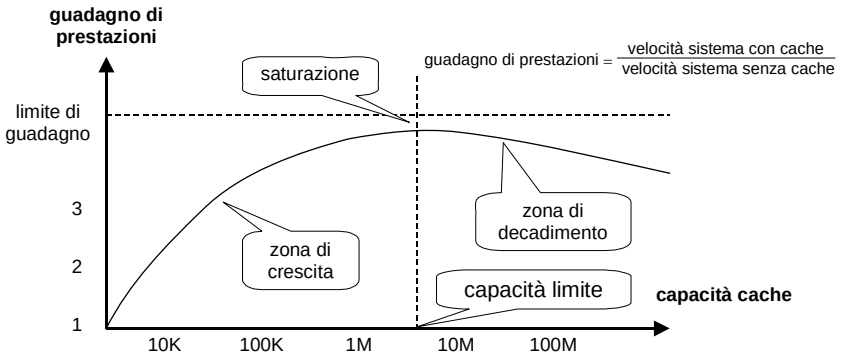
- L'operazione di scrittura è molto meno frequente della lettura
  - in generale non è obbligatorio caricare il blocco dalla mem centrale in cache
- Politiche di scrittura della parola
  - **Write-through** (immediata)
    - il processore bypassa la cache, scrive la parola solo in mem centrale e riprende attività
  - **Write-back** (differita)
    - il processore copia il blocco interessato nella cache
    - il processore scrive solo in cache e prosegue attività
    - il blocco viene marcato come modificato e in seguito (come nel caso write-back hit) verrà ricopiato in cache

Metodo di sostituzione

- Problema
  - si deve caricare un blocco in cache ma non esistono più blocchi liberi per accoglierlo
- Soluzione
  - sostituzione: scegliere un blocco della mem cache e liberarlo, per fare posto al nuovo
- L'algoritmo di sostituzione individua il blocco della cache da sostituire
    - per indirizzamento diretto il blocco è fissato
    - Least Recently Used (LRU): si sostituisce il blocco di memoria usato meno recentemente
    - Random: scelta a caso, realizzazione semplice
    - FIFO: tratto il set come coda circolare, rimpiazzo blocco in testa, realizzazione semplice

Limiti della cache

- Non serve aumentare le dimensioni della cache oltre una capacità limite, sopra cui le prestazioni del sistema di memoria **smettono** di aumentare o addirittura iniziano a **diminuire**



Limiti delle cache: motivazioni

- Se le dimensioni della cache superano i limiti di validità dei principi di località dei programmi, nella cache, oltre ai blocchi usati in modo intenso, vengono inevitabilmente a trovarsi
  - blocchi spesso **liberi** (e dunque sprecati)
  - blocchi **invecchiati**, che dopo un periodo di uso intenso vengono ormai acceduti poco di frequente (e che dunque si potrebbero lasciare senza danno in memoria centrale)
- Questo impedisce la crescita indefinita del guadagni di prestazioni, all'aumentare della capacità della cache (fenomeno di saturazione)
- Inoltre, all'aumentare delle dimensioni della cache il tempo di accesso alla cache aumenta (per motivi tecnologici), e dunque superando la capacità limite della cache le prestazioni del sistema di memoria dotato di cache iniziano a dacadere

Tecniche avanzate (cenni)

- Portare la cache sullo stesso chip della CPU
  - costoso, non praticabile se le dim sono grandi
- Aggiungo un livello intermedio
  - L1 on-chip, ck elevato, decine di Kbyte
  - L2 off-chip, dim L2> dimL1, Mbyte
- Cache dati e istruzioni separate
  - aumento il parallelismo
  - tuning mirato dei parametri
  - elevata complessità