Calcolatori Elettronici

La memoria gerarchica
Introduzione

La memoria - generalità

- ***Funzioni*:**
  - ***Supporto alla CPU*: deve fornire dati ed istruzioni il più rapidamente possibile**
  - ***Archiviazione*: deve consentire di archiviare dati e programmi, garantendone la conservazione e la reperibilità anche dopo elevati periodi di tempo**
- ***Esigenze diverse*:**
  - ***velocità* per il supporto alla CPU**
  - ***non volatilità* e *capacità* per l'archiviazione**

Caratterizzazione di una memoria

Alcuni parametri:

- ***Velocità***
  - ***tempo di accesso*** **(quanto passa tra la richiesta e la relativa risposta?)**
  - ***tempo di trasferimento*** **(quanti byte al secondo si possono trasferire?)**
- ***Capacità***
  - **quanti byte può contenere?**
- ***Costo***
  - **dati capacità e costo totale, qual è il costo per bit?**

Tecnologia ed organizzazione (1)

- ***Capacità, velocità e costo NON sono criteri indipendenti*:**
  - **la velocità si ottiene con capacità ridotta e tecnologie costose**
  - **la capacità si ottiene con tempi lunghi e tecnologie poco costose**
- **Le tecnologie *elettroniche* sono utilizzate per le memorie più *veloci***
- **Le tecnologie *magnetico/ottiche* sono utilizzate per le memorie più *capaci***

E' necessario combinare più unità di memoria

Tecnologia ed organizzazione (2)

E' necessario combinare più unità di memoria

- ***Obiettivo*:**

  Ottenere un *sistema di memoria* complessivo che

  - **fornisca all'utente una quantità di memoria pari a quella disponibile nella tecnologia più lenta**
  - **consentendo allo stesso tempo (*quasi sempre*) una velocità di accesso pari a quella garantita dalla tecnologia più veloce**

Tecnologia ed organizzazione (3)

- ***Osservazioni preliminari*:**

  - **la funzione di "supporto alla CPU" è svolta dalla memoria volatile**
  - **i tempi di funzionamento della memoria principale sono comunque sensibilmente maggiori dei tempi di funzionamento della CPU**
  - **se la memoria principale fosse più piccola i tempi di funzionamento si ridurrebbero (più piccolo è più veloce)**

Tecnologia ed organizzazione (4)

Combinare più unità di memoria volatile

Esempio (scenario semplificato)

- **sistema di memoria (volatile) formato da due unità, una piccola e veloce, e una grande ma più lenta**
- **i dati usati piu frequentemente si potrebbero tenere nella memoria veloce, in modo che la maggior parte delle volte il tempo di accesso sia quello della memoria veloce**
- **solo quando un dato non viene trovato nella memoria veloce lo si va a cercare nella memoria più lenta**

Tecnologia ed organizzazione (5)

Come si individuano i dati da mettere nella memoria veloce?

- **occorre poter prevedere con una certa previsione il comportamento dei programmi eseguiti**
- **il comportamento medio dei programmi è prevedibile, in base ad alcuni principi:**
  - **la *regola del 90/10***
  - **il principio di *località del riferimento***
    - ***località temporale***
    - ***località spaziale***

La regola del 90/10

- ***Principio empirico di progetto*:**
  - **investire risorse lì dove viene speso il tempo**
- ***Cosa accade durante l'esecuzione di un programma?***
  - **Regola del 90/10**

    il 90% del tempo viene speso sul 10% del codice (cicli….)

Il principio di località del riferimento (1)

- ***Località temporale***

  i programmi tendono a riutilizzare dati ed istruzioni che hanno usato recentemente

- ***Esempio***

```
while i < 100 {
  somma = somma + A[i]
  ..
  }
```

questa istruzione viene utlizzata ad ogni iterazione del ciclo

la variabile somma viene utlizzata ad ogni iterazione del ciclo

- ***In altre parole***

  se un'istruzione (o un dato) è stato utilizzata in un istante di tempo T, c'è un'elevata probabilità che la stessa istruzione (lo stesso dato) venga utilizzato in istanti di tempo "prossimi" a T

Il principio di località del riferimento (2)

- ***Località spaziale***

  dati vicini in memoria tendono ad essere utilizzati in istanti di tempo ravvicinati

- ***Esempi***
  - **istruzioni vicine**
  - **componenti di array durante scansioni sequenziali**

- ***In altre parole***

  se in un istante di tempo T è stata utilizzata una parola di memroia di indirizzo X, c'è un'elevata probabilità che in istanti di tempo "prossimi" a T vengano usate parole di memoria vicine a X

Sfruttiamo la regola del 90/10

- **per migliorare le prestazioni posso cercare di *abbassare il tempo necessario per gli accessi a memoria relativi al 10% del codice responsabile del 90% del tempo di esecuzione***
- **il tempo di accesso in memoria dipende dalla tecnologia usata per la realizzazione della memoria e dalla sua capacità**

Tecnologia
+ veloce = + costoso

Capacità
+ piccolo = + veloce

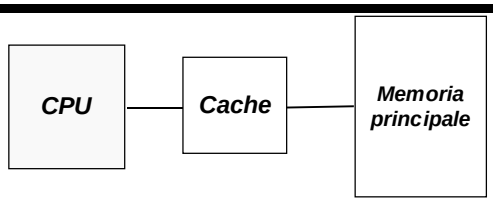
cerchiamo di mettere il "famoso" 10% in una memoria piccola e costruita con tecnologia "veloce"

Combiniamo 90/10 e località

cerchiamo di mettere il "famoso" 10% in una memoria piccola e costruita con tecnologia "veloce"

- **E' difficile, richiederebbe un'analisi complessa del codice**
- ***Sfruttiamo il principio di località***
  - **mettiamo i dati usati recentemente nella memoria piccola e veloce (*loc. temporale*)**
  - **quando spostiamo un dato nella memoria piccola e veloce portiamoci dietro anche le parole vicine (*loc. spaziale*)**

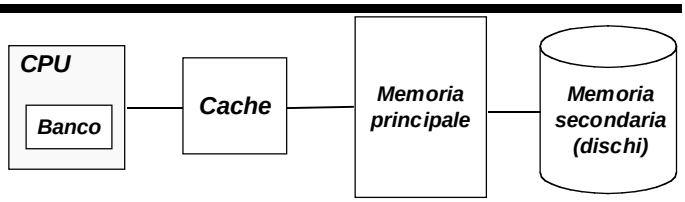
Un sistema con due livelli di memoria



Dimensioni	**< 1 MB**	**< 4 GB**
Tempo accesso **(nanosec)**	**< 10**	**< 100**
Banda passante **(MB/sec)**	**5.000 + 10.000**	**1.000 + 5.000**

quantità di informazione trasferita nell'unità di tempo

Il sistema di memoria gerarchica



Dimensioni	< 1KB	< 1 MB	< 4 GB	> 20 GB
Tempo accesso (nanosec)	< 0,5	< 10	< 100	10.000.000
Banda passante (MB/sec)	20.000 ÷ 100.000	5.000 ÷ 10.000	1.000 ÷ 5.000	20 ÷ 40

Aspetti tecnologici

- La *cache* è realizzata in tecnologia *SRAM* (RAM statica)
  - il bit è memorizzato in un flip-flop (elemento stabile)
- La *memoria principale* è realizzata in tecnologica *DRAM* (RAM Dinamica)
  - *vantaggio*: il bit è rappresentato dalla capacità tra gate e canale di un mosfet (il singolo bit occupa meno spazio, inoltre è meno costoso)
  - *svantaggio*: la carica tende a dissiparsi
  - la memoria deve essere "*rinfrescata*" ogni tanto (riletta e riscritta tutta prima che i valori dei bit vengano persi) *REFRESH*
  - le letture tendono a dissipare la carica, quindi ogni lettura è seguita da una scrittura
- La *memoria secondaria* è realizzata mediante dischi magnetici

Dove sono dati e istruzioni

- **un programma non accede a tutto il suo codice con uguale probabilita**
- **facciamo un modo che *la maggior parte delle volte* i dati e le istruzioni cercate si trovano nella cache**
- ***per la località***
  - **mantenere in cache i dati usati più recentemente**
  - **spostare "blocchi" di parole e non parole singole**

Il blocco è la più piccola quantità di informazione spostata tra due livelli di memoria contigui

- **non sempre la parola cercata sarà in cache**
  - **se c'è è "*accesso di tipo hit*" (colpito)**
  - **se non c'è è "*accesso di tipo miss*" (mancato)**
- **se si verifica un miss devo cercare la parola nel livello di memoria seguente**

Un sistema con due livelli di memoria (2)