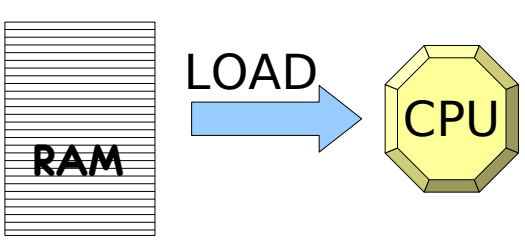
La memoria centrale

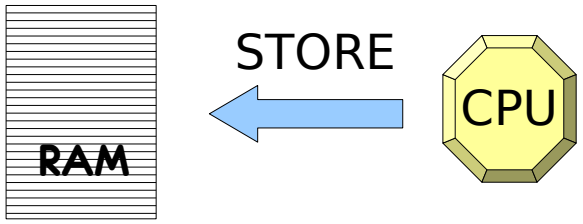
- La **memoria centrale** (**RAM**) è l'unica memoria di grandi dimensioni direttamente accessibile alla CPU
- È un vettore di parole di memoria

RAM

- L'interazione avviene:

  tramite le istruzioni esplicite load (copiatura da RAM a registro) e store (copiatura da registro a RAM),





  la CPU preleva direttamente dalla RAM le istruzioni da eseguire (*fetch* delle istruzioni)

- *L'ideale sarebbe conservare tutti i programmi e i dati in RAM*
- *... si fa così?*

No!

Perché:

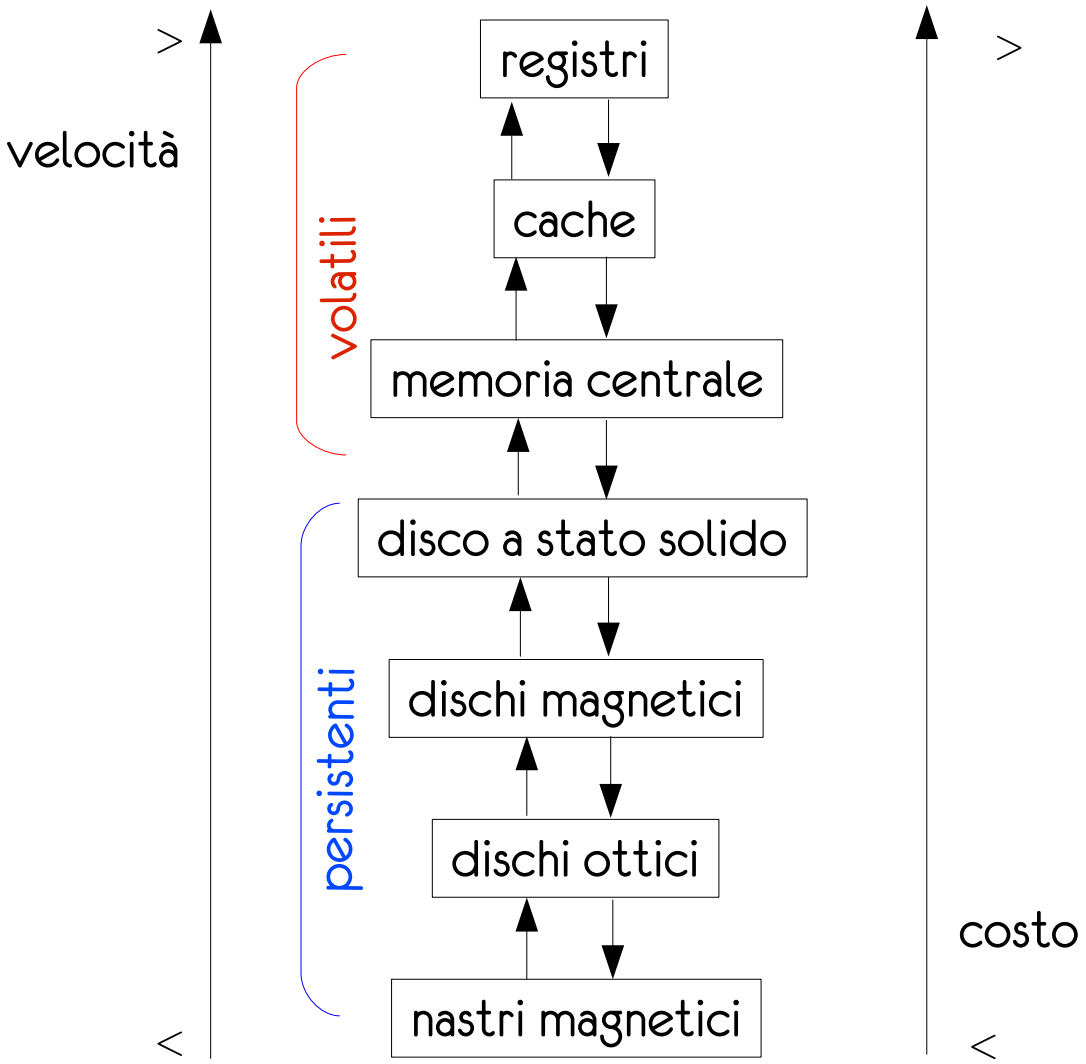
- Dispositivo di memorizzazione volatile (si cancella con l'interrompersi dell'erogazione dell'energia elettrica)
- Ha costi elevati quindi di norma ha una capacità limitata
- Occorrono ulteriori dispositivi di memoria, persistenti, capaci, meno costosi ...

La memoria secondaria

- La memoria secondaria è una memoria permanente (contrario di volatile)
- Ha una capacità elevata

- Di solito è costituita da un disco magnetico, però esistono molti tipi di supporti di memoria (permanente) diversi. Fra questi:
  - memorie a stato solido
  - nastri
  - chiavi USB
  - CD, BlueRay (memorie ottiche)
- I differenti supporti si differenziano in costo e velocità d'accesso e sulla base di tali caratteristiche possono essere ordinati in una scala gerarchica

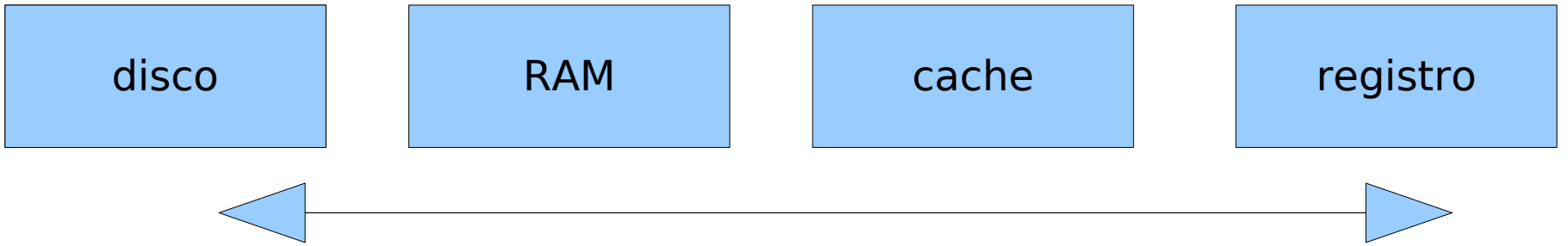
Gerarchia delle memorie



RAM
durante il normale funzionamento conserva i dati in un vettore che è volatile. Se manca la corrente il controller copia il contenuto su disco, rendendolo così permanente. A tal fine occorre appoggiarsi ad una batteria ausiliaria

le memorie sono dispositivi di I/O

I dati viaggiano attraverso le memorie

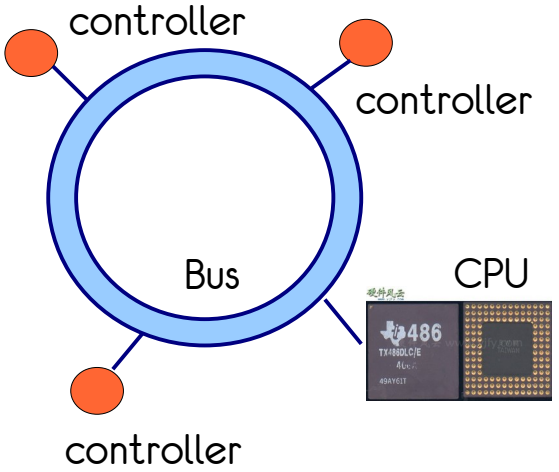


Dispositivi di I/O

- Un computer è costituito da una CPU e da un certo numero di controller di dispositivi, tutti connessi da un bus
- Ogni controller gestisce uno o più dispositivi di un certo tipo, ha una memoria interna (buffer) e dei registri specializzati
- Ogni controller ha associato un driver (un software), che funge da interfaccia verso il resto del sistema

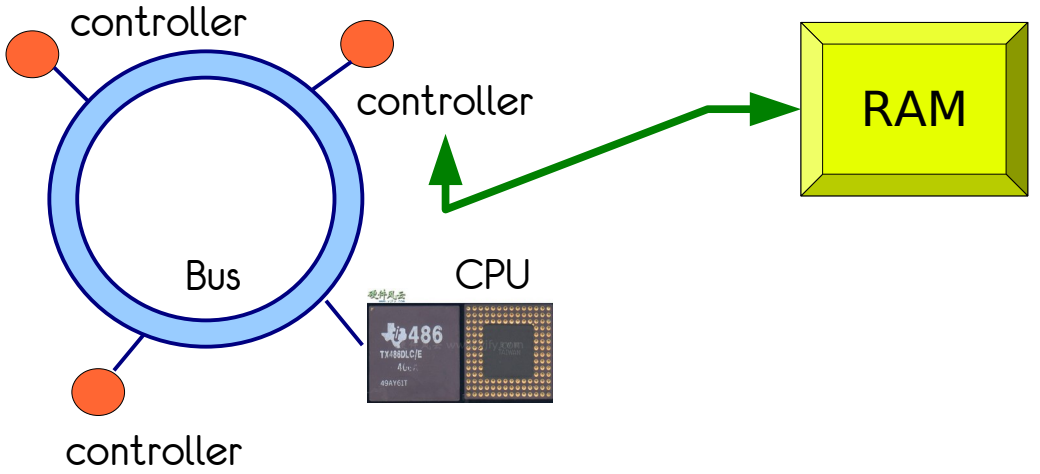
Operazione di I/O su di un certo dispositivo

il **driver** del dispositivo carica i registri del controller
il **controller** esamina i registri e sceglie un'operazione
il **controller** esegue il trasferimento dati
il **controller** avvisa il driver che l'op. è conclusa
il **driver** avvisa il SO passandogli lo stato di terminazione
se l'op. era di lettura restituisce anche i dati letti, tempo-raneam. memorizzati nel buffer locale



Dispositivi di I/O

- **Nota**: per rendere ancora più veloci le operazioni relative a un dispositivo di memoria si adotta la tecnica nota come **Direct Memory Access** (DMA)
- **Non coinvolge la CPU**
- per trasferire dati in memoria (o dalla memoria), una volta caricato il buffer e impostati i puntatori necessari, il trasferimento viene gestito dal controller che copia in o da memoria centrale direttamente



- un interrupt segnala il completamento dell'operazione

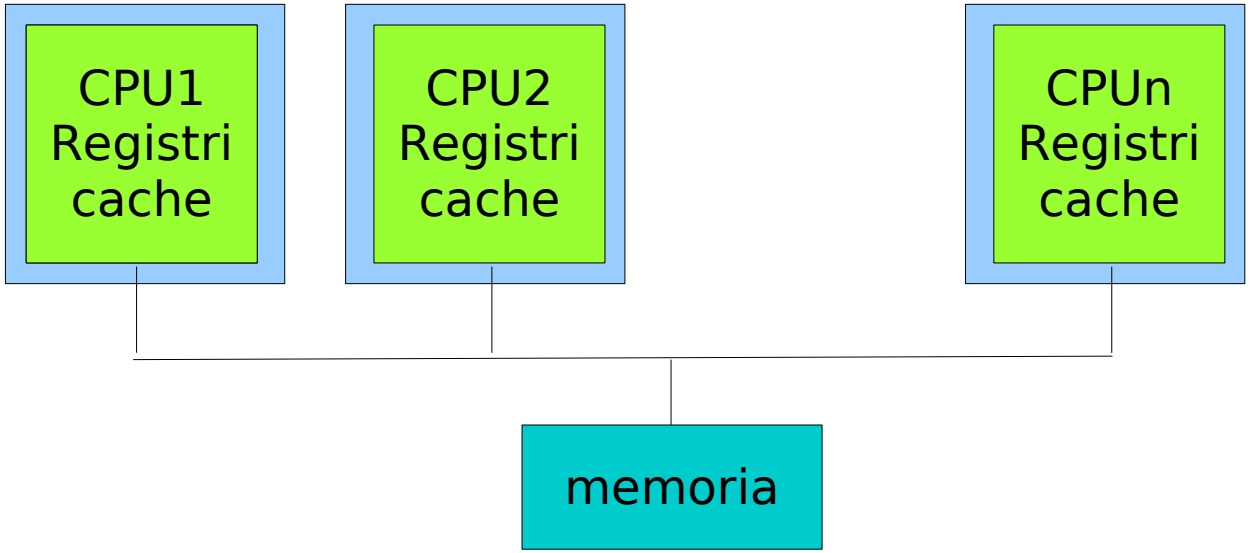
Architetture degli elaboratori

- **Monoprocessore**: una CPU. Talvolta presenti processori ausiliari dedicati ad attività specifiche, es:
  - Controller di device
  - Processori dedicati all'I/O
- **Multiprocessore**:
  - Maggiore capacità elaborativa
  - Più processori condividono device e risorse: minore costo rispetto a usare tanti sistemi monoprocessore
  - Maggiore affidabilità: un guasto a una CPU non impedisce alle altre di funzionare
  - Overhead: costo legato alla gestione dei processori e alla sincronizzazione nell'uso delle risorse

(*) Overhead: sovraccarico, tipicamente un costo in termini di qualche risorsa come il tempo di esecuzione

Sistemi multiprocessore

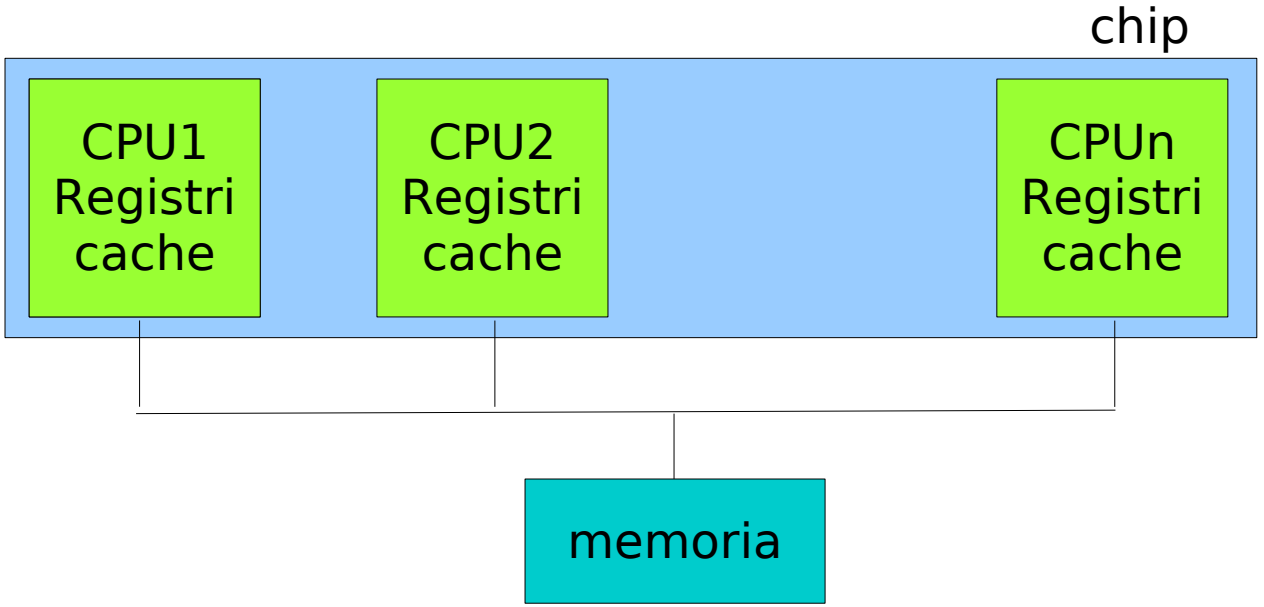
- Multielaborazione asimmetrica: esiste un processore principale che organizza e assegna il lavoro ai processori secondari
- Multielaborazione simmetrica: i processori sono pari e possono svolgere gli stessi compiti



- La scelta architetturale può dipendere sia dall'HW sia dal SO

Sistemi multiprocessore

- Single core: Ogni CPU è realizzata su un chip separato
- Multi core: Uno stesso chip contiene diverse CPU



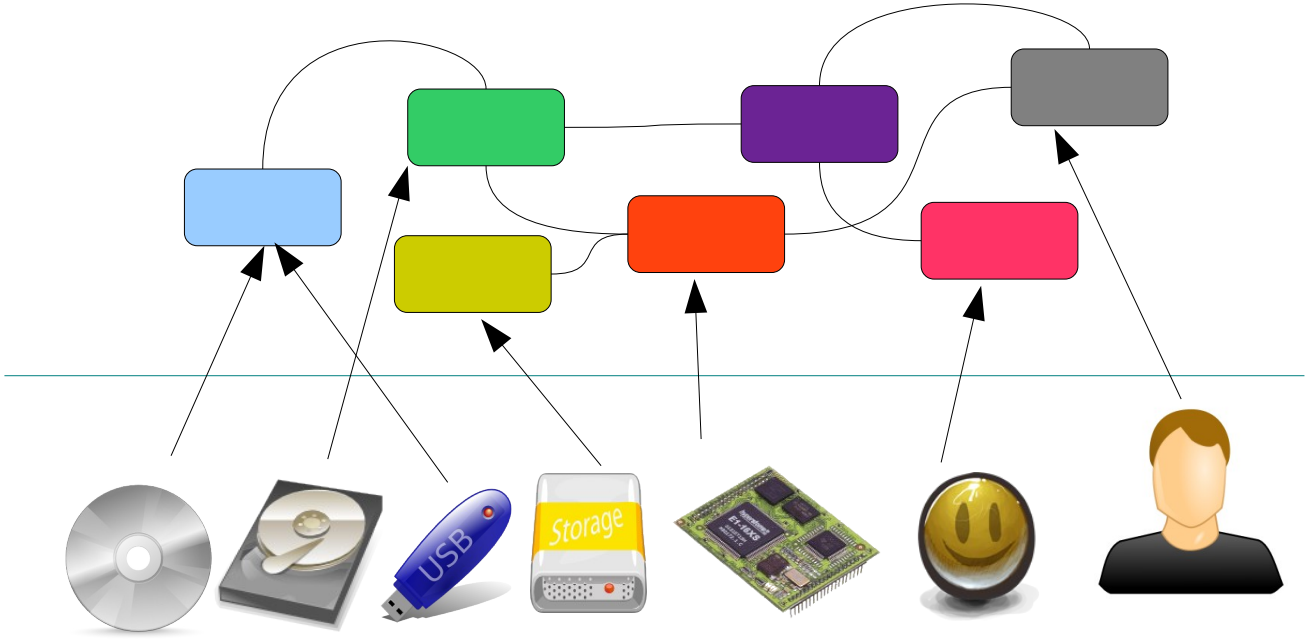
Cluster di elaboratori

- Sistemi multiprocessore costituiti da insiemi di elaboratori completi (detti nodi)
- Ogni nodo può essere single o multi-core
- È necessario uno strato software per il controllo del cluster, esempio:
  - Se un nodo va in crash il sistema di controllo riavvia I processi che erano in esecuzione
  - Distribuzione dei processi da eseguire fra I vari nodi in modo da evitare sovraccarichi e mantenere alte le prestazioni

Astrazione

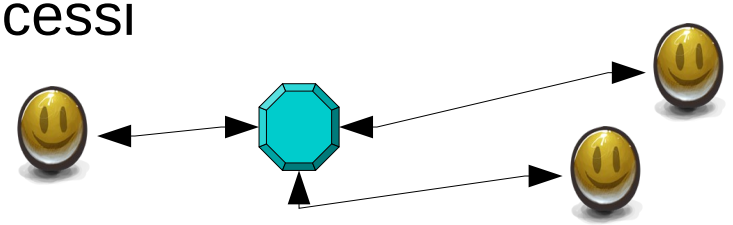
RAPPRESENTARE: definire astrazioni che rappresentano gli oggetti da gestire (processi, porzioni di memoria, risorse). Problema: quali informazioni occorre mantenere?

GESTIRE: quali strategie/politiche applicare?



Caratteristiche dei SO

① multiprogrammazione: il SO gestisce contemporaneamente un insieme di job (processi, task); deve quindi distribuire l'uso della CPU fra i vari processi



Caratteristiche dei SO

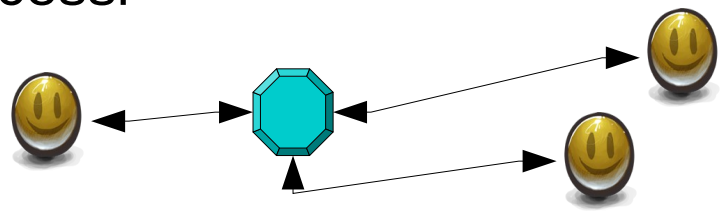
- multiprogrammazione: il SO gestisce contemporaneamente un insieme di job (processi, task):

  occorre definire delle astrazioni (strutture dati) per rappresentare I processi gestiti e strutture dati per mantenere insiemi di processi, occorre definire politiche per decidere quali processi devono essere attivi e per quanto tempo:

  - come viene rappresento un processo?
  - quali strutture usare per organizzare e gestire i processi?
  - quali politiche definire per ottimizzare l'esecuzione?

Caratteristiche dei SO

① multiprogrammazione: il SO gestisce contemporaneamente un insieme di job (processi, task); deve quindi distribuire l'uso della CPU fra i vari processi



② partizione del tempo (mutitasking): il processi possono appartenere a utenti diversi -> estensione della multiprogrammazione in cui si tiene conto dell'interazione con l'utente, la commutazione della CPU viene effettuata in modo tale da creare un parallelismo virtuale agli occhi dell'utente, che ha la percezione che solo il suo job sia in esecuzione

① + ② multiprogrammazione e multitasking avvengono tramite lo scheduling della CPU

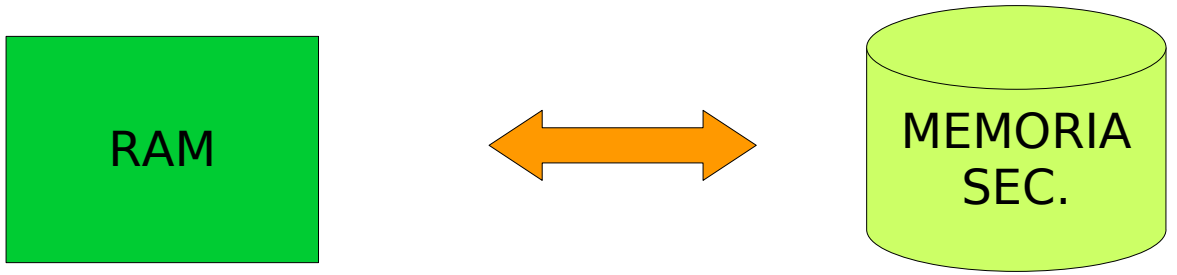
Caratteristiche dei SO

- multiprogrammazione: il SO gestisce contemporaneamente un insieme di job (processi, task)
- mutitasking: estensione della multiprogrammazione in cui si tiene conto dell'interazione con l'utente, la commutazione della CPU viene effettuata in modo tale da creare un parallelismo virtuale agli occhi dell'utente, che ha la percezione che solo il suo job sia in esecuzione (partizionamento del tempo)
- **occorre gestire gli utenti,** definire astrazioni che catturano le informazioni necessarie relative agli utenti, definire eventualmente gruppi di utenti, definire politiche di accesso (chi ha diritto ad eseguire quali operazioni e su quali file), ecc.:
  - come rappresentare gli utenti?
  - come modificare le politiche di gestione per tener conto degli utenti?
  - come interagire con gli utenti?

Caratteristiche dei SO

La memoria principale può non essere sufficiente a contentere i lavori (job) da eseguire, che possono essere conservati su **memoria secondaria**: la gestione di quali processi spostare in mem. principale (e viceversa) è detta job scheduling.

Il SO gestisce il **file system**



Caratteristiche dei SO

??? riuscite a immaginare quali tipi di informazioni occorre gestire, quale tipo di strutture occorre aggiungere per ottenere questa caratteristica **???**

Caratteristiche dei SO

Ogni aggiunta di caratteristiche mirate ad aumentare la flessibilità, la “potenza” di un SO introduce **sovrastrutture**, da gestire, cosa che comporta:

- uso di parte della memoria per mantenere tali strutture
- uso di parte del tempo di computazione per la gestione di tali strutture

È fondamentale che i SO siano implementati in modo altamente efficiente