SISTEMI OPERATIVI

06.a

Il Nucleo

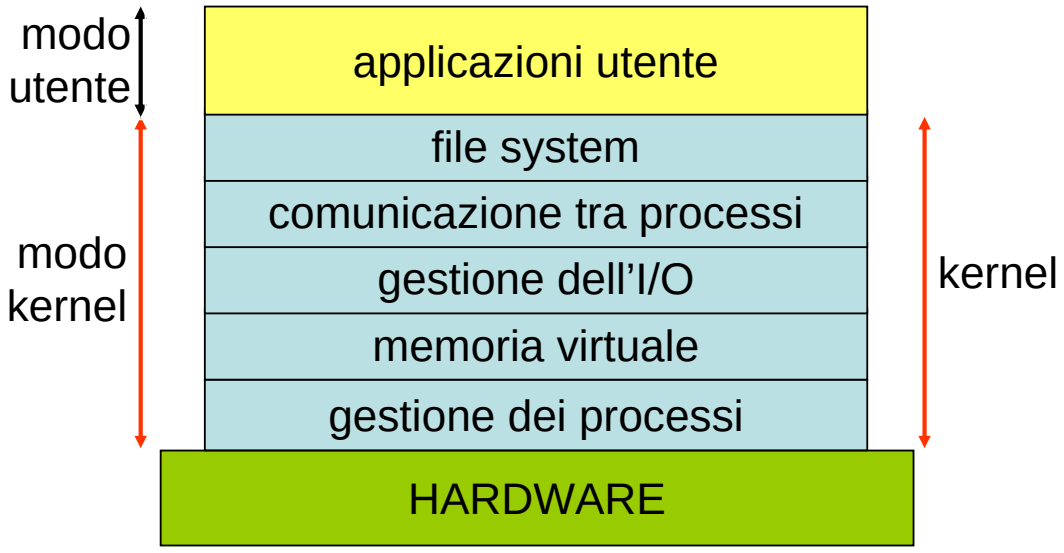
Nucleo di un SO

- Il nucleo di un SO
- Gestione delle interruzioni
- Sincronizzazione tra processi
- Dispatcher

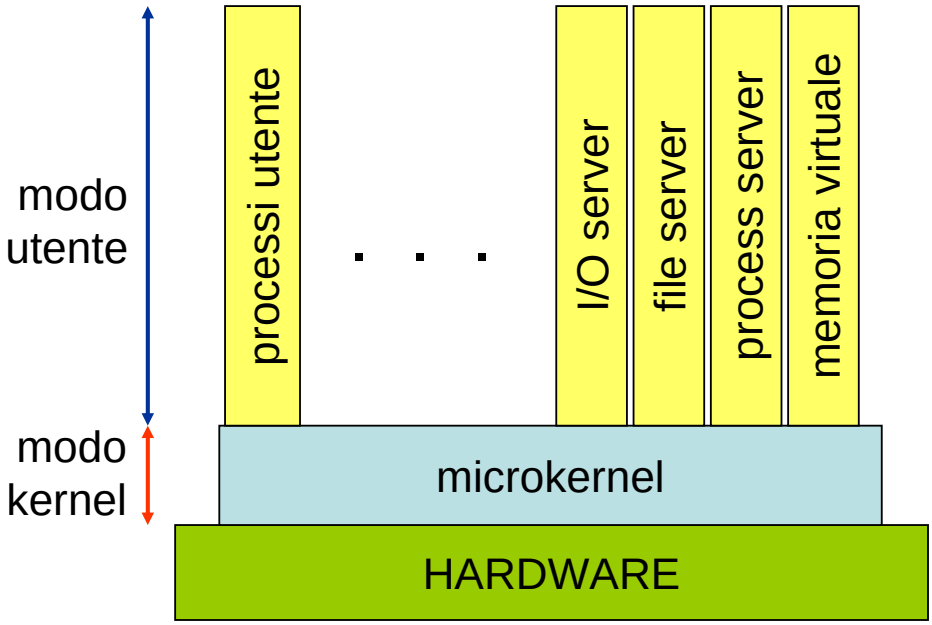
Architettura di base dei SO

- Due le strutture di riferimento:
  - a strati o livelli (layered):
    - ciascun livello fornisce funzionalità al livello superiore, usando (e nascondendo) quelle del livello inferiore,
    - il livello più "basso" del SO è il nucleo (kernel)
    - funzioni di gestione delle interruzioni, della memoria, dei processi e delle comunicazioni tra processi;
  - di tipo microkernel:
    - solo le funzioni essenziali fornite dal kernel (gest. minima della memoria, scheduler minimo, comunicazione tra processi)
    - le altre funzioni sono affidate a processi (eseguiti in modo supervisore o in modo utente) trattati dal microkernel alla stregua degli altri processi;
- La soluzione proposta nel testo è del tipo a livelli.

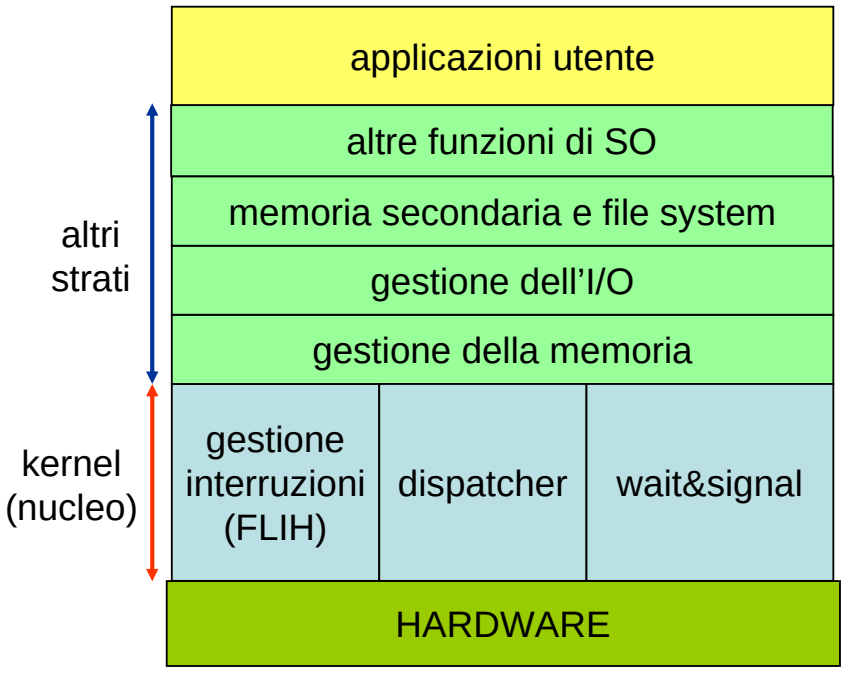
SO con architettura a strati



SO con architettura microkernel



Struttura di riferimento del SO



Funzioni essenziali nell'hardware

- Sistema di interruzioni
- Interruzioni software: System Calls o Supervisor Calls (SVC)
- Meccanismo di protezione della memoria
- Istruzioni privilegiate (modo supervisore/modo utente)
- Orologio in tempo reale (real time clock).

Funzioni essenziali nel kernel

- Gestione interruzioni
- Gestione delle sincronizzazioni tra processi (semafori con primitive wait e signal)
- Funzioni di multiprogrammazione (scheduler / dispatcher)

- La gestione delle interruzioni e delle sincronizzazioni tra processi sono strettamente legate alle gestione del processore (dispatching)
- Le funzioni legate alla creazione dei processi sono trasversali (riguardano l'allocazione di risorse quali memoria, dispositivi ecc.)

Sistema di interruzioni

- L'hardware fornisce le funzioni di:
  - commutazione del contesto
    - viene salvato il contesto del programma che viene interrotto
  - riconoscimento della interruzione
    - es. sistema di interrupt vettorizzato
    - attivazione della routine di servizio (ISR = Interrupt Service Routine) appropriata
  - gestione dei livelli di priorità
    - per stabilire quale richiesta di interruzione servire per prima, in presenza di più richieste,
    - per stabilire da quali richieste di interruzione può essere interrotta l'esecuzione di una ISR.

First Level Interrupt Handler

- Supporto efficiente dell'hardware per svolgere in modo (rapido) le 3 funzioni indicate.
- Alcune di quelle funzioni possono essere integrate dal software
  - riconoscimento in polling nella FLIH per un processore non dotato di interrupt vettorizzati
  - attivazione della ISR specifica
    - in questo caso la ISR risulta costituita da due parti:
      ISR = FLIH + ISR specifica.

Gestione delle interruzioni

- Le routine di servizio delle interruzioni fanno parte del nucleo del SO
- Le interruzioni esterne (richieste da dispositivi di I/O) e quelle interne (SVC - SuperVisor Call o SC - System Call), sono gestite allo stesso modo.
- Quando si verifica un'interruzione il processore si porta ad operare in modo supervisore: le routine di servizio delle interruzioni sono eseguite in modo supervisore.

Interruzioni nei sistemi monotask

- In un sistema non multitasking, la routine di servizio di una interruzione (ISR):
  - esegue il servizio richiesto (ad es. acquisisce un carattere),
  - rimuove la causa che ha generato la richiesta di interruzione,
  - esegue l'istruzione di ritorno da interrupt (RTI), che ripristina il contesto del programma che era stato interrotto, il quale può quindi proseguire la sua esecuzione.

Interruzioni nei sistemi multitask

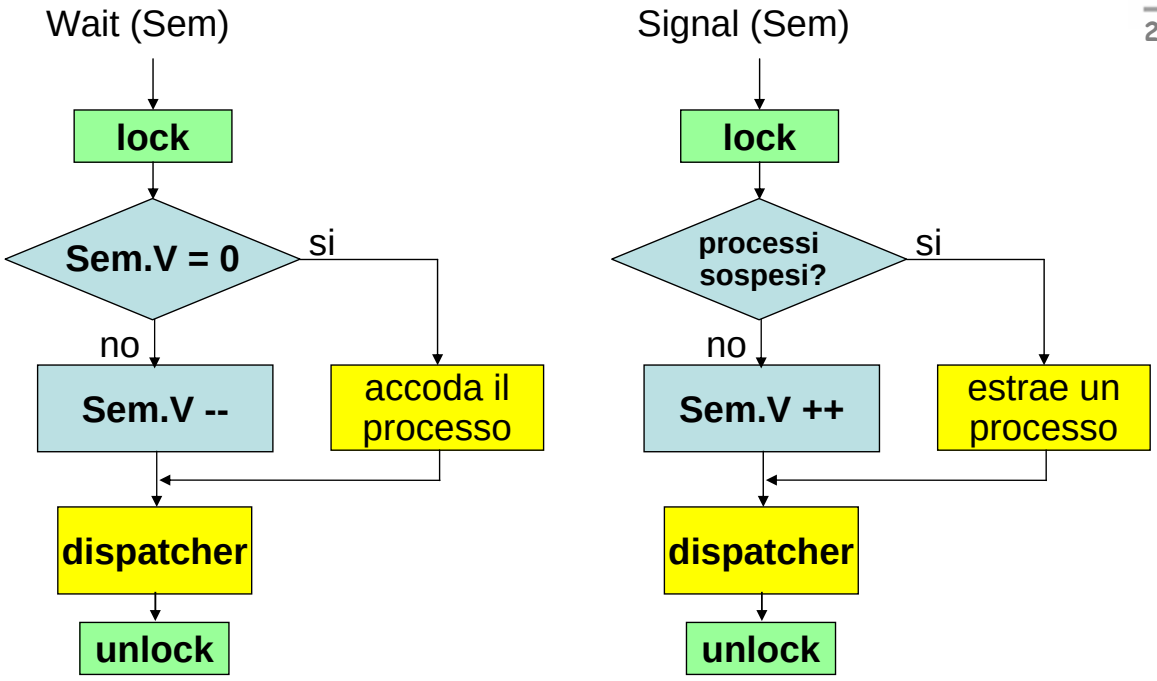
- In un sistema multitasking:
  - il servizio di una interruzione può comportare la modifica dello stato di un processo (ad es. da Waiting a Ready);
  - le ISR terminano cedendo il controllo allo scheduler (anziché al processo interrotto);
  - lo scheduler decide (in base alla politica adottata) se:
    - far proseguire il processo che era stato interrotto, o se
    - rendere Running un altro processo (scelto tra quelli Ready, ponendo nello stato Ready quello interrotto).

Gestione delle sincronizzazioni

- Il kernel fornisce le primitive di sincronizzazione tramite semafori con sospensione dei processi,
- Wait e Signal sono funzioni del kernel, attivate dai processi tramite System Call (SVC),
- La gestione della coda di processi associata al semaforo può essere affidata alle routine Wait e Signal del kernel:
  - Wait inserisce nella coda del semaforo il processo che non può proseguire,
  - Signal estrae dalla coda del semaforo un processo (se c'è) e lo pone Ready,
- Wait e Signal terminano attivando il dispatcher che decide quale processo sarà running ed esegue le eventuali commutazioni di contesto.

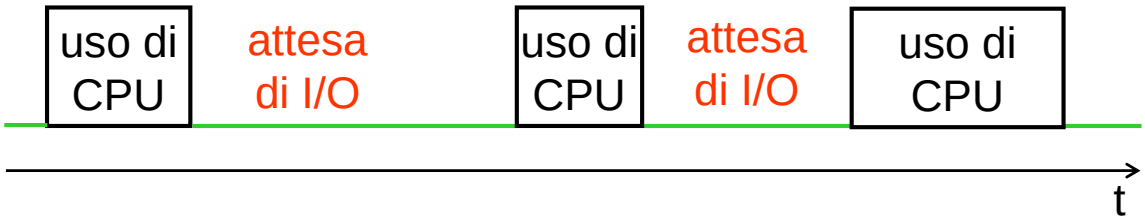
Wait e Signal

- Semaforo sospensivo



Scheduler: evoluzione dei processi

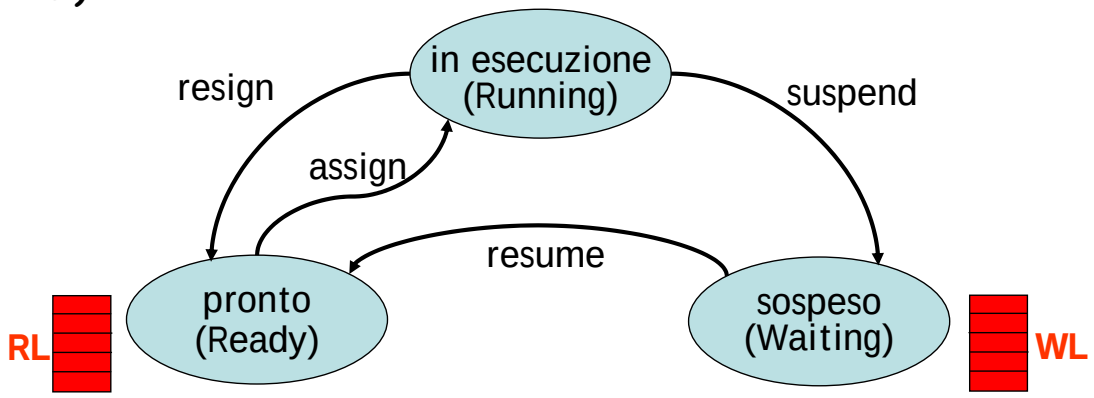
- Un processo, nella sua evoluzione, alterna:
  - intervalli di tempo in cui usa la CPU,
  - intervalli di tempo in cui attende che si completino operazioni di I/O o sincronizzazioni.



- In un SO multitasking lo scheduler consente di utilizzare il processore per eseguire altri processi durante i tempi in cui un processo è in attesa.

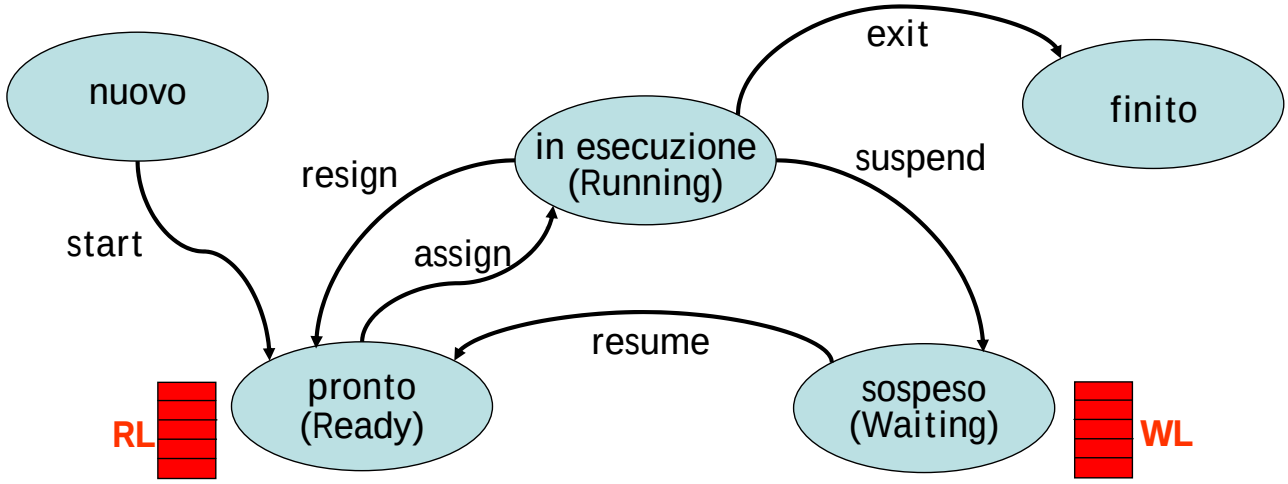
Stati di un processo

- In un SO multitasking un processo può trovarsi in uno dei 3 stati (e subire le transizioni di stato) indicati in figura:
  - in esecuzione Running: un solo processo per processore
  - pronto Ready: una lista di processi (ready list - RL)
  - in attesa Waiting: una lista di processi (waiting list - WL)



Modello a 5 stati

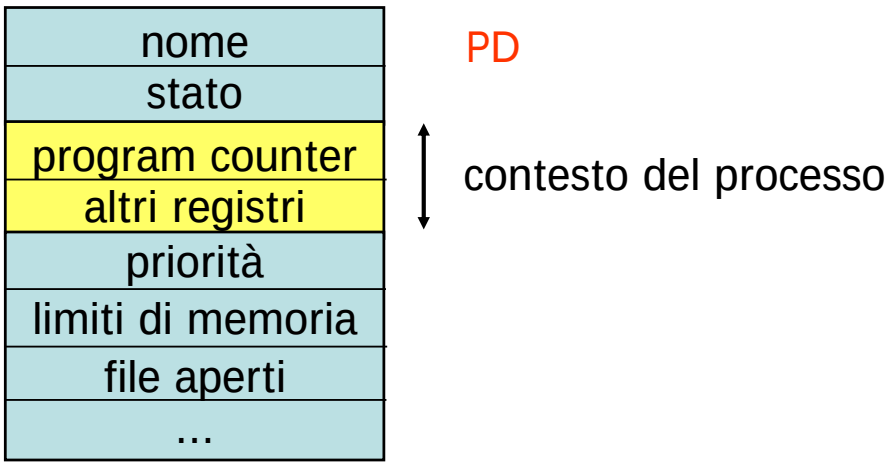
- L'attivazione può essere successiva alla creazione
- Può essere conveniente conservare qualche informazione anche dopo il completamento del processo



Descrittore di processo

- Se un processo si trova nello stato Ready o Waiting, il suo descrittore si trova in una coda associata a quello stato
- Il descrittore di processo (**PD: Process Descriptor**, o anche PCB: Process Control Block) è una struttura di dati che contiene informazioni associate al processo
  - nome e/o id del processo
  - stato del processo (waiting, ready ecc.)
  - contesto del processo (registri del processore, altri registri, ecc.)
  - altre informazioni per la gestione del processo (risorse possedute, informazioni per la gestione della memoria, ecc.)

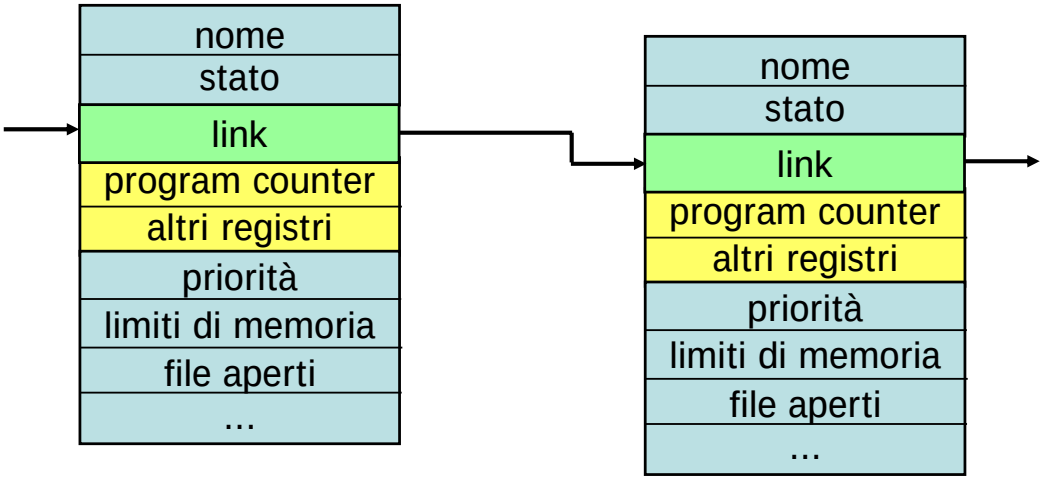
Descrittore di processo [2]



- Quando un processo esce dallo stato Running, i registri del processore (contesto) vengono ricopiati nel suo PD
- Quando un processo ritorna nello stato Running, il contesto viene ricopiato dal suo PD ai registri del processore

Liste di PD

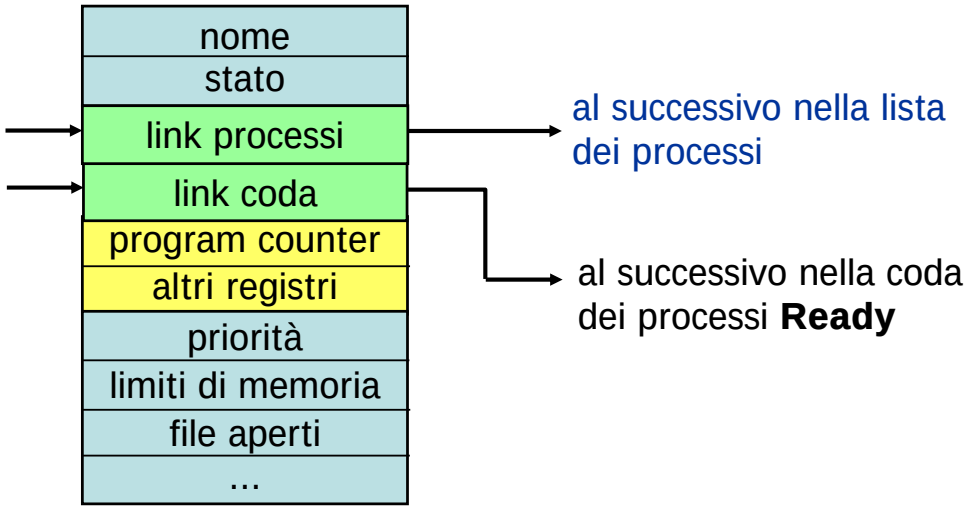
- Sono organizzate come liste concatenate: in questo modo le operazioni di inserzione e di estrazione sono rapide:



- Sono presenti più campi link (puntatore), se si prevede che un PD possa appartenere a più liste
- Possono essere utilizzate strutture più complesse (es. coda a priorità)

Liste di PD [2]

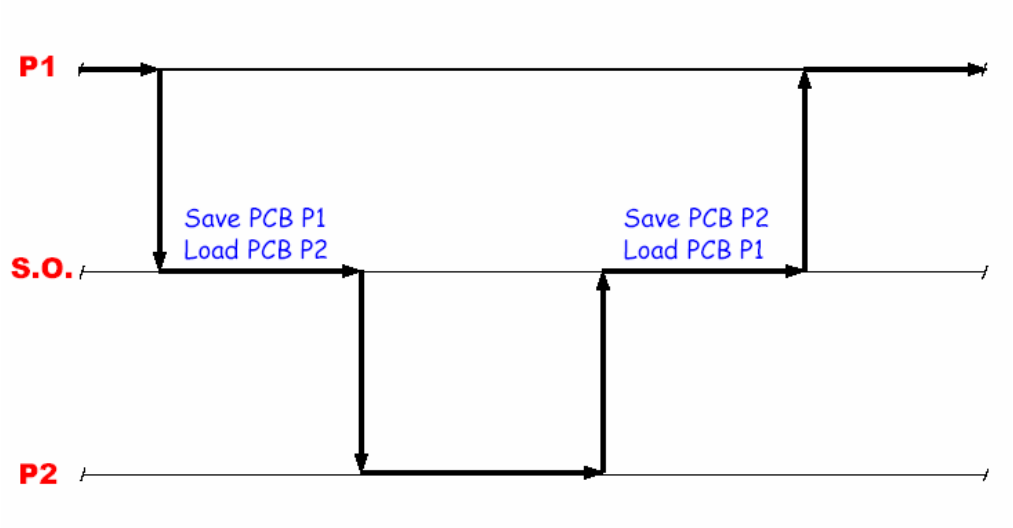
- Un PD può appartenere alla lista dei processi e ad una coda (ad es. alla Ready List)



Il dispatcher

- Il dispatcher (scheduler a basso livello) ha il compito di estrarre tra i processi nella Ready List quello da far diventare Running; quindi il dispatcher:
  - commuta il contesto (ricopia nel PD del processo Running i registri del processore e inserisce nei registri i valori estratti dal PD del processo selezionato per diventare Running);
  - porta il processore ad operare in modo utente;
  - inserisce nel Program Counter l'indirizzo della successiva istruzione del processo Running.
- Lo scheduler ad alto livello modifica le priorità dei processi Ready e Waiting. Lo scheduler ed il dispatcher interagiscono tra loro per ottenere il risultato voluto secondo la politica adottata.

Cambio di contesto



Interventi del dispatcher

- Il dispatcher interviene, per selezionare il processo da rendere **Running**, nelle seguenti circostanze:
  - quando il processo **Running** passa nello stato **Waiting** (ad es. per una richiesta di I/O),
  - quando il processo **Running** passa nello stato **Ready** (ad es. per il verificarsi di una interruzione),
  - quando un processo **Waiting** passa nello stato **Ready** (ad es. per il completamento di una sincronizzazione),
  - quando il processo **Running** termina.

Fine

Il Nucleo