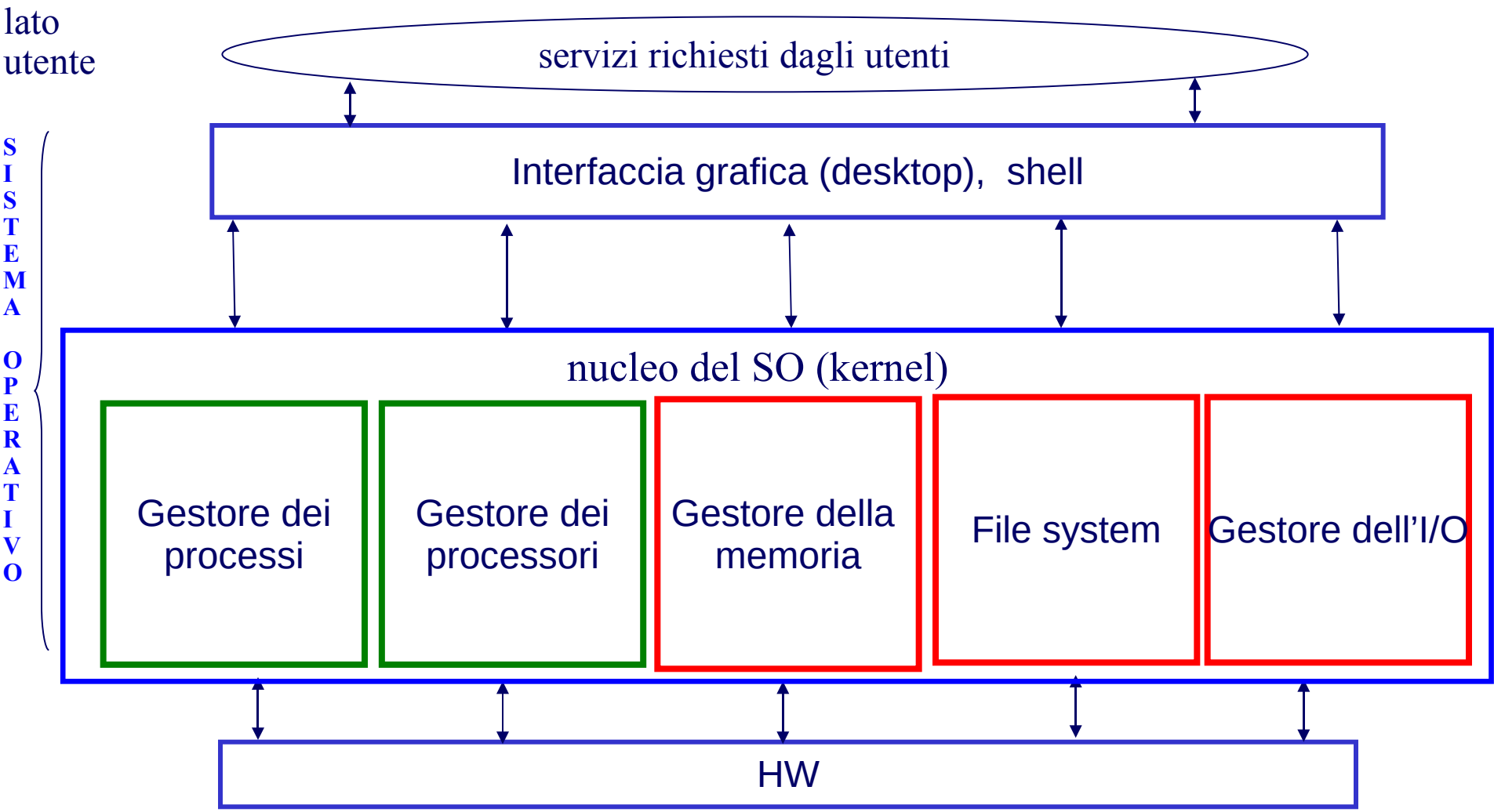
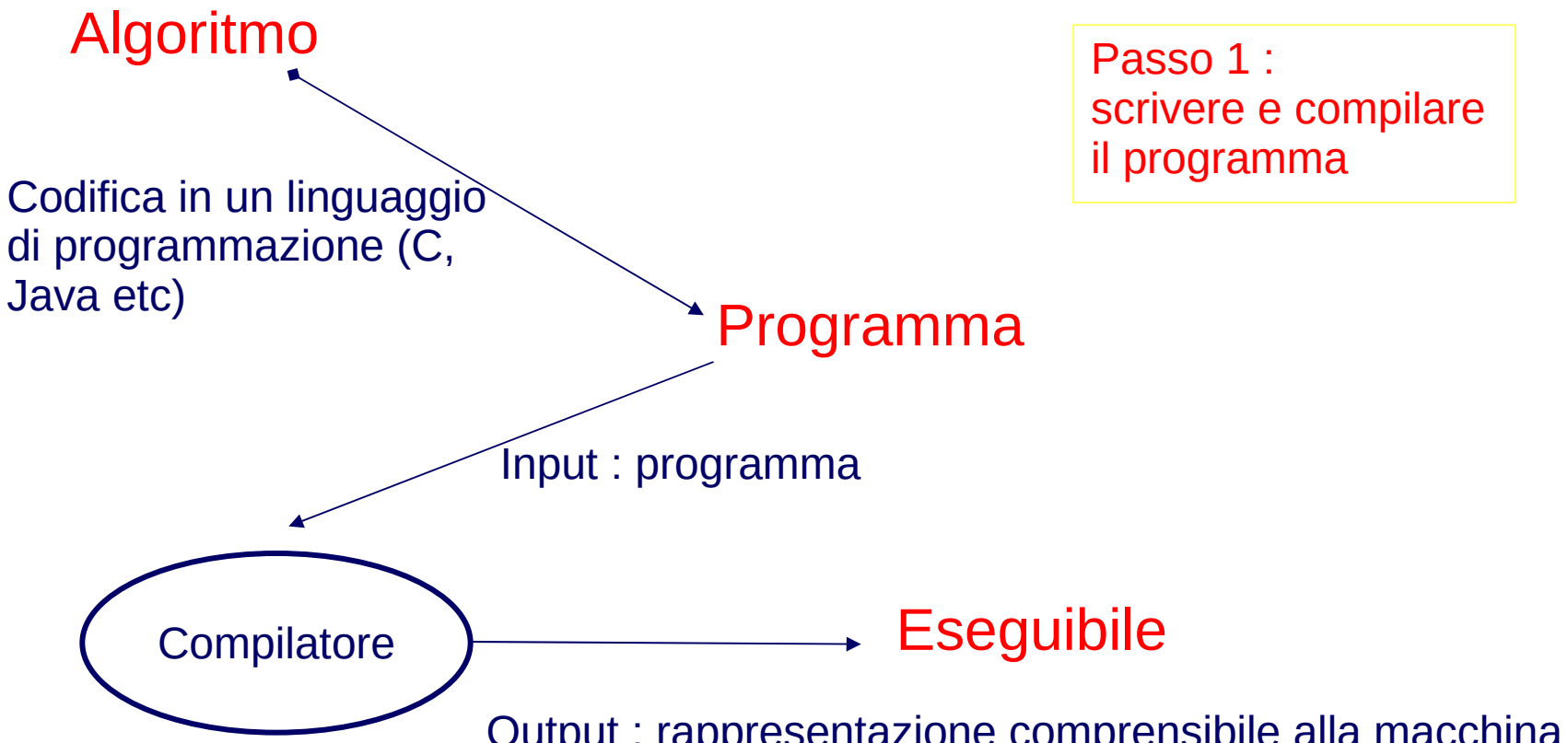
Informatica Generale
07 - Sistemi Operativi:Gestione dei processi

- **Cosa vedremo:**
  - Esecuzione di un programma
  - Concetto di processo
  - Interruzioni
  - Sistemi monotasking e multitasking
  - Time-sharing
  - Tabella dei processi
  - Stati di un processo e transizioni di stato

Quali sono le parti di un SO ?



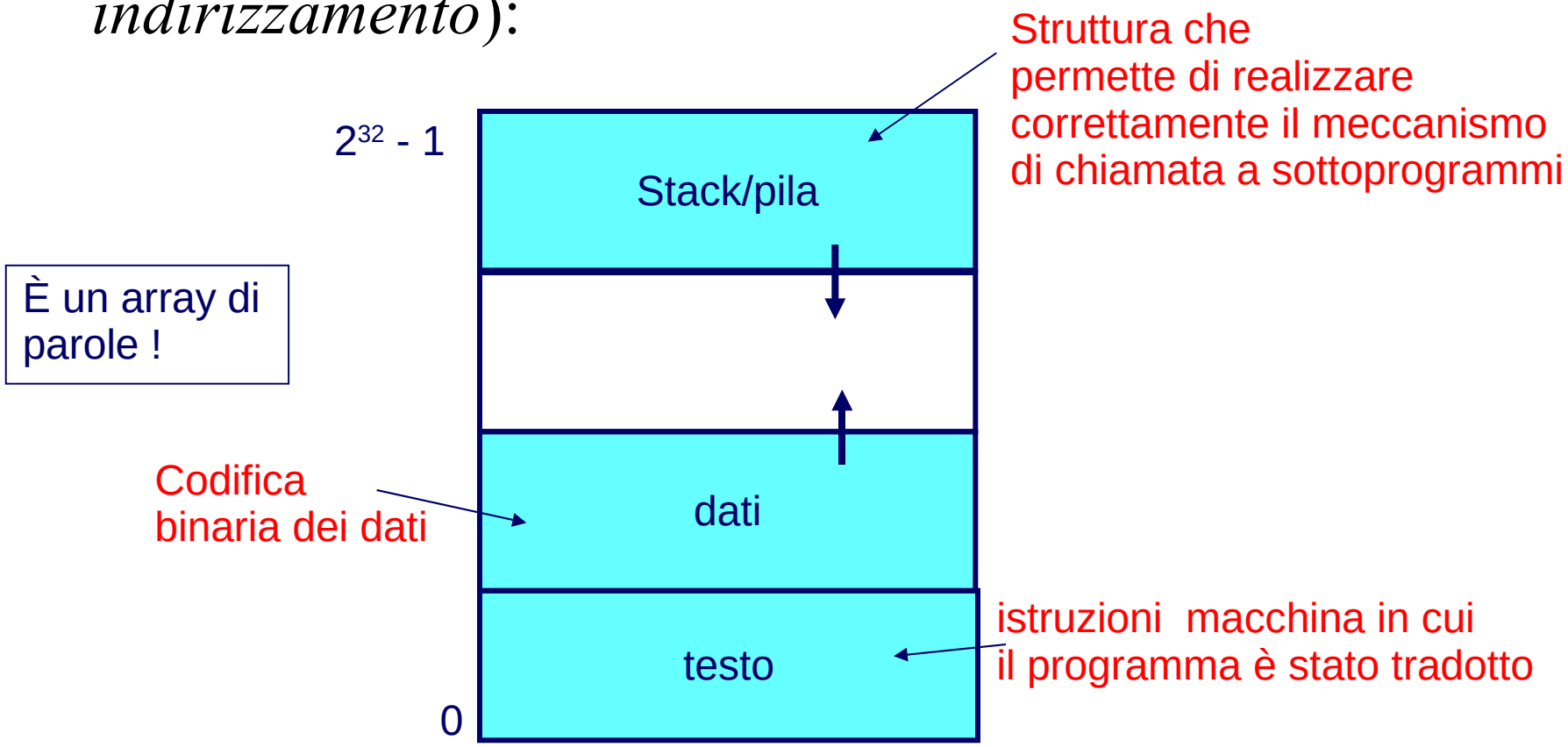
Esecuzione di un programma



Passo 1 :
scrivere e compilare il programma

Esecuzione di un programma 2

- Come è organizzata la rappresentazione binaria eseguibile del programma (*spazio di indirizzamento*):



Esecuzione di un programma 3

- Passo 2 :
  - ricopiare lo spazio di indirizzamento di un programma da memoria secondaria a RAM

Esecuzione di un programma 4

- Passo 3 :
  - modificare il PC del processore in modo che contenga correttamente l'indirizzo della prima istruzione macchina da eseguire nel nostro programma

Esecuzione di un programma 5

- Quando un programma utente è stato attivato il processore esegue una dopo l'altra le istruzioni assembler che lo compongono
- Un programma in esecuzione viene detto **processo** (*parte statica* [il programma] + *parte dinamica* [informazioni sullo stato di esecuzione del programma])
- Il **gestore dei processi** controlla la terminazione, interruzione e riattivazione, sincronizzazione dei processi

Terminazione di un processo

- Un processo termina :
  - Quando esegue una istruzione macchina di terminazione
  - Quando effettua una operazione illecita (es. cerca di accedere a memoria privata di altri processi)
  - Quando c'è un errore che non gli permette di proseguire (es. overflow, etc)
- In tutti questi casi il processore ricomincia automaticamente ad eseguire il sistema operativo ad un indirizzo prefissato

Interruzione di un processo

- Il sistema operativo può bloccare un processo in un qualsiasi istante della sua esecuzione per effettuare qualche operazione di gestione della macchina
- Questo avviene attraverso il meccanismo hardware degli **interrupt**

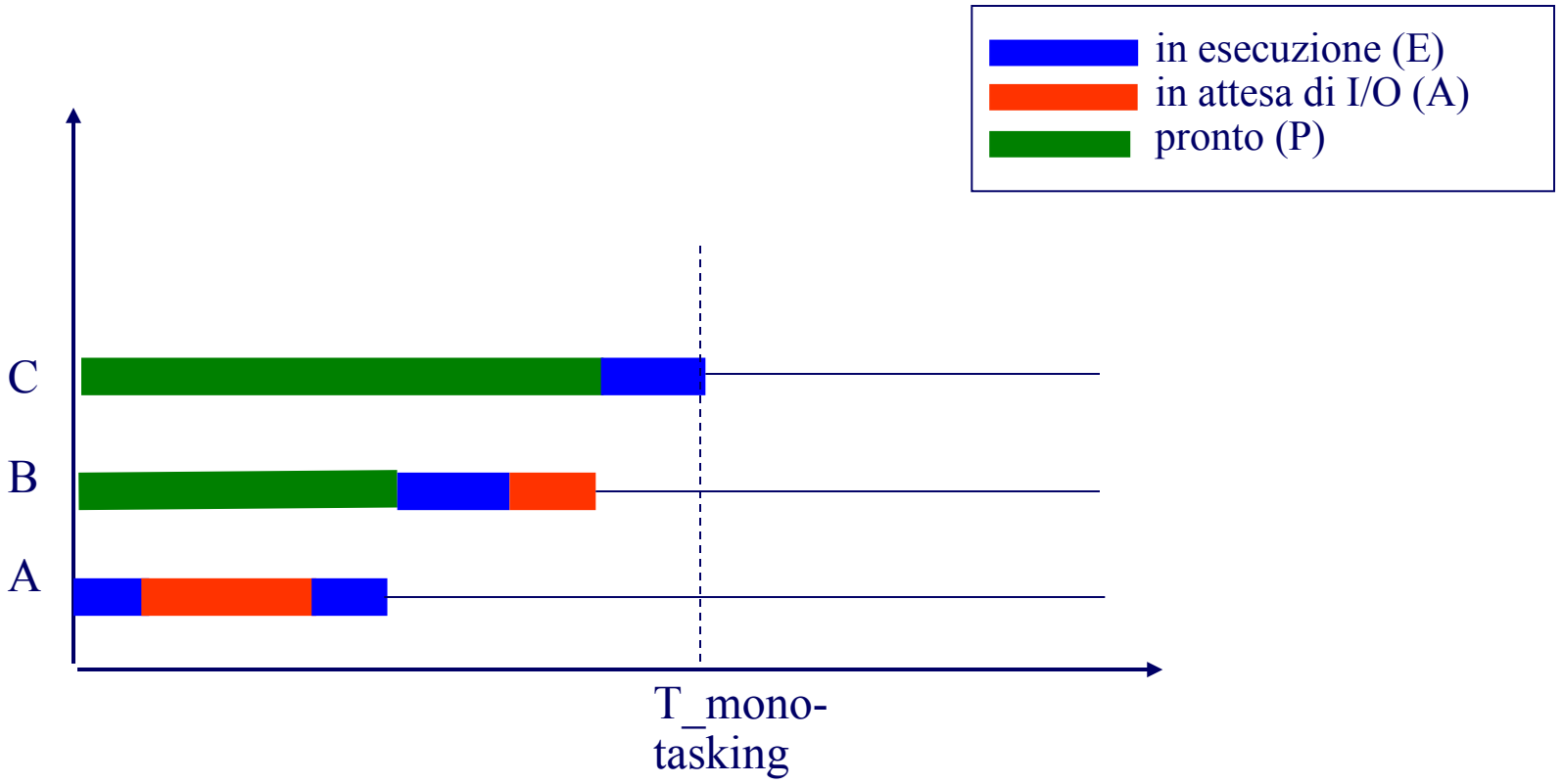
Interruzione di un processo 2

- Come funzionano gli interrupt:
  - ogni periferica può ‘richiedere attenzione’ inviando un **segnale di interrupt** usando le linee di controllo del bus
  - alla fine dell’esecuzione di ogni istruzione macchina il processore controlla la presenza di una interruzione
  - se è presente, il controllo passa automaticamente al sistema operativo, e precisamente alla parte chiamata **gestore delle interruzioni**

Sistemi monotasking

- I SO che gestiscono l'esecuzione di un solo processo per volta si chiamano *monotasking*.
- Non è possibile sospendere l'esecuzione di un processo per assegnare la CPU a un altro
- Sono storicamente i primi SO (MS-DOS)

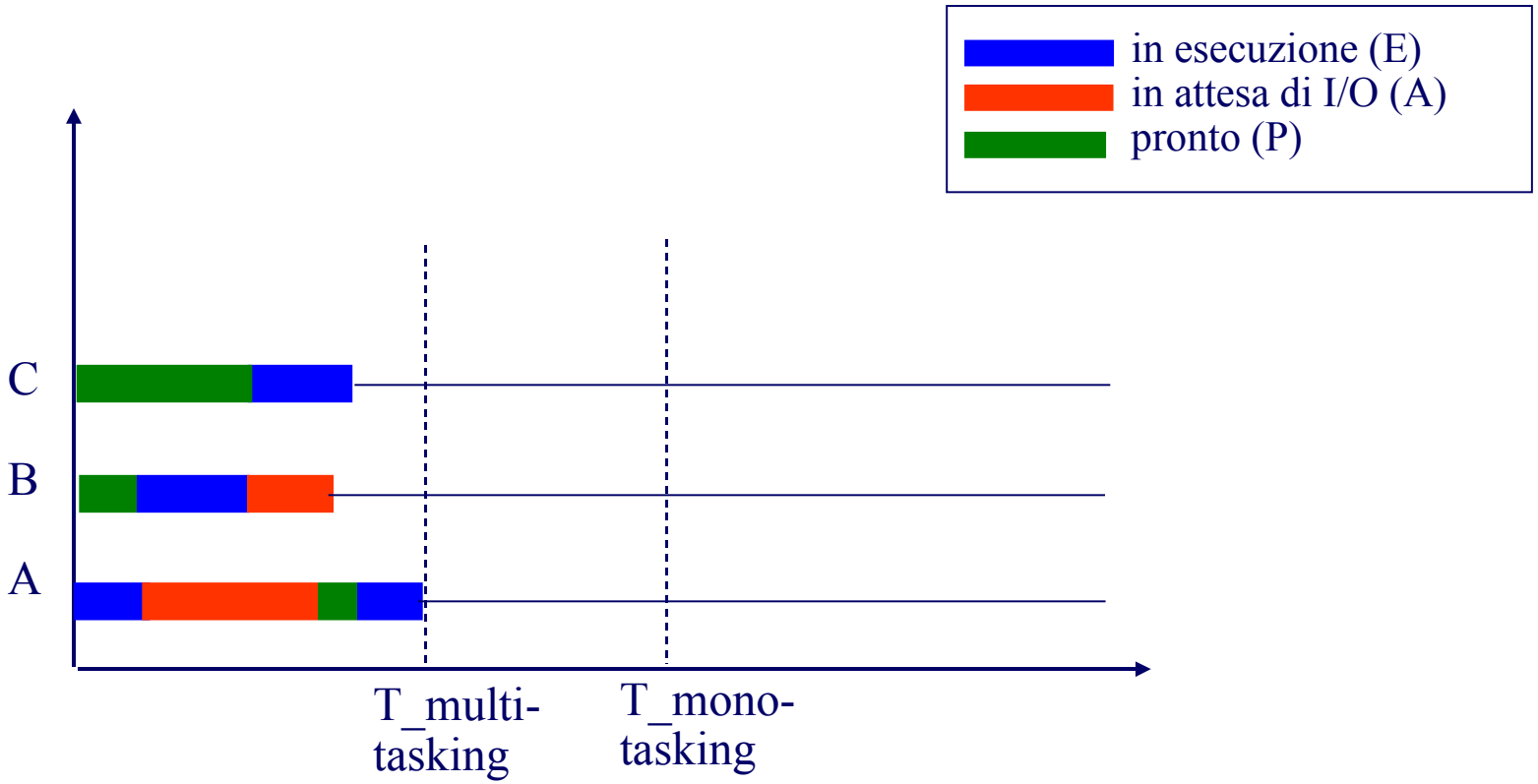
Sistemi monotasking: diagramma temporale



Sistemi multitasking

- I SO che permettono l'esecuzione contemporanea di più processi si chiamano *multitasking* (Windows-XP, Linux, Mac OS)
- Un processo può essere interrotto e la CPU passata a un altro processo

Sistemi multitasking: diagramma temporale

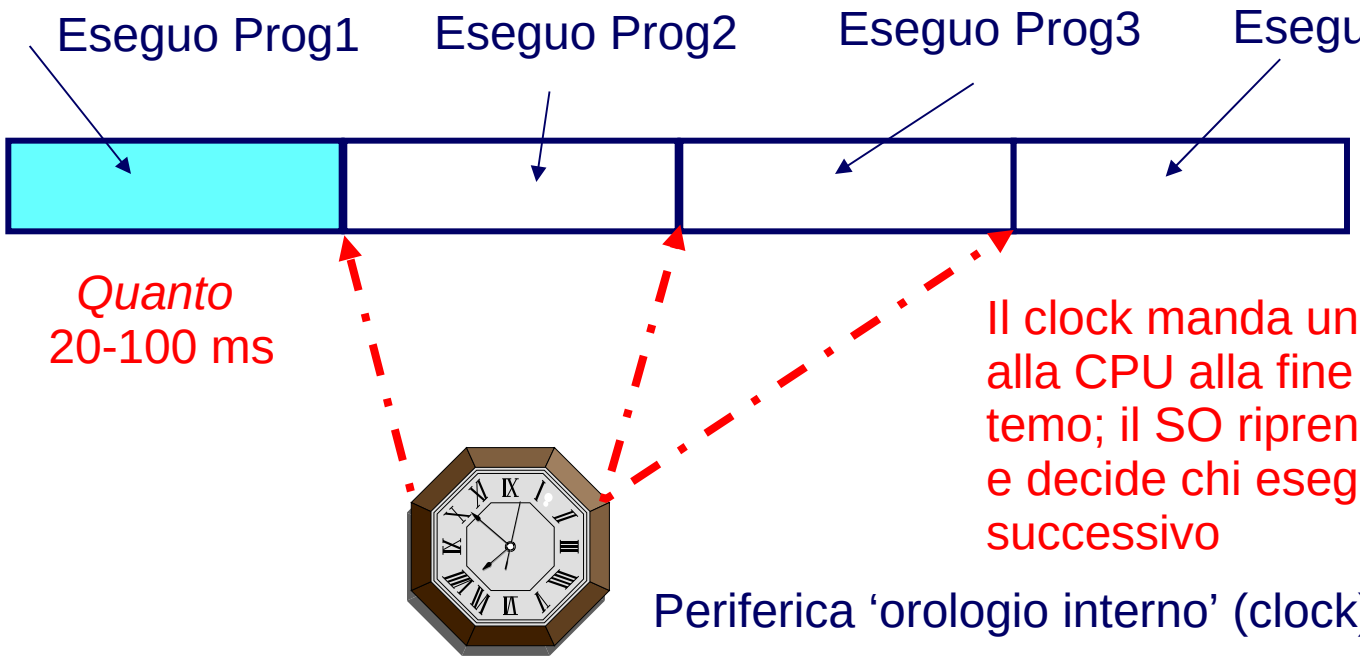


Time-sharing

- È il meccanismo che permette di far condividere il processore a tutti i programmi attivi in modo corretto
- Ogni programma ha l'impressione di avere un suo processore 'dedicato'
- Per evitare che un processo resti troppo a lungo in attesa, le risorse sono ripartite tra i processi
  - i processi vengono accodati e ciascuno ha a disposizione la CPU per un quanto di tempo (**time-slice**)
  - quando il quanto finisce il processo viene messo in fondo alla coda e viene messo in esecuzione il prossimo, cioè il primo della coda
- Due tempi di esecuzione: **elapsed time** (tempo trascorso dall'inizio dell'esecuzione del processo, compreso quello passato nella coda) e **CPU time** (tempo di uso effettivo della CPU)

Time-sharing 2

- Es: 3 programmi attivi Prog1, Prog2, Prog3
- vengono mandati in esecuzione ciclicamente



Sistemi real-time

- Nelle applicazioni real-time il tempo di completamento dell'esecuzione è un parametro critico (es. controllo rotta di un aereo, applicazioni multimediali). Di conseguenza il time sharing viene realizzato in modo diverso.
- I sistemi real-time garantiscono che i requisiti temporali vengano soddisfatti.
- Es.:
  - Ad ogni istante si esegue il processo più critico
  - I processi non vengono interrotti finché non arriva uno con priorità più alta
  - La priorità è fissa (diversamente da time-sharing)

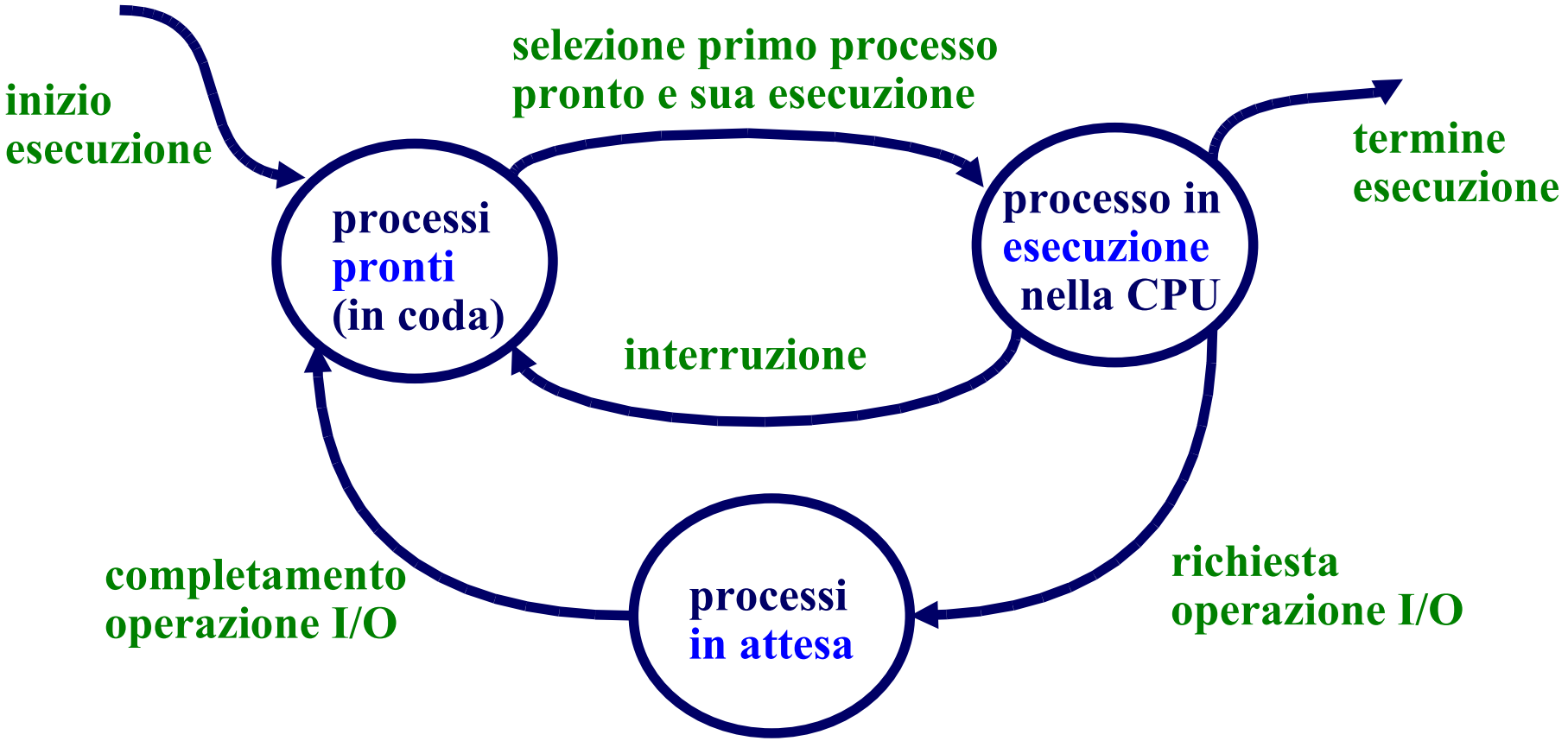
Tabella dei processi

- Ogni processo può essere in esecuzione nella CPU, o nella coda dei processi pronti, o in attesa di qualcosa. Per poter passare da uno stato all'altro le informazioni più importanti del processo devono essere salvate
- La **tabella dei processi** risiede in memoria centrale e contiene, per ogni processo, tutte le informazioni necessarie per gestirne i passaggi di stato.
  - ID processo
  - PC (Program Counter: indirizzo della prossima istruzione)
  - Registri
  - Stato (in attesa, pronto, in esecuzione)
  - Informazioni gestione memoria (memoria allocata al processo)
  - Informazioni scheduling (priorità)
  - Informazioni I/O (dispositivi allocati, etc.)

Tabella dei processi 2

- La tabella dei processi viene usata per varie operazioni, es.:
  - creazione processo
  - cambio priorità
  - liberazione risorse (se il processo è bloccato)
  - terminazione di un processo
- **Context-switching** (o **commutazione di contesto**): è l'operazione di salvataggio dei registri e dello stato del processo che si è terminato di eseguire nella CPU, e il caricamento dello stato del processo che si comincia ad eseguire nella CPU.

Ciclo di vita di un processo [Diagramma Stati-Transizioni]



I cerchi sono gli **stati**, le frecce sono le **transizioni** che rappresentano gli eventi che causano il passaggio da uno stato a un altro stato.

Transizioni di stato di un processo

- **pronto → esecuzione**:
  - il SO stabilisce quale dei processi pronti debba essere mandato in esecuzione
  - la scelta è fatta in base a un algoritmo di ***scheduling***: per esempio, si esegue il primo della coda dei processi pronti, oppure quello con priorità massima, ...
- **esecuzione → attesa**:
  - il processo chiede delle risorse di I/O o attende un evento
  - il SO salva tutte le informazioni necessarie a riprendere l'esecuzione e l'informazione relativa all'evento atteso nella tabella dei processi

Transizioni di stato di un processo

- **attesa → pronto**:
  - si verifica l'evento atteso dal processo e il SO sposta quel processo nella coda dei processi pronti

- **esecuzione → pronto**:
  - il processo in esecuzione viene interrotto dal SO (es. perché termina il quanto di tempo a disposizione, oppure perché la CPU riceve un interrupt) e lascia spazio a un altro processo pronto (o al gestore delle eccezioni)
  - il SO salva tutte le informazioni necessarie a riprendere l'esecuzione del processo dal punto in cui viene interrotta nella tabella dei processi

Esercizio

- Consideriamo tre processi A, B e C
- Supponiamo:
  - il quanto di tempo t sia 40 msec
  - ad A servono 2 quanti per essere completato e a metà del primo quanto A abbia bisogno di un input (es. attesa di richiesta stampa per 30 msec)
  - a B servono 2 quanti di CPU per essere completato e a metà del secondo si metta in attesa di una risorsa che non è disponibile
  - a C serve un quanto
- Scrivere il diagramma temporale dell'esecuzione di A, B e C, assumendo che nello stato iniziale siano tutti e tre nella coda dei processi pronti, nell'ordine A, B, C.
- Mostrare le transizioni di stato di tutto il sistema

Esecuzione in stato utente e in stato supervisore

- A seguito di una system call o di un interrupt, il SO può interrompere l'esecuzione di un processo in stato utente (es. editor testo, programma di email) e mandare in esecuzione altri programmi o effettuare operazioni di 'gestione' della macchina

- Al termine della gestione dell'interrupt o dell'esecuzione di processo del SO (in **stato supervisore**), l'uso della CPU può passare di nuovo all'esecuzione di un processo in **stato utente**

- NB. Un processo in stato utente non può mai passare allo stato supervisore (in stato supervisore si ha accesso a tutte le risorse)!

Esercizio

- Considerando l'esercizio precedente, spiegare quando l'esecuzione di un programma in modalità utente viene interrotta e il controllo della CPU passa al SO e quando poi riprende l'esecuzione in modalità utente.