Interruzioni e S.O. (1)

- E' compito del S.O. gestire le interruzioni tramite le *interrupt routines*.

- Nel semplice modello di S.O. visto, *in cui un solo programma alla volta* è presente in memoria principale, la sovrapposizione tra le attività di I/O e di elaborazione produce uno *scarso beneficio*.

Interruzioni e S.O. (2)

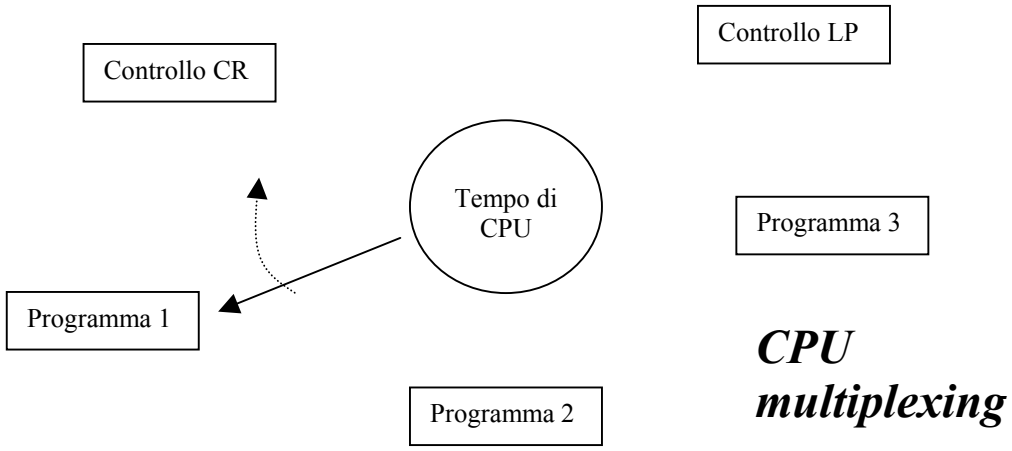
- Entrambe le attività sono relative infatti allo stesso programma, che in genere ammette un grado limitato di parallelismo tra di esse.

- Per avere un guadagno notevole nell'utilizzazione delle risorse occorre avere *più programmi presenti contemporaneamente in memoria* (multiprogrammazione), potendo così sovrapporre elaborazione ed I/O di programmi diversi.

Multiprogrammazione (circa 1965)

- Più programmi sono presenti contemporaneamente in memoria principale.

- Quando uno di essi attende per il completamento di una operazione di I/O, il controllo della CPU può essere assegnato ad un altro:



CPU multiplexing

- Occorre che l'insieme dei programmi presenti contemporaneamente in memoria principale sia scelto (e posizionato!) in modo da garantire la massima occupazione della CPU (e della memoria).

- Programmi *I/O bound* e *Compute bound*

Interruzioni in Multiprogrammazione

- *La routine di interruzione*, invece di ritornare al programma interrotto, può *scegliere*, in base ad un *determinato algoritmo*, a quale programma presente in memoria affidare il controllo della CPU.

- I programmi possono quindi essere attivati come conseguenza di una interruzione.

Interruzioni in Multiprogrammazione

- Il S.O. deve provvedere alla scelta del programma cui assegnare la CPU sulla base di *algoritmi di scheduling*.

- Per evitare che i programmi *compute bound* mantengano il controllo della CPU per tempi molto più elevati di quelli *I/O bound*, è necessario introdurre anche un interrupt periodico (timer) che comunque determina l'invocazione dello scheduler.

Interruzioni in Multiprogrammazione

- Un programma durante il tempo di CPU a lui dedicato può richiedere l'accesso ad un dispositivo già impegnato (lui non lo sa!) (eventualmente nel servizio di una richiesta precedente dello stesso programma):
  - Attenzione a scrivere in modo efficiente un programma: prima di richiedere l'uso di un dispositivo, accertarsi che sia libero; in caso contrario, si può attivare qualche altra attività.

Interruzioni in Multiprogrammazione

- Il S.O. deve pertanto:
  - *mantenere in una tabella lo stato dei dispositivi*,
  - *sospendere* un programma che intende accedere ad una risorsa già impegnata, mettendo la relativa richiesta in una coda associata al dispositivo,
  - *realizzare algoritmi di gestione per tali risorse*, scegliendo tra i programmi sospesi quello a cui attribuire la risorsa,
  - *proteggere i dati di un programma*, ...

Interruzioni (1)

Interruzioni hardware (asincrone):

- *device interrupt*, per terminazione di un trasferimento dati o per condizione di errore rilevata dalle o nelle periferiche (es. parità)
- *timer interrupt*, real-time clock per la misura del tempo
- *powerfail sense interrupt*, per salvataggio urgente dello stato del sistema

Interruzioni (2)

Interruzioni software (sincrone):

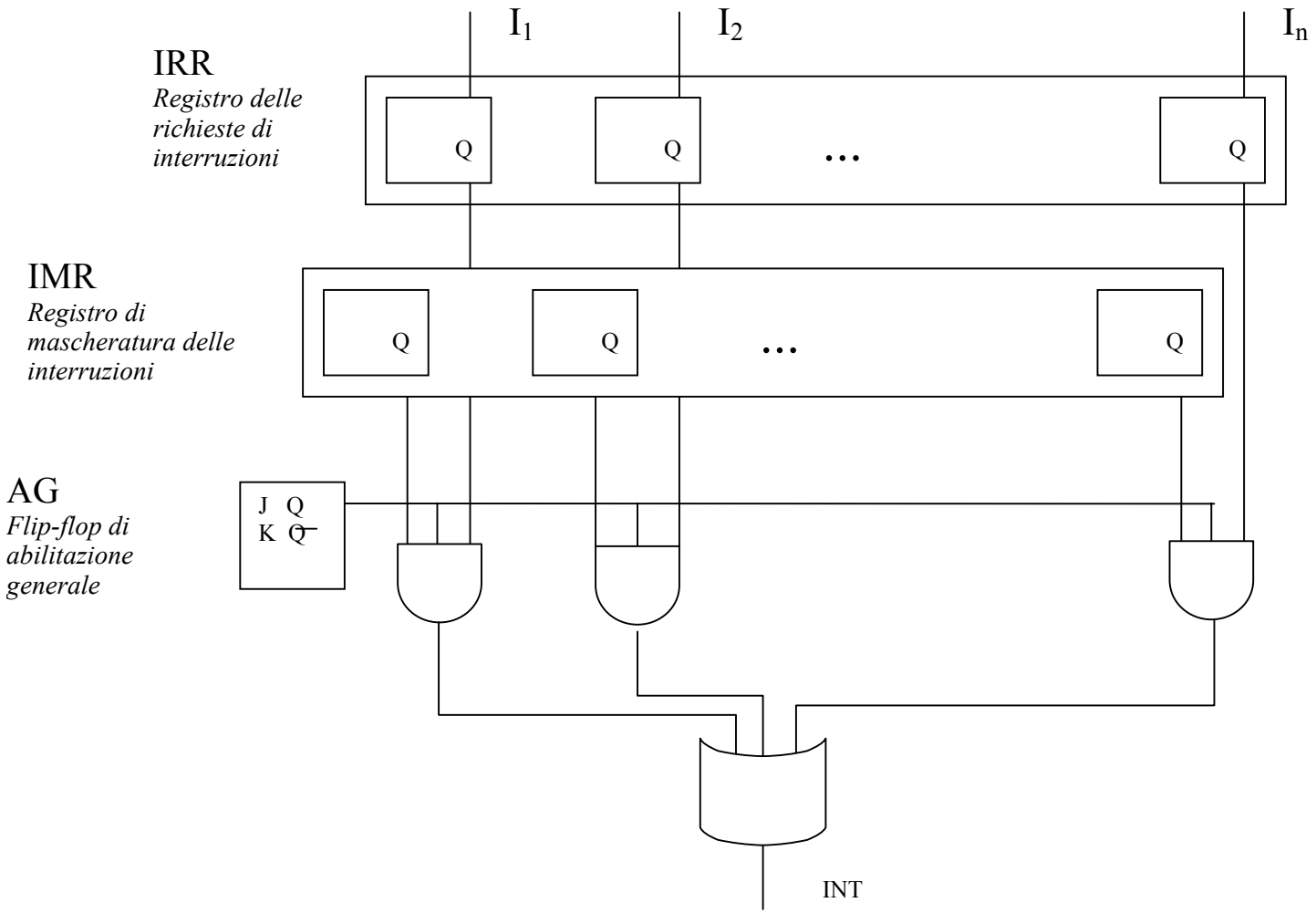
- *eccezioni* o *trap*, per tentativo da parte del programma di compiere azioni con effetti illegali, ad es. divisione per zero, overflow, opcode illegale (inesistente o privilegiato), etc. (rilevazione a livello hardware)
- *programmate* o *supervisory call (svc)*, (es. INT n), per utilizzare le funzioni del S.O. o per trasferire il controllo al S.O.

Sistema delle Interruzioni

- A ciascuna causa di interruzione è associata *un'azione* che verrà gestita dal programma di risposta alle interruzioni.

- Una causa non produce di per sé alcuna interruzione ma solo una *richiesta di interruzione*. Affinché alla richiesta segua effettivamente un'interruzione è necessario che la causa di interruzione sia *abilitata*.
- Abilitazione e disabilitazione possono essere selettive o globali.

Sistema delle Interruzioni



Sistema delle Interruzioni (1)

- Ciascun evento "Ii" causa di interruzione è memorizzato in un flip-flop IRRi e viene abilitato da IMRi ed AG

  - IRR registro di richiesta di interruzione
  - IMR registro di maschera di interruzione
  - AG flip-flop di abilitazione/disabilitazione generale delle interruzioni

Sistema delle Interruzioni (2)

- L'interruzione "i" si manifesta se:
  - AG = 1 (il sistema di interruzione è abilitato)
  - IRRi = 1 (si è presentata la causa di interruzione "Ii")
  - IMRi = 1 (l'interruzione "i" è abilitata nel registro di maschera)

- IMR e AG possono essere modificati mediante istruzioni speciali (IE, DI) o istruzioni generali (out(io_address, value))

Processo delle Interruzioni (1)

- Al verificarsi di un'interruzione occorre:

  a. salvare tutte le informazioni necessarie per la ripresa del programma interrotto
  b. individuare la causa dell'interruzione
  c. eseguire le azioni richieste (servizio dell'interruzione)
  d. ripristinare lo stato del programma interrotto e riavviarlo

Processo delle Interruzioni (2)

- A seconda della sofisticazione dell'hardware queste azioni impegnano in misura maggiore o minore il software (overhead).

  a. L'hw provvede in genere a salvare PC e PSW, mentre i registri generali vengono tipicamente salvati in software. E' necessario che il salvataggio dei registri avvenga ad interrupt disabilitati.
  Normalmente l'hw cede il controllo alla routine di servizio con interrupt disabilitati (AG=0). In caso contrario il sw deve disabilitare gli interrupt.

Processo delle Interruzioni (3)

b. L'hw può trasferire il controllo sempre al medesimo programma di risposta, che provvederà quindi ad individuare la causa dell'interruzione tramite *skip chain*, oppure direttamente a programmi di risposta separati.

c. Durante il servizio della interruzione il sistema delle interruzioni può essere riabilitato (AG), eventualmente selettivamente (IMR), purchè sia possibile il *nesting* delle interruzioni tramite *stack*.

Processo delle Interruzioni (4)

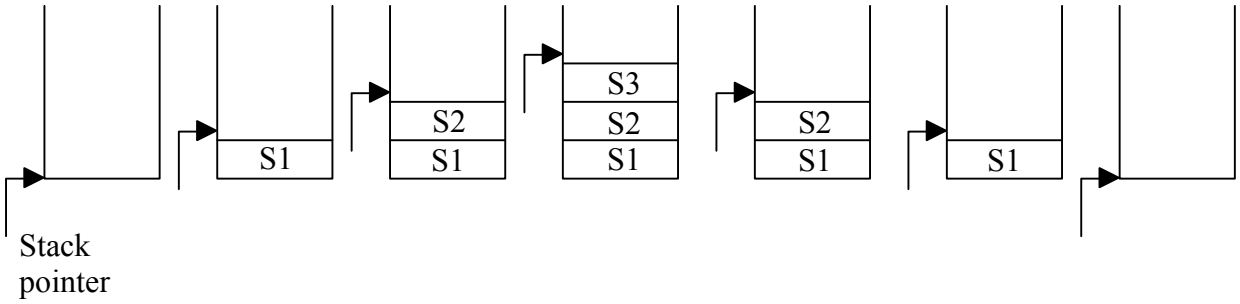
d. Il ripristino dei registri deve avvenire ad interrupt disabilitati. Un'apposita istruzione di ritorno dall'interrupt (RTI) ripristina i registri salvati via hw e riabilita le interruzioni.

Livelli di priorità

- I diversi tipi di interruzione possono suggerire una gestione con diversi livelli di priorità.

- Durante il servizio di un interrupt di livello L le interruzioni di livello K<= L restano disabilitate, mentre le altre possono essere abilitate. Questa gestione può essere realizzata o agevolata dall'hw (programmable interrupt controller) o per via sw.
- Se all'atto del ritorno esistono più richieste pendenti si serve quella di livello più elevato.

Salvataggio dello stato tramite pila



S1: stato del processore salvato all'arrivo della prima interruzione

S2: stato del processore relativo all'esecuzione del programma di risposta della prima interruzione salvato all'arrivo della seconda interruzione

S3: stato del processore relativo all'esecuzione del programma di risposta della seconda interruzione salvato all'arrivo della terza interruzione

- Salvataggio, ripristino, gestione SP non devono essere interrompibili (hw o interrupt disabilitati)

- Programma di risposta ad interruzione di livello L:
  - disabilita tutte le interruzioni
  - salva lo stato del processore nello stack e modifica SP
  - abilita solo interruzioni di livello K>L
  - esegue programma di risposta
  - disabilita le interruzioni di tutti i livelli
  - ripristina lo stato del processore modificando SP
  - esegue il ritorno da interrupt

Processi (1)

- L'evoluzione dei S.O., guidata da esigenze di efficienza, ha portato alla presenza in memoria centrale di *più programmi* in esecuzione.

- Emergono nuove *funzionalità* richieste al S.O., quali la gestione dei programmi stessi e la protezione dalla mutua interferenza.

Processi (2)

Un nuovo *punto di vista concettuale* :

- I programmi di controllo, analogamente ai dispositivi hw, sono largamente indipendenti l'uno dall'altro, ed interagiscono con altre attività di rado ed in punti ben definiti;

- La CPU viene "trasferita" da un job all'altro (anzichè "ricevere" programmi in ingresso);

Processi (3)

- La specifica sequenza di stati della CPU è scarsamente significativa ed ***impredicibile*** a causa degli interrupt;

- il sistema va concepito in termini di *specifica funzionale:*
  - delle elaborazioni,
  - del controllo dei dispositivi,
  - delle strutture dati per lo scambio di informazioni.

Processi (4)

- Un *processo* (o task) è l'unità funzionale in un S.O.

> Un processo è *controllato da un programma* ed ha bisogno di un *processore per la sua esecuzione*.

- Alcuni processi dispongono di un processore *privato* e pertanto sono permanentemente in esecuzione (ad esempio i controllori delle periferiche).
- Altri processi condividono un processore *comune* (la CPU) (ad esempio i processi utente e di sistema).

Gestione dei processi (1)

- Processo => programma in esecuzione

  (es. job batch, programma utente time-shared, task di sistema, etc.)

- Il processo è l'unità di lavoro di un sistema (che consiste di una collezione di processi):
  - *processi del S.O.* che eseguono il codice del sistema e
  - *processi utenti* che eseguono il codice utente.
- Possono essere in esecuzione *concorrentemente*.

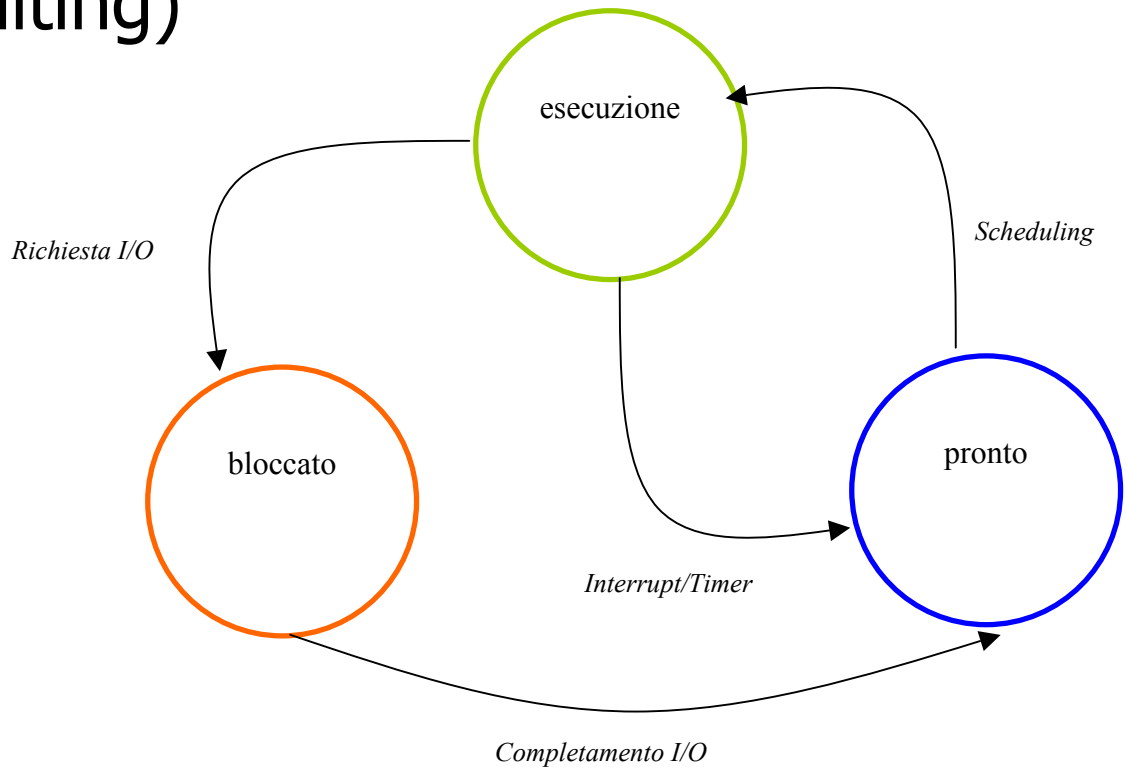
Gestione dei processi (2)

Funzioni del S.O. (riferite ai processi):

- creazione e cancellazione di processi
- sospensione e ripresa di processi
- strumenti per la sincronizzazione e comunicazione
- strumenti per il trattamento di condizioni di deadlock

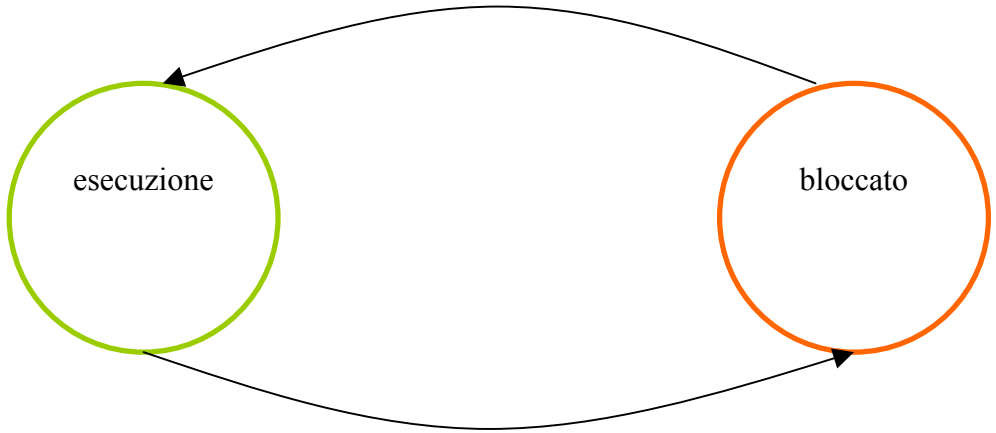
Stato di un processo (1)

- in esecuzione (running)
- bloccato (idle, waiting)
- pronto (ready)

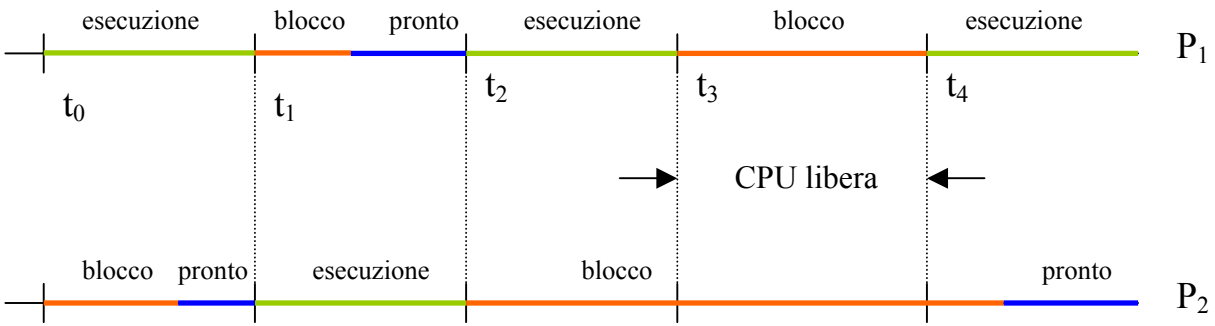


Stato di un processo (2)

Se il numero di CPU è uguale al numero dei processi:



Sistemi multiprogrammati



- Una possibile definizione alternativa di processo: "Attività svolta dalla CPU per l'esecuzione di un determinato programma"

Algoritmo, Programma, Processo

Algoritmo: Procedimento logico che deve essere seguito per risolvere il problema in esame

Programma: Descrizione dell'algoritmo tramite un opportuno formalismo (*linguaggio di programmazione*) che rende possibile l'esecuzione dell'algortimo da parte di un particolare elaboratore

Processo (sequenziale): La sequenza di eventi che genera un elaboratore quando opera sotto il controllo di un particolare programma (evento = esecuzione di una operazione)