Architettura degli elaboratori

Interruzioni

INTERRUZIONI

Generalità

- Una **interruzione** (*interrupt*) è un evento che cambia la normale sequenza di un programma in esecuzione
- Possibili cause che generano una interruzione:
  - **Malfunzionamento hardware**: Errore di parità in memoria, abbassamento della tensione di alimentazione, carta inceppata di una stampante
  - **I/O**: interazione con periferiche (tastiera, mouse, stampante,…)
  - **Programma**: overflow, divisione per zero, istruzione non riconosciuta, accesso ad un indirizzo errato, richiesta di interazione con file o dispositivi HW
  - **Timer**: interruzione periodica (es.: ogni 10ms) per analizzare il tempo speso da una applicazione e per cedere eventualmente il processore ad un'altra applicazione (*multiprogrammazione*)

INTERRUZIONI

Generalità

- Le **interruzioni** sono concepite per migliorare l'efficienza dell'elaborazione
  - Permettono di liberare il processore da compiti gravosi di sincronizzazione
  - Sono utili soprattutto per gestire le operazioni realizzate da componenti che hanno tempi di risposta superiori a quelli del processore (es. dispositivi di Input ed Output)
  - Concorrono a gestire o interrompere l'elaborazione in seguito di errori di calcolo

INTERRUZIONI

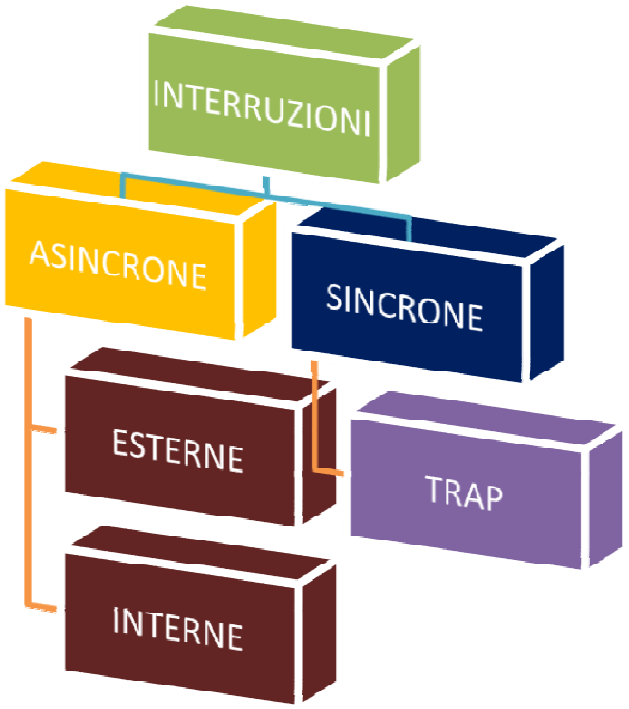
Classificazione

- Interruzioni possono essere classificate come:
  - **Interruzione hardware:** il segnale di interruzione è scaturito da un componente dell'elaboratore elettronico
    - **Interna**: interna alla CPU
    - **Esterna**: dovuta ad una periferica (per comunicare o evidenziare anomalie)
  - **Interruzione software :** il segnale di interruzione è scaturito da un programma
    - **Trap:** interruzione richiesta dal programmatore
    - **Eccezioni**: interruzione determinata da un programma (es.: una divisione per zero)

INTERRUZIONI

Classificazione

- Le interruzioni possono essere classificate anche in base alla relazione con il *clock* della macchina:
  - **Asincrone** (il loro accadimento non è noto, sono indipendenti dal clock)
    - Interruzione generata da un dispositivo di I/O; es.: inceppamento carta
    - Interruzione generate da un programma in esecuzione; es.: apertura di un file non esistente, divisione per zero,…
  - **Sincrone** (il loro accadimento è noto, avvengono in una precisa fase del clock)
    - **Trap**



INTERRUZIONI

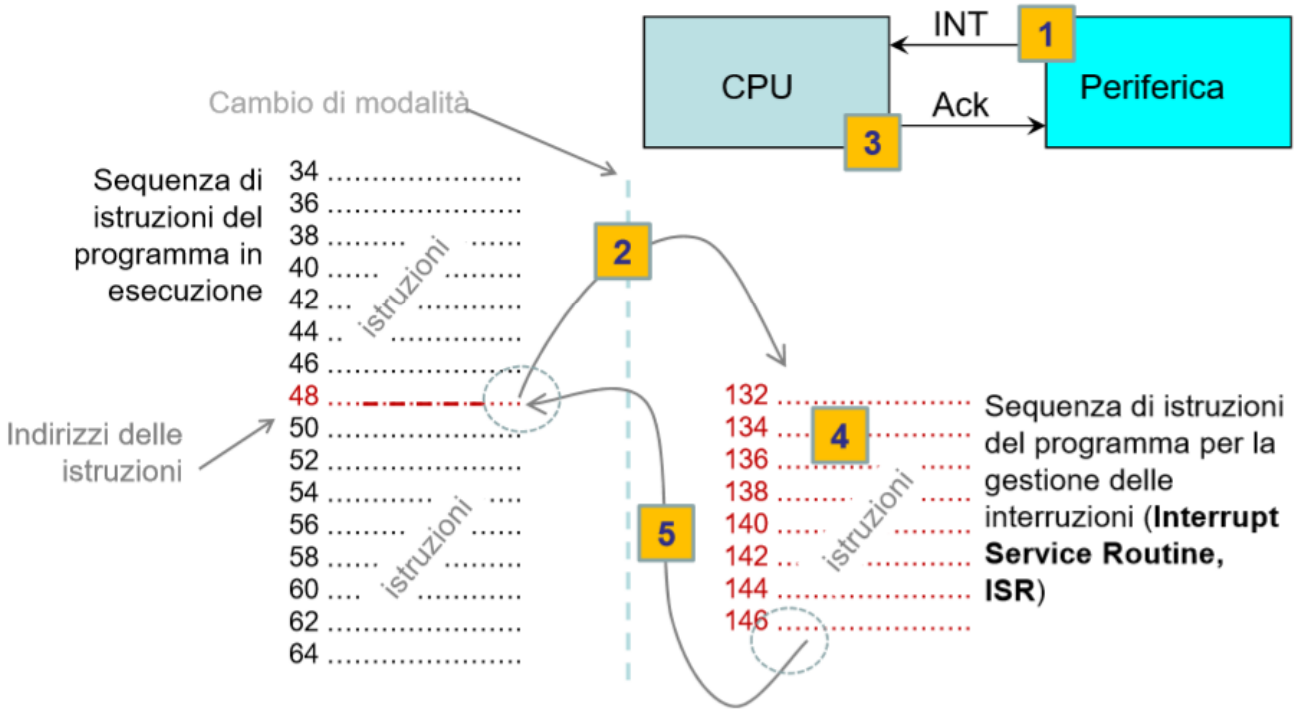
Sviluppo

- Un *interrupt* interrompe il programma in esecuzione e trasferisce il controllo a un **gestore di interruzione** deputato a svolgere le azioni appropriate
- Al loro compimento, il gestore di interruzione restituisce il controllo al programma interrotto. È suo compito far riprendere il processo interrotto esattamente dallo stesso stato e posizione in cui si trovava al momento dell'interruzione, il che implica il ripristino di tutti i registri interni allo stato precedente all'interruzione

INTERRUZIONI

Sviluppo

- Sequenza di attivazione:
  1. la Periferica alza il segnale (**INT**)
  2. il Processore interrompe il programma in esecuzione
  3. la Periferica viene informata che l'interrupt è stato ricevuto (**Ack**)
  4. viene eseguita la procedura (**ISR**)
  5. si ripristina il Programma precedentemente interrotto



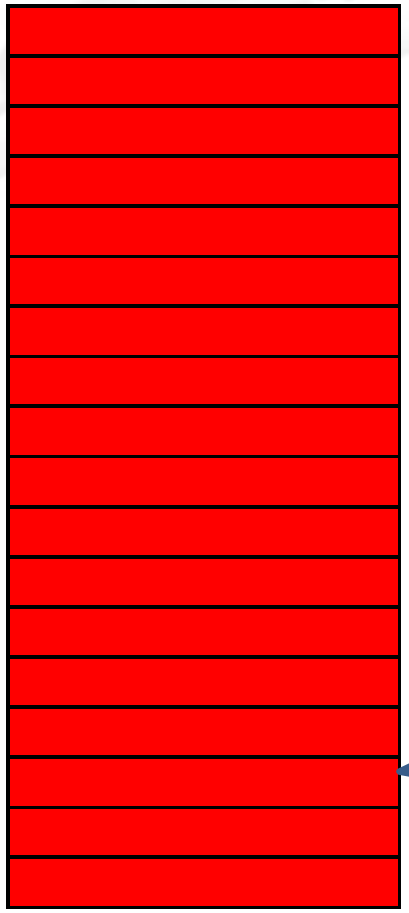
INTERRUZIONI

Gestore di interruzione

- Un **sistema di interruzione** deve:
  - garantire che non vi siano interferenze indesiderate sul programma che è interrotto tramite il salvataggio ed il ripristino delle informazioni e dello stato esistente nel momento in cui si verifica una interruzione (**commutazione del contesto**)
  - identificare il dispositivo che ha generato l'interruzione al fine di selezionare ed attivare la routine ad esso associata (**riconoscimento delle interruzioni**)
  - risolvere le diverse interruzioni generate da dispositivi diversi tramite la definizione di un ordine di gestione (**gerarchia di priorità**)

INTERRUZIONI

Attivazione di una interruzione



Esecuzione Programma

Interruzione (asincorna)

Commutazione del contesto

Riconoscimento Interruzione

Espletamento interruzione

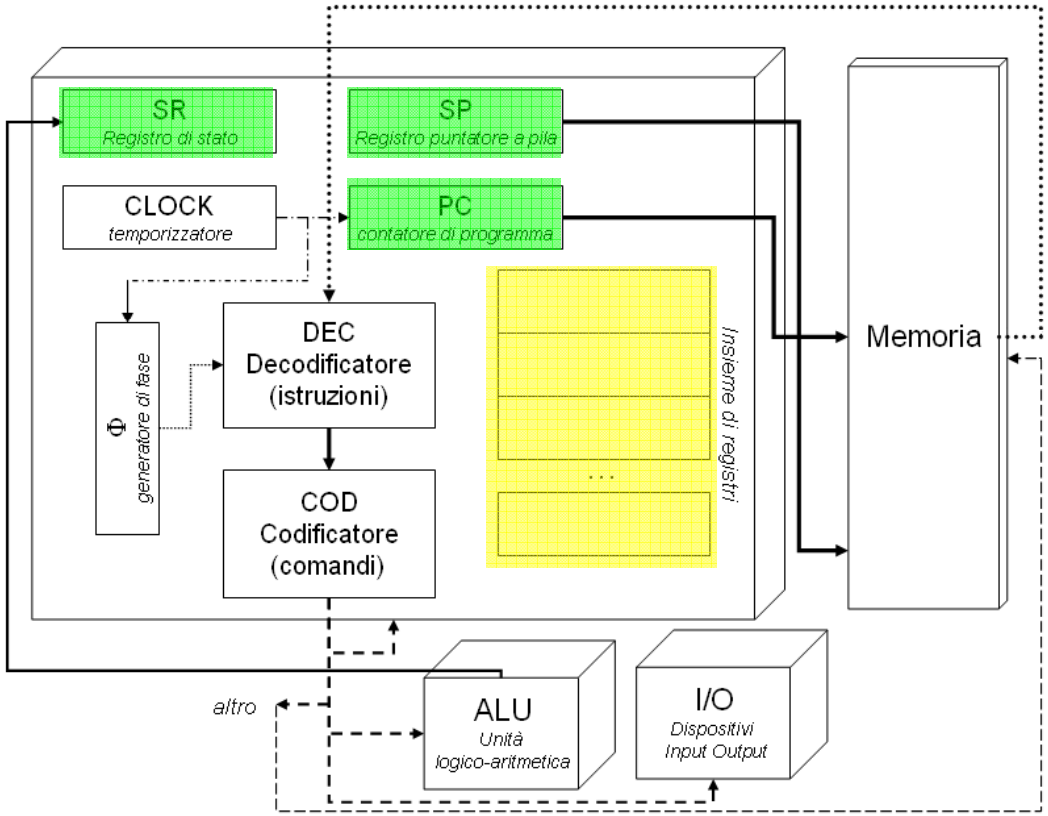
Ripristino del contesto

Ripresa del programma

INTERRUZIONI

Commutazione del contesto

- Il contesto su cui un programma opera è costituito dalle informazioni presenti nei seguenti registri (lo **stato volatile della macchina**):
  - il **Program Counter**, che contiene l'indirizzo della istruzione da cui dovrà essere ripresa l'esecuzione del programma interrotto
  - i **bit di condizione** (implicitamente conservati nel Registro di Stato) per ritornare ad una situazione coerente con le ultime istruzioni elaborate prima del salto (infatti i CC potrebbero essere modificati dalla routine di servizio)
- Ed inoltre
  - i **registri del modulo ALU,** che possono contenere dati che il programma interrotto non ha terminato di elaborare
  - I **registri ad uso generale** della CPU



INTERRUZIONI

Commutazione del contesto

- Il **salvataggio del contesto** può essere effettuato solo quando si è raggiunta una situazione ben definita e ricostruibile (esempio: prima che inizi la fase di *fetch* della istruzione successiva)
- L'operazione di salvataggio può essere intesa come una **commutazione** dal contesto del programma che è interrotto e il passaggio alla routine di servizio. Analogamente il ripristino è ancora una commutazione: dalla routine di servizio al programma che deve proseguire
- Pertanto è opportuno che non si verifichino altre interruzioni mentre sono in corso le operazioni di commutazione del contesto (**fase protetta**), altrimenti ci potrebbero essere anomalie che comprometterebbero il corretto funzionamento dell'intero sistema di calcolo

INTERRUZIONI

Commutazione del contesto

- Per far questo si può dotare il processore di un **campo di interruzione** che consenta di definire se può essere interrotto o no, affinché non sia possibile interferire con le operazioni di commutazione
- Il campo può essere conservato nel Registro di Stato e può essere di un solo bit, identificato dalla lettera **I** (*Interrupt flag*), o di più bit nel caso si voglia definire diversi gradi di interruzione (più interruzioni con priorità diverse)
  - Di solito con I=1 si accettano interruzioni con I=0 si rifiutano

Registro di stato

C	W	N	Z	P	I		

Registro di stato

C	W	N	Z	P	I	I	I

INTERRUZIONI

Metodi per il riconoscimento delle interruzioni

- L'uso di un sistema di interrupt implica, pertanto, che ci siano in memoria centrale tante **routine di servizio** (***interrupt service Routine***, **ISR**) quanti sono i dispositivi collegati al calcolatore che sono in grado di generare richieste di interruzione
- Questa moltitudine di routine prevede che vi sia dapprima la corretta **identificazione del dispositivo** e successivamente l'esecuzione della relativa ISR
- L'individuazione può avvenire:
  - tramite una sequenza di istruzioni atte all'individuazione del dispositivo che faccia da preambolo comune a tutte le routine e pertanto eseguita all'inizio di ogni interruzione (**polling**)
  - prevedendo che il dispositivo, oltre a generare la richiesta di interruzione invii una informazione più completa come ad esempio un codice di identificazione univocamente associato al dispositivo chiamante (**interruzione vettorizzata**)

INTERRUZIONI

Gerarchia delle interruzioni

- In alcuni sistemi, l'accadimento di una precisa interruzione deve essere gestita entro un **tempo massimo**. Per questo motivo si deve decidere un grado di priorità alle diverse interruzioni

  Si può pensare ad un sistema per il rilevamento termico del nocciolo di una centrale nucleare, che ogni Δt campiona la temperatura: il prelievo del dato prodotto deve avere luogo entro l'intervallo temporale fissato, altrimenti il dato non è prelevato o è distrutto dal sopraggiungere del successivo (e in un contesto così pericoloso sono due condizioni da scongiurare!!!*).* Altresì per dispositivi come i terminali video o di stampa, generalmente, la visualizzazione o la riproduzione può essere rimandata senza gravi problemi

INTERRUZIONI

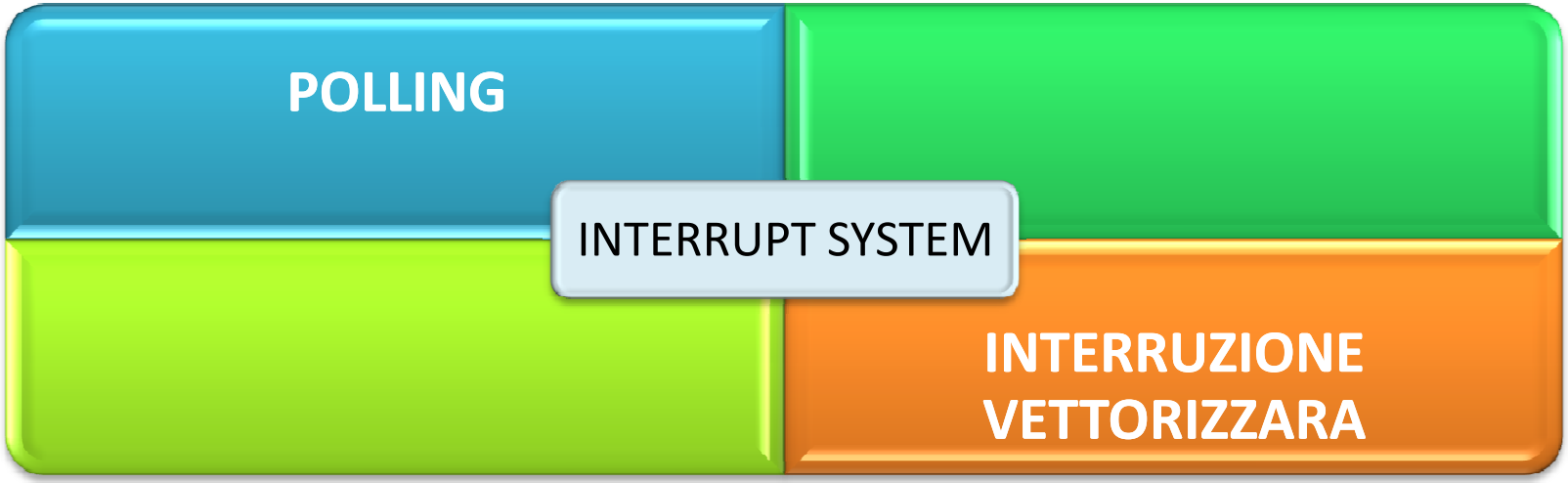
Gerarchia delle interruzioni

- Per questo è opportuno che un sistema di interruzione sia in grado di assicurare che le richiesta di interruzioni provenienti dai dispositivi con esigenze di **tempo reale** (o *real time*) più critiche siano gestite con priorità superiore rispetto alle altre
- La priorità deve essere considerata:
  - quando sono presenti contemporaneamente più interruzioni
  - quando si presenta una interruzione mentre è in corso il servizio di un'altra
- Nel primo caso è necessario che ci sia una discriminante per consentire un ordinamento che porti ad individuare il dispositivo che ha esigenze prioritarie e lasci pendenti gli altri. Nel secondo caso la priorità consente di stabilire se l'interruzione in atto debba essere sospesa a vantaggio di una nuova interruzione, oppure no

INTERRUZIONI

Progettazione di sistemi di interruzioni

- Al fine di comprendere meglio i sistemi di interruzioni saranno presentati due possibili progettazioni con i principali componenti hardware e le relative routine software



INTERRUZIONI

Polling

- Nel **polling** (o **I/O controllato da programma**)
  - la CPU accede cadenzalmente ai registri di stato delle periferiche fino a quando una periferica è pronta a colloquiare
  - Questa strategia impedisce alla CPU di fare altro durante l'attesa: la CPU si blocca in quello che si chiama *busy-waiting*

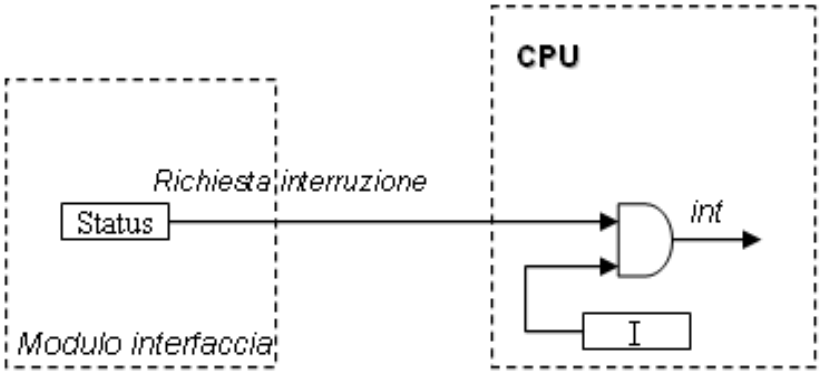
Dopo ogni istruzione il polling esegue questa subroutine:

```
IN A,(n)    #per trasferire sull'Accumulatore il contenuto di un  flag di stato
            # n è l'indirizzo del registro di stato della interfaccia di I/O
BIT b,A     #per verificare  il contenuto di un flag di stato
            #b è numero d'ordine del bit delll' Accumulatore che contiene il flag di stato
JP NZ,nn    #per  saltare alla subroutine di servizio con indirizzo nn
```

INTERRUZIONI

Polling

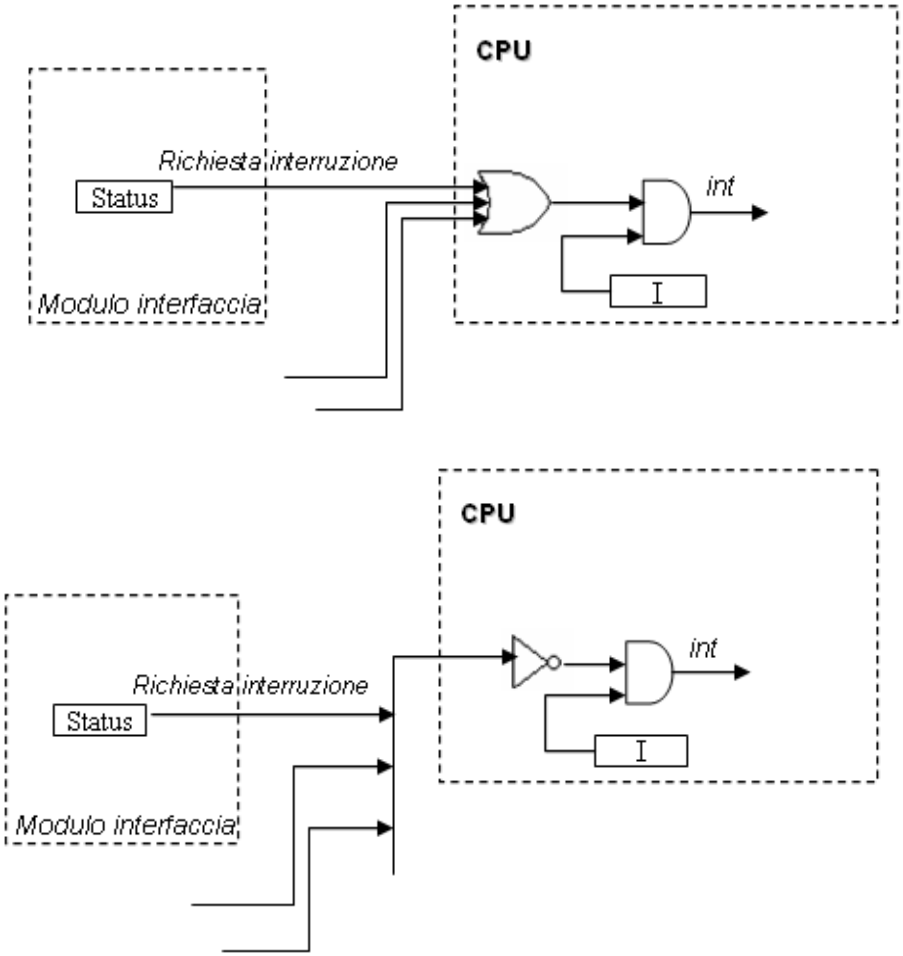
- Nel polling è necessario fornire nel modulo di interfaccia dei dispositivi, un **bit di richiesta di interruzione conservato in un registro di stato** del dispositivo che segnala la richiesta di interruzione
- Ovviamente questa linea è legata anche al bit di interrompibilità presente nella CPU (tipicamente nel registro di stato) per abilitare (1) o non abilitare (0) l'interruzione
- Il processore deve analizzare il segnale uscente da ***int*** e vedere se è 1, in questo caso – quando possibile- sarà accettata una interruzione



INTERRUZIONI

Polling

- In una architettura è logico pensare che siano collegati più dispositivi e quindi si può pensare ad una circuiteria realizzata con un **OR logico** o, più comunemente, ad un collegamento ***wired OR*** di porte *open collector* che consenta di usare un'unica linea per far pervenire alla CPU le diverse richieste



INTERRUZIONI

Polling: commutazione del contesto

- La **commutazione del contesto può avvenire in maniera più o meno radicale** (salvando tutte le informazioni contenute nei registri e nella ALU - codice semplice e compatto ma efficienza bassa; oppure solo le informazioni che effettivamente sono alterate nel corso del servizio dell'interruzione - codice complesso ma efficienza alta)
- La procedura prevede il settaggio a 0 del flag I per impedire che ci sia una altra interruzione
- Per avere un hardware molto semplice è possibile pensare che le uniche operazioni interessate alla commutazione del contesto siano quello di salvataggio del contenuto dello **PC** e dello **SR** nello stack
  - In generale si salva tutto lo stato volatile della macchina grazie al gestore (o *handler*)

Salvataggio

```
0 ←I          #clear del flag I per impedire altre interruzioni
SR ←Push      #salvataggio dello SR nello stack
PC ←Push      #salvataggio del PC nello stack
```

Ripristino

```
PC ←Pop       #ripristino valore del PC
SR ←Pop       #ripristino valore del SR
1 ←I          #set del flag I per impedire altre interruzioni
```

INTERRUZIONI

Polling: commutazione del contesto

- Infine è necessario inserire nel PC l'indirizzo della prima istruzione di routine di servizio di interruzione
- Se l'hardware consentisse una identificazione del dispositivo si potrebbe passare direttamente l'indirizzo della routine di gestione dell'interruzione associata, ma in questo caso non è possibile e pertanto si punta ad una routine di preambolo comune a tutte le routine atta al riconoscimento del dispositivi (*nell'esempio a destra la routine di preambolo risiede nell'indirizzo riservato* 0x1000)

Salvataggio

```
0 ←I          #clear del flag I per impedire altre interruzioni
SR ←Push      #salvataggio dello SR nello stack
PC ←Push      #salvataggio del PC nello stack
0x1000 ←PC    #"forzatura" del PC al valore della routine di
              #preambolo della sequenza di polling
```

Ripristino

```
SR ←Pop       #ripristino valore del SR
PC ←Pop       #ripristino valore del PC
1 ←I          #set del flag I per ccogliere altre interruzioni
```

INTERRUZIONI

Polling: identificazione dell'interruzione

- A questo punto il riconoscimento dell'interruzione può avvenire tramite il **polling**: interrogando ad uno ad uno gli *n* dispositivi fino a che non si trova il richiedente del servizio
- La sequenza di istruzione del polling prevede un salto a subroutine agli indirizzi $SERV_n$ delle corrispondenti routine di servizio del dispositivo chiamante $DISP_n$

Istruzioni di polling

JR DISP1,SERV1
JR DISP2,SERV2
JR DISP3,SERV3
…
JR DISPn,SERVn

JR: jump to subroutine se il bit di stato del DISPn è attivo

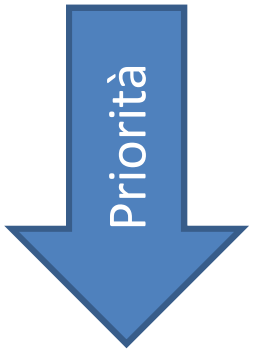
INTERRUZIONI

Polling: Gerarchia di priorità

- Per quanto riguarda la gerarchia di priorità si può pensare che l'ordine con cui i dispositivi sono analizzati dalla **sequenza di polling indica anche una gerarchia di priorità** (in caso di richieste contemporanee è esaminata la prima)
- Nel caso in cui si debba prevedere la possibilità di interrompere una routine di servizio e necessario utilizzare una istruzione di SETI che setta il bit di Interruzione (I) a 1 e renda il processore nuovamente interrompibile (ovviamente vanno presi ulteriori accorgimenti)

Istruzioni di polling

JR DISP1,SERV1
JR DISP2,SERV2
JR DISP3,SERV3
…
JR DISPn,SERVn



INTERRUZIONI

Polling: esempio

- Una semplice routine di servizio può essere implementata sfruttando il meccanismo delle interruzioni
- Si supponga di dover acquisire 4096 parole situate a partire dal settore 0x400000 dal disco magnetico (DISCOHC) e trasferirle in memoria

Dispositivo

MOVE DISCOHD_C,4096	#numero di byte da trasferire (contatore)
MOVE DISCOHD_P, 0x400000	#locazione sul disco del primo byte (indirizzo)
START DISCODH	#comando per acquisizione del primo dato

***Quando il dato è pronto viene generata una interruzione che effettua il salvataggio*:**

0 ←I	#non consente altre #interruzione (CLEAR I)
SR ←Push	#salvataggio contesto
PC ←Push	#salvataggio contesto
0x1000 ←PC	#salta al polling

INTERRUZIONI

Polling: esempio

…e punta alla sequenza di polling (che, nell'esempio, inizia alla locazione 0x1000) in cui riconosce il dispositivo attivo:

```
...                    # sequenza di polling
JR FLOPPY, SAD1        # sequenza di polling
JR DISCOCD, SAD4       # sequenza di polling
JR DISCODVD, SAD5      # sequenza di polling
JR DISCOHD, SAD3       # attiva la routine SAD3 del DISCOHD
...
```

INTERRUZIONI

Polling: esempio

```
SAD3:
        PUSH $t0                    # salva il contenuto del registro $t0
        PUSH $t1                    # salva il contenuto del registro $t1
        MOVE $t0, DISCOHD_C         # copia il valore del contatore corrente
        MOVE $t1, DISCOHD_P         # copia il valore del puntatore corrente
                  …                 # trasferisce il dato dal disco alla memoria
        SUB $t0,$t0,1  # decrementa il contatore
        ADD $t1,$t1,1 # indica il successivo indirizzo

        BNZ $t0, NEXT               # se non è ultimo dato va a NEXT
        CLEAR DISCOHD               #altrimenti rimuove la richiesta di interruzione…
        JMP EXIT                    # …e salta alla fine
NEXT:
        START DISCOHD               # altrimenti avvia l'acquisizione successiva
        MOVE DISCOHD_C, $t0         # salva valore del contatore corrente
        MOVE DISCOHD_P, $t1         # salva valore del puntatore corrente
        POP $t1                     # ripristina contenuto del registro $t0
        POP $t0                     # ripristina contenuto del registro $t1
        RTI                         # ritorna al programma interrotto RETURN FROM INTERRUPT
EXIT:
        SET I                       #consente l'acquisizione di nuova interruzione
        RTI
```

INTERRUZIONI

Interruzioni: strategie di miglioramento

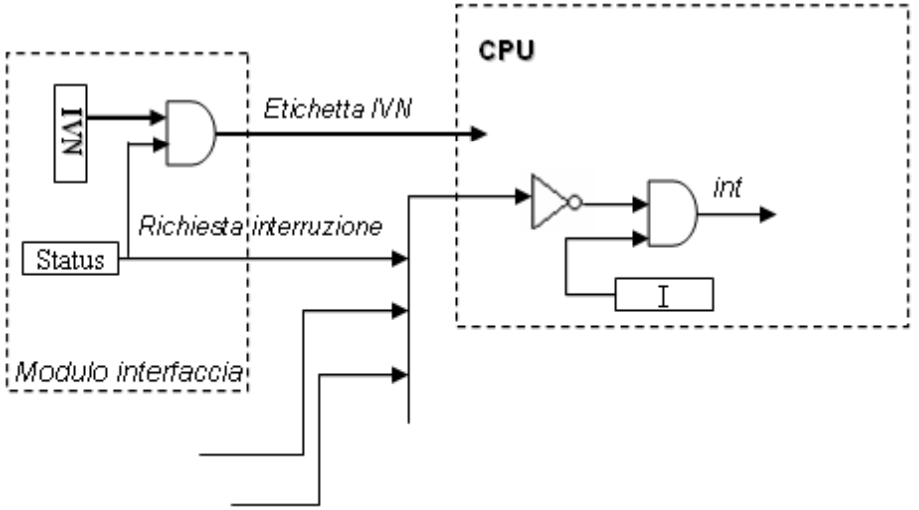
- Per migliorare l'**efficienza di un sistema di interruzione** si cerca di ottimizzare il tempo di risposta, cioè il tempo che intercorre tra l'istante in cui un dispositivo sollecita l'intervento del processore e quello in cui il processore inizia l'esecuzione della prima istruzione utile di interazione col dispositivo (dai 1-2 microsecondi a 20-30 per sistemi meno sofisticati)
- Per ottenere tale obiettivo possono essere considerati diversi approcci nelle diverse fasi che vedono coinvolti una interruzione
- Nella parte di **commutazione del contesto** si può pensare di effettuare un salvataggio delle informazioni che costituiscono il contesto **in registri appositi** (in questo modo si aumentano i costi, perché deve essere realizzato un set di registri per ogni livello di priorità corrispondente; ma si incrementa la velocità: non c'è alcuna perdita di tempo per eseguire una salvataggio nello stack)

INTERRUZIONI

Interruzione Vettorizzata

- Un ulteriore vantaggio può esserci utilizzando un **sistema di interruzione vettorizzato** *(interrupt vectored system*) grazie al quale è possibile identificare direttamente il dispositivo che richiede un servizio
- Un sistema siffatto prevede che nel modulo interfaccia di ciascun dispositivo sia presente un registro in cui è memorizzato un codice di identificazione o **numero vettorizzato di interruzione** (o *interrupt vector number*, IVN) che viene inviato al processore quando si verifica l'interruzione richiesta dal dispositivo
- Il codice potrebbe puntare direttamente alla routine di servizio, ma questa scelta non è la migliore: per questo si utilizza un **IVN** che rimanda ad un indirizzo di una locazione di memoria che contiene l'indirizzo iniziale alla routine di gestione dell'interruzione

Osservazione. La gerarchia tra i dispositivi può essere dettata dal valore del IVN



INTERRUZIONI

Interruzione Vettorizzata

- Utilizzando questo modo, il processore non ottiene solo la richiesta di interruzione, ma anche un **indirizzamento indiretto** fornito tramite l'IVN
- Il programmatore, in questo caso, deve collocare nelle celle di memoria indirizzate dai codici IVN gli indirizzi delle rispettive routine di servizio
- A questo scopo è riservata una tabella in memoria in cui risiedono gli indirizzi alle **routine di servizio** (***interrupt service Routine***, **ISR**)

Di solito la tabella risiede nella parte alta della memoria in modo tale che si possa rintracciare l'indirizzo utilizzando codici con pochi bit

Una volta finito la ISR ha una istruzione che ripristina il sistema (**RTI, return from Interrupt**)

Osservazione. Se si utilizza un IVN si dimensione 256 è possibile usare un codice di soli 8 bit

Salvataggio

```
…           #Invio segnale di blocco su altri dispositivi
0 ←I        #clear del flag I per impedire altre
            #interruzioni
SR ←Push    #salvataggio dello SR nello stack
PC ←Push    #salvataggio del PC nello stack
PC ←(IVN)   #"forzatura" del PC al vettore interruzioni
            # cioè si mette l'indirizzo contenuto nella cella
            #di memoria puntata dal valore di IVN
```

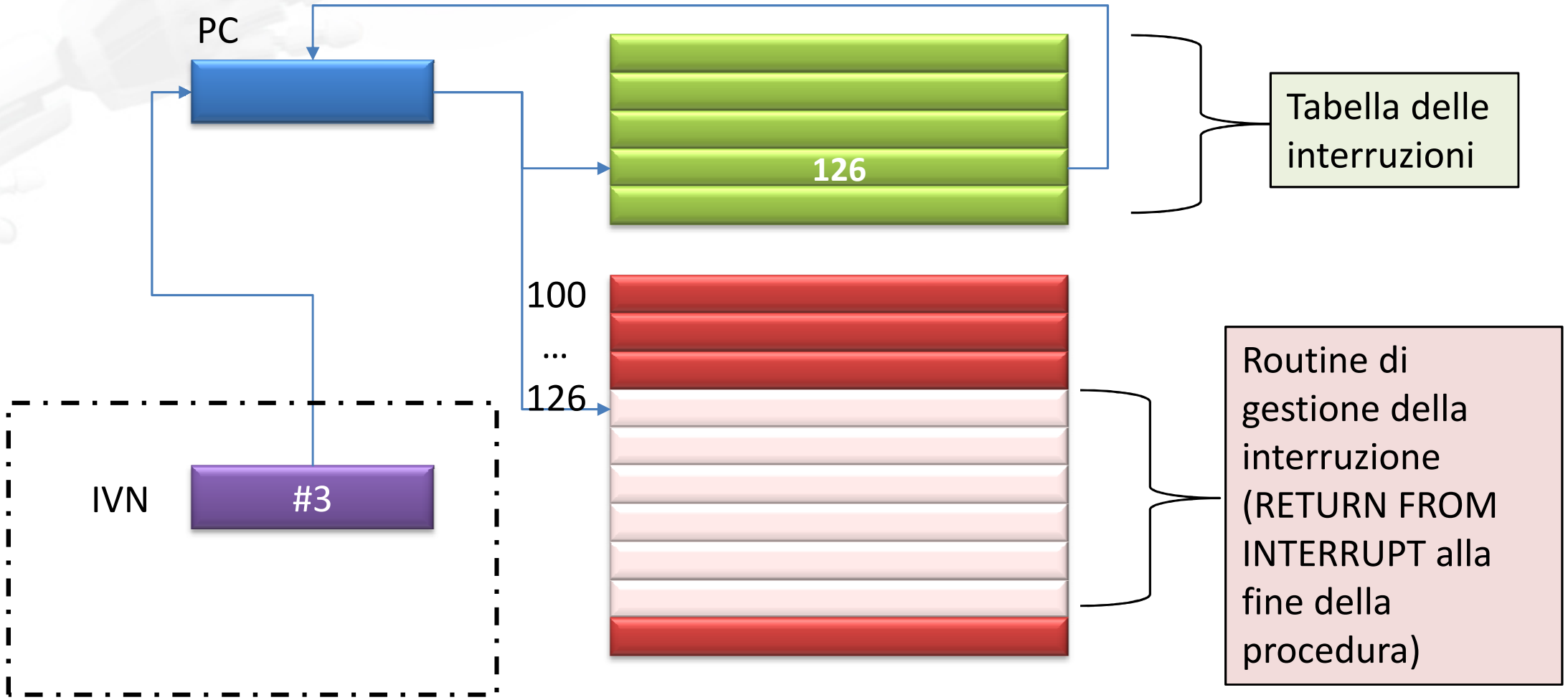
Nella ISR c'è un RITORNO DA INTERRUPT (RETI)

Ripristino

```
PC ←Pop             #ripristino PC
SR ←Pop             #ripristino SR
1 ←I        #set del flag I per impedire altre
            #interruzioni
```

INTERRUZIONI

Interruzione Vettorizzata



INTERRUZIONI

Modalità di esecuzione

- La presenza di interruzioni è spesso collegata alla gestione delle funzionalità “di basso livello” del calcolatore
  - gestione del disco, periferiche, timer, etc.
- Normalmente, i programmi per la gestione di tali funzionalità devono avere accesso a tutte le caratteristiche del calcolatore
  - è meglio evitare invece che questo sia concesso ai normali programmi utente (ad esempio quando si fa un riversamento di un file su un disco magnetico non si deve dare libertà al programmatore di scrivere dove vuole)
- Molti processori prevedono pertanto (almeno) due **modalità**:
  - **Supervisore** (accesso pieno alle funzionalità del sistema)
  - **Utente** (accesso limitato alle sole istruzioni standard)
- All’accettazione dell’interruzione, la CPU cambia modalità di esecuzione passando da Utente a Supervisore

INTERRUZIONI

Driver

- La modalità di esecuzione Supervisore è normalmente pensata per l'esecuzione di routine che sono gestite dal Sistema Operativo
- L'insieme di routine, comprese le ISR, che gestiscono l'interazione con una particolare periferica viene detto **driver**
- Ogni periferica ha il proprio funzionamento e necessita pertanto dei propri driver
  - questo è il motivo per cui l'installazione di una nuova periferica comporta normalmente l'installazione dei corrispondenti driver

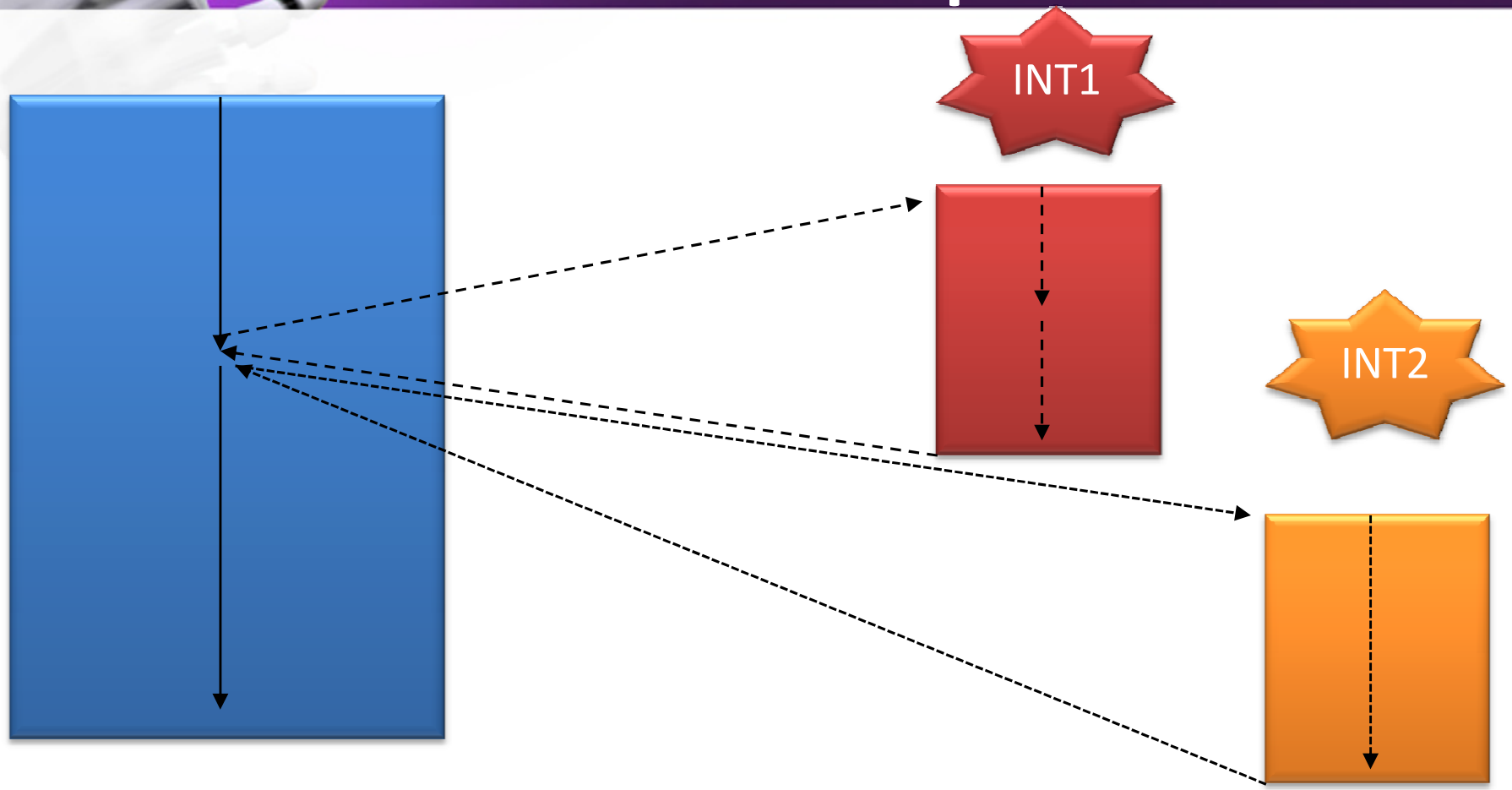
INTERRUZIONI MULTIPLE

Interruzione Vettorizzata: gestione di più interruzioni

- Nel caso di più interruzioni si può procedere in due modi
  - **Senza mascheramento**: una volta richiesta l'interruzione questa deve essere espletata senza la possibilità di essere interrotta da altre; si avrà una sequenza di interruzioni
  - **Con mascheramento**: l'interruzione può essere interrotta a sua volta per svolgerne un'altra (di solito più importante, ovvero a priorità più alta)

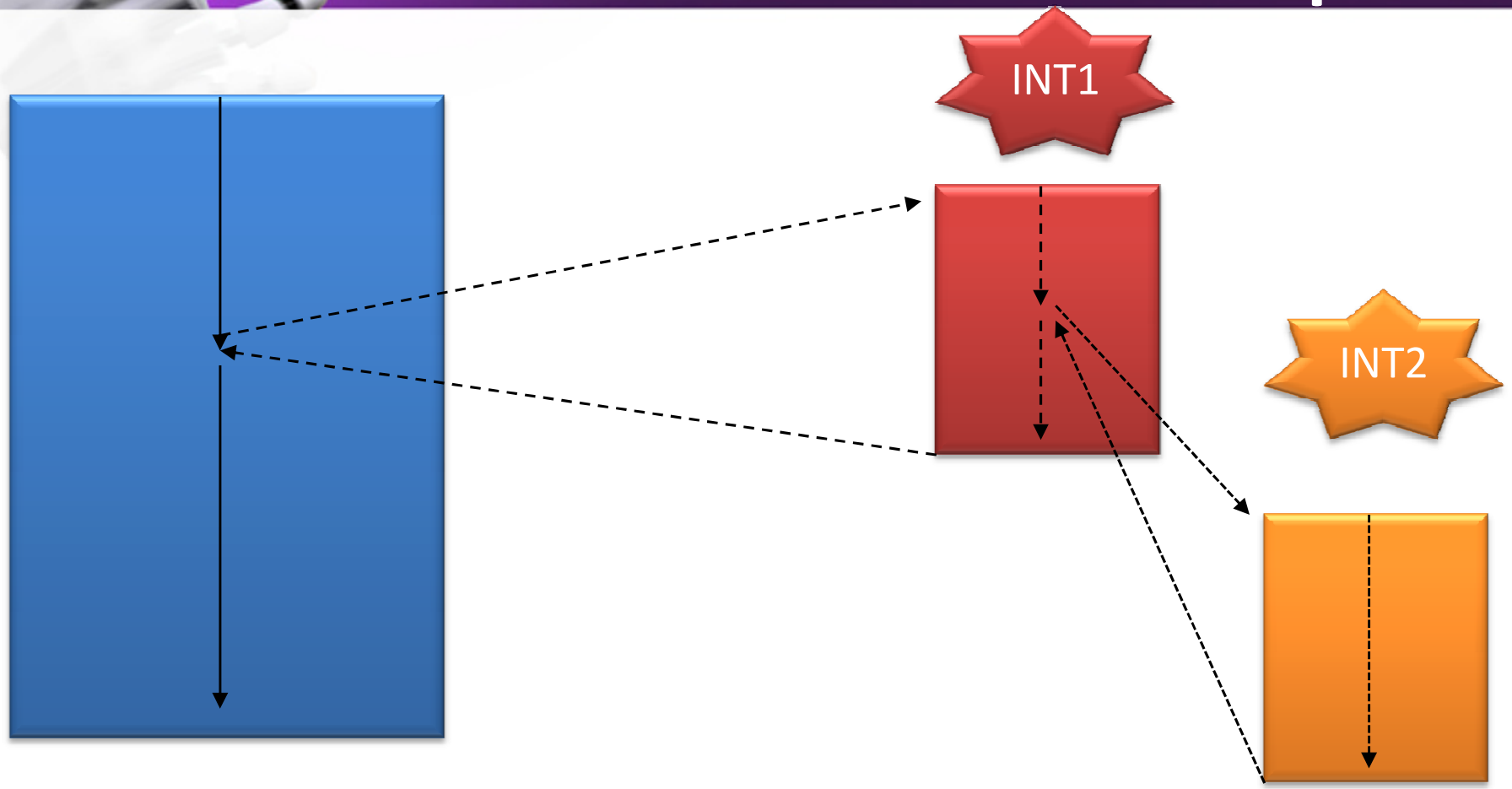
INTERRUZIONI MULTIPLE

Interruzione Vettorizzata: sequenza di interruzioni



INTERRUZIONI MULTIPLE

Interruzione Vettorizzata:interruzioni multiple innestate



INTERRUZIONI MULTIPLE

Interruzione Vettorizzata: fasi protette

- Anche se subentra una richiesta di interruzione a più alta priorità, ci sono alcune **fasi del servizio di un'interruzione** (salvataggio e ripristino del contesto) **che non possono essere interrotte (fase protetta)**

- **Sospensione esecuzione del programma corrente**
- **Salvataggio del contesto**
- **Inizializzazione del PC all'indirizzo di ingresso della routine per la gestione dell'interrupt**

Fase protetta

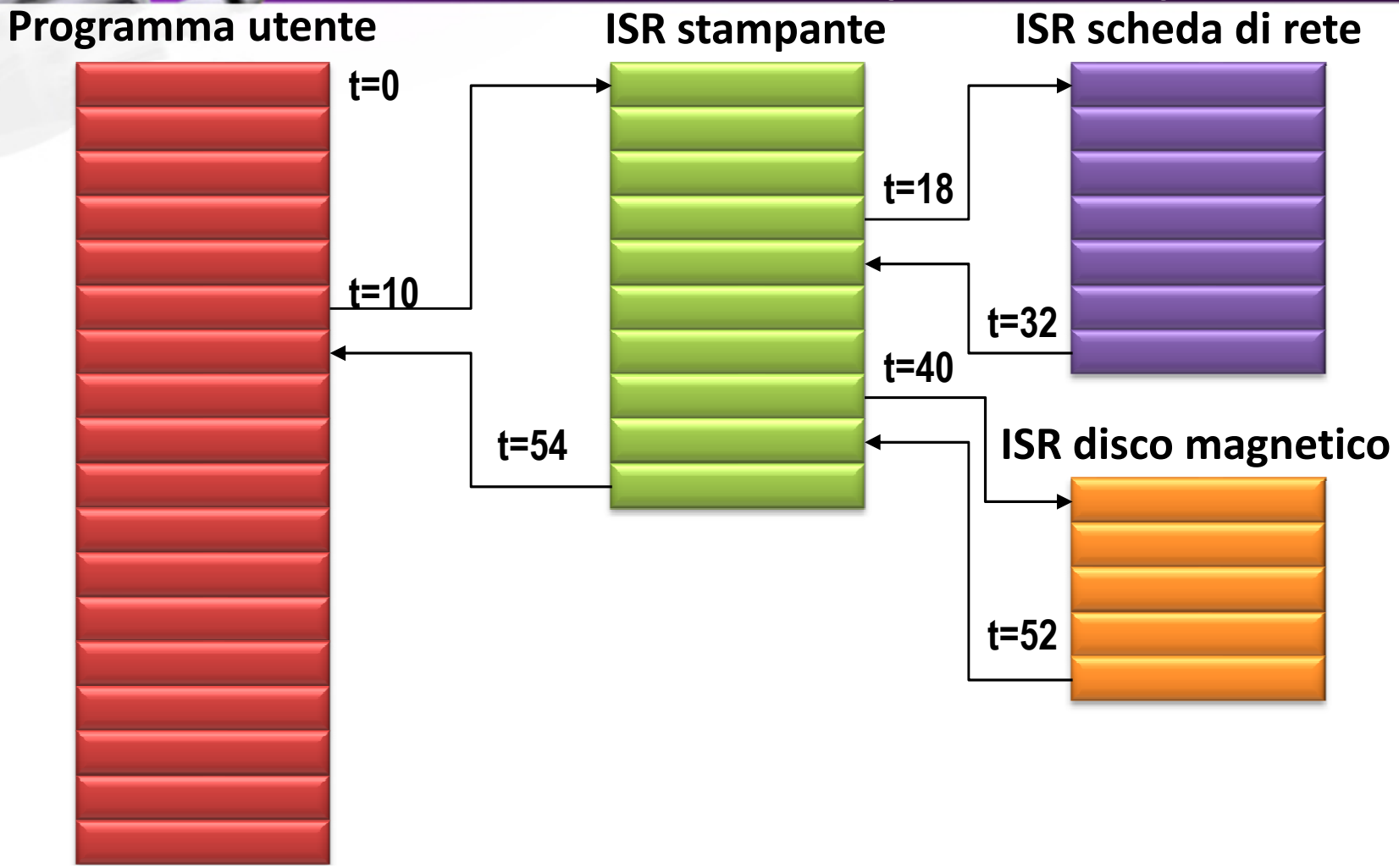
- **Esecuzione della routine di interruzione ISR**

- **Ripristino del contesto**

Fase protetta

INTERRUZIONI MULTIPLE

Interruzioni multiple: tempi



INTERRUZIONI MULTIPLE

Interruzioni multiple: tempi

- Il salvataggio dello stato incrementa il ritardo tra l'istante di ricezione della richiesta di interruzione e l'istante in cui inizia l'esecuzione della routine di interrupt
- Questo tempo viene detto **latenza di interrupt** ed indica l'intervallo temporale massimo che intercorre tra la richiesta di attenzione e l'effettivo servizio dell'interruzione

Architettura degli elaboratori

Interruzioni esterne

INTERRUZIONI ESTERNE

- Tra le principali **interruzioni esterne** vanno evidenziate:
  - Stampante
    - Richiesta di stampa
    - Stampa non collegata
    - Mancanza o inceppamento carta per la stampa
  - Mouse
  - Tastiera
    - Mancanza della tastiera
  - Video
    - Sconnessione con l'elaboratore (segnale assente)

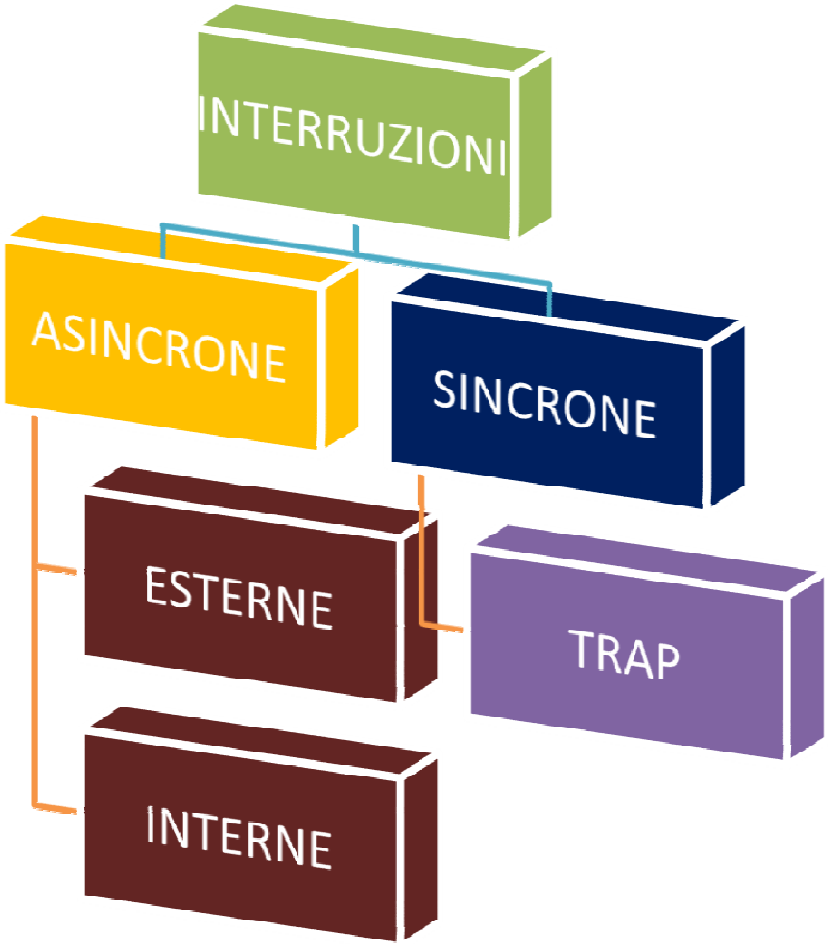
Architettura degli elaboratori

Interruzioni interne

INTERRUZIONI INTERNE

Generalità

- All'interno del calcolatore possono verificarsi anche delle **interruzioni generate dal processore stesso** (*interruzioni interne o eccezioni*) o **volutamente dal programmatore** (*trap*)
- Le interruzioni interne sono verificate da situazioni anomale rilevate dal processore nel corso dell'esecuzione di un programma e generalmente sono **asincrone** (non si sa quando accadono) come quelle esterne; mentre le trap sono **sincrone** (accadono quando stabilito dal programmatore)
- Le generazioni di queste richieste di interruzioni possono essere di tipo **recuperabili** (***soft*** o **fisiologiche**) o **irrecuperabili** (***hard*** o **patologiche**)



INTERRUZIONI INTERNE

Generalità

- Tra le principali **interruzioni interne asincrone** vanno evidenziate:
  - Il tentativo di eseguire una **divisione per zero** (*zero divide*)
  - La mancanza di **tensione di alimentazione** (*power failure*)
  - **Indirizzo errato** (*address error*): condizione che si verifica quando si chiede l'accesso ad una cella di memoria non fisicamente presente. Questa interruzione è sfruttata nella **memoria virtuale** che sarà presentata in seguito
  - Esautoramento dello stack (*stack overflow*)
- Tra quelle sincrone c'è:
  - L'esecuzione della **modalità trace** (*single step*): si verifica una interruzione dopo ogni istruzione eseguita
  - Il **TRAP**

Architettura degli elaboratori

Interruzioni software

INTERRUZIONI SOFTWARE

Generalità

- Nei processori attuali si prevede la possibilità che un programma richieda in modo esplicito un'interruzione a se stesso.
- Si ha una **interruzione software** (o trap): una interruzione voluta e scritta dal programmatore.
- Al contrario delle comuni interruzioni fino ora viste, che sono segnali asincroni (è imprevedibile stabilire quando occorrano), le interruzioni software invece si verificano in corrispondenza dell'istruzione che le richiedono (segnale sincrono)
- Il meccanismo delle trap è uguale alle altre interruzioni e **simile al salto a subroutine** (infatti le trap sono incluse di diritto nella classificazione delle istruzioni). In entrambi i casi si ha il salvataggio dello stato volatine della macchina, ma la differenza è nell'attivazione: nel salto a subroutine si conosce l'indirizzo della routine a cui si vuole saltare; nelle trap invece l'indirizzo è ricavato dalla tabella del IVN e non richiede che il programma chiamante conosca l'indirizzo del sotto-programma chiamato. Questa strategia è utile anche per richiamare moduli oggetti e librerie esterni al codice eseguibile in corso di elaborazione

INTERRUZIONI SOFTWARE

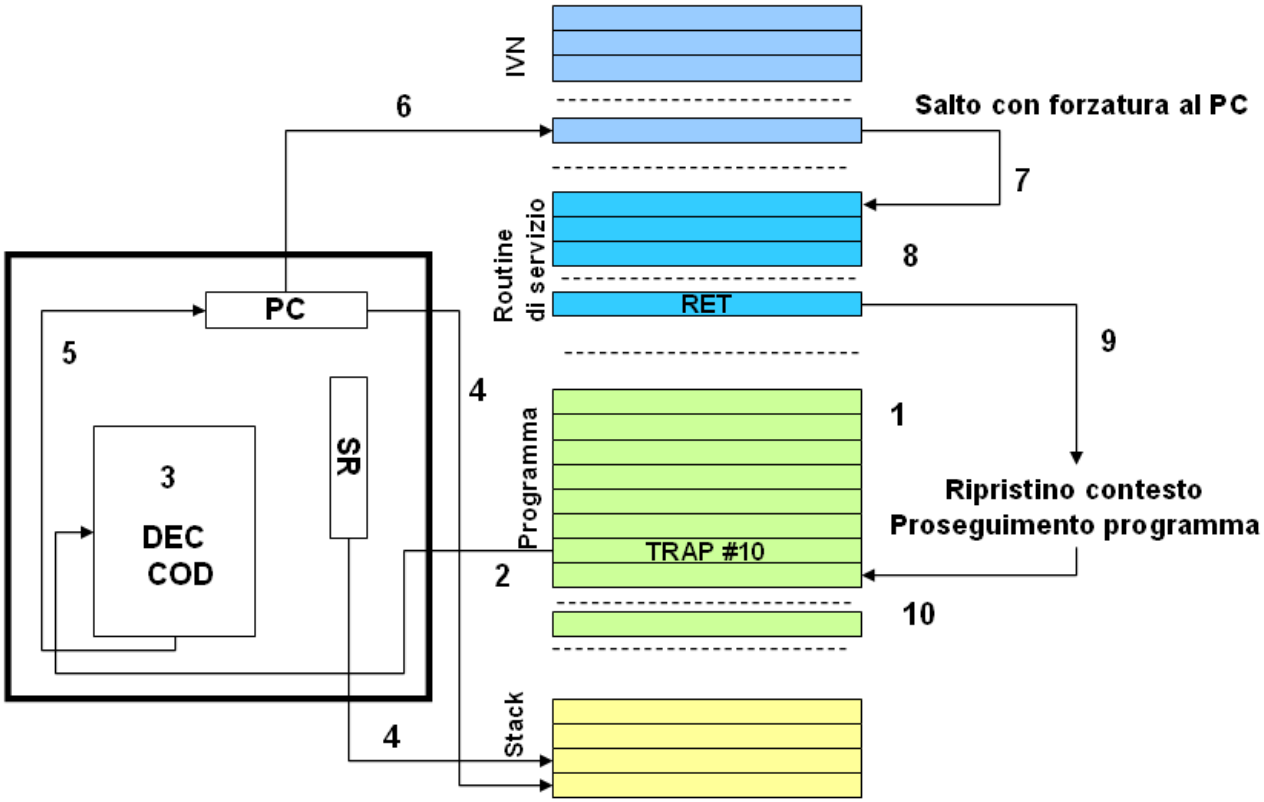
Generalità

- Il trap è utilizzato per usufruire delle librerie di sistema che in questo modo possono essere modificate (nelle versioni successive) in modo del tutto trasparente ai programmi degli utenti. Per questo motivo molto spesso si usa il sinonimo di **chiamata a sistema** (syscall)
- In questo caso l'indirizzo contenuto nella locazione di memoria puntata dal TRAP riporta l'indirizzo della routine o libreria a cui si deve accedere (esempio di multiprogrammazione), alla fine della libreria c'è un ritorno esplicito al programma originale

INTERRUZIONI SOFTWARE

Esecuzione

1. Esecuzione del programma
2. Caricamento istruzione TRAP con identificativo (o etichetta) #10
3. Decodifica istruzione (riconoscimento di interruzione software)
4. Esecuzione
   A. Commutazione del contesto: salvataggio SR e PC nello stack e dello stato volatile
5. Forzatura del PC all'indirizzo individuato dall'etichetta #10 ed eventuale mascheramento
6. Salto a IVN
7. Salto alla routine di servizio (si forza anche in questo caso il PC all'indirizzo trovato nell'IVN)
8. Esecuzione routine di servizio
9. Ripristino del contesto
   A. Recupero dallo stack del SR e del PC dello stato volatile della macchina
10. Prosecuzione istruzione del programma



INTERRUZIONI SOFTWARE

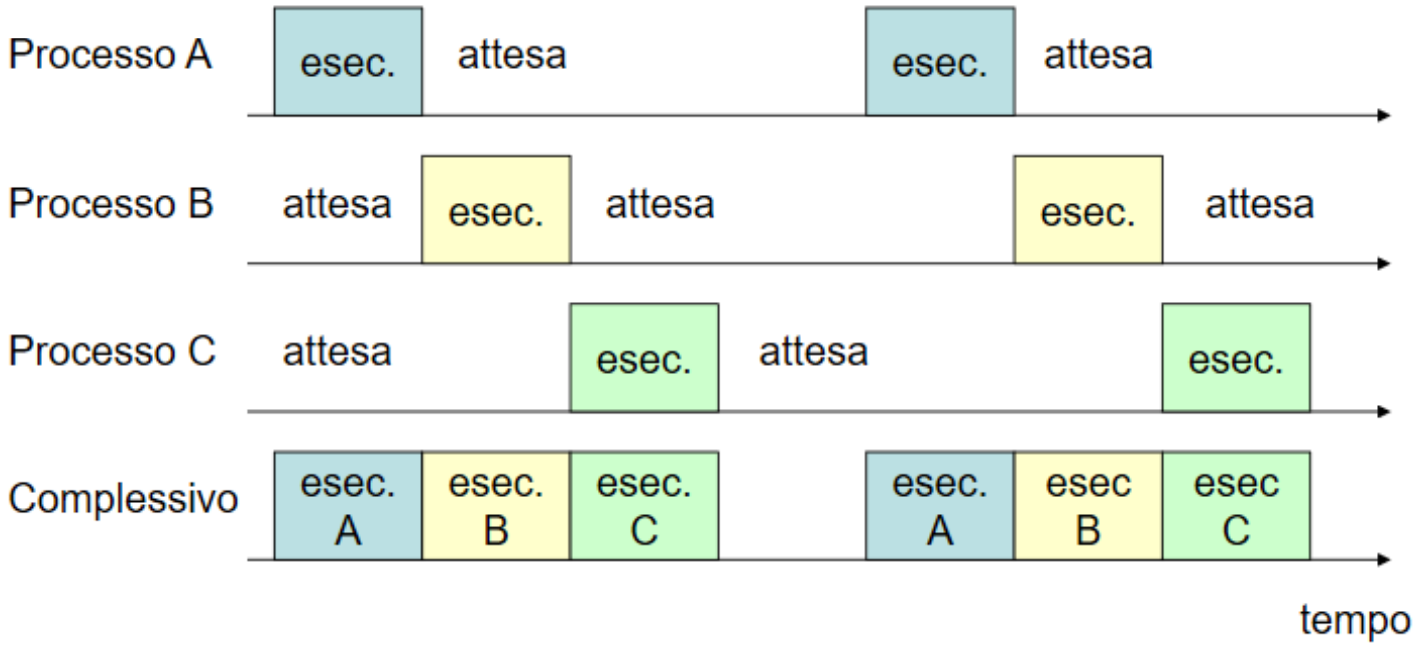
Esempio IVN

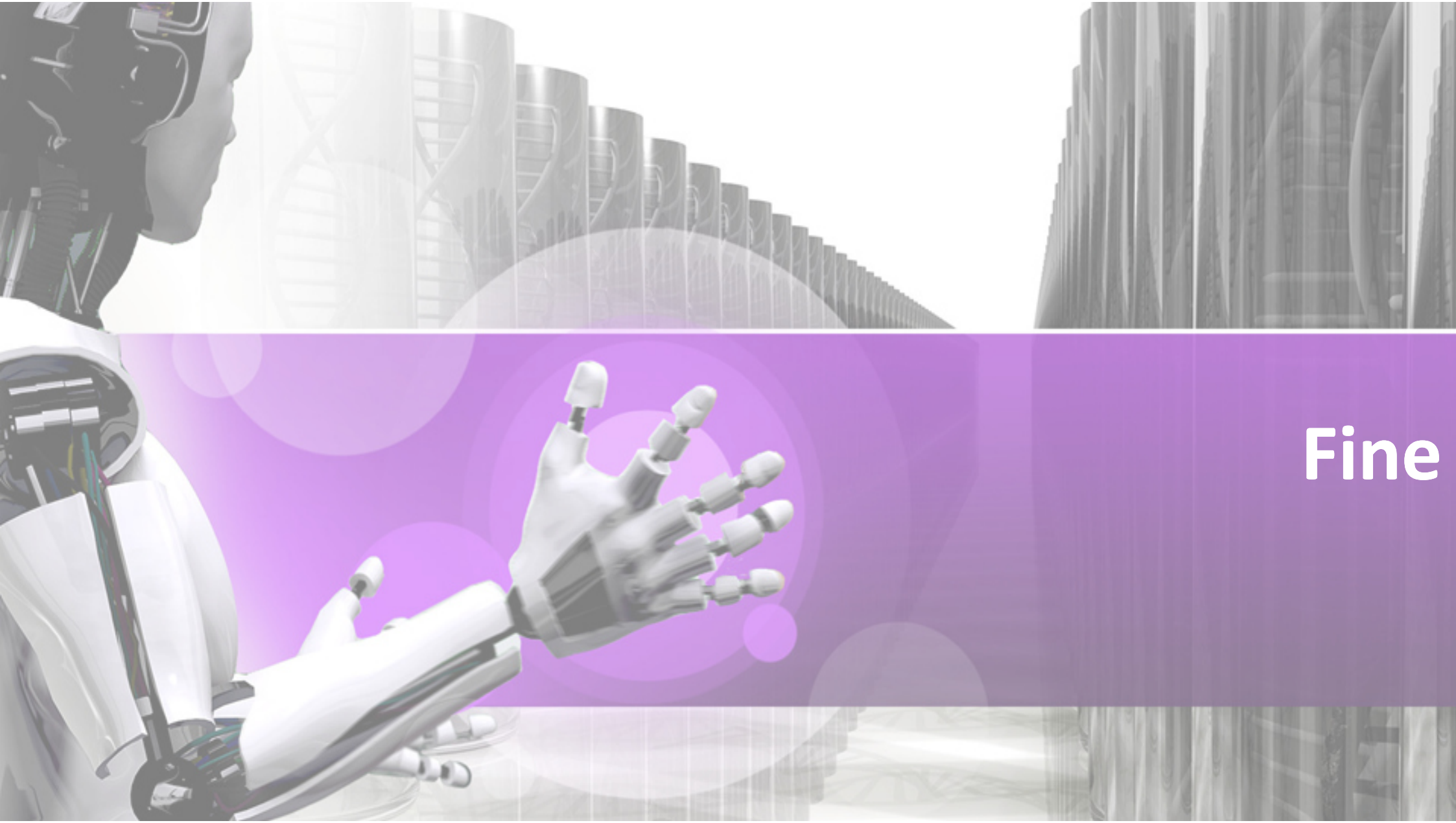
Numero vettore (IVN)	Indirizzo	Assegnamento
0	0	**Reset PC**
2	8	**Errore Bus**
3	12	**Errore di Indirizzo**
4	16	**Istruzione illegale**
5	20	**Divisione per zero**
…	…	**…**
9	36	**Trace**
…	…	**…**
12-23	48-95	**Riservati**
…		
32	128	**Trap #0**
33	132	**Trap #1**
…	…	**…**
47	188	**Trap#15**
…	…	**…**
65-255	256-1023	**Vettori interrupt utente**

INTERRUZIONI SOFTWARE

Multiprogrammazione

- Inoltre grazie alle interruzioni generate nel sistema operativo a tempi prestabiliti (grazie ad un timer) è possibile **la multiprogrammazione**: cioè più programmi eseguiti in memoria ad intervalli regolari e separati che danno un effetto di pseudo parallelismo




Fine