Corso di Calcolatori Elettronici I

Interruzioni

ing. Alessandro Cilardo

Corso di Laurea in Ingegneria Biomedica

Gestione di eventi

- Ciclo base del processore:
  - le istruzioni sono eseguite una dopo l'altra in sequenza
  - eventuali salti spostano l'esecuzione ad una differente sequenza sulla base di condizioni comunque dipendenti dal programma in esecuzione
- In molti casi, tuttavia, è necessario che il processore esegua una routine *quando si verifica un evento esterno*, indipendente dal programma in esecuzione
  - l'evento accade in momenti che non sono predicibili da parte del processore

Gestione di eventi

Esempi di eventi esterni

- pressione di un tasto sulla tastiera
- ricezione di un dato attraverso un'interfaccia di comunicazione
- interruzione da parte di un timer
- segnalazione del completamento di un'operazione di trasferimento dati tra disco e memoria, etc etc

Gestione di eventi

- Gli eventi esterni sono quasi sempre asincroni.
- Spesso si presentano con frequenza *molto più bassa* rispetto alla frequenza con la quale il processore carica ed esegue le istruzioni
- Nel ***polling*** (o *I/O controllato da programma*)
  - la CPU accede ripetutamente ai registri di stato fino a quando la periferica è pronta (come nell'esempio del Terminale)
  - impedisce alla CPU di fare altro durante l'attesa!
  - La CPU si blocca in quello che si chiama *busy-waiting*

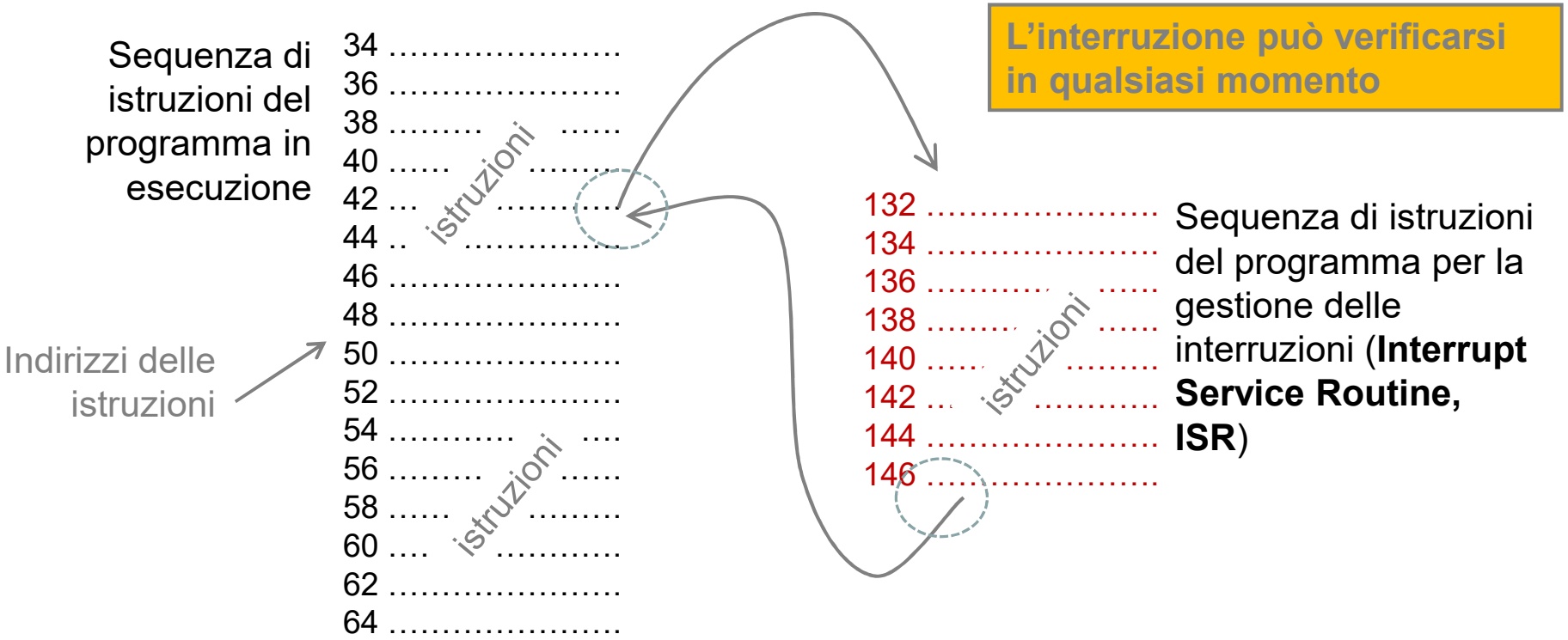
Gestione di eventi

- Alternativa: "***Interruzioni***" (***interrupts***)
  - dette anche "***eccezioni***" *(**exceptions**)*
  - Un'interruzione è un *segnale* mandato direttamente dalla periferica al processore per interromperne l'esecuzione e lanciare l'esecuzione di un programma differente
  - Il processore può eseguire altre istruzioni mentre la periferica lavora
  - Appena questa è pronta, il processore viene interrotto (tramite un'opportuna infrastruttura circuitale fornita dal sistema), e passa ad eseguire il programma che gestisce l'interazione con la periferica

Interruzioni asincrone

- Tipicamente (ma non necessariamente) le interruzioni sono innescate da un evento esterno, ad esempio la pressione di un tasto, e possono quindi **avvenire in qualsasi momento**
  - L'interruzione è ***asincrona*** rispetto al programma
- L'accettazione di un'interruzione comporta la sospensione del programma in esecuzione ed il salto ad una routine per la gestione dell'interruzione (***Interrupt Service Routine*, ISR**)
  - **NON** si tratta quindi di un salto o di una chiamata a sottoprogramma eseguita "consapevolmente" dal programma interrotto

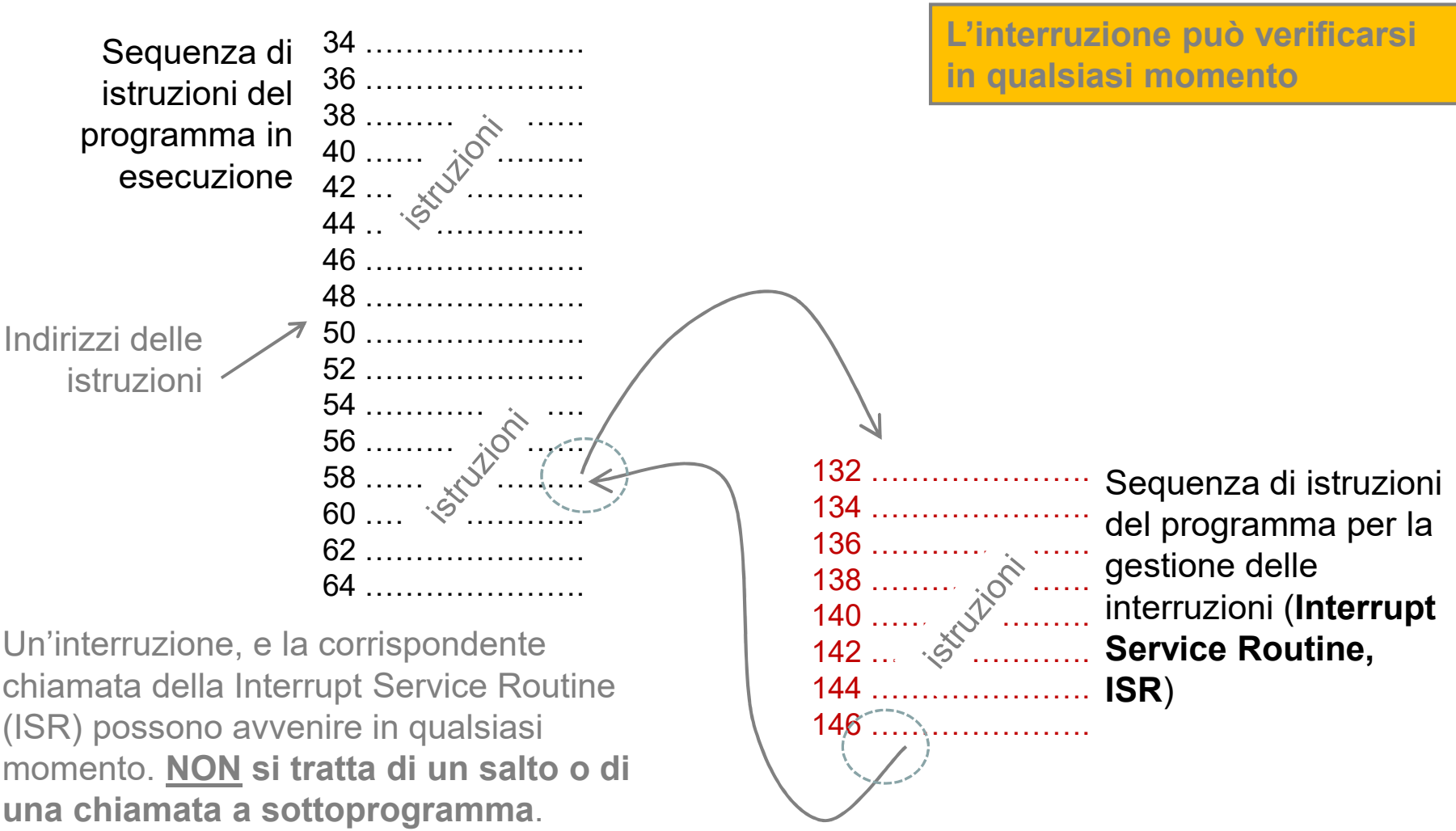
Interruzioni asincrone



Un'interruzione, e la corrispondente chiamata della Interrupt Service Routine (ISR) possono avvenire in qualsiasi momento. **NON si tratta di un salto o di una chiamata a sottoprogramma**.

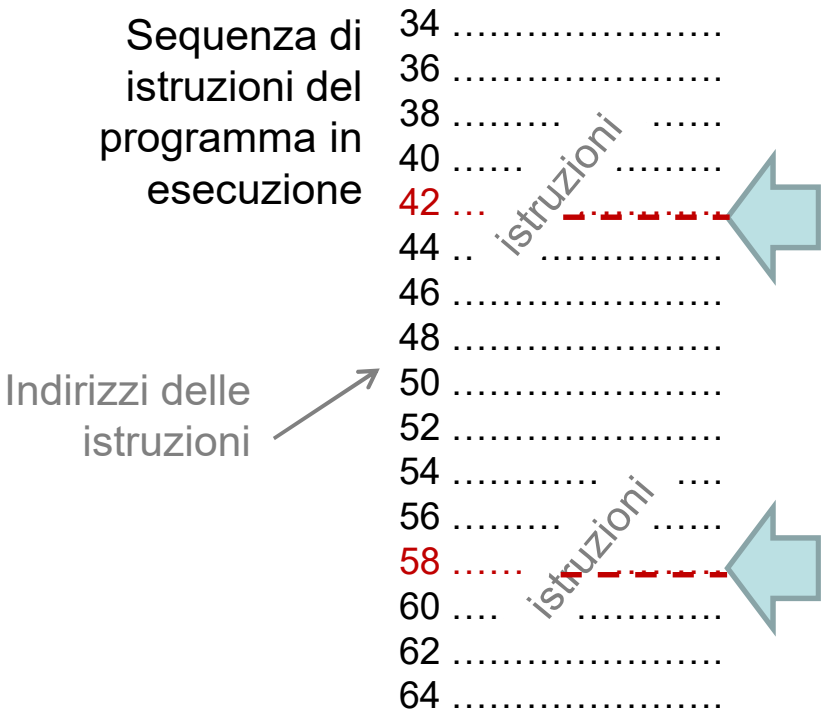
Interruzioni asincrone

L'interruzione può verificarsi in qualsiasi momento



Un'interruzione, e la corrispondente chiamata della Interrupt Service Routine (ISR) possono avvenire in qualsiasi momento. **NON si tratta di un salto o di una chiamata a sottoprogramma**.

Interruzioni asincrone

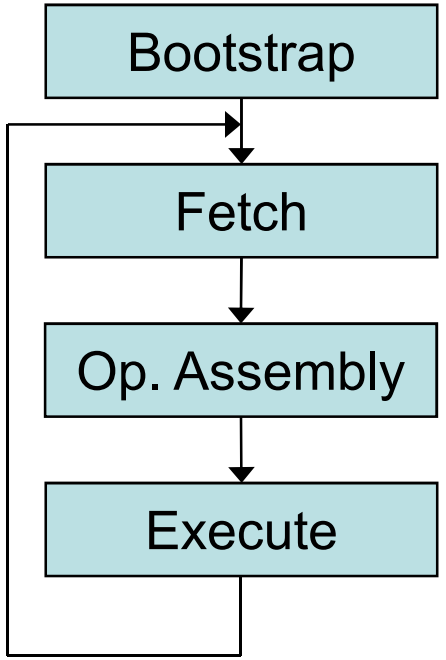


Poiché il programma viene interrotto senza alcuna possibilità di prevedere il punto esatto in cui avverrà la sospensione, è indispensabile che la ISR che passa in esecuzione **salvi e poi ripristini** lo stato del processore (**registri**) e della memoria (**stack**) in modo che tornino ad avere esattamente gli stessi valori assunti nel momento della sospensione

Un'interruzione, e la corrispondente chiamata della Interrupt Service Routine (ISR) possono avvenire in qualsiasi momento. **NON si tratta di un salto o di una chiamata a sottoprogramma**.

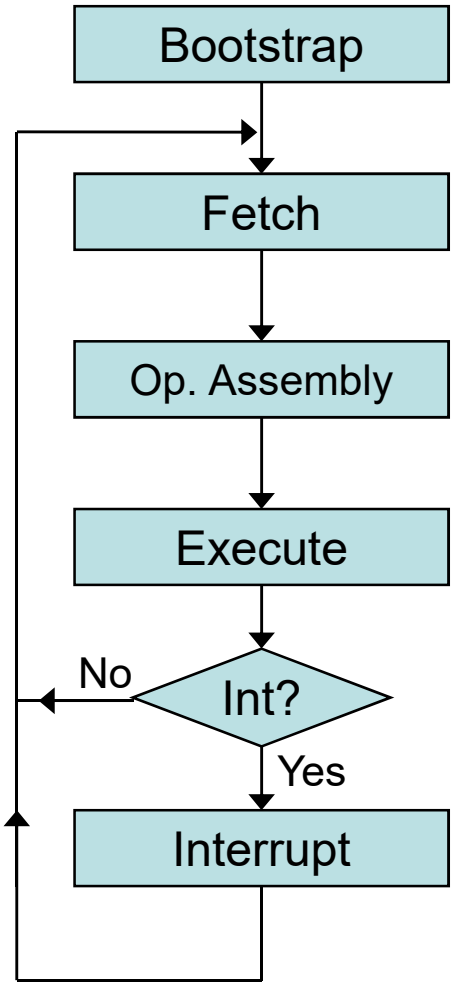
Modifica al ciclo di Von Neumann

- *Come realizzare le interruzioni*?
- Se il ciclo del processore fosse puramente quello mostrato a destra, sorgerebbero alcuni problemi:
  - un'applicazione potrebbe impadronirsi della risorsa CPU senza mai terminare
  - non ci sarebbe modo di rimuovere forzatamente un'applicazione che entri per errore in un ciclo infinito
  - il ***Sistema Operativo***, in generale, avrebbe un controllo limitato sul sistema



Modifica al ciclo di Von Neumann

- La soluzione comunemente adottata consiste nel permettere al "***supervisore***" (il Sistema Operativo) di prendere periodicamente il controllo del processore
- Questo avviene esclusivamente nel caso in cui si verifichino eventi "eccezionali", di solito asincroni con l'esecuzione del programma correntemente in corso
- In assenza di tali eventi l'elaborazione procede nella maniera consueta



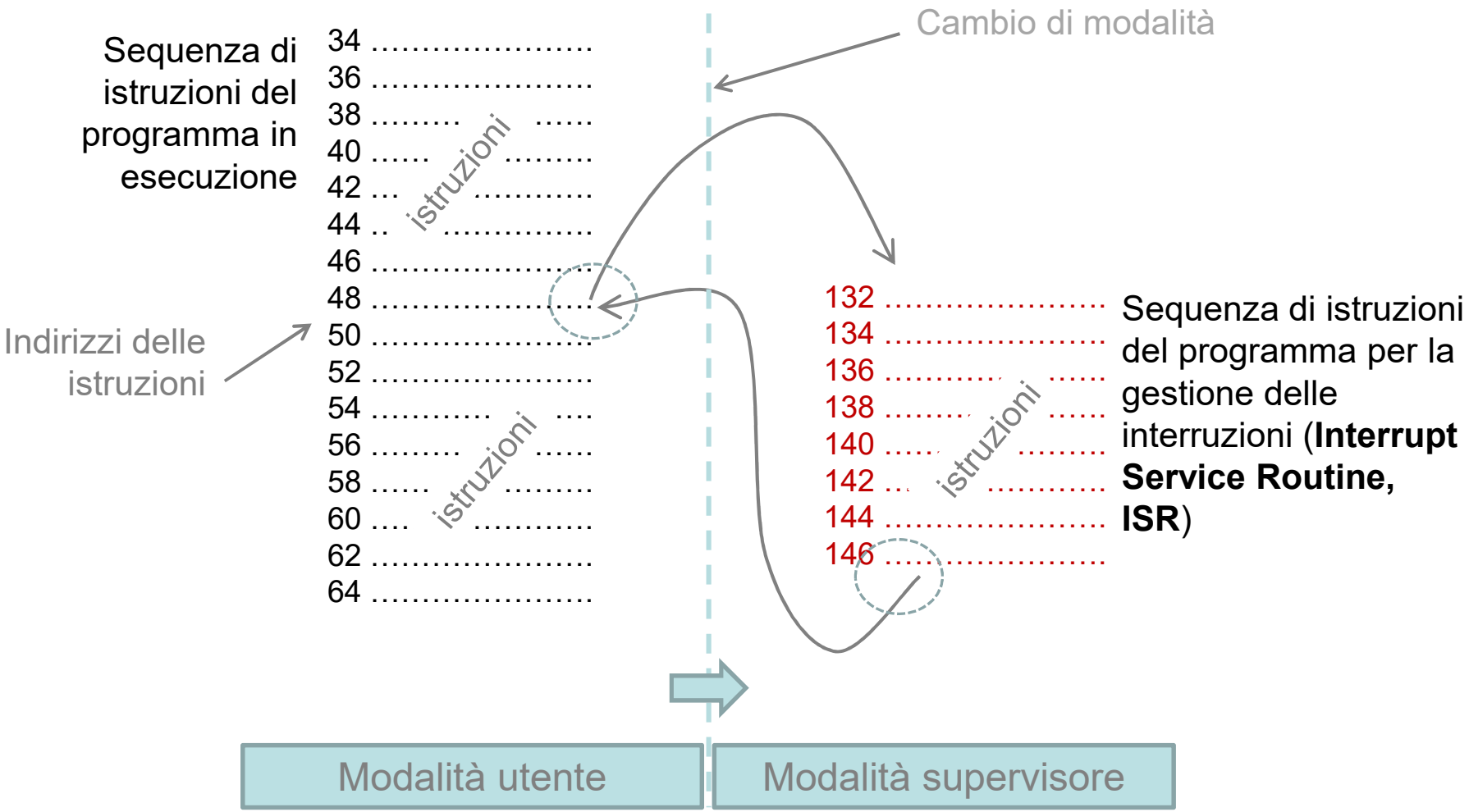
Modalità di esecuzione

- La presenza di eventi 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
  - tipicamente, è meglio evitare invece che questo sia concesso ai normali programmi utente
- Molti processori prevedono pertanto (almeno) due modalità:
  - **Supervisore** (accesso pieno alle funzionalità del sistema)
  - **Utente** (accesso limitato alle sole istruzioni di processazione)
- All’accettazione dell’interruzione, la CPU **cambia modalità di esecuzione** passando da Utente a Supervisore

Modalità di esecuzione

- La modalità di esecuzione attiva in un certo momento è parte dello *stato* del processore
- Tipicamente, in modalità di esecuzione differenti il processore accede anche a *stack differenti*
  - Ad esempio nel **M68000** la modalità è indicata dal bit **S** del *Registro di Stato* (**SR**)
  - Sono presenti due Stack Pointer (**A7** e **A7'**) usati rispettivamente in modalità Utente e Supervisore

Modalità di esecuzione



Driver

- La modalità di esecuzione Supervisore è normalmente pensata per l'esecuzione di routine del 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!

Tipologie di interruzioni

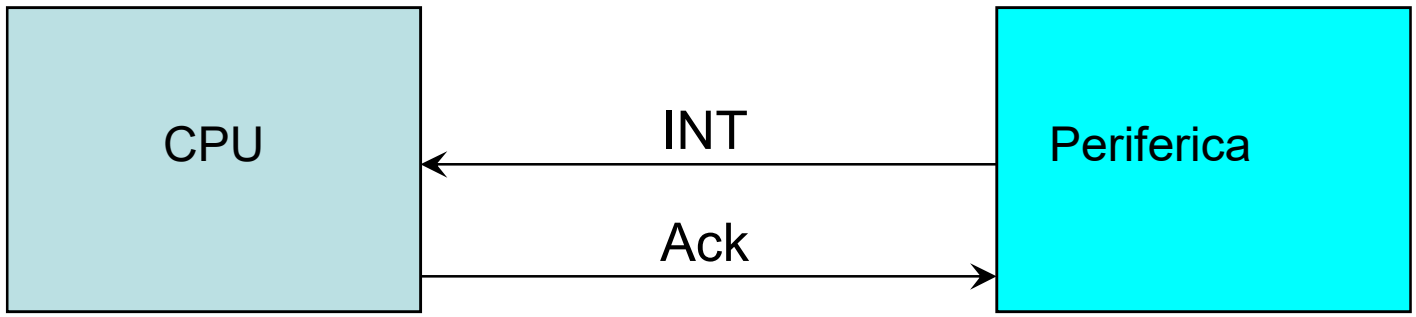
- Reset
  - Riporta la macchina in uno stato iniziale noto
  - È generato da condizioni d'errore non recuperabili
- Traps
  - Forniscono un meccanismo controllato di passaggio allo stato supervisore
  - Sono eventi sincroni (rispetto all'elaborazione)
- Interrupts in senso stretto
  - Permettono di gestire richieste di "attenzione" da parte di periferiche (tipicamente di I/O)
  - Sono eventi asincroni (rispetto all'elaborazione)

Traps

- Generano un'interruzione al termine di ogni istruzione eseguita, o in corrispondenza di particolari circostanze determinate dal programma
  - sono pertanto ***sincrone*** con il programma
- Sono usate per consentire l'esecuzione passo-passo ed il *debug* dei programmi
- Nel M68000 sono abilitate dal bit **T** del registro di Stato (**SR**)

Interruzioni: funzionamento

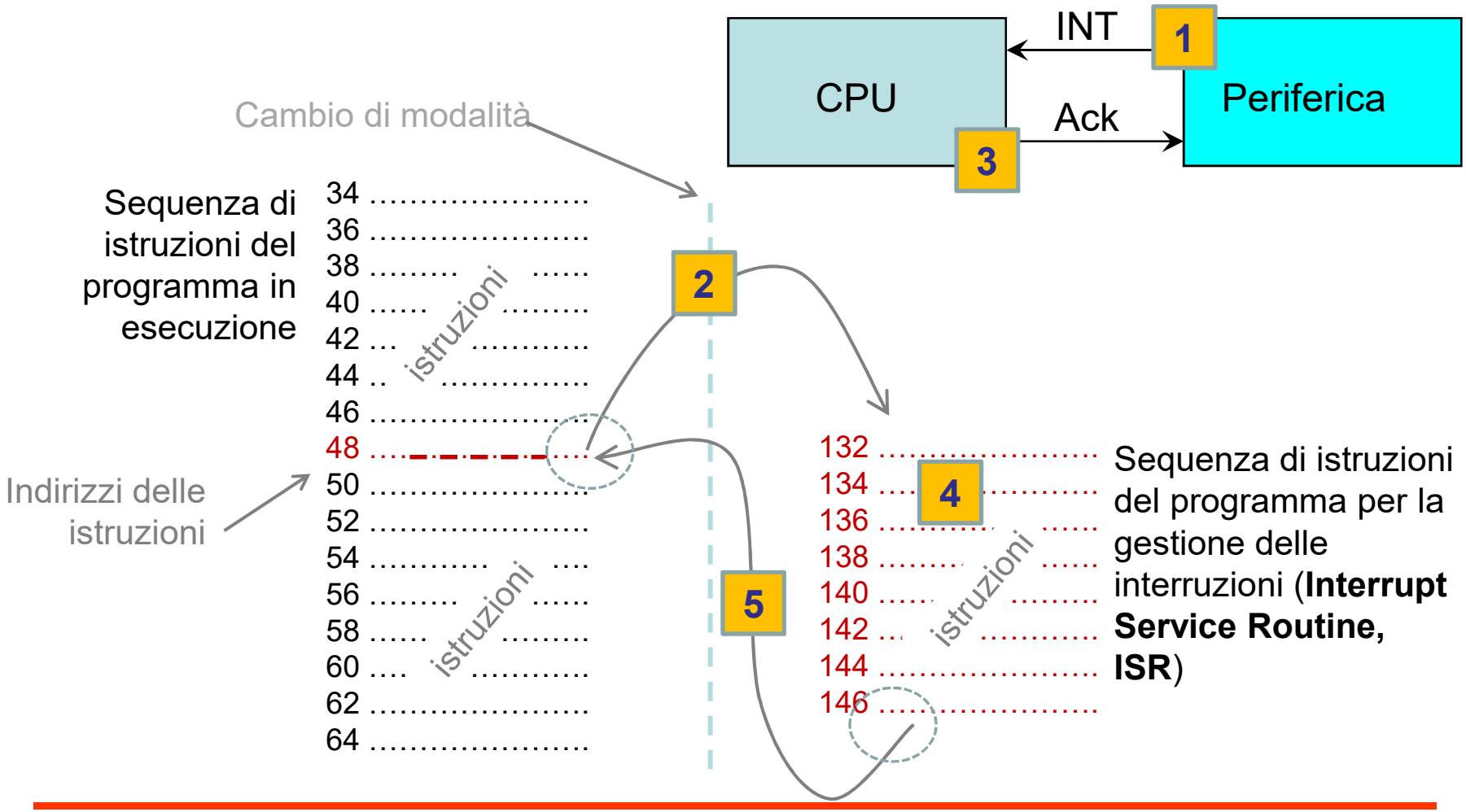
- Segnale di Interrupt (**INT**)
  - Segnale di interruzione per il processore
- Segnale di **Ack**
  - Segnale di accettazione dell'interruzione
- **ISR** (Procedura di Servizio degli Interrupt)
  - Procedura lanciata quando viene accolto l'*interrupt*



Interruzioni: funzionamento

- 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 originale

Interruzioni: funzionamento



Sequenza di attivazione: dettagli

- Esecuzione normale
- Servizio dell'interruzione
  - Salvataggio del contesto (hardware)
  - Identificazione del device
  - Salto all'*entry-point* della Interrupt Service Routine (ISR)
  - Salvataggio del contesto (software)
  - Servizio dell'interruzione
  - Ripristino del contesto (software)
  - Ripristino del contesto (hardware)
- Ripresa dell'esecuzione normale

Il salvataggio dello stato

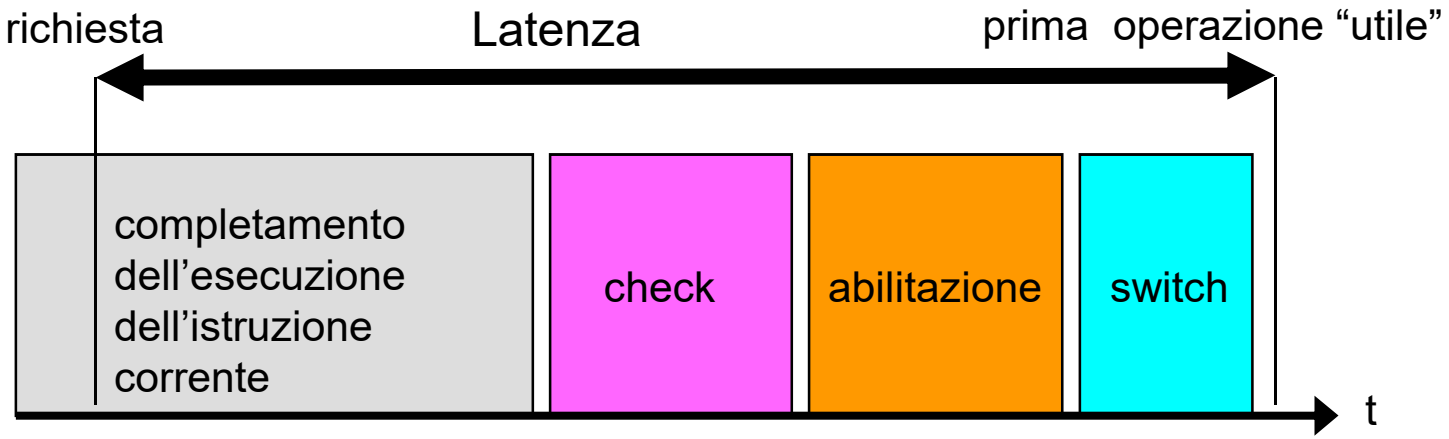
- L'elaborazione eseguita dalla ISR in risposta all'interruzione potrebbe essere completamente indipendente da quella interrotta
  - questa deve poter riprendere l'esecuzione dal punto dell'interruzione senza "accorgersi" di nulla (a parte il ritardo dovuto al servizio dell'interruzione)
  - Occorre salvare (prima) e ripristinare (dopo) lo stato del programma che viene di volta in volta interrotto
- Le informazioni da salvare e ripristinare comprendono:
  - il valore corrente del Program Counter (**PC**)
  - i flag di condizione e, in generale, tutto lo Status Register (**SR**)
  - il contenuto di qualsiasi registro che sia usato sia dal programma che dalla routine di gestione dell'interruzione

Il salvataggio dello stato

- Program Counter (**PC**) e Status Register (**SR**) vanno salvati in ogni caso
  - Il salvataggio deve necessariamente essere fatto in automatico all'accettazione dell'interruzione da parte di un apposito circuito presente nel processore
- La parte restante dello stato di esecuzione (**registri** e **Stack Pointer**) può essere salvata in software dalla stessa ISR, tipicamente mediante dei push sullo stack
  - Il salvataggio può richiedere trasferimenti di dati in memoria
  - questo carico aggiuntivo deve essere ridotto al minimo! (Le interruzioni possono essere molto frequenti)
  - meglio salvare i registri solo se questo è effettivamente necessario

Latenza di un'interruzione

- 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***
- È il tempo massimo che intercorre tra la richiesta di attenzione e l'effettivo servizio dell'interruzione



Aspetti da gestire

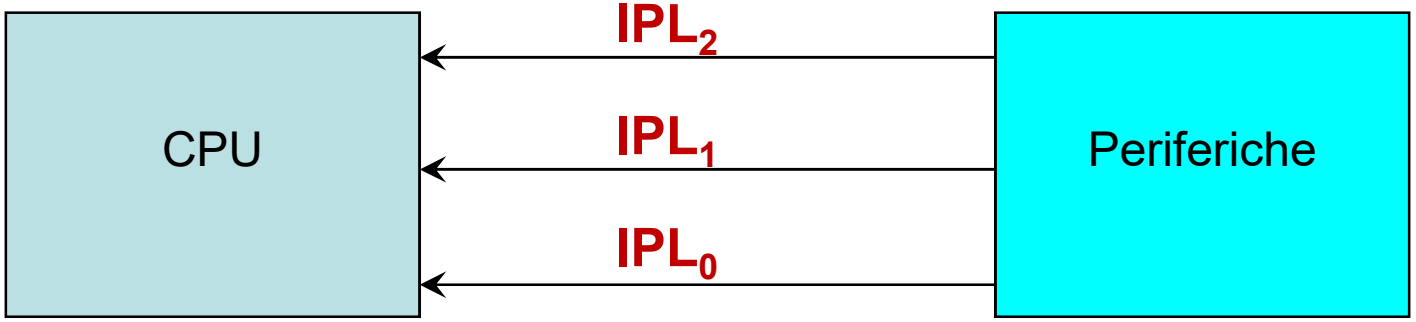
- Interrupt multipli e priorità
  - le fonti di interrupt sono molteplici ed operano indipendentemente l'una dall'altra (ad es., diverse periferiche collegate al calcolatore)
  - occorre definire una ***priorità*** per gestire il caso in cui si presentino più interrupt *nello stesso momento*
- Interrupt innestati
  - occorre gestire il caso in cui un'interrupt a priorità maggiore ne interrompa una a priorità minore
  - deve essere possibile gestire interrupt innestati (come nel caso delle subroutine), salvando e ripristinando lo stato di ciascuna ISR a sua volta interrotta

Aspetti da gestire

- Presenza di più periferiche
  - spesso le linee fisiche per la segnalazione dell'interrupt (ad es., il segnale **INT**) sono poche, o comunque in numero inferiore rispetto alle periferiche fisicamente connesse al calcolatore
  - Più periferiche possono quindi *condividere* la stessa linea fisica
  - ad esempio, un unico segnale di **INT** si alza se *almeno una* periferica ad esso collegata ha generato un interrupt
  - Quando accetta un'interrupt, però, la CPU deve adesso anche identificare la periferica che l'ha generata
  - **Interrupt vettorizzate**: ogni periferica manda un codice identificativo sul bus (chiamato ***vettore***) per farsi riconoscere dalla CPU
  - il vettore può poi essere usato per individuare la specifica ISR che deve gestire quella particolare interrupt

Interruzioni nel M68000

- Sono presenti 3 linee fisiche, che codificano sia la presenza di interrupt che il suo livello di priorità:
  - (IPL_2, IPL_1, IPL_0) = **000** → *assenza di interrupt*
  - (IPL_2, IPL_1, IPL_0) = **001** → *interrupt a priorità più bassa*
  - . . . . . .
  - (IPL_2, IPL_1, IPL_0) = **110** → *interrupt a priorità più alta*
  - (IPL_2, IPL_1, IPL_0) = **111** → *RESET* (ha priorità su tutto)

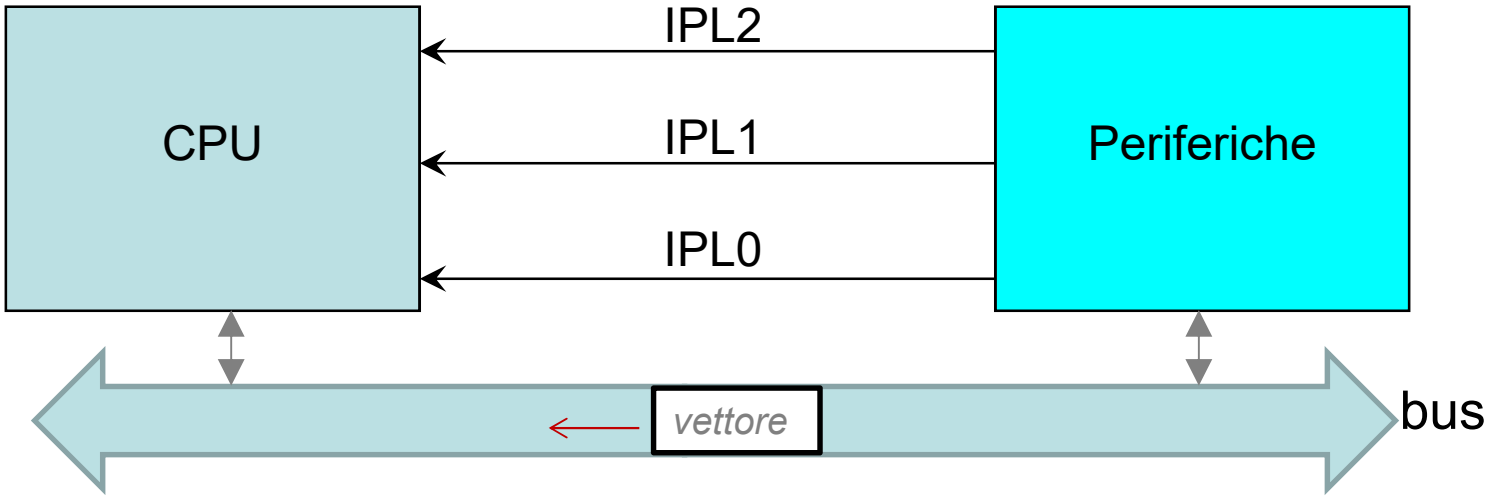


Interruzioni nel M68000

- Se il processore sta eseguendo una **ISR**, lo **Status Register** contiene all'interno dei tre bit $\mathbf{I_2}$, $\mathbf{I_1}$, $\mathbf{I_0}$ il livello di priorità dell'interrupt corrente
  - mentre il bit **S** è in ogni caso **1**, visto che una **ISR** è *sempre eseguita* in modalità supervisore
- Se si presenta un'interrupt con un **livello di priorità** minore o uguale a quello corrente, l'interrupt innestata non è ammessa e *la richiesta di interruzione non viene accettata*

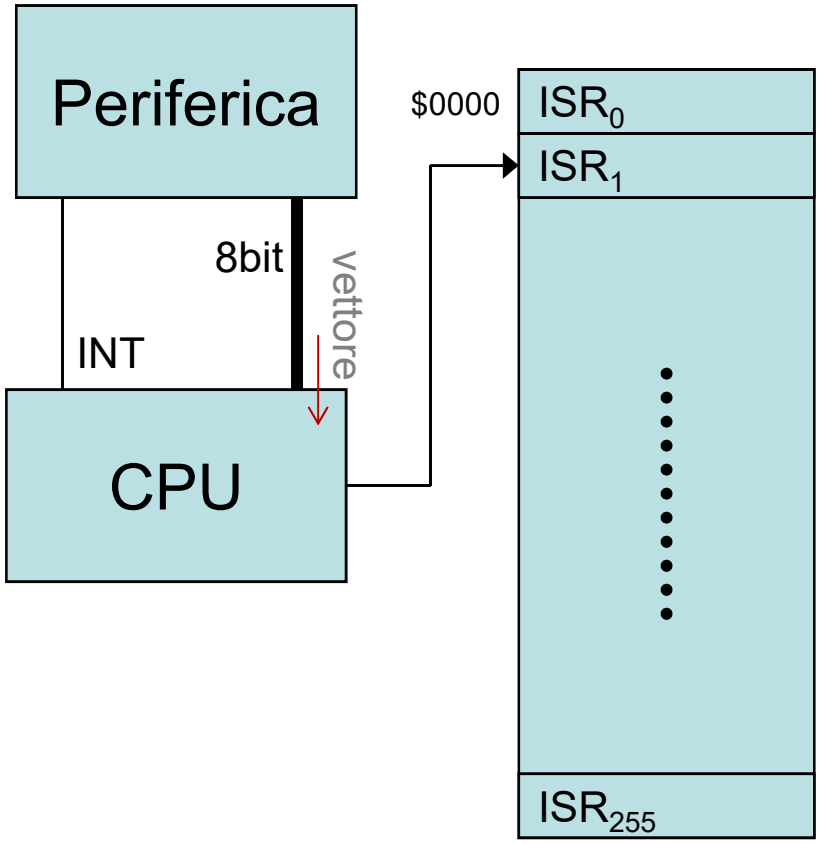
Interruzioni nel M68000

- Più periferiche possono condividere lo stesso livello di priorità
- All'accettazione dell'interruzione, la CPU aspetta sul bus di sistema un dato di 8 bit, il ***vettore***, inviato dalla periferica (diverso per ciascuna periferica)
- Il vettore è usato per identificare la specifica periferica che ha determinato l'interruzione



Interruzioni nel M68000

- Identificare la periferica significa, essenzialmente, poter individuare la corretta ISR che può gestirla
- Il processore gestisce una **tabella delle interruzioni**, contenente gli *indirizzi* di tutte le ISR presenti
- Il **vettore** è usato come *indice* nella tabella per individuare la **ISR** corretta e gestire la periferica che ha causato l'interruzione



Interruzioni nel M68000

- La tabella delle interruzioni è posta in memoria a partire dall'indirizzo 0
- Contiene 256 locazioni di 4 byte
- Ogni locazione contiene l'indirizzo dell'*entry-point* (prima istruzione) di una differente **ISR**
- E' quindi possibile avere 256 differenti ISR
  - le prime 24 hanno funzioni speciali
  - da 25 a 31: interruzioni autovettorizzate
  - da 32 a 47: trap
  - restanti: interrupt utilizzabili per l'interazione con periferiche generiche

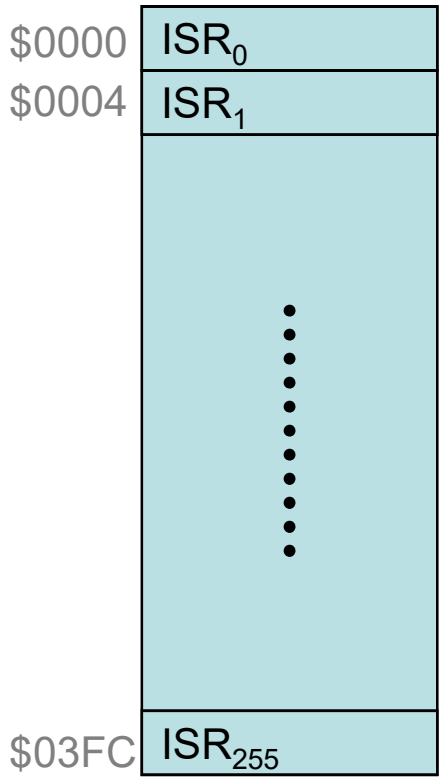


tabella delle interruzioni

M68000: tabella delle interruzioni

0	Reset (Ssp)
1	Reset (Pc)
16-23	Unassigned, Reserved
25	Level 1 Autovector
31	Level 7 Autovector
32-47	Trap #0-15 Instructions
64-255	User Device Interrupts

Interruzioni nel M68000

- Per avere interruzioni *particolarmente veloci* è possibile evitare il passaggio sul bus del vettore
- Un apposito segnale informa il processore che la periferica non presenterà il vettore
  - si parla in questo caso di ***interrupt autovettorizzate***
- La particolare **ISR** da chiamare è derivata direttamente dal livello di priorità, come segue:
  - Livello di priorità **1** → esegui la **ISR** numero **25**
  - Livello di priorità **2** → esegui la **ISR** numero **26**
  - . . . . . .
  - Livello di priorità **7** → esegui la **ISR** numero **31**

Interruzioni nel M68000

- Interruzioni autovettorizzate:
  - l'indirizzo all'interno della tabella delle interruzioni (numero di **ISR** moltiplicato per 4) è quindi derivato a partire dal livello di priorità, come segue:

(24 + livello di priorità) * 4 =

= 60_{HEX} + (livello di priorità) * 4 =

= *indirizzo della locazione nella tabella in cui è contenuto l'indirizzo dell'entry-point della ISR*

Interruzioni nel M68000