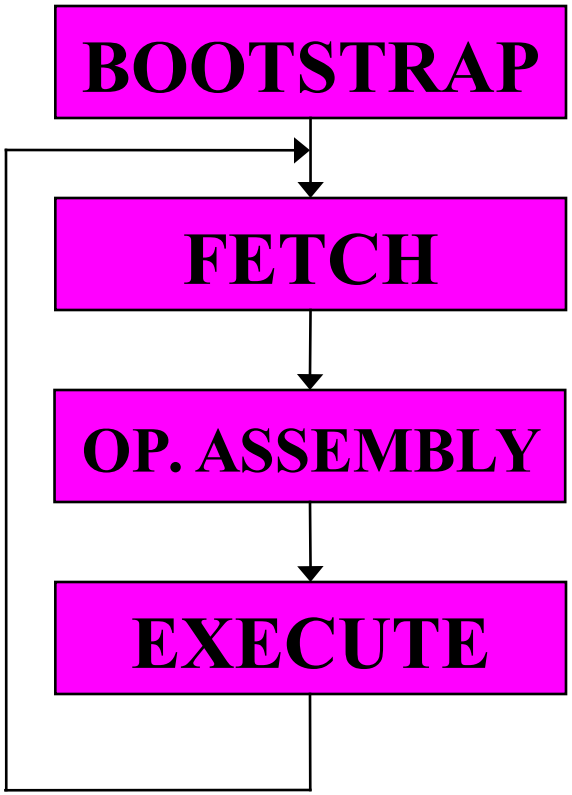
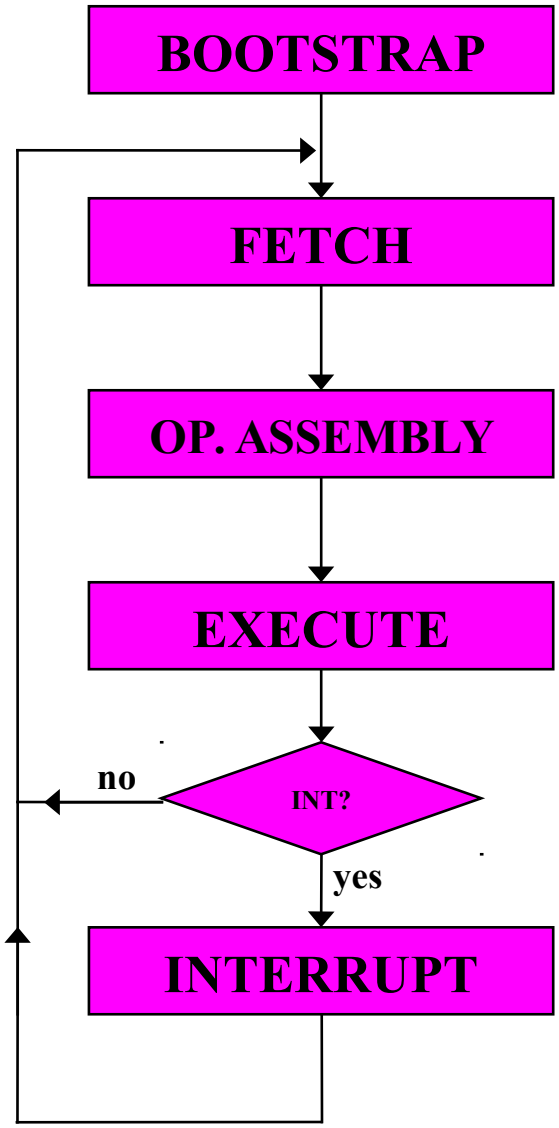
Calcolatori Elettronici
Il sistema delle interruzioni

Il ciclo del processore semplificato



- Se il ciclo del processore fosse effettivamente quello mostrato in figura, sorgerebbero alcuni problemi, come per esempio:
  - un'applicazione "prepotente" potrebbe impadronirsi della risorsa processore senza mai lasciarla
  - 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

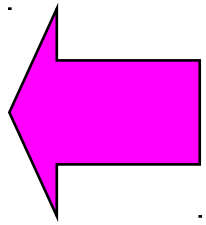
Il ciclo del processore esteso al meccanismo delle interruzioni



- La soluzione comunemente adottata consiste nel permettere al "supervisore" di prendere il controllo del processore al termine di ciascun ciclo
- 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

Innesco

```
while (true) {
  Fetch();
  Execute();
  CheckForInterrupt();
}
```



Se non succede niente di "eccezionale", non fa praticamente niente

La fase INTERRUPT (1/2)

INTERRUPT

- La fase di INTERRUPT viene eseguita nel caso in cui il segnale INT è asserito
- Questo evento è sintomatico del fatto che alcuni eventi sono “pendenti” e devono essere “serviti”
- Gli eventi possono essere di natura diversa e possono essere generati da diverse cause
- L’interruzione rappresenta il “servizio” che provvede a gestire questi eventi

La fase INTERRUPT (2/2)

INTERRUPT

- Durante questa fase del ciclo del processore, comunque, non viene eseguito un programma
- Per eseguire un programma (*software*), infatti, sarebbe necessario trovarsi all'interno del ciclo principale e muoversi tra le fasi di *fetch* ed *execute*
- Ciò che avviene nella fase di *interrupt* consiste invece in una serie di meccanismi *hardware* che "preparano" il processore a gestire l'interruzione

Eventi Eccezionali

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

Fasi

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

Il ripristino del programma

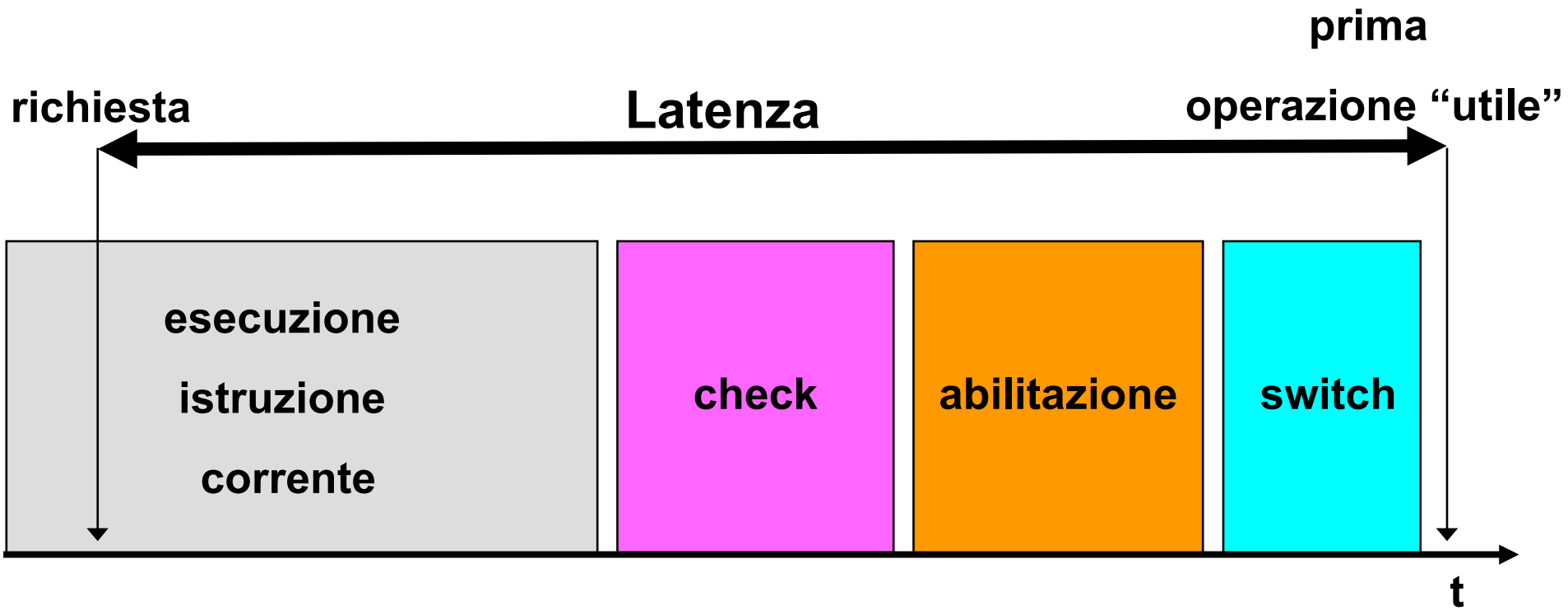
- Un' interruzione potrebbe eseguire un'elaborazione *B* completamente indipendente da quella *A* correntemente in corso sul processore, interrompendola
  - La gestione delle interruzioni deve quindi anche provvedere a mettere *A* in condizioni di continuare successivamente senza "accorgersi" di nulla
- Sorge la necessità di salvare (prima) e ripristinare (dopo) lo stato del programma che viene di volta in volta interrotto
- In questo modo *A* può continuare la sua elaborazione senza risentire in alcun modo del servizio dell'interruzione (a parte il ritardo dovuto al servizio dell'interruzione)
- Le informazioni che devono essere salvate e ripristinate comprendono di solito il PC, i flag dei codici di condizione e il contenuto di qualsiasi registro che sia usato sia dal programma che dalla routine di gestione dell'interruzione

Il salvataggio dello stato

- L'operazione di salvataggio può essere svolta in parte o completamente in *hardware* o in *software*
- Un' esigenza comune resta comunque quella di salvare lo stretto indispensabile poiché il salvataggio richiede trasferimenti di dati, eventualmente da e verso la memoria
- Data la frequenza con cui le interruzioni vengono prodotte, questo rappresenta quindi un carico aggiuntivo che deve essere ridotto al minimo
- 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*

Latenza di un'interruzione

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



Identificazione dei dispositivi (1/2)

- Se ci sono più dispositivi, il processore deve essere in grado di identificare il dispositivo che ha generato l'interruzione, poiché probabilmente diverse azioni dovranno essere intraprese a seconda del particolare dispositivo
- I dispositivi hanno una linea comune attraverso la quale segnalano richieste di interruzioni (INT)
- Quando INT è alto si pone il problema di identificare da quale dispositivo è partita la richiesta

Identificazione dei dispositivi (2/2)

- INT potrebbe alzarsi anche in seguito a richieste “contemporanee” di due o più dispositivi
- Esistono diverse soluzioni a questo problema
- Tutte le soluzioni impiegano un misto di hardware e di software
- Tutte le soluzioni dipendono fortemente sia dall’architettura del sistema che da quella del processore

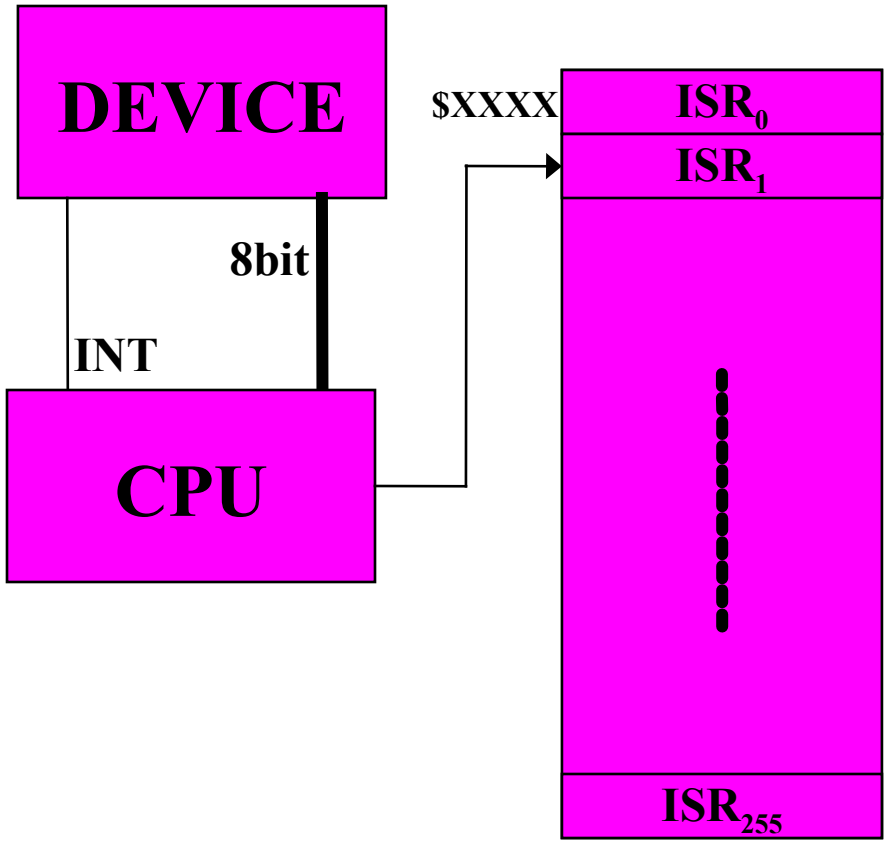
La soluzione a registri di stato

- Una possibile soluzione consiste nel dotare ogni dispositivo di un registro di stato
- Quando un dispositivo richiede un'interruzione, inizializza un bit nel registro di stato, il bit di richiesta di interrupt (Interrupt Request, IRQ)
- La procedura di servizio inizia interrogando tutti i dispositivi in un certo ordine e, non appena trova un bit alto, fa partire la corrispondente routine di interrupt
- Questa interrogazione ciclica (polling) è semplice da realizzare, ma ha lo svantaggio di richiedere un certo tempo per interrogare anche i dispositivi che non hanno invocato alcun servizio

Gli interrupt vettorizzati

- Un approccio alternativo consiste nel prevedere che sia il dispositivo stesso a fornire un proprio identificativo all'atto di una richiesta
- L'identificativo serve proprio a calcolare l'indirizzo della routine di interruzione che deve essere invocata, rendendo il meccanismo molto efficiente
- Il numero di bit utilizzati pone il limite massimo sul numero di diversi dispositivi che possono essere riconosciuti

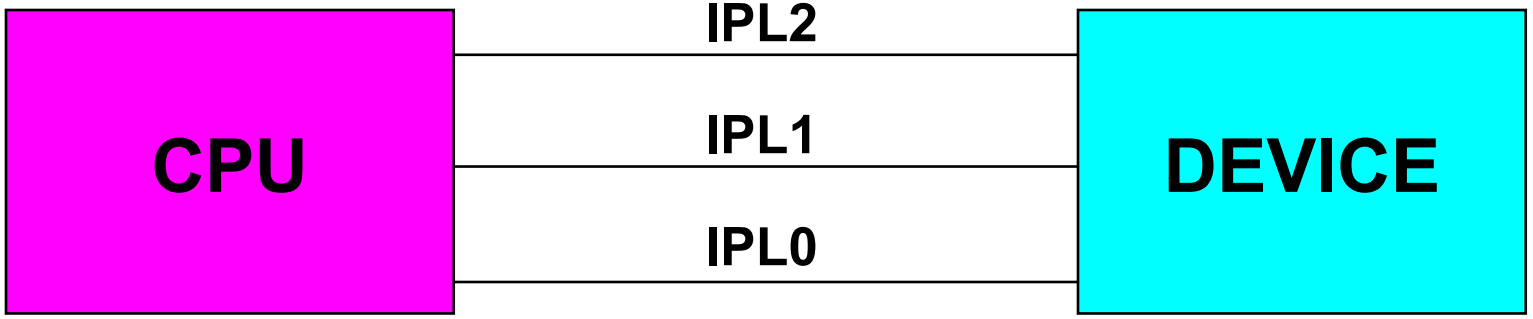
La soluzione del M68000



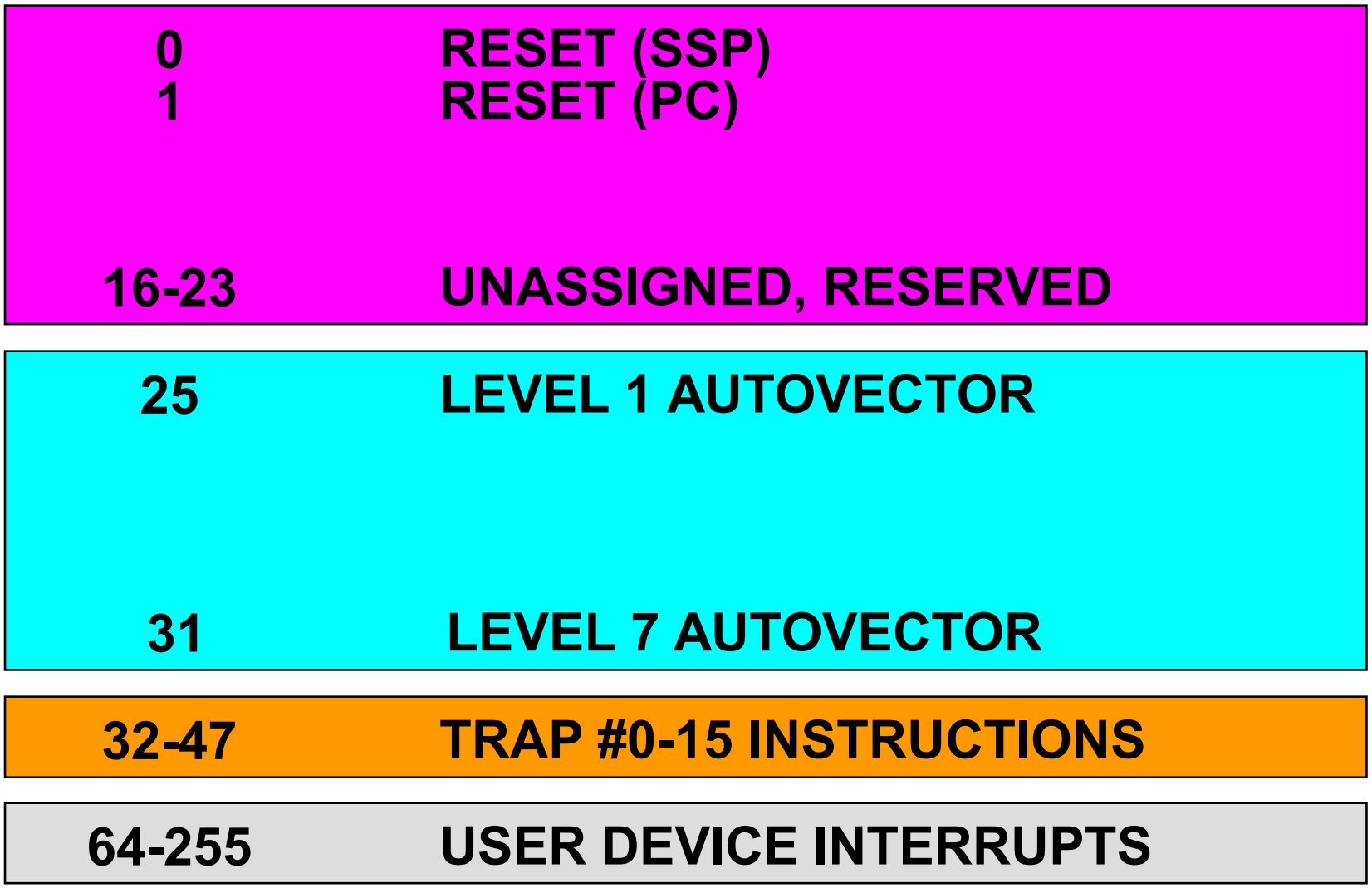
- Il processore M68000 utilizza il meccanismo degli interrupt vettorizzati
- In memoria sono presenti 256 locazioni consecutive dette *vettori di interrupt*
- Ciascuna di queste locazioni contiene l'indirizzo di una ISR
- Quando un dispositivo richiede un'interrupt, invia al processore un numero di 8 bit che rappresenta il *vettore di interrupt* da utilizzare

Gestione delle priorità

- Problemi:
  - Mascheramento
  - Abilitazione
- Soluzione del 68K:
  - Interrupt Priority Level
  - Processor Priority Level
  - Le interruzioni a priorità 7 non sono mascherabili



Exception Vector Table



Servizio mediante autovettore