Interruzioni

Nuovo Corso di Calcolatori Elettronici I

Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"

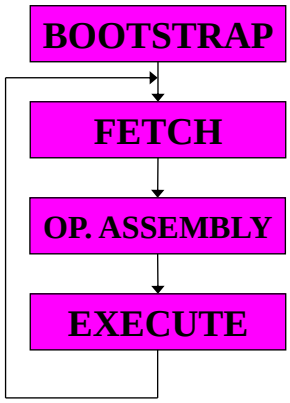
Roadmap

- Inquadramento del problema
- Il meccanismo e le cause di Interruzione
- Procedura di servizio dell'Interruzione
- Abilitazione delle Interruzioni
- Fasi
- Latenza
- Gestione delle priorità
- Identificazione di dispositivi

Inquadramento del problema

- Sistema delle interruzioni:
  - Infrastruttura per la gestione di specifici eventi
- Primo grado di libertà:
  - Esistono molti processori, con diverse organizzazioni
- Secondo grado di libertà:
  - Esistono molti circuiti hardware dedicati, con diverse caratteristiche, che possono collaborare con il processore nella gestione delle interruzioni
- Problema:
  - Fare riferimento ad un modello GENERALE
  - Evitare che tale modello sia GENERICO
    - ⇨ Ove necessario, sarà preso in esame un processore reale (MC 68K)

Il ciclo del processore semplificato



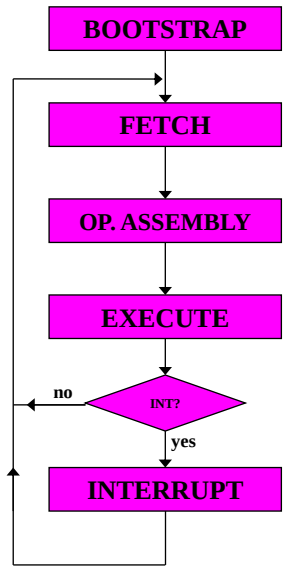
- Se il ciclo del processore fosse effettivamente quello mostrato in figura, sorgerebbero alcuni problemi, come per esempio:
  - un'applicazione "prepotente" potrebbe impadronirsi della risorsa processore senza mai lasciarla
  - non ci sarebbe modo di rimuovere forzatamente un'applicazione che entri per errore in un ciclo infinito
  - il sistema operativo, in generale, avrebbe un controllo limitato sul sistema

INTERRUPT : esempio

- In un programma che prevede operazioni di I/O, nel momento in cui inizia la comunicazione col dispositivo il programma entra in un ciclo di attesa nel quale controlla ripetutamente lo stato del dispositivo.
- Durante tale periodo il processore non esegue alcuna operazione utile.
- In linea di principio il processore può passare a fare "*altre cose*", ma perché ciò sia realizzabile è necessario un meccanismo per far sì che il dispositivo avverta il processore quando è pronto.
- Questo segnale è detto ***interrupt***.

- Questo esempio mette in mostra come (eliminando il controllo continuo da parte del processore sul dispositivo) il processore mentre attende il compimento delle operazioni di I/O può anche eseguire altre funzioni: l'utilizzo degli interrupt permette di eliminare i periodi di attesa.

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();
  Decode();
  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

"Procedura" di Servizio dell'Interrupt

- ISR: Interrupt Service Routine
- Una procedura di servizio dell'interrupt può essere vista come una normale procedura (sottoprogramma) ma …
- …una differenza importante è che un sottoprogramma esegue una funzione richiesta dal programma chiamante, mentre in generale una procedura di servizio degli interrupt può non avere nulla in comune con il programma che è in esecuzione al momento della ricezione del segnale di interrupt.

Le cause di interruzione (1/2)

- 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)
- Hardware Interrupts
  - Permettono di gestire richieste di "attenzione" da parte di dispositivi (tipicamente di I/O)
  - Sono eventi asincroni (rispetto all'elaborazione)

Le cause di interruzione (2/2)

- Reset
  - ***interruzione per riportare l'intero sistema*** in uno stato noto (reset);
- Traps
  - ***interruzione per errori nel programma*** correntemente eseguito (p. es. overflow, esecuzione di un'istruzione inesistente o privilegiata, etc.);
  - ***interruzioni programmate (software interrupt)*** generate da un programma che voglia accedere una risorsa condivisa (es. periferiche di I/O) e per questo chiede la mediazione del sistema operativo. Sono le uniche interruzioni ad essere *sincrone* con il programma correntemente in esecuzione.
- Hardware Interrupts
  - ***interruzione per guasti*** al sistema rivelati da apposite sonde;
  - ***interruzione periodica*** (p.es. ogni 10ms) per permettere al sistema operativo di computare il tempo speso da una applicazione e di cedere eventualmente la risorsa processore ad un'altra applicazione (multiprogrammazione);
  - ***interruzione di I/O:*** una periferica di I/O che informa su un suo particolare stato (p.es. pronta a ricevere dati) al fine di sincronizzarsi con il processore;

Abilitazione delle interruzioni

Abilitazione delle interruzioni

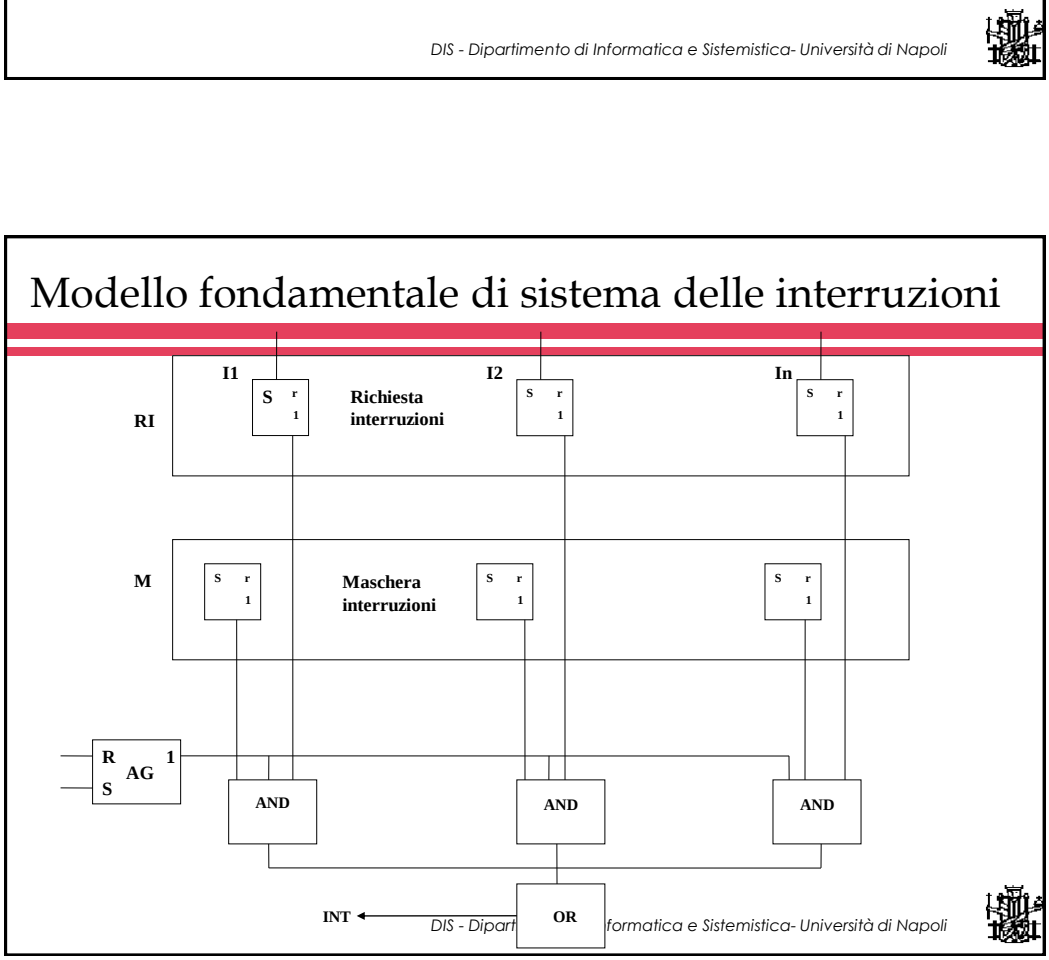
- Una ***causa di interruzione*** non provoca di per sé una interruzione ma soltanto una ***richiesta di interruzione***.
- Affinché l'interruzione sia attiva è necessario che essa sia abilitata:
  - in questo modo è possibile implementare particolari strategie per le quali si inibiscono alcune tipologie di interruzioni.
- Tale filosofia porta al ***Modello fondamentale di sistema delle interruzioni***.

Modello fondamentale di sistema delle interruzioni

- Il modello si basa su due registri speciali ed un flip-flop:
  - Registro ***Richieste di Interruzioni RI***, che memorizza le richieste di interruzioni;
  - Registro ***Maschera delle interruzioni M***, che abilita singolarmente le richieste
  - ***Flip-Flop di Abilitazione Generale, AG***, che abilita il sistema nel suo complesso.
- Ciascun evento I (causa di interruzione) posiziona il flip-flop RI_i del registro RI (richieste) ed è abilitato dal corrispondente bit M_i della maschera. Tutto il sistema è poi abilitato da AG. Il segnale INT è pertanto:

$$INT = AG \cdot \sum_i RI_i \cdot M_i$$

ove la sommatoria è da intendersi in senso booleano.



Modello fondamentale di sistema delle interruzioni

Modello fondamentale di sistema delle interruzioni

$$INT = AG \cdot \sum_i RI_i \cdot M_i$$

- Un segnale di interruzione è presente (INT=1) se si verifica che:
  - Tutto il sistema di interruzione è abilitato (AG=1)
  - La i-esima causa ha richiesto l'interruzione (RI_i=1)
  - Tale interruzione è abilitata dalla maschera (M_i=1)
- Sia i flip-flop di M sia AG sono posizionati da apposite istruzioni del linguaggio macchina:
  - Disabilita sistema delle interruzioni
  - Abilita sistema delle interruzioni
  - Disabilita (o abilita) la causa i
  - Assegna il valore m alla maschera delle interruzioni (M:=m)
- L'insieme delle azioni elaborative svolte via hardware dal processore nella fase di interrupt e via software dalla ISR viene detto ***processo delle interruzioni***.

Le Fasi di una interruzione

Sequenza di eventi durante un Interrupt (dall'esterno)

- Il Dispositivo genera il segnale;
- il Processore interrompe il programma in esecuzione;
- il Dispositivo viene informato che l'interrupt è stato ricevuto (ack);
- viene eseguita la procedura (ISR);
- si ripristina il Programma originale.

Sequenza di eventi durante un Interrupt (dall'interno)

- 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)
- 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: fase "hardware"

- Nel modello fondamentale si considera ridotto al minimo l'intervento *hardware* (quest'ultimo deve realizzare il salto alla ISR ed il salvataggio del valore corrente di PC per consentire la futura ripresa del programma interrotto):
  - Detto SAVE il registro in cui viene memorizzato PC e START l'indirizzo d'inizio della ISR ($START_i$ nel caso di interrupt vettorizzati) si ha:

    SAVE:=PC;

    PC:=START;

    AG:=0;
- La disabilitazione delle interruzioni viene effettuata al fine di evitare, che prima ancora che il processo delle interruzioni inizi, possa innestarsi un'ulteriore interruzione.

Il salvataggio dello stato: fase "software"

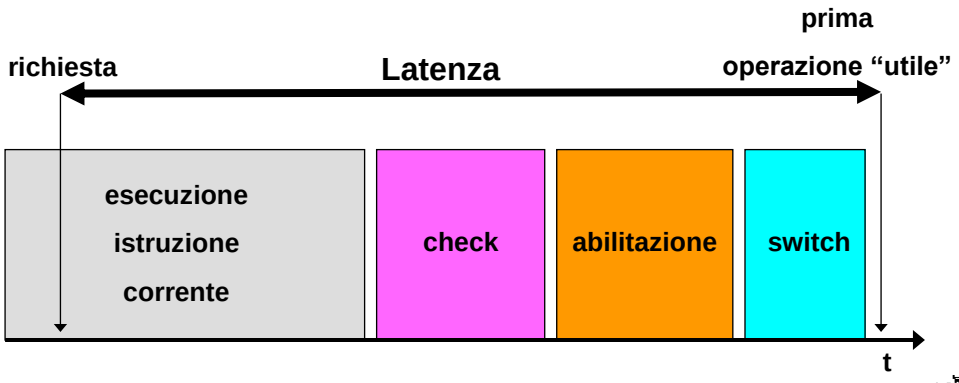
- Le altre informazioni necessarie alla corretta ripresa del programma sospeso vengono salvate dalle istruzioni che si trovano all'inizio della ISR e ripristinate dalle istruzioni che stanno alla fine.
- La parte di "salvataggio/ripristino soft*ware"* realizzato dalla ISR riguarda:
  - Salvataggio dei registri per la ripresa del programma interrotto
  - Servire l'interruzione
  - Ripristinare lo stato dei registri
  - Riprendere il programma interrotto oppure saltare all'esecuzione di un altro programma

Il *salvataggio dello stato* e *latenza di interrupt*

- 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



Nesting delle Interruzioni

- Durante il servizio di una interruzione è possibile (in linea di principio) che divenga attiva una nuova interruzione: viene allora interrotta l'interruzione e servita la nuova causa.
- Tale situazione prende il nome di ***Nesting delle Interruzioni*** e viene gestita con tecniche LIFO: l'ultimo programma ad essere stato interrotto è il primo ad essere ripreso.
- In questo caso si pone il problema della ***priorità***:
  - Quale tra due cause verificatesi simultaneamente deve essere servita per prima.
  - Quale causa può interrompere il servizio di un'altra causa.

Identificazione dei Dispositivi

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

Identificazione dei dispositivi

- Polling
  - Si interroga il registro di stato di tutti i dispositivi.
- Interrupt vettorizzato
  - Il Dispositivo manda un identificativo in risposta al segnale di ack.
- Interrupt autovettorizzato
  - L'indice viene associato direttamente al segnale di interruzione
- Linee Separate
  - Non c'è problema di identificazione

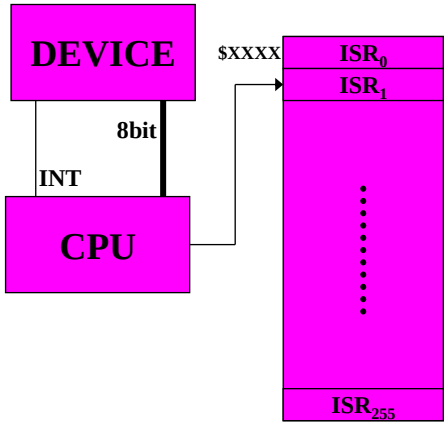
La soluzione a registri di stato (polling)

- 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

Interrupt Annidati: priorità

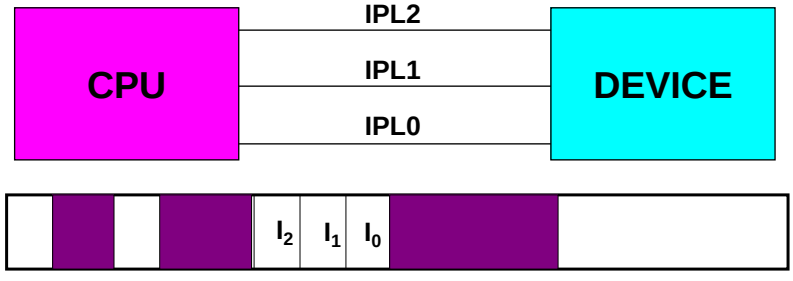
- Una richiesta di interrupt proveniente da un dispositivo ad alta priorità deve essere accettata anche mentre il processore sta servendo un'altra richiesta inviata da un dispositivo a bassa priorità (es. clock).
- Durante l'esecuzione di una ISR vengono accettate le richieste di interrupt solo da dispositivi a priorità più alta rispetto a quella corrente.
- La priorità corrente è mantenuta nel processore: il livello di priorità del processore rappresenta la priorità del programma che è in fase di esecuzione.
- Quando il processore accetta richieste a priorità più elevata imposta la sua priorità al valore di priorità della richiesta appena accettata.
- Nel 68000 il valore corrente di priorità è registrato nei bit priorità dell'SR

Richieste simultanee e priorità

- Polling
  - Non sussiste il problema sulla simultaneità. La priorità è data dall'ordine con cui viene effettuato il polling.
- Vettorizzato
  - La linea di INTR (request) può essere su più bit per gestire un primo livello di priorità
  - La linea di INTA (acknowledge) è realizzata con un collegamento dei dispositivi in *daisy-chain*. Risolve il problema della simultaneità e realizza un livello di priorità basato sui collegamenti hardware.
- Linee separate
  - Il problema non sussiste ma è una soluzione costosa. In alternativa si possono avere linee separate per gruppi di dispositivi collegati in daisy-chain.

Gestione delle priorità

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



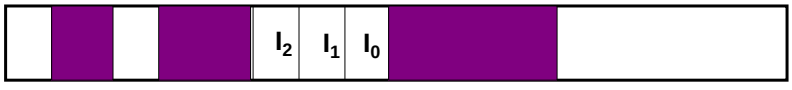
Exception Vector Table

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

Servizio mediante autovettore

$64_{HEX}=100_{DEC}$ → 25 LEVEL 1 AUTOVECTOR

$7C_{HEX}=124_{DEC}$ → 31 LEVEL 7 AUTOVECTOR



$(60 + 4 * n)_{HEX}$