LABORATORIO DI ARCHITETTURA DEI CALCOLATORI

lezione n° 19

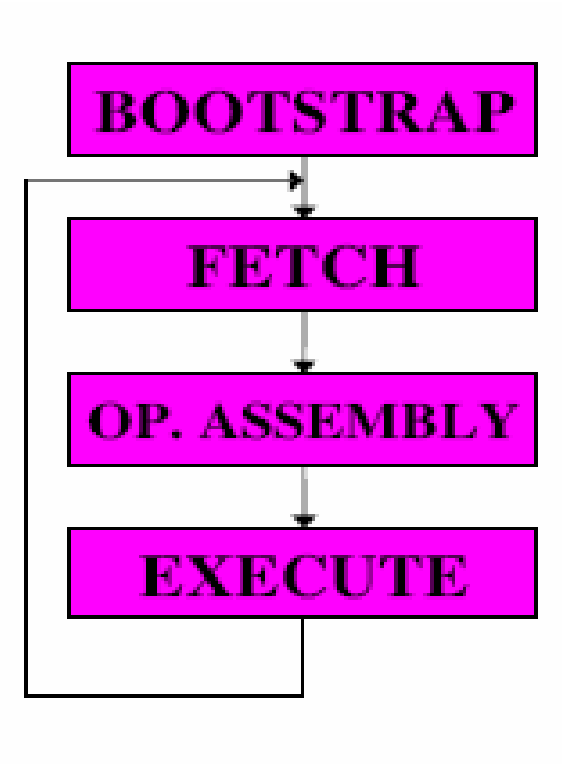
Prof. Rosario Cerbone

rosario.cerbone@libero.it

http://digilander.libero.it/rosario.cerbone

a.a. 2005-2006

Interrupt



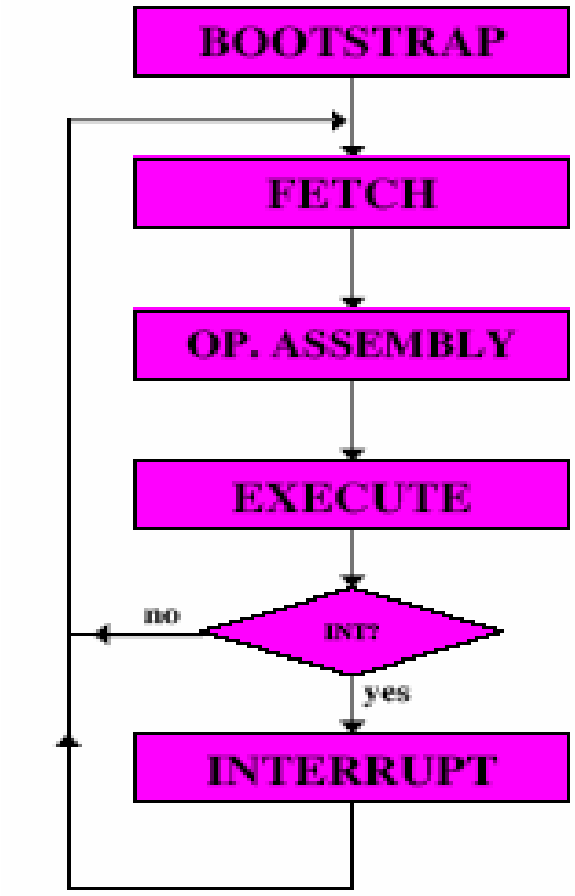
Se il ciclo del processore fosse effettivamente quello mostrato in figura, sorgerebbero alcuni problemi:

- 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

- 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 potrebbe 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***.
- 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.

Interrupt



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 in corso.

In assenza di tali eventi, l’elaborazione procede nella maniera consueta.

Il gestore delle *eccezioni* (*exception handler*) permette di gestire in maniera flessibile:
- la gestione di *interruzioni interne* ed e*sterne*
- il recupero dai guasti di sistema *catastrofici*
- altri *eventi* non associati alla normale esecuzione del programma.

Interrupt

- La fase di INTERRUPT viene eseguita nel caso in cui il segnale INT è alto.
- 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: ciascuna delle cause di interruzione richiede al sistema specifiche azioni elaborative (conteggio del tempo, segnalare un errore, rendere disponibile una risorsa, …).
- Al termine del servizio, il programma interrotto viene ripreso, se tale ripresa è compatibile con la stessa interruzione.

Interrupt

- Durante questa fase del ciclo del processore, comunque,non viene eseguito un programma. Se INT è alto, si innesca il sistema per saltare alla particolare ISR che verrà eseguita (come tutti i programmi) nel normale ciclo del processore.
- 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.

Interrupt

- Una procedura di servizio dell'interrupt può essere vista come una normale procedura (sottoprogramma).
- 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.

Interrupt. Le cause di interruzione

- Una interruzione può essere causata da necessità di natura diversa. Per esempio:
- ***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;
- ***interruzione per errori nel programma*** correntemente eseguito (p. es. overflow, esecuzione di un'istruzione inesistente o privilegiata, etc.); tali interruzioni vengono anche chiamate *traps*;
- ***interruzione per guasti*** al sistema rivelati da apposite sonde;
- ***interruzione per riportare l'intero sistema*** in uno stato noto (reset);
- ***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.

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.

Interrupt

- **Il processore ha 3 ingressi (ILP0, IPL1 e IPL2) per le richieste di interruzione: la loro decodifica specifica il livello di richiesta di interruzione. Il livello 7 è il livello a più alta priorità, mentre il livello 0 non corrisponde a nessuna richiesta di interrupt.**
- **Dunque il 68000 fornisce 7 livelli di interrupt esterni.**

Interrupt

- **Il registro di stato contiene 3 bit che indicano il livello massimo di interrupt mascherato.**
- **Il livello 7 è *l'interrupt non mascherabile* e non può essere mascherato da nessun valore della maschera di interrupt del registro di stato.**

Processamento dell'interrupt

- **Tutti gli interrupt che arrivano al 68000 sono *bufferizzati* internamente e resi *pendenti*; prima del servizio della richiesta il processore termina l'istruzione corrente.**
- **Il 68000 confronta il livello di richiesta di interruzione con il valore memorizzato nella maschera di interrupt del *registro di stato*. Se la priorità dell'interrupt pendente è minore o uguale alla priorità del processore, la richiesta di interrupt rimane pendente ed il processore esegue la successiva istruzione in sequenza.**

Processamento dell'interrupt

- **Il livello di interrupt 7 (*interrupt non mascherabile*) è sempre processato, indipendentemente dal valore della maschera di interrupt.**
- **Quando il processore ha preso la decisione di processare una richiesta di interruzione, comincia la normale sequenza di operazioni di gestione dell'interruzione.**
- **La maschera di interruzione del registro di stato è aggiornata prima dell'inizio del processamento dell'interrupt. Il livello della richiesta di interruzione che è in corso di servizio è copiato nella maschera. In questo modo l'interrupt servito non può essere interrotto se non da una nuova richiesta avente priorità più alta.**

Processamento dell'interrupt

- **Si supponga, ad esempio, che la maschera abbia livello 3. Se arriva una richiesta a livello 5, tale richiesta viene servita e la maschera viene caricata con il valore 5. Una successiva richiesta a livello 4 rimane pendente.**
- **Quando la richiesta a livello 5 termina, attraverso l'istruzione di ritorno dall'eccezione viene ripristinato il vecchio valore della maschera (pari a 3) e dunque la richiesta pendente (di livello 4) può essere servita.**

Tipi di interrupt

- **Esistono due modi con cui un dispositivo esterno che fa richiesta di interrupt può fornire il suo numero di vettore alla CPU:**
  - **un modo *vettorizzato***
  - **un modo *autovettorizzato*.**

Vettore delle eccezioni

- **Per ogni eccezione è riservato un spazio di memoria pari a due word contenenti l'indirizzo della procedura di gestione dell'eccezione.**
- **Il vettore delle eccezione è memorizzato in una tabella di 512 word che va dall'indirizzo $000000 all'indirizzo $0003FF, contenente l'indirizzo delle procedure di gestione delle eccezioni.**

Vettore delle eccezioni

Numero	*Tipo di eccezione*	*Numero*	*Tipo di eccezione*
0	Reset - `SSP` iniziale	24	Interrupt spurio
-	Reset - `PC` iniziale	25	Livello 1 interrupt autovector
2	Errore sul Bus	26	Livello 2 interrupt autovector
3	Errore di indirizzo	27	Livello 3 interrupt autovector
4	Istruzione illegale	28	Livello 4 interrupt autovector
5	Divisione per zero	29	Livello 5 interrupt autovector
6	Istruzione `CHK`	30	Livello 6 interrupt autovector
7	Istruzione `TRAPV`	31	Livello 7 interrupt autovector
8	Violazione di privilegio	32	`TRAP` #0
9	Trace	33	`TRAP` #1
10	Emulatore di linea 1010	...	...
11	Emulatore di linea 1111	47	`TRAP` #15
12	Non assegnato - riservato	48	Non assegnato - riservato
13	Non assegnato - riservato	...	...
14	Non assegnato - riservato	63	Non assegnato - riservato
15	Interrupt non inizializzato	64	User Interrupt
16	Non assegnato - riservato	...	...
...	...	254	User Interrupt
23	Non assegnato - riservato	255	User Interrupt

Interrupt vettorizzato

- **Dopo aver completato l'istruzione corrente il processore:**
  - **salva nello stack la word meno significativa del program counter**
  - **esegue un ciclo di *interrupt acknowledge* (*IACK cycle*). Durante tale ciclo il 68000 riceve dal dispositivo interrompente il numero del vettore.**

Interrupt autovettorizzato

- **Dispositivi periferici originariamente progettati per microprocessori ad 8 bit non sono predisposti al ciclo di *interrupt acknowledge*, ossia non sono in grado di rispondere sul data bus con l'appropriato numero di vettore.**
- **La soluzione a tale problema è fornita dal 68000 con il meccanismo di *interrupt autovettorizzato*. Questo si implementa facendo asserire dal periferico il segnale VPA anzichè il segnale DTACK durante il ciclo di *interrupt acknowledge*.**

Interrupt autovettorizzato

- **Il segnale VPA (*valid peripheral address*), quando asserito, informa il 68000 che il ciclo di memoria corrente rispetta le regole di accesso di memoria della famiglia 6800.**
- **Se il ciclo di bus corrente è un ciclo di interrupt acknowledge il 68000 esegue un *ciclo di read spurio*: il dispositivo interrompente non scrive il numero di vettore sul data bus.**
- **Poichè il dispositivo non genera il numero di vettore, è il processore che lo genera internamente in modo automatico.**

Interrupt autovettorizzato

- **Il 68000 riserva i numeri di vettore da 25 a 31 per le operazioni di autovettorizzazione. A ciascuno di essi è associata una delle 7 richieste da IRQ1 a IRQ7.**
- **Ad esempio se viene inoltrata la richiesta IRQ2 e successivamente viene asserito il segnale VPA durante il ciclo di interrupt acknowledge, il numero di vettore 26 è generato dal 68000 e viene eseguita la procedura di gestione dell'interrupt che si trova all'indirizzo letto nella locazione $000068.**