

LEZIONE 16/10/08 (slide interrupt 7-20)

Gli interrupt (slide 7-19)

Anche quando un dispositivo è gestito con interrupt il tempo di latenza ed il throughput rivestono un ruolo importante per le sue prestazioni.

Nell'analizzare gli interrupt si utilizza spesso il concetto di *tempo di latenza* (definito come tempo fra la richiesta dell'interrupt, generato dall'interfaccia, e l'esecuzione dell'operazione di I/O):

- T_i , tempo necessario a completare l'istruzione corrente, gli interrupt non sono sincronizzati con il clock della CPU e quindi possono arrivare in un qualsiasi momento. Se si analizza la situazione considerando il caso peggiore bisogna considerare questo tempo pari a quello di esecuzione di un'istruzione.
- T_c , tempo di attivazione della routine di gestione dell'interrupt, include il salvataggio dello stato corrente e l'aggiornamento del program counter alla locazione della procedura di gestione dell'interrupt. Queste operazioni richiedono l'accesso alla memoria e quindi aumentano il tempo di latenza.
- T_r , tempo fra l'inizio della routine e l'istruzione di trasferimento dati. Non è infatti necessario che la procedura di gestione dell'interrupt legga subito il dato dall'interfaccia della periferica, ma potrebbe prima eseguire prima una manciata di istruzioni di inizializzazione o di preparazione per la lettura del dato.

Il *transfer rate* (abbreviato TR) è la frequenza con cui arrivano i blocchi di dati provenienti dalla periferica, ipotizzando che ogni arrivo generi un interrupt. Il tempo di latenza dell'interrupt deve essere minore o uguale all'inverso del transfer rate, altrimenti vengono persi dei dati poiché la CPU sta ancora gestendo un interrupt quando arriva il successivo. Naturalmente, nella pratica, il tempo di latenza deve essere decisamente minore dell'inverso del transfer rate, di modo che la CPU possa svolgere per la maggior parte del tempo lavoro utile, e non gestione degli interrupt.

L'architettura 80x86 salva i puntatori delle procedure di gestione degli interrupt in una tabella chiamata *interrupt vector table* (abbreviata IVT) o *interrupt descriptor table* (abbreviata IDT). Sebbene questa tabella sia differente tra la modalità di funzionamento real e protected della CPU, i concetti alla base del suo funzionamento sono gli stessi. L'IVT contiene 256 elementi che rappresentano i 256 diversi interrupt gestibili dal processore 8086, ogni elemento occupa 32 bit, 16 per il registro di segmento e 16 per l'indirizzo a cui saltare nel segmento. In modalità reale l'IVT occupa lo spazio di memoria compreso tra gli indirizzi 0x0000 e 0x03FE.

Non tutti gli interrupt hanno la stessa priorità ma sono invece suddivisi in diverse categorie, inoltre il numero di un interrupt all'interno dell'IVT ne indica anche la priorità: gli interrupt con indice inferiore hanno priorità maggiore. In ordine decrescente di priorità, gli interrupt possono essere classificati nelle seguenti categorie: interrupt interni, interrupt non mascherabili, interrupt software, interrupt hardware. Un interrupt di una categoria di priorità maggiore può interromperne uno di priorità minore.

Quando si assegnano i diversi interrupt alle relative condizioni hardware generate dalle periferiche bisogna tenere conto del fatto che non tutte le periferiche operano alla stessa velocità e ogni singolo hardware ha dei limiti sul transfer rate e sul tempo di latenza. In generale si adotta la soluzione di dare una maggiore priorità agli interrupt di periferiche veloci e una priorità minore alle periferiche più lente, pur tenendo in conto che anche le periferiche lente hanno dei tempi massimi di attesa che

non devono essere superati (non si deve quindi avere il fenomeno di *starvation*, dove l'interrupt ad alta priorità passa sempre prima finendo col non far mai eseguire quello a bassa priorità).

L'architettura 80x86 prevede 4 diverse tipologie di interrupt che si distinguono per il modo in cui vengono generate, ma non per il modo in cui vengono gestite: la gestione prevede sempre l'uso della IVT e il salto ad una procedura di gestione dell'interrupt. Le tipologie di interrupt, ordinate per priorità di gestione decrescente, sono le seguenti:

- Interni, trap. Sono generati internamente dal processore quando si verifica una situazione anomala o che richiede una gestione durante l'esecuzione di un'istruzione. Un esempio è quando viene eseguita una divisione per zero, oppure quando si verifica un overflow durante un'operazione matematica. Il loro valore corrispondente nella IVT è stabilito dal progettista hardware del processore e non è modificabile, ad esempio quando si verifica una condizione di divisione per zero il processore 8086 salta alla procedura di gestione degli interrupt di indice 0 nella tabella IVT.
- Esterno non mascherabile (NMI). Si tratta di interrupt che non possono essere mascherati con le normali procedure offerte dal processore (uso delle istruzioni CLI e STI). E' possibile mascherarli operando manualmente sulla macchina scollegando l'hardware che li genera.
- Esterno mascherabile. Sono interrupt generati da periferiche esterne al processore e possono essere disabilitati, se necessario. E' possibile programmare via software da quali indici della IVT devono essere gestiti questo tipo di interrupt, questa operazione viene eseguita programmando il chip 8259.
- Software. Si tratta di interrupt sincroni generati eseguendo una particolare istruzione del processore (l'istruzione INT) che provvede ad interrompere l'esecuzione del flusso di istruzioni e a passare il controllo alla procedura di interrupt selezionata, esattamente come se l'interrupt fosse stato generato dall'esterno. Sono spesso utilizzati per richiamare delle funzioni offerte dal BIOS o dal sistema operativo; in questo senso l'istruzione INT n assomiglia ad un'istruzione CALL che salta all'indirizzo contenuto alla posizione n dell'IVT.

Gli interrupt possono essere mascherati utilizzando il flag IF (interrupt flag) del processore, questo flag può essere settato o azzerato utilizzando le istruzioni CLI (clear interrupt) e STI (set interrupt).

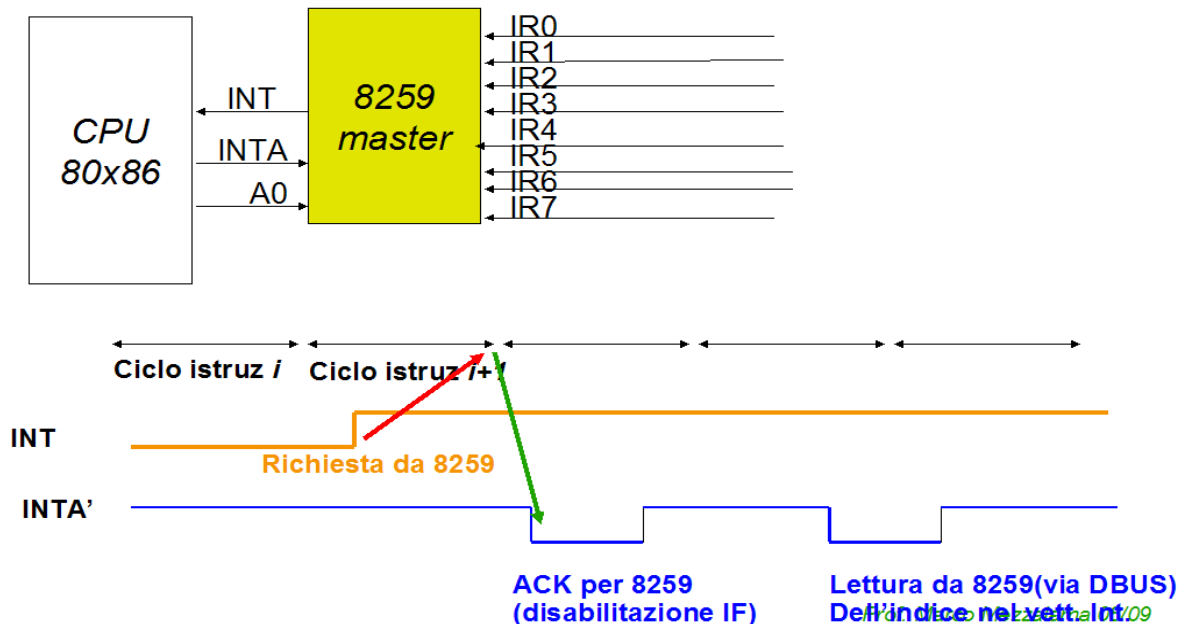
All'avvio del PC il BIOS provvede subito a caricare i puntatori nell'IVT alle principali e fondamentali procedure di gestione degli interrupt (che si trovano per ora nella ROM del BIOS). Una volta caricate le funzioni di gestione degli interrupt più importanti il BIOS provvede ad inserire ad alcuni indici specifici i puntatori alle funzioni offerte (come la gestione del disco, dello schermo, etc.).

Quando il BIOS ha terminato le attività di inizializzazione procede al caricamento del loader del sistema operativo (letto dal boot sector del disco) così da potergli passare il controllo. E' comune nei moderni sistemi operativi che le funzioni offerte dal BIOS per il controllo dell'hardware non siano adeguate, in questi casi il sistema operativo può sostituirsi al BIOS andando a modificare l'indirizzo nell'IVT relativo alle procedure di gestione dell'hardware. La flessibilità di una tabella di indirizzi degli interrupt in questo caso risulta molto utile poiché si può cambiare la procedura di gestione senza dover modificare in alcun modo il codice che la richiama (cioè senza dover ricompilare i programmi che la utilizzano).

Protocollo di interrupt (slide 21-25)

Quando un periferico invia una richiesta di interrupt (IR_n), il PIC (8259) master invia un segnale di INT al processore. Non appena termina l'esecuzione corrente, il processore si predispone a servire la richiesta di interrupt ed invia un segnale di INTA al (o ai) PIC.

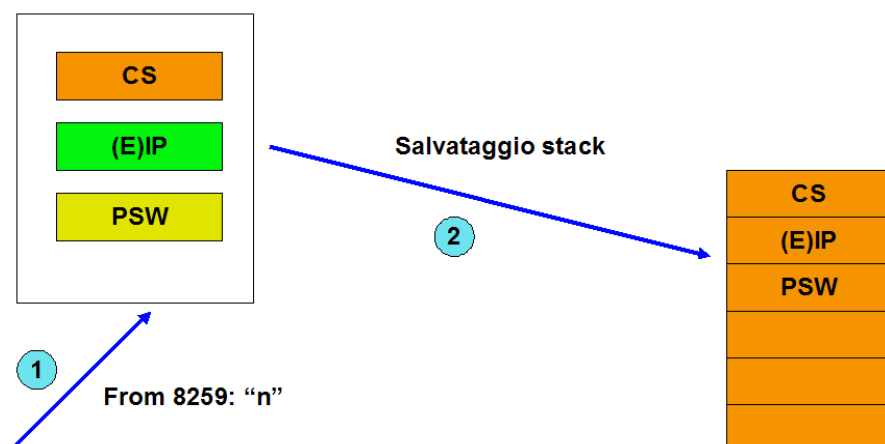
Protocollo di Interrupt(tempistica)



A questo punto il PIC seleziona la richiesta a priorità più alta ed attende un secondo INTA da parte della CPU.

Il PIC, ricevuto il secondo segnale di INTA, invia alla CPU sul data-bus un indice n (1 byte), relativo al dispositivo che ha fatto la richiesta, necessario per attivare la corrispondente routine di interruzione. Si tratta infatti dell'indice nella IVT a cui saltare.

Il processore, quindi, utilizzerà tale indice per accedere alla posizione della vector table relativa alla routine di interruzione in questione.



Il valore del program counter viene aggiornato, quindi, al nuovo valore memorizzato nella vector table. Prima di fare ciò è necessario salvare nello stack indirizzo di ritorno e flag di stato (PSW) nello stack.

