1 Lezione del 10-03-25

Torniamo sull'argomento delle interruzioni, specificando il modo in cui dobbiamo definire dei *gestori* per ogni interruzione.

Il calcolatore visto finora dispone di 4 interfacce:

- L'interfaccia *tastiera*, letta finora in controllo di programma, valutando la validità di un bit FI sul registro di stato;
- L'interfaccia *timer*, dotata di 3 singoli timer, di cui abbiamo detto il primo viene usato per generare interruzioni, il secondo non è più usato, e il terzo e connesso al *beeper speaker*;
- L'interfaccia a blocchi per *hard disk*, che accede ad un drive pilotando in base al suo stato un registro di stato (per noi era utile ad implementare la funzione di attesa del drive a controllo programma `wait_for_br()`).

Ignoriamo, per adesso, il video. Ognuna di queste interfacce può trarre beneficio dalla presenza di interruzioni:

- La tastiera potrebbe *avvertirci* dei nuovi tasti premuti, anziché costringerci a controllari;
- Il timer ci deve avvisare, al termine del conteggio del timer 0, attraverso un interruzione;
- L'hard disk, come la tastiera, ci può avvisare con un interruzione quando è pronto ad una nuova scrittura.

Questo comportamento, delle cosiddette **interruzioni esterne**, è definito nella macchina studiata dal **controllore delle interruzioni**, che è l'Intel **APIC** (*Advanced Progammable Interruption Controller*). Questo scansiona periodicamente tutte le linee generatrici di interruzione ottenute dalle varie interfacce, e invia le interruzioni corrispondenti, una per volta, alla CPU.

Chiaramente, si rende necessario specificare un **tipo di interruzione**, su 8 bit (per 256 tipi) per ogni interruzione lanciata. A questo punto, l'APIC fornirà semplicemente la possibilità di assegnare un tipo di interruzione diverso ad ogni piedino di ingresso dall interfacce, in modo che si possa assegnare ad ogni interruzione la routine di gestione più adatta.

La comunicazione fra CPU e APIC viene effettuata attraverso un *handshake* su due linee, **INTR** (*Interrupt Request*) e **INTA** (*Interrupt Acknowledge*), che comporta anche una lettura da parte della CPU di quanto l'APIC metterà sul bus (cioè il tipo di interruzione).

A questo punto, le routine vere e proprie verrano definite in una **IDT**, (*Interrupt Descriptor Table*), contenente in sequenza gli indirizzi delle prime istruzioni di ogni routine per ogni tipo di interruzione, e specificata a partire da un certo indirizzo indicato nel registro **IDTR**.

Come abbiamo visto, la reazione o meno della CPU ad una interuzione è data dall'attivazione del flag IF. Nel caso si passi effettivamente ad eseguire l'interruzione, ricordiamo che sia l'IP che lo stato dei flags verrà salvato in pila, e ripristinato a fine routine attraverso l'istruzione `IRET`.

1.0.1 Rilevamento di interruzioni da parte dell'APIC

Potremmo chiederci come fa il controllore APIC a capire quando un'interfaccia sta richiedendo una nuova richiesta.

Un primo approccio potrebbe essere di non rileggere il piedino di ingresso di quell'interfaccia, al ottenimento e successivo invio alla CPU di un interruzione, fino alla segnalazione, sempre da parte della CPU, di avvenuta gestione dell'interruzione. Questo può essere effettuato dotando l'APIC di un opportuno registro (**EOI**, *End Of Interrupt*), che la CPU andrà a modificare conclusa la gestione dell'interruzione.

Altri 2 registri di interesse sono i seguenti, entrambi su 256 bit (un bit per ogni tipo di interruzione):

- **IRR** (*Interrupt Request Register*): indica con bit alti quali interruzioni sono state inviate dalle interfacce attualmente;
- **ISR** (*Interrupt Service Register*): indica con bit alti a quali interruzioni sta rispondendo il processore attualmente. In un processore single-threaded come quello che studiamo al più uno solo dei suoi bit sarà alto in un dato momento.

1.0.2 Priorità delle interruzioni

Possiamo chiederci come l'APIC si comporta in caso di più richieste concorrenti. Un idea potrebbe essere di assegnare una priorità ad ogni richiesta, e rispondere prima alle richieste di priorità più alta.

Ci rendiamo quindi conto che alcune richieste sono più importanti di altre: ad esempio, la pressione di un tasto su tastiera può essere ignorata, se ad esempio nel frattempo ci arriva la richiesta di interruzione da parte di un timer. La pressione del tasto non si ripeterà infatti in tempo utile, mentre il timer potrebbe inviarci nuove richieste mentre ancora non siamo pronti a riceverle, e continuerà a farlo a scadenze regolari (potremmo finire per gestire solo un sottoinsieme delle richieste che ci vengono effettivamente inviate).

Vediamo quindi nel dettaglio di come i codici delle interruzioni sono definiti. I primi 4 bit rappresentano la **classe di precedenza** dell'interruzione: a classi di precedenza maggiore abbiamo gestione prioritaria delle richieste di interruzione. Il trasferimento da IRR a ISR avverrà quindi prima per richieste di classe di precedenza più alta, e poi per quelle di classe di precedenza più bassa.