1 Lezione del 11-03-25

1.0.1 Interruzione di livello o di fronte

Vediamo un dettaglio sul comportamento dell'APIC: questo può rilevare, in base alla sua configurazione, i **livelli** o i **fronti** delle variabili in ingresso.

Questo può avere delle implicazioni diverse a seconda dell'interfaccia. Ad esempio, avevamo detto che il timer in modalità 2 genera un onda quadra. Se si usa una routine lanciata dal timer a interruzione di programma, e si configura l'APIC per rilevare il livello, potrebbe essere che a routine concluse il livello del timer è sempre alto, e quindi l'interruzione viene lanciata nuovamente.

Questo è chiaramente diverso dal comportamento desiderato, ed è quindi opportuno configurare l'APIC per rilevare i soltanto fronti di salita.

1.1 Altri tipi di interruzioni

Abbiamo visto finora le **interruzioni esterne mascherabili**. Vediamo in verità che esistono altri tipi di interruzione, fra cui:

- **Interruzioni esterne non mascherabili:** cioè che non possono essere mascherate, solitamente rappresentano eventi particolarmente gravi o comunque la cui gestione ha alta importanza;
- **Eccezioni:** eventi che non arrivano dall'esterno, ma si generano all'interno del processore stesso. Questi sono particolari errori logici che potrebbe incontrare il processore, come ad esempio la divisione per 0, il tentativo di eseguire un istruzione non riconosciuta, ecc...

  Un interruzione particolare è quella rappresentata da `INT3`, l'interruzione di *debug*. Attraverso questa, un *debugger* è capace di interrompere l'esecuzione di un programma ad un certo indirizzo del suo codice macchina.

  Un'altra interruzione di debug è data dalla single step, che viene lanciata ad ogni istruzione quando è attivo un certo flag (appunto, il flag single step). Questo permette al debugger di eseguire il programma in modalità *passo singolo*, cioè eseguendo un istruzione e interrompendo, permettendo al programmatore di osservare il suo andamento passo per passo.

  Una differenza fra le interruzioni esterne e le eccezioni è che le eccezioni possono essere sollevate *durante* la lettura e esecuzione di un istruzione, quindi ad esempio mentre si stava interpetando un codice operativo (si pensi all'interruzione di operazione non riconosciuta). In verità, per assicurare l'atomicità dei cicli di esecuzione, la CPU ripristina automaticamente il suo stato a prima del lancio dell'interruzione. In particolare, possiamo distinguere 3 tipi di eccezione:

  - **Fault:** l'esecuzione non viene ancora eseguita, lo stato IP prima della sua esecuzione viene salvato (quindi si rimane alla stessa istruzione), e si può riprovare ad eseguirla dopo aver risolto l'errore;
  - **Trap:** l'esecuzione ormai è stata eseguita, e si salva l'IP successivo.
  - **Abort:** raggruppa degli eventi particolarmente disastrosi in cui l'esecuzione si arresta completamente (ad esempio la tripla eccezione).

Quando viene lanciata una *fault* o una *trap*, il processore cerca nella IDT se esiste un handler corrispondente (segnalato attraverso un bit nell'IDT stessa, alla riga della tabella corrispondente all'eccezione considerata). Nel caso questo non esista, si riprova con la fault di *doppia eccezione*, che quindi rappresenta una fault a sé. Nel caso nemmeno questo handler esista, viene lanciata una fault di *tripla eccezione*, che è di tipo *abort* e comporta quindi l'arresto del programma.