di Pierluigi Panunzi

La gestione degli interrupt

quinta parte

Siamo ormai agli sgoccioli della nostra analisi del microprocessore 80286 dell'Intel, analisi iniziata nel lontano numero 70 di MCmicrocomputer (era il gennaio dell'88...) e che ci ha portato a conoscere, quale più, quale meno, tutte le caratteristiche intrinseche di questo microprocessore, che senza dubbio rappresenta un'innovazione notevole rispetto ai fratelli minori ad 8 bit

Ad un anno e più di distanza dobbiamo ancora ripeterci sul fatto che finora il 286 non è stato utilizzato al meglio nei vari personal viceversa fioriti nel frattempo: sappiamo che, a parte alcune rarissime eccezioni, il 286 viene utilizzato come un «super-8086», del quale ha lo stesso set di istruzioni che esegue in tempi favolosi.

Tutto ciò se non andiamo a toccare l'argomento «Protected Mode» che schiude viceversa la strada alla multiprogrammazione o multi-tasking che dir si voglia.

Prime avvisaglie di sistemi operativi multi-tasking si sono avute, (il famoso DOS 4.0 ed il più famoso OS/2) ma in entrambi i casi si tratta di prodotti non ancora perfezionati che hanno fatto appena capolino nel mondo della programmazione, ma che non hanno ancora preso piede.

D'altro canto bisogna riflettere un attimo sul fatto che il microprocessore in esame è particolarmente complicato da gestire, soprattutto «da zero» e cioè scrivendone proprio il sistema operativo, che ne preveda tutte le funzionalità.

Tutto sommato poi (e lo vedremo al momento opportuno) il passaggio dal 286 al 386 non richiederà quel «gap» notevole che si aveva tra l'8086 e il 286. Tra l'altro confermiamo che prima di passare al 386, il processore oggetto della prossima serie di articoli, avremo modo di conoscere meglio un altro microprocessore utilizzato in parecchi personal e viceversa poco conosciuto: si tratta del V20 della NEC, che in pratica è un «super-clone» dell'8088, dotato tra l'altro di una certa qual «reminiscenza del passato».

Torniamo dunque al nostro 286 per analizzare le ultime exception rimaste.

L'INT 11 (0BH)

Si tratta di un interrupt (o meglio eccezione) generato dal microprocessore allorché si faccia riferimento ad un segmento «non presente» in memoria.

Sappiamo già che è questo il caso in cui, per effetto di vari e complicati meccanismi di «swapping», il segmento (di dati o di codice) su cui si vuole lavorare non è effettivamente presente nella memoria di sistema, ma viceversa è stato «scaricato» nella memoria di massa, perché così era stato deciso dal gestore della memoria: in generale il processo che aveva tale segmento in uso era stato attivato e successivamente accantonato (dal meccanismo di task-switch) salvando le sue risorse nella memoria di massa.

Dal momento che ora è stato riattivato, il processo ha bisogno di nuovo delle sue risorse, che perciò devono essere ripristinate: a ciò proprio serve l'exception connessa alla «non presenza» di un segmento.

Lungi dall'essere un'exception «punitiva» (quale potrebbe essere quella legata ad un tentativo di un programma di forzare un certo livello di protezione), viceversa si tratta di una richiesta di servizio da parte di un processo, il quale altrimenti non potrebbe proseguire nella sua esecuzione.

Ribadiamo il concetto che in questo caso non si tratta (almeno secondo un'interpretazione antropomorfica) di un'azione, da parte del sistema operativo, che si ritorce nei riguardi del task corrente il quale viceversa, poveretto, non poteva (e non potrà mai) sapere se il segmento desiderato era o meno presente in memoria: mentre con un po' di immaginazione possiamo vedere il sistema operativo interrompere brutalmente, quasi scacciare, quei task che avevano osato violare le protezioni, tentando di eseguire un'istruzione inesistente oppure sfondando il muro che delimita i propri segmenti ed in generale il proprio «campo d'azione», viceversa vediamo praticamente un docile sistema operativo che impegna gran parte delle sue energie per ripristinare in memoria quei segmenti che preceden-

INT	rest.	err.	exception
0	*		Divide error
1	*		Single step
2	*		NMI
3	*		Breakpoint
4	*		INTO overflow
5	*		BOUND range overflow
6	*		Invalid Opcode
7	*		Coprocessor not available
8		*	Double fault
9			Coprocessor segment error
A	*	*	TSS error
B	*	*	Segment Not Present
C	*	*	Stack segment error
D	*	*	General Protection

Figura 1 - Nella tabella, uguale a quella mostrata la scorsa puntata, sono indicate le «exceptions» del 256 ed i relativi numeri di interrupt.
Uno «» posto nella colonna «rest.», indica che la routine caduta in errore è «restartabile», mentre un «*» nella colonna «arr.» indica che all'exception è associata una parola sullo stack.*

temente aveva posteggiato nella memoria di massa.

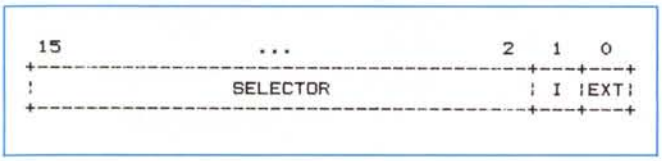
Tornando a cose più serie, all'INT 11 è associata una parola sullo stack (che possiamo vedere in figura 2), la quale indica:

— nel campo SELECTOR, il valore del «selector» relativo al segmento non presente in memoria, valore contenuto in uno dei quattro registri CS, DS, ES o SS a seconda di quale segmento era coinvolto nell'istruzione.

questo caso i registri ES e DS possono contenere valori inutilizzabili dal gestore dell'exception, così come accade per i quattro registri di segmento nel caso di task-switch, dal momento che anche in questo caso i registri in questione vengono caricati prima di testare se si abbia un eventuale errore nello stack.

L'INT 13

È questo l'interrupt associato all'exprovenuto dall'esterno (quale ad esempio un interrupt esterno, l'esecuzione di un'istruzione in single-step, la mancanza del coprocessore matematico oppure un overrun all'interno di un segmento): in questi casi il selector in genere non ha alcun riferimento con l'istruzione che



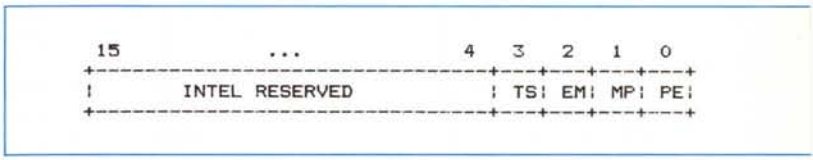


Figura 2 - La parola posta nella sommità dello stack nel caso dell'INT 11 si riferisce ad un segmento non presente in memoria, il cui SELECTOR è appunto riportato, assieme ad altri due campi, relativi ad un evento esterno (EXT) ed all'appartenenza del segmento ad un IDT (campo «I»).

Figura 3 - Questa è la struttura della MSW («Machine Status Word»), che riporta istante per istante lo stato del microprocessore: nel testo vengono spiegati uno per uno i campi che la compongono.

— Nel campo posto in corrispondenza del bit 1 (indicato con «I»), apparirà un «1» nel caso in cui il segmento in esame faccia parte di una IDT (Interrupt Descriptor Table) e cioè nel caso in cui per eseguire un'istruzione INT (o nel processamento di un interrupt esterno), l'elemento della IDT corrispondente a tale interrupt si riferisca ad un segmento appunto non presente in memoria.

— Ed infine nel campo EXT comparirà un «1» nel caso che a generare l'exception sia stato proprio un evento esterno, e cioè un interrupt di cui abbiamo già detto nel punto precedente.

Inoltre c'è da dire che il registro di segmento relativo al segmento non presente in memoria in genere non contiene un valore affidabile: in particolare abbiamo che durante un «task-switch» i registri di segmento (CS, DS, ES ed SS) vengono caricati «prima» che venga verificata l'effettiva presenza in memoria del segmento desiderato, perciò il sistema operativo (o meglio il gestore della memoria, che provvede al meccanismo di «swapping») non deve fare affidamento o peggio usare i contenuti del registro interessato, in quanto per forza di cose si genererebbe un'altra eccezione dello stesso tipo, con il che si innescherebbe un ciclo senza fine.

L'INT 12

Si ha la generazione di un INT 12, allorché si presenti uno «stack fault» e cioè uno sconfinamento dello Stack Pointer (SP) al di là dei limiti viceversa imposti per lo stack, sia a causa di un overflow (troppe «PUSH» che saturano la memoria dedicata allo stack) oppure a causa di un underflow (troppe POP che viceversa svuotano prima del tempo lo stack). Come nel caso precedente, anche in

ception più comune e più volte nominata nel corso delle varie puntate: si tratta della più generica exception ed infatti viene chiamata la «General Protection Fault» (abbreviata solitamente con «GP»).

A tale interrupt fanno capo tutte le violazioni delle protezioni non contemplate in quelle viste finora per i singoli interrupt e data perciò la sua generalità, ad essa viene associata una particolare parola sullo stack che serve ad aiutare il sistema nella comprensione di quanto sia successo.

In particolare un valore nullo può rappresentare varie possibilità di errore (come farà allora il povero sistema operativo?!...), quali i tentativi di:

— accedere ad un segmento di dati quando il valore del registro DS o ES è nullo;

— accedere ad un segmento ad un livello di privilegio maggiore al valore del CPL («Current Privilege Level») del processo in esecuzione;

— violare uno qualsiasi dei limiti non previsti in tutti i casi precedenti di interrupt;

— scrivere all'interno di un segmento che viceversa era di tipo «ready-only».

Un valore invece non nullo posto nello stack viceversa indica quasi sempre il valore di un segment selector che per qualche ragione è errato: in questi casi giocano un ruolo fondamentale i bit 0, 1 e 2 della word.

In particolare il bit 1 dice se il selector è riferito alla tabella IDT (se vale «1») oppure ad una delle due tabelle GDT o LDT (se viceversa vale «0»): in quest'ultimo caso, a decidere tra la tabella «Locale» e quella «Globale» servirà il bit 2.

Inoltre il bit 0 (che viene chiamato «EXT») indicherà con un valore «1» che l'evento che ha provocato l'exception è

era effettivamente in corso di esecuzione e che viceversa era stata interrotta.

Proprio per questo motivo, quando il bit «EXT» è settato, allora il processo interrotto risulta a buon diritto «restartabile» in quanto praticamente incolpevole di quanto è successo.

Infine, nell'ambito del «Real Mode» e cioè ben al di fuori di tutti i metodi di protezione delle risorse, l'INT 13 viene generato allorché un programma tenti di leggere o scrivere una word posta all'indirizzo 0FFFF di un certo segmento, a differenza dell'8086, che viceversa in questi casi effettuava il «wrap around», prendendo come parte più significativa della word il byte posto all'offset 0000 del segmento stesso: questo a pensarci bene significa che in «Real Mode» l'80286 non è del tutto compatibile con l'8086, anche se bisogna dire che l'eventualità prospettata è alquanto rara a presentarsi.

I registri interni del 286 e loro gestione

Abbiamo parlato più volte di una serie di registri interni del 286, riferentisi alle varie caratteristiche di tale microprocessore in modo protetto.

Conosciamo già innanzitutto il registro GDTR, il «Global Descriptor Table Register» e l'IDTR, l'«Interrupt Descriptor Table Register» i quali possono essere inizializzati a puntare alle rispettive tabelle solo ad un livello di privilegio 0, per mezzo, rispettivamente, delle istruzioni

LGDT locazione
LIDT locazione

le quali caricano nel rispettivo registro il contenuto delle tre word poste a partire dall'indirizzo «locazione» sappiamo già

che la prima word rappresenta il campo «LIMIT» e cioè l'estensione della tabella stessa, mentre la seconda word più la parte meno significativa della terza (in totale 24 bit) rappresentano invece l'indirizzo fisico iniziale della tabella.

Sappiamo inoltre che la parte più significativa della terza word non viene usata dal 286.

All'atto dell'inizializzazione del sistema, queste due istruzioni dovranno senz'altro essere eseguite, per poter permettere il corretto funzionamento del 286. Viceversa esiste la coppia di istruzioni.

```
SGDT locazione
SIDT locazione
```

che servono a salvare in memoria, a partire dall'indirizzo dato da «locazione», le tre word di cui sopra: tale coppia di istruzioni può essere viceversa eseguita a qualsiasi livello di privilegio in quanto innocua e tutto sommato di scarsa utilità.

L'istruzione

```
LLDT locazione oppure
LLDT registro
```

viceversa serve ad inizializzare il registro LDTR («Local Descriptor Table Register») a partire dal valore del selector contenuto nella word posta all'indirizzo dato da «locazione» oppure nel registro indicato: sappiamo infatti che l'LDT fa parte della GDT essendone proprio un elemento perciò avente un proprio selector di identificazione.

L'istruzione duale, la SLDT, viceversa può essere eseguita da qualsiasi livello e serve come è facile intuire a leggere il contenuto dell'LDTR.

L'istruzione

```
LTR locazione oppure
LTR registro
```

serve invece, sempre e solo a livello di privilegio o ad inizializzare il ben noto «Task Register», che sappiamo puntare al TSS («Task State Segment»), mentre la duale STR serve in maniera innocua a conoscere il contenuto di tale registro.

La MSW («Machine Status Word»)

Con tale termine ricordiamo che si intende una particolare word posta all'interno del microprocessore, che serve ad identificare in quale particolare stato si trovi il 286: in figura 3 ricordiamo in particolare quali sono gli unici 4 bit utilizzati dal 286.

Abbiamo perciò:

— il campo PE («Protected mode Enabled») è quello che indica, se settato, che il 286 è stato posto in modo protetto: la peculiarità di tale bit, vista l'importanza della sua funzione, è che non può essere banalmente azzerato, per far tornare il 286 in «Real Mode».

Infatti l'unico modo possibile è resettare il microprocessore e scusate se è poco...

— Il bit MP («Monitor Processor extension») è invece strettamente legato alla presenza del coprocessore matematico ed altre notizie sul suo uso ci porterebbero molto lontano.

— Il bit EM («EMulate processor extension») indica (ed anche per questo bit vale quanto detto per il precedente) che è attivata la simulazione via software della istruzione del coprocessore matematico (simpatico, no?!).

— Infine il bit TS («Task Switched») è ancora una volta legato alla presenza del coprocessore ed è in particolare settato «via hardware» all'atto dell'esecuzione di una funzione del coprocessore stesso e serve per la gestione di eventuali errori provocati da quest'ultima funzione eseguita, in presenza di altre istruzioni relative al coprocessore.

Il reset di tale bit avviene viceversa «via software» per mezzo dell'istruzione CLTS («CLear Task Switch»), che può essere eseguita al solo livello 0 di privilegio: ci fermiamo qui nella spiegazione per i soliti motivi...

Non ci resta da dire che la MSW può essere caricata solo al livello 0 per mezzo dell'istruzione LMSW («Load Machine Status Word»), mentre viceversa può essere al solito letta a tutti i livelli, per mezzo dell'istruzione SMSW («Store Machine Status Word»).

L'inizializzazione del 286

Come ultimo argomento riguardante l'80286, diamo un'occhiata ad alcune operazioni che si devono compiere all'inizio dei tempi, subito dopo che è stato attivato il segnale di RESET.

In particolare, per effetto di tale segnale, il microprocessore si pone in uno stato ben definito, caratterizzato dalla presenza di particolari valori posti nei registri base della CPU: si hanno nel dettaglio i seguenti valori:

FLAGS	0002H
MSW	FFF0H
IP	FFF0H
CS selector	F000H
CS base	FF0000H
CS limit	FFFFH
DS selector	000H
DS base	000000H
DS limit	FFFFH
ES selector	0000H
ES base	000000H
ES limit	FFFFH
IDT base	000000H
IDT limit	03FFH

In base a tali valori si vede che il 286 parte in «Real Mode» (bit PE=0 all'interno della MSW) ed in particolare inizia ad eseguire l'istruzione posta ad un indirizzo dato da CS:IP pari a FFFF0 ed esattamente uguale a quello generato al RESET di un 8086 (anche se con due valori differenti per i registri CS ed IP).

Volendo poi lavorare esclusivamente in «Real Mode» (cosa che succede con l'MS-DOS), allora la sequenza di operazioni da effettuare per inizializzare correttamente il sistema è praticamente quella usata dal BIOS dei PC e riguarda genericamente:

— l'allocazione in memoria di uno stack;

— l'inizializzazione di dispositivi esterni al microprocessore;

— l'inizializzazione della tabella di vettori di interrupt;

— il riempimento opportuno dei vari registri, della MSW nonché del registro dei FLAG per poi

— eseguire un programma quale può essere un «bootstrap loader» da disco rigido.

Invece per quanto riguarda il «Protected Mode» bisogna:

— innanzitutto creare le due tabelle GDT e IDT ed i relativi registri GDTR ed IDTR;

— settare (solo ora) il bit PE della MSW per entrare in modo protetto;

— effettuare un salto per mezzo di una istruzione «JMP inter-segment», in modo tale da svuotare la coda di istruzioni all'interno del processore;

— costruire un TSS («Task State Segment») relativo al primo task da eseguire;

— caricare il registro LDTR a partire da un valore posto all'interno della GDT, oppure porre tale registro a 000 nel caso che la LDT non sia necessaria;

— far puntare la coppia SS:SP ad una locazione all'interno dello Stack Segment;

— marcare come «Not Present» (nel campo NP del segment descriptor) tutti i segmenti non effettivamente presenti in memoria;

— inizializzare i bit della MSW e della parola di FLAG per ottenere una configurazione desiderata;

— inizializzare i dispositivi esterni al microprocessore;

— assicurarsi dell'esistenza di tutte le routine di gestione degli interrupt, dopodiché abilitare gli interrupt;

— infine effettuare il «bootstrap» da disco.

Ecco che dunque, dopo aver dato un'occhiata fugace alle inizializzazioni necessarie a seguito del RESET, lasciamo l'argomento «80286»: già abbiamo accennato all'inizio della puntata di cosa parleremo nella prossima. Ovviamente, come nel caso del «passaggio» tra l'8086 e l'80286, anche in questo caso si avrà un cambiamento nel titolo della rubrica, fermo restando ovviamente il redattore... MC

CONTINUA IL SUCCESSO
A FIRENZE, BOLOGNA, MILANO

- PROFESSIONALITÀ
- QUALITÀ
- CONVENIENZA
- PRONTA CONSEGNA
- ASSISTENZA DIRETTA NEI NOSTRI NEGOZI

FINO ALLA FINE DEL MESE

L. 5.000 DI SCONTO

SULL'ACQUISTO DI TRE CONFEZIONI QUALSIASI DI DISCHETTI, COSÌ QUESTA RIVISTA È GRATIS

C.D. MILANO
Via Cenisio, 12 - 20154 MILANO
Tel. 02/33100204 - Fax 02/33100835

C.D. BOLOGNA
Viale Lenin, 12 c/d - 40139 BOLOGNA
Tel. 051/494103 - Fax 051/540293

C.D. FIRENZE
Viale Matteotti, 9 - 50121 FIRENZE
Tel. 055/660524 - Fax 055/587765