di Pierluigi Panunzi

La gestione degli interrupt

seconda parte

Proseguiamo dunque l'analisi della gestione degli interrupt da parte del 286, proseguendo dal punto in cui ci eravamo fermati la scorsa puntata

Per quanto riguarda il passaggio dalla routine in corso di esecuzione alla routine di gestione dell'interrupt, valgono in prima analisi le modalità già conosciute per l'8086: in particolare nello stack verrà salvato successivamente lo stato dei flag e l'indirizzo di ritorno della routine interrotta.

Ma in quale stack vengono salvate queste informazioni?!

Ovvio, oseremmo dire, che lo stack in esame è quello proprio della routine interrotta, propriamente nello stack relativo al livello di privilegio corrente.

Però se il passaggio alla routine di interrupt richiedesse un incremento nel privilegio, ecco che allora lo stack corrente non va più bene: ecco che dunque deve essere generato un nuovo stack a livello di privilegio maggiore ed i relativi SS ed SP verranno letti dal TSS (ricordano i lettori di cosa si tratta?!).

In questo nuovo stack verranno perciò innanzitutto salvati l'SS e l'SP relativi al vecchio livello di privilegio e successivamente i flag e l'indirizzo di ritorno: nelle figure 1 e 2 possiamo vedere lo stato dello stack nei due casi or ora menzionati.

Comunque non si creda che da qui in poi tutto vada liscio: ad esempio se l'interrupt viene gestito attraverso un «interrupt gate», allora implicitamente si conta sul fatto che il segmento di codice selezionato abbia un privilegio tale da poter poi riabilitare nuovi interrupt.

Infatti se ciò non accade, l'istruzione di IRET, di fine interrupt e di ritorno alla routine interrotta, non avrà la possibilità di abilitare nuovamente gli interrupt: è questo il caso in cui il CPL è numericamente superiore al valore dell'IOPL (ricordano i lettori che cosa è l'«I/O Privilege Level»?!).

Abbiamo accennato prima allo stato dello stack nei due casi in cui rispettivamente non si abbia e si abbia una variazione di livello di privilegio nel cedere il controllo alla routine di gestione dell'interrupt: ci siamo sempre riferiti, lo ricordiamo, al caso in cui la routine di interrupt stessa venga attivata da un «trap gate» oppure da un «interrupt gate».

In realtà il tutto non è così semplice come potrebbe sembrare: vedremo ora infatti la miriade di operazioni logiche e non che l'80286 esegue allorché riceve in interrupt.

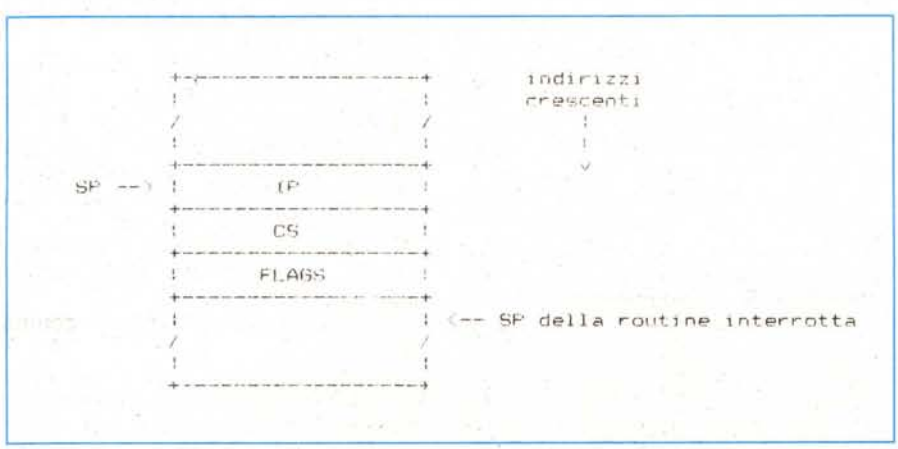

Figura 1 - Configurazione dello stack nel caso che il passaggio alla routine di interrupt non richieda transizione di livello: la situazione è perfettamente identica a quella del micro 8086.

Passo passo anche gli interrupt vengono risolti...

... Il significato del titolo apparirà chiaro alla fine, quando ci accorgeremo che tutto sommato gli interrupt vengono «effettivamente» gestiti...

Innanzitutto viene cercato l'«interrupt vector» all'interno dell'IDT, per vedere se per caso la tabella dei descrittori fosse «più corta» del necessario: nel caso che non si trovasse dunque il descriptor relativo all'interrupt allora è anche ovvio che venga generata una «GP» («General Protection»).

Visto che ci siamo, iniziamo dunque a familiarizzare con gli errori e le «exceptions», che vedremo poi come vengono a loro volta gestite; diciamo solo che ad ogni «exception» è associata

una word posta in cima allo stack, word che differirà, ovviamente, da situazione a situazione, per essere meglio identificabile dall'apposita routine che gestisce gli errori.

Se tutto va bene allora viene testato l'«Access Rights Byte» posto all'interno del «descriptor», per vedere se si riferisce ad un interrupt gate, ad un trap gate oppure ad un task gate (e di questo caso, da questo punto in poi, ci occuperemo più tardi): in caso contrario viene generata un'altra GP.

Se l'interrupt è generato via software (con l'istruzione INT), allora viene controllato che il DPL del descrittore sia numericamente maggiore o uguale al CPL, altrimenti si genera una GP.

Infine il gate che ha superato fin qui tutti i test deve effettivamente essere presente (talmente ovvio da sembrare lapalissiano) altrimenti viene generata una «NP» («Not Present» exception).

Ora, a seconda del tipo di gate si seguono due strade differenti: noi per ora analizziamo il caso del «trap gate» e dell'«interrupt gate», mentre più avanti analizzeremo il caso rimanente del «task gate», come già avevamo anticipato.

Prosegue dunque a spron battuto l'analisi del descriptor, per vedere se tutto va bene.

All'inizio infatti si esamina il selector del CS indicato nel descriptor:
— il selector deve essere diverso da zero altrimenti viene generata una GP;
— il selector deve puntare «all'interno» della tabella dei segment descriptor, altrimenti c'è la generazione di un'altra GP;
— l'«Access Rights Byte» deve indicare che il segmento è effettivamente un code segment altrimenti parte subito una GP;
— infine il segmento deve essere presente in memoria altrimenti stavolta viene generata (e tutti i lettori in coro...) una NP.

A questo punto viene testato il DPL del descrittore fin qui esaminato: se risulta in valore numerico minore al

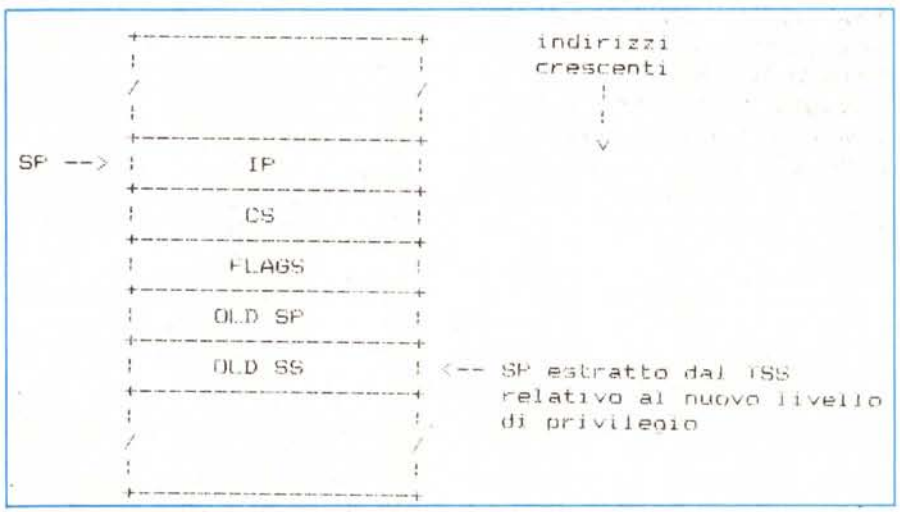

Figura 2 - Configurazione dello stack nel caso che il passaggio alla routine di interrupt richieda transizione di livello; nel nuovo stack verranno innanzitutto salvati l'SS e l'SP della routine interrotta (per il livello di privilegio corrente della routine stessa) e successivamente sono poste le informazioni solite.

privilegio corrente (CPL), allora si avrà una routine di gestione con privilegio maggiore, altrimenti la routine di gestione dell'interrupt avrà lo stesso livello.

Dapprima esaminiamo il caso di transizione di livello.

La CPU deve dapprima testare il selector ed il descrittore del nuovo stack all'interno del TSS (Task State Segment) corrente ed in particolare:
— il selector deve essere maggiore di zero, altrimenti avremo una GP;
— l'indice contenuto nel selector in esame deve essere all'interno dei limiti del descrittore relativo, altrimenti si genererà una «TS» (che deriva da «invalid Task State segment»);
— l'RPL (ricordate?!...) del selettore deve essere uguale al DPL del code segment, altrimenti si genera ancora una TS;
— i DPL dello stack segment e del code segment devono essere identici, altrimenti si ha una TS;
— il descrittore deve indicare un segmento di dati su cui si possa scrivere (infatti deve essere uno stack!), altrimenti si ha una TS;
— infine, al solito, il segmento che ospiterà il nuovo stack deve essere presente in memoria, altrimenti si avrà un'exception del tipo «SS» (che sta per «Stack segment fault»).

Supponendo dunque che finora il nuovo stack segment vada bene, si prosegue oltre:
— si testa se il nuovo stack segment abbia almeno spazio per poter allocare 10 byte (o meglio 5 word), altrimenti si otterrà un'altra SS;
— il valore contenuto nell'IP deve stare all'interno dei limiti imposti dal suo CS, altrimenti si ha una GP.

Finiti dunque questi «pochi» controlli la CPU è finalmente pronta a salvare nello stack le informazioni che più ci interessano. Ricordiamo a tal proposito, nel caso che ci si fosse persi strada facendo, che vogliamo ottenere un riempimento dello stack, come in figura 2.

A questo punto dunque la CPU estrae i nuovi valori di SS ed SP dal TSS, nonché il nuovo CS ed il nuovo IP (l'indirizzo della routine di gestione dell'interrupt...) contenuti all'interno del gate, carica i descrittori del CS e dell'SS di sui sopra ed infine effettuerà l'operazione di «push» dapprima del puntatore al vecchio stack (è la coppia SS:SP) e poi l'indirizzo di ritorno della routine che era stata interrotta.

Infine la CPU effettua le seguenti operazioni:
— aggiorna il valore del CPL al valore DPL nel nuovo segmento di codice;
— aggiorna l'RPL contenuto nel registro CS al valore del nuovo CPL;
— resetta il falg «Interrupt Enable», disabilitando perciò ulteriori interrupt, solo nel caso in cui nell'IDT di partenza si aveva a che fare con un «Interrupt Gate»;
— resetta il «Trap Flag» (TF) ed infine;
— resetta il «Nested Task Flag» (NT).

Finalmente, a partire da un interrupt con routine di gestione «aperta» da un «trap gate» o un «interrupt gate», e nel caso in cui la routine stessa è a livello di privilegio maggiore, finalmente, dicevamo, i giochi sono fatti...

Rimangono ancora due situazioni da analizzare.

La prima riguarda il caso in cui, sempre attraverso un «trap gate» o un «interrupt gate», la routine di gestione dell'interrupt sia allo stesso livello della routine interrotta.

In tal caso la CPU effettua i seguenti controlli:
— testa se i limiti dello stack corrente consentono la scrittura di almeno 6 byte (o meglio 3 word), altrimenti si avrà una SS;
— testa se l'IP è all'interno dei limiti del code segment attuale, altrimenti si ha subito una GP.

Terminati questi test, effettuerà una serie di operazioni:
— salverà nello stack i flag e l'indirizzo di ritorno della routine interrotta;
— caricherà dal gate i nuovi valori per il CS e per l'IP, per andare a puntare alla routine di gestione dell'interrupt;
— setterà il campo RPL all'interno del registro CS al valore del CPL;
— resetterà il flag di abilitazione degli interrupt, disabilitandoli, ancora una volta se e solo se si trattava di un «Interrupt Gate» ed infine;
— resetterà il «Trap Falg» («TP») ed il «Nested Flag» («NF»).

Con ciò può senz'altro dirsi concluso anche il caso di routine di interrupt «aperta» da un gate ed allo stesso livello di privilegio della routine interrotta.

Rimane infine da esaminare il complesso di operazioni svolte dalla CPU nel caso che la routine di gestione dell'interrupt sia attivabile per mezzo di un «Task Gate»: detto a parole, sembra molto più semplice, ma in realtà i meccanismi che sono insiti risultano più complicati.

In particolare dapprima la CPU esamina il selector al TSS posto all'interno del «Task Gate»:
— tale selettore deve specificare, nel bit «local/global», che la Descriptor Table è «global», altrimenti si ha una GP;
— l'indice fornito dal selettore deve trovarsi all'interno dei limiti della GDT, altrimenti si ha una GP;
— l'analisi dell'«Access Rights Byte» deve segnalare un TSS disponibile (per la cronaca un valore pari a 00001), altrimenti si ha una solita GP;
— infine, come oramai siamo abitauti, il TSS («Task State Segment») deve essere presente in memoria, altrimenti l'exception generata sarà stavolta una NP («Not Present»).

Visto che tutto è a posto, a questo punto si innesca il meccanismo di «task switching», che abbiamo già analizzato in dettaglio in una delle scorse puntate: sappiamo perciò che la CPU effettuerà una serie di controlli e salvataggi nel nuovo stack, tra i quali quello dei flag e dell'indirizzo di ritorno.

Come ultimo atto infine la CPU testerà se l'IP della routine di gestione dell'interrupt si trova all'interno dei limiti del code segment, altrimenti (ma sarebbe un peccato visto che c'eravamo quasi...) viene generata una GP.

Ma quanto ci mette la CPU?!

È questa probabilmente la domanda che si porranno i lettori a questo punto, se non se la sono fatta già prima, nel corso dell'analisi: dobbiamo ricordare che nessuno dei test finora analizzati può essere minimamente eliminato o bypassato, in quanto in «Protected Mode» nell'ambiente vige una ferrea legge marziale con costrizioni da tutte le parti. Però ovviamente la precisione nelle analisi e la richiesta di sicurezza e di protezione si «paga» in termini di tempi di esecuzione: ricordiamo che invece dal punto di vista software non ci si accorge di niente...

In particolare possiamo paragonare i tempi di esecuzione in termini di cicli di clock dapprima con i cicli richiesti dall'8086 e poi con i tempi di esecuzione nel caso di «Real Mode» e «Protected Mode», quest'ultimo scisso in tre valori a seconda del «gate» e dei livelli di privilegio.

Vediamo dunque la tabella di comparazione:

▼

micro e condizione	INT 3	INT n	INTO
8086	52	51	53 o 4
286 «Real Mode»	23	23	24 o 3
286 «Protected Mode»			
— stesso privilegio	40	40	24 o 3
— privilegio maggiore	78	78	24 o 3
— «Task Gate»	167	167	24 o 3

Terminiamo dunque questa puntata assai complessa, analizzando brevemente i valori riportati nella tabella: in particolare i valori «doppi» per l'istruzione «INTO» si riferiscono, il minore al caso in cui il flag di «Overflow» non sia settato (e perciò non si effettua il salto), mentre l'altro valore vale nel caso di Overflow.

Rispetto ai valori in cicli di clock dell'8086, il 286 in «Real Mode» si conferma ancora una volta un «8086 enhanced», velocizzato, mentre in «Protected Mode» iniziano i dolori: mentre i 40 ed al limite i 78 cicli nel caso di interrupt attraverso «trap gate» ed «interrupt gate» (ma con o senza variazione di livello di privilegio) sono in un certo senso confrontabili con i 51 o 52 cicli dell'8086, dove «casca l'asino» è proprio per quei 167 cicli di clock, la maggior parte dei quali sono richiesti dal «task switch» e dai relativi salvataggi.

Comunque con un 286 che viaggia a 10 MHz, l'esecuzione dell'istruzione di una INT nel caso peggiore richiede quasi 17 microsecondi, un tempo dell'ordine di quello relativo ad una MUL con un 8086 clock-ato a 4.77 MHz.

È chiaro però che i 17 microsecondi sembreranno tantissimi o viceversa pochissimi a seconda dell'applicazione del sistema ed in particolare degli interrupt e del loro «significato».

MK

A GENOVA...
HARDWARE & DISTRIBUZIONE

PERSONAL COMPUTER INTERCOMP

XPC30 Personal computer MS/DOS compatibile con clock a 4.77/10MHZ, 640K a bordo, scheda video compatibile CGA-Hercules seriale, parallela, orologio, mouse adapter, spazio fino a 3 unità interne da 3.5" SLIM LINE, (drive 720/1.44MB, hard disk 20/40MB) dimensioni contenute, DOS e GW basic con manuali in italiano. Ora anche nella versione VGA con processore 8086.

XAT Personal computer 80286 a 10/12MHZ 1-0 WS 512K RAM espandibili a 1024 a bordo nuovo design a dimensioni contenute che lascia spazio a n. 2 alloggiamenti da 5.25" e 2 alloggiamenti da 3.5" SLIM. Tastiera estesa, DOS e GW BASIC con manuali in italiano. Adattatore video a scelta CGA-HERCULES-VGA.

X386 Processore 80386 1MB espandibili a 2 MB a bordo, 3 slot di espansione a 8 BIT, 2 a 16 BIT, 1 a 32 BIT, 8/16 porte seriali (opzionali) 2 alloggiamenti da 5.25" e due da 3.5" hard disk da 20 a 380MB, e fino a 800MB con disco ottico, tastiera estesa, DOS e GW BASIC con manuali in italiano, schede video CGA-HERCULES VGA. Nelle versioni 16MHZ, 20MHZ, 20MHZ cache memory.

ANCHE NELLE VERSIONI TRASPORTABILI

Elaboratori Intercomp
- Affidabilità
- Professionalità
- Design
- Budget

Soluzioni hardware e software per l'azienda e l'ufficio. Schede Add-On di espansione e interfaccia, accessori, mouse, reti, hard disk.

Monitor multisync NEC

- Multisync II 14"
- Multisync Plus 15" 960×720
- Multisync XL 20" 1024×768
- NEW Scheda video NEC MVA 1024×768 256 colori, drivers software in dotazione: Autocad, Ventura, Windows, Gem, 123, Symphony, e molti altri

Stampanti 24 aghi & laser NEC

- P 2200
- P 6 PLUS
- P 7 PLUS

ed accessori in pronta consegna. Tutti i supporti soft ed hard per la grafica ed il testo a 24 aghi

NEWS Modem per PS/2 con sotware completamente italiano in dotazione. Possibilità di collegarsi con Videotel, Itapac, Prestel, Minitel ecc. configurazione ed installazione automatiche software di telemanutenzione «freeway master» hard disk 40 ed 80 MB 19 millisecondi 3.5".

SILVER SYSTEM

PRODOTTI ED ACCESSORI PER L'INFORMATICA E L'OFFICE AUTOMATION

16156 PEGLI GENOVA - VIA TEVERE 2A-8-10
TEL. (010) 680.685-689.324 - FAX (010) 686609