di Pierluigi Panunzi

La gestione degli interrupt

terza parte

Eccoci alla terza parte della nostra analisi della gestione degli interrupt da parte del microprocessore 80286: anche stavolta introdurremo dei nuovi concetti ed analogamente troveremo l'applicazione di altri già conosciuti

Il terzo tipo di gestione di interrupt

Nella prima parte della rubrica abbiamo detto che a seguito dell'arrivo di un interrupt possiamo far percorrere al 286 tre strade differenti a seconda che l'elemento all'interno della tabella IDT sia un «interrupt gate», un «trap gate» oppure un «task gate».

I primi due casi sono già stati analizzati ed ora ci manca l'ultimo, in base al quale (già si può capire dal nome) per effetto di un interrupt viene innescato un «task switch», appunto perché il descrittore associato all'interrupt pendente è proprio un «task gate».

Non entreremo ancora nei dettagli relativi alla questione del «task switch», della quale abbiamo parlato ampiamente nelle scorse puntate, ma cercheremo insieme di ritrovare in questo caso quanto già detto in precedenza.

Intanto diciamo che gestire un interrupt per mezzo di uno switch di task comporta in partenza già due importantissimi vantaggi:

— il primo è che il task attivato dall'interrupt è... un nuovo task e perciò è per sua natura completamente isolato dall'altro task, quello interrotto, sia per quanto riguarda gli indirizzi di memoria, sia per ciò che concerne i livelli di privilegio, in quanto quello del task attivato non dipende da quello del vecchio task;

— il secondo vantaggio è che è proprio grazie al meccanismo di task switch che il microprocessore salva automaticamente tutti i registri interni, a differenza di quanto accade con un «interrupt gate», allorché viene salvato nello stack solo il contenuto dei flag e la coppia CS:IP.

Comunque, analogamente a quanto accade con la gestione attraverso un «trap gate» o un «interrupt gate», anche in questo caso vengono applicati i criteri ben noti riguardo ai livelli di privilegio ed alla presenza o meno del «TSS»: come abbiamo visto parlando del task switch, si avrà ancora il settaggio del flag NT («Nested Task»), all'interno dei flag del nuovo task ed inoltre si avrà il salvataggio del «TSS selector» del task interrotto come «back link» all'interno del nuovo TSS.

All'uscita della routine di gestione dell'interrupt, che avviene per mezzo dell'istruzione IRET, si ha di nuovo un task switch, stavolta «all'indietro», con lo scopo di ritornare ad eseguire il task interrotto.

Fatto importante è che in questo caso, dato che si tratta di un task switch, viene salvato nel proprio TSS lo stato della routine di gestione dell'interrupt e ciò comporta che una successiva attivazione della routine di gestione dell'interrupt si troverà ad eseguire proprio l'istruzione sucessiva alla IRET, laddove il solerte programmatore farà bene a porre un «ricco» JMP all'inizio della routine stessa: anche se ciò può sembrare strano, non bisogna dimenticarsi che la routine di gestione dell'interrupt è un vero e proprio task e perciò deve ovviamente sottostare alle regole relative al task switching.

In particolare a seguito di tale evento si vuole che un task interrotto prosegua proprio dal punto in cui era stato fermato, a parità di condizioni e di stato precedenti allo switch, altrimenti il tutto non funzionerebbe minimamente.

Ora anche la routine di gestione dell'interrupt è un task (perché così noi abbiamo scelto...) e perciò all'atto di un nuovo task switch (ottenuto in questo caso con un'istruzione IRET, proprio come avevamo già detto a proposito del task switch) il task deve sottostare in toto alle regole: in questo caso il salvataggio del suo stato non sarebbe necessario così come non sarebbe logico aspettarsi che un'ulteriore chiamata a tale routine riprenda dall'istruzione successiva alla IRET.

Comunque, come detto, ciò è facilmente aggirabile, mentre ovviamente il paragone è presto fatto con le routine attivate da un «interrupt gate», che prevedono invece l'attivazione della routine sempre e solo a partire dal suo entry-point: ma si sa che queste routine non sono task, quindi vanno trattate con differente filosofia.

Da tutto quanto detto finora appare subito chiaro che un interrupt task (task attivato a causa del sopraggiungere di

un interrupt), deve evitare assolutamente l'arrivo di altri interrupt mentre è in corso di esecuzione: in caso contrario, e cioè se a causa di un interrupt si fa riferimento ad un TSS di un task già in corso di esecuzione, allora scatta l'oramai consueto meccanismo di protezione con la generazione di una particolare «exception».

tipo di IRET	cicli di clock
ritorno "normale" (real mode)	17
ritorno "normale" (protected mode)	31
ritorno da minore privilegio	55
ritorno da un task differente (NT=1)	169

Figura 1 - Numero di cicli di clock richiesti per l'esecuzione di un'istruzione IRET a seconda del tipo di contesto in cui si trova tale istruzione: intendiamo per «normale» un ritorno da una routine di interrupt innescata nella maniera più semplice (alla maniera dell'8086, tanto per fare un esempio) con salvataggio nello stack dei flag e della coppia CS:IP.

Quale «gate» scegliere per gestire un interrupt?

Alla luce di tutti questi fatti, dunque, siamo in grado di poter scegliere quale gate utilizzare per attivare una routine di interrupt.

Sappiamo che con un task gate abbiamo la garanzia di un completo isolamento tra il task interrotto e quello di gestione dell'interrupt, soprattutto per quel che riguarda il salvataggio ed il successivo caricamento dei registri della CPU, ma il tutto lo paghiamo al prezzo assai salato di tempi di esecuzione improponibili in certe particolari situazioni.

Viceversa, dunque, laddove la velocità è tutto, allora conviene di più un «trap» od un «interrupt gate»: in tutti i casi, a livello di programmazione, la routine di gestione dell'interrupt terminerà con un'istruzione IRET.

Altra differenza tra le due possibilità (task switch o gestione «normale»), lo ricordiamo, sta nel fatto che il task switch avviene indipendentemente dal livello di privilegio del task in corso di esecuzione e che perciò dovrà essere interrotto, mentre nell'altro caso la routine di gestione dell'interrupt deve poter girare ad un livello di privilegio uguale o maggiore di quello della routine in corso di esecuzione.

Quattro chiacchiere sulla IRET

Abbiamo parlato del fatto che la routine di gestione dell'interrupt deve terminare con una IRET: a tal proposito desideriamo ancora una volta mostrare la differenza tra i tempi di esecuzione di tale istruzione nei vari casi possibili, fermo restando il codice operativo dell'istruzione nei vari casi (riconoscibili però «dal contesto»).

Facendo riferimento alla figura 1, si hanno perciò i seguenti quattro casi:

— il ritorno da una routine di gestione di interrupt, in modo reale dura 17 cicli di clock se in «modo reale» e 31 se in «protected mode»; in entrambi i casi si tratta di un «normale» ripristino dallo stack dei flag e della coppia CS:IP;

— il ritorno da una routine posta a differente livello di privilegio di quello della routine interrotta comporta 55 cicli di clock;

— infine il ritorno che avviene per mezzo di un task switch, individuabile dalla presenza del bit NT («Nested Task») della parola di flag posto ad 1, richiede la bellezza di 169 cicli di clock.

Analogamente a quanto fatto nel caso dell'istruzione INT, analizziamo ora passo passo le varie operazioni che compie il microprocessore quando sta eseguendo un'istruzione IRET.

Innanzitutto si va a testare il famoso bit NT il quale, se settato, indicherà che il ritorno avviene da un task verso un altro task; ora analizziamo dunque questo caso, in base al quale vengono effettuate le seguenti operazioni:

— viene esaminato il «back link» contenuto nel TSS indirizzato dal valore corrente del TR («Task Register»);

— questo back link deve «puntare» ad un oggetto posto nella tabella «globale» (il bit «Local/Global» deve indicare «Global»);

— il valore di «indice» al suo interno deve rimanere nell'ambito dei limiti della GDT;

— una volta «puntato» all'elemento della GDT, il campo «AR» («Access Rights byte») deve indicare un TSS;

— il nuovo (sarebbe meglio dire il vecchio...) TSS deve avere il bit BUSY settato (altrimenti a partire da quale task c'era stato il task switching?): nel caso in cui uno di questi quattro punti desse un risultato negativo allora il meccanismo viene interrotto e viene altresì generata una «exception» relativa ad un «Invalid Task State», brevemente indicata come «TS»;

— infine il TSS (che ricordiamo sta per «Task State Segment») deve essere presente, altrimenti si ha una «Not Present exception», detta in breve «NP»;

— se tutto va bene allora a questo punto viene effettuato un task switch (al contrario...) senza che però venga settato il bit NT, con il che viene ripristinato lo stato del task interrotto, grazie a tutte le informazioni poste nel TSS individuato dal back link che finora era stato testato;

— il task appena abbandonato, che era quello relativo alla routine di gestione dell'interrupt, viene marcato come «NOT BUSY»;

— infine il valore del registro IP deve rimanere ben all'interno dei limiti dettati dal code segment descriptor, altrimenti viene generata una «General Protection exception», detta genericamente «GP».

Nel caso invece in cui il bit «Nested Task» («NT») è nullo allora si ha il caso di ritorno «più normale» da una routine di interrupt, senza tanti problemi dovuti allo switch di task, ma magari con cavilli legati al cambiamento di livello di privilegio.

In particolare:

— si testa che la seconda parola all'interno dello stack sia all'interno dei limiti imposti dallo stack segment descriptor altrimenti si ha una «Stack Segment exception», detta «SS»;

— l'RPL («Requested Privilege Level») posto all'interno del selettore del CS relativo all'indirizzo di ritorno deve essere maggiore o uguale al CPL («Current Privilege Level»), altrimenti si genera un «General Protection exception» («GP»);

— ora a seconda che l'RPL è maggiore oppure uguale al CPL si hanno due strade completamente differenti.

Ora analizziamo il caso di uguaglianza dei due livelli di privilegio:

— le tre word poste sulla sommità dello stack devono essere all'interno dei limiti dettati dallo stack segment descriptor, altrimenti si avrà una «Stack Segment exception» («SS»);

— in particolare la word riferita al CS deve indicare un selector non nullo ed anche all'interno dei limiti della descriptor tabelle altrimenti si ha una «General Protection exception» («GP»);

— il byte dei «diritti di accesso» (il ben noto «Access Rights byte», «AR») deve indicare che il segmento puntato dal CS è proprio un code segment altrimenti si ha una «GP»;

— il «DPL» («Descriptor Privilege Level») deve essere minore o uguale al «CPL», altrimenti si ha una «GP»;

— il segmento sin qui trovato deve essere «presente» in memoria, altri-

menti si ha una «Not Present exception», detta «NP»;
— il valore letto dallo stack per il registro IP deve rimanere rigorosamente entro i limiti imposti dal code segment descriptor, altrimenti si avrà una «GP»;
— se tutto finora è andato bene viene caricata dallo stack la coppia CS:IP e la word contenente i flag;
— viene incrementato il registro SP di 6.

Con il che il controllo passa alla routine che era stata interrotta.

Invece se c'era differenza tra i livelli di privilegio RPL e CPL, allora si effettuano le seguenti operazioni:
— si controlla che le cinque word poste sulla sommità dello stack siano all'interno dei limiti imposti dallo stack segment descriptor, altrimenti si avrà una «SS»;
— la word relativa al CS deve indicare un selector non nullo ed all'interno dei limiti della descriptor table altrimenti si ha una «GP»;
— l'«Access Rights byte» («AR») deve indicare che il segmento puntato dal CS è proprio un code segment altrimenti si ha una «GP»;
— il «DPL» («Descriptor Privilege Level») deve essere minore o uguale al «CPL», altrimenti si ha una «GP»;
— infine il segmento di codice deve essere «presente» in memoria, altrimenti si ha una «NP»;
— si esamina l'SS ed il suo descrittore associato (ricordiamo che nel caso in cui vi sia transizione di livelli di privilegio, allora viene salvata anche la coppia SS:SP) ed in particolare il selettore deve essere diverso da zero e compreso tra i limiti imposti dal descrittore, altrimenti si ha una «GP»;
— si verifica l'uguaglianza tra l'RPL contenuto all'interno dell'SS e l'RPL relativo al CS di ritorno ed in caso contrario viene generata una «GP»;
— si controlla che l'«Access Rights byte» indichi per il segmento in esame la possibilità di scrittura di dati, altrimenti si ha una «GS»;
— il «Descriptor Privilege Level» («DPL») del segmento individuato in precedenza deve essere uguale all'RPL già verificato prima, altrimenti si ha una «GP»;
— lo stack segment in esame deve essere presente in memoria, altrimenti si avrà una «NP»;
— infine l'IP deve avere un valore contenuto all'interno dei limiti imposti dal code segment selector altrimenti si ha una «GP».

Dopo questi controlli:
— viene caricata dallo stack la coppia CS:IP, la word contenente i flag nonché la coppia SS:SP;
— viene settato il «Current Privilege Level» («CPL») al valore posto all'interno del registro CS.

Infine, per quanto riguarda i registri DS ed ES (e questo è un fatto nuovo...), viene testata la validità e cioè:
— viene controllato che il valore contenuto sia entro i limiti gestiti dal data segment descriptor;
— l'«Access Rights byte» deve indicare che il segmento in esame è un segmento di dati oppure un segmento leggibile di codice;
— il livello di privilegio «DPL» deve essere minore o uguale al «CPL»;
— se qualcuna di queste condizioni non è verificata, allora il registro implicato viene posto a zero, altrimenti viene lasciato così com'è.

Con il che dunque viene dato il controllo alla routine il cui indirizzo è stato appena posto in CS:IP e cioè alla routine che era stata interrotta dall'interrupt e dalla sua routine di gestione.

Con questo abbiamo terminato la presente puntata: nella prossima parleremo più diffusamente delle «exception» a partire dai differenti tipi, per arrivare ai meccanismi che ne scaturiscono. MC

PERSONAL SELF SERVICE
SUPER MARKET DELL'INFORMATICA

MEMORIE DI MASSA E CONTROLLER

1 - Hard disk 20 Mb SEAGATE ST225	L. 440.000
2 - Hard disk 40 Mb SEAGATE ST251	L. 770.000
3 - Hard disk 20 Mb KALOK 3,5"	L. 495.000
4 - Hard disk 40 Mb SEAGATE 3,5"	L. 825.000
5 - Floppy drive 360 Kb (5,25")	L. 143.000
6 - Floppy drive 1,2 Mb (5,25")	L. 176.000
7 - Floppy drive 720 Kb (3,50")	L. 165.000
8 - Floppy drive 1,44Mb (3,50")	L. 198.000
9 - Meccanica per FD 3,5" a FD 5,25"	L. 27.500
10 - Streamer TEAC 140 / 65 Mb	L. 1.110.000
11 - Controller hard-disk per XT + cavi	L. 110.000
12 - Controller floppy disk per XT + cavi	L. 44.000
13 - Controller H.D. F.D. per AT + cavi	L. 220.000

MAINBOARD

14 - Mainboard i8088 10 Mhz (0KRAM)	L. 154.000
15 - Mainboard i80286 6/8/10 Mhz (0KRAM)	L. 495.000
16 - Mainboard i80286 6/12 Mhz	L. 550.000
17 - Mainboard i80386 16 Mhz (512KRAM)	L. 3.080.000

VARIE

18 - Alimentatore 150 Watt	L. 99.000
19 - Alimentatore 180 Watt	L. 110.000
20 - Gruppo intervento 300 Watt durata 30m	L. 660.000
21 - Gruppo intervento 500 Watt durata 18m	L. 880.000
22 - Scocca orizzontale XT	L. 110.000
23 - Scocca verticale	L. 330.000
24 - Tastiera 102 tasti italiana	L. 110.000
25 - Cavo per stampante parallela	L. 16.500

INTERFACCE

26 - Adattatore grafici MGA/CGA	L. 99.000
27 - Adattatore per stampante parallela	L. 27.500
28 - Adattatore seriale RS232 1 P	L. 44.000
29 - Adattatore seriale RS232 2 P	L. 66.000
30 - Adattatore EGA	L. 275.000
31 - Adattatore Super EGA	L. 330.000

ESPANSIONI

32 - EPROM 2764 / 27128 / 27256	L. 12.000
33 - RAM 4464-12	L. 23.000
34 - RAM 4164-12	L. 11.000
35 - RAM 41256-15	L. 16.500
36 - RAM 41256-12	L. 18.700
37 - RAM 41256-08	L. 22.000
38 - Scheda di esp. memoria XT 576 0KRAM	L. 77.000
39 - Scheda di esp. memoria AT 2Mb 0KRAM	L. 220.000
40 - Coproces. matem. per XT 8087 8Mhz	L. 330.000
41 - Coproces. matem. per AT 80287 8Mhz	L. 550.000
42 - Coproces. matem. per 386 80387 16 Mhz	L. 935.000

MONITOR

43 - Monitor colori 14" media risoluz.	L. 440.000
44 - Monitor 14" alta risoluzione EGA	L. 770.000
45 - Monitor colori 14" AR. Multisinch	L. 1.100.000
46 - Monitor monocromatico 12"	L. 180.000
47 - Monitor monocromatico 14" DUAL basc.	L. 230.000

MOUSE

48 - Modem asinc. comp. HAYES 300/1200 baud	L. 247.000
49 - Modem est. comp. HAYES 300/1200 baud	L. 275.000
50 - Mouse meccanico per interfaccia	L. 88.000

STAMPANTI OLIVETTI

51 - Olivetti DM 100	L. 400.000
52 - Olivetti DM 282	L. 650.000
53 - Olivetti DM 292	L. 900.000
54 - Olivetti DM 296	L. 1.200.000
55 - Nastri stampanti di ogni marca	

STAMPANTI CITIZEN

56 - 120D 80 col. 120 cps. NLQ 20 cps	L. 363.000
57 - MSP1OE 80 col. 160 cps. NLQ 40 cps	L. 517.000
58 - MSP15E 132 col. 160 cps. NLQ 40 cps	L. 588.000
59 - MSP45E 132 col. 240 cps. NLQ 50 cps	L. 770.000

SOFTWARE PACCHETTI AZIENDALI

60 - Gestione distinta base	L. 600.000
61 - Gestione contabilità generale	L. 600.000
62 - Gestione magazzino	L. 200.000
63 - Gest. impegni - bolle - fatt/ne	L. 300.000
64 - Gestione portafogli effetti	L. 200.000
65 - Gestione statistica vendita	L. 200.000
66 - Gestione costo prodotto	L. 200.000

SOFTWARE PACCHETTI

67 - LOGISTIX (Foglio lavoro, grafici, dbase)	L. 200.000
68 - VOLKSWRITER 3 (Gestione Testi)	L. 400.000
69 - SUPER BASE (Data Base Relazionale)	L. 400.000
70 - LAYOUT (Integratore testi e grafica)	L. 300.000
71 - PORTEX (Agenda - Rubrica e Word Proc)	L. 300.000

OFFERTE SPECIALI

72 - Hard disk 20Mb MICROSCIENCE con controller e cavi	L. 445.000

Prezzi I.V.A. esclusa
Prenotazione tel. 06/4125486
Orario: 9.00 - 13.00 / 15.00 - 19.00

CONDIZIONI GENERALI DI VENDITA
Consegna a domicilio, prov. di Roma L. 30.000
Consegna altre zone, mezzo corriere
Pagamento contanti

Sconti quantità: da L. 1.000.000 a L. 2.000.000 5%
oltre L. 2.000.000 10%
Sabato mattina aperto

È UNA INIZIATIVA **UNIWARE S.R.L.** VIA CASAL DE' PAZZI, 82 00156 - ROMA TEL.: 06/4125486