di Pierluigi Panunzi

La gestione degli interrupt

prima parte

Eccoci arrivati dunque a parlare degli interrupt, che come ben noto rivestono una fondamentale importanza nella «vita» di un sistema dotato di microprocessore. Brevemente facciamo un po' di storia su questo argomento: fin dai primi microprocessori (vedasi l'8080 ed il 6502) gli interrupt hanno avuto un ruolo importantissimo, in quanto consentivano (e da qui il loro nome) di interrompere il normale svolgimento di un programma in istanti assolutamente non determinabili a priori, come suol dirsi completamente «asincroni» rispetto alla viceversa rigorosa successione degli eventi interni al micro, scandita come ben noto dal clock di sistema

Fin dagli inizi gli interrupt avevano la caratteristica di essere causati, innescati, da eventi esterni al processo in corso, eventi che però dovevano assolutamente essere tenuti in conto e gestiti: i primi micro erano dotati di un unico pin attraverso cui poter far arrivare un segnale di interruzione.

Il problema era invece di poter gestire più interrupt e non uno solo ed allora è stata adottata in seguito la strategia di associare, ad ogni tipo di interrupt che si potesse presentare, un ben determinato byte che il microprocessore legge dal data bus, byte che deve essere fornito dal dispositivo che genera l'interruzione e byte che serve proprio al micro per individuare la fonte dell'interruzione, allo scopo di gestirla correttamente.

Parallelamente perciò ai microprocessori sono sorti i cosiddetti «interrupt controller» capaci di gestire parecchi interrupt e di informare la CPU di quale evento esterno abbia richiesto l'interruzione.

Sappiamo che l'8086 può gestire fino a 256 interruzioni, se opportunamente aiutato da un interrupt controller e per dovere di compatibilità anche nell'80286 è stata mantenuta tale capacità anche se, come vedremo, la gestione avverrà in modo completamente differente (seppur ancora una volta in maniera trasparente e ciò invisibile da parte del programmatore).

Dicevamo dunque che l'interrupt è fondamentalmente un evento esterno: in realtà già dall'8080 un interrupt poteva essere scatenato anche da software, con un'apposita istruzione, la gloriosa RST («ReSTart»), diventata INT per l'8086 e per il 286; però bisogna far attenzione al fatto che con le istruzioni citate NON si genera un interrupt che poi il micro gestirà, ma bensì più semplicemente viene effettuata la chiamata alla routine di gestione dell'interrupt, «simulando» in un certo qual modo il comportamento della CPU in risposta ad un interrupt.

Secondo questa seconda interpretazione, un interrupt «software» è diventato un evento completamente «sincrono», nel senso che si sa già quando debba accadere.

Per giunta poi il fatto che con un'istruzione di INT si attiva una subroutine è servito splendidamente ai progettisti dei sistemi operativi per creare routine facilmente richiamabili con istruzioni di appena 2 byte a differenza di una normale chiamata tramite CALL «inter-segment» che viceversa richiede 5 byte: ci stiamo riferendo alle varie «INT 10H», «INT 13H», ecc. (tanto per citarne due),

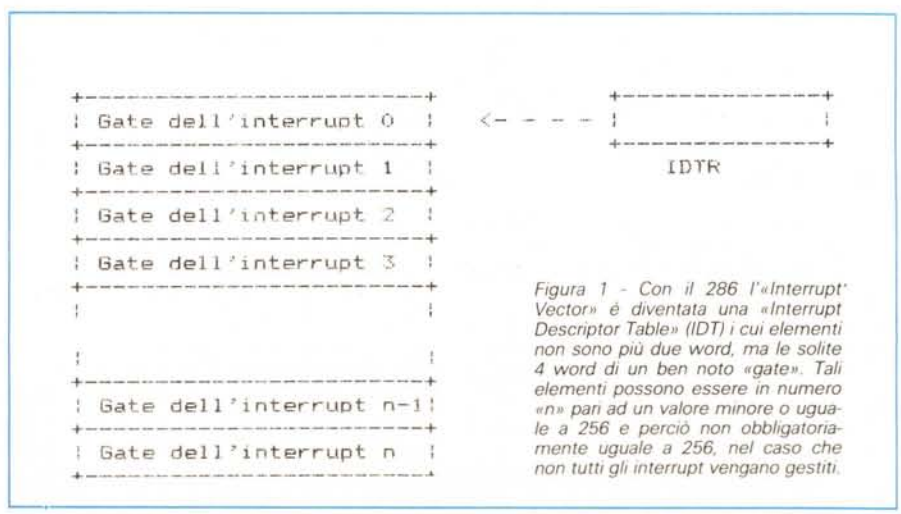

Figura 1 - Con il 286 l'«Interrupt Vector» è diventata una «Interrupt Descriptor Table» (IDT) i cui elementi non sono più due word, ma le solite 4 word di un ben noto «gate». Tali elementi possono essere in numero «n» pari ad un valore minore o uguale a 256 e perciò non obbligatoriamente uguale a 256, nel caso che non tutti gli interrupt vengano gestiti.

usate dall'MS-DOS per gestire rispettivamente il video e le unità a dischi.

In questo caso dunque l'istruzione INT ha perso completamente il suo legame con gli interrupt mentre è stata sfruttata la sua parentela con le CALL.

Con l'80286 agli interrupt viene associata un'altra funzione legata stavolta all'insorgere di errori o violazioni da parte di programmi: abbiamo più volte citato il fatto che «... se si ha una violazione o se l'istruzione vuole effettuare un'operazione illecita, allora viene generata una cosiddetta «exception», che segnala al sistema operativo l'insorgere di un problema...».

In questo caso dunque non c'è però un'istruzione che viene eseguita, ma viceversa è proprio la CPU che si accorge che l'istruzione in corso di esecuzione effettua un'operazione illecita: è dunque una «parte della CPU» che informa la restante parte che è successo qualcosa di grave.

In effetti quando viene generata un'eccezione significa il più delle volte che si è tentata una violazione di memoria, di livelli di privilegio, insomma è stata tentata un'azione che tentava di minare la ferrea ed invalicabile struttura basata sui privilegi e sull'inviolabilità di un task da parte di altri processi: praticamente in ognuna delle puntate di questa rubrica abbiamo visto che di motivi validi a scatenare un «exception» ce ne sono a bizzeffe e per tutti i gusti.

Visti dunque i «tre volti» di un'interrupt (evento esterno, evento software, exception), andiamo ora ad analizzare in quale modo il 286 li gestisce.

Gli interrupt

Dal momento che un interrupt (proveniente indifferentemente da una delle tre «fonti», tanto che d'ora in poi trascureremo di differenziarle, se non dove serva) viene gestito da un'apposita routine, ecco che, alla luce di quanto abbiamo imparato riguardo ai «task», si possono avere due situazioni ben differenti, gestite (inutile dirlo) in modo diverso.

Infatti si può avere una routine di gestione dell'interrupt appartenente allo stesso contesto del task che era in corso di esecuzione, come pure può capitare che tale routine appartenga ad un contesto completamente differente: va da sé che nel primo caso il passaggio dal task alla routine avviene in maniera alquanto indolore, mentre nel secondo caso si dovrà sottostare a tutte le regole di un vero e proprio «task switching».

Mentre il primo caso sarà in genere il più rapido (in termini di tempi di esecuzione), viceversa il secondo caso risulta indispensabile laddove ci sia la necessità di «isolamento» tra il task in corso di esecuzione e l'interrupt che cerca di farsi strada.

Comunque a prescindere da questa notevole differenza, dicevamo prima che la CPU può identificare uno di 256 interrupt, ad ognuno dei quali è associato il ben noto «interrupt vector».

Mentre nell'8086 (ed ancora nel 286 in «Real Mode») tale vettore era posto rigorosamente in memoria a partire dalla primissima locazione (posta all'indirizzo 0000:0000) e constava di 256 coppie di word indicanti l'indirizzo (in forma «offset:segment») delle singole routine di gestione degli interrupt, nell'80286, (in «Protected Mode», laddove regna sovrano il criterio dei privilegi e della virtualità della memoria) la questione è leggermente più complicata. Infatti nel nostro caso invece di un interrupt vector avremo una «Interrupt Descriptor Table» (in gergo IDT) e cioè una solita e ben nota struttura posta in memoria ed i cui elementi sono nient'altro che dei «gate»: tali elementi potranno essere un qualsiasi numero minore o uguale a 256, a seconda del numero effettivo di interrupt che desideriamo gestire e comunque gli elementi relativi ad interrupt non utilizzati si indicheranno con un valore nullo nell'«Access Rights Byte».

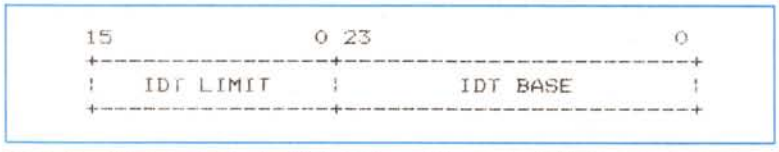

Figura 2 - Struttura interna dell'IDTR («Interrupt Descriptor Table Register»), simile ad analoghi registri relativi alle tabelle di descrittori: il campo LIMIT indica la lunghezza della tabella stessa, mentre il campo BASE rappresenta l'indirizzo della prima cella di memoria in cui è allocata la tabella in questione.

In ogni caso si avranno le consuete protezioni relative ai vari tentativi di effettuare operazioni illecite, quali ad esempio accedere ad un elemento oltre al limite della tabella dei descrittori, oppure nel caso che il descrittore a cui si fa riferimento sia di tipo non ammissibile.

Inoltre si ha che i primi 32 interrupt (da 00H ad 1FH) sono riservati all'Intel (particolare che i progettisti dell'MS-DOS non hanno tenuto bene in conto), mentre i restanti sono completamente a disposizione dell'utente.

Abbiamo detto che tale IDT è posta in memoria: analogamente alle altre «Descriptor Table», l'allocazione della IDT sarà indicata da un apposito (nuovo) registro.

Facendo riferimento alla figura 1, vediamo che l'IDT è posta in memoria ad un indirizzo fisico posto in un particolare registro interno alla CPU, l'DTR («Interrupt Descriptor Table Register»), la cui struttura interna può essere osservata in figura 2, mentre ogni singolo elemento della IDT è come detto un gate e cioè una ben nota struttura formata da 4 word, che possiamo infine vedere in figura 3.

Abbiamo detto che gli elementi di una IDT sono dei gate: in particolare possono essere dei tre tipi seguenti

— «interrupt gate»
— «trap gate»
— «task gate»

che analizzeremo singolarmente.

Per ora diciamo che nel caso dei primi due gate, l'interrupt verrà gestito nell'ambito del task correntemente in corso di esecuzione, mentre viceversa

nel caso di un «task gate» si avrà ovviamente un «task switch».

Gli interrupt provenienti dall'hardware

Sappiamo già che un evento esterno può interrompere la CPU per mezzo di appositi segnali presentati sui due pin INTR e NMI del microprocessore: al primo pin verranno convogliati (in genere tramite un «interrupt controller» di cui abbiamo già parlato) tutti gli interrupt di tipo «mascherabile» e che cioè potranno essere «mascherati» (inibiti) dal software, semplicemente resettando il flag IF («Interrupt Flag») presente appunto nel word contenente i Flag; al secondo pin saranno invece connessi circuiti che gestiscono eventi in generale catastrofici quali la caduta di tensione dell'alimentazione oppure un errore nelle memorie (evento questo per l'appunto sfruttato nei PC).

In quest'ultimo caso l'interrupt si dice appunto «non mascherabile» (NMI sta infatti per «Non Maskable Interrupt») nel senso che non sarà possibile inibirlo via software, ma anzi avrà la priorità assoluta di esecuzione, anche se nell'istante in cui viene generato era in corso di esecuzione una routine di interrupt e magari altri interrupt erano stati già memorizzati («pending») ed erano perciò in attesa di essere eseguiti.

Analogamente a quanto succede con l'8086, nel caso di interrupt mascherabili, il programmatore può scegliere, all'interno della routine di gestione dell'interrupt, se permettere o meno l'attivazione di un'altra routine di interrupt a seguito di un altro evento esterno, seguendo o meno un criterio di priorità: nel caso in cui si decidesse di inibire la gestione di un nuovo interrupt (ma fatta salva la sua «memorizzazione»), allora basterà azzerare il citato flag per mezzo dell'istruzione CLI, per poi resettarlo (con l'istruzione STI) allorché si ammetta l'intrusione di nuovi interrupt, in genere proprio prima dell'istruzione IRET di chiusura della routine in corso di esecuzione.

Abbiamo già detto che l'interrupt controller fornirà alla CPU (durante la fase di «Interrupt Acknowledge») un byte sul DATA BUS, indicante il numero corrispondente all'interrupt che si vuole gestire: viceversa nel caso dell'NMI, il numero dell'interrupt è già prefissato a 2 (come nell'8086) tanto è vero che nel caso dell'NMI manca (perché inutile...) la fase di «Interrupt Acknowledge».

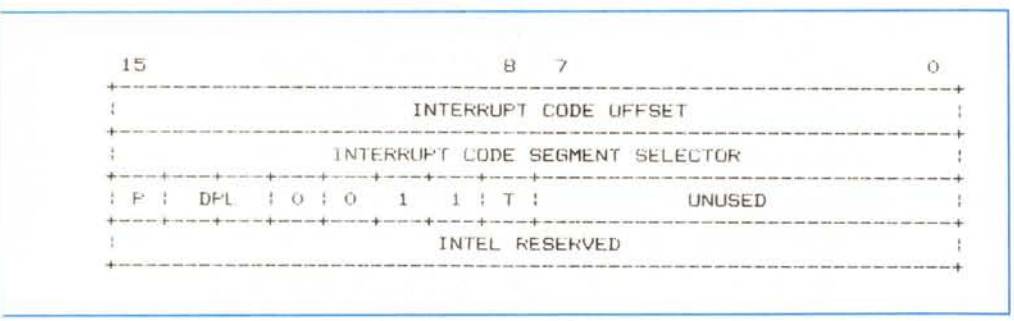

Figura 3 - Struttura di un «Interrupt Gate» e di un «Trap Gate», a seconda del valore del campo T: se T=1 allora si tratta di un «Trap Gate».

Gli interrupt software

Abbiamo già citato l'istruzione INT (in genere «INT n» con «n» pari al numero di un interrupt, «INT 3» usata in genere per gestire i breakpoint ed «INTO» per gestire gli overflow) nonché gli interrupt generati a seguito di violazioni alla struttura di privilegi e protezioni tipica del 286 in «Virtual Mode».

In entrambi i casi tali interrupt non sono interrompibili ed in particolare quelli generati da condizioni di errore, sono associati ad una word posta sullo stack, indicante il tipo di errore generato ed hanno un vettore di interrupt compreso tra 0 e 31: per tale motivo l'Intel sconsiglia l'uso dell'istruzione INT «n» con valori di «n» inferiori a 32, avvertimento ancora una volta disatteso dai viceversa solerti progettisti dell'MSDOS, per compatibilità verso l'8086.

«Interrupt Gate» e «Trap Gate»

Facendo riferimento alla figura 3, vediamo la struttura dei due gate in esame, ormai standardizzata, nella quale innanzitutto il campo «T» serve a distinguere formalmente un «Trap Gate» (per T=1) da un «Interrupt Gate» (T=0): la differenza tra l'uno e l'altro gate risiede nel differente comportamento nei riguardi di ulteriori interrupt eventualmente generati.

In particolare l'«Interrupt Gate» indica che la procedura sarà da attivare con gli interrupt automaticamente disabilitati (come se ci fosse una CLI), mentre viceversa l'accesso ad una routine di gestione di un interrupt tramite un «Trap Gate» avviene senza alterare lo stato del flag di interrupt (IF): si noti che «non si altera lo stato» e non «si setta lo stato», come potrebbe venire in mente dato che viceversa con l'«Interrupt Gate» lo stato viene «resettato». Ciò vuol dire ovviamente che l'utente è libero, nel caso dei «Trap Gate», di abilitare o meno gli interrupt successivi (o meglio, non dimentichiamocelo, «abilitare o meno la gestione degli interrupt successivi», dal momento che comunque gli interrupt vengono memorizzati).

Analizzando dunque la figura 3, riconosciamo gli oramai ben noti campi o comunque campi di facile interpretazione:

— i campi «Interrupt Code Offset» e «Interrupt Code Segment Selector» indicano l'indirizzo iniziale della routine di gestione dell'interrupt, indirizzo iniziale espresso come coppia «selector: offset» e non come un indirizzo fisico;

— il campo «P» indica al solito la presenza (P=1) o meno (P=0) in memoria della routine di gestione, con tutte le conseguenze del caso se P=0...;

— il campo DPL rappresenta sempre il livello di privilegio del descrittore;

— del campo «T» abbiamo già detto ed infine;

— dei due campi «UNUSED» ed «INTEL RESERVED» non c'è nulla da aggiungere se non che è vivamente sconsigliato scriverci qualcosa, in previsione di un passaggio del software al 386 il quale, come vedremo nella naturale evoluzione della presente rubrica, viceversa utilizza tali campi (seppur in parte, demandando ad un prossimo 80486 la completa utilizzazione dei residui campi...).

Tornando un istante sul campo DPL, c'è da aggiungere che questo viene gestito solo nel caso di istruzioni «INT n», «INT 3» ed «INTO» (ed al solito il CPL, il «Current Privilege Level», del programma in corso di esecuzione deve risultare minore o uguale del DPL), mentre viceversa viene ignorato ovviamente nel caso di interrupt esterni: non è ben chiaro infatti come un evento esterno possa avere un proprio livello di privilegio!

Con questo terminiamo questa puntata ed anticipiamo che nella prossima analizzeremo in dettaglio le innumerevoli operazioni che la CPU compie (in maniera trasparente, è ovvio) quando inizia a gestire un interrupt per mezzo di un gate. MC

S.C.COMPUTERS s.a.s.

via E.Fermi 4, 40024 Cast.S.Pietro T. (BO)
tel. 051 - 943500 (2 lin.ric.aut. + fax)

Confrontate attentamente queste configurazioni e questi prezzi con altre inserzioni di questa rivista:

PC XT: L. 1.550.000

8088/2, clock a 5 e **8 MHz**, zero wait, 512 Kbytes di RAM, 1 **drive** da 360Kb 1 **Hard Disk** da 20 Mbytes, **Controllers** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Scheda **Hercules** Hi Res, **Tastiera** Avanzata 101/2 tasti, Cabinet tipo ATcon chiave, Aliment. 200 W, **Monitor** 12" TTL Hi-Res, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

E' inoltre disponibile un modello analogo, ma con clock a 10 MHz e microprocessore NEC V20

PC AT: L. 2.150.000

80286, clock a 6 e **12 MHz**, zero wait, 512 Kbytes di RAM, 1 **drive** da 1,2 Mbytes, 1 **Hard Disk** da 20 Mbytes, **Controller** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Porta **Seriale** Doppia RS 232, Scheda **Hercules** Hi Res, **Tastiera** Avanzata 101/2 tasti, Cabinet con chiave, Aliment., **Monitor** 12" TTL Hi-Res, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

PC AT/VGA: L. 3.150.000

80286, clock a 6 e **12 MHz,** zero wait, 512 Kbytes di RAM, 1 **drive** da 1,2 Mbytes, 1 **Hard Disk** da 20 Mbytes, **Controller** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Porta **Seriale** Doppia RS 232, Scheda **VGA, Tastiera** Avanzata 101/2 tasti, Cabinet con chiave, Aliment., **Monitor 14" Philips per VGA**, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

PC 386/40: L. 5.800.000

80386, clock **16 MHz** , 1 Mbytes di RAM, 1 **drive** da 1.2 Mbytes, 1 **Hard Disk** da **40 Mbytes** veloce (29 ms), **Controller** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Porta **Seriale** Doppia RS 232, Scheda **Hercules** Hi Res, **Tastiera** Avanzata 101/2 tasti, Cabinet, Aliment., **Monitor** 12" TTL Hi-Res, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

TOSHIBA

PROMOZIONALE sul **T 1600:**
prenotateVi SUBITO !!!

COPROCESSORI & RAM

RAM 100-120-150 ns, 64K -256K-1Mb	TELEFONARE
8087	L. 249.000
8087/2	L. 320.000
80287	L. 349.000
80287/8	L. 535.000
80287/10	L. 610.000
80387/20	L. 1.060.000

Condizioni particolari per
☆☆ RIVENDITORI !!! ☆☆

Tutti i prezzi sono da intendersi IVA 19% esclusa, ma comprendono un anno di garanzia TOTALE f.co ns. sede. Siamo in grado di spedire la merce a mezzo corriere in tutta Italia entro 36 ore dal ricevimento di un acconto pari al 10% dell'importo totale tramite Vaglia Telegrafico. Spedizione gratuita, se effettuata a mezzo posta.

OFFERTE del MESE

Nuovissima Stampante **EPSON LQ 500**, 24 AGHI, 80 col., 150 car.per sec., doppia velocita' in Letter Quality rispetto alla LX800, grafica bidire-z., 8 Kb di buffer, foglio singolo e modulo continuo, int. parallela: . . **L. 699.000**

Mouse emulaz. MICROSOFT e Mouse System Mouse, con porta-mouse, tappetino e lo splendido Dr.HALO III originale e mauali . . . . . **L. 120.000**

Hard Disk 20 Mbytes . . . . . **L. 390.000**

TELEFAX TOSHIBA:

O M O L O G A T I ! ! ! ! !
nuovissimi: TF 111, 211, 311,
a partire da L. 2.500.000

COMMODORE - ATARI

❋❋❋ *OFFERTE NATALIZIE !!!* ❋❋❋
I migliori prezzi, ma con
GARANZIA ITALIANA.
☎☎ TELEFONATECI !!! ☎☎