di Pierluigi Panunzi

La gestione degli interrupt

quarta parte

In questa quarta parte parleremo in dettaglio delle cosiddette «exceptions» e cioè di quegli interrupt che vengono generati «internamente» dal microprocessore quando questo incontra un'istruzione o una particolare situazione che tenderebbe a violare uno dei tanti meccanismi di protezione: abbiamo detto «tenderebbe a violare» in quanto è praticamente impossibile riuscire intenzionalmente a prevaricare queste sofisticate protezioni, almeno a livello di programma applicativo, dato che invece a livello di sistema tutto o quasi tutto è possibile

Le «exceptions»

Abbiamo già parlato parecchie volte delle «exceptions», dicendo genericamente che se, ad esempio, ad un certo punto si va ad indirizzare al di là dei limiti imposti per una segmento, allora viene generata una exception.

In particolare con tale termine si intende la generazione di un interrupt al quale in alcuni casi viene associata nello stack una word che serve meglio ad identificare il tipo di errore riscontrato.

Dal momento che si tratta di un interrupt, nasce ancora una volta la questione di come viene materialmente innescato tale interrupt, se attraverso un «interrupt gate», un «trap gate» oppure un «task gate».

Infatti nei primi due casi è lo stesso task ad aver generato l'errore che può gestirne le conseguenze, mentre nell'ultimo caso sarà un altro task a gestire l'exception: ciò non deve suonare strano, dal momento che in alcuni casi si può (sempre che il sistema operativo sia d'accordo) generare un errore voluto, la cui gestione non viene demandata al sistema operativo ma viene operata «in casa».

In altri casi poi l'exception non è strettamente legata ad un errore vero e proprio: è il caso in cui si vuole accedere ad un segmento «non presente» in memoria in quell'istante: ecco che in questo caso non si tratta di un errore (in quanto difficilmente un sistema operativo siffatto avrebbe una vita lunga), ma di una «richiesta di servizio», consistente nel richiedere il caricamento in memoria del segmento incriminato, sempre che tutte le condizioni al contorno siano soddisfatte.

Viceversa in caso di errore vero e proprio il sistema, innescato dall'apposita routine di interrupt, prenderà i dovuti provvedimenti, tra i quali c'è pure quello del «restart» del programma dal punto in cui si era generato l'errore, ovviamente dopo aver preso le precauzioni del caso.

Anche questo può sembrare strano perché in generale si potrebbe pensare che un programma che ha generato errore non debba essere «restart-ato»: in pratica però ci sono dei casi in cui il «restart» oltre che possibile è obbligatorio (quale l'esempio già citato di un task che accede ad un segmento non presente in memoria: in questo caso, dopo aver caricato tale segmento, è più che naturale che il controllo ritorni proprio al task!), mentre in generale sarà il sistema operativo a decidere se l'errore è restartabile, al solito, ponendo valori opportuni in registri coinvolti nella generazione dell'errore.

Abbiamo finora visto nelle varie puntate che esistono parecchie possibilità di violazioni ed errori, che in alcuni casi sono state accorpate in un unico interrupt, avendosi poi la necessità di specificare meglio il tipo d'errore grazie alla parola presente nello stack.

In figura 1 vediamo la tabella sinottica indicante l'elenco delle «exception» legate ai primi interrupt: a tal proposito ricordiamo che l'Intel definisce come «reserved» gli interrupt che vanno da INT 00 ad INT 1F.

Da questa tabella dunque riconosciamo alcune delle exception di cui abbiamo parlato più e più volte, quale su tutte la numero 0D, la «general potection» («GP»): dalla tabella vediamo pure, nelle colonne indicate con «rest.» e «err.», rispettivamente la possibilità di «restart» del programma al termine dell'esecuzione della routine di gestione dell'errore e la presenza o meno di una word di descrizione nello stack. In entrambi i casi un «*» indica rispettivamente la restartabilità e la presenza della word.

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
In questa tabella sono indicate tutte le «exceptions» previste nel 286 ed i relativi numeri di interrupt. Un «» in colonna «rest.» indica che la routine caduta in errore è «restart-abile», mentre un «*» nella colonna «err.» indica che all'exception è associata una parola sullo stack.*

Prima di analizzare le «exceptions» in dettaglio, diciamo che in genere la word di errore è il «selector» di un segmento incriminato (laddove l'exception coinvolge un segmento), oppure un valore specifico negli altri casi.

C'è da dire infine che questa word viene scritta per ultimo nello stack che risulterà «attivo» quando verrà eseguito il gestore dell'errore: quest'ultimo, in tal modo si troverà subito la «word» (con un semplice POP) e non dovrà andare viceversa a cercarla chissà dove tra i vari stack relativi ai vari processi e/o ai vari livelli di privilegio...

L'INT 0 e l'INT 5

Si tratta di interrupt provenienti da errori legati al cattivo funzionamento di due istruzioni, rispettivamente la DIV (o IDIV) e la BOUND:

— per quanto riguarda la DIV e la IDIV tale errore viene generato per un overflow sulla divisione e cioè se il quoziente non può essere rappresentato nel risultato oppure se il divisore è nullo (caso classico!);

— invece la BOUND dà errore nel caso che l'indice di un array esca dai limiti prestabiliti.

In entrambi i casi le istruzioni sono restartabili nel senso più generale del termine e cioè che non vengono nemmeno eseguite.

L'INT 1, l'INT 2 e l'INT 3

In questo caso non si tratta in realtà di exception, ma di meccanismi particolari che riguardano rispettivamente:

— la possibilità di eseguire in «single step» una certa routine;

— la gestione del Non Maskable Interrupt (NMI);

— la gestione di Breakpoint.

Di questi argomenti in parte abbiamo già parlato fin dalle prime puntate, oppure, al solito, richiederebbero grande spazio e nozioni che non ci interessa porre in evidenza. Passiamo perciò oltre...

L'INT 4

In questo caso si tratta dell'interrupt generato dall'istruzione INTO, allorché a seguito di una operazione aritmetica risulta settato il flag OF («Overflow Flag»).

Ricordiamo a tal proposito che il flag di overflow non è una specie di flag di «riporto» come invece il nome potrebbe far pensare: si tratta invece di un flag settato nei casi in cui, ad esempio sommando due quantità positive, il risultato esce negativo.

Per i distratti diciamo che ciò non è dovuto certo ad un momentaneo impazzimento dell'altrimenti insospettabile 80286 (ma anche l'8086 et similia si comportano alla stessa maniera!), ma all'impossibilità di rappresentare il risultato dell'addizione nello stesso modo in cui sono rappresentati gli addendi: ad esempio se lavoriamo con byte dotati di segno (i cui valori rappresentabili vanno da −128 a 127) possono capitare situazioni di errore.

Ad esempio sommando 1 a 127 (che è il massimo numero positivo esprimibile, la cui codifica binaria è 01111111), non si ottiene 128, in quanto non rappresentabile, ma bensì −128, rappresentato correttamente dal valore binario 10000000: il bit più significativo, che negli addendi era nullo, indicando quantità positive, è ora diventato 1, viceversa indicando un valore negativo.

Comunque la stessa cosa può succedere con le sottrazioni, le moltiplicazioni e le divisioni, allorché si abbia per forza di cose a che fare con quantità dotate di segno: moltiplicando due quantità positive il risultato potrebbe venire negativo solo perché tale valore non è più rappresentabile e perciò viene coinvolto l'incolpevole bit di segno; analogamente il discorso vale se si moltiplicano due quantità negative che possono originare un risultato ancora negativo e così via per tutte le possibili combinazioni...

L'INT 6

Ecco dunque un qualcosa di nuovo, non presente nei modelli «precedenti» al 286: si tratta di un'exception generata allorché il microprocessore incontra un'istruzione il cui op-code non è tra quelli riconosciuti.

In effetti questo è un fatto nuovo: fin dai tempi dei vari 8080 e Z80 il comportamento dei microprocessori in tali casi risultava in alcuni casi imprevedibile, mentre in altri (è il caso dello Z80) comportava l'esecuzione di istruzioni lecite ma non annoverate tra quelle ufficiali.

Eppoi sui codici sconosciuti si basano le istruzioni di nuovi microprocessori di una stessa famiglia oppure «compatibili», per il primo caso l'esempio lampante è il 286 che sfrutta «buchi» nella tabella degli op-code dell'8086 per trovare posto alle nuove istruzioni (anche il 386, come vedremo, farà così), mentre per il secondo caso, l'esempio che citiamo è quello del ben noto microprocessore V20 della NEC, che risulta una super-copia dell'8088, nel senso che prevede si tutte le istruzioni dell'88, ma ne aggiunge altre più qualche altro meccanismo «sfizioso» sui quali già fin d'ora promettiamo di ritornare, visto il notevole uso di tale processore nei «cloni» o «cinesi». Il tutto, prima di affrontare l'argomento 80386...

L'INT 7 e l'INT 9

Si tratta di due exception legate all'uso di un processore ausiliario, in generale il coprocessore matematico 80287: su tale argomento non entreremo nei dettagli, in quanto ci porterebbe alquanto fuori strada.

Diciamo solamente che l'INT 7 è generato allorché il 286 incontra un'istruzione specifica del 287, ma che non risulta presente a livello hardware nella scheda del computer, mentre il secondo è generato allorché, in presenza di un 287, un'istruzione fa riferimento a dei dati posti al di fuori dei limiti viceversa stabiliti per un segmento: in tal caso è ancora ovviamente il 286 ad accorgersi della richiesta illecita ed altrettanto ovviamente lancerà l'interrupt senza eseguire l'istruzione incriminata.

Successivamente verrà eseguita la routine di gestione di tale exception, ma non si potrà mai restart-are il programma: questo perché in realtà la routine di gestione dell'errore dovrà provvedere a resettare il coprocessore ed in tal caso è giocoforza che il programma ricominci daccapo e certo non può riprendere dal punto interrotto in quanto si sono perse ovviamente tutte le informazioni numeriche all'interno del 287, appena resettato.

Bisogna dire tra l'altro che il 287 DEVE essere resettato in quanto se per caso il 286 incontrasse successivamente un'istruzione del 287 (ad eccezione della FNINIT che viceversa è proprio il «reset» del 287), allora si avrebbe un cosiddetto «deadlock»: la CPU attende una risposta dal coprocessore matematico (ed in particolar modo il segnale di «not busy», indicante che il 287 ha eseguito le sue operazioni), mentre quest'ultimo ancora stava aspettando i dati precedenti.

Tutto questo detto in parole povere: in realtà il meccanismo è alquanto complesso e richiederebbe cognizioni che ci porterebbero a sconfinare in argomenti nuovi e non previsti.

Tra l'altro c'è pure da dire che non necessariamente il task interrotto dall'interrupt 9 è proprio quello che tentava di eseguire l'istruzione incriminata (strano ma vero!!) ed allora dovrà sempre essere cura del sistema operativo tener conto istante per istante di quale è il task che ha per ultimo usato (o meglio, tentato di usare) il coprocessore matematico: se il task interrotto non era quello incriminato, allora può essere tranquillamente restart-ato, mentre se si tratta proprio di quello incriminato allora abbiamo visto che ciò deve risultare impossibile.

L'INT 8

Tale interrupt viene generato allorché una singola istruzione genera due diffe-

renti exception, ad esempio nel caso di un programma a livello di privilegio 3 (quindi un applicativo) che cadrebbe in errore per una «general protection» su di un segmento non presente e perciò genererebbe anche una «segment not present exception» (l'INT 11 che analizzeremo dopo): in tal caso viceversa viene generato un unico INT 8.

Ed è proprio per gestire tale interrupt che deve essere usato un task gate per avere una corretta gestione: infatti la routine di gestione, un task, per quanto detto, potrà identificare il task che ha generato l'errore grazie al «back link» presente nel proprio TSS.

Per chi non ricordasse che cosa è il «back link» consigliamo di rileggersi le ultime puntate della rubrica dove abbiamo parlato dei «Task State Segment» (TSS).

In particolare l'interesse posto nel «back link» è proprio quello al quale si trova l'istruzione che ha generato il «double fault», ma che ancora non è stata eseguita.

Invece nel disgraziatissimo caso in cui oltre alle due exception ne venisse generata una terza, allora l'80286 entrerebbe in stato di «shutdown», dal quale si può uscire solo con un reset o un NMI.

Dal momento che ci troviamo in casi particolarmente strani e catastrofici, è divertente segnalare che l'NMI potrebbe non riuscire nell'intento di schiodare il 286 dall'impasse, se per caso si riscontrassero errori anche durante la routine di gestione dell'NMI...

In tal caso solo un reset rimetterebbe tutto a posto, ma con le ben note conseguenze di far ricominciare tutto daccapo, tra l'altro facendo uscire il processore dal Protected Mode.

L'INT 10

Tale interrupt è generato, durante un «task switch», allorché si fa riferimento ad un TSS non valido, a causa di una delle seguenti possibilità, delle quali più o meno avevamo già parlato in occasione della descrizione del meccanismo di switch:

— il valore del limite del TSS è minore di 43 e perciò non consente la corretta gestione del TSS stesso;
— il selector della LDT non è valido oppure l'LDT non è presente in memoria;
— il selector dello Stack Segment punta al di fuori dei limiti prescritti;
— lo Stack Segment non è un segmento in cui sono possibili operazioni di scrittura;
— i livelli di privilegio dello Stack Segment, del suo selector nonché il CPL («Current Privilege Level») non sono di valore adeguato;
— il selector del Code Segment punta al di fuori dei limiti consentiti;
— ancora il selector non fa riferimento effettivamente ad un Code Segment;
— i livelli di privilegio legati al Code Segment non sono adeguati;
— i selector relativi al DS o all'ES puntano al di fuori dei limiti della tabella dei segmenti;
— infine i segmenti DS o ES non sono segmenti in cui possono essere effettuate operazioni di lettura.

Lo spazio è tiranno...

... e non ci permette di continuare ad analizzare i rimanenti INT B, C e D, dei quali parleremo in dettaglio nella prossima puntata, insieme ad altre considerazioni di carattere generale. Con il che potremmo anche terminare l'argomento «80286», per passare ad altro, ma non anticipiamo troppo i tempi. MC