di Pierluigi Panunzi

I livelli di privilegio e le protezioni

Fin dalla prima puntata di questa serie di articoli abbiamo parlato del fatto che il microprocessore 80286 consente di gestire le risorse (la memoria, i dati ed i programmi) in modo «protetto», avendo la possibilità di inibire l'accesso ad alcune risorse da parte di processi che non ne hanno il «privilegio»: in questa puntata parleremo in modo dettagliato di cosa si intende nel «mondo dell'80286» con il concetto di «privilegio», inteso in generale come uno dei tanti «attributi logico-fisici» assegnati ad una data risorsa dal sistema operativo

Sappiamo dalle ultime puntate che ogni processo possiede la facoltà di accedere a due tipi di risorse di memoria, definite in genere con il termine di «spazio di memoria locale» e di «spazio di memoria globale»: sappiamo che a tale proposito ad ogni processo viene associata una coppia di tabelle (la LDT e la GDT) all'interno delle quali sono riportate tutte quelle informazioni atte ad individuare quali e quanti sono i segmenti di memoria accessibili al processo.

Senza scendere ancora una volta nei dettagli sui quali ci siamo già soffermati, ricordiamo perciò che nella LDT compaiono i «descriptor» dei segmenti «locali», mentre nella GDT compaiono i segmenti «globali», che fanno parte del «corredo» di risorse associate ad un certo processo: in particolare le risorse «locali», come dice il loro nome, sono di esclusiva proprietà del processo e non possono essere in alcun modo toccate da altri processi, così come il nostro processo non potrà andare a curiosare nell'ambito «locale» di altri processi a lui «paralleli».

Invece per quanto riguarda le risorse «globali», già dal nome deve essere ben chiaro che si tratta in generale di risorse condivise, comuni a tutti i processi e perciò supervisionate appunto dal sistema operativo che ne consentirà l'accesso all'uno o all'altro processo a seconda di opportuni criteri di «scheduling» definiti a priori ed insiti nel sistema operativo stesso.

Ovviamente al singolo processo non è minimamente consentito l'accesso a risorse propriamente locali del sistema operativo ed il tutto è regolamentato dal cosiddetto «Privilege Level» (livello di privilegio), che altro non è che un numero associato ad ogni processo ed in genere ad ogni risorsa e rappresenta una sorta di «lasciapassare», di «visto di ingresso», attribuito ad insindacabile giudizio del sistema operativo.

I «Privilege Level»

In particolare i livelli di privilegio sono (solamente) quattro e sono numerati da 0 a 3, dove il livello «0» è quello tipico del supervisore e dove, scendendo di gerarchia, il livello «3» è associato ai processi d'utente, ed in particolare al nostro programma.

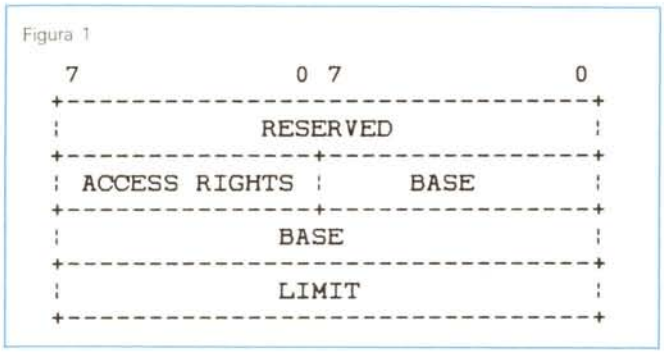
Ecco che perciò per accedere ai segmenti di memoria (che potranno contenere codice oppure dati) bisogna innanzitutto avere un livello di privilegio tale che ci consenta di accedervi, ma poi sappiamo che, anche se ne abbiamo il privilegio, scatteranno altre soglie di protezione quali gli «access rights» ed il controllo dei limiti fisici.

Per quanto riguarda gli «access rights», sappiamo che questi definiscono la modalità di accesso ad un determinato segmento (ad esempio accesso a sola lettura, oppure accesso per esecuzione, oppure accesso in lettura e/o scrittura, ecc.), che viene confrontata con il modo con cui il programma vuole gestire i dati del segmento in esame ed inoltre, se l'accesso è di tipo lecito, appositi meccanismi controlleranno che il programma non possa uscire al di fuori dei limiti del segmento usato, sia esso di dati che di istruzioni da eseguire.

Ancora una volta poniamo l'accento sul fatto che sia i metodi già visti di protezione («access rights» e controllo dei limiti fisici dei segmenti), sia quello che stiamo ora analizzando (il livello di privilegio), sono realizzati interamente dall'hardware interno del microprocessore e perciò non sono in alcun modo «scardinabili» dal punto di vista software.

Abbiamo già detto che i livelli variano tra 0 e 3: in particolare risorse a livello 0 (generalmente dati e programmi del sistema operativo) non sono accessibili da programmi aventi altri livelli di privilegio, mentre viceversa programmi a pri-

Figura 1



Struttura di un Segment Descriptor.

vilegio possono accedere a risorse di privilegio inferiore (cioè di valore numerico maggiore). In generale un programma avente un certo privilegio può accedere a risorse aventi lo stesso livello di privilegio oppure un valore numerico maggiore, secondo una scala gerarchica molto stretta.

La regola mnemonica è presto fatta: tanto più è alto il privilegio (minor valore numerico), tante più sono le risorse a cui si può accedere e viceversa.

A pensarci bene, poi, il fatto di avere quattro livelli di privilegio è una notevole estensione di quelli che sono i dettami canonici di un computer, che in genere prevede l'esistenza di due livelli o meglio «stati» (quello «supervisore» e quello «di utente»): un programma d'utente non è in grado di sapere cosa fa e non fa il supervisore il quale viceversa ha proprio il compito di controllare tutto.

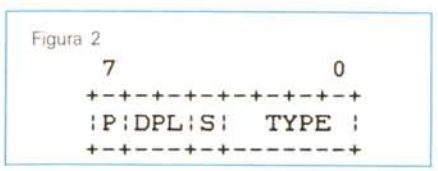
Con quattro livelli di privilegio si possono attribuire, oltre a quello riservato al supervisore, ben tre livelli ai processi d'utente, secondo un meccanismo a «strati» che sono stati subito paragonati alla struttura di una cipolla: sfogliando gli strati più esterni (livelli a basso privilegio) si arriva mano mano al centro, nel cui nucleo è posto quello che disolito si chiama «kernel» (dall'inglese «nocciolo»), e cioè l'insieme di routine e di dati che costituiscono il sistema operativo.

A complicare ulteriormente la faccenda, in alcuni casi il livello di privilegio è una caratteristica «costante», un attributo associato ad una risorsa, mentre in altri casi tale attributo è «dinamico», mutevole istante dopo istante, a seconda della «storia» del processo in corso di esecuzione.

In particolare il privilegio associato ad un descrittore di un segmento (appartenente ad una LDT o ad una GDT) è assegnato da parte del sistema operativo una volta per tutte all'istante di creazione del descrittore stesso e cioè sia quando il sistema è creato con il cosiddetto «System Builder», sia quando il programma che usa tale segmento viene caricato dalla memoria di massa alla memoria fisica del sistema stesso.

Invece il livello di privilegio di un «task» (che abbiamo incontrato più volte, anche se non abbiamo ancora descritto formalmente in dettaglio) è un valore che varia dinamicamente istante dopo istante, a seconda del livello di privilegio del segmento di programma che è correntemente in esecuzione.

Figura 2



I quattro campi in cui è suddiviso l'«Access Rights Byte».

Ciò può sembrare a prima vista un po' strano, ma basta rifletterci su un poco: abbiamo più volte detto infatti che un programma (o meglio «task») possiede anche delle parti «globali», in genere routine di sistema che come tali non debbono per forza essere duplicate per ogni programma che ne faccia richiesta d'uso.

Ma tali routine ad un certo istante devono essere eseguite (ovviamente!) da parte di un task in generale a livello di privilegio inferiore (ricordarsi che ciò implica un livello numerico maggiore...): esistono a tal uopo delle ben precise regole che consentono appunto il passaggio da un segmento ad un certo privilegio ad uno di privilegio maggiore (quello in cui ci sono le routine di sistema condivise) e per questo semplice fatto ecco che il nostro task cambierà per forza di cose il suo livello di privilegio.

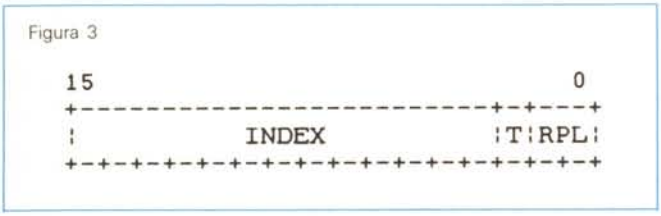
Tornando ai «segment descriptor», sappiamo che il livello di privilegio viene posto all'interno dell'«access rights byte» (il byte dei diritti di accesso), nel campo chiamato DPL («Descriptor Privilege Level»): nella figura 1 riportiamo l'oramai ben nota struttura di un segment descriptor e nella figura 2 l'«esplosione» dei campi dell'«access rights byte».

Invece per quanto riguarda il livello dinamico di un task, si ha che tale valore viene posto nel campo detto CPL («Current Privilege Level»), rappresentato dai due bit meno significativi del CSD (che ricordiamo essere il «Code Segment Descriptor»), che riportiamo in figura 3.

A questo punto dovrebbe se non altro essere ben chiaro il perché del nome CPL associato al valore «corrente» del privilegio di un task.

In realtà è stato indicato, all'interno

Figura 3



La struttura di un Segment Register Selector.

dello schema rappresentativo di un segment selector, il termine RPL invece che il già citato CPL: in realtà, come vedremo, si tratta in questo caso di un «Requested Privilege Level» (livello di privilegio richiesto), che, se abilitato, diventerà un «Current Privilege Level».

I passaggi tra un livello e l'altro: primi cenni

Ecco che perciò la regola generale che consente l'accesso a risorse poste ad un certo livello di privilegio da parte di programmi ad un altro livello di privilegio viene ora migliorata dicendo che ad un certo task viene concesso l'uso di risorse aventi un livello di privilegio minore o uguale (maggiore o uguale in termini numerici) del CPL e cioè del valore attuale del livello di privilegio.

Ciò non deve ovviamente stupire in quanto evidentemente una routine di sistema (supponiamo posta al livello 1), chiamata da un task avente il livello originario (quello iniziale d'utente) posto a 3, deve poter usare i propri dati, magari posti al livello 2, altrimenti inaccessibili al nostro task.

Per quanto riguarda l'accesso a segmenti di codice, questo è consentito solo verso livelli di privilegio uguale, mentre i «salti» a privilegi maggiori devono avvenire attraverso particolari meccanismi (dei quali parleremo in dettaglio) coinvolgenti i cosiddetti «gate».

Questo fatto sottolinea ancor più la possibilità da parte di un programma ad accedere con relativa semplicità a segmenti di codice di uguale livello di privilegio, mentre per effettuare «salti di qualità», sarà necessario impegnare maggiori «risorse del microprocessore», il che comporterà come primo risultato evidente un allungamento dei tempi di esecuzione di istruzioni quali i salti e le chiamate a subroutine.

Detto questo per i segmenti di codice, occupiamoci dei segmenti di dati per dire che anche per questi ultimi vale il controllo che il livello di privilegio sia inferiore a quello del programma che vuole utilizzare i dati.

Ma in entrambi i casi c'è un fatto nuovo che viene tenuto in conto prima ancora del controllo dei livelli di privilegio. Si tratta del fatto che con l'80286 i segmenti hanno solo quattro tipi di accesso consentito, che potrà apparire alquanto limitante, se paragonato alla libertà di accesso di un 8086/88.

In particolare le quattro possibilità di accesso ad un segmento sono le seguenti:

— **"RO"** = **"Read Only"** e cioè a sola lettura
— **"RW"** = **"Read/Write"** e cioè a lettura-scrittura
— **"EO"** = **"Execute Only"** e cioè a sola esecuzione
— **"ER"** = **"Execute/Read"** e cioè a sola esecuzione e lettura.

Fermo restando che questi sono gli unici quattro attributi che si possono e debbono assegnare ad un segmento (già dalla fase di definizione del segmento, a livello «Assembler», prima di procedere all'assemblaggio del nostro programma), ecco che l'accesso solo in alcuni casi sarà possibile, mentre sarà vietato in altri anche se il privilegio lo consentirebbe: ricordiamo infatti che questo tipo di controllo viene effettuato «prima» di testare i privilegi, che perciò sono ininfluenti in questo caso.

Ecco che perciò volendo caricare il registro di segmento DS (relativo cioè a dati), non potremo farlo con un segmento di tipo «EO», così come pure accade nel caso del registro di segmento ES (che per certi versi è simile al già citato DS). Nel caso che il registro in questione è l'SS, allora ad esso viene consentito solo l'accesso a segmenti di tipo «RW» mentre vengono negati tutti gli altri tipi, ovviamente... (ad esempio pensate all'utilità di uno stack in cui non si possa scrivere, ma solo leggere!).

Infine per quanto riguarda le possibilità di caricamento del registro CS con il «selector» di un segmento e perciò per ciò che concerne la possibilità di «saltare» ad un altro segmento, si ha che la cerchia è ristretta ai soli segmenti che consentono l'esecuzione del codice e cioè i segmenti con attributo «EO» e «ER», essendo, vietati i segmenti puri di dati (di tipo «RW») e contenenti costanti (di tipo «RO»).

Nella tabella A abbiamo sintetizzato la situazione indicando con «1» il fatto che l'accesso è consentito e con «0» l'impossibilità: per accesso ad un segment register intendiamo in questi casi la possibilità di caricamento del registro stesso con opportune istruzioni.

Con questo terminiamo questa puntata, mentre nella prossima ritorneremo più in dettaglio sul concetto di «Requested Privilege Level», sul quale abbiamo nettamente sorvolato. MC

Tabella A

registro di segmento	tipo di segmento RO	RW	EO	ER
CS	0	0	1	1
DS	1	1	0	1
ES	1	1	0	1
SS	0	1	0	0

AMPEX

La comunicazione è un fatto importante. La sicurezza, la velocità e la chiarezza dell'informazione sono dati essenziali per un terminale. I Terminali Ampex offrono una vasta scelta di soluzioni per colloquiare in diverse emulazioni (VT 100 e VT 220, per citare solo le più famose) e un modello con tastiera AT compatibile.

distributore HARDWARE BUSINESS SYSTEMS s. r. l.

SEDE: Via G. Jannelli, 218 - 80131 Napoli - Tel. 081/254913-465501 - Fax 081/7701694
FILIALE: Via A. Ambrosini, 177 - 00147 Roma - Tel. 06/5425161

IL VALORE AGGIUNTO AL TUO BUSINESS