di Pierluigi Panunzi

Il Protected Mode

A partire da questa puntata ci addentreremo sempre di più all'interno del microprocessore 80286, con lo scopo di conoscerne tutti i suoi segreti, rappresentati da avanzate caratteristiche a metà strada tra il software e l'hardware e che sono notevolmente innovative rispetto agli altri microprocessori a 16 bit. Inizieremo a conoscere più da vicino il cosiddetto «Protected Mode» che, come già abbiamo accennato, consente al programmatore di sfruttare appieno tutte le caratteristiche di gestione della memoria virtuale, dalla sua allocazione nell'ambito della memoria fisica alla protezione vera e propria. Introdurremo inoltre concetti nuovi quali i «descrittori di segmenti» per arrivare poi ai «task» che, come vedremo, sono una generalizzazione spinta del concetto di «modulo di programma». Ma andiamo con ordine, anche perché gli argomenti non sono molto semplici: diamo dunque un'occhiata alla «Memory Management»

La gestione della memoria virtuale

Stiamo dunque parlando di «Protected Mode»: in questa modalità di funzionamento il programmatore (o meglio il programma stesso) non vede più la memoria del sistema così come si era abituati con l'8086, laddove la corrispondenza tra l'indirizzo di una singola locazione di memoria e l'effettiva allocazione nell'ambito della memoria fisica era di «uno a uno», ma viceversa deve sottostare alla regola che un qualsiasi indirizzo richiesto della memoria deve essere vagliato, «filtrato» e convertito da apposite strutture hardware-software, che genereranno l'indirizzo fisico, effettivo, della locazione desiderata.

In modo protetto infatti ogni riferimento alla memoria passa attraverso il meccanismo di gestione-protezione, sia che si faccia riferimento ad una singola cella di memoria, sia che si faccia riferimento ad un segmento (di codice o di dati indifferentemente) oppure ancora sia che si faccia riferimento a strutture ben note quali ad esempio lo stack.

Come suol dirsi, in questo caso la memoria fisica ed il meccanismo che permette di accedervi in ogni situazione sono completamente «trasparenti» per l'utente, e cioè del tutto «invisibili», come se praticamente non esistessero: della loro presenza ci si accorge in pratica in base all'esito di certe operazioni oppure in base al fatto che certe altre non sono proprio possibili, almeno avendo in mente il funzionamento dell'8086. Altra conseguenza e caratteristica peculiare del meccanismo di gestione della memoria è la seguente: mentre nel «mondo 8086» la memoria a cui si faceva riferimento era la vera e propria memoria «fisica», in un certo senso statica e prefissata una volta per tutte, ora nel «mondo 80286» si ha a che fare con una memoria completamente dinamica, prefissata solo in alcune parti fondamentali, e perciò in genere completamente svincolata dalla quantità di memoria «fisica» esistente nel sistema.

Viene privilegiata ed esaltata in questo caso la «rilocabilità» di un modulo software, anche laddove il contenuto del programma stesso non sembrerebbe a prima vista rilocabile, come pure si possono risolvere brillantemente problemi legati alla presenza di più processi attivi in un unico sistema, i quali processi hanno bisogno ognuno di una certa quantità di memoria e di risorse che devono essere per forza di cose condivise, magari perché la memoria di sistema effettivamente a disposizione è inferiore alla somma delle quantità di memoria viceversa richieste dai singoli processi.

In tal modo il singolo processo («task») in ogni istante in cui è attivato vede sempre presenti la memoria e le risorse di cui ha bisogno, senza sapere che cosa succede effettivamente istante dopo istante: con opportuni sistemi operativi che sfruttano le notevoli possibilità offerte dall'80286 si potranno ad esempio attuare meccanismi di «swapping» (tanto cari ai sistemi di minicomputer ed ai main-frame) che prevedono l'allocazione effettiva della memoria solo allorché il processo ha bisogno di risorse, con conseguente accantonamento di quanto era presente in precedenza nella memoria di massa, ad esempio il disco fisso.

Comunque non ci allontaniamo troppo da quanto ci siamo proposti di dire, ma ogni tanto è inevitabile effettuare qualche volo qua e là, dal momento che gli argomenti sono tanti ed alcuni particolarmente stimolanti e suggestivi.

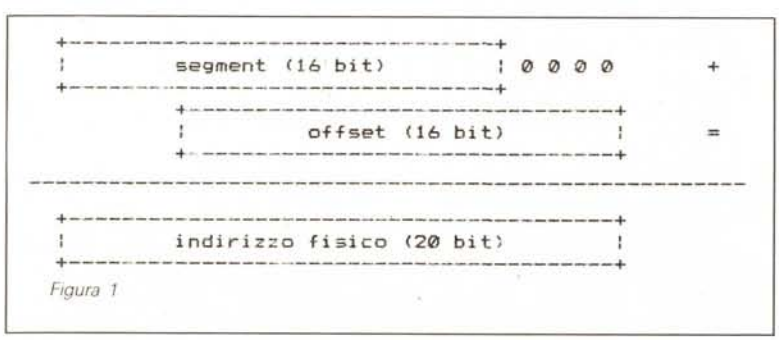


Figura 1

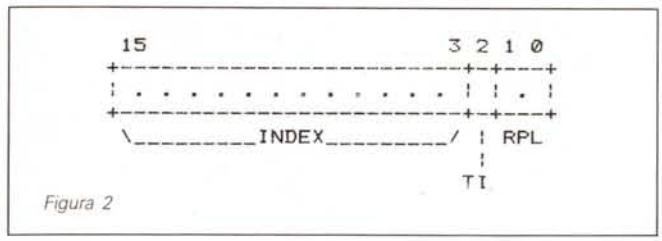


Figura 2

Il meccanismo della memoria virtuale

Abbiamo dunque detto e forse lo ripeteremo anche altre volte in quanto si tratta di un concetto fondamentale, che il singolo programma, facendo riferimento ad esempio ad una locazione di memoria, non sa assolutamente a quale effettivo indirizzo fisico tale cella sia allocata e soprattutto non ne ha alcuna possibilità di saperlo, a meno di non conoscere a fondo (noi e non il programma) il sistema operativo e perciò i suoi vari settaggi oppure a meno di non andare a leggere (sempre noi, non certo il programma) i valori che il microprocessore pone sul bus degli indirizzi istante dopo istante, cosa alquanto complessa...

Comunque a parte queste considerazioni, è rimasto ovviamente dal «mondo 8086» il meccanismo di indirizzamento di una certa locazione di memoria per mezzo di una coppia di valori a 16 bit: nell'8086 tali word erano dette «segment» ed «offset» e servivano ad individuare perfettamente la cella di memoria, il cui indirizzo a 20 bit (come sappiamo bene) era ottenuto shiftando a sinistra di 4 posizioni il valore di «segment» e sommandoci dunque il valore dell'«offset», per un totale di 1 Mbyte indirizzabile.

Si aveva dunque una situazione come appare in figura 1. Invece nel caso dell'80286 si ha una situazione leggermente differente: le due quantità a 16 bit formano un'unica quantità a 32 bit che rappresenta dunque un indirizzo virtuale, gestito dal «Memory Manager» all'interno del microprocessore.

Sarà quest'ultimo modulo ad effettuare i calcoli opportuni per tirar fuori l'indirizzo fisico a partire dalla quantità a 32 bit e ciò avviene in modo differente a seconda se il micro si trovi in «Real Mode» o in «Protected Mode».

Ancora una volta in «Real Mode» non si ha alcuna differenza con quello che conosciamo riguardo l'8086, mentre con il «Protected Mode» si ha un differente significato nel contenuto della parte alta dei 32 bit.

Per comodità di analisi, piuttosto che riferirsi ad una quantità a 32 bit, è preferibile ancora una volta scinderla in due parti a 16 bit, delle quali la meno significativa è ancora un «offset» (senza altri inghippi sotto...), mentre la parte più significativa viene detta «segment selector» (o più semplicemente «selector»), in quanto rappresenta per l'appunto un selettore all'interno di una tabella: ma andiamo con ordine, osservando in figura 2 la struttura del «selector».

In essa notiamo la presenza di ben tre campi, rispettivamente il campo «INDEX», il campo «TI» («Table Indicator»), ed il campo «RPL» («Requested Privilege Level»). A parte il campo «INDEX» gli altri due campi non promettono niente di buono... Del campo «RPL», poi, è ancora prematuro parlare.

Rimbocchiamoci dunque le maniche: il più innocuo campo «INDEX», assieme al bit «TI», non è niente altro che un puntatore (indice) all'interno di una tabella detta «Descriptor Table», tabella che contiene delle informazioni (descrizioni) importanti riguardo a tutti i segmenti, di dati o di codice o di stack che siano.

Di queste tabelle ne esistono due (selezionabili a seconda del valore del campo «TI») ed hanno ognuna fino a 8192 elementi: ecco che infatti i 13 bit costituenti il campo «INDEX» permettono di selezionare uno degli 8192 elementi della tabella individuata da «TI».

Queste due tabelle vengono dette GDT («Global Descriptor Table», quando TI = 0) e LDT («Local Descriptor Table», quando TI = 1) e proprio i loro nomi derivano dal fatto più generale che tutto lo spazio di memoria virtuale a disposizione di un processo (il Mbyte di cui sopra) viene considerato per metà «globale» e per metà «locale».

Con il termine «globale» si intende uno spazio di memoria utilizzato da procedure di sistema, routine di libreria, servizi vari nonché da dati comuni, che come tali possono essere condivisi da tutti i processi, senza che però si abbiano inutili duplicazioni in quanto ad esempio due processi differenti hanno bisogno per la loro esecuzione di una stessa routine.

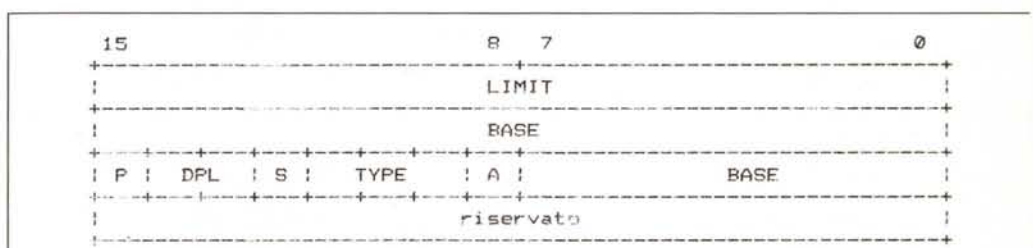


Figura 3

D'altro canto però questo fatto comporta che le risorse in comune devono essere protette contro accessi indebiti da parte di processi che viceversa non potrebbero accedervi: questi sono comunque problemi che si risolvono con i vari sistemi di protezione che vedremo nel seguito.

Invece con il termine «locale» si intende quell'altra metà del Mbyte virtuale a cui ogni singolo «task» può accedere e che però è inaccessibile ad altri task: in tal modo ogni singolo task ha una sua zona «privata» dove sono allocati il codice ed i dati, inviolabile da altri task (diciamo così, dello stesso livello), mentre tutti i task possono far riferimento a risorse per l'appunto «globali», uniche, condivise e non duplicate.

Ecco che dunque il singolo task potrà fare riferimento a segmenti (di codice o

di dati, indifferentemente) posti sia nell'area «globale» sia nell'area «locale»: per ogni segmento a cui si farà riferimento ci sarà un apposito «selector» che andrà a puntare (tramite il campo «INDEX») al relativo elemento all'interno della GDT (se il campo TI è nullo) o della LDT (se TI vale 1), laddove troverà le caratteristiche del segmento desiderato.

Tutto questo perché abbiamo detto e ripetuto che un qualunque segmento viene allocato dinamicamente in memoria ed allora ci deve essere scritto da qualche parte nella memoria innanzitutto l'ampiezza del segmento stesso (che sappiamo poter essere da 1 byte a 64 kbyte) e poi ovviamente l'indirizzo fisico di dove è stato allocato il segmento in quel particolare istante.

Siamo perciò arrivati alla descrizione del singolo elemento della GDT o LDT, elemento che è una struttura formata da 8 byte (vedi figura 3). Alcuni campi di tale struttura potranno essere compresi solo a tempo debito, mentre alcuni di essi sono proprio quanto ci aspettavamo: analizziamoli uno per uno.

— Il campo etichettato con «LIMIT», posto nei byte 0 ed 1, rappresenta l'ampiezza in byte del segmento, ampiezza che può variare tra 1 byte e 64 k, valore quest'ultimo rappresentato forse da un campo «LIMIT» pari a 0.

— Il campo indicato con «BASE», formato dai tre byte 2, 3 e 4 della struttura, con il byte 4 come più significativo, rappresenta invece l'indirizzo fisico del segmento nell'ambito di 16 Mbyte di memoria, l'indirizzo a cui è stato attualmente allocato il segmento in esame.

— Il campo formato dai cinque sottocampi («P» che sta per «Present», «DPL» che sta per «Descriptor Privilege Level», «S» che sta per «Segment descriptor», «TYPE» che indica il tipo di segmento ed «A» che sta per «Accessed») e posto nel byte 5 della struttura prende il nome di «Access Rights Byte» e cioè «byte dei diritti di accesso»: non è certo questo il momento di parlare di tale byte e dei singoli campi che lo compongono, anche perché così appesantiremmo la già ponderosa puntata, mentre diciamo che già dal nome «diritti di accesso» si può avere un vago sentore del fatto che il suo contenuto riguarderà la possibilità da parte del programma di accedere al segmento in esame, sia per problemi di privilegio legati alla protezione, sia per problemi di modalità di accesso al segmento, sia per problemi legati all'eventuale assenza del segmento in memoria.

— L'ultimo campo relativo ai byte 6 e 7, in cui abbiamo scritto «riservato», deve infine essere sempre posto a 0, in modo da poter mantenere la compatibilità futura con l'80386.

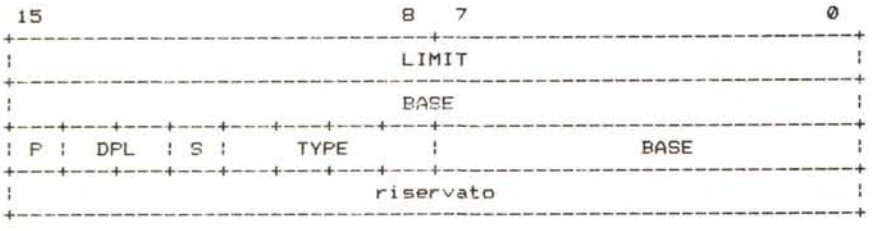


Figura 4

Complichiamo ulteriormente la faccenda... La tabella GDT, che come sappiamo contiene i descrittori dei segmenti globali, viene effettivamente ed ovviamente posta in memoria ed occuperà una certa zona di essa, a partire da una certa locazione e per una certa lunghezza (non è detto infatti che tutti gli 8192 elementi siano sempre presenti, anzi...): in definitiva occuperà un certo segmento le cui caratteristiche (appunto l'indirizzo fisico iniziale e l'ampiezza) vengono finalmente poste all'interno di un apposito registro interno della CPU (una novità...), detto sistema operativo ed il suo valore dunque si suppone rimanga inalterato fino.... allo spegnimento del sistema: dovrebbe essere infatti chiaro che la struttura a cui punta, la GDT, deve essere un «punto fermo» all'interno della memoria, in quanto contiene informazioni preziosissime e vitali per tutto il sistema ed i processi che su esso convivono. Viceversa abbiamo visto che ogni singolo task deve poter avere a disposizione un'area di memoria virtuale destinata a lui, «privata» o meglio «locale», ma data la fragilità e temporaneità di un task (cosa su cui torneremo...) nella memoria, ecco che la LDT di un singolo task deve seguire le sorti del processo al quale appartiene.

Se infatti in un certo istante le risorse di un certo processo sono state «swappate» e cioè «tolte di mezzo» dalla memoria fisica e salvate perciò sull'unità di memoria di massa, il tutto per far posto al task correntemente in uso, che viceversa ha bisogno di memoria in quantità, ecco che non avrebbe più senso mantenere in memoria anche la LDT del processo «scomparso».

Perciò istante per istante, a seconda del processo che correntemente è in esecuzione, esisterà da qualche parte in memoria la sua LDT ed allora il suo indirizzo fisico ed ampiezza saranno memorizzati in un altro registro interno della CPU detto LDTR («Local Descriptor Table Register»): ma i valori da porre in tale registro dove si ricavano?!

È presto detto: sono posti all'interno di particolari elementi della GDT, come se si trattasse di segmenti globali di codice o dati, ma viceversa (grazie al bit «S» posto a 0) consentono di identificare ogni LDT di ogni singolo task come un segmento a parte.

Oltre al bit «S» posto a 0 l'elemento appare leggermente differente da quello visto per i segmenti «segmenti», soprattutto per quel che riguarda il fantomatico «Access Rights Byte» ed infatti l'elemento della GDT in esame ha la struttura che appare in figura 4 dove appunto il campo «TYPE» è ora a 4 bit.

Anche questa nuova struttura ha un suo nome particolare e dato che servirà anche per altre caratteristiche del sistema, ecco che è stata denominata con «Descriptor Table Descriptor», che nel nostro caso ha il nome più appropriato di «LDT Descriptor», che se non altro è un po' più chiaro...

In definitiva, e con questo terminiamo, l'elemento in questione (l'«LDT Descriptor») punterà ad un particolare segmento di memoria, che non è altro che la LDT relativa a un certo task. Cosa conterrà questa LDT? Ovvio, no!? Conterrà i descrittori dei segmenti usati localmente dal nostro diabolico task, descrittori che non sono altro che l'elenco di informazioni utili ad identificare il nostro segmento dal punto di vista delle possibilità di accesso, della sua ampiezza in memoria e della sua effettiva allocazione in memoria.

Prima di congedarci, poniamo un quesito subdolo: come accidenti farà l'80286 a farci indirizzare la cella «ALFA» nell'altrimenti innocua istruzione

```
MOV AX, ALFA
```

senza che il sistema nervoso del povero programmatore subisca danni irreparabili?!

Nella prossima puntata sveleremo l'arcano e cioè il meccanismo complesso, ma «trasparente» di conversione tra indirizzo virtuale e fisico. MC

Ultimobyte

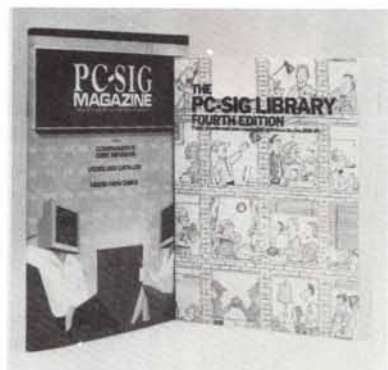
Abbiamo raggiunto Quota 1000

The PC-SIG Library è la più grande biblioteca di programmi al mondo per PC Olivetti, IBM e compatibili. Abbiamo ormai raggiunto la fatidica "Quota 1000" con una scelta che non teme confronti: utilities, giochi, linguaggi di programmazione, spreadsheet, WP, corsi, applicazioni finanziarie, grafica e altro.

La pubblicazione bimestrale *Shareware*, che viene inviata gratuitamente a tutti i nostri Associati, riporta le ultime novità, recensioni e commenti. Ogni disco, indipendentemente dal contenuto, costa 18.000 lire (IVA compresa). Il catalogo commentato, in Inglese, costa 25.000 lire.

Ecco che aria tira ad alta quota

☐ **710 INSTACALC** Spreadsheet residente in memoria
☐ **711 BONZOWARE** Giochi per adulti
☐ **712 JAPANESE FOR BUSINESS AND TRAVEL** Corso di Giapponese
☐ **715 NEWSBASE** Database specializzato per l'editoria
☐ **722 COMPOSER** Editor musicale su tre ottave
☐ **723 SUPER PINBALL** Collezione di 5 giochi del flipper
☐ **726 GOALSEEKER** Per scoprire i numeri che soddisfano un dato obiettivo: funziona con Lotus, Multiplan, Supercalc4 e VP Planner
☐ **727 POWERMENU** Shell per il Dos
☐ **731 LOCATE2** Per cercare stringhe nei files
☐ **734 EXTENDED DOS** Aggiunge prestazioni al Dos
☐ **750 LABEL MASTER** Gestione delle Mailing List
☐ **755 CANTONESE TUTOR** Corso di Cinese
☐ **761 THE IMP SHELL** Ambiente per lo sviluppo di sistemi esperti
☐ **763 FINGER PAINT** Programma di disegno per CGA e Hercules
☐ **764 COMPASS** Database, Word Processor, Spreadsheet, Contabilità, Calendario e Mailing List: 5 moduli integrati
☐ **765 GALAXY** Word Processor velocissimo e facilissimo
☐ **780 BRIDGEPAL** Il gioco del Bridge
☐ **788 IMAGE 3D FOR EGA** Disegno tridimensionale per la scheda EGA
☐ **791 POKER AND ULTIMA21** Poker e Blackjack
☐ **794 COMPOSER** Editor grafico musicale a una sola voce
☐ **798 PROMENU** Creazione di menu
☐ **814-815 MODULA2 TUTORIAL** Corso sul linguaggio Modula2
☐ **816-817 TURBO C TUTORIAL** Corso sul linguaggio Turbo C
☐ **819 CROSSWORD CREATOR** Per chi ama le parole crociate
☐ **822 FILE COMMANDO** Utilities per la gestione dei files
☐ **826 ADVENTURE ADDICTION** Collezione di 5 programmi di avventura (solo testo)
☐ **828 EDRAW** Per disegnare schemi, diagrammi a blocchi e circuiti stampati
☐ **830-831 WAMPUM** Implementazione del linguaggio dBase III
☐ **833-834 GRAPHTIME II** Programma di disegno per la scheda Hercules
☐ **836 DISK COMMANDO** La nostra risposta alle Norton Utilities
☐ **837-838 HOME MONEY MANAGER** Contabilità familiare
☐ **840 SHORTCUT** Shell per il Dos residente in memoria
☐ **841 MANDELBROT MAGIC** La geometria dei frattali
☐ **842 BOX** Per creare maschere di input **cal-**
☐ **846-847 LOTUS LEARNING SYSTEM** Come usare Lotus 1-2-3
☐ **851-852 STAR CATALOGUE** Per gli appassionati di astronomia
☐ **861-862-863 STATMATE/PLUS** Analisi statistica
☐ **864-865 CATALIST** Per gestire liste di indirizzi ed etichette di spedizione
☐ **870 HGCIBM** Emulatore della scheda CGA sulla Hercules
☐ **873 THE WINDOW BOSS-C** Come si gestiscono le finestre in C
☐ **883 XXXPERT** Per creare programmi diagnostici basati su domanda e risposta
☐ **885 MISC UTILITIES** Raccolta di utilities per programmatori
☐ **891 WHEEL OF MISFORTUNE AND SOLITAIRE** Giochi



Oltre 100.000 copie vendute nel mondo. 4° Edizione rinnovata e ampliata

☐ **893 PRIVATE LINE AND WEAK LINK** Cifratura dei dati e collegamento PC-to-PC sulla porta seriale (fino a 115.000 baud)
☐ **895 SCOUT AND SIMS** Gestione dei files su disco e creazione di menu: entrambi residenti in memoria

La Membership in regalo

Oggi, acquistando il volume The PC-SIG Library *più 5 dischi a vostra scelta diventate automaticamente e gratuitamente Associatiper un anno. Con sole 115.000 lire vi assicurate la pubblicazione bimestrale di aggiornamento al catalogo, nonché il diritto ad uno sconto sull'acquisto di altri dischi.*

Compilate subito il tagliando e speditelo oggi stesso. Non dovete obbligatoriamente scegliere tra i titoli sopra elencati: esaminate il catalogo a casa vostra e decidete con tutta calma.

ULTIMOBYTE S.r.l. - Via Aldo Manuzio, 15 - 20124 Milano
Ordini telefonici: 02/65.97.693
Tutti i prezzi esposti comprendono l'IVA. Aggiungere all'importo di ogni ordine il contributo fisso di L. 4.000 per spese di spedizione.

☐ **SI** aderisco alla vostra proposta di Membership
☐ Inviatemi a L. 115.000 "The PC-SIG Library", la Newsletter e 5 dischetti. Scelgo: _____ _____ _____ _____ _____
cod. 1 cod. 2 cod. 3 cod. 4 cod. 5

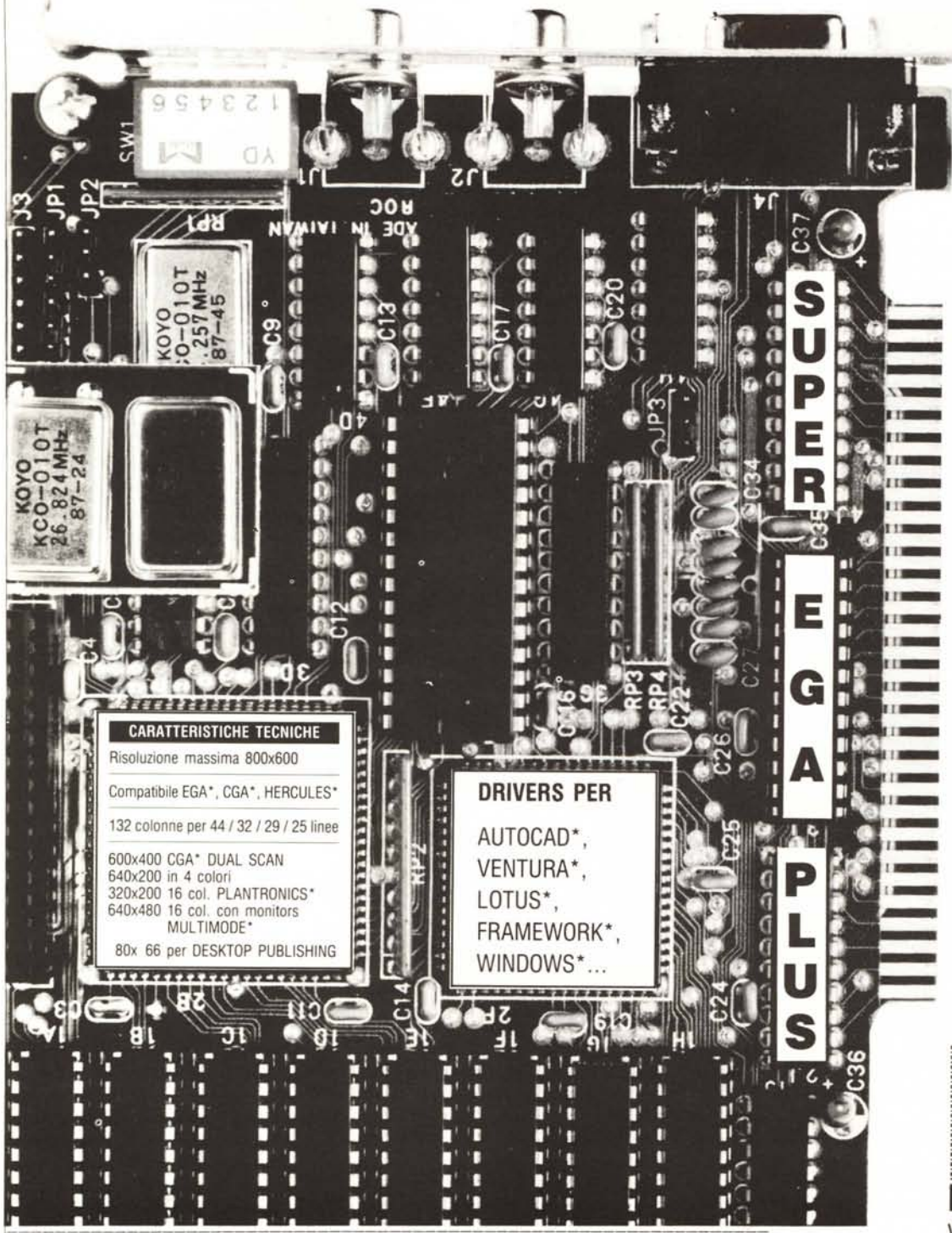
☐ A semplice richiesta e senza ulteriori spese mi invierete i rimanenti _____ dischetti che mi spettano.

☐ **NO** non desidero diventare Socio. Rinuncio alla Newsletter e allo sconto.
Inviatemi comunque _____
Totale da pagare L. _____ + L. 4.000 = L. _____

☐ Allego assegno/vaglia postale
☐ Pagherò al postino in contrassegno

NOME _____ COGNOME _____
VIA _____ CITTÀ _____ (_____)
CAP _____ P.IVA/Cod. Fisc. _____
(solo se si desidera fattura)

435.000 MOTIVI PER COMPRARLA
800x600 MOTIVI PER POSSEDERLA



DESIDERO RICEVERE INFORMAZIONI RELATIVE A:

☐ SUPEREGA PLUS 800x600	L. 435.000
☐ P.EGA 640x480 (come SUPEREGA ma solo fino 640x480)	L. 345.000
☐ ANKO MOUSE 6000 (compatibile MICROSOFT* SERIAL e MOUSE SYSTEM MOUSE*)	L. 99.000
☐ SCHEDA ANALOGICO/DIGITALE 12 BIT	L. 204.000
☐ MODEM STANDARD HAYES* V21 - V23	L. 354.000

*) MARCHI REGISTRATI

AZIENDA _____

NOME E COGNOME _____

VIA _____ N. _____

CAP _____ CITTA' _____ PROV. _____

RIVENDITORE _____ UTILIZZATORE FINALE _____

N. TELEFONO (_____) _____

EXECUTIVE

VIA BOVARA, 16
22053 **LECCO** (CO)
TEL. (0341) 364706
FAX (0341) 365646