di Pierluigi Panunzi

La gestione della memoria

In questa puntata inizieremo a conoscere, guardandole da vicino, alcune caratteristiche interne del 286, caratteristiche che in un certo senso rappresentano una prima estensione di quelle corrispondenti dell'8086: cominceremo dunque a parlare di memoria, fisica e virtuale, e dei registri interni

Abbiamo detto la volta scorsa che il 286 già dall'aspetto è una notevole espansione dell'8086, a cominciare dal numero dei pin relativi all'Address bus, fermi restando i 16 bit per il Data Bus.

Mentre nell'8086 avevamo 20 bit di indirizzo, per un totale di 1 Mbyte tondo tondo di memoria indirizzabile, nel 286 i bit di indirizzo sono saliti a 24 consentendo ora un range di indirizzamento pari a 16 Mbyte: per un confronto, il 386 possiede 32 bit di indirizzo, che consentono la bellezza di 4 Gbyte (sì proprio Gigabyte) solo per quel che riguarda l'offset, mentre prendendo in esame anche il cosiddetto «selector» di altri 14 bit, si arriva a poter indirizzare la mostruosa quantità di 64 Tbyte, dove la T sta per «Tera» e cioè 1000 Giga, pari a 64 milioni di Mbyte!

Per divertimento vediamo di capire «quanti» sono 64 Tbyte. Al momento si trovano in commercio schede di espansione della memoria del PC dotate al massimo di 8 Mbyte di RAM: ebbene per coprire i 64 Tbyte ne occorrebbero la bellezza di 8 milioni...

È inimmaginabile lo spazio necessario a contenere questo «kit d'espansione», ma soprattutto l'alimentatore necessario ed il mulino a vento che provvede al raffreddamento del tutto!

Tornando al nostro «piccolo» 286 abbiamo parlato di 16 MByte. Sappiamo già che il 286 ha due modi di funzionamento, «protetto» e «reale»: in pratica con il «Real Mode» si ha la possibilità materiale di indirizzare i 16 MByte di memoria fisica, mentre i complessi meccanismi previsti dal «Protected Mode» consentono di espandere la memoria virtuale indirizzabile ad 1 GByte (prodigi della tecnologia...).

In particolare, proseguendo e completando la filosofia di gestione della memoria introdotta con l'8086, anche nel 286 si ha a che fare con segmenti di memoria, rappresentati anche qui da un insieme di byte di memoria posti ad indirizzi consecutivi, stavolta non più di ampiezza fissa (che era 64 kbyte), ma variabile a scelta del programmatore tra 1 byte (!) e 64 kbyte. Ora questi segmenti possono essere allocati in memoria in un qualsiasi punto (come già si aveva nell'8086), ma in questo caso esiste un'apposita tabella di 16384 elementi ognuno indicante tra l'altro l'indirizzo iniziale del segmento rispettivo: dato che questi segmenti possono essere dunque 16k ed ognuno può essere di 64 kbyte, al massimo ecco che arriviamo al Gbyte di memoria virtuale indirizzabile.

Ma come fanno a stare questi mille Mbyte in un massimo di 16 Mbyte di memoria effettivamente e «fisicamente» presente? È questa la domanda che ci si può porre a questo punto (per non pensare ai 64 Tbyte «sparsi» negli appena 4 Gbyte di memoria fisica del 386...), e la risposta è in prima analisi molto semplice: solo una parte dei 1000 Mbyte (sarebbero 1024 in realtà, anche se a questi livelli la normativa è ancora carente, basta capirsi) è effettivamente «presente» in memoria in un certo istante, mentre la rimanente è posta ad esempio in una periferica esterna quale un'unità a dischi fissi, pronta per essere «caricata» in memoria al momento opportuno.

Il bello è che il tutto è praticamente «trasparente» (inteso sempre come «invisibile») per l'utente, che non si accorge nemmeno di quanto sta succedendo: la tabella di «indirizzi di segmenti» di cui parlavamo consente, tra le tante informazioni in essa contenute, anche l'indicazione del fatto che il segmento desiderato è o meno presente in memoria e per far ciò è necessario, ovviamente, un semplice bit (il bit «P»). Nel caso in cui il segmento non sia presente, ecco che sarà cura del sistema operativo far sì che sia allocato in memoria, magari spostando dei segmenti preesistenti o eliminandoli temporaneamente dalla memoria (ovviamente resettando il bit «P» dell'elemento corrispondente al segmento in esame, nella tabella di cui sopra), con il che il tutto può proseguire.

Comunque mai come ora «segmen-

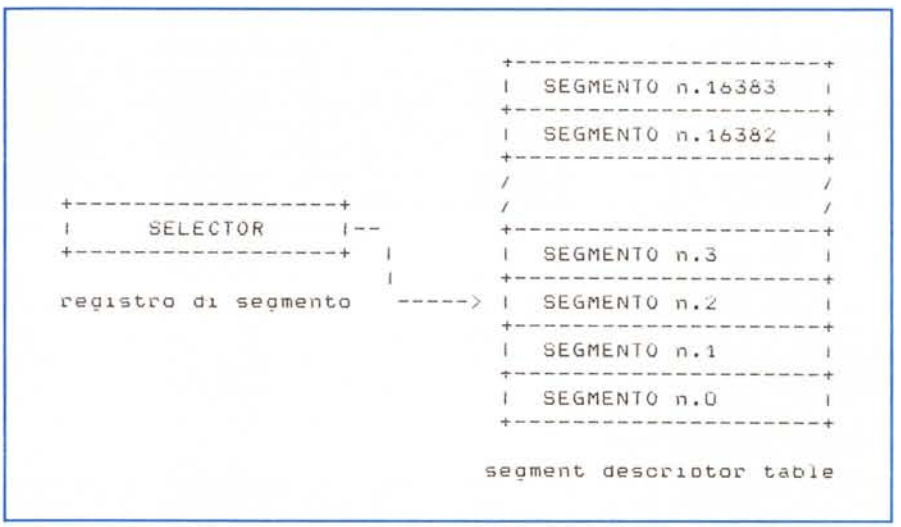


Figura 2 - Un registro di segmento è in realtà un puntatore («selettore») ad una di 16384 voci di una tabella. Ogni voce è ampia 4 byte e contiene informazioni riguardanti il segmento puntato.

to» deve essere considerato come unità logica di programmazione, tanto è vero che viene fortemente consigliato di scrivere i programmi in modo più modulare possibile, laddove dunque un modulo sarà un segmento di ampiezza opportuna: ben vengano dunque i progammi ricchi di subroutine da porre ognuna in un modulo, a tutto vantaggio dei tempi di esecuzione e di spazi di memoria occupati.

Tutto questo per quanto riguarda i segmenti di codice, che come vedremo potranno essere solamente letti ed eseguiti (addio ai codici auto-modificanti...), ma conferendo così al nostro programma un grado di protezione inviolabile; stesso discorso vale per i dati, che potranno essere organizzati in varie strutture di dati poste in segmenti differenti, magari alcune «locali» e perciò utilizzabili solo dal nostro programma (altra protezione) e magari altre invece condivisibili da più programmi, in ogni caso con possibilità di accedervi in lettura e/o scrittura. Infine abbiamo lo stack, anch'esso confinato in appositi segmenti, dai quali sarà ben difficile (è infatti impossibile...) ad esempio uscire per trabocco e perciò sconfinare in zone di dati o di codice, come accade per un qualsiasi altro microprocessore. In figura 1 vediamo un esempio di struttura della memoria fisica (a sinistra), nella quale sono presenti dei moduli di programmi, di dati e lo stack, mentre sulla destra abbiamo altri moduli (intesi sempre come «segmenti») appartenenti alla memoria virtuale, ma fisicamente non presenti.

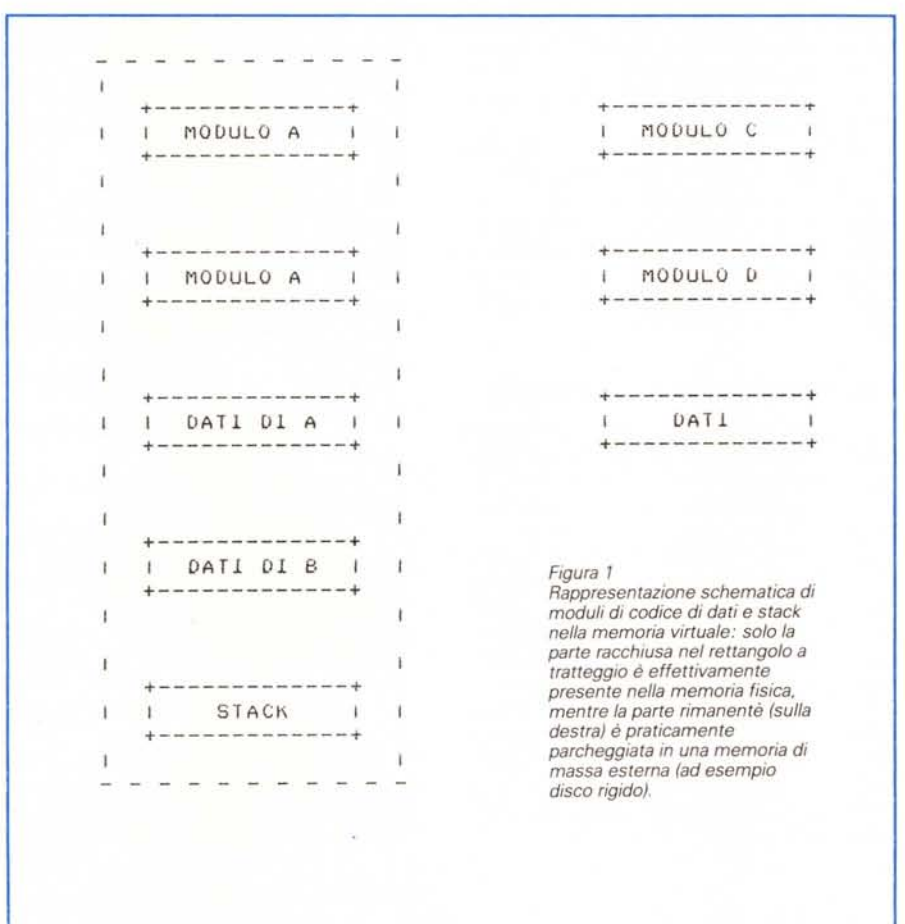


Figura 1
Rappresentazione schematica di moduli di codice di dati e stack nella memoria virtuale: solo la parte racchiusa nel rettangolo a tratteggio è effettivamente presente nella memoria fisica, mentre la parte rimanente (sulla destra) è praticamente parcheggiata in una memoria di massa esterna (ad esempio disco rigido).

I registri interni

In un certo senso in questo paragrafo descriviamo solamente una parte dei registri effettivamente presenti all'interno del 286, in pratica quelli che l'«utente medio» vede ed ai quali può accedere: in particolare conosceremo altri registri molto importanti che però sono «visibili» solo in modo particolare, principalmente solo a livello di sistema operativo, e che contengono informazioni basilari per il corretto funzionamento del tutto.

Come già accadeva nell'8086, si possono dividere in tre gruppi:
— i registri generali
— i registri di segmento
— i registri di stato e di controllo.

I registri generali

Abbiamo in questo caso una completa identità di registri e di funzioni rispetto all'8086: accanto all'accumulatore AX, al registro base BX, al registro con-

tatore CX ed al registro dati DX, troviamo ancora il «Base Pointer» BP, lo «Stack Pointer» SP ed i due registri indice SI e DI.

Sono ovviamente tutti a 16 bit ed i primi quattro citati possono essere ancora suddivisi in coppie di registri a 8 bit, dando così luogo ai ben noti AH, AL, BH, BL, CH, CL, DH e DL.

Il loro uso, come detto, è esattamente lo stesso finora visto nell'8086, dal momento che le istruzioni aggiuntive del 286 non alterano la filosofia di utilizzo dei registri in questione.

I registri di segmento

Anche in questo caso, almeno esteriormente, non si notano grossi cambiamenti nei registri segmento, che ancora una volta hanno un loro proprio «campo di applicazione»: il CS per i segmenti di codici, il DS per i segmenti di dati, l'SS per il segmento contenente lo stack e l'ES per un segmento ulteriore di dati. Ma, e qui nasce la differenza, sappiamo che di segmenti dati ne esisteranno tanti, così come per i segmenti di codice, alcuni posti in memoria fisica altri sparsi qua e là nelle periferiche: un nostro programma potrà accedere in genere solo ad un certo numero di tali segmenti, ogni volta indicandoli per mezzo dell'apposito registro di segmento.

Ora però all'interno del registro di codice o di dati non c'è più (come accadeva nell'8086) un valore rappresentante il paragrafo di memoria dal quale partiva il segmento desiderato, ma dal momento che i segmenti possono o meno essere effettivamente presenti in memoria, ecco che in realtà il registro di segmento sarà semplicemente un puntatore all'interno della «tabella dei segmenti» (fig. 2) a cui abbiamo accennato in precedenza: per tale motivo tutti i registri di segmento ora si chiamano «selector» e cioè ad esempio il CS è il «Code Segment Selector».

Ecco che dunque sarà cura del sistema operativo gestire questa tabella (che si chiama «Segment Descriptor Table») per fa sì che un programma possa accedere ad un certo segmento (sempre che ciò sia lecito, come vedremo parlando delle protezioni).

In pratica cambia la filosofia che sta alla base di istruzioni coinvolgenti i registri di segmento: mentre finora con l'8086 le istruzioni

```
MOV AX,DATA
MOV DS,AX
```

caricano in AX il valore del paragrafo a cui inizia il segmento denominato DATA e poi lo depositano in DS, con l'80286 in AX prima e in DS poi verrà caricato il valore di un selettore e ad esempio il valore 34 se, nella tabella di descrizione dei segmenti, il segmento DATA occupa la trentacinquesima posizione. In questo caso dunque non si sa nemmeno a quale indirizzo fisico della memoria è posto il «descrittore» del segmento DATA, ma si sa solo che è il trentacinquesimo della tabella, di cui non è possibile sapere (a livello utente) l'indirizzo iniziale, viceversa settato dal sistema operativo con un'apposita istruzione.

Un altro esempio è dato dal metodo tipico di caricare il Data Segment con il valore del Code Segment per mezzo delle seguenti istruzioni

```
PUSH CS
POP DS
```

In questo caso, analogamente a quanto visto prima, nello stack non viene salvato (e poi ripreso) il valore di un paragrafo di memoria, ma bensì il valore del «selettore» relativo al CS attuale: ancora una volta il tutto in previsione di un sistema di protezione veramente a prova di scassinatura.

Ovviamente in tutti questi casi analizzati i codici operativi delle istruzioni rimangono esattamente gli stessi, ma sarà il 286 a funzionare diversamente dall'8086, cosa che accade praticamente per tutte le istruzioni, anche per le più innocue, (apparentemente) quale le MOV.

I registri di stato e di controllo

Questi registri servono a mantenere registrato istante per istante lo stato del microprocessore e viceversa a controllarne le funzionalità: sono l'Instruction Pointer IP ed il registro dei flag.

Per quanto riguarda l'IP non c'è nulla da aggiungere a quanto già si sapeva riguardo l'8086: in particolare l'IP contiene l'offset della successiva istruzione, quella da eseguire, offset ovviamente riferito al Code Segment, quello «vero» e cioè quello descritto nella «Segment Descriptor Table» e non il registro CS; in questo caso non ha più senso parlare di coppia di registri CS:IP che identifica l'indirizzo della prossima istruzione da eseguire.

Per ciò che riguarda i flag c'è da dire che ai soliti e ben noti 9 flag sono da affiancare altre due quantità, una a due bit che rappresenta l'«Input Output Privilege Level» («IOPL») ed una ad un bit solo, il «Nested Task flag» («NT»).

In particolare la parola a 16 bit rappresentante i flag stavolta è costituita nel modo di figura 3: dove

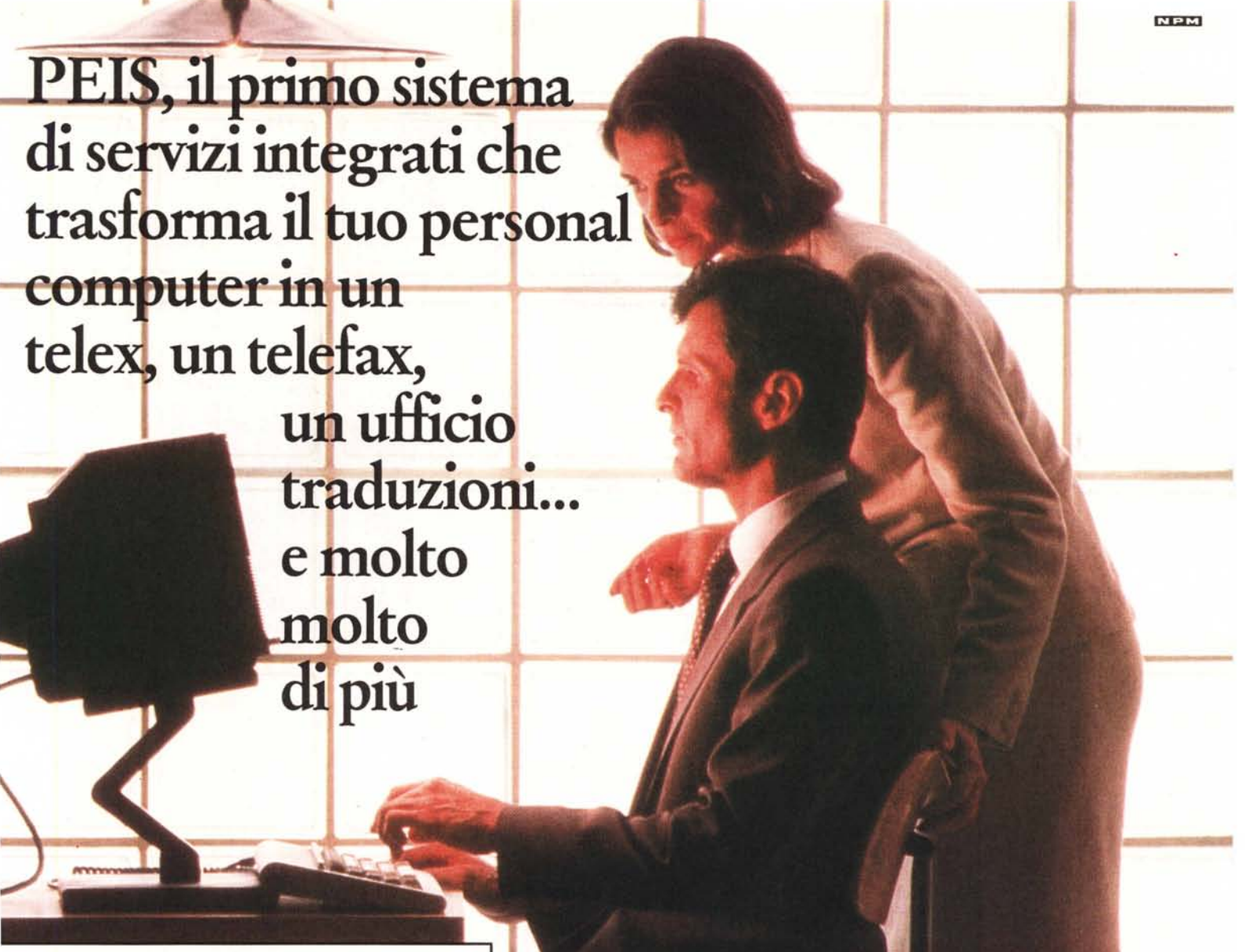
Figura 3

— «x» significa che il bit non è utilizzato
— NT è il nuovo «Nested Task Flag»
— IOPL è l'«I/O Privilege Level»: insieme al precedente, questi tre bit in totale non hanno molto significato per l'utente, mentre viceversa sono settati dal sistema operativo che ne fa largo uso. Meccanismi opportuni di protezione fanno sì che qualunque tentativo da parte dell'utente di cambiare lo stato di tali bit venga frustrato: non ha molto senso cambiare qualcosa di cui non si ha un'immediata conoscenza, soprattutto perché, pur conoscendo il significato intrinseco di tali bit (cosa che faremo tra qualche puntata), il loro contenuto varia dinamicamente e non c'è possibilità da parte dell'utente di utilizzare le informazioni in essi contenuti.
— OF è l'«Overflow Flag», che viene settato da quelle operazioni tra quantità dotate di segno in casi particolari
— DF è il «Direction Flag», che serve per auto-incrementare o decrementare i registri indice nelle operazioni sulle stringe
— IF è l'«Interrupt Flag», settato all'arrivo di un interrupt
— TF è il «Trace Flag», che permette di operare in single-step
— SF è il «Sign Flag», e cioè il bit più significativo di un operando
— ZF è lo «Zero Flag», settato se il risultato di un'operazione è nullo
— AF è il l'«Auxiliary Flag», settato da particolari operazioni
— PF è il «Parity Flag», settato a seconda della parità del risultato di un'operazione
— CF è il «Carry Flag», il ben noto flag di riporto.

Con questo terminiamo questa seconda puntata e rimandiamo alla prossima in cui parleremo dei modi di indirizzamento e dei tipi di dati. MC

NPM

PEIS, il primo sistema di servizi integrati che trasforma il tuo personal computer in un telex, un telefax, un ufficio traduzioni... e molto molto di più



La PEIS è un pacchetto di Servizi Informativi che utilizza il personal computer (di qualsiasi marca) e le normali linee telefoniche. Con la PEIS è possibile mandare e ricevere in tempo reale messaggi scritti riservati, utilizzare servizi telex e telefax, ottenere traduzioni, informazioni commerciali e altri servizi. Ogni utente dei Servizi Informativi PEIS riceve un Indirizzo ed una Password, con i quali può collegarsi al Servizio PEIS. Oltre ai Servizi predisposti, per i quali verranno addebitati mensilmente solo i reali utilizzi, si può direttamente usufruire, compreso nel prezzo di abbonamento annuo, del Servizio di Posta Elettronica.

La Posta Elettronica è un sistema che permette di rimanere in contatto con il proprio ufficio e con i propri clienti da qualunque città in Italia e all'estero con il solo costo di una telefonata urbana.

La PEIS è il più moderno ed integrato strumento di lavoro sul mercato, pensato e sviluppato per aumentare la produttività e l'efficienza. Il numero dei professionisti e delle aziende già utenti lo dimostrano.

Telefona o spedisci subito il coupon, allegato.

La PEIS ti dà il benvenuto nel futuro.

SEAT **A tutti gli abbonati alla Peis verrà offerto in omaggio un abbonamento per un anno alle Pagine Gialle Elettroniche**

Compila e spedisci in busta a: *PEIS* Via Carbonara 1, 40126 Bologna

Nome ..... Cognome .....

Via ..... N. ..... Tel. .....

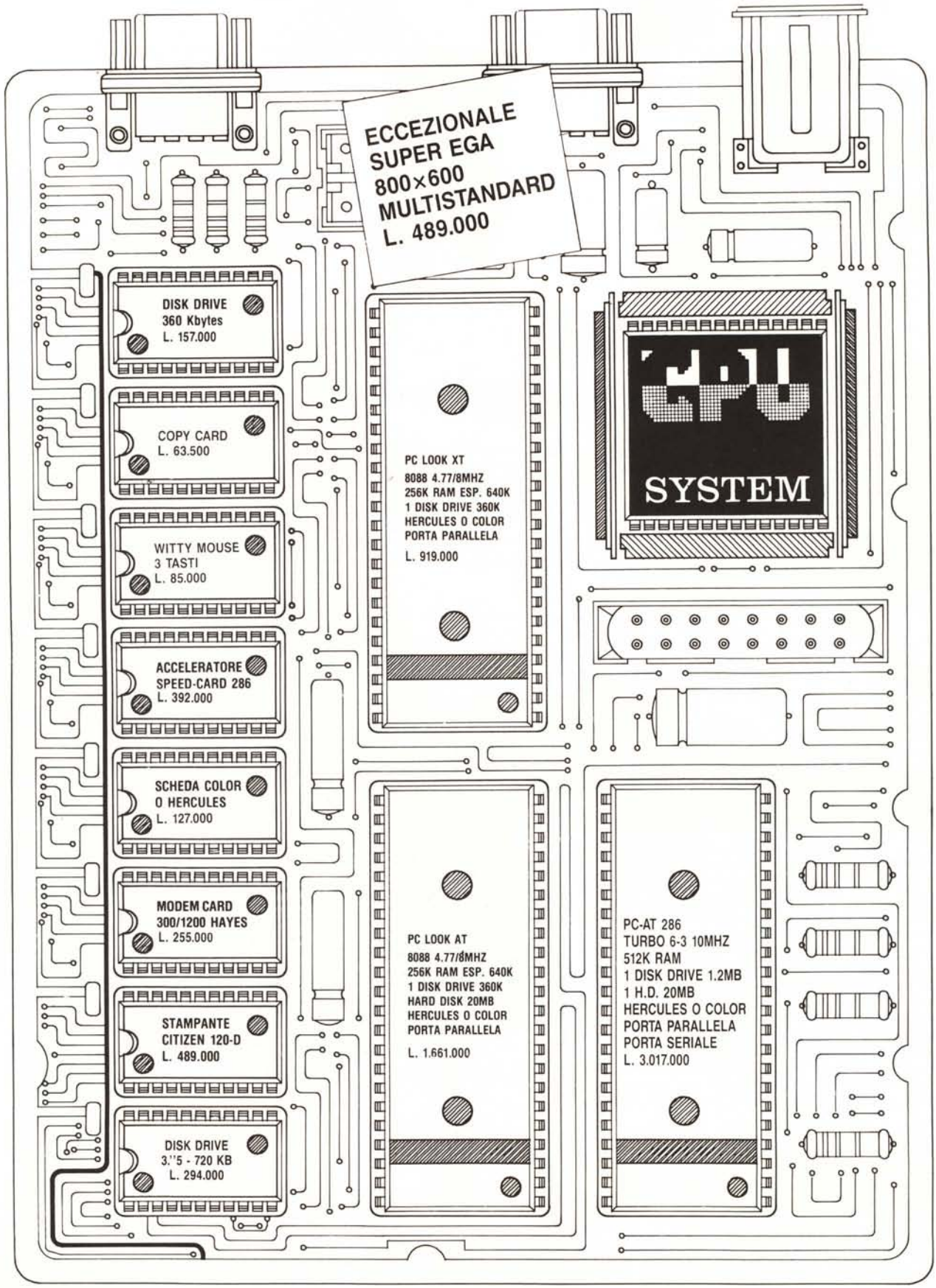
CAP ..... Città ..... Prov. .....

☐ Vorrei sottoscrivere un contratto annuale alla Peis al prezzo di L. 95.000 + Iva, accludo assegno o contante per L. 112.100

☐ Vorrei ricevere senza impegno da parte mia maggiori informazioni sul servizio Peis.

PEIS

Posta Elettronica International Service
Bologna - Tel. (051) 26.78.39 - Tlx 520626 TELEBO I
Milano - Tel. (02) 67.01.956 - Tlx 520560 INTSV I
La Peis utilizza Telefax della Face Standard



CPU - 50127 FIRENZE - Via M. Ulivelli 39/r - Tel. 055-4361096 - TELEX 574354 SEAC I - FAX 055/4361096

CPU - 50047 PRATO (FI) - Via Settesoldi 32 - Tel. 0574/434554