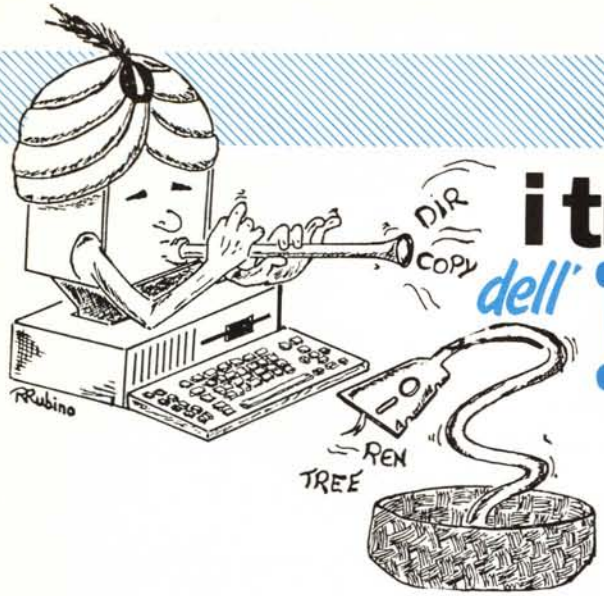

i trucchi dell' MS-DOS

a cura di Pierluigi Panunzi

Quinta parte

I comandi esterni

Riprendiamo il discorso lasciato in sospeso la scorsa puntata e torniamo perciò a parlare delle caratteristiche di un file di tipo ".exe".

Il discorso era stato interrotto quasi alla fine dell'analisi di quella parte di un programma di tipo ".exe", detta in gergo "program header": in breve, rimandando per i dettagli alla puntata precedente, si tratta di una zona di memoria posta in testa ad un programma, nella quale sono contenute informazioni di vitale importanza per il file stesso e che consentono poi al sistema operativo di allocare il programma (all'atto dell'esecuzione) in una qualunque zona di memoria.

Per comodità del lettore riportiamo la tabella A già pubblicata nella scorsa puntata, dove abbiamo rappresentato schematicamente il significato delle word costituenti l'"header" di un file di tipo ".exe".

Prima di terminare l'analisi delle word, segnaliamo un piccolo errore apparso nella scorsa puntata: in particolare nel corso dell'analisi delle word, ad un certo punto troviamo il paragrafo relativo alle "word n. 6" e dopo un po' il paragrafo relativo alle "word n. 6, 7".

In realtà, come è facile vedere dalla tabella stessa, il paragrafo "word n. 6" deve in realtà intendersi riferito alla "word n. 5", così come i richiami all'interno del paragrafo stesso.

Detto questo sono rimaste da analizzare due word relative ai cosiddetti "relocation table item".

Vediamo dapprima di cosa si tratta e poi andremo a vedere cosa significa il contenuto delle word.

Quando noi scriviamo un programma sia in Assembler che in linguaggio ad alto livello, con lo scopo di ottenere un file che giri correttamente sotto

HEADER DI UN FILE DI TIPO ".HEX"

word	significato	unita'
1	valore 5A4DH (esadecimale)	-
2	lunghezza file modulo 512	bytes
3	(lunghezza file + header) / 512	pages
4	numero "relocation table items"	-
5	ampiezza dell'header	paragr.
6	minima area alla fine	paragr.
7	massima area alla fine	paragr.
8	variazione per SS (Stack Segment)	paragr.
9	offset di SP (Stack Pointer)	byte
10	checksum del file	-
11	offset di IP (Instruction Pointer)	byte
12	variazione per CS (Code Segment)	paragr.
13	puntatore primo "item"	byte
14	numero di overlay	-
...		
i	offset dell'item	byte
i + 1	segment dell'item	paragr.
...		

Tabella A

MS-DOS, dobbiamo far sì che il programma ottenuto sia in tutto e per tutto "rilocabile" e cioè possa essere eseguito qualsiasi sia il punto in memoria in cui esso venga caricato e per essere vero questo fatto bisogna che il programma sia completamente "rilocabile".

Mentre questo fatto avviene automaticamente con i linguaggi (compilatori) ad alto livello, nel caso di un programma in linguaggio Assembler si potrebbero creare delle situazioni in cui il programma diventerebbe "position dependent" e cioè potrebbe essere eseguito solo in una ben determinata zona della memoria.

Senza scendere nei particolari, diciamo che ciò può succedere quando l'incauto o esperto programmatore stabilisca un valore per uno o più segmenti di cui è composto il programma, invece di lasciare il compito di assegnare ad ogni segmento una sua posizione all'interno della memoria al sistema operativo MS-DOS.

Nel caso in cui in un programma compaiano più segmenti tra i quali avvengono salti, sappiamo che all'atto della codifica dell'istruzione stessa comparirà l'indirizzo completo del punto in cui si vuole saltare, inteso come "offset" e come "segment".

A questo punto si innesca un meccanismo alquanto complesso legato alla "rilocabilità": analizziamo con un piccolo esempio come fa l'Assembler (il solito MASM) a codificare un salto inter-segment. Vedremo infine, grazie al "debug", in quale modo viene codificato il file ".exe" ottenuto.

Supponiamo dunque di considerare il seguente programmino:

```
CODE1   SEGMENT
        ASSUME CS:CODE1
START   LABEL NEAR
        NOP
        XOR AX,AX
        JMP FAR PTR LAB
CODE1   ENDS

CODE2   SEGMENT
        ASSUME CS:CODE2
        NOP
LAB     LABEL FAR
        MOV AX,1234H
        ADD AX,8765H
CODE2   ENDS
        END START
```

```
0000                            CODE1   SEGMENT
                                        ASSUME CS:CODE1
0000                            START   LABEL NEAR
0000  90                                NOP
0001  33 C0                             XOR AX,AX
0003  EA 0001 ---- R                    JMP FAR PTR LAB
0008                            CODE1   ENDS

0000                            CODE2   SEGMENT
                                        ASSUME CS:CODE2
0000  90                                NOP
0001                            LAB     LABEL FAR
0001  B8 1234                           MOV AX,1234H
0004  05 8765                           ADD AX,8765H
0007                            CODE2   ENDS
                                        END START
```

Figura 1

Assemblandolo otteniamo il "listato" in figura 1 dove vediamo appunto la creazione di due segmenti di codice ed un salto tra un segmento e l'altro: nella codifica del salto suddetto compare il codice operativo (OEAH), l'offset dell'etichetta LAB (offset all'interno del "suo" segmento di appartenenza) pari a "0001H", ma per quanto riguarda il segmento compare la scritta "---- R" indicante che a questo punto l'assemblatore non sa assolutamente quale sia il suo valore, demandando il compito non già al "linker", ma bensì all'MS-DOS stesso quando caricheremo il programma in memoria per eseguirlo.

Percorriamo dunque questa strada, linkando il programma ed ottenendo così un file di tipo ".exe" che analizzeremo riportandone il "dump della memoria" in figura 2.

Riconosciamo dunque, a partire dal byte "0000" proprio l'"header" del quale stiamo parlando dalla scorsa puntata: infatti i primi due byte valgono 4DH e 5AH!

Il valore successivo, che si legge 0017H dal momento che si tratta sempre di word, rappresenta la lunghezza in byte del nostro programma, pari perciò a 23 byte (proprio quelli tra l'indirizzo 0200H e 0216H).

Successivamente troviamo i seguenti valori:

0002H: l'ampiezza del file in multipli di 512 byte, compreso l'header stesso.

0001H: il numero di "relocation item", che analizzeremo subito.

0020H: l'ampiezza dell'header in paragrafi, tale che il programma vero e proprio inizia a 0200H.

0000H: minimo numero di paragrafi richiesti alla fine del programma caricato.

FFFFH: massimo numero di paragrafi richiesti alla fine del programma.

0000H: valore da aggiungere al registro SS.

0000H: valore assunto dal registro SP.

56B2H: valore della checksum, che il lettore diligente potrà verificare...

0000H: valore che assumerà il registro IP.

0000H: valore da aggiungere al registro CS

001EH: locazione all'interno dell'header del primo "relocation item", del quale parleremo tra breve.

0000H: numero di overlay.

0000H: parola di cui l'IBM non fornisce il significato.

0006H
0000H è questo il primo ed unico "relocation item" nella forma "offset-segment".

Il programma vero e proprio è come detto posto a partire dall'offset 0200H, laddove si riconoscono i byte delle istruzioni, così come apparivano nel listing fornito dal "MASM" (vedi figura 3).

```
0000:  4D 5A 17 00 02 00 01 00 20 00 00 00 FF FF 00 00
0010:  00 00 B2 56 00 00 00 00 1E 00 00 00 01 00 06 00
0020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030:  ...
       ...
01F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0200:  90 33 C0 EA 01 00 01 00 00 00 00 00 00 00 00 00
0210:  90 B8 34 12 05 65 87
```

Figura 2

```
0200:   90 33 C0 EA 01 00 01 00 00 00 00 00 00 00 00 00
0210:   90 B8 34 12 05 65 87
```

Figura 3

```
AX=0000  BX=0000  CX=0017  DX=0000
SP=0000  BP=0000  SI=0000  DI=0000
DS=66CC  ES=66CC  SS=66DC  CS=66DC  IP=0000

66DC:0000   90 33 C0 EA 01 00 DD 66-00 00 00 00 00 00 00 00
66DC:0010   90 B8 34 12 05 65 87
```

Figura 4

Da qui vediamo che i due segmenti sono posti all'inizio di due paragrafi ed i byte lasciati liberi dal codice sono riempiti con "00". Quello che ci interessa è vedere la codifica dell'istruzione "JMP FAR PTR LAB", che in questo caso è (associando i byte come word laddove serva, così come fa il MASM)

EA 0001 0001

dove il primo 0001 era l'offset di LAB, mentre al posto del vecchio "– – – – R" troviamo ora 0001.

La tabellina con cui abbiamo analizzato il "dump" ci diceva che all'offset 001EH dell'header si trovava l'inizio dell'elenco di "relocation item" (un unico item nel nostro caso) ed a tale offset troviamo la coppia di word data da 0006 a 0000: questo dice che all'offset 0006 nel segmento 0000 (a partire dall'inizio del programma vero e proprio) è posto in item da alterare. Guarda caso si tratta proprio di quello "0001" dell'istruzione di salto.

Questa word non è altro che un elemento dipendente dalla rilocabilità, da alterare in funzione del punto della memoria in cui il programma verrà caricato.

Vediamo dunque, alla luce di tutte queste conoscenze come agisce il sistema operativo all'atto del caricamento di un programma in memoria, con particolare attenzione per gli item ora visti:

— calcola l'indirizzo fisico della memoria a partire dal quale possa caricare il programma dato, prefissando in particolare un valore comune per i quattro registri di segmento, detto "start segment".
— viene letto l'header e stabilita l'ampiezza fisica e vera del file, grazie ai valori contenuti nelle word 3, 5 e 2.
— viene caricato in memoria il programma vero e proprio, così come è riportato nel file.
— per ogni relocation item effettua le seguenti operazioni: il valore del "segment" contenuto nell'item è sommato allo "start segment": ciò in unione all'"offset" contenuto nell'item, fornisce l'indirizzo di una word del programma che deve essere alterata per motivi di rilocabilità; individuata dunque la word da alterare, al valore in essa contenuta si somma proprio il valore dello "start segment" e tale valore ottenuto viene posto nella word stessa.
— finiti di elaborare i "relocation item", vengono dunque alterati i valori dei registri SS, SP ed IP nonché di CS secondo quanto contenuto nelle word dell'header relative a tali registri.
— il controllo viene così ceduto alla routine posta all'indirizzo dato da CS:IP. Era ora...

Nel nostro semplice esempio di tali relocation item ce n'è appena uno, ma in genere passando a programmi più complicati ce ne potranno essere moltissimi: tanto più un programma è lungo e complesso tanto più lungo è già il tempo di caricamento in memoria del file stesso, tempo al quale dobbiamo aggiungere il tempo di calcolo dei "relocation item" (sempre pochissimo, ma non certo nullo).

Supponendo di essere noi il sistema operativo, decidiamo di voler porre il programma in esame all'interno del segmento 1111H: allora i vari registri di segmento verranno posti tutti ad 1111H (DS ed ES per default, SS e CS perché il termine aggiuntivo all'interno dell'header è nullo in entrambi i casi).

Scorrendo dunque i "relocation item" ne troviamo uno posto all'offset 0006 del segmento 0000H a partire dal segmento 1111H: ciò vuol dire che nella cella di memoria del computer, posta all'indirizzo 1111H:0006 c'è una word che dobbiamo alterare.

Il suo contenuto era 0001H che sommato al valore dello "start segment" (pari a 1111H) dà un valore definitivo pari ad 1112H.

Abbiamo verificato il tutto per mezzo del debugger, con il quale abbiamo ottenuto un insieme di valori dei registri ed un'allocazione effettiva e finale della memoria data da quanto in figura 4, dove vediamo che lo "start segment" vale 66DCH (tanto per CS quanto per SS), mentre per quel che riguarda i valori di DS ed SS rimandiamo il discorso alla prossima puntata.

Eccolo lì! All'indirizzo 66CCH:00100H troviamo finalmente il valore rilocato del segmento all'interno dell'istruzione di salto, pari 66DDH e cioè uno in più (secondo quanto diceva il "relocation item" rispetto allo "start segment").

Ecco dunque il fatidico salto avverrà all'indirizzo 66DDH:0001H, dove troveremo proprio le istruzioni che avevamo deciso noi.

Per inciso tale indirizzo può essere scritto come 66DCH:0011H e ciò spiega il perché si trovi proprio nella linea successiva del dump.

Torniamo ora dunque all'analisi delle word costituenti l'header di un file di tipo ".exe", per concludere l'argomento.

Word 4,13 - la word n . 4 indica dunque il numero di "item" che dovranno essere modificati prima dell'esecuzione del programma, mentre la word n. 13 indicherà a quale offset del file stesso inizia la tabella contenente gli indirizzi di tutti gli item da correggere. Dal momento che i programmi possono superare la barriera di 64K, ecco che per individuare una word all'interno di un programma, il nostro item, abbiamo bisogno del suo indirizzo completo formato da segment ed offset, relativi però all'inizio fisico (lo "start segment") della parte del file relativa al programma vero e proprio. Le coppie di word puntate a partire dal contenuto della word n. 13 dunque saranno gli indirizzi completi degli item da correggere, indirizzi formati da una word di offset e da una word di segmento.

Terminata dunque l'analisi delle word facenti parte dell'"header" di un file di tipo ".HEX", diamo l'appuntamento alla prossima puntata dove parleremo diffusamente (come promesso), del programma "exe2bin" e, se ne avremo lo spazio, analizzeremo un'altra struttura logica caratteristica dell'MS-DOS, il cosiddetto "Program Segment Prefix".

MC

GVH linea computer

GVH - Via Della Selva Pescarola, 12/2 - 40131 Bologna - Tel. 051/6346181 r.a. -
Telex 511375 GVH I - Fax 051/6346601 ATTENZIONE NUOVO INDIRIZZO

MODELLO **CARD XT**

Personal computer XT compatibile versione compatta (occupa metà spazio rispetto alle dimensioni del P14T). Il Card XT ha 7 Slots, 5 per schede lunghe e 2 per schede corte. Corredato di tastiera KB84. Un drive da 360K già installato.

L. 740.000

MODELLO **CARD AT**

Personal computer AT compatibile. Clock 6-8 MHz. Microprocessore 80286/8, memoria a 512K già installata. Un drive da 1,2M.
Hard disk da 20 MB con controller, made in Japan corredato di tastiera KB100 (100 tasti).

L. 2.450.000

MODELLO **P14 T**

Personal computer montato, collaudo garantito 12 mesi, contenitore metallico Look AT.
Scheda madre turbo con il clock 8 MHz 256 K di memoria RAM installati espand. 640 K. Un Drive da 360 K installato. Adattatore Floppy Disk Drive con cavo. Tastiera KB84 (84 tasti). **L. 745.000**

Le schede video sono fornite su richiesta del cliente

TASTIERE

TASTIERA KB84: Pratica, ergonomica, con 10 tasti funzione. Con 84 tasti compatibile XT/AT. Approvata norme Fcc L. 106.000

TASTIERA KB100: Pratica, ergonomica, con 12 tasti funzione. Con 100 tasti, compatibile XT/AT, 3 Led. Approvata norme FCC L. 150.000

MONITOR

CDM 1200: Video Monocromatico 12", fosfori verdi(gn) o arancio(or). Ingresso video composito. Definizione orizzontale e verticale: 1000 linee.L. 189.000

M14H1: Video Monocromatico 14" peper white Può essere collegato sia alla scheda Hercules sia alla scheda colore, doppia frequenza. L. 280.000

PHILIPS BM 7513: Monitor a fosfori verdi 12", Tubo a 90 gradi. Risol. orizz.: 920 Piz. risol. Vert.: 350 Pix. L. 185.000

HR 31350: Monitor a colori da 14" per scheda Ega ad alta risoluzione (640×350). Doppia frequenza di scansione: 15,75 KHz e 21,85 KHz/0,31 Dot pitch. L. 740.000

STAMPANTI

STAMPANTE LSP 120 D: Stampante Citizen 80 colonne o 132 colonne compatte. Velocità 120 Cps. Nlq 24 Cps, protocollo IBM/EPSON. L. 499.000

STAMPANTE PST 001: Mitsubishi. Stampante standard con matrice ad aghi ed interfaccia parallela.
Può stampare 136 colonne a 180 cps (pica). Compatibile con grafica IBM 80 cps. Stampa bidirezionale. L. 950.000

SCHEDE DI ESPANSIONE

G7 A - Multi display card. Eccezionali caratteristiche in una scheda corta; Novità — Uscita colore grafica Rgb o Hercules a scelta — Uscita monocromatica o colore videocomposta — Uscita parallela per stampante — Uscita per joystick — Uscita mouse — Uscita penna ottica L. 160.000

CX 20 - Scheda monocromatica ad alta risoluzione tipo Hercules 100% compatibile. Completa di uscita parallela. L. 103.000

CX 25 - Scheda grafica video colori con connettore standard Rgb con uscita videocomposita, risoluzione 640×200 (bianco/nero). Con uscita parallela per stampante. L. 103.000

CX 27 - Xega card. Nuovissima scheda compatibile Ega e Hercules contemporaneamente. Include 256 Kb Ram e una Ram come generatore di caratteri con un monitor Ega. Questa scheda fornisce una scelta di 16 colori o 64 colori in modo grafico con una risoluzione di 640×350 Pixel.
In modo monocromatico ha una grafica di 640×350 oppure 720×348 Pixel. Lavora con tutti i software disponibili: Lotus, Window, auto cad, gem, ecc. Tecnologia LSI. per XT/AT. L. 450.000

NUOVA - CX 29 - De Lux Dega card. caratteristiche superiori; compatibile con: Ega (Enhanced graphics adapter); Cga (color graphics adapter); Mda (monochrome display adapter; Hga (Hercules graphics adapter). Inoltre fornisce i seguenti modi: doppia scansione in modo Cga (640×400), risoluzione Pga 640×480 E 132 colonne x 44. La doppia scansione in modo Cga è trasparente per il softwar Cga ed aumenta la qualità nei monitor Multisync. La risoluzione Pga (640×480) fornisce un'immagine molto migliore dell'Ega nei monitor Multisync. La scheda Dega è perfetta per Word Processing, Cad Cam, ecc. per XT/AT L. 590.000

CX 30 - Scheda Multi I/O, con 2 porte seriali (una montata) una porta parallela, orologio calendario; connettore per joystick. Per XT. L. 93.000

CX 150 - Schede Multiseriale con 4 uscite RS 232. Per AT. Utilizzata con sistema operativo DOS e XENIX. L. 260.000

CX 50 - Scheda seriale RS 232. Per XT/AT L. 52.000

CX 52 - Scheda controller per 2 drive con cavo per 2 drive, per XT.L. 50.000

CX 54 - Scheda controller per floppy disk. Completo di cavi per 2 drive da 3"1/2 e 720K e 5" 1/4 da 360K o 1,2M. Collegabile a 4 drive differenti e contemporaneamente. L. 110.000

CTXT - Scheda controller per Hard Disk. Supporta 2 HD da 20 MB con interfaccia ST 506. Completo di cavi. Per XT L. 180.000

CX 70 - Scheda 576Kb Ram (senza Ram). L. 48.000

CX 71 - Scheda 640Kb Ram (senza Ram). L. 52.000

CX 286 - Speed card velocizzatore Spd per XT con clock a 4,77M. Migliora le prestazioni dell'XT portandole a quelle di un At. La memoria cache permette un'aumento della velocità di 6,6 volte. Utilizza un microprocessore 80286. Attenzione: utilizzabile solo con clock a 4,77 MHz. L. 490.000

CX 43 - Scheda AD-DA - 12 Bit - 500 μS/V L. 330.000

CX 40 - Scheda Eprom/Prom writer con 4 porte. Programma Le Eprom. Per XT/AT. L. 320.000

CX 32 - Scheda Multi I/O. Fornisce 2 uscite. Serieali RS 232 (una montata + una opzionale), una uscita parallela, una uscita game (Joystick). Per AT. L. 110.000

CX 38 - Multifunzione con espansione di memoria da 1,5 Mb. Fornisce 2 uscite seriali con una montata, una uscita parallela per stampante; una uscita game (joystik). 6 banchi di memoria per 1,5 Mb. Per AT. L. 310.000

CTRL - Scheda controller universale per Hard Disk e Floppy Disk. Controller completo per 4 Drive e 2 HD. Floppy Drive da 3,5/720 K e 5"1/4 da 360K e 1,2M. Interfaccia per HD tipo St 502/St 412, corredata di cavi. Per AT. L. 350.000

PARTI STACCATE

LH 4 - Floppy Disk Drive a trazione diretta da 360K versione Slim. Marca Teac. garantiti. L. 218.000

LH 6 - Floppy Disk Drive a trazione diretta da 360K versione Slim. Marca Acc. garantiti. L. 199.000

LH 3 - Floppy Disk Drive a trazione diretta da 1,2M versione Slim. Marca Acc. garantiti. L. 240.000

MB 4 - Main Board Turbo 640 Kb (scheda madre), con 8 slot e 256 Kb di memoria Ram già installati. Clock 8 MHz. L. 310.000

PX - Alimentatore da 150 Watt. Interruttore laterale. Alta affidabilità. Per XT L. 110.000

ACCESSORI

CP 25 - Cavo per stampante parallela; lunghezza 1,8 mt. L. 14.000

CS 25 - Cavo per stampante seriale; lunghezza 1,8 mt. L. 14.000

CS 35 - Cavo per collegamento per il box DSB 2, 1,8 mt. L. 14.000

CR 25 - Cavo per collegamento per monitor RGB. L. 14.000

SK 14 - Dischetti 5" 1/4, DF DD Bulk (minimo 100 pezzi) L. 1.000

HD 5126 - Hard Disk da 20 MB senza controller meccanica JAP Garantiti 1 anno L. 720.000

SM 1200 - Modem interno su scheda corta. Approvato norme FCC. Hayes compatibile. Self test. Corredato di manuali e software per XT e AT.L. 230.000

GM 3 - Genius Mouse, Endecorder ottici, per PC XT/AT compatibili; 3 pulsanti per il disegno. L. 97.000

GM 6 - Genius mouse endecoder ottici, per PC XT/AT compatibili; 3 pulsanti per il disegno, massima traccia disegnabile 500 mm/sec. Risoluzione 0,12 mm/dot, 200 DP. Connettore D-25P Standard. Applicazioni software: Dbase 3, Multiplan, Wordstar, Autocad, ed altri programmi compatibili. Uscita RS 232. L. 180.000

DSB 2 - Box di commutazione per uscita parallela. Permette l'uso di due stampanti con un computer, oppure due computer con una stampante. L. 75.000

CH 2 - Chassis Completo di alimentazione e accessori meccanici. Permette il montaggio di Hard Disk-Floppy, streamer tape o altri accessori esterni al computer. Fornisce un'alimentazione da 5V-2A e 12V 3/5 A. Completo di cavo piatto e accessori L. 210.000

GRUPPO DI CONTINUITÀ UPS 250. Potenza di spunto 259 W. Tempo di lavoro con batteria interna 15 minuti L. 440.000

P.S.: I marchi IBM XT AT sono marchi registrati dalla International Business Machines.

PREZZI IVA ESCLUSA

DISTRIBUTORI GVH PER LINEA COMPUTER

BOLOGNA - BOTTEGA ELETTRONICA - Tel. 051/550761
CERIGNOLA - DISCOTECA OMNIA
MILANO - CRC ITALIA - Tel. 02/6071515
MODENA - ELECTRONIC CENTER - Tel. 059/210512
TREVISO - ELB TELECOM - Tel. 0422/66600
SALERNO - GENERAL COMPUTER - Tel. 089/237835
ROMA - S.T.E. srl - Tel. 06/5425465

CENTRI ASSISTENZA TECNICA

BOLOGNA - GVH - Tel. 051/6346352
MILANO - CRC ITALIA - Tel. 02/6071515
NAPOLI - I.T.S. sas - Tel. 081/7513800