di Sergio Polini

Goto globali, Errori critici

Il tema dell'allocazione dinamica della memoria, introdotto il mese scorso, ci accompagnerà ancora per qualche tempo; si tratta infatti di un aspetto tipico del Pascal ma del tutto estraneo a chi abbia esperienza solo di BASIC, FORTRAN o COBOL. Ora tuttavia cambiamo argomento. Succede questo: si parla di Turbo Pascal non solo sulla rivista, ma anche su MC-Link. Qui c'è naturalmente maggiore immediatezza, il colloquio tra gli utenti è più diretto, capita frequentemente che qualcuno proponga un suo problema e qualche altro poco dopo gli suggerisca una soluzione. Mi è capitato recentemente di rispondere a due utenti che avevano chiesto aiuto per problemi di sicuro interesse, tanto che ho ritenuto utile sottoporre le relative soluzioni anche a quei lettori che non avessero accesso a MC-Link

Primo problema: come si può tornare al main body (il corpo principale) di un programma da qualunque procedura, a qualunque livello sia nidificata? Il Turbo Pascal infatti consente solo goto locali al blocco in cui è definita la corrispondente label, non anche i goto globali previsti dal Pascal standard.

Quella del goto è una vecchia storia: si va da chi non ne può proprio fare a meno a chi non ne vuole nemmeno sentire parlare. È ormai assodato, nonché dimostrato, che è possibile scrivere qualsiasi tipo di programma senza usare i goto, ma è anche vero che talvolta questi, sebbene «brutti» informaticamente, sono semplicemente utili e comodi. Un caso è proprio quello dell'uscita da una funzione o procedura per saltare direttamente al «main» (è questo l'uso che Wirth fa del goto nei programmi contenuti nel suo *Algoritmi + Strutture di dati = Programmi*), ma anche in generale da una procedura ad un'altra, ad esempio ad una che gestisca situazioni di errore. I goto globali sono quindi anche più comodi di quelli locali.

Secondo problema: come controllare da programma quelli che il DOS chiama «errori critici», quali l'output a un disco danneggiato o il cui drive è rimasto aperto, o a una stampante che non esiste o è spenta o è senza carta? Non si tratta di un problema banale: la direttiva

Listato 1

```
{ SETJMP.PAS }

program SetjmpDemo;
{$I SETJMP.INC}
procedure Salta;
begin
  writeln('Ora ci provo...');
  LongJmp(JumpBuffer,1);
  writeln('Qui non ci dovevo arrrivare');
  halt
end;
begin
  if SetJmp(JumpBuffer) <> 0 then begin
    writeln('Ha funzionato!');
    exit;
  end;
  Salta
end.
```

«I» non consente di gestire questo tipo di errori di I/O, che possono tuttavia causare seri inconvenienti, quali la brusca interruzione di un programma senza la possibilità di chiudere i file aperti. Anche il glorioso DBIII andava in tilt in situazioni del genere, e si è dovuto aspettare il DBIII Plus per un maggiore controllo delle situazioni «critiche». Urge una soluzione.

Goto globali

Dicevamo che la Borland ha voluto scostarsi dallo standard in favore di goto solo «locali», come in C. Il C tuttavia è accompagnato da una nutrita libreria di funzioni standard, tra cui vi è una coppia setjmp e longjmp che consente di «saltare» dove si vuole. Proveremo quindi a farcene una versione in Turbo Pascal.

Setjmp va chiamata proprio lì dove vogliamo poi saltare (non si usa una label): il programma passa per la chiamata della funzione e ne controlla il risultato; se questo è zero (e SetJmp ritorna sempre zero) si va oltre, altrimenti si eseguono le istruzioni cui si vuole saltare con LongJmp. Quest'ultima infatti agisce sullo stack in modo tale da simulare un ritorno da SetJmp con un valore diverso da zero.

Detto così non è certo molto chiaro; conviene guardare ad un esempio (listato n. 1). Il programma SetjmpDemo chiama subito SetJmp passando come parametro una variabile JumpBuffer (sulla quale torneremo tra un attimo); la funzione ritorna zero e quindi la condizione dell'if non viene verificata; si trascura pertanto quello che segue il «then begin» per andare oltre. Viene poi chiamata la procedura Salta, che chiama LongJmp con due parametri: la stessa variabile JumpBuffer e un intero *diverso da zero*. LongJmp ricopia da JumpBuffer l'indirizzo dell'istruzione cui era prima tornata SetJmp (quella cioè che verificava se il risultato era diverso da zero) e provoca un nuovo ritorno a quella stessa istruzione, dopo aver però fatto in modo che il risultato sia diverso da zero. Si torna così al main per eseguire di nuovo il test, che questa volta ha esito positivo. Il programma scrive «Ha funzionato!» e termina. Tutto accade come se al posto di Longjmp vi fosse un goto ad una label posta subito dopo il «then begin».

Vediamo come funziona. Abbiamo visto nella puntata di marzo che nello stack di ogni procedura, nella locazione BP+2, è conservato l'indirizzo dell'istruzione cui la procedura ritorna dopo il RET; abbiamo anche visto che in BP+0 è conservato il valore che lo stesso BP aveva nella procedura chiamante e che BP contiene il valore che viene poi riassegnato a SP subito prima del RET. Sono questi i tre valori (indirizzo di ritorno, BP «salvato» e BP «corrente» destinato a diventare SP) che consentono di definire, e quindi di riprodurre, l'ambiente in cui una procedura opera. In SETJMP.INC (listato n. 2) vengono dichiarati un tipo «jmpbuf» come array di tre interi destinati ad ospitare quei tre valori, una funzione SetJmp e una procedura Longjmp (ho riprodotto tra parentesi graffe, oltre alla istruzioni Assembler necessarie per preparare gli inline statement, anche quelle prodotte automaticamente dal compilatore). SetJmp mette in DS:BX l'indirizzo di una variabile appartenente al tipo jmpbuf e passata come parametro-variabile, deposita in questa i tre valori e ritorna zero. Notare che per ritornare un valore nullo viene messo uno zero in BP+8: viene infatti generata automaticamente dal compilatore una istruzione che assegna ad AX il valore contenuto in BP+8; come abbiamo visto a marzo, dopo che una funzione ritorna alla routine che l'aveva chiamata, si suppone che il risultato sia appunto in AX. LongJmp riceve come

Listato 2

```
{  SETJMP.INC  }

type
  jmpbuf = array[1..3] of integer; { ret, bp "corrente", bp "salvato" }

var
  JumpBuffer: jmpbuf;

function SetJmp(var jbuf: jmpbuf): integer;
begin
                    { PUSH BP          ; istruzioni generate }
                    { MOV  BP,SP       ;  dal compilatore    }
                    { PUSH BP          }
  inline(
    $C5/$5E/$04/    { LDS  BX,[BP+4]   ; indirizzo di jbuf   }
    $8B/$46/$02/    { MOV  AX,[BP+2]   ; return address     }
    $89/$07/        { MOV  [BX],AX                          }
    $89/$6F/$02/    { MOV  [BX+2],BP   ; bp "corrente"      }
    $8B/$46/$00/    { MOV  AX,[BP]     ; bp "salvato"       }
    $89/$47/$04/    { MOV  [BX+4],AX                        }
    $33/$C0/        { XOR  AX,AX       ; azzera ax          }
    $89/$46/$08);   { MOV  [BP+8],AX   ; risultato := 0     }
                    { MOV  AX,[BP+8]   ; istruzioni generate }
                    { MOV  SP,BP       ;  dal compilatore    }
                    { POP  BP          }
                    { RET  6           }
end;

procedure LongJmp(var jbuf: jmpbuf; result: integer);
begin
                    { PUSH BP          ; istruzioni generate }
                    { MOV  BP,SP       ;  dal compilatore    }
                    { PUSH BP          }
  inline(
    $8B/$46/$04/    { MOV  AX,[BP+4]   ; result              }
    $C5/$5E/$06/    { LDS  BX,[BP+6]   ; indirizzo di jbuf   }
    $8B/$6F/$02/    { MOV  BP,[BX+2]   ; bp "corrente"       }
    $8B/$0F/        { MOV  CX,[BX]     ; return address      }
    $89/$4E/$02/    { MOV  [BP+2],CX                         }
    $8B/$4F/$04/    { MOV  CX,[BX+4]   ; bp "salvato"        }
    $89/$4E/$00);   { MOV  [BP],CX                           }
                    { MOV  SP,BP       ; istruzioni generate }
                    { POP  BP          ;  dal compilatore    }
                    { RET  6           }
end;
```

parametro-variabile la stessa JumpBuffer e ne copia i tre valori: l'indirizzo cui doveva tornare SetJmp in BP+2, il BP «salvato» nella locazione BP+0, il BP «corrente» in BP; pone però il valore del parametro «result» in AX. Poi vengono eseguite le solite istruzioni di chiusura; in particolare il RET provocherà un ritorno all'istruzione successiva alla chiamata di SetJmp, ma, poiché AX è diverso da zero, sarà come se si fosse eseguita una chiamata a SetJmp con risultato diverso da zero.

Tutto qui.

Interrupt 24h

I lettori di MC sanno sicuramente cosa è un interrupt sotto MS-DOS. Tuttavia, come dicono alla RAI, ripetiamo qualche nozione a beneficio di chi si fosse collegato in questo momento. Gli interrupt vennero introdotti per consentire al processore di gestire eventi «esterni»: quando si preme un tasto sulla tastiera, quando scatta il clock interno, parte un interrupt. Questo fa sì che il processore interrompa quello che stava facendo per eseguire una routine il cui indirizzo (segmento e offset) viene conservato in una tabella posta nella parte più bassa della RAM. Questo meccanismo viene però anche usato come alternativa alle tradizionali chiamate di subroutine, in quanto ha rispetto a queste un notevole vantaggio: la routine chiamante non deve conoscere l'indirizzo di quella che vuole chiamare, dal momento che questo è conservato nella tabella che dicevamo. Un programma può quindi chiamare le funzioni del DOS, ad esempio, pur non sapendo dove queste sono memorizzate, e soprattutto senza essere condizionato dagli inevitabili cambiamenti negli indirizzi che intervengono tra una versione del DOS e la successiva. Non occorre sapere «dove stanno» le funzioni del DOS, basta sapere che le si chiama con un interrupt numero 21H. L'istruzione «INT 21H» salva i flag nello stack e poi salta alla routine il cui indirizzo è contenuto nel trentaquattresimo elemento della tabella (l'equivalente decimale di 21H è 33, ma il primo interrupt ha numero 0). Se la routine chiamata termina con un IRET, vengono ripristinati i flag e si torna alla istruzione subito successiva a «INT 21H».

Dato cher INT opera in maniera analoga a una CALL, è possibile che la routine chiamata da un INT ne chiami a sua volta un'altra nello stesso modo. In particolare alcune funzioni del DOS, quando riscontrano un «errore critico», chiamano l'interrupt 24H. Un esempio di «errore critico» è dato dal tentativo di scrivere su un dischetto danneggiato: in questi casi è ovviamente inutile insistere, non rimane che rinunciare e uscire dal programma; altro esempio è l'invio di dati a una stampante che non c'è o è spenta: si

Listato 3

```
{ INCNUM.PAS }

program IncNum;
{$C-} { necessario perche' funzioni KeyPressed }
const
  n: integer = 0;
var
  reg: record case integer of
    1: (ax,bx,cx,dx,bp,si,di,ds,es,flags: integer);
    2: (al,ah,bl,bh,cl,ch,dl,dh         : byte)
  end;
  SegInt5, OfsInt5: integer;
procedure Int5;
begin
                {          PUSH  BP     ; istruzioni generate }
                {          MOV   BP,SP  ;  dal compilatore    }
                {          PUSH  BP     }
  inline(
    $50/        { prologo: PUSH  AX     ; cfr. Manuale, pag. 214 }
    $53/        {          PUSH  BX     }
    $51/        {          PUSH  CX     }
    $52/        {          PUSH  DX     }
    $56/        {          PUSH  SI     }
    $57/        {          PUSH  DI     }
    $1E/        {          PUSH  DS     }
    $06/        {          PUSH  ES     }
    $FB);       {          STI          }
  n := n + 1;
  inline(
    $07/        { epilogo: POP   ES     ; cfr. Manuale, pag. 214 }
    $1F/        {          POP   DS     }
    $5F/        {          POP   DI     }
    $5E/        {          POP   SI     }
    $5A/        {          POP   DX     }
    $59/        {          POP   CX     }
    $5B/        {          POP   BX     }
    $58/        {          POP   AX     }
    $8B/$E5/    {          MOV   SP,BP  }
    $5D/        {          POP   BP     }
    $CF)        {          IRET         }
end;
begin
  reg.ax := $3505;       { Determina l'indirizzo originario    }
  MsDos(reg);            {  della routine associata all'INT 5  }
  SegInt5 := reg.es;     { Conserva in SegInt5 il segment      }
  OfsInt5 := reg.bx;     {  in OfsInt5 l'offset                }
  reg.ds := CSeg;        { Sostituisci la routine di INT 5     }
  reg.dx := ofs(Int5);   {  con la procedura Int5              }
  reg.ax := $2505;
  MsDos(reg);
  repeat
    writeln('Shift-PrtScr per incrementare n (valore di n :',n,')');
    writeln('Un altro tasto per finire');
    delay(500)
  until KeyPressed;
  reg.ds := SegInt5;     { Rimetti a posto INT 5 }
  reg.dx := OfsInt5;
  reg.ax := $2505;
  MsDos(reg)
end.
```

può accendere la stampante e ritentare, oppure non rimane che uscire dal programma. La routine chiamata da INT 24H non pretende di giudicare da sé cosa sia meglio fare; mostra quindi su video un messaggio che descrive il tipo di errore e chiede: «Annulla, Riprova, Ignora?» (le versioni più recenti del DOS offrono anche l'opzione «Tralascia»). «Annulla» vuol dire fine del programma e ritorno al DOS, «Ignora» e «Tralascia» provocano il completamento della funzione del DOS che era incappata nell'errore e il suo ritorno al programma, «Ignora» come se tutto fosse andato bene, «Tralascia» con un codice d'errore.

Se non ci piacesse questo tipo di comportamento, potremmo sostituire alla routine normalmente chiamata da INT 24H un'altra routine scritta da noi: basta sostituire l'indirizzo memorizzato nella tabella. Vi sono due funzioni del DOS che si occupano di queste cose: la 35H ritorna in ES:BX l'indirizzo della routine che viene eseguita quando viene attivato l'interrupt il cui numero è indicato in AL; la 25H fa sì che, quando viene attivato l'interrupt indicato in AL, venga eseguita la routine il cui indirizzo viene specificato mediante DS:DX.

Gestione delle interruzioni

Così il manuale italiano traduce l'inglese «Interrupt Handling», ovvero la preparazione di un «interrupt handler», di una routine che venga chiamata quando viene attivato un interrupt.

Il programma INCNUM, nel listato numero 3, ne propone forse l'esempio più semplice possibile. Quando premo contemporaneamente i tasti Shift e PrtSc viene generato un interrupt numero 5; a questo è associata una routine del BIOS che invia alla stampante una copia di quanto appare sul video. Nel nostro programma sostituiamo a questa routine un'altra che invece incrementa una costante tipizzata «n».

Il programma per prima cosa usa la funzione 35H del DOS per salvare nelle variabili SegInt5 e OfsInt5 l'indirizzo, segmento e offset, della routine associata all'interrupt 5; usa poi la funzione 25H per sostituire a questa una procedura Int5. Viene quindi eseguito un ciclo che mostra su video il valore corrente di «n»; premendo Shift-PrtSc «n» viene incrementata di 1, premendo un altro tasto il programma termina, dopo aver rimesso le cose a posto. Si usa infatti ancora la funzione 25H del DOS per associare nuovamente all'interrupt 5 la routine il cui indirizzo era stato prima salvato nelle variabili SegInt5 e OfsInt5. Unici interrupt che possono non essere «rimessi a posto» sono: quello che contiene l'indirizzo della routine chiamata quando un programma termina (22H), quello che parte quando si preme Ctrl-C (23H) e il nostro 24H, in quanto ci pensa il DOS stesso (i dati relativi sono infatti conservati nel Program Segment Prefix).

La procedura Int5 consiste in un prologo, nell'istruzione di incremento e un epilogo. Prologo ed epilogo servono a mantenere inalterato il contesto del programma: la routine attivata da un interrupt può infatti partire in qualsiasi momento (nel nostro caso siete voi che decidete quando premere Shift-PrtSc, e potete farlo quando vi pare); è quindi necessario che vengano prima salvati e poi ripristinati tutti i registri. Le istruzioni generate automaticamente dal compilatore provvedono a salvare BP nello stack e SP in BP, quelle indicate a pagina 214 del manuale servono a salvare gli altri, con quattro eccezioni: CS, IP, SS e i flag. A CS, a IP e ai flag provvedono la chiamata dell'interrupt e poi l'istruzione IRET, mentre per SS vi sono due casi possibili: o non viene alterato dall'interrupt (che quindi usa lo stesso stack del programma «interrotto»), oppure è la stessa routine associata all'interrupt che, se lo modifica, deve incaricarsi di rimetterlo a posto.

Il «prologo» salva i registri nello stack e poi, con STI, abilita altri interrupt (ogni volta che viene generato un interrupt viene azzerato l'«Interrupt Enable Flag», e ciò fa sì che il processore ignori altri interrupt); l'«epilogo» ripristina i registri ed esegue un IRET.

Quello che si può fare e quello che no

Il programma INCNUM.PAS usa una costante tipizzata. Abbiamo già visto a suo tempo che una costante tipizzata è in realtà una variabile inizializzata, la cui principale caratteristica, per quello che ora ci interessa, è che risiede nel code segment invece che nel data segment.

Una routine definita in un programma Turbo Pascal 3.0 risiede infatti nello stesso code segment in cui si trova tutto il resto del programma, costanti tipizzate comprese, e quindi l'accesso a queste è possibile anche se quella routine viene associata ad un interrupt. Non è così per le normali variabili globali, in quanto queste risiedono nel data segment ed è ben probabile che, quando parte l'interrupt, il registro DS sia stato alterato; questo in particolare quando si tratta di un interrupt chiamato dalle funzioni del DOS, come il 24H. È tuttavia possibile dichiarare una costante tipizzata di tipo integer, assegnare a questa il valore di DS (fornito dalla funzione predefinita DSeg; cfr. pagina 205 del manuale), e usarla poi per riassegnare al registro DS il valore che questo aveva prima dell'interrupt. Il manuale accenna a questa tecnica a pagina 215, ma non fornisce alcun esempio; vediamo quindi come modificare INCNUM .PAS per fare di «n» una variabile globale.

Invece della dichiarazione della costante avremo:

```
const
  DataSeg: integer = 0;
var
  n: integer;
```

Nel main body, prima di sostituire l'interrupt, assegneremo a DataSeg il valore ritornato da DSeg (DataSeg := DSeg); nella procedura Int5, *dopo* aver salvato DS nello stack (ad esempio subito prima di STI), aggiungeremo:

```
$2E/$8E/$1E/DataSeg/
```

ovvero: MOV DS, CS:DataSeg.

Ci sono però problemi per i quali la soluzione è molto meno semplice, per i quali ci limitiamo ad un breve cenno.

Il DOS può fare «una sola cosa per volta»: non può eseguire una funzione senza aver terminato quella che stava eseguendo. Un interrupt può partire in qualsiasi momento, anche mentre il DOS è alle prese con input da tastiera, output alla stampante, ecc. Se la routine associata all'interrupt usa a sua volta le funzioni del DOS, c'è il rischio che il DOS non ci capisca più niente (il problema è stato discusso in termini meno banali nelle conferenze C e TURBOPAS di MC-Link).

Si tratta di una limitazione piuttosto pesante, e le possibili soluzioni sono tutte un po' acrobatiche (uso cauto del BIOS, scrittura di proprie routine di I/O, uso di funzioni non documentate del DOS). Per fortuna in alcuni casi si dispone di margini di manovra un po' più ampi, ed uno di questi casi è proprio quello dell'INT 24H: possono essere usate le funzioni del DOS chiamate tramite INT 21H con numero da 01 a 0CH.

Il mese prossimo vedremo come procedere in pratica. MC

Computer Discount Italia®

la più ampia selezione ai prezzi più competitivi

HARDWARE

hardware originale con garanzia ITALIANA di 1 anno

PERSONAL COMPUTER

Olivetti M10 Portatile, 8KB, completo di alimentatore, borsa e manuale	395.000
Olivetti M15 Portatile, 2 FDU 3 1/2", 512KB RAM, con alimentatore, cavi, borsa, manuale e MS-DOS 3.2 originale	1.595.000

Olivetti M24 SP (10 MHz)

1 FDU 360KB, 1 HDU 20MB, 640KB RAM completo

L. 2.990.000 + I.V.A.

Olivetti M24 "grigio", 2 FD 360 KB, 640 KB RAM completo	2.250.000
Olivetti M24 "grigio", 1 FDU 360KB, 1 HDU 20MB, 640KB RAM completo.	2.750.000
Olivetti M24 "bianco", 2 FDU 360KB, 640KB RAM completo	2.150.000
Olivetti M24 "bianco", 1 FDU 360KB, 1 HDU 20MB, 640KB RAM completo.	2.650.000
Olivetti M28, 1 FDU 1.2MB, 1 HDU 20MB, 512KB RAM completo	**3.750.000**
Olivetti M240, 2 FDU 360KB, 640KB RAM completo	2.590.000
Olivetti M240, 1 FDU 360KB, 1 HDU 20MB, 640KB RAM completo	3.290.000
Olivetti altri modelli	Telefonare
BIT Computers	**Telefonare**
Portatili ZENITH	Telefonare

STAMPANTI - PLOTTER

Stampante Panasonic KX-P1081 - 80 col., 120 c.p.s.	490.000
Stampante Panasonic KX-P1083 - 80 col., 240 c.p.s.	850.000
Stampante Panasonic KX-P1592 - 136 col., 180/270 c.p.s.	**840.000**
Stampante Panasonic KX-P1595 - 136 col., 240/360 c.p.s.	1.150.000
Stampante Panasonic KX-P1540 - 24 aghi - 240/360 c.p.s.	1.290.000
Stampante Laser Epson GQ-3500	3.490.000
Stampante NEC P2200, 24 aghi in STOCK	**Telefonare**
Stampanti NEC 24 aghi	Telefonare
Stampanti Epson	Telefonare
Stampanti OKI	Telefonare
Plotter Panasonic VP-6803P	1.850.000
Plotter Roland DXY 880/A	**1.590.000**
Plotter PL10 per Olivetti M10	295.000

DISK DRIVE - STREAMER

Hard Disk Seagate ST225, 20 MB completo di controller e cavi	**550.000**
Hard Disk NEC D5146H "veloce", 40 MB	890.000
Hard Disk NEC 20 MB su scheda	950.000
Hard Disk Fujitsu 40MB, 3.5" "veloce"	990.000
Hard Disk Fujitsu 70MB "Full-size"	**1.750.000**

HARDCARD PLUS

Hard Disk su scheda, semplicità di installazione e affidabilità, 35 ms. tempo medio di accesso

Versione 20MB L. 1.190.000 + I.V.A.

Versione 40MB L. 1.590.000 + I.V.A.

Garanzia 1 anno

Hard Disk Alta Capacità Voice Coil DRI	Telefonare
Floppy Drive da 3 1/2" completo di kit per l'alloggiamento e cavi	**290.000**
Box Autoalimentato Esterno per Hard Disk e Sistemi di Backup	390.000
Streamer Esterno 60MB TECMAR trasportabile, per DOS e XENIX V	2.090.000
Host Adapter per Streamer Esterno, versione XT-AT e comp.	350.000
Host Adapter per Streamer Esterno, versione Microchannel	450.000
Altri Sistemi di Backup	Telefonare

MONITOR

Monitor NEC MultiSync GS 14" Monoc.900x700	490.000
Monitor NEC MultiSync II 14" Colore 800x600	**1.290.000**
Monitor NEC MultiSync Plus 15" Colore 960x720	1.890.000
Monitor NEC MultiSync XL 20" 1.024x768 Colore	4.490.000
Monitor 14" Colori per Olivetti M24-M28 Basculante, Ris. 640x400	790.000
Monitor Hantarex Boxer 14" CGA/Herc.	280.000
Monitor Colore Hantarex EGA/CGA 14"	750.000

Monitor NEC MultiSync II
+
Scheda VEGA DE LUXE originale

Risoluzione 752x410 a 16 colori

L.1.790.000 + I.V.A.

MODEM

Smartmodem "HAYES" originale, interno PC, 1200/2400 baud, omologato SIP	1.190.000
Modem con telefono "WD 1600", Comp. Hayes, V21/V22, 300/1200 baud	395.000
Pocket Modem 1200P, Comp. Hayes, V21/V22, 300/1200 baud, alim. a batteria	550.000
Modem MC10 per Olivetti M10	290.000

SCHEDE - CHIP - MOUSE

Coprocessore Matematico Intel 8087 5 Mhz	275.000
Coprocessore Matematico Intel 8087 8 Mhz	**350.000**
Coprocessore Matematico Intel 80287 6 Mhz	380.000
Coprocessore Matematico Intel 80287 8 Mhz	630.000
Coprocessore Matematico Intel 80287 10 Mhz	690.000
Coprocessore Matematico Intel 80387 16 Mhz	1.090.000
Microsoft Mouse per PS/2	295.000

MICROSOFT MOUSE

Lo standard di mercato

L. 280.000 + I.V.A.

Scheda Microsoft MACH 10	590.000
Scheda Microsoft MACH 20	Telefonare
Scheda VEGA VGA	**750.000**
Scheda VEGA DE LUXE	590.000
Schede Varie	Telefonare

VARIE

Tavoletta Grafica Summagraphics 12"x12", stylo, curs. 4 puls., alim. e cavo	1.250.000
Fotocopiatore Triumph-Adler, Mod. 209 (Mita 1001)	1.390.000
Telefax Canon Modello 210	2.950.000

Telefax MURATA M-1

Gruppo III, 9600 baud, (anche fotocopiatore)

L.1.990.000 + I.V.A.

Garanzia 1 anno

Macchine per Scrivere Olivetti	Telefonare
Calcolatrici Olivetti	Telefonare

SOFTWARE

software originale, sigillato, con garanzia ufficiale e possibilità di aggiornamento

Prodotti Microsoft Corp.

Excel (It.)	**790.000**
Word 3.0 (It.)	690.000
Word 3.0 Network (5 users) (It.)	1.990.000
Word 3.0 (XENIX)	890.000
Word 4.0	**Telefonare**
Multiplan 3.0 (It.)	390.000
Multiplan 2.0 (XENIX)	429.000
Chart 2.0 (It.)	395.000
Chart 3.0	590.000
Combinazione Multiplan 3 (It.) Chart 2 (It.)	690.000
Works (It.)	**350.000**
Rbase System (It.)	1.090.000
Project 3.0 (It.)	690.000
Project 4.0	Telefonare
Access	390.000
Windows 2	**Telefonare**
Windows 386	**Telefonare**

BASTA CON LE COPIE!
IL SOFTWARE ORIGINALE È MIGLIORE

Excel (Apple Macintosh) (It.)	690.000
Word 3 (Apple Macintosh) (It.)	690.000
Works (Apple Macintosh) (It.)	**450.000**
File (Apple Macintosh)	295.000
QuickBASIC	**165.000**
QuickC	**185.000**
BASIC Interpreter	590.000
BASIC Compiler	460.000
C Compiler	**595.000**
Windows 2 Toolkit	**690.000**
COBOL Compiler	990.000
FORTRAN Compiler	595.000
BASIC Interpreter (XENIX)	495.000
BASIC Compiler (XENIX)	**990.000**
COBOL Compiler (XENIX)	1.390.000
FORTRAN Compiler (XENIX)	990.000
Pascal (XENIX)	990.000
Prodotti Lotus Development Corp.	
Lotus 1-2-3 Rel. 2.01 (It.)	690.000
Lotus Symphony 1.2 (It.)	**890.000**
Manuscript	690.000
Freelance Plus	690.000
Prodotti Ashton-Tate	
Framework II (It.) + controllo ortografico	1.090.000
dBASE III Plus (It.)	**990.000**
Rapid File (It.)	590.000
Multimate Advantage II	790.000
Prodotti Micropro International	
WordStar 4.0 (It.)	595.000
WordStar 2000 Plus Rel 2.0 (It.)	790.000
Prodotti Autodesk	
AUTOCAD Base (It.)	690.000
AUTOCAD ADE 2 (It.)	4.360.000
AUTOCAD ADE 3 (It.)	5.950.000
Altre Marche	
Aldus PageMaker (It.)	**1.090.000**
MS-DOS 3.1 originale Olivetti con GW-BASIC	100.000
MS-DOS 3.2 + GW-BASIC Italiano Originale Microsoft	145.000
Nantucket Clipper	1.190.000
RANK XEROX Ventura Publisher (It.)	1.390.000

DISCHETTI

FORMATO 3 1/2"		FORMATO 5 1/4"	
		Dischetti Bulk	750
Micro Mito SS/DD	2.190	DS/DD Made in U.S.A.	850
Micro Mito DS/DD	2.600	**Duratech DS/DD**	**1.050**
SONY SS/DD	**2.600**	Mito DS/DD	1.750
SONY DS/DD	3.500	Mito-Quad 96TPI 780KB	1.950
SONY DS/HD 2MB	8.900	Mito-Mega 96TPI 1.2MB	3.100

Garantiti 20 milioni di passaggi, ordine minimo 100 pezzi

Termini e condizioni di vendita

◆ Prezzi unitari al netto di I.V.A. ◆ **Spese di spedizione a carico della COMPUTER DISCOUNT.** ◆ Pagamento in contrassegno con assegno circolare intestato a COMPUTER DISCOUNT S.R.L. ◆ Ci riserviamo di accettare ordini di importo inferiore a L. 300.000. ◆ La merce si intende salvo il venduto. ◆ Altri prodotti disponibili su richiesta. ◆ **Leasing e pagamenti rateali.** ◆ Contratti di assistenza. ◆ **Ulteriori sconti per quantità.** ◆ La presente offerta è valida sino al 15 giugno 1988 e sostituisce ogni nostra precedente offerta.

per ordini ed informazioni telefonare allo

055 - 22.99.851

oppure scrivere a

Computer Discount S.R.L.

Via Accursio, 2 - 50125 FIRENZE

Telefax 055-2280674