VIC da zero

di Tommaso Pantuso

*Sul numero 28 di MC abbiamo introdotto la nozione d'*interrrupt *ed abbiamo discusso sull'importanza di una tale tecnica per il colloquio di un microprocessore con le sue periferiche. Abbiamo anche visto, sempre in linea teorica, come esso viene gestito dal 6502, il processore del VIC-20. Vogliamo oggi ritornare su tale argomento per analizzarlo più a fondo e vedere come sia possibile, da parte dell'utente, manipolare la* routine d'interrupt *di tale computer per asservirla ai propri scopi.*

L'interrupt del VIC

Pensiamo sia il caso di fare un breve riepilogo sul modo in cui viene gestito l'interrupt nel VIC per dar modo a tutti di capire gli argomenti che seguiranno, anche a quelli che hanno avuto la "sfortuna" di non aver letto l'articolo contenuto nel numero cui abbiamo accennato.

La funzione dell'interrupt è sintetizzata brevemente nella parola stessa che significa interruzione. In pratica quando un microprocessore riceve una richiesta d'interruzione, esso smette di eseguire ciò che stava eseguendo, va a controllare chi ha inviato la richiesta, effettua le operazioni specifiche comandate per il tipo di periferica che deve servire, eseguendo alcune routine dette di *manipolazione dell'interrupt*, e ritorna a fare ciò che stava facendo prima dell'interruzione. Naturalmente, prima di diramare verso la routine di manipolazione, un processore deve conservare tutte le informazioni che gli saranno utili al rientro per riprendere il lavoro da dove lo aveva interrotto, come se nulla fosse successo; queste informazioni sono conservate in un posto chiamato *area di stack* ma per ora non ci occuperemo di questa. Aggiungiamo che ci sono due tipi fondamentali di interrupt: uno di tipo mascherabile, IRQ, ed uno di tipo non mascherabile, NMI.

```
ROUTINE DI MANIPOLAZIONE
DELL'IRQ

FF72 : 48          : PHA
FF73 : 8A          : TXA
FF74 : 48          : PHA
FF75 : 98          : TYA
FF76 : 48          : PHA
FF77 : BA          : TSX
FF78 : BD 04 01    : LDA  $0104,X
FF7B : 29 10       : AND  #$ 10
FF7D : F0 03       : BEQ  $FF82
FF7F : 6C 16 03    : JMP ($0316)
FF82 : 6C 14 03    : JMP ($0314)
```

Figura 1 - Questa routine è la prima che viene eseguita dopo una diramazione a causa di una richiesta d'interruzione e provvede a salvare il contenuto dell'accumulatore, del registro x e del registro y nello stack. Le ultime due istruzioni sono dei salti a due diverse locazioni che contengono un puntatore a 16 bit: il primo viene effettuato se si rileva un interrupt di tipo Break ed il secondo se si rileva un IRQ.

Una richiesta di attenzione da parte di una periferica tramite una interruzione del primo tipo può essere nascosta al microprocessore che quindi la ignorerà e proseguirà il suo lavoro, mentre una richiesta del secondo tipo giungerà sempre al microprocessore e non c'è modo di nasconderglielo.

Fatto questo breve riepilogo, occupiamoci più da vicino del VIC-20 e dei suoi interrupt sulla linea IRQ (Interrupt ReQuest).

Su tale linea il processore del VIC riceve sessanta richieste di interruzione al secondo. Ciò significa che ogni sessantesimo di secondo esso interrompe il suo lavoro, ad esempio l'esecuzione di una linea di programma, al fine di eseguire le routine di manipolazione. Egli infatti dopo ogni richiesta, completa per prima cosa l'operazione che sta compiendo e poi salta alle locazioni, di memoria ROM, FFFE-FFFF (65534 e 65535 in decimale) che "puntano" ad una routine che manovra l'IRQ.

Molti di voi si chiederanno a questo punto cosa significa dire che due locazioni di memoria "puntano l'indirizzo di partenza" di un sottoprogramma e riteniamo quindi doveroso un chiarimento.

Spesso in un programma si trovano delle istruzioni di salto non diretto tramite le quali il programma va ad esaminare il contenuto di una locazione che contiene l'indirizzo effettivo a cui esso si deve spostare. Tale locazione punta quindi (indica) lo spostamento finale. L'utilità di un tale modo di agire sarà compresa nel corso della lettura di questo articolo.

Per fare un esempio "umano" il sistema si comporta come un fattorino che debba fare delle consegne ma non conosca a priori l'indirizzo dei destinatari perchè esso gli viene comunicato di volta in volta scrivendolo su di un foglietto e depositandolo in una cassetta. Egli deve perciò, ogni volta che vuole consegnare un pacco (il cui contenuto supponiamo sia sempre lo stesso), recarsi nel posto in cui si trova la cassetta, prelevare l'indirizzo ed effettuare la consegna. L'indicazione del foglio di carta equivale al puntatore e la cassetta alla locazione di memoria in cui esso è depositato.

Leggere l'indirizzo che viene indicato da un puntatore in memoria è altrettanto semplice che leggere quello scritto sul foglietto. Come certo saprete (e chi non lo sa lo saprà ora), la memoria del VIC è paragonabile per la sua conformazione ad un libro di 256 pagine su ciascuna delle quali sono scritte (o si possono scrivere) 256 parole. Per completare l'analogia diremo quindi che ogni parola del libro equivale ad un byte nel computer e che ogni pagina equivale a 256 byte. Se torniamo alle locazioni da cui siamo partiti per questa breve digressione, cioè FFFE ed FFFF, se an-

diamo a leggere il loro contenuto tramite il comando *peek* troveremo:

PEEK (65354) = 114
PEEK (65535) = 255.

A questo punto voi direte: "e allora?". Allora il contenuto del registro di valore più elevato indica la *pagina* in cui è contenuto l'indirizzo e quello di valore inferiore indica a quale parola bisogna fermarsi scorrendo la pagina dal basso verso l'alto. In altre parole, l'indirizzo effettivo è fornito dall'operazione:

255 × 256 + 114 = 65394;

è questa una regola generale per calcolare dove puntano due registri tramite il loro contenuto.

Tornando a noi, a 65394 (FF72 esadecimale) inizia una routine, indicata in figura 1, che conserva nello stack, che come detto è un'area appositamente studiata per operazioni di immagazzinamento dati da recuperare in un secondo momento, il contenuto dell'accumulatore del microprocessore (PHA:PusH A = spingi il contenuto di A nello stack); del registro X (TXA : Transfer X A; PHA = trasferisci il contenuto di X in A e spingi poi A nello stack); del registro Y (TYA; PHA, analoga alla precedente). Il *contatore di programma* e il *registro di stato* vengono preservati automaticamente nello stack del 6502.

Dopo tale operazione, il sistema va ad esaminare il contenuto di altre due locazioni di memoria e cioè 0314 - 0315 (788 e 789 decimale) se si è verificato un IRQ e 0316 - 0317 se è avvenuto un altro tipo di interruzione, il BRK (BReaK) di cui noi non parleremo.

Tale contenuto punta, con il solito sistema, ad una locazione di memoria da cui parte un insieme di sottoprogrammi di gestione dell'interruzione. Ricapitoliamo le ultime cose.

Dopo una richiesta di IRQ il processore legge il contenuto dei registri FFFE ed FFFF il quale fornisce l'indirizzo d'inizio di una routine che salva nello stack dei dati importanti e che verranno ripresi al ritorno. Terminata l'esecuzione di tale routine il processore va ad esaminare il contenuto di altri due registri di memoria e precisamente 0314 e 0315 il cui contenuto punta ad una locazione da cui inizia una serie di routine che per il VIC provvedono ad aggiornare l'orologio TI (e quindi TI$), producono il lampeggiamento del cursore, controllano i tasti del registratore e verificano se è stato premuto un tasto sulla tastiera per agire di conseguenza. Questi passaggi sono illustrati nella figura 2.

I lettori più acuti si chiederanno perché l'indirizzo della seconda routine non venga fornito direttamente alla fine della prima senza passare attraverso il puntatore memorizzato in 0314 e 0315 risparmiando così un passaggio.

Il fatto importante è che questi ultimi registri sono due locazioni della memoria RAM a cui noi possiamo accedere da programma per memorizzare in esse un nuovo indirizzo che invii ad una serie di routine da noi progettate per gli scopi più svariati.

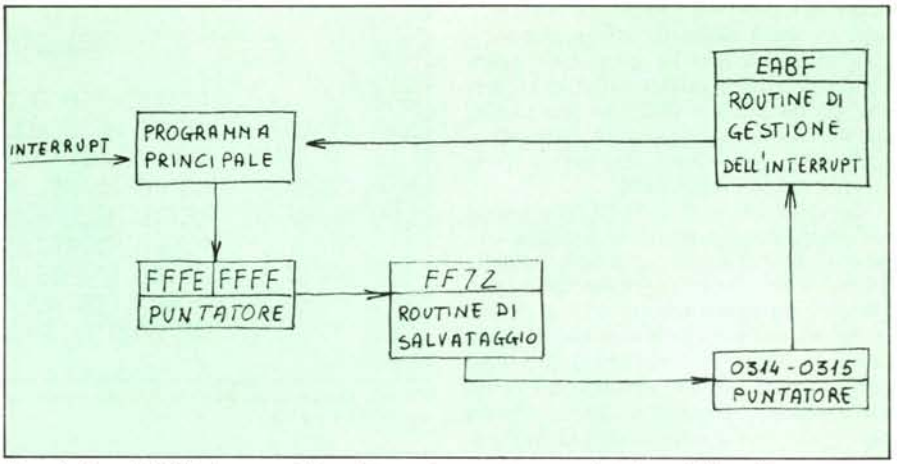


Figura 2 - Quando il 6502 riceve una richiesta di interruzione tramite un segnale inviato sulla linea d'interrupt, esso va all'indirizzo indicato dal contenuto dei registri FFFE ed FFFF (per un IRQ). Da lì inizia una routine di salvataggio di parametri importanti nell'area di Stack che termina con un salto alle locazioni 0314 e 0315, dove è contenuto l'indirizzo di partenza di una serie di routine di gestione. Normalmente per il VIC tali routine partono da EABF e quando è terminata la loro esecuzione il controllo ritorna al programma principale. Tutto questo avviene 60 volte al secondo.

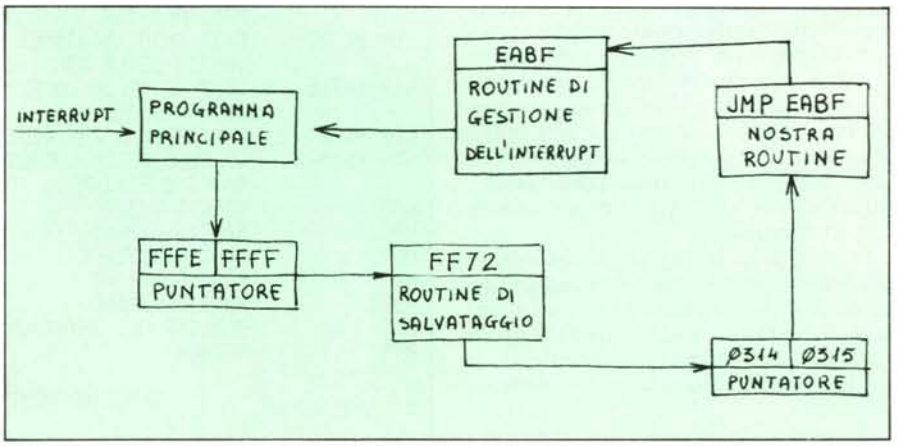


Figura 3 - Noi possiamo modificare il contenuto dei registri 0314 e 0315 per farlo puntare ad una routine scritta da noi alla fine della quale manderemo il controllo ai cicli di gestione dell'Interrupt.

```
10 REM ********************************
20 REM ****** MINIASSEMBLER ESA ******
30 REM ******                  ******
40 REM ******     (C)TP 1984   ******
50 REM ********************************
60 POKE36879,25 :REM CAMBIA COLORE SFONDO
70 PRINT"□MM  MINIASSEMBLER ESA   ■" :PRINT
80 PRINT" M ↑ ■  PER BACK UPM "
90 PRINT" M ★ ■  PER STOPMMMM"
100 PRINT"          ──────────"
110 PRINT"    PREMI UN TASTO
120 PRINT"          ──────────"
130 PRINTTAB(110)"M      (C) TP 1984     ■"
140 GETA$:IFA$=""THEN140
150 INPUT"□MIND, DI START>■";I:PRINT
160 PRINT" IND     OPCOD   DCODM"
170 PRINTI;
180 INPUT"■";A$
190 IF A$="↑"THEN I=I-1:GOTO170
200 IFA$="*"THENEND
210 A=ASC(A$)-48:B=ASC(RIGHT$(A$,1))-49
220 N=B+7*(B>9)+16*(A+7*(A>9))
230 PRINT"□■■■■■■■■■■■■■■■■";N
240 POKEI,N:I=I+1:GOTO170
250 PRINTPEEK(283):GOTO250
260 REM "□" = SHIFT+CLR HOME
270 REM "M" = CURSORE VERTICALE
280 REM "M" = CTRL+RVS ON
290 REM "■" = CTRL+RVS OFF
300 REM "□" = SHIFT+CURSORE VERTICALE
310 REM "■" = CURSORE ORIZZONTALE
```

Listato 1 - Programma per inserire in macchina i codici operativi delle istruzioni del 6502.

Questo nuovo sistema di operazioni è indicato in figura 3.

Il contenuto di tali locazioni è:

PEEK(788) = 191
PEEK(789) = 234

quindi normalmente esse indirizzano alle routine che iniziano da

234 × 256 + 191 = 60095

cioè EABF esadecimale.

Precisiamo che, alla fine di una nostra eventuale routine, dobbiamo inviare il programma a questo indirizzo se vogliamo che vengano eseguite anche le manipolazioni tradizionali.

Capite ora l'importanza che assume il fatto di effettuare dei salti indiretti quando l'indirizzo a cui sono destinati può essere modificato perché depositato in RAM.

Con la speranza di essere stati abbastanza chiari sull'argomento, abbandoniamo la teoria ed andiamo a vedere come si possono sfruttare in pratica le cose descritte.

Modifica del puntatore: il linguaggio macchina

Visto che si tratta di modificare le locazioni di memoria 0314 e 0315 (788 e 789

decimale), potreste pensare che basti scrivere in esse i valori desiderati tramite il solito comando di poke per risolvere il problema. Verrebbe infatti naturale credere che, per inviare il controllo ad una routine che si trova a partire dall'indirizzo esadecimale 6000 (24576 decimale), basti scrivere

```
POKE 788,96 : POKE 789,0
```

(infatti $96 \times 256 + 0 = 24576$) per ottenere l'effetto desiderato, ma le cose non vanno così. Infatti il contenuto delle suddette locazioni può essere aggiornato solo "turbando" momentaneamente il normale svolgimento delle operazioni del VIC, facendo in modo che, per un attimo, esso non risponda più alle richieste d'interruzione permettendoci di andare a scrivere nei registri che ci interessano. Si tratta di un'operazione molto semplice che però richiede di scrivere una brevissima routine in LM che sortisca l'effetto desiderato. Tale routine è riportata in figura 4 ampiamente commentata, ma noi daremo ulteriori chiarimenti per i meno esperti. Questi ultimi potranno o scegliere di approfondire un argomento fondamentale come la programmazione in linguaggio macchina (LM) che non è poi così difficile come sembra, (almeno ai primi livelli), o seguire semplicemente le metodologie che andiamo a descrivere. Noi cercheremo comunque di essere sufficientemente chiari descrivendo, istruzione per istruzione, i programmi in LM che riportiamo.

Sempre ai meno esperti diciamo che il computer comprende solo il linguaggio dei numeri e scrivere in LM equivale ad inserire in macchina un programma formato da numeri, codificando numericamente le istruzioni che vogliamo impartire al microprocessore.

È come se un ricco signore che voglia parlare poco dica al suo maggiordomo "uno" se vuole mangiare, "due" se vuole bere e "tre" se vuole uscire in automobile a fare una passeggiata (per il momento fermiamoci qui). Allora se egli pronuncia, rivolto al maggiordomo, "uno, tre", significa che vuole mangiare e *poi* uscire a fare una passeggiata in automobile, mentre se dice "tre, due" significa che vuole prima uscire in automobile e *poi* (probabilmente al ritorno dalla passeggiata) bere. Le istruzioni numeriche prendono il nome di *codici operativi* e sono rappresentate, per il 6502, da numeri in notazione esadecimale. Essendo difficile ragionare con dei simboli sintetici come quelli numerici, qualcuno ha pensato bene di semplificare la vita ai programmatori, facendo in modo che essi potessero inserire in macchina delle parole che ricordassero l'operazione da compiere lasciando poi alla macchina il compito di trasformarle in un numero. Sono nati così i programmi assemblatori tramite i quali è possibile comunicare alla macchina i codici operativi non direttamente ma tramite un' istruzione detta *mnemonica*. Ad esempio se vogliamo dire alla macchina di caricare un dato nell'accumulatore del microprocessore (un suo particolare registro), per esempio il numero 10 (0A in esadecimale),

```
                ABILITAZIONE DELLA NOSTRA ROUTINE

SEI           : DISABILITA GLI INTERRUPT
LDA #$ LB     : METTE NELL'ACCUMULATORE IL BYTE BASSO
              : DELL'INDIRIZZO D'INIZIO DELLA NOSTRA ROUTINE
STA $0314     : LO COPIA NELLA LOCAZIONE DECIMALE 788
LDA #$ HB     : METTE NELL'ACCUMULATORE IL BYTE ALTO
              : DELL'INDIRIZZO D'INIZIO DELLA NOSTRA ROUTINE
STA $0315     : LO COPIA NELLA LOCAZIONE DECIMALE 789
CLI           : RIABILITA GLI INTERRUPT
RTS           : RITORNA AL PROGRAMMA PRINCIPALE
```

Figura 4 - Questa routine disabilita gli interrupt diretti alla CPU per il tempo necessario alla modifica del contenuto dei registri 0314 e 0315.

```
                          ABILITAZIONE

SEI           : DISABILITA GLI INTERRUPT  : 78          : 120           02A2
LDA #$ B9     : BYTE BASSO LOCAZIONE DI   : A9 B9       : 169 185
              : INIZIO DELLA ROUTINE BEEP :             :
STA $0314     : LO PONE NELLA LOCAZIONE   : 8D 14 03    : 141  20   3
              : DECIMALE 788              :             :
LDA #$ 02     : BYTE ALTO LOCAZIONE DI    : A9 02       : 169   2
              : INIZIO DELLA ROUTINE BEEP :             :
STA $0315     : LO PONE NELLA LOCAZIONE   : 8D 15 03    : 141  21   3
              : DECIMALE 789              :             :
LDA #$ F4     : CARICA IN A 244 DECIMALE  : A9 F4       : 169 244
STA $900A     : LO PONE NELLA LOCAZIONE   : 8D 0A 90    : 141  10 144
              : 36875 E ATTIVA IL TONO    :             :
NOP           : SPAZIO LIBERO             : EA          : 234
NOP           : SPAZIO LIBERO             : EA          : 234
NOP           : SPAZIO LIBERO             : EA          : 234
NOP           : SPAZIO LIBERO             : EA          : 234
NOP           : SPAZIO LIBERO             : EA          : 234
CLI           : ABILITA GLI INTERRUPT     : 58          :  88
RTS           : RETURN                    : 60          :  96

                       DISCRIMINAZIONE

LDA #$ 40     : 64 DECIMALE               : A9 40       : 169  64       02B9
BIT $00C5     : ESEGUE L'AND DI A CON IL  : 2C C5 00    :  44 197 0
              : CONTENUTO DEL REG. 197    :             :
BNE 02E3      : SE NON E' SODDISFATTO     : D0 23       : 208  35
              : SALTA ALL'ULTIMA ISTRUZ.  :             :

                         BEEP SOFT

LDA #$ 0F     : 15 DECIMALE IN A          : A9 0F       : 169  15
STA $900E     : LO PONE NEL REGISTRO DEC. : 8D 0E 90    : 141  14 144
              : 36878 E ATTIVA IL VOLUME  :             :
LDA #$ 00     : 0 DECIMALE IN A           : A9 00       : 169   0
STA $911B     : LO PONE NEL REGISTRO DEC. : 8D 1B 91    : 141  27 145
              : 37147 POSIZIONANDO ACR    :             :
LDA #$ 00     : 0 DECIMALE IN A           : A9 00       : 169   0
STA $9118     : CARICA LATCH BASSO DI T2  : 8D 18 91    : 141  24 145
LDA #$ FF     : 255 DECIMALE IN A         : A9 FF       : 169 255
STA $9118     : CARICA LATCH ALTO  DI T2  : 8D 19 91    : 141  25 145
              : E AVVIA IL CONTEGGIO      :             :
LDA #$ 20     : 32 DECIMALE IN A          : A9 20       : 169  32
BIT $911D     : ESEGUE L'AND DI A CON IL  : 2C 1D 91    : 44   29 145   02D6
              : TENUTO DEL REGISTRO IFR   :             :
BEQ 02D6      : SE NON E' SODDISFATTO     : F0 FB       : 240 251
              : VA  ALL'ISTRUZ. PRECED.   :             :
LDA $9118     : CARICA IN A IL CONT.DI T2 : AD 18 91    : 173  24 145
LDA #$ 00     : 0 DECIMALE IN A           : A9 00       : 169   0
STA $900E     : LO PONE NEL REGISTRO DEC. : 8D 0E 90    : 141  14 144
              : 36878 E DISATTIVA VOLUME  :             :
JMP $EABF     : SALTA ALL'INTERRUPT ORIG. : 4C BF EA    : 76  191 234   02E3
```

Figura 5 - Questa routine produce un beep, udibile dall'altoparlante del televisore, ogni volta che viene premuto un tasto.

dobbiamo comunicare, avendo a disposizione un tastierino numerico, l'istruzione A9 0A che significa proprio: carica nell'accumulatore (A9) il numero 10 (0A).

Avendo invece a disposizione una tastiera alfanumerica ed un programma assemblatore, potremo comunicare alla macchina l'istruzione

LDA 0A

dove LDA deriva dalla parola LoaD A cioè *carica l'accumulatore* che viene convertita in termini numerici dall'assemblatore (assembler). Tutto qui. Per fare i primi programmi non rimane quindi che imparare i codici operativi od i comandi mnemonici delle istruzioni del microprocessore che si ha a disposizione (a seconda che si possegga o no un assembler), e cominciare a fare un po' di pratica.

Nel listato 1 (pag. 87) vi presentiamo un programmino, impropriamente chiamato MINIASSEMBLER ESA, che vi permette di inserire in macchina direttamente i codici operativi delle istruzioni in esadecimale, in quanto penserà esso a tradurli in notazione valida per il Basic del VIC (decimale). Non pensiamo sia il caso di ulteriori commenti in proposito, in quanto il programma stesso invierà i messaggi di richiesta delle opportune informazioni. Aggiungiamo che

IND. DI START >

richiede l'indirizzo da cui si vuole incominciare a memorizzare il programma.

Primo passo: la routine di abilitazione

In figura 4 è riportata la routine in LM che permette di modificare il contenuto dei registri 0314 e 0315 per fare puntare il loro contenuto a quella costruita da noi: descriviamola.

SEI

disabilita gli interrupt forzando ad "uno" il bit di disabilitazione del *registro di stato* del microprocessore. Dopo questa istruzione la CPU non risponde più alle richieste di interruzione. Possiamo a questo punto andare a modificare il valore del puntatore. Questo si fa ponendo il byte basso dell'indirizzo a cui si punta, LB, in 0314 e quello alto, HB, in 0315 (ricordate che, come detto, deve sempre essere: HB × 256 + LB = indirizzo puntato).

Ciò può essere ottenuto con la sequenza

```
LDA #$ LB
STA $0314
```

cioè: carica nell'accumulatore, A, il numero esadecimale LB e copialo nel registro 0314;

```
LDA #$ HB
STA $0315
```

stessa procedura per HB.

Tenete presente che il trasferimento di un numero in memoria con il 6502 non può essere fatto direttamente, ma bisogna prima depositare il numero in oggetto in un registro di passaggio, chiamato accumulatore, e poi da questo trasferirlo all'indirizzo desiderato.

CLI : CLear I

rimette a zero il bit di disabilitazione degli interrupt, contenuto nel *registro di stato* e riabilita la macchina a ricevere di nuovo le interruzioni che seguiranno. Ricordatevi di inserire sempre dopo una routine di manipolazione degli interrupt, questa istruzione per evitare che vadano perse delle richieste d'interruzione.

RTS : ReTurn to Subroutine

rimanda il controllo al programma principale e chiude la routine.

A questo punto la macchina, 60 volte al secondo, andrà ad eseguire il programma che inizia all'indirizzo esadecimale HBLB (ricordate che gli indirizzi devono essere memorizzati in due byte, cioè essere a 16 bit, per poter coprire tutta la memoria da 0 a 65535).

Una volta inizializzato il sistema, bisogna necessariamente che esso trovi ad HBLB una routine "eseguibile" (vedremo poi in che senso), altrimenti può succedere di tutto, ad esempio che la macchina si blocchi.

Bisogna quindi che il programma di abilitazione descritto sia seguito subito dalla routine da noi progettata.

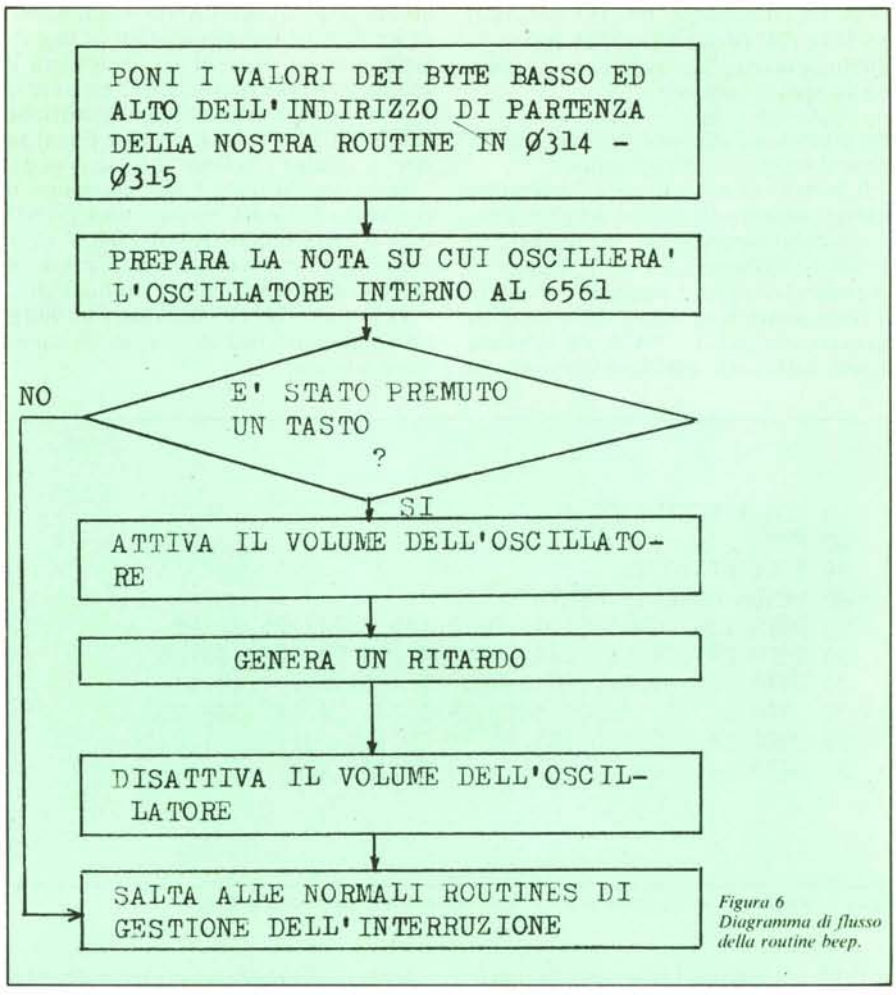


Figura 6
Diagramma di flusso della routine beep.

La nostra routine: il beep

La routine che andremo ad inserire serve a produrre un suono, ogni volta che si preme un tasto sulla tastiera, che verrà emesso dall'altoparlante del televisore collegato al computer (chi possiede un monitor non provvisto di altoparlante non si senta escluso dal discorso perché abbiamo pensato anche a lui!).

Di questo programma viene riportato il listato in figura 5.

Si osservi che, come specificato all'inizio dell'articolo ed illustrato nella figura 3 (pag. 87), esso termina con un'istruzione di salto all'indirizzo EABF da dove iniziano le *routine di gestione dell'interrupt*, alla fine delle quali si ritorna al programma principale (JMP EABF = salta all'indirizzo esadecimale EABF; JMP deriva da JuMP).

Passiamo alla descrizione del programma. Esso è diviso in tre sezioni: quella di abilitazione, quella di discriminazione che serve a verificare se è stato premuto un tasto e la routine vera e propria denominata beep soft.

I vari passaggi saranno più chiari se si osserva il diagramma logico che descrive l'algoritmo impiegato, illustrato in figura 6.

La routine inizia dall'indirizzo esadecimale 02B9 (697 decimale) poiché da 02A1 a 02FF (673 e 767 decimali) è situato un "pezzo" di RAM non utilizzata dal sistema e sufficiente ad ospitare programmi in LM lunghi fino a 94 byte. Questo indirizzo di partenza è posto nello locazioni 0314 e

0315, rispettivamente: B9 (185 decimale) in 0314 e 12 (2 decimale) in 0315. Infatti, se effettuate la semplice verifica che vi abbiamo insegnato, otterrete:

$2 \times 256 + 185 = 697$

che è proprio l'indirizzo decimale da cui inizia il nostro sottoprogramma.

Il *beep* è ottenuto tramite l'oscillatore audio contenuto all'interno del chip 6561. Come certo saprete, esso può oscillare su tre ottave differenti (cioè su tre gamme di frequenza) ed ognuna di queste può essere attivata ponendo il valore della nota da generare nel registro 36874 per la prima ottava, nel registro 36875 per la seconda ed

```
10 REM * BEEP SOFT *
20 REM
30 FORI=674TO741
40 READA:POKEI,A:NEXT
50 DATA 120,169,185,141,20,3,169,2,141,21,3,169
60 DATA 244,141,10,144,234,234,234,234,234,88,96
70 DATA 169,64,44,197,0,208,35,169,15,141,14,144
80 DATA 169,0,141,27,145,169,0,141,24,145,169,255
90 DATA 141,25,145,169,32,44,29,145,240,251,173,24
99 DATA 145,169,0,141,14,144,76,191,234
```

Figura 7 - Programma per introdurre in macchina la routine in LM che produce il beep.

in 36876 per la terza. I valori da introdurre per ottenere le differenti note variano da 128 a 254. Per udire effettivamente un suono, che viene emesso dall'altoparlante, bisogna abilitare il volume dello stadio finale di amplificazione del chip che varia tra quindici livelli possibili, regolabili tramite un valore compreso tra 0 e 15 posto nel registro 36878. Il valore "0" inibisce l'audio.

Durante la routine di abilitazione viene introdotto nel registro 900A (36874 decimale) il valore F4 (244 decimale), che abilita l'oscillatore ad oscillare sulla frequenza di circa 395 hertz. Gli spazi liberi lasciati in tale sezione del programma serviranno ad inserire in seguito delle altre istruzioni che esplicheranno un'importante funzione, ma per ora non ci occuperemo di esse.

Passiamo a descrivere il comportamento della parte denominata DISCRIMINAZIONE.

Nella locazione C5 (197 decimale) è contenuto un valore che varia a seconda del tasto premuto e ne rappresenta il codice di tastiera. Se non viene pigiato alcun tasto, tale registro contiene il valore 40 (64 decimale): la nostra routine comincia col caricare nell'accumulatore il valore esadecimale 40 e ne esegue l'AND con il contenuto del registro suddetto (BIT 00C5). Se il contenuto di 00C5 e dell'accumulatore (che in questo momento contiene 40) sono uguali, significa che non è stato premuto nessun tasto, quindi l'AND è soddisfatto ed un flag del registro di stato (il flag Z) viene posto a zero: in tali condizioni il sistema non deve continuare ad eseguire la nostra routine e salta all'ultima istruzione (BNE 02E3 = Branch on Not Equal to zero = dirama alla linea 02E3 se Z = 0).

Se invece un tasto è stato premuto, il contenuto di A e di C5 sono diversi, l'AND tra essi non è soddisfatto ed il flag Z viene posto ad 1. In tali condizioni si passa ad eseguire il resto della nostra routine e cioè:

a) si pone 0F (15 decimale) in 900E (36878 decimale) ed in tal caso il volume viene attivato;

b) si predispone uno dei timer del VIA 6522 a funzionare come generatore di ritardo hardware;

c) si caricano i latch ed i contatori con il valore del ritardo e si avvia il conteggio.

Quando il conteggio è terminato, viene posto ad *uno* il bit corrispondente a T2 (che è il timer che stiamo usando) nel registro dei flag del 6522 posto all'indirizzo 911D (37149 decimale).

Quando tale flag è ad *uno*, nel registro 911D si trova il valore 20 (32 decimale): noi carichiamo allora 20 nell'accumulatore ed eseguiamo l'AND con il contenuto di tale registro. Se esso non è soddisfatto, significa che il timer non ha finito il conteggio e si riesegue l'AND, altrimenti si riazzera il registro dei flag del VIA, grazie all'artificio di una lettura nel registro contenente il latch basso di T2 (9118), e si salta all'indirizzo EABF da cui partono le consuete routine di gestione delle interruzioni.

Non ci siamo soffermati molto sull'uso dei registri del 6522 poiché lo abbiamo fatto già ampiamente nei numeri 28 e 29 della rivista e dovreste perciò aver acquisito una certa dimestichezza con essi.

Ed ora?

Se osservate il listato della routine descritta, esso riporta nella parte destra la codifica delle istruzioni mnemoniche in esadecimale ed in decimale. Le prime possono essere introdotte tramite il programma MINIASSEMBLER ESA riportato in questo articolo, oppure, come ormai ben sapete, memorizzate tramite il comando POKE.

Non dovete stancarvi a comporre il programmino che esplichi l'ultima funzione descritta perché ve lo forniamo noi in figura 7.

Copiate tale programma ed avviatelo. Naturalmente dopo il RUN non osserverete nulla di diverso ma non preoccupatevi perché il programma è stato memorizzato e quindi potete anche dare il NEW che cancellerà le istruzioni in Basic (se non lo fate è lo stesso).

Ponete al giusto volume il televisore e digitate

```
SYS 674
```

che renderà operativa la routine di preparazione dell'interrupt e ritornerà poi al Basic. Da questo momento in poi potrete usare normalmente il vostro VIC, solo che ora, ogni volta che voi premerete un tasto, udirete un *beep* prodotto dall'altoparlante del TV.

Simpatico, no?

I commenti

Benché la routine funzioni bene in pratica e sia sufficientemente valida in linea teorica, essa può spesso risultare scomoda per determinate operazioni. Infatti, se bisogna semplicemente scrivere o listare dei programmi o cose del genere, tutto va bene, mentre se si deve fare un uso molto frequente del cursore per inserzioni o correzioni, la velocità di ripetizione viene turbata dall'eccessivo ritardo.

Noi abbiamo introdotto il ritardo massimo ponendo in 9118 il valore FF, cioè 255 decimale. Voi potrete modificarlo con il comando

```
POKE 720,N
```

dove N può variare da 1 a 255, che modifica una linea del programma in LM, oppure sostituendo lo stesso valore nell'ultimo dato della linea 80 del listato della figura 7.

Vi accorgerete che sotto un certo valore, normalmente sotto N = 100, il suono udito sarà sempre più distorto e simile più ad UN PRR... che ad un beep. Decidete comunque voi cosa volete fare, se cambiare tale valore o no.

Questo problema sarà risolto nel prossimo numero dove presenteremo lo stesso programma modificato in modo che possa fornire un segnale di controllo ad un circuitino esterno appositamente studiato, il quale provoca un *beep* ogni volta che è sollecitato e genera un proprio ritardo indipendentemente dalla durata del segnale di controllo.

Ci preoccuperemo di evitare che la nostra routine venga disabilitata premendo insieme i tasti STOP e RESTORE descrivendo il modo in cui disabilitare il tasto RESTORE.

Vi forniremo inoltre altre routine di interesse ed utilità.

Appuntamento al prossimo mese.

PHILIPS

PHILIPS

Siate all'avanguardia con PHILIPS

È facile da usare e da trasportare; Vi seguirà da ufficio a ufficio ed in un attimo sarà pronto all'uso. Gestirà per Voi listini, budget, bilanci e proiezioni.
Sarà la macchina da scrivere preferita della Vostra segretaria, sarà la soluzione per la Vostra amministrazione.

P2000 C un passo avanti nell'ufficio

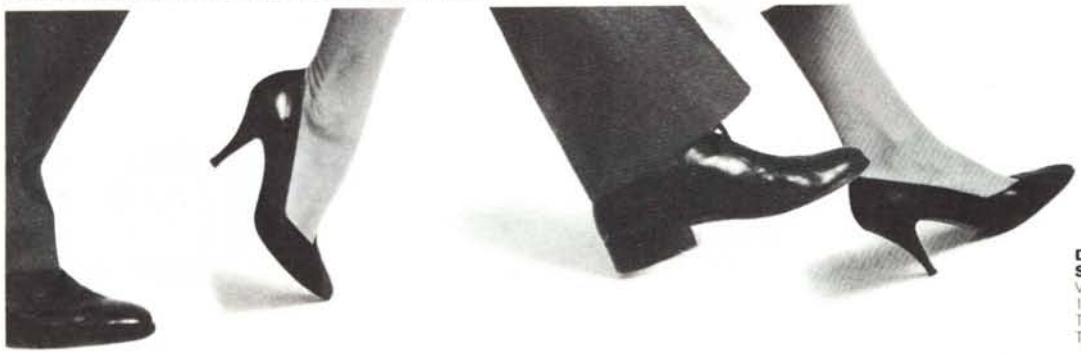
con software compreso: i notissimi WordStar* e CalcStar*; TESI* un prodotto Sigesco che Vi gestirà lo schedario, gli archivi, lo scadenziario, e ogni tipo di informazione.
Presso i Distributori Sigesco, pronti per una dimostrazione, programmi per ogni esigenza.

P2000 C

1 - 64 Kb di RAM utente, 256 Kb di RAM aggiuntiva per disco virtuale.
2 - 2 floppy da 5" 1/4 con capacità fino a 640 Kb cadauno.
3 - CP/M* per un immediato accesso alla più ampia libreria di software esistente.
4 - Monitor 9", 24 linee per 80 colonne 32 Kb di RAM per gestione video, 512x252 punti.
5 - Interfaccie: RS 232, floppy esterni, hard disk, data comunication, monitor esterno, IEE 488.

da **L. 3.650.000**
software compreso con 2 FD da 160 Kb cadauno, CP/M*, WordStar*, CalcStar* e TESI*

a **L. 4.950.000** con 2 FD da 640 Kb, cadauno, CP/M*, WordStar*, CalcStar*, MailMerge*, InfoStar*, TESI*



* WordStar, CalcStar, MailMerge, InfoStar, sono marchi della MicroPro International.
* CP/M è un marchio della DIGITAL Research.
* TESI è un marchio della Sigesco Italia S.p.A.

Distributore ufficiale per l'Italia:
SIGESCO Italia S.p.A.
Via Giulia di Barolo, 22 bis
10124 TORINO
Tel. (011) 839.81.81 (centr.)
Telex 220533 GALIL-I - Telefax 518612

informatique
Mail Service

LOCKSMITH 5.0 238.000 + IVA
(Anche per APPLE IIe)
SNAPSHOT COPYKIT 299.000 + IVA
(per Apple IIe)
THE SAVIOR 238.000 + IVA
(Locksmith per IBM PC/XP)

IL PIÙ GRANDE ASSORTIMENTO DI PERIFERICHE PER APPLE E IBM

NOVITÀ PER APPLE

ADATT. DA CONTR. NORMALE A DUODISK	85.000 + IVA
COMP.6502-Z80-64K TAST. STACCATA	1.250.000 + IVA
EPROM PROGRAMMER 27 16/32/64	345.000 + IVA
ESP.192K + 80 COL.X APPLE II/E(COMPR.64K)	499.000 + IVA
LOCKSMITH 5.0 COPY PROTECTED DSK	238.000 + IVA
MR. FIX-IT SUPER UTILITY OMEGA	199.000 + IVA
SCHEDA D/A PARLANTE APPLE II + /E	98.000 + IVA
SCHEDA Z80 + 64K (NEW CP/M 3.0)	499.000 + IVA
SNAPSHOT COPYKIT PER APPLE II/E	299.000 + IVA
TASTIERA PER MUSIC SYS.4 OTTAVE	1.350.000 + IVA
TIPO GRAPPLER + CON BUFFER 16-64K	349.000 + IVA

NOVITÀ PER IBM PC

ESPANSIONE 64-512K + SERIALE 232	599.000 + IVA
HERCULES GRAPHIC CARD PER PC IBM	1.189.000 + IVA
KOALA PAD PER IBM PC ED XT	349.000 + IVA
LOTUS 1-2-3	1.285.000 + IVA
MICROSOFT 64-256 + SER. + PAR. + CLOCK	947.000 + IVA
MICROSOFT MOUSE PER PC IBM	450.000 + IVA
MULTIMATE WORD PROCESSOR PER IBM	990.000 + IVA
QUADRAM QUADLINK (PC IN APPLE)	1.650.000 + IVA
THE SAVIOR LOCKSMITH PER PC IBM	238.000 + IVA
U-MICRO 4 FUNZ (EXP/SER/PAR/CLK)	850.000 + IVA
U-MICRO IBM CONV A/D 12bit 16CH	1.250.000 + IVA
U-MICRO IBM PC I/O BOARD 48 CH	470.000 + IVA

HARDWARE

COMPUTERS	
COMP.48K + TAST.NUM + 32 FUNZIONI	980.000 + IVA
COMP.64K-6502 + Z80 + TAST.NUM. + FUNZ	1.150.000 + IVA
OSBORNE ONE PORTABLE 2 × 100K	1.950.000 + IVA
DISK-DRIVES-INTERFACCE PER APPLE	
APPLE DISK CONTROLLER ORIGINALE	176.000 + IVA
DISK CONTROLLER APPLE II COMPAT.	95.000 + IVA
MITAC DRIVE 1423K APPLE COMPAT.	499.000 + IVA
SLIM DISK-DRIVE PER APPLE II	499.000 + IVA
ESPANSIONI/COPROCESSOR PER APPLE	
COPROCESSOR MICROFRAME Z80 CP/M	199.000 + IVA
COPROCESSOR MOTOROLA 6809/APPLE	599.000 + IVA
SCHEDA ESPANSIONE MICROFRAME 128	499.000 + IVA
SCHEDA ESPANSIONE MICROFRAME 16K	120.000 + IVA
U-MICRO 68000 COPROC. × APPLE II	950.000 + IVA
SCHEDE AD 80 COLONNE PER APPLE	
64K + 80 COL. PER APPLE II/E	219.000 + IVA
SCHEDA 80 COLONNE COMPAT. VIDEX	199.000 + IVA
U-TERM SCHEDA 80 COLONNE	299.000 + IVA
INTERFACCE PRINTERS PER APPLE	
GRAF. + INTERFACCIA tipo GRAPPLER	129.000 + IVA
INT.CENTRONICS TIPO EPSON ONE	89.000 + IVA
INT.CENTRONICS TIPO EPSON TWO	129.000 + IVA
MBI VIP CARD GRAF/SER/PARALLELA	249.000 + IVA
INTERFACCE BUFFERIZZ. PER APPLE	
INT. BUFFER 16K PAR/SER/GRAFICA	449.000 + IVA
BUFFERS ESTERNI STANDARD	
BUFFER 8K CENTRONICS/CENTRONICS	249.000 + IVA
BUFFER ESTERNO CENT/CENT 16-64K	299.000 + IVA
CLOCKS/CALENDARS PER APPLE	
APPLETIME INTERF.CLOCK/CALENDAR	199.000 + IVA
U-DT DIGITAL I/O TIMER	275.000 + IVA
U-MICRO CLOCK CALENDAR TIMER	275.000 + IVA
U-TIM INTERFACCIA TIMER	215.000 + IVA
INT.COMUNICAZIONI SERIALI E PAR.	
INTERFACCIA SERIALE RS232C	129.000 + IVA
SCHEDA 6522 PARALLELA UNIVERSALE	129.000 + IVA
U-MICRO U-S232 INT.SERIALE COMPL.	199.000 + IVA
INTERFACCE PER RETI PER APPLE	
U-NET CAVO DI COLLEGAMENTO	49.000 + IVA
U-NET SATELLITE KIT	249.000 + IVA
U-NET STARTER KIT	999.000 + IVA
CONVERTITORE A/D D/A PER APPLE	
CONVERTITORE A/D 87us 16 CANALI	298.000 + IVA
CONVERTITORE A/D 87us 8 CANALI	240.000 + IVA
SCH.PARAL.UNIV. 24 FILI CON 8255	240.000 + IVA
SCHEDA 16 INPUT OPTOISOLATI	395.000 + IVA
SCHEDA 16 OUTPUTS OPTOISOLATI	395.000 + IVA
U-A/D CONVERTITORE 12 BITS 25us	1.150.000 + IVA
U-BCD CONVERTITORE PER DPM	215.000 + IVA
HARDWARE MISCELLANEOUS PER APPLE	
BAR WAND PENNA OTTICA A BARRE	199.000 + IVA
EPROM CON INVERSE PER VIDEX	29.000 + IVA
JOYST. × APPLE II + E/E AUTOCENTERING	47.000 + IVA
MUSIC SYSTEM a 16 registri	599.000 + IVA
SNAPSHOT TWO (solo per AP II +)	249.000 + IVA
SPEECHLAB SCHEDA PARLANTE	199.000 + IVA
SUPERTALKER SCHEDA PARLANTE	199.000 + IVA
U-MICRO PROTEZIONE HARDWARE	99.000 + IVA
SCHEDE PER IBM PC E XT	
CMC INTERF. × MACC.SCR.IBM	990.000 + IVA
MBI IC-MAGIC	199.000 + IVA

SOFTWARE

SOFTWARE VARIO PER APPLE	
BOOT PER VISICALC CON VIDEX	29.000 + IVA
BOOT per A. WRITER 2.0 con VIDEX	29.000 + IVA
DAKIN'S PROGRAMMING AIDS DOS 3.3	199.000 + IVA
DOS SOURCE LISTATO DEL DOS 3.3	49.000 + IVA
HI-DOS VIRTUAL DISK E ROUT.128K	29.000 + IVA
MANUALE MUSIC SYSTEM + DISCHETTI	49.000 + IVA
THE FILER-UTILITIES PER DOS 3.3	40.000 + IVA
THE MANAGER DOS RELOCATOR	29.000 + IVA
U-MICRO PERSPECT DRAWING PACKAGE	90.000 + IVA
U-MICRO VERSA VISICALC EXPAND	49.000 + IVA
VISI 255 ADVANCED (NEW FEATURES)	49.000 + IVA
VISI + CONSOLIDATOR per VISICALC	29.000 + IVA
SOFTWARE COMINFOR PER APPLE	
COMINFOR ADA-ANALISI DATI	499.000 + IVA
COMINFOR APPLE'S DOCTOR	49.000 + IVA
COMINFOR DATA BASE	179.000 + IVA
COMINFOR DOCTOR MATRIX ±1	99.000 + IVA
COMINFOR PTERO WORD PROCESSOR	149.000 + IVA
COMINFOR RELAX PTERO TO P.D.B.	99.000 + IVA
SOFTWARE OMEGA MICROWARE × APPLE	
THE INSPECTOR DISK UTILITY	115.000 + IVA
WATSON DISK LOGICAL UTILITY	115.000 + IVA
LINGUAGGI E S.O. PER APPLE	
FORTH 79 WITH MANUAL	79.000 + IVA
PACKAGE COMPLETO PER 6809	199.000 + IVA
U-MICRO STRUCTURED BASIC APPLE II	199.000 + IVA
C.A.L.L. APPLE SOFTWARE	
CALL APPLE BIG MAC MACROASS + TED	29.000 + IVA
CALL APPLE GLOBAL PROGR.LINE ED.	29.000 + IVA
CALL APPLE HIGHER FONTS DISCO	25.000 + IVA
CALL APPLE HIGHER TEXT PLUS	29.000 + IVA
CALL APPLE SYMBOL SIMON ASS.DBUG	29.000 + IVA
CALL APPLE THE SPREADSHEET 2.0	99.000 + IVA
APMAIL PRO MAIL LIST PER PRO-DOS	48.000 + IVA
PROZAP ZAP UTILITIES PER PRO-DOS	35.000 + IVA
CATER KILLER GIOCO GRAFICO	19.000 + IVA
DISK ANALYZER UTILITY PER DISCO	19.000 + IVA
MICRO WRITER IIe WORD PROCESSOR PER APPLE IIe	29.000 + IVA
SCRAMBLER UTILITY PER DISCO CON PROTEZIONE	19.500 + IVA

garanzia 90 giorni

PRINTERS EPSON e TALLY: TELEFONATE!
PRODOTTI APPLE RICHIEDETE LE CONDIZIONI!

(Sui prodotti APPLE non effettuiamo mail service ma offriamo le migliori condizioni del mercato).

apple computer
Macintosh
Lisa
DISTRIBUTORI PREFERENZIALI Verbatim
CONCESSIONARIO IBM PERSONAL COMPUTER
IBM

tessera
super sconto fedeltà
per maggiori informazioni
linea calda telefonica
(0165-765173-765174)

CONDIZIONI DI VENDITA

Inviate il tagliando compilato accompagnato da lire 2000 in francobolli per ricevere tutto il pacchetto di informazioni relative ai prodotti ed alle condizioni di spedizione e pagamento. Sarete automaticamente inseriti nella nostra mailing list.
Per ulteriori informazioni telefonate al 0165/765173-765174 (Cinzia) le linee sono a vostra disposizione.
Non inviate denaro contante.
L'Informatique si riserva di variare i prezzi in ogni momento a causa della fluttuazione delle valute.

SPEDITEMI:
A) Informazioni e listini su carta (allego lire 2.000)
B) Gli articoli indicati nella lettera allegata (firmata) e di cui questo tagliando fa parte integrale

Nome .....
Cognome .....
Indirizzo .....
Telefono .....
Firma .....

Spedire a: INFORMATIQUE Avenue du Conseil Des Commis, 16 - 11100 Aosta

HOT-LINE è:
AOSTA – Informatique
BRESCIA – Il computer
MANTOVA – Antek Computers
RIMINI – Computer Center
ROMA/LATINA – Easy Byte
TORINO – AB Computer
TORINO – Cominfor
TRENTO – SI. GE. Computer Shop

HOT LINE
risorse, idee e soluzioni