RIVISTA SU CASSETTA

MSX COMPUTER MAGAZINE

N.10 Sped in abb. post. Gr. III L. 9.000

10 PROGRAMMI SU CASSETTA

BY M. MRSEK

CORSO L/M

UNA MAGLIETTA IN REGALO!

a chi si abbona a

MSX COMPUTER MAGAZINE

✸

sei magnifiche cassette di programmi di gioco e di utilità, sempre più belle e ricche!

✸

il prezzo dell'abbonamento (Lire 50 mila) è bloccato per sei numeri e non ti verranno quindi richiesti aumenti (già subito intanto risparmi 4 mila lire)!

✸

avrai subito, direttamente a casa, un'elegante maglietta (realizzata con le riviste consorelle Elettronica 2000 e Load'n'Run) assolutamente gratis!

ABBONATI OGGI STESSO

Basta inviare un vaglia ordinario (quello rosa, da richiedere in un qualunque ufficio postale) di lire 50 mila.
Indica esattamente da quale fascicolo desideri l'abbonamento ed i tuoi dati chiari e precisi. Indirizza a MSX Computer Magazine, C.so Vitt. Emanuele, 15 - 20122 Milano.

MSX Computer Magazine è edita da Arcadia srl,
C.so Vitt. Emanuele 15, Milano.
Tel. 02/706329 (solo giovedì h. 15-18).
Una copia L. 9.000.
Fotocomposizione: Composit.
Stampa: Garzanti,
Milano. Distribuzione: SO.DI.P. Angelo
Patuzzi srl, Via Zuretti 25, Milano.
Registrato Trib. Milano N. 52 del 2/2/85.
Resp. Sira Rocchi.
Sped. in abb. post. Gr. III/70.

Manoscritti, disegni, fotografie
e programmi inviati non si
restituiscono anche se non pubblicati.

MSX COMPUTER MAGAZINE 10

IN QUESTO NUMERO

★ L'ISTRUZIONE DEF FN

★ CORSO LINGUAGGIO MACCHINA

★ IN DIRETTA DAI LETTORI

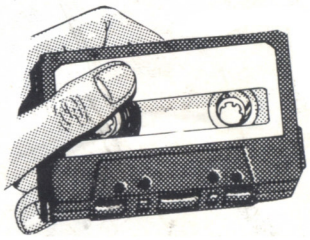
★ DIECI PROGRAMMI DIECI

- THE BALL
- STAR WARS
- TIRO AL PICCIONE
- SALTO IN ALTO
- LUPIN III
- PLAY +
- DRAGAMINE
- L'EREDITÀ
- ITALIA
- PROCOR

MSX TAPE SOFT

I programmi che vi presentiamo in questo decimo numero di MSX Computer Magazine sono tutti compatibili con qualsiasi sistema MSX. Ecco per voi ben 10 programmi.

Ricordate di collegare la spina del controllo motore alla presa REM del vostro registratore, se quest'ultimo la possiede. Assicuratevi che la spina rossa sia collegata alla presa MIC del



registratore e la spina nera alla presa EAR.

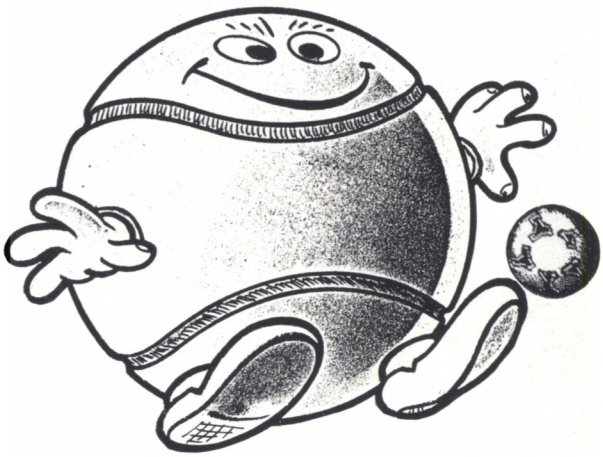
Se il vostro mangiacassette non possiede la presa REM, fate particolare attenzione a quando un programma è stato caricato o deve essere caricato, affinché il nastro scorra per il giusto tempo.

Appena vedete apparire sul video, dopo un comando di caricamento, la

1 THE BALL

("BALL" - 32 KRAM)

di F. Mainieri

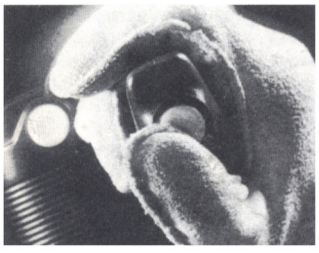


Digitate CLOAD"" e date il RUN. Appena caricato, il programma inizializza tutte le variabili e crea uno schermo di presentazione. Il gioco è davvero ben fatto, la grafica tridimensionale. Dovete aiutare una pallina ad uscire da un complicato labirinto, spostandola con i tasti cursore oppure con il joystick. Attenzione: più tenete premuto il tasto, più la pallina acquista velocità! Ricordatevi che, come ogni pallina che si rispetti, anche la nostra obbedisce alle leggi fisiche di inerzia, attrito e forza di gravità. Sembra facile governarla, ma vi accorgerete presto che non lo è. Qualche suggerimento: il pavimento blu è ghiacciato, quindi prudenza perché manca l'attrito. E non è l'unica difficoltà che incontrerete! La partenza nel labirinto è casuale, così come le stanze del portone e della "chiave". Il portone lo riconoscete facilmente e valicarlo vuol dire avercela fatta: ma si può passare solo se si possiede la "chiave". Attenti al guardiano della porta: è velenosissimo e basta un suo morso per farvi fuori, anche nel caso possediate altre palline d'emergenza. La "chiave" non è rappresentata come tale: si tratta di un fluido magico da assorbire passando sopra un'apposita torretta con uno strano pavimento bianco. Infine, non uscite di pista, potreste cadere. Ovviamente il gioco non dà punteggio perché si tratta di una sfida all'ultima... pallina!

scritta OK, spegnete il registratore.

ATTENZIONE: tutti i programmi vanno caricati con il comando CLOAD (i caratteri racchiusi tra virgolette, nelle parentesi, rappresentano il nome con cui è stato registrato il programma) ed eseguiti con l'istruzione RUN.

Accanto al nome con il quale è registrato il programma vi è l'indicazione



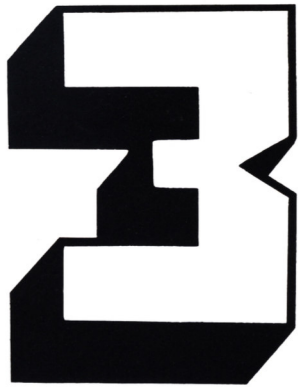
della quantità di memoria che il computer deve avere per caricare il programma stesso.

Nella cassetta allegata a questo fascicolo troverete sul lato A:

The Ball, Star Wars, Tiro al piccione, Salto in alto, Lupin III.

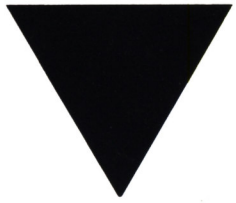
Sul lato B:

Play +, Dragamine, L'Eredità Crackstone, Italia, Procor.



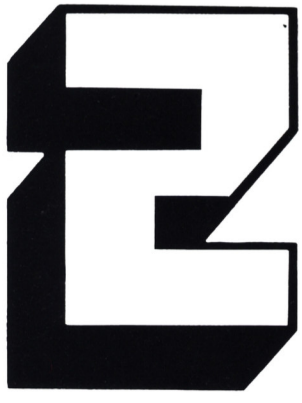
TIRO AL PICCIONE

("CACCIA" - 16 KRAM)



di P. Mattiazzi

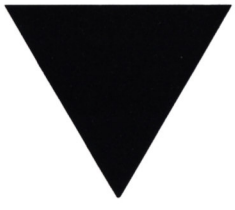
Battete CLOAD "CACCIA" per caricare il gioco, il cui scopo è quello di colpire più piccioni possibile. Ci sono vari livelli ad ognuno dei quali il cacciatore deve abbattere sei prede avendo a disposizione un determinato numero di proiettili. Nel primo livello i colpi a disposizione sono 15, che diminuiranno di 2 nei tre seguenti livelli fino a ridursi a 9. Dal quinto livello in poi aumenterà progressivamente la velocità dei piccioni, tanto per dare un po' di pepe al gioco. Il punto rosso indica il mirino del fucile, che può essere mosso utilizzando il joystick connesso alla porta 1. Per colpire la preda premete il tasto fire. Ogni piccione colpito vale 10 punti. Ricordate che ogni volta che mancate un piccione e quest'ultimo attraversa lo schermo incolume, il proiettile successivo partirà dal punto più lontano dalla traiettoria dell'uccello, ovvero dalla terza base. A proposito... inserite i caps!



STAR WARS

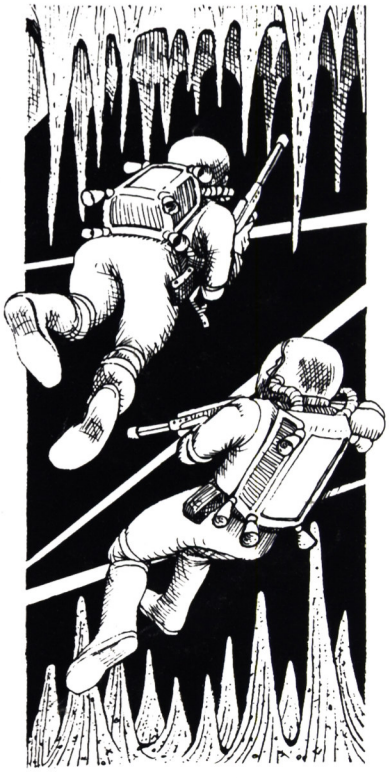
("WARS" - 16 KRAM)

di A. Morreale



Eccovi nel bel mezzo di una battaglia stellare e questa volta non si tratta della solita lenta ed inesorabile discesa di marzianini. Vi trovate invece proprio al centro della scena più famosa di "Guerre Stellari", il noto film di G. Luckas, circondati da uno scenario tridimensionale che ripropone il famoso cunicolo, nel quale sfrecciate a tutta birra a bordo della vostra navetta. Le astronavi nemiche vi vengono incontro spedite: ovviamente il vostro compito è quello di distruggerne il maggior numero possibile. Per colpire le astronavi avversarie dovrete innanzitutto portarvi sulla loro traiettoria (il che richiede intuito e colpo d'occhio), e sparare prima che sia troppo tardi, prima cioè che la navetta nemica vi sia addosso e vi distrugga. Avete a disposizione 3 astronavi (4 se superate i 5.000 punti), ed è possibile giocare sia con il joystick che con la tastiera. Se sceglierete la tastiera avrete ovviamente maggiori difficoltà nell'eseguire le virate in diagonale, che richiedono la pressione contemporanea di due tasti del cursore.

Un consiglio: per individuare la traiettoria delle astronavi nemiche tenete bene d'occhio la loro ombra... Buona caccia! Per caricare digitate CLOAD "WARS".



4

SALTO IN ALTO

("ATL" - 32 KRAM)

di L. Bencivenni

Questa volta dovete percorrere 110 metri simulati, saltando gli ostacoli che vi vengono incontro nel minor tempo possibile.

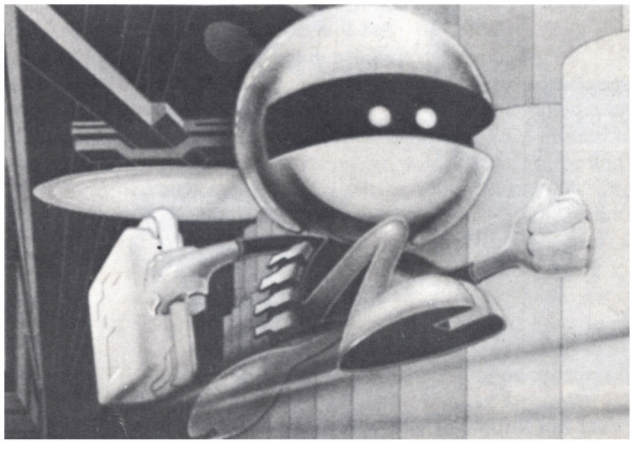
La fatidica scritta Game Over arresterà il gioco. Cozzando contro gli ostacoli, è ovvio, perderete del tempo e sarete superati dagli altri concorrenti. Gli unici comandi da usare sono la barra spaziatrice oppure il tasto fire del joystick 1, comandi che servono anche a ricominciare il gioco.

Listando il programma troverete un'interessante routine in linguaggio macchina che si occupa dello scroll laterale del video. Per caricare il programma digitate CLOAD "ATL".

5

LUPIN III

("LUPIN" - 32 KRAM)



di D. Montresor

Caricato il programma e dato il RUN, bisogna attendere che vengano caricati i due programmi seguenti registrati con BSAVE. Il primo, "CARATT", serve per i caratteri speciali usati nel gioco; il secondo, "LEVEL", serve per le dieci schermate diverse dei relativi livelli.

Scopo del gioco è recuperare il maggior numero di oggetti preziosi (di colore giallo). Per poter passare al livello successivo si debbono prendere i numeri sparsi nella stanza, che servono per aprire la porta a combinazione. Per prendere gli oggetti ed i numeri e per aprire la porta, basta passarci davanti. Gli ostacoli sono i robot di guardia forniti di eliche e, nei livelli successivi al secondo, gli oggetti che intralciano il cammino (di color magenta): con una breve rincorsa e premendo il pulsante di fuoco si possono evitare.

Ogni oggetto recuperato significa 200 punti; ogni livello superato, 1.000 punti; ogni dieci livelli c'è una vita in omaggio. È previsto l'uso sia del joystick che della tastiera.

PLAY +

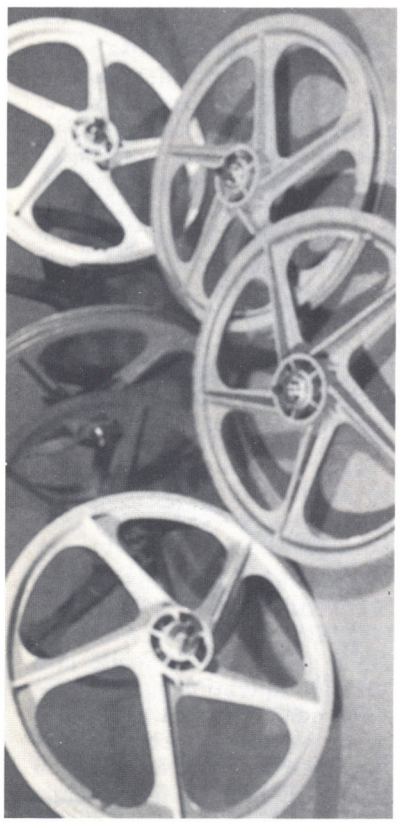
("PLAY+" - 32 KRAM)

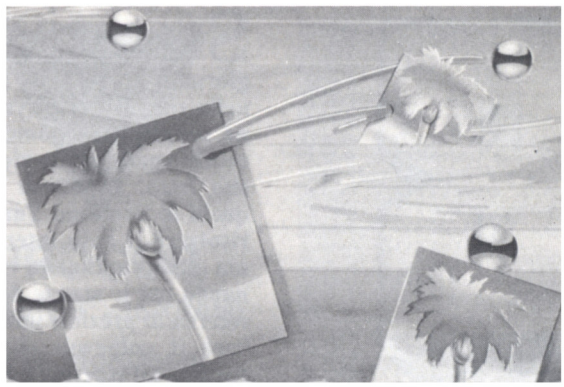
di M. Belardi

È un gioco al quale possono partecipare da 1 a 6 giocatori. Il programma provvede a stilare la graduatoria ed il record, ed è composto di 4 giochi concatenati ai quali si accede facilmente.

Si tratta praticamente di una gara su quattro specialità. Al termine il programma chiede se si vogliono cambiare i giocatori oppure se far giocare di nuovo gli stessi. Nel primo caso vengono azzerati i nomi ed i punteggi dei giocatori, lasciando inalterato il nome del record-man ed il suo punteggio, proponendo così il record da battere; nel secondo caso si azzerano solo i punteggi lasciando inalterati sia i nomi che il record. Il programma è lungo 15K e si carica con CLOAD "PLAY+".

Dopo il caricamento e dato il RUN, compare il titolo e, successivamente, vi verrà chiesto il numero dei giocatori, che dovrà essere da 1 a 6. In seguito si dovranno immettere i nomi dei giocatori (massimo 9 caratteri per nome) e si sceglierà se giocare con il joystick o con la tastiera.



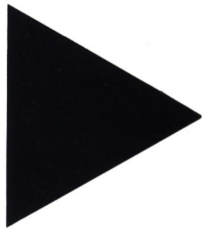


Il tabellone dei record mostrerà tutti i nomi con i relativi punteggi e visualizzerà, in alto, il nome ed il punteggio del record-man.

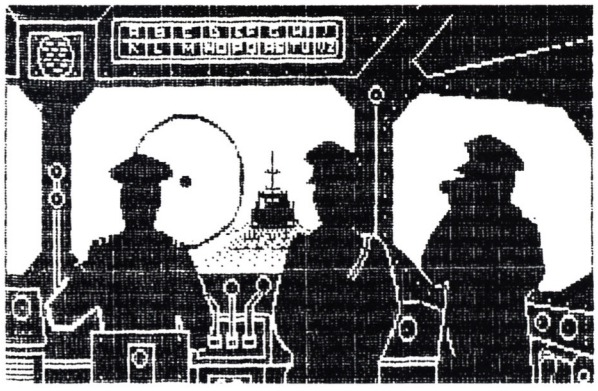
Per iniziare si deve premere il pulsante del joystick (o la barra spaziatrice, se si è scelto di giocare con la tastiera): al centro comparirà il nome del primo giocatore. Premendo ancora, comincia il primo gioco, al termine del quale il computer indicherà il nome del prossimo giocatore e via così fino a che non avranno giocato tutti. Dopo ogni gioco verrà mostrato il tabellone dei punteggi ed il record fino a quel momento realizzato. La sfida prosegue così fino all'esaurimento di tutte e quattro le fasi. Tre giochi sono a prova unica mentre uno, il Breakout, è in 4 fasi: ogni volta che un giocatore perde la palla il gioco passa al prossimo e così via fino a che tutti i giocatori non finiscono le palle a disposizione. Al termine del quarto gioco si vedrà il tabellone e, premendo il pulsante, verrà chiesto se si vogliono cambiare i giocatori mediante una richiesta a tempo. Se non si preme prima che scada il tempo, il gioco ricomicia con gli stessi giocatori ma con i punteggi azzerati, tranne il record.

DRAGAMINE

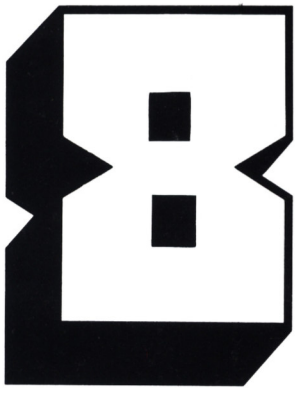
("DRAGAM" - 16 KRAM)



di E. Medvet

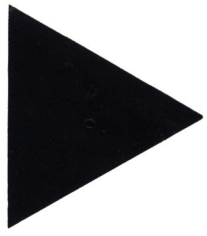


Caricato il gioco con CLOAD "DRAGAM" venite catapultati a bordo di un immaginario sottomarino dal quale, attraverso un potente telescopio, vedete un'abbondante porzione di mare. Purtroppo la zona è minata e non potete attraversarla fino a quando le mine non verranno rimosse. Ai vostri ordini dunque, ecco una nave dragamine abilitata a far esplodere le mine stesse con il lancio di bombe di profondità. Naturalmente tocca a voi trasmettere alla nave le coordinate di arrivo delle bombe, coordinate che dovrete rilevare dalle due colonnine, una orizzontale ed una verticale, in perenne movimento ai bordi del vostro periscopio graduato. Per bloccare il liquido (che scorre nelle colonnine) al punto desiderato basta premere la barra: attenzione ad allineare perfettamente con il punto centrale della mina che volete far esplodere. Spettatore muto ed impassibile dei vostri sforzi, un pesce rosso che assiste alla scena e, con i suoi movimenti, allieta e distende i nervi. Ci sono tre livelli per i quali è richiesta una precisione sempre maggiore. Via dunque con le mine.



L'EREDITÀ

("FLATS" - 32 KRAM)



di N. Paggin

Siete, in questa simpatica avventura-giallo, nei panni del giornalista Francis Flats, ospite nella villa di Lord Cyrus Crackstone proprio nel momento in cui il vecchio gentiluomo viene ucciso nel suo letto. In attesa che arrivino quelli di Scotland Yard, cercate di risolvere il caso interrogando prima gli abitanti della villa ed esaminando poi le stanze della casa alla ricerca di preziosi indizi. Per accusare il colpevole, infatti, dovete fornire all'ispettore Jones di Scotland Yard indizi sufficienti ad incolpare l'assassino (siete in Inghilterra, non in Italia!). Indizi: ogni risposta data dai vari indiziati, oppure ogni oggetto trovato nelle stanze hanno un numero, il numero dell'indizio, appunto. Se l'indizio vi appare importante per la soluzione, consigliamo di segnare su un foglietto il numero corrispondente, che dovrà poi essere fornito come prova al momento della soluzione del giallo. Durante la partita è possibile consultare un indizio fornendo il numero di quest'ultimo al computer (I = VISIONE INDIZI). Movimento: dopo aver interrogato i testimoni è possibile muoversi all'interno della villa per raccogliere indizi (M = MOVIMENTO). In ogni stanza che visiterete dovrete agire come farebbe un buon investigatore APRENDO, ESAMINANDO, LEGGENDO tutto quello che vi capita sotto mano. I comandi si danno come in un classico adventure-game con frasi composte dal verbo più il nome. Ad esempio: APRO ARMADIO, ESAMINO TAVOLO etc. Appendice: consultandola, avrete notizie molto utili su tutti gli abitanti della villa. Soluzione: quando vi sentite pronti a dare la soluzione premete S = SOLUZIONE e fornite prima il nome del colpevole, quindi il numero corrispondente ai 4 indizi che ritenete siano proprio prove.

9

ITALIA

("ITALIA" - 32 KRAM)

di F. Luglio

Ecco ora un programma didattico per imparare, divertendosi, qualcosa della geografia del nostro Paese. Si tratta di riconoscere le regioni italiane, contraddistinte da una lettera dell'alfabeto, e posizionarle nel secondo riquadro in modo da ricomporre l'Italia intera come in un simpatico, insolito puzzle. Premendo il tasto relativo ad ogni regione, la potrete posizionare (sovrapponendo il suo bordo al bordo della regione adiacente), con i tasti cursore, nel secondo riquadro: il primo è d'aiuto per trovare le coordinate esatte. Due puntini fanno da guida (solo due regioni hanno la città). Premendo RETURN, il computer controllerà se la posizione è esatta: se lo è, appariranno i dati relativi alla regione e, premendo nuovamente RETURN, si passerà alla regione successiva. Se invece la posizione non corrisponde perfettamente, bisognerà premere F1 e riprovare. Prima di farlo potrete chiedere aiuto premendo il tasto Z che vi darà le coordinate di colonna e riga e farà riapparire il bordo dell'Italia: fate quindi coincidere lo sprite con i bordi dell'Italia nel primo riquadro. Il puzzle terminerà con il posizionamento esatto di tutte e venti le regioni e con la visualizzazione, da parte del computer, dei dati globali.

10

PROCOR

("PROCOR" - 16 KRAM)

di D. Montresor

```
100 SCREEN0:CLEAR200,
110 ON ERROR GOTO 30
120 PRINT:PRINT"*";:
130 IF A$="M" THEN15
140 IF A$="D" THEN21
150 LINEINPUTA$:A=VAL
160 PRINT:GOSUB280:V=
170 GOSUB240:L=V*16:
:L=L+V:IF E=1 THEN19
180 GOTO160
190 IF A$=CHR$(8) THE
THEN A=A+1 ELSEIF A$=
200 GOTO160
210 LINEINPUTA$:A=VAL
220 FOR L=0 TO 15:GO
):S=S+V:GOSUB290:A=A+
90:PRINT:NEXT
230 PRINT:GOSUB260:IF
240 E=0:GOSUB260:IF A
ELSEIF A$>CHR$(70) TH
```

Ecco un'utility che permette di eliminare tutti gli spazi fra un'istruzione e l'altra di un qualsiasi programma precedentemente salvato su cassetta in codice ASCII. Tutti i REM vengono ridotti al simbolo dell'apostrofo e non eliminati, perché c'è sempre qualche programmatore che riferisce un GOTO o un GOSUB ad una linea di REM e ciò riduce notevolmente lo spazio occupato.

La memoria occupata dal programma (senza contare lo spazio per stringhe e variabili) è di 1250 byte; quando è in funzione occupa invece tutta la memoria.

Dato il RUN, appare la richiesta del nome del programma che si vuole correggere; se questo viene omesso, vengono accettati tutti i programmi ma, in seguito, il programma corretto viene registrato senza nome.

L'utility mostra quindi, di volta in volta, la riga da correggere e la riga corretta, per poter effettuare un controllo visivo non indispensabile; finita la correzione, si preme ENTER per far ripartire il motore del registratore quindi, dopo aver posizionato il nastro al punto voluto, si preme SELECT per iniziare la registrazione del programma accorciato, che occuperà circa il 20% in meno di memoria.

Naturalmente non vengono eliminati gli spazi racchiusi tra le virgolette di PRINT etc.; se il programma da correggere è più lungo di 22KRAM o di 700 linee appare l'apposito avviso ed il programma può venire registrato solo fino alla riga corretta. Quindi è meglio dividere in due il programma da correggere e poi riunire i due spezzoni corretti con MERGE. Adattare l'utility al Quick-disk non è difficile.

UTILITY

L'ISTRUZIONE DEF FN

L'USO DELL'ISTRUZIONE DEF FN E 10 ESEMPI DI FUNCTION UTILI, DA INSERIRE NEI VOSTRI PROGRAMMI.

a cura di G. RICCOBONO

Quando in un programma dobbiamo utilizzare ripetutamente uno stesso insieme di istruzioni, possiamo utilizzare le classiche istruzioni di salto condizionato e incondizionato. Tipicamente, utilizzeremo l'istruzione FOR-NEXT per ripetere sequenzialmente un blocco di istruzioni, mentre utilizzeremo le istruzioni GOTO e GOSUB per saltare, quando necessario, al set di istruzioni a cui siamo interessati.

Quando, in particolare, dopo aver svolto questo set di istruzioni, dobbiamo ritornare nel punto del programma da cui siamo partiti, utilizzeremo l'istruzione GOSUB, che ci consente per l'appunto di chiamare una subroutine da un qualunque punto del programma e ritornare, a svolgimento completato, a quel preciso punto del programma da cui siamo partiti.

Un ruolo analogo viene espletato, da un certo punto di vista, dall'istruzione DEF FN, anche se le modalità di impiego sono diverse.

Questa istruzione ci permette infatti di definire una funzione di più parametri, che possiamo poi richiamare, nel corso del programma, a nostro piacimento ripartendo, a esecuzione avvenuta, dall'istruzione successiva a quella di chiamata della function.

Analizzando quanto detto con un'ottica diversa (e più rispondente al carattere delle function), possiamo dire che definire una function equivale a "inventare" o, per meglio dire, "costruire", una nuova istruzione nella gamma di istruzioni a nostra disposizione.

Ma lasciamo che sia un esempio a chiarirci le idee.

Supponiamo, a titolo puramente esemplificativo, che durante un programma dobbiamo eseguire diverse volte l'elevazione di un numero alla terza potenza. Ogni volta che ciò è necessario dovremo prendere il numero, moltiplicarlo tre volte per se stesso e metterlo in una nuova locazione di memoria; cioè, indicando con X il numero di partenza e con Y il risultato dell'elevazione a potenza, dovremo ogni volta inserire una riga di programma del tipo:

```
Y= X*X*X
```

oppure del tipo:

```
Y=X^3
```

Nel caso specifico dell'elevazione alla terza potenza, dover introdurre una riga siffatta, non costituisce nessun intralcio, essendo la riga stessa corta e di facile scrittura. Ben diversa sarà però la situazione quando anziché elevare un numero a una data potenza, dovremo calcolare espressioni matematiche molto complicate, come ad esempio l'arcoseno e l'arcocoseno che, a differenza dell'istruzione arcotangente, non sono disponibili direttamente nel set di istruzioni messeci a disposizione dal BASIC MSX.

In questo caso torna allora comodo (e anzi consigliabile) l'utilizzo dell'istruzione DEF FN.

Nel caso dell'esempio appena citato, dovremo comportarci così:

```
10 SCREEN 0:COLOR 15,4,4:CLS
20 DEF FN POT(X)=X*X*X
.
.
.
150 Y=FN POT(A)
.
.
.
280 E=FN POT(8)
.
.
.
```

dove alla riga 20 definiamo la funzione POT dell'argomento X come l'elevazione alla terza potenza della variabile X; alla linea 150 poniamo nella variabile Y la terza potenza della variabile A, che chiaramente dovrà essere stata definita precedentemente; alla linea 280 ritroviamo ancora la funzione predefinita POT, solo che stavolta l'argomento è

PHOTO PHILIPS

esplicito (il numero 8).

È chiaro che a partire dalla linea 150, fino a che non si trova una nuova assegnazione della variabile Y, la variabile Y stessa conterrà il valore di X elevato al cubo; analogamente, dalla linea 280 in poi, fino a nuovo ordine, troveremo nella variabile E il valore 512.

Ovviamente la funzione POT potrà essere utilizzata molte altre volte nel corso del programma anzi, affinché diventi vantaggioso utilizzare una DEF FN, bisogna proprio che essa venga utilizzata nel programma diverse volte.

Notiamo da ultimo come, da un punto di vista puramente formale, avremmo potuto sostituire la riga 150 del finto listato precedente, con una chiamata di subroutine, ottenendo qualcosa del tipo:

```
140 X=A
150 GOSUB 1000
160 Y=POT
.
.
.
1000 POT=X*X*X
1010 RETURN
```

Si vede subito come quest'ultimo procedimento sarebbe stato molto più lungo e noioso, rendendo oltretutto facile l'introduzione di errori. Sempre nel finto listato del primo esempio, possiamo notare come la riga 20 costituisca praticamente la definizione di una nuova istruzione, del tutto equivalente, come formato e tipo di utilizzo, alle usuali istruzioni BASIC.

Un ultimo vantaggio da non sottovalutare, è costituito dalla possibilità di passare uno o più parametri, nel caso invece della subroutine, siamo costretti a inserire, nel secondo esempio, la riga 140 al solo scopo di passare il parametro che va' elevato a potenza e la riga 160 per porre il risultato nella variabile Y.

Vediamo adesso insieme alcune DEF FN utili.

1) CONVERSIONE DA ESADECIMALE A DECIMALE

Il BASIC MSX mette a disposizione, attraverso l'istruzione HEX$(n), la conversione di un numero da decimale a esadecimale.

Non è invece presente, nel set di istruzioni MSX-BASIC, la conversione inversa, cioè quella da esadecimale a decimale. Quando necessario, potrete ottenere nei vostri programmi quest'ultima conversione attraverso l'introduzione delle due seguenti righe di programma:

```
10 HX$="0123456789ABCDEF"
20 DEF FN HD(H$) = (INSTR(HX$,MID$
(H$,1,1))—1)*4096
+ (INSTR(HX$,MID$(H$,2,1))—1)*256
+ (INSTR(HX$,MID$(H$,3,1))—1)*16
+ (INSTR(HX$,MID$(H$,4,1))—1)
```

Ricordatevi che è fondamentale la definizione della stringa HX$ prima della definizione della function e che l'argomento da passare alla function (cioè per intenderci H$), deve essere una stringa di 4 caratteri.

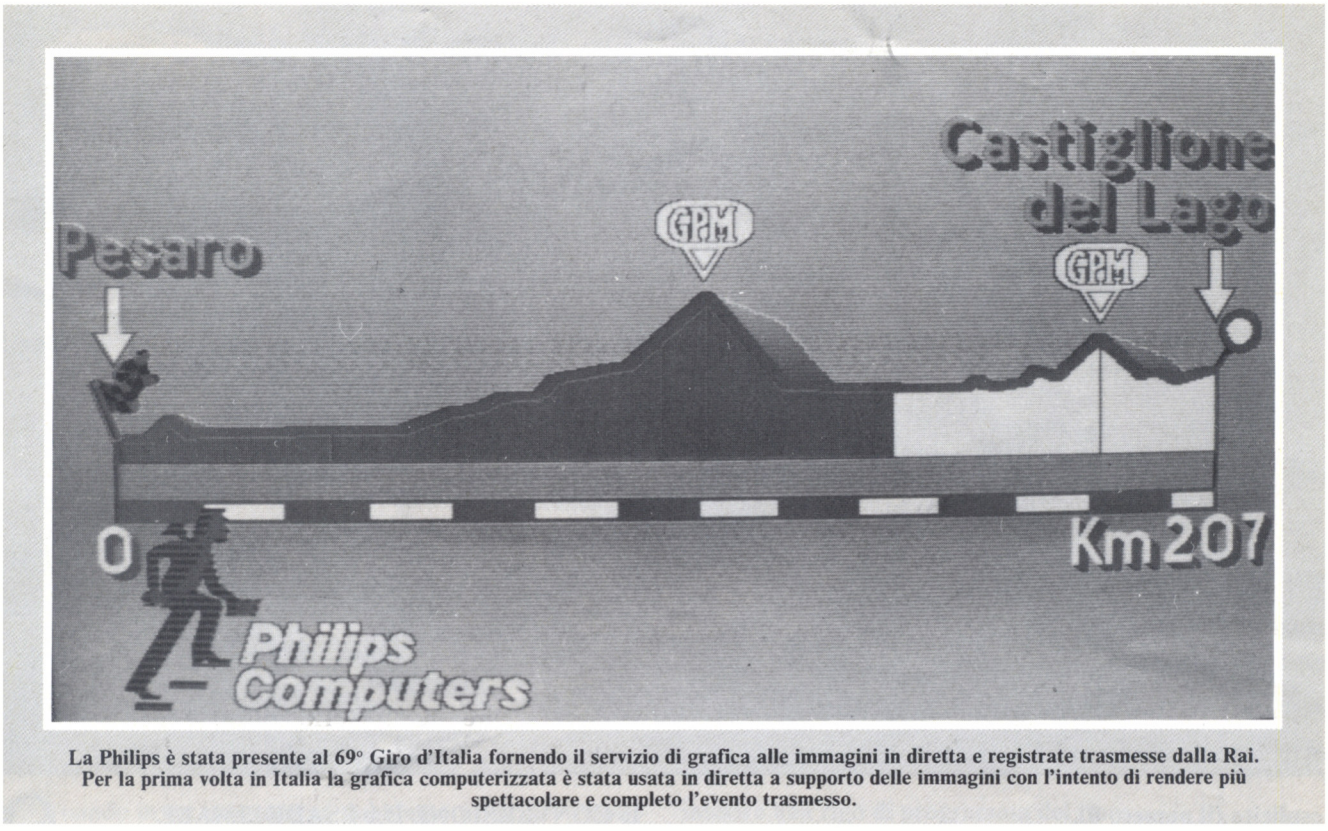
2) CONVERSIONE DA OTTALE A DECIMALE

Anche in questo caso il BASIC MSX ci mette a disposizione solamente la conversione ottale/decimale, ma non l'inversa.

Quest'ultima la potete ottenere attraverso l'istruzione:

```
DEF FN OD(O$) = VAL (MID$(O$,1,1))*32768
+ VAL(MID$(O$,2,1))*4096
+ VAL(MID$(O$,3,1))*512
+ VAL(MID$(O$,4,1))*64
+ VAL(MID$(O$,5,1))*8
+ VAL(MID$(O$,6,1))
```

Anche qui dovete ricordare che la stringa che passate come parametro deve essere di sei caratteri.



La Philips è stata presente al 69° Giro d'Italia fornendo il servizio di grafica alle immagini in diretta e registrate trasmesse dalla Rai. Per la prima volta in Italia la grafica computerizzata è stata usata in diretta a supporto delle immagini con l'intento di rendere più spettacolare e completo l'evento trasmesso.

3) FUNZIONE ARCOCOSENO

Come già accennato, il BASIC MSX contiene nel proprio set di istruzioni soltanto una delle funzioni inverse delle funzioni trigonometriche e per la precisione la funzione arcotangente (ATN). Per avere a nostra disposizione anche l'arcocoseno possiamo definire nei nostri programmi la seguente function:

```
DEF FN ACS(C)=1.5708—2*ATN(C/(1+SQR(1—C*C)))
```

Ricordatevi che il risultato vi verrà dato in radianti, la function per la conversione in gradi verrà presentata nella prossima puntata.

4) FUNZIONE ARCOSENO

Valgono tutte le considerazioni di cui al punto precedente. Per ottenere la funzione arcoseno inserite:

```
DEF FN ANS(N)=2*ATN(X/(1+SQR(1—X*X)))
```

5) COTANGENTE E ARCOCOTANGENTE

Nessuna di queste due istruzioni è presente direttamente nel linguaggio del nostro calcolatore. Avendo però a disposizione la funzione tangente (TAN) e arcotangente (ATN), possiamo ottenere le due funzioni in questione attraverso la definizione delle due function:

```
DEF FN COTAN(X)=1/TAN(X)
DEF FN ARCCTN(X)=ATN(1/X)
```

6) LOGARITMO IN BASE GENERICA

Il linguaggio BASIC dei calcolatori MSX, offre ai suoi utenti la possibilità di calcolare il logaritmo in base "e" di un qualunque numero positivo. Se volessimo calcolare il logaritmo in base A di un numero X (sempre positivo!!) con A diversa da "e", possiamo utilizzare una function del tipo:

```
DEF FN LGA(A,X)=LOG(X)/LOG(A)
```

Ricordatevi che sia la base A che l'argomento X devono essere positivi!! Se poi il logaritmo che vi serve calcolare è sempre, nel corso del programma, con la stessa base, potete evitare di mettere la base come parametro di chiamata e sostituire il "LOG A" a denominatore direttamente col suo valore effettivo.

7) SENO IPERBOLICO

Le quattro funzioni iperboliche possono essere calcolate abbastanza agevolmente attraverso l'uso delle istruzioni DEF FN ed EXP. La prima di esse, cioè il seno iperbolico, lo si può ottenere con:

```
DEF FN SNH(X)= .5 * (EXP(X)—EXP(—X))
```

8) COSENO IPERBOLICO

È ottenibile con:

```
DEF FN CSH(X)= .5 * (EXP(X)+EXP(—X))
```

9) TANGENTE IPERBOLICA

È ottenibile con:

```
DEF FN TNH(X)=1—2*EXP(—X)/(EXP(X)+EXP(—X))
```

10) COTANGENTE IPERBOLICA

È ottenibile con:

```
DEF FN CTNH(X)=1+2*EXP(—X) / (EXP(X)—EXP(—X))
```

ASSEMBLER

IL LINGUAGGIO MACCHINA

COME PROGRAMMARE IN LINGUAGGIO MACCHINA. I CODICI ISTRUZIONE DEL MICROPROCESSORE Z80 A

(5ª PUNTATA)

di EMANUELE DASSI

Prima di aggredire gli argomenti di questa puntata, è opportuno tentare un riassunto di quanto abbiamo appreso sino ad ora. Abbiamo visto che lo Z80 ha 22 registri suddivisi in tre gruppi: i registri principali, quelli secondari ed i registri di uso speciale. A questi registri si assegnano dei valori (istruzioni di caricamento), se ne modificano i contenuti con operazioni aritmetiche, logiche e con la manipolazione dei singoli bit. Vi è infine un registro particolare, quello dei flags F, il cui contenuto segnala alcuni risultati specifici di operazioni aritmetiche, logiche e shift. Ed è grazie al registro F che è possibile eseguire operazioni di test, confronto, e così via.

Lo Z80 è un microprocessore ad 8 bit assai potente che, oltre che offrire le istruzioni di base (quelle già prese in considerazione nelle puntate precedenti), ha anche comandi "strutturati" (o, per dir meglio, istruzioni) che svolgono più operazioni: i comandi di trasferimento blocchi di memoria e di ricerca.

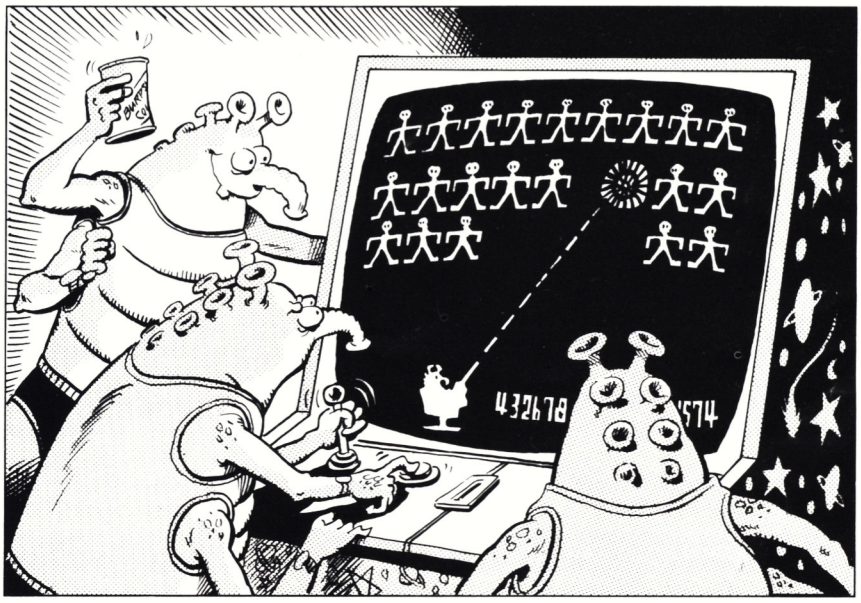
Le istruzioni di trasferimento blocchi di memoria sono: LDI, LDIR, LDD ed LDDR, mentre quelle di ricerca dei blocchi sono: CPI, CPIR, CPD e CPDR.

L'istruzione LDI (LD sta per LoaD - carica - ed I sta per INCREMENT - incremento -) esegue le seguenti operazioni:

1) copia il contenuto della locazione di memoria puntata dal registro HL nella locazione di memoria puntata dal registro DE
2) incrementa di uno il valore di DE
3) incrementa di uno il valore di HL
4) decrementa di uno il valore di BC

In termini di programmazione, l'istruzione LDI funziona come il seguente programmino:

```
LD A, (HL)
```



LD (DE), A
INC DE
INC HL
DEC BC

sintetizzando in un unico comando cinque istruzioni (potente, non è vero?!). Inoltre, il tempo d'esecuzione dell'istruzione LDI è decisamente minore di quello che occorre per eseguire le cinque istruzioni appena viste, e non viene usato l'accumulatore. A questo punto, per comprendere l'importanza di LDI, al di là delle osservazioni appena fatte bisogna commentare il comportamento dei flags. I flag S (segno), Z (zero), e C (carry) rimangono invariati mentre P/V (parità/overflow) risulta essere a 0 se il risultato del decremento di BC è 0, altrimenti ad 1.

A che cosa può servire l'istruzione LDI? La risposta è più chiara con un esempio. Supponiamo di voler creare una subroutine che copi la zona di memoria che si estende dall'indirizzo 10000 all'indirizzo 10500 a partire dalla locazione 20000. La subroutine si realizza nel seguente modo:

LD HL, 10000
LD DE, 20000
LD BC, 500
CONT : LDI
JP PE, CONT
RET

Si predispongono i registri HL, DE e BC come segue: HL punta all'indirizzo iniziale da copiare (10000), DE all'indirizzo dove iniziare a trasferire i dati (20000) e BC il numero di byte da trasferire (500). L'istruzione LDI viene eseguita fino a quando l'istruzione JP PE, CONT risulta essere vera, cioè fino a che il flag PE rimane al valore 1 e quindi BC è maggiore di 0.

L'istruzione LDIR (la R indica Repeat-ripeti-) è una versione più completa del comando LDI. Le quattro operazioni svolte da quest'ultima, infatti, rimangono valide anche per LDIR, la quale però continua a trasferire i dati finché BC non risulta essere uguale a 0. Riprendendo in considerazione il programmino visto sopra ed applicando il comando LDIR, la subroutine di trasferimento dati diventa così:

LD HL, 10000
LD DE, 20000
LD BC, 500
LDIR
RET

In pratica, i parametri per controllare la zona di memoria da trasferire rimangono gli stessi: HL inizio memoria da copiare; DE inizio memoria nella quale trasferire i dati; BC numero di byte da trasferire. LDIR continua ad essere eseguita fino a che BC è diverso da 0, senza bisogno di un'istruzione di JP. L'esecuzione di LDIR pone sempre a 0 il flag P/V e mantiene inalterati gli altri (S, Z e C) proprio perché LDIR è un'istruzione a ripetizione automatica e quindi è inutile, a differenza di LDI, considerare il risultato di P/V.

Vediamo ora di precisare quando usare l'istruzione LDI, oppure LDIR, visto che il loro comportamento è pressoché simile. Lasciando ovviamente piena libertà al programmatore, va precisato che se vogliamo trasferire un solo byte è conveniente usare LDI perché più veloce di LDIR. Viceversa, volendo manipolare più byte, è meglio programmare con LDIR; innanzitutto perché più veloce, infine perché la leggibilità del programma risulta più chiara. Considerando quindi nuovamente la subroutine d'esempio, che deve copiare 500 byte, stabiliamo che è più corretto e più veloce usare LDIR.

Gli altri due comandi di trasferimento dati sono LDD (LD sta per LoaD -carica- e D per Decrement - decremento-) ed LDDR (la R indica Repeat -ripeti-); LDD ed LDDR sono identici ai rispettivi LDI ed LDIR con la sola differenza che, anziché incrementare i registri HL e DE, essi agiscono sugli stessi con un'operazione di decremento. Per il resto, tutte le considerazioni fatte intorno alle prime due istruzioni valgono anche per le seconde. La differenza sostanziale fra LDI-LDIR ed LDD-LDDR è che una zona di memoria verrà copiata a partire dal valore alto per finire al valore basso. Riprendendo l'esempio di questo articolo e volendo utilizzare l'istruzione LDD, la subroutine diventa:

LD HL, 10500
LD DE, 20500
LD BC, 500
CONT : LDD
JP PE, CONT
RET

È chiaro che la copiatura dei 500 byte avviene, per HL, a partire dall'indirizzo 10500 per finire all'indirizzo 10000, mentre per il registro DE va dall'indirizzo 20500 all'indirizzo 20000. Anche in questo caso val la pena di fare qualche considerazione su quando usare istruzioni di trasferimento dati con incremento o decremento. La risposta dipende ovviamente dal genere di problema che bisogna risolvere ma, in generale, l'uso dell'una o dell'altra istruzione è affidato alla libera scelta del programmatore; ciò perché, ripetiamo, le due istruzioni sono praticamente simili.

Passiamo ora a descrivere le istruzioni di ricerca dei blocchi. Così come per i comandi di trasferimento dati, anche per quelli di ricerca dei blocchi le istruzioni CPI e CPIR sono molto simili alle istruzioni CPD e CPDR; la

LISTATO ASSEMBLER

```
10   11   00   00     LD   DE,00 ; contatore dato
20   3E   nn          LD   A,nn ; dato da cercare
30   21   nn   nn     LD   HL,nn nn ; inizio ricerca
40   01   nn   nn     LD   BC,nn nn ; HL+BC = fine ricerca
50   ED   B1   CONT:CPIR
60   E0              RET  PO
70   13              INC  DE
80   18   FA         JR   CONT
```

Serve a contare il numero di byte il cui valore è uguale a quello contenuto nel registro A (linea 20), a partire dall'indirizzo indicato dal registro HL (linea 30) fino all'indirizzo HL+BC. L'istruzione RET PO controlla se tornare al programma chiamante (quindi solo se BC = 0), altrimenti prosegue con l'istruzione successiva (linea 70) la quale incrementa il contatore (registro DE) dei byte trovati.

differenza esistente è indicata nella terza lettera dell'istruzione: I per Increment -incremento- e D per Decrement -decremento-. L'istruzione CPI (CP sta per ComPare -confronto-) esegue queste operazioni:
1) confronta il valore dell'accumulatore con quello della locazione di memoria indirizzata dal registro HL; se i due valori sono uguali il flag Z (zero) viene posto ad 1, altrimenti Z risulta 0
2) incrementa di 1 il registro HL
3) decrementa di 1 il registro BC e, se quest'ultimo è 0, allora il flag P/V (parità/overflow) è posto a 0; altrimenti, è posto a 1.

Il flag S varia a seconda del risultato delle operazioni viste sopra, mentre il flag C rimane inalterato. L'equivalente del comando CPI espresso in altre istruzioni Assembler Z80 è:
CP (HL)
INC HL
DEC BC
quindi l'istruzione CPI svolge la funzione di altre tre istruzioni.

Quale può essere, anche in questo caso, l'utilizzo di CPI? Questa istruzione si rivela molto utile quando si voglia ricercare, in un blocco di memoria, una dato il cui valore è descritto dall'accumulatore.

Supponiamo ad esempio di voler ricercare il primo byte di valore 200 nell'area di memoria che si estende dall'indirizzo 11200 all'indirizzo 11315. La subroutine che svolge tale funzione sarà:
LD A, 200
LD HL, 11200
LD BC, 115
CONT : CPI
JP Z, TROVATO
JP PE, CONT
RET

I registri A, HL e BC vengono inizializzati prima dell'esecuzione del comando CPI. Precisamente, A assume il valore da cercare (200), HL l'indirizzo dove iniziare la ricerca (11200), mentre BC è il contatore dei byte da testare (115). Infatti, a partire dall'indirizzo 11200, vengono confrontati 115 byte fino ad arrivare all'indirizzo 11315. L'istruzione JP Z, TROVATO fà proseguire il programma a partire dall'indirizzo TROVATO e viene eseguita solo se il flag Z è 1, solo cioè se il byte di valore 200 è stato trovato. Altrimenti, il programma passa all'istruzione JP PE, CONT la quale viene eseguita solo se il flag PE è 1, solo cioè se BC non è 0 e quindi non si è arrivati alla fine della memoria da testare, perciò si ripete il controllo con CPI.

L'istruzione CPIR (R sta per Repeat -ripeti-) funziona allo stesso modo di CPI, con la differenza che il controllo ripetuto viene eseguito automaticamente, senza ricorrere ad istruzioni di JP. Il programma prosegue oltre CPIR quando il valore contenuto nell'accumulatore è stato trovato, oppure quando BC ha raggiunto il valore di 0.

Vediamo il precedente programma di ricerca utilizzando l'istruzione CPIR:
LD A, 200
LD HL, 11200
LD BC, 115
CPIR
RET

In sintesi: i parametri utilizzati con CPIR sono gli stessi che con CPI e la differenza con il programma precedente è, oltre che il risparmio di due istruzioni, la maggiore velocità di esecuzione della ricerca del valore 200 nell'area di memoria a partire dall'indirizzo 11200 fino all'indirizzo 11315. Ovviamente il programma che utilizza la subroutine appena descritta dovrà valutare, dopo l'esecuzione di quest'ultima e tramite il registro BC, se il dato di valore 200 è stato trovato. Più precisamente, se BC è uguale a 0 vuol dire che la ricerca ha avuto esito negativo. Viceversa, il registro HL punta all'indirizzo contente il valore 200.

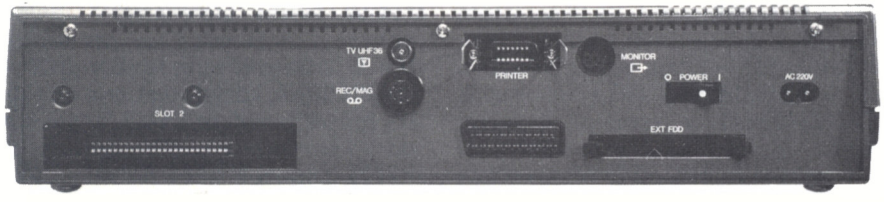
Come abbiamo visto in precedenza, le istruzioni CPD e CPDR sono molto simili alle rispettive CPI e CPIR; l'unica differenza è che il registro HL, anziché essere incrementato, viene decrementato di uno. Così, per realizzare la subroutine di ricerca, utilizzando CPDR inizializzeremo il registro HL all'indirizzo 11315, in quanto verrà decrementato fino al valore 11200.

Il programma del listato numero 1 è, pur essendo breve, di grande utilità e sfrutta l'istruzione CPIR. La sua funzione è quella di contare il numero di byte il cui valore è uguale a quello contenuto nel registro A (linea 20), a partire dall'indirizzo indicato dal registro HL (linea 30) fino all'indirizzo HL + BC. L'istruzione RET PO controlla se tornare al programma chiamante (quindi solo se BC = 0), altrimenti prosegue con l'istruzione successiva (linea 70) la quale incrementa il contatore (registro DE) dei byte trovati. Infatti, se P/V è 1 vuol dire che BC è diverso da 0 e che, essendo usciti dall'istruzione CPIR, è stato trovato un byte di valore uguale a quello dell'accumulatore. Appare chiara l'utilità di questa subroutine in un qualsiasi programma che debba controllare velocemente solo alcuni dati in un blocco ben preciso di memoria.

Concludiamo descrivendo le istruzioni di "exchange" (scambio) tra registri. Per comprendere il loro funzio-

namento bisogna rifarsi agli argomenti esposti nella prima puntata di questo corso e menzionati all'inizio di questo articolo: i registri secondari. Lo Z80, oltre che possedere i registri primari (A, B, C, D, E, H, L), ha anche un banco di registri detti secondari (A', B', C', D', E', H', L'). Questi ultimi possono essere utilizzati in alternativa ai registri primari semplicemente rimpiazzando i loro valori nei registri primari e quelli di questi ultimi nei registri secondari. In pratica, le operazioni dello Z80 agiscono tutte sui registri primari, quindi i registri secondari rappresentano un banco di memoria interno allo Z80 per i registri primari. Le istruzioni che consentono di scambiare il contenuto dei registri secondari con quello dei registri primari e viceversa sono: EX AF, AF' ed EXX. La prima scambia i valori dei registri primari A ed F (F sta per registro dei flags) con i secondari A' ed F'.

Schermo ad 80 colonne, 512 colori grazie al nuovo videoprocessore, memoria RAM da 256 KB, unità a dischi incorporata: queste alcune delle ottime caratteristiche del nuovo MSX 2 Philips (codice VG 8235) qui sopra nella foto.



Il computer potrà essere collegato ad un TV, ad un monitor, ad un registratore, alla stampante etc. Qui sotto, l'interfaccia RS232 della Sony, fondamentale per utilizzare il computer per comunicazioni via modem: a proposito, ricordiamo che è sempre disponibile il nostro servizio gratuito di Posta Elettronica cui ci si può collegare chiamando 02/70.68.57 ad ogni ora.

Il registro dei flags secondario deve essere usato con particolare attenzione per evitare di considerare i "nuovi" flags risultati di operazioni eseguite sotto il controllo dei flags principali.

L'istruzione EXX invece, scambia i valori di B, C, D, E, H ed L con quelli dei registri B', C', D', E', H' ed L'. L'utilizzo di queste istruzioni consente di memorizzare i valori dei registri primari senza ricorrere all'uso della memoria.

Il gruppo delle istruzioni di exchange comprende anche comandi di scambio tra registri primari: EX DE, HL ed EX (SP), rr.

La prima istruzione scambia tra loro i contenuti dei registri DE con HL; la seconda istruzione, invece, scambia i contenuti di memoria puntati dal registro stack pointer SP con i contenuti di rr, dove rr può essere HL, IX oppure IY. Le istruzioni di exchange possono essere paragonate allo statment SWAP presente anche nel linguaggio basic dell'MSX. Difatti le istruzioni:

```
LD B,D
LD C,E
LD D,H
LD E,L
LD H,B
LD L,C
```

oppure:

```
PUSH DE
POP BC
PUSH HL
POP DE
PUSH BC
POP HL
```

svolgono la stessa funzione, in un tempo maggiore ed utilizzando BC, di EX DE, HL.

Anche in occasione di questa puntata del nostro corso di Assembler è apparso chiaro quanto sia potente lo Z80. A questo punto le vostre conoscenze circa la programmazione di questo microprocessore possono dirsi quasi complete... Ma vedremo di risentirci, via via con le cose più importanti.

■

LETTERE

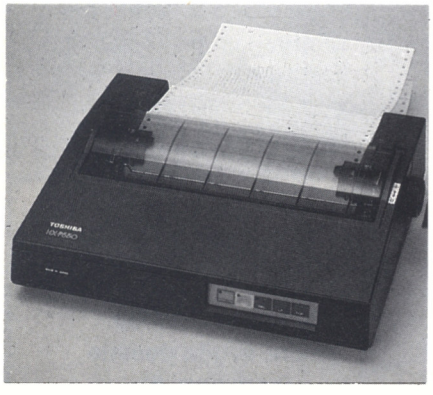
NON GESTITI DA BASIC

Utilizzo un Sony HB 75 in connessione con un Canon VF 100: mi rimangono liberi 23430 KB invece degli usuali 28815. Ciò mi impedisce di trasferire alcuni interessanti programmi da cassetta a disco... Ho pensato di utilizzare il vostro interessante programma apparso nell'articolo della memoria fantasma (MSX n. 5, ndr) ma mi succede che...

Sergio Ferrini - Roma

L'utility cui ti riferisci simulando le istruzioni PEEK e POKE si serve dei 32K RAM non gestiti da Basic. Quindi non è possibile sfruttarla per caricare programmi basic più lunghi di 28815 bytes. Il programma serve solo per memorizzare dati semplici ad un byte tra 0 e 255.

Per le altre tue domande (interfacce utili per usare il computer magari come strumento di misura) ti consigliamo di seguire la nostra rivista consorella Elettronica 2000 che tratta appunto di hardware specifico per microcomputer.

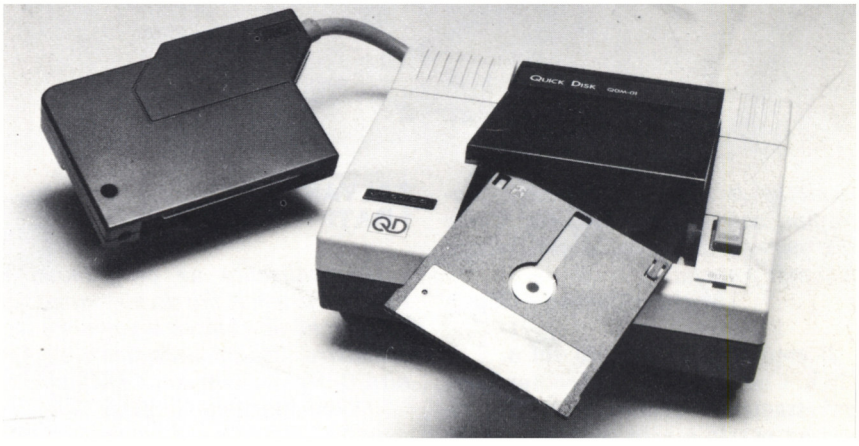


VOGLIO I GRAFICI

Ho scoperto casualmente la vostra rivista in un'edicola napoletana: mi complimento per i contenuti. Ho una Yamaha CX5M che uso per applicazioni musicali e un plotter stampante Sony C41 ma stampo tranquillamente solo i testi mentre in nessun modo riesco a stampare i grafici. Ho provato a...

Ermanno Alviggi - Benevento

Non è questione di computer, floppy-driver o modulo sonoro, il plotter è uno strumento studiato apposta per fare grafici su carta ed ha tutta una serie di istruzioni, richiamabili con caratteri di controllo, capaci di tracciare linee, cerchi, ellissi sul foglio inserito nel plotter. Per fare l'hard copy del video su plotter bisogna utilizzare un programmino ben diverso da quello per una stampante; per esempio per copiare un cerchio dal video su carta non bisogna fare l'hard copy ma basta richiamare su plotter l'istruzione relativa. Consulti quindi il manuale allegato al suo plotter e riproduca l'immagine su video scomponendo la stessa in più elementi geometrici tracciabili dalla periferica grafica.



QUIK DISK

Ho comprato da poco il QUIK DISK, accessorio di una certa utilità. Non riesco però ad immettere da tastiera il nome del file quando lo si apre con l'istruzione OPEN.

Con il registratore aprendo un file con le istruzioni sotto riportate si riesce ad inserire il nome del file da tastiera:

10 input "nome";N$
20 open N$ for output AS # I
30 print # I "segue dati da conservare
40 close # I

Pur tentando in diverse maniere con il QUIK DISK non sono riuscito ad ottenere quanto desideravo. Ottengo sempre messaggi di errore. Per questo chiedo aiuto con la speranza che su un prossimo numero avrò la risoluzione del problema.

È da tener presente che battendo la seguente linea: OPEN "QD:N$" FOR OUTPUT AS # I si salvano i dati con il nome N#.

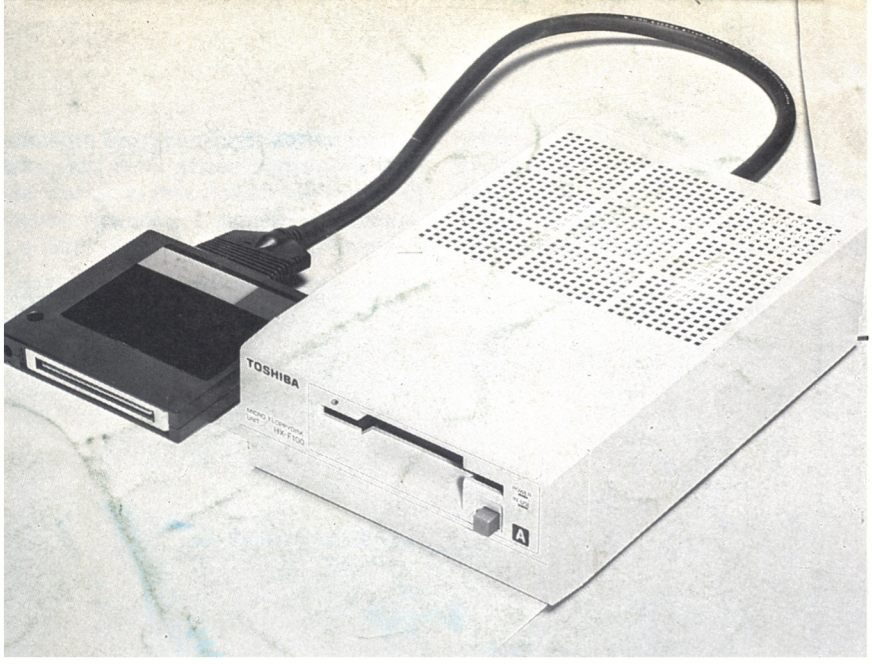
Spero di essere stato chiaro.

Cosimo Pisanello, Felline (LE)

Bisogna riscrivere l'istruzione 20 come segue: 20 OPEN"QD"+N$ FOR OUTPUT AS # 1.

PER IL COPIATORE

Mi ha interessato il programma che



permette la duplicazione dei programmi caricati con BLOAD. Come fare per gli altri casi e poi (troppe domande forse)...

Michele Caruso - Capua

Il copiatore pubblicato sul n. 7 di MSX Computer Magazine funziona correttamente, comunque se non riesci ad usarlo e vuoi copiare un programma salvato con il comando BSAVE devi seguire le seguenti operazioni:
- carica il programma copiatore ed annota le locazioni di memoria di inizio, fine e start del programma che vuoi copiare;
- spegni il computer, riaccendilo e digita i seguenti comandi:

CLEAR 50, inizio

dove inizio è l'indirizzo di memoria dove inizia il programma.
Successivamente digita:

BLOAD "CAS:"

e carica il programma da copiare. Appena appare la scritta OK metti una cassetta vergine nel registratore e predisponi quest'ultimo alla registrazione. Ora puoi salvare il programma digitando:
BSAVE "CAS:nome", inizio, fine, start
dove inizio, fine e start sono i valori trovati con il programma copiatore.
Se vuoi copiare un programma caricabile con RUN "CAS:" è sufficiente caricare il programma con LOAD "CAS:" e salvarlo con SAVE "CAS: nome". Infatti un programma che si carica con RUN "CAS:" è salvato come file ASCII, quindi col comando SAVE e non CSAVE.

SALVARE I DATI

Ho realizzato un programma di Elenco Fornitori ma da mesi sono fermo perché non riesco a salvare i dati sul nastro. Cioè quando ricarico mi dà tutti 0. Ho provato con CHR$ ma...

Gian Battista Nichetti, Soresina

Per poter trasferire dei dati su nastro bisogna aprire un file su nastro e questo viene fatto con la seguente istruzione:
OPEN "CAS:PIPPO" FOR OUTPUT AS # 1
la quale apre un file di nome "PIPPO" e sul quale si può scrivere indirizzando l'istruzione PRINT # 1. Facciamo un esempio; vogliamo scrivere nel file PIPPO la stringa "CASA", il numero 10 e la variabile A%, il cui valore non ci interessa. Per fare questo scriveremo:
PRINT # 1; "PIPPO", 10, A%
A questo punto chiudiamo il canale di scrittura con il comando CLOSE # 1. Per la lettura dei dati scritti su nastro dovremo innanzitutto posizionare il nastro all'inizio della registrazione del file PIPPO e poi ripetere le operazioni precedenti sostituendo l'istruzione di scrittura con quella di lettura:
OPEN "CAS:PIPPO" FOR INPUT AS # 1
INPUT # 1; A$, A, B%
CLOSE # 1
In A$ avremo la stringa "PIPPO", in A il valore 10 e in B% il valore della variabile A% precedentemente salvato.
Per far si che la lettura di un file avvenga correttamente bisogna che la sintassi di scrittura sia uguale a quella di lettura.

Non lasciare solo il tuo computer

r marchetti

Technimedia
00141 Roma, via Valsolda 135 - tel. (06) 898654 - 899526

SPLENDIDO

UN LOOK
COLORATO
PER LA TUA
CASSETTA

RITAGLIA
LUNGO
IL BORDO
SEGNATO
IN NERO
E
PIEGA
SEGUENDO
IL
TRATTEGGIO
INDICATO

PERSONALIZZA
LA
CASSETTA
CON IL
TUO NOME

MSX COMPUTER
MAGAZINE
PER LA TUA
SOFT-TECA

RIVISTA PROGRAMMI PER MSX

MSX Computer Magazine 10

Per caricare digitare CLOAD

SOMMARIO

LATO A:

- THE BALL
- TIRO AL PICCIONE
- SALTO IN ALTO
- LUPEN 3
- STAR WARS

LATO B:

- PLAY+
- DRAGAMINE
- L'EREDITA CRAKSTONE
- ITALIA PUZZLE
- PROCOR

QUESTA CASSETTA È DI

NOME

COGNOME

VIA N.

CITTÀ

MSX COMPUTER MAGAZINE