30 ottobre 1983 - Sped. Abb. Post. gr.III/70 - Suppl. a Computer N. 61 - Distr. Me. Pe. Spa

commodore

COMPUTER CLUB

N. 6

Lire 2000

La rivista degli utenti di sistemi Commodore

Giochi e programmi per il Vic ed il 64

- Alterraggio
- Monitor TIM
- Roulette russa
- Antiaerea
- Forza 4

Un programma completo per il word processing

Assembler per tutti

I nuovi prezzi Commodore

SOMMARIO

Attualità	**Francesco Salvi e le novità Commodore**	**6**
	Vuoi partecipare a Tandem?	**33**
Software Vic	**Atterraggio**	**8**
	Antiaerea	**19**
	Forza 4	**20**
Software 64	**Roulette russa**	**12**
	Simulazione monitor TIM	**10**
Software sistemi	**Forza 4**	**20**
	Un word processing completo	**35**
Recensioni	**Esplora l'universo col Vic ed il 64**	**15**
Didattica	**Impara a programmare col Vic (sesta lezione)**	**25**
	Assembler per tutti	**46**
Mercato	**Tutti i prezzi dell'hardware Commodore**	**51**

Commodore Computer Club - Rivista indipendente per gli utenti di sistemi Commodore
Direttore responsabile: Michele di Pisa
Redazione: Alessandro De Simone,
Direzione, redazione: Piazza Arduino, 3 - 20149 Milano - Tel. (02) 434354-435717
Pubblicità: Milano Paola Bevilacqua, Gianluigi Centurelli, Tina Ronchetti, Villa Claudio Viale Famagosta, 75 - 20142 Milano - tel. (02) 8467348/9/40
Prezzi e abbonamenti: la rivista esce bimestralmente. Prezzo per una copia Lire 2.000. Arretrati il doppio. Abbonamento per dieci fascicoli lire 18.000. Abbonamento annuo cumulativo alle riviste Computer e Commodore Computer Club (tariffa riservata agli studenti) L. 24.000. I versamenti vanno indirizzati a Minisystems Italia s.r.l., mediante assegno bancario, vaglia o utilizzando il c/c postale n. 11909207.
Composizioni: Minisystems Italia
Selezioni: Org. Aldo Ghiacci
Stampa: La Litografica s.r.l. - Busto A.
Registrazione: Tribunale di Milano n. 370 del 2/10/1982 - Sped. in abb. post. gr. III n. 70 quale supplemento alla rivista Computer - Pubbl. inferiore al 70%

Atterraggio

Gira su Vic inespanso e richiede l'uso dei seguenti comandi: B per iniziare la discesa, S per virare a sinistra e D per girare verso destra.

```
100 REM ATTERRAGGIO
110 REM ------------
120 REM    DI VIZIO
130 REM    MAURIZIO
140 REM TEL 039/743092
141 REM ------------
150 POKE36879,8
160 PRINT"
170 IFV=3THEN730
180 FORK=0TO50
190 X%=RND(1)*22
200 Y%=RND(1)*22
210 L=7680+X%+22*Y%
220 POKEL,42:NEXT
230 X=0
240 FORK=XTO21
250 POKE7680+K,160
260 FORC=0TO100:NEXT
270 POKE7680+K,32
280 GETA$
290 IFA$="B"THEN380
300 NEXT
310 FORK=21TOXSTEP-1
320 POKE7680+K,160
330 FORC=0TO100:NEXT
340 POKE7680+K,32
350 GETA$
360 IFA$="B"THEN380
370 NEXT:GOTO240
380 POKE8144,232
390 POKE8145,232
400 POKE8152,232
410 POKE8153,232
420 POKE8160,232
430 POKE8161,232
440 FORY=0TO21
450 L=7680+K+22*Y
460 P=PEEK(L)
470 IFP=42THEN570
480 IFP=232THEN630
490 POKEL,160
```

```
500 FORT=0TO300:NEXT
510 POKEL,32
520 GETA$
530 IFA$=""THEN560
540 IFA$="D"THENK=K+1
550 IFA$="S"THENK=K-1
560 NEXT
570 A$="
580 PRINTA$
590 A$="
600 PRINTA$
610 FORT=0TO2500:NEXT
620 V=V+1:GOTO160
630 FORK=0TO100
640 POKE36879,K
650 NEXT
660 POKE36879,81
670 PRINT"
680 A$="ATTERRAGGIO"
690 PRINTA$
700 PRINT"NUMERO";N+1
710 FORT=0TO3000:NEXT
720 N=N+1:GOTO150
730 PRINT"
740 PRINT"TERMINATO
750 PRINT"
760 FORK=0TO3000:NEXT
770 PRINT"ANCORA";
780 INPUTA$
790 A$=LEFT$(A$,1)
800 IFA$="S"THENRUN
810 POKE36879,27
820 PRINT"
READY.
```

Il sistema solare visibile

Un videogioco istruttivo per Vic 20 e Commodore 64.

E' UNO dei più eccitanti fra i nuovi giochi introdotti dalla Commodore per il 64 ed il Vic 20, ma proprio per questo è anche uno dei più difficili. Richiede tutte le abilità dei videogiochi più complessi, ma al contempo vi obbliga ad imparare e a conoscere il sistema solare come un astronomo o un astronauta.

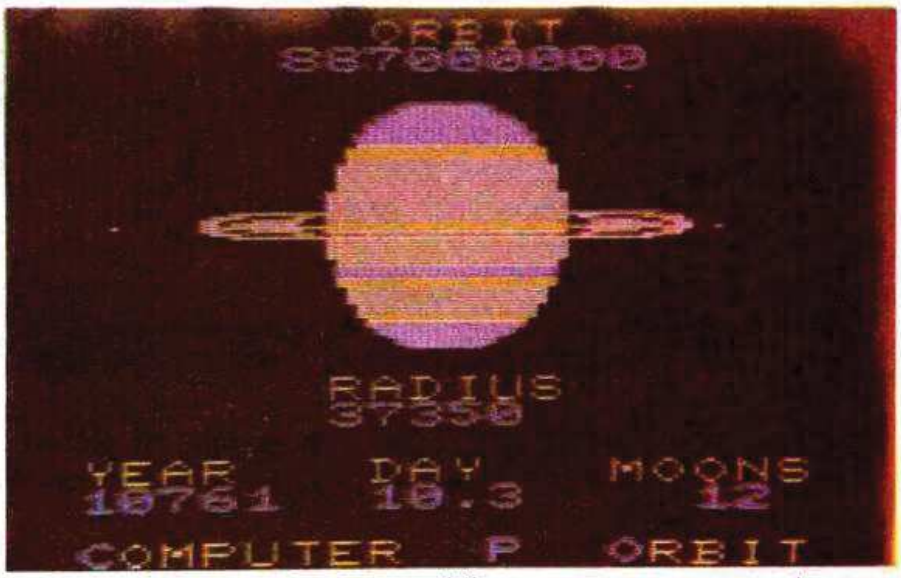
Tu sei il comandante d'una navicella spaziale in viaggio attraverso il sistema solare. L'astronave ha una velocità di crociera d'oltre un miliardo di miglia ed è equipaggiata con un sistema computerizzato, che vi consente sempre nuove ed eccitanti scoperte.

Quando viene inserita la cartuccia, visualizza Mercurio, Venere, la Terra e Marte, che ruotano attorno al sole. Subito dopo appare la vostra astronave con, in primo piano, il pianeta Terra ed una serie di dati circa la sua dimensione ed il suo moto rotatorio. Nel corso del viaggio potrete visitare gli altri pianeti del sistema solare.

Per cominciare a giocare dovete premere F1.

Navigazione

Per navigare disponete d'uno schermo di controllo che consente di controllare e modificare la



posizione della navicella, muovendovi dal centro dello schermo verso le orbite di Mercurio, Venere, Terra, Marte e Giove. I numeri, che compaiono accanto ad ogni pianeta, indicano la distanza dal sole delle rispettive orbite in milioni di miglia. In rosso è segnata la posizione della vostra navicella. Sulla sinistra, in alto, è segnata l'altezza dell'astronave (sempre in milioni di miglia) rispetto all'orbita dei pianeti.

Una telecamera rotante, installata sulla navicella, consente d'osservare le orbite ed i movimenti dei pianeti in una prospettiva tridimensionale. A sinistra, in basso, un apposito numerino blu indica l'angolo verso cui è rivolta la telecamera: O significa che sta guardando perpendicolarmente sotto l'astronave: man mano che raggiunge gli 80 gradi, l'obiettivo si sposta fino a puntare ciò che sta davanti. Per vedere le orbite dei pianeti, occorre premere il tasto V (da view = vista). Per tornare alla strumentazione bisogna battere O.

I comandi per azionare la navicella sono:

CRSR → per orientare la navicella verso destra (l'orientamento è segnato dal punto rosso);

CRSR ↖ per invertire la direzione:

← per virare a sinistra:

↖ per andare avanti:

G ordina alla navicella di spostarsi nella direzione segnata prima:

U muove la navicella verso l'alto;

D muove la navicella verso il basso.

La navicella è in grado di effettuare due itinerari animati preprogrammati. In tal caso, il tasto S consente di esplorare la superficie del sistema a distanza ravvicinata. Quando il giro è finito premete O. Col comando A, la navicella si allontanerà di 500 milioni di miglia dall'orbita di Giove, verso l'orbita della Terra. Quando lo spostamento è finito, occorre battere ancora O. Questi spostamenti lasciano l'astronave a soli 15 milioni di miglia dalle orbite solari, cioè molto più in basso di dove potete arrivare pilotando voi stessi l'astronave. Volendo, potete effettuare le vostre esplorazioni da questa altezza, oppure, potete ritornare a 30 milioni di miglia battendo F3 ed F1.

I pianeti in primo piano

Per vedere da vicino i pianeti Terra, Giove e Saturno, è sufficiente battere P. Lo stesso tasto serve per cambiare pianeta.

Sullo schermo in alto, si vede il diagramma del sole (a sinistra) e dei sette pianeti più vicini. Una piccola stella blu, indica il pianeta che avete zoomato. Il termine ORBIT indica la distanza in miglia dal sole; RADIUS indica il raggio dell'orbita (sempre in miglia). L'anno (YEAR) ed il giorno (DAY), indicano la data terrestre. Solo alcune lune (MOONS) di Giove e Saturno, possono essere viste dall'astronave.

Ogni pianeta ha una propria forza di gravità, e WEIGHT (peso), indica quanto pesereste su quel pianeta, supposto che, sulla terra pesiate 160 libre (ogni libra è pari a 453 grammi). AGE, indica la vostra età in anni di quel pianeta, supponendo che sulla terra ne abbiate 24.

A tale scopo, prima di visualizzare i vari dati, il computer planetario vi chiede di segnalare quale pianeta vi interessa (battendo il tasto P di PLANET), e di fornire i vostri dati personali (battere Y per richiederli) ed in particolare l'età (Age) ed il peso (Weight). Per fornire questi ultimi dati dovete digitare C (da correct, esatto) se la cifra proposta dal computer è giusta, oppure M (more = di più) se va accresciuta, o L (less = meno), se va diminuita.

Il computer planetario

Se vuoi maggiori informazioni sui sei principali pianeti (Mercurio, Venere, Terra, Marte, Giove e Saturno) o vuoi fare dei raffronti tra gli stessi, batti un numero da 1 a 6. I dati visualizzati sono:

- *RADIUS*, ossia il raggio in miglia. I raggi di Giove e Saturno, indicano la distanza tra lo strato più denso di nuvole ed il centro;
- *YEAR*, tempo di roteazione attorno al sole, espresso in giorni terrestri;
- *MOONS*, numero delle lune relative;
- *ORBIT*, velocità di roteazione del pianeta lungo la propria orbita in miglia per secondo;
- *ESCAPE*, velocità (in miglia per secondo) necessaria ad un'astronave per sfuggire alla forza di gravità del pianeta;
- *TO ATM*, temperatura in gradi Farenheit sulla superficie delle nuvole del pianeta;
- *WARM ATM*, temperatura massima in Farenheit;
- *SURFACE*, temperatura sulla superficie del pianeta.

Se invece di numeri compare un asterisco, significa che quella precisa informazione non è ancora disponibile.

Per mettere a confronto i dati di un pianeta già selezionato con quelli di un altro, occorre battere A, finchè questo pianeta non compare sotto la scritta ASTROCALC. ▪

Antiaerea

```
10 REM - ANTIAEREA PER VIC 20  NON ESPANSO -
12 REM BALDI EMANUELE  . TEL 039/749652
13 :
14 REM F1=PER PUNTARE IL CANNONE IN DIAGONALE
18 REM F3=PER PUNTARE IL CANNONE IN AVANTI
22 REM F5=PER PUNTARE IL CANNONE VERSO L' ALTO
1000 REM   SPAZIO PER
1001 REM   SPARARE .
1002 Z=36878:O=36876:B=36877
1010 POKE36879,14:PRINT"□":PRINT"■■■■     ANTIAEREA":GOSUB1420
1020 PRINT"■DIFFICOLTA'(1-100)":INPUTD
1030 IFD<20THENS=20:GOTO1050
1040 S=D
1050 PRINT"HAI";S;"COLPI"
1060 PRINT"■   PREMI UN TASTO"
1070 GETB$:IFB$=""THENGOTO1070
1080 PRINT"□":POKE36869,254:POKE56,24:POKE55,0:POKE650,255:Q=8186:H=35
1090 FORK=0TO23:READA:POKE6144+8*32+K,A:NEXTK
1100 DATA 0,0,0,0,0,0,0,0,24,24,24,24,24,24,0,24,16,12,12,255,64,35,20,8
1110 FORK=0TO23:READA:POKE6144+8*35+K,A:NEXTK
1120 DATA0,60,12,127,64,35,52,104,56,95,127,255,128,89,20,0,62,62,20,255,1,2,2,1
64
1130 FORK=0TO23:READA:POKE6144+8*38+K,A:NEXTK
1140 DATA64,224,112,56,28,8,1,2,0,0,231,24,255,24,36,36,56,110,192,170,213,192,1
10,56
1150 FORK=0TO15:READA:POKE6144+K+8*41,A:NEXTK:DATA4,4,12,127,64,35,52,104
1160 DATA0,253,253,0,0,0,0,0
1170 POKE7890,39
1180 FORK=0TO506:IFPEEK(Q-K-1)<>32THEN1280
1190 POKEQ-K,H:POKEQ+1-K,36:POKEQ+2-K,37
1200 GETA$
1210 FORF=0TOD:NEXTF
1220 IFA$=" "THENGOSUB1290
1230 IFA$="■"THENH=34
1240 IFA$="■"THENH=35
1250 IFA$="■"THENH=41
1260 POKEQ-K,32:POKEQ+1-K,32:POKEQ+2-K,32
1270 IFNOTV>STHENNEXTK
1280 PRINT"□":POKE36869,240:PRINT"HAI DISTRUTTO";P:PRINT"AVVERSARI":END
1290 IFH=34THENGOSUB1390:FORX=1TO23:IFPEEK(Q-K-X*23)=39THEN1370
1300 IFH=34THENPOKEQ-K-X*23,38:POKEQ-K-X*23,32:NEXTX:V=V+1:IFP>3ANDC<70THENPOKEQ
-K-16,40
1310 IFH=34THENRETURN
1320 IFH=41THENGOSUB1390:FORX=1TO22:IFPEEK(Q-K-22*X)=39THEN1380
1330 IFH=41THENPOKEQ-K-X*22,33:POKEQ-K-X*22,32:NEXTX:V=V+1:IFP>3ANDC<70THENPOKEQ
-K-16,40
1340 IFH=41THENRETURN
1350 GOSUB1390:FORX=1TO10:IFPEEK(Q-K-X)=40THENP=P+1:POKEQ-K-X,32:GOSUB1410
1360 POKEQ-K-X,42:FORR=0TO5:NEXTR:POKEQ-K-X,32:NEXTX:V=V+1:RETURN
1370 P=P+1:POKEQ-K-23*X,32:C=RND(1)*110:POKE7680+C,39:GOSUB1410:RETURN
1380 P=P+1:POKEQ-K-22*X,32:C=RND(1)*110:POKE7680+C,39:GOSUB1410:RETURN
1390 POKEZ,15:FORL=1TO30:FORM=250TO240STEP-1:POKEQ,M:NEXTM:FORM=240TO250
1400 POKEQ,M:NEXTM:POKEZ,0:RETURN
1410 POKEQ,0:POKEB,220:FORL=10TO0STEP-1:POKEZ,L:FORM=1TO130:NEXTM,L:POKEB,0:RETU
RN
1420 FORG=0TO3:POKEZ,15:FORL=1TO10:FORM=180TO235STEP2:POKEQ,M:FORN=1TO10:NEXTN,M
:POKEQ,0
1430 FORM=1TO100:NEXTM:POKEZ,0:NEXTG:RETURN
```

Forza 4

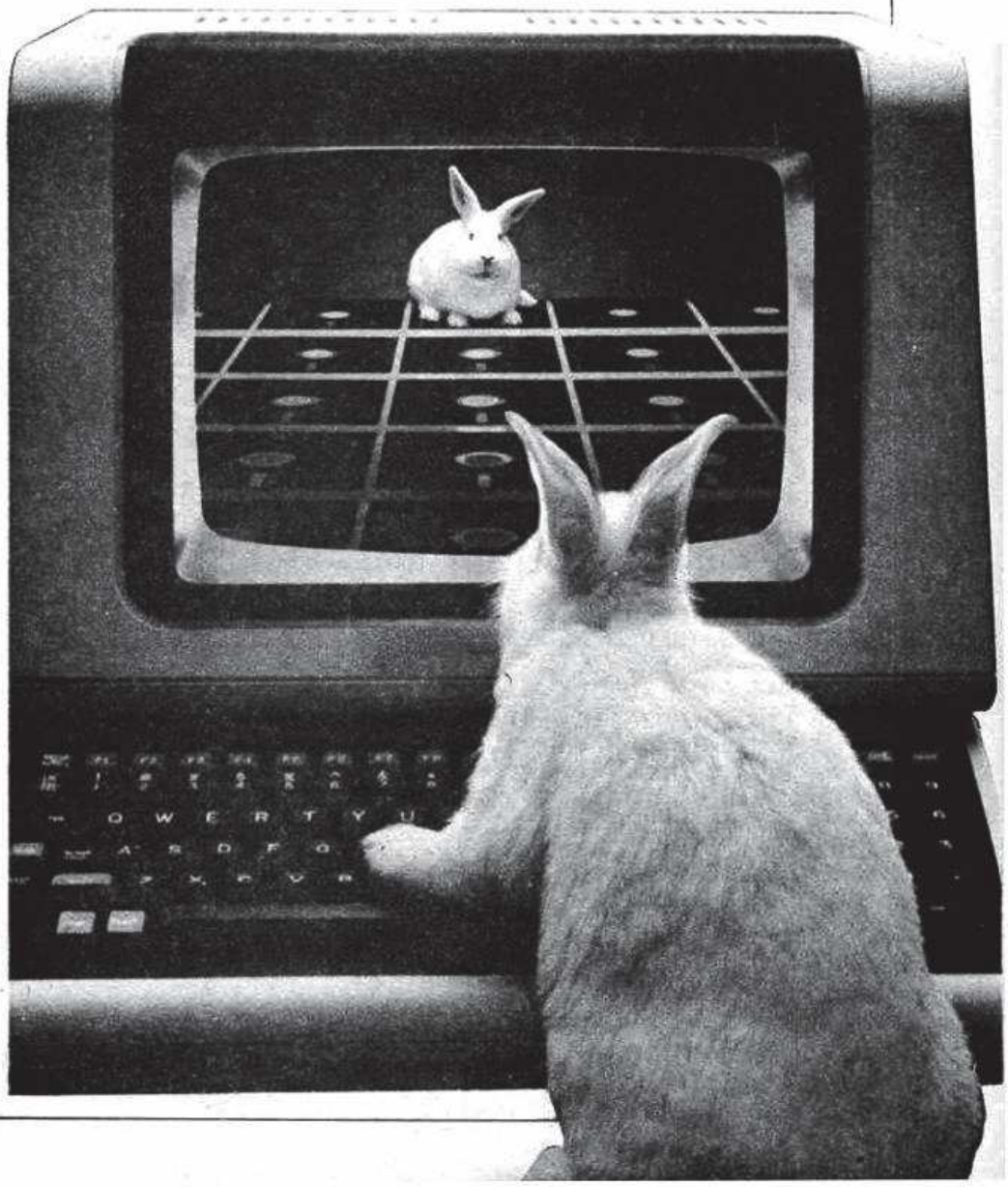
Un programma che è stato fatto per il PET, e di cui viene data anche la versione per il fratello minore VIC 20.

IL GIOCO consiste nel riuscire a mettere 4 pedine dello stesso giocatore di seguito, sia orizzontalmente, verticalmente, che diagonalmente.

Detto così può apparire facile: sembra un'ennesima edizione del tris, solo che invece di 3, sono 4 le pedine da mettere in fila. Ma non è affatto così: infatti, sul video ci sono 12 colonne (10 per ovvi motivi nella versione per il VIC), e ad ognuna di esse corrisponde una lettera dell'alfabeto (dalla A alla L), in cui si lasciano cadere le pedine, una alla volta a turno per i due giocatori.

Queste pedine non si fermano dove vogliono, nè dove vorremmo si fermassero, ma hanno un ordine ben preciso: la prima, che per esempio metteremo nella colonna B, andrà sul fondo; la seconda (sempre in B), andrà immediatamente sopra alla prima, e così via.

Può capitare a volte di non accorgersi di aver vinto - e soprattutto di avere perso - perchè si era troppo attenti a seguire il



proprio piano di gioco, ed è il computer che ce lo fa capire con lampeggi delle 4 pedine in fila e con suoni.

All'inizio bisogna dare il nome dei due giocatori, quindi, inizia subito la partita: verranno disegnate le colonne sul video e la scritta "player 1" apparirà in basso a destra. Bisogna quindi dire al computer in quale colonna vogliamo mettere la nostra pedina - premendo il tasto relativo alla lettera indicante la colonna - ed allora la si vedrà cadere dall'alto della colonna scelta. Poi comparirà la scritta "player 2" e toccherà quindi al secondo giocatore, e così via fino a quando uno dei due non avrà vinto.

Quando una colonna è piena non vi si possono più mettere pedine e, se si tenta , il computer ci dice di rifare. Alla fine del gioco viene data la classifica delle vittorie dei giocatori e viene chiesto se si vuole giocare ancora e, nel caso affermativo, si comincierà una nuova partita.

Come funziona

Dalla linea 150 alla 220, vengono chiesti i nomi dei due giocatori (G1$ e G2$), e vengono formate le stringhe A$ e D$, che servono per i print sul video.

Da 250 a 300 vengono visualizzate le colonne.

Da 310 a 450 viene chiesto il nome della colonna in cui si vuol mettere la pedina, la quale viene fatta cadere (390-450).

Le linee da 470 a 740 servono per controllare se un giocatore ha vinto, cioè se ha messo 4 pedine di seguito.

Il procedimento per controllare questo è abbastanza semplice: supponiamo che il computer stia controllando se le 4 pedine sono in fila orizzontalmente, (verticalmente e diagonalmente, il procedimento è il medesimo).

Partendo dalla pedina che sta immediatamente di fianco a quella appena introdotta, prosegue fino a quando ne trova una dell'avversario o fino a quando non c'è alcuna pedina, e poi torna indietro contando le pedine: se risultano essere 4 (o anche di più, come può capitare), vuol dire che si è vinto.

La linea 990 si è resa necessaria per il caso in cui si voglia mettere la pedina in una colonna già piena.

Versione Vic

Le uniche varianti riguardano i valori dei POKE sia per il video che per il suono. Per il resto la versione per il Vic è essenzialmente identica a quella del PET.

```
10 POKE36879,8 :PRINT"■"
100 REM ************************
110 REM ** (C) BELLU' GIOVANNI **
115 REM **   VERSIONE  VIC 20  **
116 REM **     RIVEDUTA DA     **
117 REM **    CIANI GIUSEPPE   **
120 REM ************************
130 PRINT"□"
140 FORK=1TO20:D$=D$+" ":NEXT
150 PRINT"■NOME PRIMO GIOCATORE■■":INPUTG1$
170 PRINT"■■■NOME SECONDO GIOCATORE■■":INPUTG2$
180 POKE36878,15
190 SS=36876
200 PRINT"□"
210 FORK=1TO15:PRINT"| | | | | | | | | | |":NEXT
220 PRINT"└┴┴┴┴┴┴┴┴┴┴┴┘"
230 A$="■":FORK=1TO17:A$=A$+"■":NEXT
240 B$="●":C$="O":A=7724:B=81:C=87
250 PRINTA$D$
260 PRINT"■□ A B C D E F G H I J"
```

```
270 U=1:PRINTA$"PLAYER 1        ";:GOSUB310
280 D=B:GOSUB330
290 D=C:U=2:PRINTA$"PLAYER 2        ";:GOSUB310:GOSUB330:GOTO270
300 END
310 GETM$:IFM$<"A"ORM$>"J"THEN310
320 PRINTM$;:RETURN
330 REM
340 W=A+1+(ASC(M$)-65)*2
350 FORK=WTO8073STEP22:I=I+1
360 IFPEEK(K)<>32THENPOKEK-22,D:W=K-22:K=8073:GOTO380
370 POKEK,D:FORPP=1TO10:NEXT
380 POKES8,135+I*4:FORO=1TO2*I:NEXT:POKES8,0:POKEK,32:NEXT:I=I+2
390 POKES8,135+I*4:FORJ=1TO20*I:NEXT:POKES8,0:IFI<4THENPRINT"RIFAI":I=0:GOTO1000
395 I=0
400 FORJ=1TO4
410 IFPEEK(W+J*2)<>DTHENA1=J:J=4:GOTO430
420 NEXTI:J=0:GOTO440
430 NEXTJ:J=0
440 FORJ=1TO4
450 IFPEEK(W+(A1)*2-J*2)=DTHENR=R+1:IFR=4THENPRINT"HAI VINTO":P=2:GOTO680
460 NEXT:R=0
470 J=0
480 FORJ=1TO4
490 IFPEEK(W+20*J)<>DTHENA2=J:J=4
500 NEXTJ:J=0
510 FORJ=1TO4
520 IFPEEK(W+A2*20-J*20)=DTHENR=R+1:IFR=4THENPRINT"HAI VINTO":P=20:GOTO680
530 NEXT:R=0
540 J=0
550 FORJ=1TO4
560 IFPEEK(W+22*J)<>DTHENA3=J:J=4
570 NEXTJ:J=0
580 FORJ=1TO4
590 IFPEEK(W+A3*22-J*22)=DTHENR=R+1:IFR=4THENPRINT"HAI VINTO":P=22:GOTO680
600 NEXT:R=0
610 FORJ=1TO4
620 IFPEEK(W+24*J)<>DTHENA4=J:J=4
630 NEXTJ:J=0
640 FORJ=1TO4
650 IFPEEK(W+A4*24-J*24)=DTHENR=R+1:IFR=4THENPRINT"HAI VINTO":P=24:GOTO680
660 NEXT:R=0
670 RETURN
680 IFP=2THENX=A1
690 IFP=20THENX=A2
700 IFP=22THENX=A3
710 IFP=24THENX=A4
720 J=4:NEXT
730 FORZ=1TO30
740 FORJ=1TO4
750 POKEW+X*P-J*P,42
760 POKES8,75:FORH=1TO1:NEXT:POKES8,0:NEXTJ
770 FORK=1TO10:NEXT
```

```
780 FORJ=1TO4
790 POKEW+X*P-J*P,D
800 POKESS,174:FORH=1TO1:NEXT:POKESS,0:NEXTJ
810 NEXTZ
820 IFD=81THENP1=P1+1:GOTO840
830 P2=P2+1
840 PRINT"■"
850 PRINT"■PLAYER 1  :■":PRINTG1$:PRINT"■PUNTI ";P1
860 PRINT"■■■PLAYER 2  :■":PRINTG2$:PRINT"■PUNTI ";P2
870 PRINT"■■■■ANCORA ? (S/N)"
880 GETA$:IFA$=""THEN880
890 IFA$<>"S"ANDA$<>"N"THEN880
900 IFA$="N"THENEND:END:END
910 GOTO180
1000 POKEW,32: FORQ=1TO1000:NEXT:ONUGOTO270,290

READY.
```

```
                VERSIONE PET
        ************************
150 PRINT"■"
160 A$="■":FORK=1TO17:A$=A$+"■":NEXT
170 PRINT"■■■■■■■"TAB(10)"F O R Z A - 4 "
180 PRINT:PRINTTAB(8)"BY BELLU' GIOVANNI"
190 FORK=1TO40:D$=D$+" ":NEXT
200 PRINTA$"NOME ■PRIMO■ GIOCATORE";:INPUTG1$
210 PRINTA$D$
220 PRINTA$"NOME ■SECONDO■ GIOCATORE";:INPUTG2$
230 POKE59467,16:POKE59466,16
240 SS=59464
250 PRINT"■"
260 FORK=1TO15:PRINT"| | | | | | | | | | | | | |":NEXT
270 PRINT"└┴┴┴┴┴┴┴┴┴┴┴┴┴┴┘"
280 B$="●":C$="o":A=32848:B=81:C=87
290 PRINTA$D$
300 PRINT"■□ A B C D E F G H I J K L"
310 U=1:PRINTA$"PLAYER 1      ■■■■";:GOSUB350
320 D=B:GOSUB370
330 D=C:U=2:PRINTA$"PLAYER 2      ■■■■";:GOSUB350:GOSUB370:GOTO310
340 END
350 GETM$:IFM$<"A"ORM$>"L"THEN350
360 PRINTM$:RETURN
370 REM
380 W=A+1+(ASC(M$)-65)*2
390 FORK=WTO33500STEP40:I=I+1
400 IFPEEK(K)<>32THENPOKEK-40,D:W=K-40:K=33500:GOTO420
410 POKEK,D:FORPP=1TO10:NEXT
420 POKESS,15+I*10:FORO=1TO2*I:NEXT:POKESS,0:POKEK,32:NEXT:I=I+2
430 POKESS,15+I*10:FORJ=1TO20*I:NEXT:POKESS,0
440 IFI=4THENPOKEW-40,160
450 IFI<4THENPRINTA$"■RIFAI■            ":I=0:GOTO990
```

```
460 I=0
470 FORJ=1TO4
480 IFPEEK(W+J*2)<>0THENA1=J:J=4:GOTO500
490 NEXTJ:J=0:GOTO510
500 NEXTJ:J=0
510 FORJ=1TO4
520 IFPEEK(W+(A1)*2-J*2)=0THENR=R+1:IFR=4THENPRINT"■■HAI VINTO":P=2:GOTO750
530 NEXT:R=0
540 J=0
550 FORJ=1TO4
560 IFPEEK(W+38*J)<>0THENA2=J:J=4
570 NEXTJ:J=0
580 FORJ=1TO4
590 IFPEEK(W+A2*38-J*38)=0THENR=R+1:IFR=4THENPRINT"■■HAI VINTO":P=38:GOTO750
600 NEXT:R=0
610 J=0
620 FORJ=1TO4
630 IFPEEK(W+40*J)<>0THENA3=J:J=4
640 NEXTJ:J=0
650 FORJ=1TO4
660 IFPEEK(W+A3*40-J*40)=0THENR=R+1:IFR=4THENPRINT"■■HAI VINTO":P=40:GOTO750
670 NEXT:R=0
680  ORJ=1TO4
690 IFPEEK(W+42*J)<>0THENA4=J:J=4
700 NEXTJ:J=0
710 FORJ=1TO4
720 IFPEEK(W+A4*42-J*42)=0THENR=R+1:IFR=4THENPRINT"■■HAI VINTO":P=42:GOTO750
730 NEXT:R=0
740 RETURN
750 IFP=2THENX=A1
760 IFP=38THENX=A2
770 IFP=40THENX=A3
780 IFP=42THENX=A4
790 J=4:NEXT
800 FORZ=1TO30
810 FORJ=1TO4
820 POKEW+X*P-J*P,42
830 POKESS,75:FORH=1TO1:NEXT:POKESS,0:NEXTJ
840 FORK=1TO10:NEXT
850 FORJ=1TO4
860 POKEW+X*P-J*P,D
870 POKESS,174:FORH=1TO1:NEXT:POKESS,0:NEXTJ
880 NEXTZ
890 IFD=81THENP1=P1+1:GOTO910
900 P2=P2+1
910 PRINT"■"
920 PRINT"PLAYER 1  :";G1$;" = PUNTI ";P1
930 PRINT"PLAYER 2  :";G2$;" = PUNTI ";P2
940 PRINT"■■■ANCORA ?"
950 GETQ$:IFQ$=""THEN950
960 IFQ$<>"S"ANDQ$<>"N"THEN950
970 IFQ$="N"THENEND:END:END
980 GOTO230
990 POKEW,160:FORQ=1TO1000:NEXT:ONUGOTO290,310
READY.
```

7.Le altre funzioni (seconda parte)

Funzioni definite dall'utente

Il Computer ha la possibilità di utilizzare funzioni definite dall'utente mediante l'uso dell'istruzione DEF FNx, che assume la forma generale seguente:

numero di linea DEF FNx (variabile) = espressione

Tutte le funzioni definite in questo modo devono avere all'interno del programma un solo nome definito da FNx, dove x e il nome di variabile.

Tutte le funzioni hanno un argomento fittizio dato dalla variabile racchiusa fra parentesi, mentre quello reale, usato quando la funzione viene successivamente menzionata nel programma, sarà differente dal primo nella definizione della funzione. Per esempio, l'espressione precedente, usata per arrotondare un numero, può essere definita come funzione con l'istruzione seguente:

50 DEF FNR(A)=INT(A/F + 0,5)*F

che può venire in seguito utilizzata nello stesso programma per arrotondare un numero, ad esempio alle centinaia e ad una cifra decimale come riportato nella tabella 7.5.

```
140 F=0.1
150 C1=FNR(C)
160 PRINT "B=";B" B1=";B1
165 PRINT "C=";C;" C1=";C1
170 PRINT
180 GOTO 60
190 END
```

Tabella 7.5 Programma di arrotondamento di numeri

Le funzioni definite dall'utente possono contenere a loro volta, nella loro definizione, altre funzioni dello stesso tipo. Per esempio, il programma della tabella 7.6 tabula i seni di un certo numero di angoli (noti) e gli angoli corrispondenti espressi in gradi alla seconda cifra decimale.

```
20 PRINT"   SENO  GRADI"
30 PRINT"   ----  -----"
40 PRINT
50 DEF FNR(A)=INT(A/0.1+0.5)*0.1
60 DEF FND(U)=180/(ATN(1)*4)
70 DEF FNA(S)=(ATN(S/SQR(1-S*S)))*FND(U)
80 INPUT X
90 IF X=0 THEN 160
100 IF X<1 THEN 130
105 IF X>1 THEN PRINT"ERRORE": GOTO 80
110 D=90
120 GOTO 140
130 D=FNR(FNA(X))
140 PRINT TAB(1);X;TAB(9);D
150 GOTO 80
160 END
```

Tabella 7.6 Seni e angoli (in gradi)

Esercizi

Problema 1 - **Raggio di un cerchio circoscritto**

Scrivete un programma che permette di trovare il raggio di una traccia circolare passante attraverso i punti che formano un triangolo. Il raggio (r) del cerchio che inscrive il triangolo è dato dalla formula:

$$r = \frac{a}{2\text{sen}A} = \frac{b}{2\text{sen}B} = \frac{c}{2\text{sen}C}$$

dove a, b, c e A, B, C sono rispettivamente i lati e gli angoli del triangolo.

Il programma è riportato nella tabella A 10, i

risultati, cioè: a = 452m, b = 386m e c = 739 m sono invece indicati nell'Appendice B. (Nota: cos B = $(a^2 + c^2 - b^2)/2ac$.)

Problema 2 - Aree di triangoli

L'area di un triangolo con i lati a, b e c e gli angoli A,B e C, può essere calcolata se sono noti tre lati o due lati e l'angolo compreso usando una delle due formule seguenti:

area del triangolo $=\sqrt{(s(s-a)\ (s-b)\ (s-c))}$

dove 2s = a+b+c

o area = 1/2 absenC o 1/2bcsen A o 1/2acsenB

Scrivete un programma che riporti le aree dei triangoli della Tabella 7.7 usando le intestazioni adatte. Ricordate che i valori sconosciuti devono essere fissati a zero. Fate in modo che tutte le aree vengano calcolate in centimetri quadrati con numeri arrotondati alla prima cifra decimale. Il programma è riportato nella Tabella A11, i risultati nell'appendice B.

Lati del triangolo (cm)			Angolo compreso
a	*b*	*c*	*(in gradi)*
17.2	9.8	14.1	0
0	74	98	125.4
292	0	405	30.5
10.3	15.6	0	69
−1	0	0	0

Tabella 7.7 Dati usati per calcolare le aree dei triangoli

Problema 3 - Volumi dei solidi

Il volume di un solido con l'area della sezione trasversale (A) uniforme e di altezza (H) è dato dalla formula:

V = A x H

Nella tabella 7.8 sono riportate, assieme ai loro codici, le aree delle sezioni trasversali dei solidi più comuni.

Codice	Solido	Area sezione trasversale
1	cubo	base x altezza
2	cilindro	$\pi \times r^2$
3	poliedro esagon.	$1/2\ \sqrt{27} \times r^2$

b = base h = altezza l = lato r = raggio

Tabella 7.8

Scrivete un programma che calcoli i volumi di questi solidi, in modo che tutte le dimensioni vengano espresse in centimetri. Il nome del solido va indicato nell'istruzione DATA in modo che il programma generi assieme al volume il nome del solido cui esso si riferisce. Per porre fine all'esecuzione del programma, potete usare il codice 0. Per avere dei codici di prova adatta, attenetevi a quanto riportato nella tabella 7.9, ricordando comunque che potete aggiungere anche dati di vostra scelta.

Codice	Prima dimensione *(b, l, r)*	Seconda dimens. *(h o zero)*	Altez. *(h)*	N. di cifre dec.
2	4.5	0	1.75	2
3	12.6	0	250	0
1	5.3	7.0	4.2	1

Tabella 7.9 Dati per il problema "Volumi di solidi"

Il programma permette all'utente di arrotondare il risultato ad un certo numero di cifre decimali usando un fattore di scala (F) (vedi il paragrafo dal titolo "Arrotondamento").

VIC 20
"COMPUTER DELL'ANNO"

VIDEOGIOCHI
disponibili varie cassette videogame.

PERIFERICHE
si possono collegare direttamente registratore, dedicato, floppy, stampante, joystick.

TASTIERA
professionale con 66 tasti, 256 combinazioni di colori, 512 segni diversi per creare grafici, disegni, videogiochi.

SOFTWARE
una vasta gamma di programmi per tutte le necessità.

SUONO
3 toni e 9 ottave disponibili per creare le tue musiche ed effetti speciali.

VIC 20 è il computer più diffuso nel mondo, oltre un milione di esemplari venduti. Perchè?
1. VIC 20 è facile da usare: lo colleghi alla TV e premi i tasti. Tutto qui. - 2. Ci puoi giocare: dagli scacchi agli invasori spaziali, e puoi inventarti i tuoi giochi.
3. Lo usi per la scuola, per risolvere problemi e compiti. - 4. Lo impieghi in casa, per la contabilità entrate-uscite, ma anche per le ricette e per il bioritmo...
5. Costa solo 423.000 lire più iva. - 6. Puoi imparare a programmare in Basic, la lingua del futuro.
7. È stato scelto come "computer dell'anno" dalle migliori riviste internazionali di informatica.
8. È della Commodore, la N° 1 in Europa e in Italia nel campo dei Personal Computer.
Perchè non farti un regalo intelligente?

commodore COMPUTER

Assembler per tutti

Tra i regali del prossimo fascicolo un assembler per tutti i Commodore

Un facile approccio al linguaggio "naturale" del vostro computer.

di **Alessandro de Simone**

IL sistema di numerazione usato nei microprocessori è quello binario puro e pertanto difficilissimo da rappresentare, ma con un piccolo artificio si può semplificare tutto notevolmente ricorrendo al sistema di numerazione esadecimale consistente di sedici simboli:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Per esempio, il numero 12 in base dieci è 1100 in binario puro e semplicemente C in esadecimale (fig. 1).

Poichè (fig. 2) la memoria del computer è una successione di gruppi di 8 bit ciascuno, detti parole o bytes, noi dovremmo scrivere un programma in binario puro, proprio come indicato in figura 2; figurarsi la confusione che ne verrebbe fuori per un programma di un centinaio di parole.

Se però dividiamo idealmente con una linea tratteggiata la 'striscia' otteniamo due gruppi di quattro bit ciascuno per ogni parola che, tradotti in codice esadecimale, sono molto più facili da scrivere in memoria e da controllare in seguito. Le prime tre parole in figura 2 anzichè 10101001, 11111111, 10001101 si scrivono A9 FF 8D ecc. Naturalmente c'è bisogno di un programma che traduca da codice esadecimale, scritto mediante la tastiera del PET utilizzando cifre e lettere, in binario puro, dato che il computer ragiona solamente con questo sistema.

Tale programma, piuttosto sofisticato, è residente nei PET dotati di nuove ROM e richiamabile battendo l'istruzione SYS(4).

Diamo comunque per scontato che il lettore sappia usare tale programma.

L'esempio considerato in figura 2 è un semplicissimo programma in L.M. (linguaggio macchina). Prima di commentarlo definiamo l'accumulatore: questo è un registro di otto bit interno alla CPU (6502) attraverso il quale viene elaborata una grande quantità di informazioni presenti in un programma. Infatti quando facciamo partire il programma

Numero decim.	Numero esadec.	Binario puro
0	0	0 0 0 0
1	1	0 0 0 1
2	2	0 0 1 0
3	3	0 0 1 1
4	4	0 1 0 0
5	5	0 1 0 1
6	6	0 1 1 0
7	7	0 1 1 1
8	8	1 0 0 0
9	9	1 0 0 1
10	A	1 0 1 0
11	B	1 0 1 1
12	C	1 1 0 0
13	D	1 1 0 1
14	E	1 1 1 0
15	F	1 1 1 1

Figura 1

	Indir. Base 16		Dato contenuto nell'indirizzo	
Indir. base 10	Zona alta	Zona bassa	binario	esadec.
			1010 1001	A9
826	03	3A	1111 1111	FF
827	03	3B	1000 1101	8D
828	03	3C	0000 0000	00
829	03	3D	1000 0000	80
830	03	3E	0110 0000	60
831	03	3F	....	....
832	03	40	....	....
833	03	41	....	....
.....	...	....	....	....
.....	...	...		

Figura 2

A9 FF 8D 00 80 60 la CPU carica il dato A9, e poichè le abbiamo detto che la locazione 033A contiene il primo dato da elaborare, capisce che A9 è un'istruzione ed esattamente, leggendo A9, carica nell'accumulatore la parola contenuta nella successiva locazione (033B), cioè FF.

Tale concetto è di fondamentale importanza: la CPU non può sapere se una certa locazione di memoria contiene un dato oppure un'istruzione. Sa soltanto (perchè è microprogrammata in tale modo) che la parola di partenza del programma deve considerarla come un'istruzione e continua l'elaborazione con questo presupposto.

Se infatti facciamo girare il programma da 033B anzichè da 033A, la CPU "pensa" che FF sia l'istruzione da eseguire e va subito in 'tilt' perchè non esiste questa istruzione fra le 151 istruzioni del 6502.

L'istruzione A9 è un'istruzione a due byte, cioè quando la CPU legge A9, sa che la locazione di memoria successiva contiene un dato e non un'istruzione, caricato, allora, il dato FF nell'accumulatore, leggerà il contenuto di 033C, che è SD, istruzione, questa volta, a tre byte.

Un'istruzione a due byte viene conteggiata nel modo seguente: comando + dato; così similmente 8D, come tutte le istruzioni a tre byte, è costituita da: comando + dato + dato.

Incontrando 8D, la 6502 eseguirà il comando trascrivendo il dato presente nell'accumulatore, nella locazione di memoria il cui indirizzo è rappresentato, in esadecimale, dai dati contenuti nelle due locazioni di memoria successive a quelle contenente il comando 8D.

In parole un po' semplici la CPU scrive FF nella locazione, il cui indirizzo è 8000 esadecimale. Attenzione: il primo byte successivo a 8D contiene la parte bassa dell'indirizzo (LA= low address) mentre la terza parola contiene la parte alta (HA= high address).

Dato che vogliamo scrivere il dato FF, che è in accumulatore, nella locazione 8000 scriviamo 8D 00 80 e non 8D 80 00. Tale modo di fare è una caratteristica di tutte le istruzioni a tre byte che incontreremo prima di scrivere la LA poi la HA.

Dopo aver eseguito l'istruzione 8D, la CPU incontra l'istruzione 60 (istr. a un solo byte) equivalente al RETURN. Poichè non è stata chiamata alcuna subroutine si ritorna automaticamente al BASIC uscendo in tal modo dal L.M. Su tale concetto torneremo più avanti. Per ora basti sapere che se in un programma in L.M. c'è un RETURN (60)... in più, si ritorna al BASIC.

Vediamo ora a che cosa serve il programma appena esaminato. La memoria di schermo è rappresentata da 1000 locazioni di memoria (25 righe per 40 colonne) a partire dall'indirizzo 80 00. Scrivendo allora FF in 80 00, apparirà il simbolo "■" nella prima locazione della memoria di schermo (in alto a sinistra). Modificando opportunamente il contenuto di 0800 e facendo rigirare il programma in L.M., faremo apparire nell'angolo in alto a sinistra tutti i simboli grafici del PET. Si potrebbe obiettare che il medesimo risultato si poteva ottenere semplicemente mediante il BASIC.

```
10 PRINT "■"
```

Ciò è vero, però c'è da fare una prima considerazione: per far apparire in L.M. il simbolo "■", abbiamo utilizzato solamente 6 byte; con il BASIC invece occuperemmo ben 10 byte, senza contare che si è dovuto utilizzare l'interprete del BASIC impiegando molto tempo. La sostanziale differenza fra i due metodi si nota quando si vorrà riempire tutto lo schermo con il carattere desiderato. Con il BASIC scriveremmo, ad esempio:

```
10 A=1
20 B=1000
30 PRINT "[clr]"   REM [PANCELLAZIONE
40 FOR I=A TO B          SCHERMO
50 PRINT "■"
60 NEXT
70 END
```

Eseguendo il programma si può nettamente seguire il formarsi del 1000 simboli.

Proviamo ora a caricare il programma in L.M. che segue:

Ind. part.	Dati
033A	A9 80 8D 48 03 A9 00 8D
0342	47 03 A9 FF 8D 00 80 EE
034A	47 03 D0 F6 AD 48 03 C9
0352	83 D0 01 60 EE 48 03 4C
035A	44 03 .. ..

e... SYS(826)... è sorprendente, vero? Ma vediamo come gira:

```
100 I = 826
110 READ A$: IF A$="-1" THEN END
120 A1$ = LEFT$(A$,1)
130 A2$ = RIGHT$(A$,1)
140 IF A1$<"A" THEN 160
150 A1$ = STR$(ASC(A1$)-55)
160 IF A2$<"A" THEN 180
170 A2$ = STR$(ASC(A2$)-55)
180 A = VAL(A1$) * 16 + VAL(A2$)
200 POKE I,A: I = I+1
220 GOTO 110
230 DATA A9,80,8D,48,03,A9,00,8D
240 DATA 47,03,A9,FF,8D,00,80,EE
250 DATA 47,03,D0,F6,AD,48,03,C9
260 DATA 83,D0,01,60,EE,48,03,4C
270 DATA 44,03,-1
```

REMarks

033A A9 80 Nell'accumulatore viene scritto il numero 80 (in effetti viene scritto il numero binario puro 1000 0000 e non ripeteremo più questo concetto).

033C 8D 48 03 Il valore ora presente nell'accumulatore (80) viene riportato così come è nella locazione 03 48. Attenzione che nell'accumulatore è ancora presente 80.

033F A9 00 Nell'accumulatore viene ora scritto 00 cancellando il precedente 80; naturalmente ciò che è stato scritto in 03 48 non viene modificato.

0341 8D 47 03 Vedi istruzione 033C.

0344 A9 FF Vedi istruzione 033A oppure 033F.

0346 8D xx xx I due gruppi di xx stanno a significare che possiamo scrivere qualsiasi valore esadecimale: infatti quando in seguito faremo partire il programma, le istruzioni 033C e 0341 provvederanno a scrivere la parte alta e bassa dell'indirizzo; a questo punto dell'esecuzione del programma, appare in alto a sinistra il simbolo "■", che vogliamo trascrivere anche nella seconda posizione, terza, ecc., fino a riempire tutto lo schermo. Dobbiamo pertanto fare in modo di impartire l'istruzione 8D 01 80 e poi 8D 02 80... fino a 8D FF 83. Per evitare di scrivere mille istruzioni (quante sono le celle dello schermo) utilizziamo un loop (fig. 3).

INDIR. ESA.	OP. COD.	MNEMONICO
033A	A9	LDA# 80
033B	80	
033C	8D	STA 0348
033D	48	
033E	03	
033F	A9	LDA#00
0340	00	
0341	8D	STA 0347
0342	47	
0343	03	
0344	A9	LDA# FF
0345	FF	
0346	8D	STA 8000
0347	00	
0348	80	
0349	EE	INC 0347
034A	47	
034B	03	
034C	D0	BNE
034D	F6	
034E	AD	LDA 0348
034F	48	
0350	03	
0351	C9	CMP# 83
0352	83	
0353	D0	BNE
0354	01	
0355	60	RTS
0356	EE	INC 0348
0357	48	
0358	03	
0359	4C	JMP 0344
035A	44	
035B	03	

Fig. 3

0349 EE 47 03 Questa istruzione (EE) aumenta di una unità il contenuto della locazione, il cui indirizzo è 0347: da 00 passa a 01, oppure quando sarà contenuto A8, passerà ad A9 oppure (importante) da FF a 00.

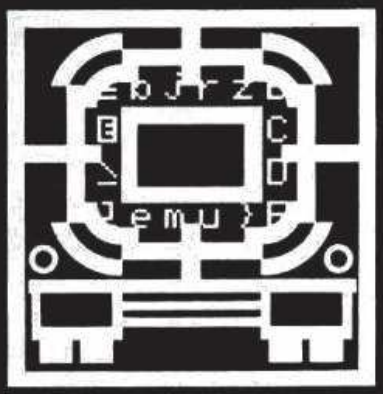
034C D0 F6 L'istruzione D0 è un'istruzione a due byte che fa parte delle così dette istruzioni di BRANCH (salto condizionato). Arrivati a questo punto, il proseguimento dell'elaborazione dipende dal risultato dell'ultima operazione effettuata. Nel caso specifico, se EE 47 03 fornisce un valore uguale a 00, l'istruzione D0 viene ignorata ed il programma prosegue con l'istruzione successiva (034E). In caso contrario la CPU legge il primo bit del secondo byte dell'istruzione (fig. 4:

0344	A9		
0345	FF		F6
0346	8D		F7
0347	00		F8
0348	80		F9
0349	EE		FA
034A	47		FB
034B	03		FC
034C	D0		FD
034D	F6	1111 0110	FE
034E	AD		FF
034F	.. ..	....	....
.....	.. ..	....	....
0353	D0		
0354	01	0000 0001	
0355	60		01
0356	EE		02
0357	.. ..	....	....
.....	.. ..	....	....

Fig. 4

LAVEZZINI
TRASPORTI S.p.A.

SEZIONE TRASPORTI SPECIALI
MOVIMENTAZIONE MACCHINE ELETTRONICHE



PERSONALE SPECIALIZZATO

ATTREZZATURE SPECIALI
AUTOMEZZI CON GRU
AUTOMEZZI CON SPONDA IDRAULICA
STAIR-ROBOTS
PIATTAFORME AEREE
PARANCHI
CARRELLI SPECIALI

MAGAZZINI ANTIPOLVERE

MOVIMENTAZIONE INTERNA CENTRI OPERATIVI

ALLESTIMENTO STAND PER FIERE E MANIFESTAZIONI

Direzione commerciale:
Milano Via Pecori Giardi 5 - Tel. 5397541
Telex 312631

1111 0110)e, se esso è 0,eseguirà un salto in avanti, mentre se è 1 (nostro caso) farà un salto indietro. Per salto si intende il numero di parole (e non istruzioni) che bisogna escludere per rintracciare la successiva istruzione da eseguire. Nel nostro caso, la CPU esegue un conteggio alla rovescia partendo dall'istruzione successiva a D0 (034E), definendola come FF. In seguito va all'indietro fino a che arriva al numero contenuto nel secondo byte di D0, cioè F6, corrispondente all'ind. 0345.

Il programma, allora, continuerà ad essere eseguito dalla parola successiva (sempre all'indietro) a 0345, e cioè 0344 (A9). Bisogna pertanto prestare la massima attenzione al secondo byte delle istruzioni di BRANCH perchè, in caso di errore, potremmo far continuare il programma da un dato anzichè da un'istruzione oppure da un'istruzione indesiderata. E' ovvio che nelle prossime puntate amplieremo questi concetti. Notiamo facilmente che il blocco di istruzioni 0344-034D sarà eseguito 255 volte fino a che cioè 0347 non conterrà 00, consentendo alla CPU di ignorare D0 F6. Ogni volta che il blocco viene eseguito, lo schermo viene riempito di 256 simboli di "■■". Poichè lo schermo è di 1000 caratteri, il blocco deve essere eseguito quasi quattro volte. Vediamo come:

034E AD 48 03 carica in accumulatore il dato contenuto in 0348.

0351 C9 83 Questa istruzione a due byte compara, mediante sottrazione, l'accumulatore con il numero esa. 83: se i due numeri risultano uguali fornirà 00 come risultato (attenzione: l'accumulatore non viene comunque 'modificato' nè tantomeno il numero esa. 83).

0353 D0 01 In questo caso, a differenza di 034C, il primo bit del secondo byte è 0 (fig. 4) e pertanto nel caso in cui il risultato della comparazione 0351 sarà uguale a 00, sarà saltata una parola ed il programma continuerà da 0356.

0355 60 Return subroutine: si esce dal L.M. e si ritorna al BASIC.

0356 EE 48 03 A questa istruzione (vedi 0349) si giunge se il risultato della comparazione 0351 indica che la HA di 0346 (cioè 8D 47 80) non è ancora passato al valore 83, cioè se siamo ancora nella memoria di schermo.

Riepilogando, il blocco 0344-034D viene eseguito 255 volte mentre 0344-0359 viene eseguito 4 volte prima di ritornare al BASIC.

0359 4C 44 03 Istruzione di salto incondizionato: la continuazione del programma passa all'istruzione contenuta in 0344.

```
100 SYS (826)
110 GET A$ IF A$="" THEN 110
120 A= ASC(A$) POKE 837,A
130 GOTO 100
```

Il lettore potrà provare la validità di questa routine in linguaggio macchina, in precedenza caricata, con il programmino qui sopra suggerito. ■