Microprocessori a confronto

Il programma di questo mese serve ad introdurre il discorso sulle nuove possibilità del microprocessore dell'Apple IIc e del IIe (enhanced); si tratta infatti di un miniassembler leggermente potenziato e scritto in Basic. Il potenziamento consiste, oltre che nella gestione dei nuovi codici operativi del 65C02, nella possibilità di inserire e cancellare istruzioni e nei comandi per il salvataggio del programma oggetto su disco. Il fatto poi che sia stato scritto in Basic consente una facile personalizzazione oppure l'estensione delle capacità senza eccessivi sforzi; d'altra parte la lentezza del Basic si nota un pochino nell'attesa che trascorre tra l'inserimento della riga e il conseguente assemblaggio in memoria.

Già che siamo in tema di istruzioni particolari perché non pensare anche ai possessori di un «semplice» Apple II+ o IIe? Ecco allora un elenco (e un metodo di ricerca) delle istruzioni «segrete» del 6502!

Assemblatore in Basic

di Giampiero Torello - Biella (VC)

Questo programma permette di scrivere programmi in mnemonico ed è rivolto principalmente a chi conosce l'assembly.

L'ho scritto perché trovavo piuttosto scomodo dover caricare l'integer Basic e compiere quella serie di passaggi che permettono di giungere all'assemblatore residente in memoria; inoltre quest'ultimo non permette di correggere agevolmente il programma scritto.

La parte del programma che si occupa della traduzione in codice oggetto delle istruzioni Assembler è basata su un semplice sistema a matrice; i codici operativi di ogni istruzione sono contenuti nei DATA. L'istruzione inserita, in mnemonico, viene analizzata e, a seconda che sia di tipo implicito, assoluto, immediato, ecc., le viene associato un numero, che va da 1 a 13 (quanti sono i tipi di indirizzamento del 6502), che corrisponde alla colonna della matrice. L'istruzione vera e propria (es: LDA) viene confrontata con quelle contenute nelle prime due linee di DATA; se risulta appartenere al set di istruzioni del 65C02 le viene associato un numero, che va da 1 a 64 (quante sono le istruzioni del processore), che corrisponde alla riga della matrice.

Per il posizionamento sulla riga sono dovuto ricorrere ad una routine in Assembler, pubblicata su un numero precedente di MC, che svolge la funzione dell'istruzione RESTORE N, dove N è un numero di linea contenente un DATA. In un vettore viene inserito il solo codice oggetto dell'istruzione Assembler, mentre in un altro viene inserito sia il codice operativo che l'istruzione Assembler; quest'ultimo vettore serve per l'edit del programma, che permette l'inserimento o l'eliminazione delle istruzioni.

```
10  FOR I = 770 TO 826
20  READ J: POKE I,J: NEXT
30  DATA 32,190,222,32,103,221,32,82,231,32,26,214,
      176,3,76,124,217,166,156,165,155,208,1,202,233
      ,1,133,125,134,126,96
```

Programma in Basic per la creazione della routine di Restore numero di riga. Una volta fatto girare si deve salvare il codice oggetto con: BSAVE RESTORE, A$300,L$1F.

Uso del programma

All'apparizione del menu pricipale si può scegliere tra sei opzioni: premendo "0" si esce dal programma e si ritorna al Basic. Se si sceglie "INSERIMENTO PROGRAMMA", comparirà la domanda: "INDIRIZZO DI PARTENZA?"; si dovrà inserire l'indirizzo di memoria dal quale si vuole che inizi il programma in Assembler; lo si può inserire sia in decimale che in esadecimale, facendolo precedere da "$". Apparirà poi un cursore lampeggiante in basso a sinistra, ad indicare che potrà iniziare l'inserimento del programma; le istruzioni di questo andranno inserite come segue: es. LDA #$0F (gli operandi devono essere scritti in esadecimale); per motivi tecnici bisogna inserire un punto al posto della virgola negli indirizzamenti indicizzati (es: LDX $23.Y). Se si sbaglia a scrivere si può correggere con le due frecce cursore (sinistra e destra), oppure eliminare quello che si è scritto con il tasto DEL. Una volta premuto il CR, se l'istruzione è corretta, si vedrà comparire il codice oggetto

```
0300-  20 BE DE   JSR   $DEBE
0303-  20 67 DD   JSR   $DD67
0306-  20 52 E7   JSR   $E752
0309-  20 1A D6   JSR   $D61A
030C-  B0 03      BCS   $0311
030E-  4C 7C D9   JMP   $D97C
0311-  A6 9C      LDX   $9C
0313-  A5 9B      LDA   $9B
0315-  D0 01      BNE   $0318
0317-  CA         DEX
0318-  E9 01      SBC   #$01
031A-  85 7D      STA   $7D
031C-  86 7E      STX   $7E
031E-  60         RTS
```

Disassemblato della routine di Restore XXX, permette di posizionare il puntatore dei DATA all'inizio di qualsiasi riga. Se si sostituiscono gli indirizzi $7D e $7E con i valori $B8 e $B9 si trasforma in una GOTO calcolata. Si utilizza con CALL 768,xxx dove xxx può essere un numero o una formula qualsiasi. Salvare con BSAVE RESTORE, A$300,L$1F

```
100  REM       Assemblatore
110  REM            65 C 02
120  IF  PEEK (770) = 222 THEN 127
125 D$ =  CHR$ (4): PRINT D$"BLOAD RESTORE"
127  DIM V$(500),V1$(500),I0$(64)
130  DATA ADC,AND,ASL,BCC,BCS,BEQ,BIT,BMI,BNE,BPL,
     BRA,BRK,BVC,BVS,CLC,CLD,CLI,CLV,CMP,CPX,CPY,D
     EC,DEX,DEY,EOR
140  DATA INC,INX,INY,JMP,JSR,LDA,LDX,LDY,LSR,NOP,
     ORA,PHA,PHP,PHX,PHY,PLA,PLP,PLX,PLY,ROL,ROR,R
     TI,RTS,SBC,SEC,SED,SEI,STA,STX,STY,STZ,TAX,TA
     Y,TRB,TSB,TSX,TXA,TXS,TYA
151  DATA  ,,109,101,105,125,121,97,113,117,,,114,
152  DATA  ,,45,37,41,61,57,33,49,53,,50,
153  DATA  ,10,14,6,,30,,,,22,,,
154  DATA  ,,,,,,,,,,144,,
155  DATA  ,,,,,,,,,,176,,
156  DATA  ,,,,,,,,,240,,
157  DATA  ,,,44,36,137,60,,,,52,,,
158  DATA  ,,,,,,,,,48,,
159  DATA  ,,,,,,,,,208,,
160  DATA  ,,,,,,,,,16,,
161  DATA  ,,,,,,,,,128,,
162  DATA  0,,,,,,,,,,,
163  DATA  ,,,,,,,,,80,,
164  DATA  ,,,,,,,,,112,,
165  DATA  24,,,,,,,,,,,
166  DATA  216,,,,,,,,,,,
167  DATA  88,,,,,,,,,,,
168  DATA  184,,,,,,,,,,,
169  DATA  ,,205,197,201,221,217,193,209,213,,210,

170  DATA  ,,236,228,224,,,,,,,
171  DATA  ,,204,196,192,,,,,,,
172  DATA  ,58,206,198,,222,,,,214,,,
173  DATA  202,,,,,,,,,,,,
174  DATA  136,,,,,,,,,,,,
175  DATA  ,,77,69,73,93,89,65,81,85,,82,
176  DATA  ,,26,238,230,,254,,,,246,,,
177  DATA  232,,,,,,,,,,,,
178  DATA  200,,,,,,,,,,,,
179  DATA  ,,76,,,,124,,,,108,
180  DATA  ,,,32,,,,,,,,,
181  DATA  ,,173,165,169,189,185,161,177,181,,178,

182  DATA  ,,174,166,162,,190,,,,,,182
183  DATA  ,,172,164,160,188,,,180,,,
184  DATA  ,74,78,70,,94,,,,86,,,
185  DATA  234,,,,,,,,,,,,
186  DATA  ,,13,5,9,29,25,1,17,21
187  DATA  72,,,,,,,,,,,,
188  DATA  8,,,,,,,,,,,,
189  DATA  218,,,,,,,,,,,,
190  DATA  90,,,,,,,,,,,,
191  DATA  104,,,,,,,,,,,,
192  DATA  40,,,,,,,,,,,,
193  DATA  250,,,,,,,,,,,,
194  DATA  122,,,,,,,,,,,,
195  DATA  ,42,46,38,,62,,,,54,,,
196  DATA  ,106,110,102,,,126,,,,118,,,
197  DATA  64,,,,,,,,,,,,
198  DATA  96,,,,,,,,,,,,
199  DATA  ,,237,229,233,253,249,225,241,245,,,199,

200  DATA  56,,,,,,,,,,,,
201  DATA  248,,,,,,,,,,,,
202  DATA  120,,,,,,,,,,,,
203  DATA  ,,141,133,,157,153,129,145,149,,,146,
204  DATA  ,,142,,,,,,,,150
205  DATA  ,,140,132,,,,,148,,,
206  DATA  ,,156,100,,158,,,,116
207  DATA  170,,,,,,,,,,,,
208  DATA  168,,,,,,,,,,,,
209  DATA  ,,28,20,,,,,,,,,
210  DATA  ,,12,4,,,,,,,,,
211  DATA  186,,,,,,,,,,,,
212  DATA  138,,,,,,,,,,,,
213  DATA  154,,,,,,,,,,,,
214  DATA  152,,,,,,,,,,,,
300  FOR ZZ = 1 TO 64: READ I0$(ZZ): NEXT
500  HOME : VTAB 3: HTAB 5
510  PRINT "6 Fine lavoro": PRINT : HTAB 5
520  PRINT "1 Inserimento programma": PRINT : HTAB
     5
530  PRINT "2 Correzione programma": PRINT : HTAB
     5
540  PRINT "3 Caricamento da disco": PRINT : HTAB
     5
550  PRINT "4 Memorizzazione su disco": PRINT : HTAB
     5
560  PRINT "5 Visualizzazione": PRINT : HTAB 5
570  PRINT "6 Operazioni dos": PRINT : HTAB 5
580  PRINT ">";
590  GET A$
595  IF A$ = "0" THEN  HOME : END
600  PRINT :A = VAL (A$)
610  IF A = 0 THEN 500
620  ON A GOTO 640,1570,2220,2430,2570,2700
630  GOTO 500
640  HOME
650  INPUT "INDIRIZZO DI PARTENZA ? ";S1$
660  IF S1$ =  CHR$ (27) THEN 500
670  IF  LEFT$ (S1$,1) = "$" THEN D$ =  RIGHT$ (S1
     $, LEN (S1$) - 1):D8 = 16:N8 = 10:FL = 1: GOSUB
     2920:S1 =  VAL (N$): GOTO 690
680 S1 =  VAL (S1$)
690 ST = S1 - 1:K = 0:FE = 0
700  IF ST < 0 THEN 640
710  HOME : VTAB 24: PRINT S1$;: GOSUB 2910
720  GET B$
722 V =  PEEK (37):H =  POS (0)
724  IF B$ =  CHR$ (21) THEN A =  SCRN( H,2 + V) +
     16 *  SCRN( H,2 + V + 1):B$ =  CHR$ (A - 128)

730  IF B$ =  CHR$ (8) AND ( POS (0) = 0 OR  LEN (
     A$) <  = 1) THEN  CALL  - 198: GOTO 720
740  IF B$ =  CHR$ (8) THEN A$ =  LEFT$ (A$, LEN (
     A$) - 1): HTAB  POS (0): GOTO 720
750  IF B$ =  CHR$ (13) THEN  CALL  - 868: GOTO 86
     0
760  IF B$ =  CHR$ (127) THEN  HTAB 1: CALL  - 958
     : GOSUB 2910: GOTO 720
770 I1 = I1 + 1: IF I1 > 3 THEN 820
780  IF B$ =  CHR$ (27) THEN EN = ST: GOTO 500
790  IF B$ < "A" OR B$ > "Z" THEN  CALL  - 198:I1 =
     I1 - 1: GOTO 720
820  IF I1 = 4 THEN  IF B$ <  > " " THEN B$ = " " +
     B$
830 A$ = A$ + B$
840  PRINT B$;
850  GOTO 720
860 IS$ =  LEFT$ (A$,3):A2$ = "":O1$ = "":F4 = 0
870  IF  LEN (A$) = 3 THEN 950
880  FOR ZZ = 5 TO  LEN (A$)
890 A1$ =  MID$ (A$,ZZ,1)
900  IF A1$ = "$" OR A1$ = "@" OR A1$ = "(" OR A1$
     = ")" OR A1$ = "," OR A1$ = "X" OR A1$ = "Y"
      THEN A2$ = A2$ + A1$: GOTO 920
910 O1$ = O1$ + A1$: IF  LEN (O1$) > 4 THEN 1090
920  NEXT ZZ: IF  LEN (O1$) <  = 2 THEN O$ = O1$:F
     L = 1: GOTO 950
930 FL = 2:O$ =  RIGHT$ (O1$,2)
940  IF  LEN (O1$) = 3 THEN O1$ = "0" + O1$:A$ =  LEFT$
     (A$, LEN (A$) - 3) + O1$
950 P1$ = O$: IF  LEN (A$) = 3 AND (IS$ = "ASL" OR
     IS$ = "LSR" OR IS$ = "ROL" OR IS$ = "ROR" OR
     IS$ = "INC" OR IS$ = "DEC") THEN NOD = 2:O1$ =
     "": GOTO 1100
960  IF  LEFT$ (IS$,1) = "B" AND IS$ <  > "BRK" AND
     IS$ <  > "BIT" THEN NOD = 1:FA = 0: GOTO 313
     0
970  IF  LEN (A$) = 3 THEN NOD = 1:O1$ = "": GOTO
     1100
980  IF A2$ = "$" AND  LEN (A$) > 7 THEN NOD = 3: GOTO
     1100
990  IF A2$ = "$" AND  LEN (A$) = 7 THEN NOD = 4: GOTO
     1100
1000  IF A2$ = "#$" THEN NOD = 5: GOTO 1100
1010  IF A2$ = "$,X" AND  LEN (A$) > 9 THEN NOD =
     6: GOTO 1100
1020  IF A2$ = "$,Y" AND  LEN (A$) > 9 THEN NOD =
     7: GOTO 1100
1030  IF  LEFT$ (A2$,1) = "(" AND  LEN (O1$) > 2 AND
     IS$ <  > "JMP" THEN 1090
1040  IF A2$ = "($,X)" THEN NOD = 8: GOTO 1100
1050  IF A2$ = "($),Y" THEN NOD = 9: GOTO 1100
1060  IF A2$ = "$,X" AND  LEN (A$) = 9 THEN NOD =
     10: GOTO 1100
1070  IF A2$ = "$,Y" AND  LEN (A$) = 9 THEN NOD =
     12: GOTO 1100
1080  IF A2$ = "($)" THEN NOD = 12: GOTO 1100
1090  CALL  - 198: CALL  - 198: GOSUB 2910: GOTO 7
     20
1100  REM  Trova il codice
1110  IF FE = 1 THEN  GOSUB 3530
1120  FOR ZZ = 1 TO 64
1140  IF IS$ = I0$(ZZ) THEN 1160
1150  NEXT ZZ: CALL  - 198: CALL  - 198: GOSUB 291
     0: GOTO 720
1160  CALL 768,150 + ZZ
1190  FOR ZZ = 1 TO NOD
1200  READ NC$
1210  NEXT : IF NC$ = "" THEN  PRINT " ADRS.ERR."
     : GOTO 1090
1220 ST = ST + 1:NC =  VAL (NC$)
1230  POKE ST,NC
1240 F2 = 0: IF F4 = 1 THEN 1280
1250  IF NOD = 1 OR NOD = 2 THEN F2 = 1: GOTO 1310

1260 O8 = 16:N8 = 10
1270  GOSUB 2920
1280 ST = ST + 1: POKE ST, VAL (P8$)
1290  IF FL <  > 2 THEN 1310
1300 ST = ST + 1: POKE ST, VAL (P4$)
1310 F1 = FL
1320  IF  LEN (O1$) = 2 OR  LEN (O1$) = 1 OR NOD =
     1 THEN O$ =  STR$ (ST - 1): GOTO 1350
1330  IF  LEN (O1$) > 2 THEN O$ =  STR$ (ST - 2): GOTO
     1350
1340 O$ =  STR$ (ST)
1350 FL = 3:O8 = 10:N8 = 16: GOSUB 2920:P3$ = N$: IF
      LEN (P3$) < 4 THEN P3$ = "0" + P3$
1360 O$ =  STR$ (NC): GOSUB 2920: IF  LEN (N$) < 2
     THEN N$ = "0" + N$
1370  VTAB 22: CALL  - 912: CALL  - 958: HTAB 1
1380  IF F2 = 1 THEN  PRINT P3$; SPC( 5 -  LEN (P3
     $))N$;"           ": GOTO 1480
1390  PRINT P3$; SPC( 5 -  LEN (P3$))N$;" ";
1400  IF F4 = 1 THEN P1$ = P4$
1410  IF  LEN (P1$) < 2 THEN P1$ = "0" + P1$
1420  PRINT P1$;" ";
1430  IF F4 = 1 THEN 1460
1440  IF F1 = 2 AND  LEN (P2$) < 2 THEN P2$ = "0" +
     P2$
1450  IF F1 = 2 THEN  PRINT P2$;
1460  IF F1 = 1 THEN  PRINT "  ";
1470  PRINT " ";
1480  PRINT A$
1490 K = K + 1:V$ = P3$ + " " + N$: IF F2 = 1 THEN
     V1$(K) = V$:V$ = V$ + "           ": GOTO 1530
1500 V$ = V$ + " " + P1$
1510  IF F1 = 2 THEN V1$(K) = V$ + " " + P2$:V$ =
     V$ + " " + P2$ + " ": GOTO 1530
1520 V1$(K) = V$:V$ = V$ + "  "
1530 V$ = V$ + A$: IF FE = 1 THEN V$(K1) = V$: GOTO
     1900
1540 V$(K) = V$
1550  GOSUB 2910: GOTO 720
1560 :
1570  REM   Edit programma
1580 :
1590  HOME : VTAB 23:K1 = 0
1600 K1 = K1 + 1: PRINT V$(K1)
1610  GET C$
1620  IF C$ = " " THEN 1600
1630  IF C$ =  CHR$ (27) THEN 500
1640  IF C$ =  CHR$ (1) THEN 2070
1650  IF C$ =  CHR$ (5) THEN 1680
1660  IF C$ =  CHR$ (9) THEN K1 = K1 + 1: PRINT : GOTO
     1840
1670  GOTO 1610
1680  REM  Cancellamento istruzione
1690  GOSUB 3430:FL = 1
1700  FOR J = K1 TO K
1710 V$(J) = V$(J + 1):V1$(J) = V1$(J + 1): NEXT
1720  FOR J = K1 TO K - 1
1730 O$ =  LEFT$ (V$(J),4):O8 = 16:N8 = 10: GOSUB
     2920:S2 =  VAL (N$) - DC
1740 O8 = 10:N8 = 16:O$ =  STR$ (S2): GOSUB 2920: IF
      LEN (N$) < 4 THEN N$ = "0" + N$
1750 V$(J) =  RIGHT$ (V$(J), LEN (V$(J)) - 4):V$(J
     ) = N$ + V$(J)
1760 V1$(J) =  RIGHT$ (V1$(J), LEN (V1$(J)) - 4):V
     1$(J) = N$ + V1$(J)
1770  IF J = K1 THEN  GOSUB 1800
1780  NEXT
1790 K = K - 1:EN = EN - DC: VTAB 22: CALL  - 912:
     CALL  - 958: PRINT V$(K1): GOTO 1610
1800  FOR Z = S2 + DC TO EN
1810  POKE Z - DC, PEEK (Z)
1820  NEXT
1830  RETURN
1840  REM  Inserimento istruzione
1850  FOR Z = K TO K1 STEP  - 1
1860 V$(Z + 1) = V$(Z):V1$(Z + 1) = V1$(Z)
1870  NEXT
1880 O$ =  LEFT$ (V$(K1),4):O8 = 16:N8 = 10: GOSUB
     2920:ST =  VAL (N$) - 1
1890  GOSUB 2910:FE = 1: CALL  - 198: GOTO 720
1900  GOSUB 3430
1910  IF DC = 3 THEN V1$(K1) =  LEFT$ (V$(K1),13)
1920  IF DC = 2 THEN V1$(K1) =  LEFT$ (V$(K1),10)
1930  IF DC = 1 THEN V1$(K1) =  LEFT$ (V$(K1),7)
1940  FOR J = K1 + 1 TO K
1950 O$ =  LEFT$ (V$(J),4):O8 = 16:N8 = 10: GOSUB
     2920:S2 =  VAL (N$) + IN
1960 O8 = 10:N8 = 16:O$ =  STR$ (S2): GOSUB 2920: IF
      LEN (N$) < 4 THEN N$ = "0" + N$
1970 V$(J) =  RIGHT$ (V$(J), LEN (V$(J)) - 4):V$(J
     ) = N$ + V$(J)
1980 V1$(J) =  RIGHT$ (V1$(J), LEN (V1$(J)) - 4):V
     1$(J) = N$ + V1$(J)

1990  NEXT
2000 EN = EN + IN
2010 BF = K1:K1 = K: GOSUB 3430
2020  IF DC = 3 THEN V1$(K) =  LEFT$ (V$(K),13)
2030  IF DC = 2 THEN V1$(K) =  LEFT$ (V$(K),10)
2040  IF DC = 1 THEN V1$(K) =  LEFT$ (V$(K),7)
2050 K1 = BF
2060  GOTO 1610
2070  REM  Adattamento dei salti relativi
2080  FOR J = 1 TO K
2090  IF  LEN (V1$(J)) = 10 THEN 2120
2100  NEXT
2110  GOTO 500
2120 AA$ =  MID$ (V1$(J),6,2)
2130  IF AA$ = "10" OR AA$ = "30" OR AA$ = "50" OR
     AA$ = "70" OR AA$ = "80" OR AA$ = "90" OR AA$
     = "B0" OR AA$ = "D0" OR AA$ = "F0" THEN  GOSUB
     2150
2140  GOTO 2100
2150 O$ =  LEFT$ (V1$(J),4):O8 = 16:N8 = 10: GOSUB
     2920:ST =  VAL (N$) - 1
2160 FA = 1:O1$ =  RIGHT$ (V$(J),4): GOTO 3130
2170  IF  LEN (P4$) < 2 THEN P4$ = "0" + P4$
2180 V1$(J) =  LEFT$ (V1$(J),8) + P4$
2190 V$(J) =  LEFT$ (V$(J),8) + P4$ +  RIGHT$ (V$(
     J),13)
2200  POKE ST + 2, VAL (P8$)
2210  RETURN
2220  REM  Caricamento da disco
2230  VTAB 22: INPUT "NOME DEL PROGRAMMA ? ";NP$
2240  IF NP$ =  CHR$ (27) THEN 500
2250 NP$ = NP$ + ".TXT":FL = 1:O8 = 16:N8 = 10
2260  PRINT D$;"OPEN "NP$
2270  PRINT D$;"READ "NP$
2280  INPUT S1: INPUT EN: INPUT K
2290  FOR I = 1 TO K: INPUT V1$(I): INPUT V$(I): NEXT

2300  PRINT D$;"CLOSE "NP$
2310 ST = S1 - 1
2320  FOR Z = 1 TO K
2330  IF  LEN (V1$(Z)) = 10 THEN N2 = 2
2340  IF  LEN (V1$(Z)) = 13 THEN N2 = 3
2350  IF  LEN (V1$(Z)) = 7 THEN N2 = 1
2360  FOR J = 1 TO N2
2370 ST = ST + 1: IF J = 1 THEN O$ =  MID$ (V1$(Z)
     ,6,2)
2380  IF J = 2 THEN O$ =  MID$ (V1$(Z),9,2)
2390  IF J = 3 THEN O$ =  MID$ (V1$(Z),12,2)
2400  GOSUB 2920: POKE ST, VAL (N$): NEXT
2410  NEXT
2420  GOTO 500
2430  REM  Memorizzazione su disco
2440  HOME : INPUT "NOME DEL PROGRAMMA ? ";NP$: IF
     NP$ =  CHR$ (27) THEN 500
2450  PRINT : INPUT "VUOI SALVARE IL PROGRAMMA COM
     E FILE DI TESTO (T) O BINARIO (B) ? ";CA$
2460  IF CA$ = "T" THEN 2500
2470 LN = (EN - S1) + 1:NP$ = NP$ + ",A" +  STR$ (
     S1) + ",L" +  STR$ (LN)
2480  PRINT D$;"BSAVE "NP$
2490  GOTO 500
2500 NP$ = NP$ + ".TXT"
2510  PRINT D$;"OPEN "NP$
2520  PRINT D$;"WRITE "NP$
2530  PRINT S1: PRINT EN: PRINT K
2540  FOR I = 1 TO K: PRINT V1$(I): PRINT V$(I): NEXT

2550  PRINT D$;"CLOSE "NP$
2560  GOTO 500
2570  REM  Visualizzazione
```

Listato del programma in Basic che permette di assemblare e correggere un programma in linguaggio macchina scritto per il 65C02.

```
2580  HOME : INPUT "VUOI LA VISUALIZZAZIONE SU VID
      EO (V) O SU STAMPANTE (S) ? ";CA$: HOME
2590  IF CA$ = CHR$ (27) THEN 500
2600  IF CA$ = "S" THEN  PRINT D$;"PR#1";FV = 0: GOTO
      2620
2610 FV = 1
2620 J = 0
2630  FOR I = 1 TO K
2640  PRINT V$(I);J = J + 1
2650  IF FV = 1 AND J = 23 THEN  GET B$
2660  NEXT
2670  IF CA$ = "S" THEN  PRINT D$;"PR#0"
2680  IF CA$ = "V" THEN  GET B$: PRINT
2690  GOTO 500
2700  REM  Operazioni dos
2710  HOME : VTAB 3: HTAB 5
2720  PRINT "0 Ritorno al main menu": PRINT : HTAB
      5
2730  PRINT "1 Catalog": PRINT : HTAB 5
2740  PRINT "2 Del files": PRINT : HTAB 5
2750  PRINT "3 Rename files": PRINT : HTAB 5: PRINT
      ">";
2760  GET B$: PRINT
2770  IF B$ = "0" THEN 500
2780  IF B$ = "1" THEN  PRINT D$;"CATALOG": GET A$
      : PRINT
2790  IF B$ = "2" THEN 2870
2800  IF B$ = "3" THEN 2820
2810  GOTO 2710
2820  VTAB 22: INPUT "NOME DEL FILE ? ";VN$
2830  IF VN$ =  CHR$ (27) THEN 2710
2840  INPUT "NUOVO NOME ? ";NN$
2850  PRINT D$;"RENAME ";VN$;",";NN$
2860  GOTO 2710
2870  VTAB 22: INPUT "NOME DEL FILE ? ";VN$
2880  IF VN$ =  CHR$ (27) THEN 2710
2890  PRINT D$;"DELETE ";VN$
2900  GOTO 2710
2910 A$ = "":I1 = 0: HTAB 15: RETURN
2920  REM  Conversione esadec<->dec
2930 I2 = 1
2940 X = 0:N = 0:L% = 0:X% = 0:N$ = ""
2950  IF D$ = "0" OR D$ = "00" THEN N$ = "0": GOTO
      3030
2960  FOR I = 1 TO  LEN (D$)
2970 X =  ASC ( MID$ (D$,I,1)) - 48:X = X - 7 * (X
      > 9):N = N + D8 * X
2980  NEXT
2990 N = N + .5:L% =  LOG (N) /  LOG (NB)
3000  FOR I = L% TO 0 STEP  - 1
3010 X% = N / NB ^ I:N = N - X% * NB ^ I:N$ = N$ +
      CHR$ (48 + X% + 7 * (X% > 9))
3020  NEXT
3030  IF FL <  > 2 THEN PB$ = N$: RETURN
3040  IF I2 = 1 THEN P8$ = N$
3050 I2 = I2 + 1
3060  IF I2 = 2 THEN 3090
3070 PA$ = N$
3080  RETURN
3090  IF  LEN (O1$) = 4 THEN D$ =  LEFT$ (O1$,2): GOTO
      3110
3100 D$ =  LEFT$ (O1$,1)
3110 P2$ = D$
3120  GOTO 2940
3130  REM  Calcolo per i salti relativi
3140  IF S1 = ST THEN S3 = ST: GOTO 3160
3150 S3 = ST + 1
3160 FL = 1:F4 = 1:D$ = O1$:DB = 16:NB = 10: GOSUB
      2920:S2 =  VAL  (N$)
3170 DF = S3 - S2: IF  ABS (DF) > 128 THEN  CALL  -
      198: CALL  - 198: GOTO 720
3180  IF DF > 0 THEN 3220
3190 S3 = S3 + 2:DF = S2 - S3:
3200 D$ =  STR$ (DF):FL = 1:DB = 10:NB = 16: GOSUB
      2920:P4$ = N$:P8$ = D$: IF FA = 1 THEN 2170
3210  GOTO 1100
3220 DF = DF + 2:D$ =  STR$ (DF):DB = 10:NB = 2
3230  GOSUB 2920:N2$ = ""
3240  IF  LEN (N$) < 8 THEN  GOSUB 3400
3250  FOR Z =  LEN (N$) TO 1 STEP  - 1
3260 AA$ =  MID$ (N$,Z,1)
3270  IF AA$ = "1" THEN 3300
3280 N2$ = AA$ + N2$
3290  NEXT
3300 N2$ = AA$ + N2$
3310  FOR Z1 = Z - 1 TO 1 STEP  - 1
3320 AA$ =  MID$ (N$,Z1,1)
3330  IF AA$ = "1" THEN AA$ = "0": GOTO 3350
3340 AA$ = "1"
3350 N2$ = AA$ + N2$
3360  NEXT Z1
3370 D$ = N2$:DB = 2:NB = 16: GOSUB 2920:P4$ = N$:
      D$ = N$:DB = 16:NB = 10
3380  GOSUB 2920:P8$ = N$: IF FA = 1 THEN 2170
3390  GOTO 1100
3400  FOR Z = 1 TO 8 -  LEN (N$)
3410 N$ = "0" + N$
3420  NEXT : RETURN
3430  REM  Calcolo del decremento
3440 I3 = 0
3450  FOR Z = 1 TO  LEN (V$(K1))
3460 AA$ =  MID$ (V$(K1),Z,1)
3470  IF AA$ = " " THEN I3 = I3 + 1
3480  NEXT
3490  IF I3 = 7 THEN DC = 2
3500  IF I3 = 8 THEN DC = 1
3510  IF I3 = 5 THEN DC = 3
3520  RETURN
3530  REM  Calcolo incremento
3540  IF  LEN (O1$) > 2 THEN IN = 3: GOTO 3570
3550  IF  LEN (O1$) = 1 OR  LEN (O1$) = 2 THEN IN =
      2: GOTO 3570
3560 IN = 1
3570  FOR Z = EN TO ST STEP  - 1
3580  POKE Z + IN, PEEK (Z)
3590  NEXT
3600  RETURN
```

e l'istruzione stessa, poi nuovamente il cursore per un nuovo inserimento. Per ritornare al menu principale bisogna premere ESC, come in quasi tutte le altre parti del programma.

Premendo "2", nel menu principale, si entra nell'edit e viene visualizzata la prima istruzione; premendo la barra spaziatrice verranno visualizzate le successive. Se si vuole inserire un'istruzione bisogna portarsi con il cursore, premendo la barra, sotto l'istruzione che precede quella che si vuole aggiungere e premere CTRL I; un suono segnalerà che si può procedere con l'inserimento. Premuto il CR si dovrà attendere un certo tempo, che dipende dalla lunghezza del programma che si è inserito, dopodiché riapparirà il cursore. Per eliminare un'istruzione bisogna portarcisi sotto con il cursore e premere CTRL E; anche per questa operazione si deve aspettare un certo tempo. Se si fanno delle modifiche al proprio programma tramite l'edit e, nel programma, sono presenti degli indirizzamenti relativi, alla fine delle correzioni (prima di ritornare al menu pricipale), si deve premere CTRL A; questo serve a ricalcolare i salti che effettuano le diramazioni. Per tornare al menu premerre ESC.

L'opzione "CARICAMENTO DA DISCO" serve per riprendere e correggere un programma che era stato scritto in precedenza ed era stato memorizzato sotto forma di file. Per questa operazione ci vuole un po' di tempo perché il programma viene inserito in memoria byte per byte. Una volta caricato si ritorna automaticamente al menu.

Con "MEMORIZZAZIONE SU DISCO", si possono salvare i programmi in due modi: come file binario, quindi come si salvano abitualmente i programmi Assembler, o come file di testo. Il secondo metodo permette di correggere i programmi in un secondo tempo, ma questi non sono direttamente eseguibili, mentre il primo no. Quando si sceglie di registrare i programmi sotto forma di file, viene aggiunto al nome del programma il suffisso ".TXT" (esso è aggiunto automaticamente anche quando si carica il programma).

Con la scelta "VISUALIZZAZIONE", si può avere il listato del programma in Assembler su video o su stampante. L'interfaccia della stampante (DMP o IMAGE WRITER) è supposta nello slot 1. Se la visualizzazione avviene su schermo, il listato si blocca ad ogni schermata; per farlo continuare basta premere un tasto qualsiasi.

L'ultima opzione serve per poter compiere alcune utili operazioni DOS senza dover uscire dal programma. Per il ritorno al menu principale premere "0".

Note

La routine di RESTORE numero di riga citata dall'autore era piuttosto datata (e conteneva pure un piccolo bug) l'ho perciò sostituita con una nuova che, utilizzando le routine dell'interprete è molto più corta ed affidabile.

Il fatto di non poter utilizzare la virgola negli mnemonici dipende dalla routine che salva il sorgente come file di testo; se si scrive un pezzetto di programma che sostituisce le virgole con i punti in fase di salvataggio e viceversa in fase di lettura; si può assemblare utilizzando la notazione corretta cambiando il punto con la virgola nelle righe 900, 1010... 1070.

A volte se si scrive un indirizzo a tre cifre il disassemblato lo visualizza errato, ma il codice oggetto è corretto, conviene quindi scrivere sempre gli indirizzi con due o quattro cifre.

Attenzione a non assemblare programmi nell'area $300.$317 che è occupata dalla routine di restore, né prima della locazione $2000 (ci sta il programma in Basic e le sue variabili) o oltre la locazione $8000 (ci si trovano i vettori di editing).

Sarebbe anzi opportuno modificare il programma in modo che la fase di assemblaggio si svolgesse in una WORK AREA sempre libera (ad esempio da $3000 a $6000) e solo al momento del salvataggio venissero corretti gli indirizzi.

Questo programma è disponibile su disco presso la redazione. Vedere l'elenco dei programmi disponibili e le istruzioni per l'acquisto a pag. 159.

Le istruzioni del 65C02

Con la realizzazione del 65C02 i progettisti si erano prefissati il raggiungimento di due mete, la prima era il risparmio energetico, onde consentire il montaggio del 65C02 sui nuovi personal portatili, la seconda era di aggiungere al set di istruzioni alcuni comandi utili e togliere qualche bug ancora presente nelle ultime release del 6502. Il tutto mantenendo la massima compatibilità con il set di istruzioni precedente in modo da poter utilizzare tutto il software finora sviluppato.

TABELLA DEI CODICI DEL 65C02

ADC +	CLI	LDY	SBC +
AND +	CLV	LSR	SEC
ALS	CMP +	NOP +	SED
BCC	CPX	ORA +	SEI
BCS	CPY	PHA	STA +
BEQ	DEC +	PHP	STX
BIT +	DEX	PHX *	STY
BMI	DEY	PHY *	STZ *
BNE	EOR +	PLA	TAX
BPL	INC +	PLP	TAY
BRA *	INX	PLX *	TRB *
BRK	INY	PLY *	TSB *
BVC	JMP +	ROL	TSX
BVS	JSR	ROR	TXA
CLC	LDA +	RTI	TXS
CLD	LDX	RTS	TYA

* Nuova istruzione + Nuove possibilità

Le nuove istruzioni consentono la diramazione senza alcun test (BRA = BRANCH ALWAIS), il controllo dei singoli bit di memoria (TRB = TEST and RESET BIT e TSB = TEST and SET BIT) e i trasferimenti diretti dei registri X ed Y da, e verso, lo STACK (PHX, PHY, PLX, PLY) e infine la possibilità di azzerare direttamente una locazione di memoria (STZ = Storage Zero).
Sono state inoltre aggiunte nuove possibilità ad alcune istruzioni preesistenti tanto che certune si sono in pratica trasformate in nuove istruzioni; è il caso della INC e della DEC che, se scritte senza argomento, lavorano sul contenuto dell'accumulatore e che alcuni assemblatori hanno ribattezzato (a torto secondo me) DEA ed INA. La BIT è ora anche immediata (anche se in modo immediato esegue solo l'AND tra il dato e l'accumulatore), è stato aggiunto un JMP indicizzato indiretto e tutti i codici non utilizzati sono stati trasformati in NOP che, a seconda del tipo, possono ora occupare da 1 a 3 byte e impiegare da 1 ad 8 cicli (vedi tabella 1). Inoltre è stato aggiunto un indirizzamento indiretto che non utilizza più il registro Y.
Sono poi stati corretti alcuni bug: il JMP indicizzato che oltrepassa il confine di pagina ora funziona, un interrupt che si presenti durante un BRK viene eseguito regolarmente, il decimal-mode viene cancellato ad ogni Reset od Interrupt, e, sempre in decimal-mode, vengono settati correttamente i flag N, V e Z.
Per quanto riguarda le caratteristiche hardware, tutti gli ingressi non utilizzati possono essere lasciati liberi in quanto collegati internamente al positivo con una resistenza da circa 1Mohm, si può mettere direttamente un quarzo tra Φ1 e Φ12 o se, si utilizza un generatore esterno, è sufficiente la sola fase Φ2; alcuni problemi possono insorgere con le periferiche non standard Apple in quanto le operazioni di lettura/scrittura sono ora composte da due cicli di scrittura e uno di lettura anziché due di lettura e una di scrittura come in precedenza avveniva nel 6502.
Tutte queste informazioni riguardano ovviamente coloro i quali volessero provare a sostituire brutalmente il loro 6502 con la versione CMOS (Complementary Metal Oxide Semiconductor).

Le istruzioni nascoste del 6502

Il set di istruzioni del 6502 è composto da 56 istruzioni definite standard; questo significa che chiunque costruisca un microprocessore 6502 deve garantire che quelle istruzioni si comportino nel dovuto modo. In realtà nella ROM interna del microprocessore i codici delle istruzioni che non fanno parte dello standard non sono lasciati vuoti, ma fanno qualche cosa, solo che questo qualche cosa non è ben definito e non è detto che corrisponda allo

Nuove NOP

Cod.	Byte	Cicli
x2	2	2
x3 x7 xB xF	1	1
44	2	3
54 D4 F4	2	4
5C	3	8
DC FC	3	4

Tabella 1 - Le istruzioni non ancora implementate sul 65C02 sono utilizzabili come NOP, in tal caso, avendo diverse lunghezze e differenti tempi di esecuzione possono anche essere utilizzate per generare precise temporizzazioni.

Istruzioni nascoste del 6502

Nome Descrizione	Operazione	Modo di Indirizzo	Mnemonico	Codice	N.Byte
DEC decrementa	M-1 := M	(Ind,X)	DEC (Zp,X)	C3	2
		(Ind),Y	DEC (Zp),Y	D3	2
INC incrementa'	M+1 := M	(Ind,X)	INC (Zp,X)	E3	2
		(Ind),Y	INC (Zp),Y	F3	2
		ZeroP,Y	INC Zp,Y	F7	2
'nota: distrugge il contenuto dell'Accumulatore					
LAX carica in A ed X	M := A=X	ZeroP	LAX Zp	A7	2
		ZeroP,Y	LAX Zp,Y	B7	2
		Assoluto	LAX abs	AF	3
		Ass, Y	LAX abs,Y	BF	3
		(Ind,X)	LAX (ZP,X)	A3	2
		(Ind),Y	LAX (Zp),Y	B3	2
STZ Azzera	0 := M	ZeroP	STZ Zp	87	2
		ZeroP,Y	STZ Zp,Y	97	2
		Assoluto	STZ abs	8F	3
		Ass, Y	STZ abs,Y	9F	3
		Ass, X	STZ abs,X	9C	3'
		(Ind,X)	STZ (Zp,X)	83	2
		(Ind),Y	STZ (ZP),Y	93	2
'nota: Il Byte Alto deve essere diverso da zero!					
ST1 mette ad 1	1 := M	ZeroP,X	ST1 abs,X	9C	3'
'nota: Il Byte alto deve essere zero !					
NOP2	-	-	NOP2	34	2
NOP3	-	-	NOP3	3C	3

Tabella 2 - Ventuno dei codici non utilizzati nel 6502 corrispondono ad altrettante nuove istruzioni. In tabella quelle del 6502 della Synertek. Da notare la ST1 che, pur essendo a tre byte, lavora solo in pagina zero e deve avere perciò la parte alta uguale a zero.

stesso «qualche cosa» di un altro costruttore.

Come scoprire allora cosa fanno i codici illeciti? Si può saperlo scrivendo un programmino composto da poche istruzioni tra cui una di quelle illecite.

Un programma tipo è il seguente:

```
LDA #$20
STA $06
STA $08 questo pacchetto prepara il campo
per gli indirizzamenti indiretti.
LDA #$30
STA $07
LDA #$40
STA $09
...
LDA #$AC (un valore quasiasi)
LDY #$01
LDX #$02
...
XXX codice illecito
    $06 operando
...
BRK
BRK almeno due.
```

Poi occorre scrivere una serie di numeri (a caso) nelle locazioni da $2000 a $2030. A questo punto salvate il programma su disco e poi togliete il disco dal drive e date il GO al programma di prova. Si possono presentare due casi: o il computer si inchioda, oppure, eseguita l'istruzione illecita, va in Break mostrando il contenuto dei registri. Se si inchioda il codice illecito non è buono (a meno che non si voglia utilizzare proprio per proteggere un programma); premere il reset e rifare il bootstrap. A volte, soprattutto con codici inferiori a $50, il tasto di reset non funziona nemmeno e si deve spegnere e riaccendere la macchina; in ogni caso non cercare di effettuare operazioni sul disco dopo che il programma di prova ha bloccato la macchina.

Nel caso in cui si sia raggiunta l'istruzione BRK bisogna innanzitutto vedere il valore del program counter che è di due indirizzi maggiore della locazione che lo ha generato, dal valore del program counter si scopre subito la lunghezza dell'istruzione corrispondente al codice in prova, visto che il BRK eseguito è il primo dopo l'ultimo byte dell'istruzione. Poi guardando cosa è successo ai valori dei registri si cerca di scoprire cosa ha combinato l'istruzione in esame, se i registri non hanno cambiato valore si deve andare a leggere tutti i valori delle locazioni di memoria che potrebbero essere state interessate dal comando, nel nostro esempio le locazioni 6, 7, 8, 9 e 2000, 2001, 2002, 2030, 2031, ecc.

Nella pagina accanto (tabella 2) trovate i codici illeciti del mio Apple (che monta un 6502 della Synertek) e che ho trovato abbastanza facilmente in un pomeriggio di lavoro (la maggior parte del tempo si perde a rifare il boot). Se qualcuno ne scopre altri e ce li comunica ci premureremo di spargere la voce.

Il DYNAMO per Apple II

Molti lettori, interessati a sviluppare programmi di simulazione in DYNAMO (presentato nell'ultima puntata della simulazione su MC n. 46), hanno chiesto dove sia possibile reperire il compilatore MICRO-DYNAMO per Apple ed IBM PC.

Il disco col programma viene venduto dalla ADDISON WESLEY come supplemento al libro:

Introduction to computer simulation the system dynamic approach
di Nancy Roberts
I.S.B.N. codice del libro: 0201.064.146; codice del software: 0201.10650.7

Quanto corre questo programma?

Le invio una routine Assembler, ottenuta modificando un esercizio tratto dalla serie di articoli «impariamo a programmare in Assembler» (MC dal n. 20 al n. 27), che permette di pulire contemporaneamente tutte e due le pagine grafiche. Ora vorrei sapere come si fa a calcolarne la velocità di esecuzione.
Alberto Cipolla (TV)

Per calcolare la velocità di esecuzione di un programma in linguaggio macchina si deve moltiplicare il tempo che impiega il microprocessore ad eseguire le singole istruzioni (tempo che si ricava dalla tabella 1 pubblicata a pagina 82 su MC n. 27) per il numero di volte che queste vengono eseguite. Ricordando che alcune istruzioni come i salti o gli indirizzamenti indicizzati necessitano di un ciclo in più qualora si verifichi la diramazione o si attraversi il confine di pagina (vedi figura sottostante). Naturalmente è impensabile usare questo metodo per programmi molto lunghi. MC

```
                                   CICLI      VOLTE

300- A9 A1      LDA #$A1            2          1
302- A0 20      LDY #$20            2          1
304- 8C 0D 03   STY $30D            4          1
307- A0 40      LDY #$40            2          1
309- A2 00      LDX #$00            2         64
30B- 9D 00 20   STA $2000           5         64 * 256
30E- CA         DEX                 2         64 * 256
30F- D0 FA      BNE $30B            3         64 * 255
311- EE 0D 03   INC $30D            6         64
314- 88         DEY                 2         64
315- D0 F2      BNE $309            3         64 (-1)
316- 60         RTS                 6          1

                              CICLI TOTALI :     164495
                TEMPO TOTALE (CLOCK=1 Mhz) :   0.16 secondi
```

Inviate i vostri programmi

Alcuni lettori ci chiedono, nelle loro lettere, come sottoporre i loro programmi a MC.

Registrate i vostri lavori su cassetta o disco (se il programma è proprio molto corto può bastare il semplice listato; certo, la cassetta non guasta mai...), corredateli dell'opportuna documentazione e spedite il tutto alla redazione, indicando magari sulla busta la rubrica interessata.

Tutti i programmi che arrivano sono esaminati ed i migliori pubblicati.

Purtroppo non possiamo restituire, per ragioni organizzative, il materiale che ci viene inviato, anche in caso di mancata pubblicazione.

Ricordatevi che migliore è la documentazione, maggiore è la possibilità che il vostro lavoro venga pubblicato: spiegate quindi chiaramente il funzionamento del programma ed accludete tutto quello che pensate possa essere utile (elenco variabili e via dicendo). Soprattutto non dimenticate di indicare il computer sul quale il programma gira, né il vostro nome e indirizzo e, se possibile, il numero di telefono. Indicate anche, per la retribuzione se il programma sarà pubblicato, luogo e data di nascita, domicilio fiscale e codice fiscale (partita IVA, se la possedete).

Il compenso per i programmi pubblicati varia normalmente fra le 40 e le 150.000 lire, a seconda della qualità del lavoro inviato; eventuali programmi di particolare complessità ed interesse potranno essere valutati al di fuori di questo standard, previ accordi con la redazione.