Assembler in BASIC for the PET

Eric Brandon

The most important tool the machine language programmer has is his assembler. If he has 32K, a disk, new ROMs, and $170, he can buy MAE or the Commodore assembler. Otherwise, his choice is limited. If the thing he lacks the most is the $170, or if he just wants to dabble in machine language, he will often end up with Newmon, Supermon, or a similar extended monitor that has a simple assembler. This is fine, but these assemblers are not symbolic, and he must calculate branches, jumps, and the like every time he modifies or relocates it.

Since I have an 8K PET with old ROMs, the only option open to me was to write my own symbolic assembler. I had to decide whether to write it in machine language or BASIC. Since I could not face the bleak prospect of writing a parser in machine language, I wrote it in BASIC. An added bonus is that it works on all ROMs (I have tested it on 2.0 and 3.0), and can easily be used by readers of **COMPUTE!** with other machines, since they all use the 6502.

The first thing you must do after typing it in, is to change line 1 to suit your memory size. MEM is the number of lines of machine code it can hold. When MEM is large, not only does it eat up memory, but it slows down the assembly process as well. M2 is the number of symbols it can keep track of. For an 8K PET, I suggest MEM = 40 and M2 = 20; larger values are likely to run out of memory. For 16K or 32K PETs, MEM = 200 and M2 = 100, unless you know you are going to use more lines or symbols.

When you RUN it, you will see the number 1, and a white cursor beside it. This means you are in line 1, and that it is waiting for input into the label field. If you type to the end of the field, hit SPACE, or hit RETURN, you will jump to the next field. The length of the label field is 6 characters, the operation field is 3, and the operand field is 10 characters long. A SPACE or RETURN in the operand field will put you at the beginning of the next line. When in the label field, there are two special commands you can type in. You type "FIX" when you have made a mistake. FIX returns you to the previous line so that if you type FIX on line 20, a 19 with a cursor beside it will appear underneath.

Typing "EXIT" will bring you to a menu. Type "I" to input some more code. You will be asked at what line you wish to begin inputting. Type "D" to delete. You will be asked for a starting and an endinng line number. The starting line and everything up to, but not including, the ending line will be deleted. To delete just one line, type its number as both starting and ending lines. Type "N" to insert. You will be asked what line to begin inserting at and how many lines to insert. All lines including and below the one you specified will be moved down the number of lines you said, leaving a gap of blank lines behind. Type "L" for list; you supply the beginning and ending line numbers. Type "A" to assemble your program. You must specify whether you want the output to go to the screen printer. Note that if you choose the screen, there will often be lines of more than 40 characters since the output was designed for the printer. If you plan to use the screen exclusively, I suggest you modify lines 1180–1210 to make the output less than 40 characters long. Note that your program has been POKEd into memory and may be executed with an SYS after assembly. Type "Q" to quit the program. If you hit Q accidentaly, a GOTO 300 will return you to the program with your work intact.

I have included a sample printout which you should consult as I explain the operation of the assembler. As you can see, a symbol table is generated before the actual code. An "=" as the operation will set the symbol on the left equal to the value of the expression on the right. The first line should be an asterisk equal to a value. This sets the origin of the program in memory. The origin may be set only once, and only in the first line; any other attempt will give unpredictable results. Immediate addressing must be indicated with a pound sign as in lines 8 and 16 of my sample program. Hexadecimal numbers must be preceded by a dollar sign, and must be 1 to 4 characters in length. Binary numbers must be preceded by a percentage sign, and may be of any length. Decimal is assumed by default. A symbol must begin with a letter, and contain only letters and numbers. In accumulator addressing, the operand must be the letter "A", therefore "A" is an illegal symbol, although symbols may contain and begin with the letter "A". Addition within the operand field is non-standard. Only

```
CHAR =     $03E0
SCRN =     $0020
AGAIN =    $0342
LOOP =     $0349
 1                          *       =    $33A
 2                  CHAR    =    $3E0
 3                  SCRN    =    $20
 4    033A A9 00            LDA  #0
 5    033C A8               TAY
 6    033D 8D E0 03         STA  CHAR
 7    0340 85 20            STA  SCRN
 8    0342 A9 80     AGAIN  LDA  #$80
 9    0344 85 21            STA  SCRN+
10    0346 AD E0 03         LDA  CHAR
11    0349 91 20     LOOP   STA  (SCRN),Y
12    034B C8               INY
13    034C D0 FB            BNE  LOOP
14    034E E6 21            INC  SCRN+
15    0350 A6 21            LDX  SCRN+
16    0352 E0 84            CPX  #$84
17    0354 D0 F3            BNE  LOOP
18    0356 EE E0 03         INC  CHAR
19    0359 D0 E7            BNE  AGAIN
20    035B 60               RTS
```

symbols can be added to and addition is done by following the symbol with plus signs as in lines 9, 14, and 15 of the sample program. The number of plus signs is equal to the number to be added. If BUFFER = 30, LDA (BUFFER+ +,X) means LDA (32,X). I suggest that when writing self-modifying code, you put the code to be modified physically before the code that modifies it. Otherwise, you are almost sure to get an error message. By the way, my sample program fills the screen with every possible character, and is an excellent demonstration of the speed of machine language. It is written for the old ROM PET, but will work on the new ROMs if line 3 is changed to SCRN = B5.

Here is a quick summary of the part of my program for those who may wish to modify it:

Lines 100–200 : control the input, and use the input routine at line 4000.
Lines 300–600 : execute the command options other than ASSEMBLE.
Lines 660–770 : create the symbol table.
Lines 790–1220 : assemble the code.
Lines 4000–4160 : input routine
Lines 5000–5510 : are the op-code tables
Lines 6000–6100 : find the numerical value of the operand.
Lines 7000–7040 : Convert hexadecimal numbers to decimal.
Lines 8000–8020 : Convert binary numbers to decimal.
Lines 9000–9020 : Convert decimal numbers to hexadecimal.
Lines 1000–10100 : Separate the labels, operations, and operands from the packed array A$.

The program is quite compact because it had to be compressed to fit in an 8K PET. As it is, 8K can only hold about 40 lines of machine code along with the program before running out of memory.

Improvements that could be made are: compacting the code even more, putting in READ/WRITE to cassette or disk routines, putting a BYT pseudo-op, and much more. In the meantime, it is a better assembler than the non-symbolic ones, and hopefully will be of use to PET owners; especially those with old ROMs who have woefully few assemblers available for them. If you find any bugs, make any improvements or have any questions about my program, please write me at:

Eric Brandon
36 Hartfield Road
Islington, Ontario
Canada
M9A 3C9

If you don't want to type it in, send me a blank cassette and $5, and I'll make you a copy.

ASSEMBLER/EDITOR 1.5 — ERIC BRANDON

```
 1 MEM=50:M2=20
 5 PRINT"■"
 10 DIMA$(MEM),S$(M2),V(M2),LI(3)
 15 H$="0123456789ABCDEF"
 100 LN=1
 110 PRINTLN;:TB=5:LT=6:GOSUB4000:IFIN$="EXIT"THEN300
 120 IFIN$="FIX"THENLN=LN-1:PRINTCHR$(-13*(ASC(GT$)<>13));:GOTO110
 125 IF GT$=CHR$(13)THENPRINT"■";
 130 A$(LN)=IN$+" ":TB=13:LT=3:GOSUB4000:A$(LN)=A$(LN)+IN$+" "
 160 IF GT$=CHR$(13)THEN200
 170 TB=18:LT=10:GOSUB4000:A$(LN)=A$(LN)+IN$
 190 IF GT$<>CHR$(13)THENPRINT
 200 LN=LN+1:GOTO110
 300 PRINT"■RI■NPUT ■D■ELETE I■N■SERT ■L■IST ■A■SSEMBLE ■Q■UIT"
 310 PRINT"COMMAND ?";
 320 GETCM$:IFCM$=""THEN320
 325 PRINTCM$:IFCM$<>"I"THEN410
 340 INPUT"LINE ";LN:IFLN=0THEN300
 350 GOTO110
 410 IF CM$<>"D"THEN460
 420 INPUT"■LINES - FROM,TO ";FL,LL:IFFL<>LLTHEN430
 422 FORT=FLTOMEM-1:A$(T)=A$(T+1):NEXTT:GOTO300
 430 FORT=LL TO MEM:A$(T-LL+FL)=A$(T):A$(T)="":NEXTT:GOTO300
 460 IFCM$<>"N"THEN500
 470 INPUT"FIRST LINE,NUMBER";FL,LL:FORT=MEM-LLTOFLSTEP-1:A$(T+LL)=A$(T):NEXTT
 490 FORT=FLTOFL+LL-1:A$(T)="":NEXTT:GOTO300
 500 IF CM$<>"L"THEN580
 510 INPUT"LINES FIRST,LAST";FL,LL:FORT=FLTOLL
 521 IF LEN(A$(T))=0THENPRINTT:GOTO565
 525 LI(1)=0:LI(2)=0:LI(3)=0:LI=0:FORQ=1TOLEN(A$(T))
 540 IF MID$(A$(T),Q,1)=" "THENLI=LI+1:LI(LI)=Q
 545 NEXTQ:IFLI(3)=0THENLI(3)=Q-1
 550 PRINTTTAB(5)LEFT$(A$(T),LI(1))TAB(13)MID$(A$(T),LI(1)+1,LI(2)-LI(1));
 560 PRINTTAB(18)RIGHT$(A$(T),LI(3)-LI(2)+1)
 565 NEXTT:GOTO300
 580 IFCM$<>"Q"THEN600
 590 PRINT"■ET BACK IN WITH ■GOTO 300■":END
 600 IFCM$<>"A"THEN300
 605 PRINT"■RS■CREEN OR ■P■RINTER ?";
 610 GETDV$:IFDV$=""THEN610
 620 PRINTDV$:IFDV$="S"THENDV=3:GOTO650
 640 DV=4
 650 CLOSE1:OPEN1,DV:SB=1
 660 FORT=1TOMEM:GOSUB10000:IFLB$=""THEN710
 670 IF OC$<>"="THEN700
 680 GOSUB6000:IFLB$="*"THENPC=NU:OG=NU:GOTO770
 690 S$(SB)=LB$:V(SB)=NU:SB=SB+1
 692 N=V(SB-1):GOSUB9000
 695 PRINT#1,S$(SB-1)" ="LEFT$("          ",8-LEN(S$(SB-1)))"$"R$:GOTO770
 700 S$(SB)=LB$:V(SB)=PC:SB=SB+1
 702 N=V(SB-1):GOSUB9000
 705 PRINT#1,S$(SB-1)" ="LEFT$("          ",8-LEN(S$(SB-1)))"$"R$
```

```
710 IFOC$=""THEN770
715 IFOP$=""THENPC=PC+1:GOTO770
717 IFOP$="A"THENPC=PC+1:GOTO770
720 IFLEFT$(OC$,1)<>"B"OROC$="BIT"OROC$="BRK"THEN740
730 PC=PC+2:GOTO770
740 IFLEFT$(OC$,1)="J"THENPC=PC+3:GOTO770
750 GOSUB6000:IFNU<256THENPC=PC+2:GOTO770
760 PC=PC+3
770 NEXTT
790 PC=OG:ER=0
800 FORT=1TOMEM:GOSUB10000:IFOC$=""THEN1220
805 IFOC$="="THENO1$=OP$:MV$="  ":PC$="    ":IL=0:GOTO1160
810 IFOP$=""THENAM$="G":IL=1:GOTO1060
820 IFOP$="A"THENAM$="H":IL=1:GOTO1060
825 X=0:Y=0:I=0:M=0:Z=0
830 FORQ=1TOLEN(OP$):Q$=MID$(OP$,Q,1):IFQ$=")"THENI=1:GOTO865
840 IFQ$="#"THENM=1:GOTO865
865 NEXTQ
866 FORQ=1TOLEN(OP$)-1:Q$=MID$(OP$,Q,2)
867 IFQ$=",Y"THENY=1:GOTO870
868 IFQ$=",X"THENX=1
870 NEXTQ
875 O1$=OP$:GOSUB6000
876 IFNU<256THENZ=1
880 IFLEFT$(OC$,1)="B"ANDOC$<>"BRK"ANDOC$<>"BIT"THEN1000
890 IFZTHEN940
900 IFXTHENAM$="K":GOTO1030
910 IFYTHENAM$="L":GOTO1030
920 IFITHENAM$="M":GOTO1030
930 AM$="N":GOTO1030
940 IFMTHENAM$="I":GOTO1030
950 IFIANDYTHENAM$="O":GOTO1030
960 IFIANDXTHENAM$="P":GOTO1030
970 IFXTHENAM$="Q":GOTO1030
980 IFYTHENAM$="R":GOTO1030
990 AM$="S":GOTO1030
1000 AM$="J":IFNU>PC+1THENOS=NU-PC-2:IFOS>127THENER=1
1010 IFNUMBER<PC+1THENOS=254+NU-PC:IFOS<128THENER=1
1020 IFER=1THENPRINT"■TOO LONG CONDITIONAL BRANCH":GOTO300
1025 FO=OS:IL=2:GOTO1060
1030 IFZ=0THEN1050
1040 FO=NU:IL=2:GOTO1060
1050 SO=INT(NU/256):FO=(NU/256-SO)*256:IL=3
1060 RESTORE:FORW9=1TO56:READI$:IFLEFT$(I$,3)=OC$THENCD$=I$:W9=100
1070 NEXTW9:IFW9=57THENPRINT"■ILLEGAL MNEMONIC":GOTO300
1080 FORW9=4TOLEN(CD$)STEP3:IFMID$(CD$,W9,1)=AM$THENLW=W9:W9=100
1090 NEXTW9:IFW9<100THENPRINT"■ILLEGAL ADDRESSING MODE":GOTO300
1100 MV$=MID$(CD$,LW+1,2):N$=MV$:GOSUB7000
1110 POKEPC,V:IFIL=1THEN1140
1120 POKEPC+1,FO:IFIL=2THEN1140
1130 POKEPC+2,SO
1140 N=PC:GOSUB9000:PC$=R$:PC=PC+IL
1150 N=FO:GOSUB9000:FO$=R$:N=SO:GOSUB9000:SO$=R$
1160 IFIL<3THENSO$="  "
1170 IFIL<2THENFO$="  "
1175 IF AM$="H"THENO1$="A"
1180 PRINT#1,TLEFT$("    ",4-LEN(STR$(T)))PC$" ";
1190 PRINT#1,MV$" "RIGHT$(FO$,2)" "RIGHT$(SO$,2)"  ";
1200 PRINT#1,LB$LEFT$("        ",8-LEN(LB$))OC$LEFT$("     ",5-LEN(OC$));
1210 PRINT#1,O1$:O1$=""
1220 NEXTT:GOTO300
3999 END
4000 IN$="":NL=0:PRINTTAB(TB);
4020 PRINT"■ ■";
4030 GETGT$:IFGT$=""THEN4030
```

```
4031 IF GT$>"Z"ORGT$<" "ANDGT$<>CHR$(13)
     ANDGT$<>CHR$(20)THEN4030
4035 NL=NL+1
4040 IF GT$=CHR$(20)ORGT$=CHR$(13)THEN4100
4045 IFGT$=" "THENPRINT" ";:RETURN
4050 PRINTGT$;:IN$=IN$+GT$
4060 IF NL=LTTHEN4100
4070 GOTO4020
4100 IF GT$<>CHR$(20)THEN4150
4105 IF LEN(IN$)<2THEN4120
4110 PRINT" ■■";:NL=NL-2:IN$=LEFT$(IN$,
     LEN(IN$)-1):GOTO4020
4120 IFLEN(IN$)=0THENNL=NL-1:GOTO4020
4130 PRINT" ■■";:NL=NL-2:IN$="":GOTO4020
4150 IFGT$=CHR$(13)THENPRINT" "
4160 RETURN
5000 DATAADCN6DS65I69K7DL79P61071075
5010 DATAANDN2DS25I29K3DL39P21031035
5020 DATAASLH0AN0ES06K1EQ16
5030 DATABCCJ90,BCSJB0,BEQJF0
5060 DATABITN2CS24
5070 DATABMIJ30,BNEJD0,BPLJ10,BRKG00
5110 DATABVCJ50,BVSJ70,CLCG18,CLDGD8
5150 DATACLIG58,CLVGB8
5170 DATACMPNCDSC5IC9KDDLD9PC10D1QD5
5180 DATACPXNECSE4IE0
5190 DATACPYNCCSC4IC0
5200 DATADECNCESC6KDEQD6
5210 DATADEXGCA,DEYG88
5230 DATAEORN4DS45I49K5DL59P41051055
5240 DATAINCNEESE6KFEQF6
5250 DATAINXGE8,INYGC8
5270 DATAJMPN4CM6C
5280 DATAJSRN20
5290 DATALDANADSA5IA9KBDLB9PA10B1QB5
5300 DATALDXNAESA6IA2LBERB6
5310 DATALDYNACSA4IA0KBCQB4
5320 DATALSRH4AN4ES46K5EQ56
5330 DATANOPGEA
5340 DATAORAN0DS05I09K1DL19P01011Q15
5350 DATAPHAG48,PHPG08,PLAG68,PLPG28
5390 DATAROLH2AN2ES26K3EQ36
5400 DATARORH6AN6ES66K7EQ76
5410 DATARTIG40,RTSG60
5430 DATASBCNEDSE5IE9KFDLF9PE10F1QF5
5440 DATASECG38,SEDGF8,SEIG78
5470 DATASTAN8DS85K9DL99P81091095
5480 DATASTXN8ES86R96
5490 DATASTYN8CS84Q94
5500 DATATAXGAA,TAYGA8,TSXGBA,TXAG8A
5510 DATATXSG9A,TYAG98
6000 AD=0
6005 Q$=LEFT$(OP$,1):IFQ$="$"ORQ$="%"OR
     (ASC(Q$)>64ANDASC(Q$)<91)THEN6030
6010 IFASC(Q$)>47ANDASC(Q$)<58THEN6030
6020 OP$=RIGHT$(OP$,LEN(OP$)-1):GOTO6000
6030 Q$=RIGHT$(OP$,1):Q1=ASC(Q$):IF(Q1
  >47ANDQ1<58)OR(Q1>64ANDQ1<91)THEN6050
6035 IFQ$="+"THEN6050
6040 OP$=LEFT$(OP$,LEN(OP$)-1):GOTO6030
6050 IFRIGHT$(OP$,2)=",X"THENOP$=
     LEFT$(OP$,LEN(OP$)-2)
6052 IFRIGHT$(OP$,2)=",Y"THENOP$=
     LEFT$(OP$,LEN(OP$)-2)
```

```
6053 IFRIGHT$(OP$,1)=")"THENOP$=LEFT$(OP$,LEN(OP$)-1)
6055 IFLEFT$(OP$,1)="$"THENN$=OP$:GOSUB7000:NUMBER=V:GOTO6100
6060 IFLEFT$(OP$,1)="%"THENN$=OP$:GOSUB8000:NUMBER=V:GOTO6100
6070 IFASC(LEFT$(OP$,1))<58THENNUMBER=VAL(OP$):GOTO6100
6075 IFRIGHT$(OP$,1)="+"THENAD=AD+1:OP$=LEFT$(OP$,LEN(OP$)-1):GOTO6075
6080 FORW1=1TOM2:IFS$(W1)=OP$THENNUMBER=V(W1):W1=999
6090 NEXTW1:IFW1=M2+1THENPRINT"?UNDEFINED SYMBOL ERROR":GOTO300
6100 NU=NU+AD:RETURN
7000 IFLEFT$(N$,1)="$"THENN$=RIGHT$(N$,LEN(N$)-1)
7010 V=0:IFLEN(N$)=4THEN7030
7020 N$=LEFT$("0000",4-LEN(N$))+N$
7030 FORR2=1TO4:D$=MID$(N$,R2,1):TV=ASC(D$)-48:IFTV>9THENTV=TV-7
7040 V=TV*16↑(4-R2)+V:NEXTR2:RETURN
8000 IFLEFT$(N$,1)="%"THENN$=RIGHT$(N$,LEN(N$)-1)
8010 V=0:FORZ=LEN(N$)TO1STEP-1:V=V+VAL(MID$(N$,Z,1))*2↑(LEN(N$)-Z):NEXTZ
8020 RETURN
9000 FD=INT(N/4096):N=(N/4096-FD)*4096:SD=INT(N/256):N=(N/256-SD)*256
9010 TD=INT(N/16):N=INT((N/16-TD)*16):R$=MID$(H$,FD+1,1)+MID$(H$,SD+1,1)
9020 R$=R$+MID$(H$,TD+1,1)+MID$(H$,N+1,1):RETURN
10000 IFA$(T)=""THENOC$="":LB$="":GOTO10100
10005 LI(1)=0:LI(2)=0:LI(3)=0:LI=0
10010 FORR2=1TOLEN(A$(T)):IFMID$(A$(T),R2,1)=" "THENLI=LI+1:LI(LI)=R2
10020 NEXTR2:IFLI(3)=0THENLI(3)=R2-1
10030 LB$=LEFT$(A$(T),LI(1)):OC$=MID$(A$(T),LI(1)+1,LI(2)-LI(1))
10040 OP$=RIGHT$(A$(T),LI(3)-LI(2)+1)
10050 IFLB$=" "THENLB$="":GOTO10070
10060 LB$=LEFT$(LB$,LEN(LB$)-1)
10070 OC$=LEFT$(OC$,LEN(OC$)-1)
10080 IFOP$=" "THENOP$="":GOTO10100
10090 OP$=RIGHT$(OP$,LEN(OP$)-1)
10100 RETURN
```