Apple Assembly Line

Volume 1 -- Issue 4 January, 1981

There are, as of Christmas Eve, 179 of you subscribing to the Apple Assembly Line! Last month I wondered if circulation could double, from 85, but we did even better! Also, several stores have decided to carry the AAL for sale like a magazine. We are growing a lot faster than I predicted, and I like it!

In This Issue...

First "Disk of the Quarter"

Every three months I collect onto one disk all the source programs published in AAL for the quarter. QD#1 (for October, November, and December of 1980) is now available, for $15. You can save a lot of typing.

If you would like to help promote the newsletter, here is a nice offer: you sign up four new subscribers, and send me their mailing addresses and money, and I will send you a "Disk of the Quarter" FREE and POSTPAID!

Those Compatible Disassemblers

Bob Zant and Bob Kovacs both report that their new two-pass disassemblers are selling well. Well enough to warrant advertising again! Have you bought a copy yet?

TAB Locations in S-C ASSEMBLER II Version 4.0

For some reason, people are always asking me where the tab stops are kept, because they want to change them. The old version 3.2 manual gives the patch locations for the three tab stops, but they are different in version 4.0. You will find them at:

	column	location
1st tab	14	$140D:0B
2nd tab	18	$1411:0F
3rd tab	27	$1402:18

Note that the value stored in memory is three less than the column number.

How to Move Memory

One of the most common problems in assembly language programming is the problem of moving data from one place in memory to another.

Moving Little Blocks: If you only need to move one or two bytes of data from one place to another in memory, it is easy. You might do it like this:

```
     LDA SOURCE
     STA DEST
     LDA SOURCE+1
     STA DEST+1
```

Or, if the A-register was busy but X and Y were not, you might write:

```
     LDX SOURCE
     LDY SOURCE+1
     STX DEST
     STY DEST+1
```

If you know ahead of time exactly how many bytes you want to move, and exactly where you want it copied from and to, you can write a very fast loop. For example, suppose I know that I want to copy 20 bytes from BUFFER1 into BUFFER2, and that there is no overlap. Then I can write:

```
         LDX #19
 LOOP    LDA BUFFER1,X
         STA BUFFER2,X
         DEX
         BPL LOOP
         ...
```

The loop moves the last byte first, then the next-to-last, and so on until the first byte in BUFFER1 is moved into BUFFER2. If it is important to move them in the opposite direction (first byte first, last byte last), you can change the loop this way:

```
         LDX #0
 LOOP    LDA BUFFER1,X
         STA BUFFER2,X
         INX
         CPX #20
         BCC LOOP
         ...
```

Terminating the loop can be done in various ways. The two examples above do it with a count in the X-register. Another way is to use a data sentinel. For example, the last byte to be moved, and only the last byte, might contain the value $00, or $FF, or anything you choose. Then after moving a byte, you can check to see if the sentinel byte was just moved. If it was, you are finished moving. Here is an example using a sentinel of $00:

```
         LDX #-1
 LOOP    INX
         LDA BUFFER1,X
         STA BUFFER2,X
         BNE LOOP
         ...
```

Pascal Language promoters often recommend the sentinel technique; however, in Assembly Language, you must be very careful if you plan to use it. The sentinel you choose today may become a valid data value tomorrow!

Moving Bigger Blocks: All of the examples so far will only work if the total number of bytes to be moved is less than 256. What if you need to move a larger block?

When I need to move a large block of data from one place to another, I frequently use the MOVE subroutine in the Apple Monitor ROM. It starts at $FE2C, and looks like this:

```
FE2C- B1 3C    MOVE   LDA (A1L),Y  MOVE (A1 TO A2)
FE2E- 91 42           STA (A4L),Y       TO (A4)
FE30- 20 B4 FC        JSR NXTA4
FE33- 90 F7           BCC MOVE
FE35- 60              RTS
```

The subroutine NXTA4 (at $FCB4) increments A4L,A4H ($42,43), which is the destination address. Then it compares A1L,A1H ($3C,3D) to A2L,A2H ($3E,3F); the result of the comparison is left in the Carry Status Bit: Carry is set if A1 is greater than or equal to A2. Finally, the subroutine increments A2L,A2H ($3E,3F).

To use the MOVE subroutine, you have to set the starting address of the block to be copied into $3C,3D; the last address of the block to be copied into $3E,3F; and the starting address of the destination into $42,43. You also need to be sure that the Y-register contains zero before you start. Here is an example:

```
LDY #0            CLEAR Y-REGISTER
LDA #BUFFER1      START ADDRESS OF SOURCE
STA $3C
LDA /BUFFER1
STA $3D
LDA #BUFFER1.END  END ADDRESS OF SOURCE
STA $3E
LDA /BUFFER1.END
STA $3F
LDA #BUFFER2      START ADDRESS OF DESTINATION
STA $42
LDA /BUFFER2
STA $43
JSR $FE2C
...
```

Because it is there, the Monitor MOVE subroutine is handy. But it is not a general subroutine. If the source and destination blocks overlap, you may get funny results. For example, If I try to move the data between $1000 and $10FF up one byte in memory, so that it runs from $1001 to $1100, the MOVE subroutine will not work. Instead, it will copy the contents of $1000 into every location from $1001 through $1100.

The MOVE subroutine is also not very fast. Anyway, it is not as fast as it could be. Steve Wozniak evidently wrote with size in mind (to make it fit in ROM) rather than speed.

The Applesoft ROMs contain several subroutines for moving data around in memory. Here is one used during execution to move the array table up to make room for a new simple variable:

```
                   1000 *---------------------------------
                   1010 *       BLTU -- FROM THE APPLESOFT ROM
                   1020 *       $D393 THROUGH $D3D5
                   1030 *---------------------------------
                   1040 *       ON ENTRY:
                   1050 *           Y,A AND HIGHDS CONTAIN DESTINATION END + 1
                   1060 *           LOWTR CONTAINS LOWEST ADDRESS OF SOURCE
                   1070 *           HIGHTR CONTAINS HIGHEST SOURCE ADDRESS + 1
                   1080 *---------------------------------
                   1090 *       PAGE-ZERO VARIABLE NAMES FROM "THE APPLE ORCHARD"
                   1100 *       VOL. 1, NO. 1, PAGES 12-18.
006D-              1110 STREND .EQ $6D,6E   TOP OF ARRAY STORAGE
0094-              1120 HIGHDS .EQ $94,95   BLTU'S DESTINATION POINTER
0096-              1130 HIGHTR .EQ $96,97   BLTU'S SOURCE END POINTER
009B-              1140 LOWTR  .EQ $9B,9C   BLTU'S SOURCE START POINTER
                   1150 *---------------------------------
D3E3-              1160 REASON .EQ $D3E3    CHECK IF ENOUGH MEMORY
                   1170 *---------------------------------
0800- 20 E3 D3     1180 BLTU    JSR REASON  BE SURE (Y,A) < FRETOP
0803- 85 6D        1190         STA STREND  NEW TOP OF ARRAY STORAGE
0805- 84 6E        1200         STY STREND+1
0807- 38           1210         SEC         COMPUTE # OF BYTES TO BE MOVED
0808- A5 96        1220         LDA HIGHTR
080A- E5 9B        1230         SBC LOWTR
080C- 85 5E        1240         STA $5E     SAVE PARTIAL PAGE AMOUNT
080E- A8           1250         TAY         ALSO IN Y
080F- A5 97        1260         LDA HIGHTR+1
0811- E5 9C        1270         SBC LOWTR+1
0813- AA           1280         TAX         NUMBER OF WHOLE PAGES IN X
0814- E8           1290         INX
0815- 98           1300         TYA         # BYTES IN PARTIAL PAGE
0816- F0 23        1310         BEQ .4      NO PARTIAL PAGE
0818- A5 96        1320         LDA HIGHTR  BACK UP HIGHTR BY PARTIAL PAGE #
081A- 38           1330         SEC
081B- E5 5E        1340         SBC $5E
081D- 85 96        1350         STA HIGHTR
081F- B0 03        1360         BCS .1
0821- C6 97        1370         DEC HIGHTR+1
0823- 38           1380         SEC
0824- A5 94        1390 .1      LDA HIGHDS  BACK UP HIGHDS BY PARTIAL PAGE #
0826- E5 5E        1400         SBC $5E
0828- 85 94        1410         STA HIGHDS
082A- B0 08        1420         BCS .3
082C- C6 95        1430         DEC HIGHDS+1
082E- 90 04        1440         BCC .3       ...ALWAYS
0830- B1 96        1450 .2      LDA (HIGHTR),Y
0832- 91 94        1460         STA (HIGHDS),Y
0834- 88           1470 .3      DEY
0835- D0 F9        1480         BNE .2       LOOP TO END OF THIS 256 BYTES
0837- B1 96        1490         LDA (HIGHTR),Y  MOVE ONE MORE BYTE
0839- 91 94        1500         STA (HIGHDS),Y
083B- C6 97        1510 .4      DEC HIGHTR+1    DOWN TO NEXT BLOCK OF 256
083D- C6 95        1520         DEC HIGHDS+1
083F- CA           1530         DEX          PAGE COUNT
0840- D0 F2        1540         BNE .3
0842- 60           1550         RTS
```

Since this code moves from the end of the block backwards, it will safely move a block up in memory. However, it would not be safe to use with an overlapping range down in memory; it will do the same thing as the Monitor MOVE subroutine.

The Applesoft subroutine is faster than the Monitor subroutine, because the least significant half of the pointer is kept in the Y-register instead of in page-zero of memory. The INY instruction takes only two cycles, whereas an INC instructions takes five. The three cycles saved in moving each byte add up to nearly 25 milliseconds in moving 8K bytes. The extra overhead of setting up the pointers is more than paid for.

Additional time is saved in the termination test. Instead of testing after moving every byte with a LDA, CMP, LDA, SBC sequence, the number of full 256-byte blocks to be moved is put in the X-register; only a DEX instruction once out of every 256 bytes is needed. This saves over 100 milliseconds in moving an 8K block. By putting the incrementing and testing code in line, rather than in a subroutine like NXTA4, we save the JSR and RTS time. This amounts to another 100 milliseconds in moving an 8K block.

<u>A General Move Subroutine</u>: Can we write a subroutine which will move a block of data from one place to another regardless of overlap and direction? Of course! All we have to do is test at the beginning for direction, and choose which method to use accordingly.

Here is a fast subroutine which will move any block of memory anywhere you want. You call it by putting the starting address of the source block in A1L,A1H; the end address of the source in A2L,A2H; and the start address of the destination in A4L,A4H. (This is the same way you set up the Monitor MOVE subroutine.) I wrote it to be used with the control-Y monitor command.

```
                  1000 *---------------------------------
                  1010 *      GENERAL MOVE SUBROUTINE
                  1020 *---------------------------------
                  1030 *      BRUN THE PROGRAM TO SET UP AS CONTROL-Y
                  1040 *      MONITOR ROUTINE
                  1050 *---------------------------------
                  1060 *      USE LIKE MONITOR MOVE SUBROUTINE:
                  1070 *      A1L,A1H -- SOURCE START ADDRESS
                  1080 *      A2L,A2H -- SOURCE END ADDRESS
                  1090 *      A4L,A4H -- DESTINATION START ADDRESS
                  1100 *---------------------------------
0000-             1110 BLOCK.SIZE .EQ $00,01
003C-             1120 A1L     .EQ $3C
003D-             1130 A1H     .EQ $3D
003E-             1140 A2L     .EQ $3E
003F-             1150 A2H     .EQ $3F
0042-             1160 A4L     .EQ $42
0043-             1170 A4H     .EQ $43
03F8-             1180 CONTROL.Y   .EQ $3F8
                  1190 *---------------------------------
                  1200 CONTROL.Y.SETUP
0800- A9 4C       1210         LDA #$4C      JMP OPCODE
0802- 8D F8 03    1220         STA CONTROL.Y
0805- A9 10       1230         LDA #GENERAL.MOVE
0807- 8D F9 03    1240         STA CONTROL.Y+1
080A- A9 08       1250         LDA /GENERAL.MOVE
080C- 8D FA 03    1260         STA CONTROL.Y+2
080F- 60          1270         RTS
                  1280 *---------------------------------
                  1290 GENERAL.MOVE
0810- 48          1300         PHA           SAVE REGISTERS
0811- 98          1310         TYA
0812- 48          1320         PHA
0813- 8A          1330         TXA
0814- 48          1340         PHA
0815- E6 3E       1350         INC A2L       BUMP END ADDRESS ONCE
0817- D0 02       1360         BNE .1
0819- E6 3F       1370         TNC A2H
```

```
081B- 38        1380 .1      SEC              COMPUTE SIZE OF BLOCK
081C- A5 3E     1390         LDA A2L
081E- E5 3C     1400         SBC A1L
0820- 85 00     1410         STA BLOCK.SIZE
0822- A5 3F     1420         LDA A2H
0824- E5 3D     1430         SBC A1H
0826- 85 01     1440         STA BLOCK.SIZE+1
0828- AA        1450         TAX
0829- E8        1460         INX              NUMBER OF BLOCKS TO MOVE
082A- A5 3C     1470         LDA A1L          DETERMINE DIRECTION
082C- C5 42     1480         CMP A4L
082E- A5 3D     1490         LDA A1H
0830- E5 43     1500         SBC A4H
0832- 90 06     1510         BCC .2           A1 < A4
0834- 20 43 08  1520         JSR MOVE.DOWN
0837- 4C 3D 08  1530         JMP .3
083A- 20 63 08  1540 .2      JSR MOVE.UP
083D- 68        1550 .3      PLA              RESTORE REGS
083E- AA        1560         TAX
083F- 68        1570         PLA
0840- A8        1580         TAY
0841- 68        1590         PLA
0842- 60        1600         RTS
                1610 *---------------------------------
                1620 MOVE.DOWN
0843- A0 00     1630         LDY #0
0845- CA        1640         DEX              ANY WHOLE BLOCKS LEFT?
0846- F0 0E     1650         BEQ .2           NO
0848- B1 3C     1660 .1      LDA (A1L),Y      MOVE 256 BYTES
084A- 91 42     1670         STA (A4L),Y
084C- C8        1680         INY
084D- D0 F9     1690         BNE .1
084F- E6 3D     1700         INC A1H          POINT AT NEXT BLOCK
0851- E6 43     1710         INC A4H
0853- CA        1720         DEX              ANY MORE WHOLE BLOCKS?
0854- D0 F2     1730         BNE .1           YES
0856- A6 00     1740 .2      LDX BLOCK.SIZE  ANY EXTRA BYTES IN A SHORT BLOCK?
0858- F0 08     1750         BEQ .4           NONE LEFT
085A- B1 3C     1760 .3      LDA (A1L),Y
085C- 91 42     1770         STA (A4L),Y
085E- C8        1780         INY
085F- CA        1790         DEX
0860- D0 F8     1800         BNE .3
0862- 60        1810 .4      RTS
                1820 *---------------------------------
                1830 MOVE.UP
0863- 18        1840         CLC              COMPUTE DESTINATION END + 1
0864- A5 42     1850         LDA A4L
0866- 65 00     1860         ADC BLOCK.SIZE
0868- 85 42     1870         STA A4L
086A- A5 43     1880         LDA A4H
086C- 65 01     1890         ADC BLOCK.SIZE+1
086E- 85 43     1900         STA A4H
0870- A0 00     1910         LDY #0
0872- F0 0B     1920         BEQ .3            ...ALWAYS
                1930 *---MOVE A WHOLE BLOCK-----------
0874- B1 3E     1940 .1      LDA (A2L),Y      MOVE BYTES 255 THRU 1 IN BLOCK
0876- 91 42     1950         STA (A4L),Y
0878- 88        1960 .2      DEY
0879- D0 F9     1970         BNE .1
087B- B1 3E     1980         LDA (A2L),Y      MOVE LOWEST BYTE IN BLOCK
087D- 91 42     1990         STA (A4L),Y
087F- C6 3F     2000 .3      DEC A2H
0881- C6 43     2010         DEC A4H
0883- CA        2020         DEX              ANY MORE BLOCKS?
0884- D0 F2     2030         BNE .2           YES
                2040 *---MOVE SHORT BLOCK IF ANY-------
0886- A6 00     2050         LDX BLOCK.SIZE
0888- F0 08     2060         BEQ .5           NONE LEFT
088A- 88        2070 .4      DEY
088B- B1 3E     2080         LDA (A2L),Y
088D- 91 42     2090         STA (A4L),Y
088F- CA        2100         DEX
0890- D0 F8     2110         BNE .4
0892- 60        2120 .5      RTS
```

A Computed GOSUB for Applesoft

How many times I have wished for one! I guess I am spoiled from FORTRAN and Apple Integer BASIC. The Computed GOTO is also left out, but I saw that one written up in a recent newsletter. The author said he didn't know how to do the Computed GOSUB, so here it is!

```
                     1000 *--------------------------------
                     1010 *      &GOSUB <EXPRESSION>
                     1020 *--------------------------------
00B0-                1030 TKN.GOSUB  .EQ $B0
                     1040 *--------------------------------
DEC0-                1050 AS.SYNCHR  .EQ $DEC0
D3D6-                1060 AS.MEMCHK  .EQ $D3D6
00B8-                1070 AS.TXTPTR  .EQ $B8,B9
0050-                1080 AS.LINNUM  .EQ $50,51
DD67-                1090 AS.FRMNUM  .EQ $DD67
D941-                1100 AS.GOTO1   .EQ $D941
D7D2-                1110 AS.NEWSTT  .EQ $D7D2
E752-                1120 AS.GETADR  .EQ $E752
                     1130 *--------------------------------
                     1140        .OR $300
                     1150 VARIABLE.GOSUB
0300- A9 B0          1160        LDA #TKN.GOSUB  CHECK IF &GOSUB
0302- 20 C0 DE       1170        JSR AS.SYNCHR
0305- A9 03          1180        LDA #3          CHECK IF ROOM ON STACK
0307- 20 D6 D3       1190        JSR AS.MEMCHK
030A- A5 B9          1200        LDA AS.TXTPTR+1
030C- 48             1210        PHA             STACK TXTPTR
030D- A5 B8          1220        LDA AS.TXTPTR
030F- 48             1230        PHA
0310- A5 51          1240        LDA AS.LINNUM+1
0312- 48             1250        PHA             STACK CURRENT LINE NO.
0313- A5 50          1260        LDA AS.LINNUM
0315- 48             1270        PHA
0316- A9 B0          1280        LDA #TKN.GOSUB  MARK STACK
0318- 48             1290        PHA
0319- 20 67 DD       1300        JSR AS.FRMNUM   EVALUATE FORMULA
031C- 20 52 E7       1310        JSR AS.GETADR   CONVERT TO INTEGER
031F- 20 41 D9       1320        JSR AS.GOTO1    USE GOTO CODE
0322- 4C D2 D7       1330        JMP AS.NEWSTT
```

Lines 1160 and 1170 check the token after the "&" to see if it is "GOSUB"; if not, you will get a big SYNTAX ERROR. Lines 1180 and 1190 check the stack to see if there is room for another GOSUB entry; if not, you get an OUT OF MEMORY error. Lines 1200-1290 push the data on the stack that will be needed to RETURN. Lines 1300 and 1310 compute the value of whatever expression follows the &GOSUB, and turn it into an integer that looks just like a line number. Finally, lines 1320 and 1330 simulate a normal GOTO. That's all there is to it! •

Here is a sample Applesoft program using the new &GOSUB statement:

```
 10 POKE 1013,76: POKE 1014,0: POKE 1015,3   (set up &-vector)
 20 INPUT X                   (read a subroutine number 1-4)
 30 &  GOSUB X*100            (GOSUB to 100, 200, 300, or 400)
 40 GOTO 20
100 PRINT 100: RETURN         (four silly subroutines)
200 PRINT 200: RETURN
300 PRINT 300: RETURN
400 PRINT 400: RETURN
```

Putting COPY in S-C ASSEMBLER II........................Lee Meador

I just looked at the first AAL Disk of the Quarter. The first item of business was to incorporate the changes into my copy of the assembler.

The lower-case mod and the .DA mod went just as described in AAL. However, when it came to the COPY stuff, I found that I wasn't really happy to load it at $800 and hope it didn't get clobbered. Here's what I did....

I changed the origin of the COPY program to $25A0 (since I already have a special printer driver at $2500.259F). The COPY program runs from $25A0 through $266F, so I changed the symbol table origin by typing "$1011:27". This sets the bottom of the symbol table at $2700. I put a ".TF B.SC COPY MODS" line in, to write the object on a binary file.

After assembling, I BLOADed the file B.SC COPY MODS into memory. Then I could have plugged in the USR vector like Bob suggested, but I wanted a real "COPY" command. Therefore I searched around in the assembler until I found the command table. I put the letters "COP" and the program address over the top of the tape SAVE command entry, by typing "$2746:43 4F 50 9F 25". I felt the loss of the tape SAVE command was worth it, to get a real COPY command.

Now the command "COPY 1000,1050,2500" will copy lines 1000 through 1050 into the place right before line 2500. The USR command is still intact and I'm ready for some more changes!

EDIT Command for S-C ASSEMBLER II....................Mike Laumer

At last! Owners of the S-C ASSEMBLER II Version 4.0 can now have the power of an EDIT command similar in function to the popular "Program Line Editor" (PLE) by Neil Konzen. (PLE only works with Integer BASIC and Applesoft, although some wizards have figured out how to interface it with the S-C Assembler.) The program presented here will patch itself into Version 4.0 to turn the "USR" command into an EDIT command.

Several weeks ago Bob Sander-Cederlof contacted me about some contract programming, to help out on various projects he had in mind. So I suggested lunch, and we met to discuss some of his projects. I was amazed at the list (as long as my arm!) of the ideas for just one of his products, the S-C Assembler II. (If you liked version 3.2, as I did; if you are thrilled with version 4.0, as I am; then version 5.0 will ....) So I picked out a couple that would be fairly straightforward and would let me pick up the internal structure of the assembler gradually.

After signing a non-disclosure agreement, I obtained the source files and made a listing of the assembler. Lucky for me I have a brand new Epson MX-80 printer! I think it is the greatest!

Thursday, I made the listing. Friday I looked at the listing. Friday night I began writing code for the EDIT command. Saturday from 9AM till 1AM I wrote more code, read it through, and rewrote it. Sunday morning I typed it into my Apple and eliminated the assembly errors (typos). And by 11AM, with the exception of two trivial bugs, I had it working! I nearly fell out of my chair! A 377-line program worked on the first run!

After you type in the program, assemble it, and BRUN it, the USR command will work as an edit command. If you type the command USR with no line number, it will do nothing. If you type USR and one line number, it will list the line on the bottom of the screen and set you up to edit it. If you type USR and two line numbers, separated by a comma, all the lines in the range will be set up to edit, one at a time.

How to Use EDIT: Twelve editing functions are available, and you may see fit to add some more. Each function is selected by typing a control character. If you type a normal character, it will write over the top of the characters already in the line. The control characters and their associated functions are:

Control-B Move to beginning of line.
Control-D Delete character beneath cursor.
Control-E Move to end of line.
Control-F Find a character; the character searched for is typed after the control-F; repeatedly typing the same character will keep looking for successive occurrences.
Control-H Backspace (left arrow).
Control-I Insert characters before current cursor position.
Control-M (Return) Stop editing the line, and submit it to the line input routine in the assembler.

Control-O Same as control-I, except next character may be any control character.
Control-Q Same as control-M, but line beyond cursor is truncated.
Control-T Skip to next tab stop.
Control-U (Right arrow) Move cursor forward.
Control-X Kill edit, does not submit line.

How EDIT Works: When you BRUN the file B.EDIT (after assembly has written the object code there!), the code in lines 1360-1530 is executed. This patches the USR command vector to jump to EDIT (line 1720), and makes some patches inside the assembler. The patches only work for version 4.0! Their purpose is to make the code which processes a source line into a subroutine.

Lines 1540-1620 are part of the patch code for the source line processing subroutine.

Lines 1720-2040 determines the number of line numbers typed, and searches for them in the source program. Then E.LIST is called for each line to be edited.

Lines 2050-2360 list the source line on the screen and also stuff it into the line input buffer at $0200. All changes will be made in the buffer, not in the source program.

Lines 2370-2530 read a key from the keyboard and search the command table. If the key is found in the table, then DOIT is called to execute the command. If the key is not found, I assume it is a type-over character. The command table search is actually performed by a rather neat subroutine inside the assembler, called SEARCH.

Lines 2540-2690 process a type-over character, in which the key just typed replaces the character under the cursor. Then the modified line in the buffer is re-displayed on the screen.

Lines 2700-2750 position the cursor at the beginning of line 19, where the source line will be listed.

Lines 2760-2900 display the line from the buffer. Display always starts at line 19 on the screen. Control characters are shown in inverse video.

Lines 2910-4090 process the various commands. Each processor is written as a subroutine. The RTS returns to line 2520; at this point the Carry Status is used to flag whether or not to re-display the source line from the buffer.

Lines 4100-4260 read a character from the keyboard by calling on the monitor RDKEY subroutine. The internal line buffer index is also converted to cursor line and column position on the screen.

```
                  1000 *-------------------------------
                  1010 *  EDIT COMMAND FOR S-C ASSEMBLER II VERSION 4.0
                  1020 *
                  1030 *      WRITTEN BY MIKE LAUMER
                  1040 *                 DECEMBER 6, 1980
                  1050 *-------------------------------
                  1060        .OR $0800
                  1070        .TF B.EDIT2
                  1080 *-------------------------------
                  1090 *  SYSTEM EQUATES
                  1100 *-------------------------------
FDED-             1110 MON.COUT   .EQ $FDED
FF3A-             1120 MON.BELL   .EQ $FF3A
FD0C-             1130 MON.RDKEY  .EQ $FD0C
FC42-             1140 MON.CLREOP .EQ $FC42
FC22-             1150 MON.VTAB   .EQ $FC22
0024-             1160 CH     .EQ $24
0025-             1170 CV     .EQ $25
03D0-             1180 DOS.REENTRY .EQ $03D0
                  1190 *-------------------------------
                  1200 *  ASSEMBLER EQUATES
                  1210 *-------------------------------
1026-             1220 GNL    .EQ $1026
1063-             1230 NML    .EQ $1063
1779-             1240 PLNO   .EQ $1779
12C5-             1250 GNB    .EQ $12C5
1874-             1260 DOIT   .EQ $1874
164B-             1270 SEARCH .EQ $164B
14F6-             1280 SERTXT .EQ $14F6
14FE-             1290 SERNXT .EQ $14FE
12AF-             1300 NTKN   .EQ $12AF
003A-             1310 A0L    .EQ $3A,3B
003C-             1320 A1L    .EQ $3C,3D
00DD-             1330 SRCP   .EQ $DD,DE
0200-             1340 WBUF   .EQ $0200
00D3-             1350 CURRENT.LINE.NUMBER .EQ $D3,D4
                  1360 *-------------------------------
                  1370 *  ENTRY POINT FOR BRUN.  ACTIVATES
                  1380 *  THE USR ASSEMBLER COMMAND.
                  1390 *-------------------------------
0800- A9 3C       1400 ENTRY  LDA #EDIT
0802- 8D 07 10    1410        STA $1007     PATCH ASM USR COMMAND
0805- A9 08       1420        LDA /EDIT
0807- 8D 08 10    1430        STA $1008
080A- A9 60       1440        LDA #$60      PATCH NML TO MAKE IT
080C- 8D 25 11    1450        STA $1125         A SUBROUTINE
080F- A9 4C       1460        LDA #$4C
0811- 8D 63 10    1470        STA NML
0814- 8D 78 10    1480        STA $1078
0817- A9 24       1490        LDA #NEW.NML
0819- 8D 64 10    1500        STA NML+1
081C- A9 08       1510        LDA /NEW.NML
081E- 8D 65 10    1520        STA NML+2
0821- 4C D0 03    1530        JMP DOS.REENTRY
                  1540 *-------------------------------
                  1550 *  PATCH ROUTINES FOR ASSEMBLER
                  1560 *-------------------------------
0824- 20 2A 08    1570 NEW.NML JSR MY.NML
0827- 4C 26 10    1580        JMP GNL
082A- A0 00       1590 MY.NML LDY #0
082C- 20 8D 12    1600        JSR $128D
082F- 20 4A 11    1610        JSR $114A
0832- 4C 66 10    1620        JMP $1066
                  1630 *-------------------------------
                  1640 *  LOCAL VARIABLES FOR EDIT COMMAND
                  1650 *-------------------------------
0835- 00 00       1660 NEXT   .DA 0
0837- 00 00       1670 END    .DA 0
0839- 00          1680 CHAR   .DA #0
083A- 00          1690 EDPTR  .DA #0
083B- 00          1700 FKEY   .DA #0
                  1710 *-------------------------------
083C- CA          1720 EDIT   DEX
083D- CA          1730        DEX
083E- 30 41       1740        BMI .2        NO ARGUMENTS
0840- F0 40       1750        BEQ .4        1 ARGUMENT
0842- 20 6E 08    1760        JSR .3        2 ARGUMENTS
0845- A2 3C       1770        LDX #A1L      FIND END PTR
0847- 20 FE 14    1780        JSR SERNXT
```

```
084A- A5 E6     1790          LDA $E6
084C- 8D 37 08  1800          STA END
084F- A5 E7     1810          LDA $E7
0851- 8D 38 08  1820          STA END+1
0854- AD 36 08  1830 .1       LDA NEXT+1
0857- 85 DE     1840          STA SRCP+1
0859- 48        1850          PHA
085A- AD 35 08  1860          LDA NEXT
085D- 85 DD     1870          STA SRCP
085F- CD 37 08  1880          CMP END
0862- 68        1890          PLA
0863- ED 38 08  1900          SBC END+1     PAST END LINE?
0866- B0 19     1910          BCS .2        YES, EXIT
0868- 20 87 08  1920          JSR E.LIST    NO, LIST AND EDIT
086B- 4C 54 08  1930          JMP .1        TRY FOR NEXT LINE
086E- A2 3A     1940 .3       LDX #A0L      FIND START PTR
0870- 20 F6 14  1950          JSR SERTXT
0873- A5 E4     1960          LDA $E4
0875- 85 DD     1970          STA SRCP
0877- 8D 35 08  1980          STA NEXT      SAVE NEXT LINE ADRS
087A- A5 E5     1990          LDA $E5
087C- 85 DE     2000          STA SRCP+1
087E- 8D 36 08  2010          STA NEXT+1
0881- 60        2020 .2       RTS
0882- 20 6E 08  2030 .4       JSR .3        SEARCH FOR LINE
0885- 90 FA     2040          BCC .2        NOT FOUND EXIT
0887- 20 23 09  2050 E.LIST   JSR E.POSN    POSITION FOR EDIT
088A- 20 42 FC  2060          JSR MON.CLREOP  PREPARE DISPLAY
088D- 20 C5 12  2070          JSR GNB       GET LINE SIZE
0890- 18        2080          CLC
0891- 6D 35 08  2090          ADC NEXT      COMPUTE NEXT LINE ADRS
0894- 8D 35 08  2100          STA NEXT
0897- 98        2110          TYA
0898- 6D 36 08  2120          ADC NEXT+1
089B- 8D 36 08  2130          STA NEXT+1
089E- 20 C5 12  2140          JSR GNB       GET LINE NUMBER FOR DISPLAY
08A1- 85 D3     2150          STA CURRENT.LINE.NUMBER
08A3- 20 C5 12  2160          JSR GNB
08A6- 85 D4     2170          STA CURRENT.LINE.NUMBER+1
08A8- 38        2180          SEC
08A9- 66 F8     2190          ROR $F8       STUFF WBUF FLAG
08AB- 20 79 17  2200          JSR PLNO
08AE- 46 F8     2210          LSR $F8       TURN OFF FLAG
08B0- A9 20     2220          LDA #$20      SPACE AFTER LINE #
08B2- A2 00     2230          LDX #0
08B4- 8E 3A 08  2240 .1       STX EDPTR
08B7- 09 80     2250          ORA #$80      FORCE VIDEO BIT
08B9- 9D 04 02  2260          STA WBUF+4,X  STORE INTO INPUT BUFFER
08BC- C9 A0     2270          CMP #$A0      TEST FOR CONTROL CHAR
08BE- B0 02     2280          BCS .2        OK, IF NOT
08C0- 29 7F     2290          AND #$7F      OUTPUT INVERSE ALPHA
08C2- 20 ED FD  2300 .2       JSR MON.COUT  PRINT CHAR
08C5- 20 AF 12  2310          JSR NTKN      GET NEXT TOKEN
08C8- AE 3A 08  2320          LDX EDPTR
08CB- E8        2330          INX
08CC- C9 00     2340          CMP #0        END TOKEN?
08CE- D0 E4     2350          BNE .1        NO, PRINT IT
08D0- 9D 04 02  2360          STA WBUF+4,X  YES, PUT IT IN TOO
08D3- A2 00     2370 E.LINE   LDX #0
08D5- 8E 3A 08  2380 E.0      STX EDPTR
08D8- 20 2D 0A  2390 E.1      JSR E.INPUT   GET INPUT CHAR
08DB- A9 4C     2400 E.2      LDA #EDTB
08DD- 85 02     2410          STA $2
08DF- A9 0A     2420          LDA /EDTB
08E1- 85 03     2430          STA $3
08E3- A9 39     2440          LDA #CHAR
08E5- 85 12     2450          STA $12
08E7- A9 08     2460          LDA /CHAR
08E9- 85 13     2470          STA $13
08EB- 20 4B 16  2480          JSR SEARCH    SEARCH EDIT COMMAND TABLE
08EE- D0 0A     2490          BNE .2        NOT IN TABLE
08F0- AE 3A 08  2500          LDX EDPTR
08F3- 20 74 18  2510          JSR DOIT      EXECUTE COMMAND ROUTINE
08F6- 90 DD     2520          BCC E.0       NO DISPLAY ON RETURN
08F8- B0 23     2530          BCS .5        DISPLAY ON RETURN
08FA- AE 3A 08  2540 .2       LDX EDPTR     MUST BE TYPE OVER
08FD- AD 39 08  2550          LDA CHAR
0900- C9 A0     2560          CMP #$A0
0902- B0 06     2570          BCS .4
```

```
0904- 20 3A FF 2580 .3      JSR MON.BELL ERR IF CONTROL KEY
0907- 4C D8 08 2590         JMP E.1
090A- BD 05 02 2600 .4      LDA WBUF+5,X SEE IF END OF LINE
090D- D0 03    2610         BNE .6       TYPE OVER IF NOT
090F- 9D 06 02 2620         STA WBUF+6,X SHIFT OVER END OF LINE
0912- AD 39 08 2630 .6      LDA CHAR     STUFF CHAR INTO BUFFER
0915- 9D 05 02 2640         STA WBUF+5,X
0918- E0 F9    2650         CPX #256-5-2 TEST BUFFER SIZE
091A- F0 01    2660         BEQ .5       TYPE OVER LAST CHAR IN BUFFER
091C- E8       2670         INX          INSTEAD OF BUFFER END
091D- 20 2E 09 2680 .5      JSR E.DISP   DISPLAY LINE
0920- 4C D5 08 2690         JMP E.0      GET NEXT EDIT COMMAND
               2700 *-------------------------------
0923- A9 13    2710 E.POSN LDA #19      POSITION TO LINE 19,
0925- 85 25    2720         STA CV
0927- A9 00    2730         LDA #0       COLUMN 0
0929- 85 24    2740         STA CH
092B- 4C 22 FC 2750         JMP MON.VTAB
               2760 *-------------------------------
092E- 8E 3A 08 2770 E.DISP STX EDPTR
0931- 20 23 09 2780         JSR E.POSN   POSITION DISPLAY
0934- A2 FF    2790         LDX #$FF
0936- E8       2800 .1      INX
0937- BD 00 02 2810         LDA WBUF,X   GET BUFFER CHAR
093A- F0 0C    2820         BEQ .3       END OF BUFFER
093C- C9 A0    2830         CMP #$A0     CONTROL CHAR?
093E- B0 02    2840         BCS .2       NO
0940- 29 7F    2850         AND #$7F     PRINT INVERSE ALPHA
0942- 20 ED FD 2860 .2      JSR MON.COUT PRINT CHAR
0945- 4C 36 09 2870         JMP .1       NEXT CHAR
0948- 20 42 FC 2880 .3      JSR MON.CLREOP CLEAN ANY REMAINING SCREEN
094B- AE 3A 08 2890         LDX EDPTR
094E- 60       2900         RTS
               2910 *-------------------------------
094F- A2 00    2920 E.BEG  LDX #0   SET CURSOR TO BEGINNING OF LINE
0951- 18       2930         CLC
0952- 60       2940         RTS
               2950 *-------------------------------
0953- BD 05 02 2960 E.DEL  LDA WBUF+5,X IS THIS THEN END OF
0956- F0 0C    2970         BEQ .2
0958- E8       2980 .1      INX
0959- BD 05 02 2990         LDA WBUF+5,X SHIFT TO LOWER MEMORY
095C- 9D 04 02 3000         STA WBUF+4,X TO DELETE CHAR
095F- D0 F7    3010         BNE .1
0961- AE 3A 08 3020         LDX EDPTR
0964- 38       3030 .2      SEC          RETURN WITH DISPLAY
0965- 60       3040         RTS
               3050 *-------------------------------
0966- BD 05 02 3060 E.END  LDA WBUF+5,X END OF BUFFER?
0969- F0 03    3070         BEQ .1       YES
096B- E8       3080         INX          NO
096C- D0 F8    3090         BNE E.END    TRY END AGAIN
096E- 18       3100 .1      CLC          RETURN NO DISPLAY
096F- 60       3110         RTS
               3120 *-------------------------------
0970- BD 05 02 3130 E.FIND LDA WBUF+5,X END OF BUFFER?
0973- D0 08    3140         BNE .2       NO
0975- 8D 3B 08 3150 .1      STA FKEY     YES SO ERR
0978- 20 3A FF 3160         JSR MON.BELL RING BELL
097B- 18       3170         CLC          RETURN NO DISPLAY
097C- 60       3180         RTS
097D- 20 2D 0A 3190 .2      JSR E.INPUT  GET 1 CHAR
0980- 8D 3B 08 3200         STA FKEY     SAVE KEY TO LOCATE
0983- E8       3210 .3      INX
0984- BD 05 02 3220         LDA WBUF+5,X TEST BUFFER
0987- F0 EC    3230         BEQ .1       END OF BUFFER
0989- CD 3B 08 3240         CMP FKEY     NO, SEE IF KEY
098C- D0 F5    3250         BNE .3       NO, GO FORWARD
098E- 20 2D 0A 3260         JSR E.INPUT  TRY ANOTHER KEY
0991- CD 3B 08 3270         CMP FKEY     SAME CHAR?
0994- F0 ED    3280         BEQ .3       YES, SEARCH AGAIN
0996- 68       3290         PLA
0997- 68       3300         PLA
0998- 8E 3A 08 3310         STX EDPTR    NO, EXIT POINTING HERE
099B- 4C DB 08 3320         JMP E.2
```

```
                3330 *--------------------------------
099E- 8A        3340 E.BKSP TXA           AT BEGINNING?
099F- F0 01     3350        BEQ .1        YES, STAY THERE
09A1- CA        3360        DEX           BACKUP
09A2- 18        3370 .1     CLC           RETURN NO DISPLAY
09A3- 60        3380        RTS
                3390 *--------------------------------
09A4- 20 2D 0A  3400 E.OVR  JSR E.INPUT   READ CHAR
09A7- 4C B1 09  3410        JMP E.INS1    SKIP CONTROL CHECK
                3420 *--------------------------------
09AA- 20 2D 0A  3430 E.INS  JSR E.INPUT   READ CHAR
09AD- C9 A0     3440        CMP #$A0      CONTROL CHAR POPS USER OUT
09AF- 90 24     3450        BCC E.INS2    OF INSERT
09B1- E0 F9     3460 E.INS1 CPX #256-5-2 END OF BLOCK
09B3- F0 01     3470        BEQ .1        YES STAY THERE
09B5- E8        3480        INX
09B6- 8E 3A 08  3490 .1     STX EDPTR
09B9- 48        3500 .2     PHA           CHAR TO INSERT
09BA- BD 04 02  3510        LDA WBUF+4,X SAVE CHAR TO MOVE
09BD- A8        3520        TAY
09BE- 68        3530        PLA           GET CHAR TO INSERT
09BF- 9D 04 02  3540        STA WBUF+4,X PUT OVER SAVED CHAR
09C2- E8        3550        INX
09C3- 98        3560        TYA           INSERT SAVED CHAR
09C4- D0 F3     3570        BNE .2        IF NOT BUFFER END
09C6- 9D 04 02  3580        STA WBUF+4,X STUFF END CODE
09C9- 8D FA 02  3590        STA WBUF+256-5-1  INSURE A  END CODE
09CC- AE 3A 08  3600        LDX EDPTR
09CF- 20 2E 09  3610        JSR E.DISP    DISPLAY LINE
09D2- 4C AA 09  3620        JMP E.INS     GET NEXT INSERT CHAR
09D5- 68        3630 E.INS2 PLA           SEND CHAR TO
09D6- 68        3640        PLA           COMMAND SEARCH
09D7- AE 3A 08  3650        LDX EDPTR
                3660 *--------------------------------
09DA- 4C DB 08  3670        JMP E.2
09DD- A9 00     3680 E.RETQ LDA #0        CLEAR REST OF LINE
09DF- 9D 05 02  3690        STA WBUF+5,X
09E2- 20 2E 09  3700        JSR E.DISP    DISPLAY LINE
09E5- A2 FF     3710 E.RET  LDX #$FF      SUBMIT LINE TO ASSEMBLER
09E7- E8        3720 .1     INX           COMPUTE LINE SIZE
09E8- BD 00 02  3730        LDA WBUF,X
09EB- D0 FA     3740        BNE .1
09ED- CA        3750        DEX
09EE- 86 E1     3760 .2     STX $E1       SAVE SIZE
09F0- 68        3770        PLA
09F1- 68        3780        PLA
09F2- 4C 2A 08  3790        JMP MY.NML    SUBMIT THE LINE
                3800 *--------------------------------
09F5- E0 14     3810 E.TAB  CPX #20       < COL 20?
09F7- B0 0E     3820        BCS .1        NO
09F9- BD 05 02  3830        LDA WBUF+5,X END OF BUFFER?
09FC- F0 09     3840        BEQ .1        YES
09FE- E8        3850        INX           MOVE FORWARD
09FF- E0 07     3860        CPX #7        TAB MATCH?
0A01- F0 04     3870        BEQ .1
0A03- E0 0B     3880        CPX #11       TAB MATCH?
0A05- D0 EE     3890        BNE E.TAB
0A07- 18        3900 .1     CLC           RETURN WITHOUT DISPLAY
0A08- 60        3910        RTS
                3920 *--------------------------------
0A09- BD 05 02  3930 E.RIT  LDA WBUF+5,X END OF BUFFER
0A0C- D0 0C     3940        BNE .1        NO
0A0E- 9D 06 02  3950        STA WBUF+6,X
0A11- A9 A0     3960        LDA #$A0      PUT A BLANK
0A13- 9D 05 02  3970        STA WBUF+5,X TO EXTEND LINE
0A16- E0 F9     3980        CPX #256-5-2
0A18- F0 01     3990        BEQ .2
0A1A- E8        4000 .1     INX           MOVE AHEAD
0A1B- 18        4010 .2     CLC           RETURN NO DISPLAY
0A1C- 60        4020        RTS
                4030 *--------------------------------
0A1D- A9 DC     4040 E.ABORT LDA #$DC     OUTPUT BACKSLASH
0A1F- 8D 05 02  4050        STA WBUF+5
0A22- A9 00     4060        LDA #0
0A24- 8D 06 02  4070        STA WBUF+6
0A27- 20 2E 09  4080        JSR E.DISP    SHOW CANCEL
0A2A- 4C 26 10  4090        JMP GNL       GET NEXT COMMAND
```

```
               4100 *--------------------------------
0A2D- A9 13    4110 E.INPUT LDA #19
0A2F- 85 25    4120         STA CV
0A31- 8A       4130         TXA          POSITION TO CURSOR
0A32- 18       4140         CLC
0A33- 69 05    4150         ADC #5
0A35- C9 28    4160 .1      CMP #40      THIS LINE?
0A37- 90 07    4170         BCC .2       YES
0A39- 38       4180         SEC
0A3A- E9 28    4190         SBC #40
0A3C- E6 25    4200         INC CV       ON NEXT LINE
0A3E- D0 F5    4210         BNE .1
0A40- 85 24    4220 .2      STA CH
0A42- 20 22 FC 4230         JSR MON.VTAB SET BASL
0A45- 20 0C FD 4240         JSR MON.RDKEY INPUT A CHAR
0A48- 8D 39 08 4250         STA CHAR
0A4B- 60       4260         RTS
               4270 *--------------------------------
               4280 *  COMMAND TABLE
               4290 *--------------------------------
0A4C- 03 01    4300 EDTB    .DA #3,#1    ITEM SIZE, KEY SIZE
0A4E- 82 4E 09 4310  .DA #$82,E.BEG-1    ^B
0A51- 84 52 09 4320  .DA #$84,E.DEL-1    ^D
0A54- 85 65 09 4330  .DA #$85,E.END-1    ^E
0A57- 86 6F 09 4340  .DA #$86,E.FIND-1   ^F
0A5A- 88 9D 09 4350  .DA #$88,E.BKSP-1   ^H
0A5D- 89 A9 09 4360  .DA #$89,E.INS-1    ^I
0A60- 8D E4 09 4370  .DA #$8D,E.RET-1    ^M
0A63- 8F A3 09 4380  .DA #$8F,E.OVR-1    ^O
0A66- 91 DC 09 4390  .DA #$91,E.RETQ-1   ^Q
0A69- 94 F4 09 4400  .DA #$94,E.TAB-1    ^T
0A6C- 95 08 0A 4410  .DA #$95,E.RIT-1    ^U
0A6F- 98 1C 0A 4420  .DA #$98,E.ABORT-1 ^X
0A72- 00       4430  .DA #0
```

Lines 4270 through the end are the command table. The first line defines the entry size and key size for the SEARCH subroutine: 3 bytes per entry, with a one byte key at the front of each entry. The remaining two bytes of each entry are the starting-address-minus-one of the command processor routine. A final $00 byte terminates the table.

Warning! I have used the patch for Bob's assembler which allows a list of .DA items! Lines 4270-4420 require this patch to be installed. You can read about the patch in Apple Assembly Line for December, 1980, on page 9. If you have not installed the patch, then lines 4270-4420 need to be re-written with each .DA item on a separate source line.

Well, you better get typing on that Apple, I know this is one routine you can't wait to key in. I know I couldn't wait to create it! Or, if you can wait, you can get the source on the next Disk of the Quarter from Bob.

Apple Assembly Line is published monthly by S-C SOFTWARE, P. O. Box 5537, Richardson, TX 75080. Subscription rate is $12/year, in the U.S.A., Canada, and Mexico. Other countries add $6/year for extra postage. All material herein is copyrighted by S-C SOFTWARE, all rights reserved. Unless otherwise indicated, all material herein is authored by Bob Sander-Cederlof. (Apple is a registered trademark of Apple Computer, Inc.)