S-C Assembler II Disk Version 4.0

for the Apple II(TM) and Apple II PIus(TM)

Apple II and Apple II Plus are trademarks of Apple Computer, Inc.

S-C Software
P.O. Box 280300
Dallas, Texas 75228
(214) 324-2050

Upgrade Kit

The disk included with this package contains DOS 3.2.1, in 13-sector format. It is bootable by both 13-sector and 16-sector controllers.

The disk is not protected, and you are free to make as many copies as you like FOR YOUR OWN PERSONAL USE. Since it is a 13-sector disk, you must use a 13-sector copy program to make copies. (Copies made from the original disk will no longer be bootable by a 16-sector controller.)

You can make a 16-sector version by INITializing a blank 16-sector disk, and then using MUFFIN to copy every file. (MUFFIN is a program on your DOS 3.3 System Master Diskette.)

TABLE OF CONTENTS

S-C SOFTWARE
P.O. Box 280300
Dallas, TX 75228
(214) 324-2050

Version 4.0 of S-C ASSEMBLER II contains many new features which make it easier to use, faster, and more powerful.

Now one integrated version runs in any Apple II (with either Integer BASIC or Applesoft ROMs, either old or Autostart monitor ROM). It is compatible with the Apple Language System also.

The S-C ASSEMBLER still only requires 24K 6f RAM and one disk drive to operate. If more memory is available, it will be used. Very large programs can now be developed, using the "INCLUDE" and "TARGET FILE" capabilities. These allow assembly of multiple source files, and direct storage of object code on binary files.

Sample programs on the disk to illustrate new features as well as being important utilities. They include:

1. Sample printer driver, for printers driven through the game port.
2. Shrink program, to compress blanks in old source programs
3. Binary string search, to search memory for a binary pattern. Includes a "wild card" capability.

Blanks are now compressed in source files to conserve memory and disk space. The compression algorithm replaces any string of consecutive blanks with a single coded byte. Old source files are still compatible, but the SHRINK program which is on the disk will convert them if you wish. The savings due to compression usually amounts to about 20 per cent.

The Assembler now uses a more memory-efficient method of storing the symbol table, with variable length entries. The symbol table is maintained in alphabetical order, using a high-speed hashing scheme. The symbol table is maintained in memory until a new assembly is begun or the NEW command is typed. This allows the MGO and VAL commands to be more effective.

Assembler error messages are now printed with two bells, to call attention to their presence. After an assembly error, the offending line is listed automatically, in a position for easy editing.

SUMMARY OF NEW FEATURES

New Editing Features:

* **Ability to append source programs from disk or tape**
* **Automatic line numbering**
* **Parameterized RENUMBER**
* **Memory usage display**
* **Escape IJKM with or without Autostart ROM**
* **Tabs set up for 6-char labels**
* **Star-dash line automatic generation**
* **No conflict with D. C. Hayes**

New Assembly Features:

* **Multiple source files, using .IN directive**
* **Object code directly to disk file, using .TF directive**
* **Listing on and off using .LIST OFF and ·LIST ON**
* **·PG to issue form feeds during listing**
* **.BS to reserve a block of storage**
* **Tremendous speed increase**
* **Labels up to 32 characters**
* **Labels may include period**
* **Local labels**
* **After assembly error, the bad line is listed**
* **Value of .EQ and address of .BS are printed on listing**
* **Assembler and DOS memory is protected during assembly**
* **ASCII literals in address expressions**
* **Symbol table printed in alphabetical order**

EDITING FEATURES

S-C ASSEMBLER II already has the reputation of being one of the easiest editor/assemblers to use. The new features Version 4.0 provides make it even better!

Automatic Line Numbering:

Perhaps the most common request I have received from owners of the previous versions of S-C ASSEMBLER II is for some method of automatic line numbering. Version 4.0 includes a convenient and powerful method.

Any time the cursor is at the beginning of a line, typing a control-I will cause the next line number to be generated. Immediately after loading, the "next line number" will be 1000. The number will be displayed as four digits and a trailing blank. The cursor will be in position for the first character of a label, or the asterisk for a comment line.

The "next line number" is always the value of the previously entered line number plus the current "increment". The increment is normally 10, but you can set it to any reasonable value with the INCREMENT command.

If you type the control-I in any other position than the beginning of a line, it will cause a tab to the next tab stop.

TAB Stops:

The standard tab stops have been changed to allow for a six-character label before the opcode. Of course, you may use any length label from 1 to 32 characters, followed by a blank and an opcode; but the use of the tab stops make for nicer looking programs. (Longer labels look nicer when left on a line by themselves, if possible.)

If control-I is typed after the last tab stop, a single space will be generated. If control-I is typed at the beginning of a line, rather than a normal tab of all spaces, the next line number and a space will be generated.

Some printer interface cards use control-I for setting various modes. This conflicts with the use of control-I for a tab character. If you wish to change the tabbing character, you may do so. It is stored at location $100F. An alternative is to change the printer interface control character, which is usually stored at $06F8+slot#.

Cursor Controls:

Some previous versions of S-C ASSEMBLER II used control-A through control-F as aliases for the escape-A through escape-F codes. This feature has been replaced by the new Apple standard cursor movement controls: escape-I, -J, -K, and -M. whether or not you have the Autostart ROM, you can use these codes when you are in version 4.0.

Escape-L:

Version 4.0 uses the escape-L in two ways, depending on the position in the input line.

If you are at the beginning of a line, escape-L will cause the first six characters of the line to be changed to " LOAD ". Then the rest of line will be read off the screen, and issued as a command. This mimics a function in the very popular "Program Line Editor", by Neil Konzen. The purpose is to load a source file from the disk, after displaying a CATALOG.

If you are at the position in which a comment or label would begin, escape-L will generate an asterisk and a line of dashes across the screen. This is a very commonly used line in setting off blocks of comments. You will find many examples of its use in the sample programs on the version 4.0 disk.

If the Escape-L is typed in any other position of the input line, it is ignored.

There are seven new commands which have been added to S-C ASSEMBLER II. The JOIN command has been replaced by the new HIDE and MERGE commands. The RENUMBER command has been modified to offer several new options. The PRT command has been changed to make it more generally usable.

HIDE	Use with LOAD and MERGE to join a source program to one already in memory.
INCREMENT #	Set the increment for automatic line numbering.
MEMORY	Display the beginning and ending memory addresses of the source program and the symbol table.
MERGE	Use with HIDE and LOAD to join a source program to one already in memory.
RENUMBER	Renumber all lines of the source program: number the first line 1000, and use an increment of 10.
RENUMBER #	Renumber all lines of the source program: number the first line #, and use an increment of 10.
RENUMBER #1,#2	Renumber all lines of the source -program: number the first line #1, and use an increment of *2.
RENUMBER #1,#2,#3	Renumber from line #3 through the end of the source program: number line #3 as #1, and use an increment of #2.
RESTORE	Use after an aborted assembly if within an "included" source file (prompt is "I:") to go back to the root source program.
USR	User defined command.
VAL expression	Print the value of the expression

HIDE and MERGE Commands:

These two commands, when used with the LOAD command, allow you to join a program from disk or tape to a program that is already in memory. (This replaces the tape JOIN command in previous versions of S-C ASSEMBLER II.)

HIDE temporarily changes the HIMEM pointer so that it appears as if there were no source program in memory. To remind you that you are HIDE-mg, the prompt symbol changes to "H:". After HIDE-ing a program, you can LOAD another one from disk or tape. Then you type MERGE to join the two programs together.

After this sequence of commands, the program which was already in memory will follow after the program just LOADed. If the line numbers are not already as you wish them to be, you can use RENUMBER to assign new ones.

For example, suppose that we have two source programs on the disk named "SRCONE" and "SRCTWO". We want to join them together so that "SRCONE" precedes "SRCTWO".

```
:LOAD SRCONE
:LIST

1000 *  PROGRAM NUMBER ONE
1010 MAIN   JSR SUBROUTINE
1020        RTS

:LOAD SRCTWO
:LIST

1000 *   SUBROUTINE TO DO SOMETHING
1010 SUBROUTINE
1020        LDA BLAH.BLAH.BLAH
1030        STA SOMEWHERE
1040        RTS

:HIDE
H:LIST

H:LOAD SRCONE
H:LIST
1000 *  PROGRAM NUMBER ONE
1010 MAIN   JSR SUBROUTINE
1020        RTS

H:MERGE
:LIST

1000 *   PROGRAM NUMBER ONE
1010 MAIN   JSR SUBROUTINE
1020        RTS
1000    *   SUBROUTINE TO DO SOMETHING
1010 SUBROUTINE
1020        LDA BLAH.BLAH.BLAH
1030        STA SOMEWHERE
1040        RTS

:RENUMBER
:LIST

1000 *   PROGRAM NUMBER ONE
1010 MAIN   JSR SUBROUTINE
1020        RTS
1030 *   SUBROUTINE TO DO SOMETHING
1040 SUBROUTINE
1050        LDA BLAH.BLAH.BLAH
1060        STA SOMEWHERE
1070        RTS
```

INCREMENT Command:

Sets the increment used for automatic line number generation. The increment is normally 10, butyou may set it to any value between 0 and 9999. (Of course, an increment of 0 makes no sense. Neither does a large value like 9999. But you can use them if you wish!)

```
:INC 5          (set increment to 5)
```

MEMORY Command:

Displays the beginning and ending memory addresses of the source program and of the symbol table.

```
:MEMORY
SOURCE PROGRAM: $94F3-9600
  SYMBOL TABLE: $2500-2674
```

Memory between the top of the symbol table and the bottom of the source program is free to be used without clobbering anything.

The assembler automatically protects (during assembly) memory from $1000 to the top of the symbol table, and from the bottom of the source program through $FFFF. This insures that your object program will not clobber the assembler, the source program, or DOS.

RENUMBER Command:

Renumbers all or part of the lines in your source program with the specified starting line number and increment. There are three optional parameters for specifying the line number to assign the first renumbered line (base), the increment, and the place in the program to begin renumbering (start). There are four possible forms of the command:

```
    :REN            Renumber the whole source program:
                    BASE=1000, INC=10, START=0

    :REN #          Renumber the whole source program:
                    BASE=#, INC=10, START=0

    :REN #1,#2      Renumber the whole source program:
                    BASE=#1, INC=#2, START=0

    :REN *1,#2,#3   Renumber all lines from #3 through the end:
                    BASE=#1, INC=#2, START=#3
```

The last form above is useful for opening up a "hole" in the line numbers for entering a new section of code.

```
    :LIST
    1000 *   LITTLE RENUMBER EXAMPLE
    1005 SAMPLE LDA $35
    1006        STA $37
    1010        RTS

    :RENUMBER
    :LIST
    1000 *   LITTLE RENUMBER EXAMPLE
    1010 SAMPLE LDA $35
    1020        STA $37
    1030        RTS

    :RENUMBER 100
    :LIST
    0100 *   LITTLE RENUMBER EXAMPLE
    0110 SAMPLE LDA $35
    0120        STA $37
    0130        RTS

    :RENUMBER 2000,4
    :LIST
    2000 *   LITTLE RENUMBER EXAMPLE
    2004 SAMPLE LDA $35
    2008        STA $37
    2012        RTS

    :RENUMBER 3000,10,2008
    :LIST
    2000 *   LITTLE RENUMBER EXAMPLE
    2004 SAMPLE LDA $35
    3000        STA $37
    3010        RTS
```

RESTORE Command:

Restores the root source program if an assembly is aborted while inside an "included" module.

The "root source program" is the source program that is in memory at the time the "ASM" command is issued. If this source program uses the ".IN" directive to include additional source files, it is possible that assembly might be aborted while the "root" program is "hidden". An assembly may be aborted either manually by typing a RETURN key during the listing phase, or automatically due to an error in the source program.

If the assembly is aborted during the time that the root program is hidden, the prompt character changes from ":" to "I:". The RESTORE command will reset the memory pointers so that the root program is no longer hidden, and change the prompt character back to ":".

You do not have to use the RESTORE command after an abort unless you wish to get back to the root source program for editing purposes. If you type the ASM command, the assembler automatically restores before starting the assembly.

If an assembly aborts due to an error in a source line, you may correct the source line, SAVE the module on the appropriate file, and type ASM to restart the assembly.

USR Command:

This command is provided for the user who wishes to add his own commands to the S-C ASSEMBLER II.

When you type the command "USR", a JSR $1006 instruction is executed. If you have not installed a JMP to your own program at $1006, the command is equivalent to a "no operation" command. You can write a program to process your own command, and put a JMP instruction to it at $1006.

The S-C ASSEMBLER II command parser scans for up to two numeric (line number) fields following the command. The scan skips over any non-numeric characters until it either finds a digit or the end of line. If any numbers are found, the first one will be stored in $3A-$3B, and the second at $3C-$3D. the contents of the x-register indicates how many numbers were found: X=0 if no numbers found, X=2 if one number found, and X=4 if two numbers found.

The entire command line is stored in the monitor input buffer, starting at $0200. An index to the next character after the last digit of the second number (or the end of line if there were less than two numbers) is in $06.

PRT Command:

In previous versions of S-C ASSEMBLER II, the PRT command turned on a printer driver for a particular printer. This driver was built-in to the assembler, and was designed for the practical Automation DMTP uP-6 printer, using a serial scheme through the game port. Very few customers were able to use the PRT command without modifying the program.

Version 4.0 provides a more general technique, without wasting memory for those who have no need for the PRT command. (Those who have an Apple parallel of Serial printer Interface Card may use the PR#slot command.) When you type the command "PRT", a JSR $1009 instruction is executed. If you have not installed a JMP to your own printer driver at this location, the command is equivalent to a "no operation" command. You can write a program to drive your own printer through the game port, or through any interface board which requires special handling.

If you have no need for the PRT command, you may use it as a second USR command.

VAL command:

The VAL command will evaluate any legal operand expression, and print the value in hexadecimal. It may be used to quickly convert decimal numbers to hexadecimal, to determine the ASCII code for a character, or to find the value of a label from the last assembled program.

```
:VAL 12345
3039
:VAL-21846
AAAA
:VAL 'X
0058
:VAL LOOPA+3
084E
```

Seven new assembler directives have been added to S-C ASSEMBLER II.

.BS expression	Reserve <expression> bytes at the current location in the object program.
.LIST OFF	Turn off the assembly listing.
.LIST ON	Turn on the assembly listing.
·PG	Page eject. Issues a Form Feed character.
.IN file name	Include a source program from the specified file.
.TF file name	Put the object program on the specified file.
.US	User programmable directive.

Block Storage: .BS exp

Reserves a block of <exp> bytes at the current location in the program. The expression specifies the number of bytes to advance the location counter. If there is a label, it assigned the value at the beginning of the block.

The address of the beginning of the block will be printed in the address column of the assembly listing.

If the object code is being stored directly into memory, no bytes are stored for the .BS directive. However, if the object code is being written on a file using the .TF directive, the .BS directive will write <exp> bytes on the file. All the bytes written will have the value $00.

Listing Control: `.LIST OFF`
`.LIST ON`

This pair of directives turns the assembly listing on and off. If .LIST OFF is put at the beginning of the source program, and no .LIST ON is used, no listing at all will be produced. The program will assemble much faster without a listing, as most of the time is consumed in putting characters on the screen and scrolling the screen up.

If you put .LIST OFF at the beginning of your source program, and .LIST ON at the end, only the alphabetized symbol table will print.

You may also use this pair of directives to bracket any portion of the listing you wish to see or not see.

Page Control: `.PG`

Prints an ASCII Form Feed character ($0C). If the assembly listing is being printed on a printer which recognizes this character, a form feed will occur and the next listing line will appear at the top of the next page. The .PG line itself is not listed.

Include: .IN file name

Causes the contents of the specified source file to be included in the assembly.

The program which is in memory at the time the ASM command is typed is called the "root" program. Only the root program may have .IN directives in it. If you attempt to put .IN directives in an included program, the "NESTED .IN" error will print.

When the .IN directive is processed, the root program is temporarily "hidden" and the included program is loaded. Assembly then continues through the included program. When the end of the included program is reached, it is deleted from memory and the root program is restored. Assembly then continues with the next line of the root program.

If you type the MON C command (a DOS command) before beginning assembly, the LOAD commands issued by the assembler will be printed with the listing. Each included program is loaded in turn during pass one of the assembly, and again during pass two.

The .IN directive is useful in assembling extremely large programs, which cannot fit in memory all at once. It is also useful for connecting together a library of subroutines with a main program.

The <file name> portion of the directive is in standard DOS format, and may include volume, slot, and drive number.

Target File: .TF file name

Causes the object code generated to be stored on a binary file, rather than in memory. Only the code which follows the .TF directive will be stored on the file. Code will be stored on the file until another .TF directive is encountered, or until a .TA or .OR directive is encountered.

The <file name> specifier may include volume, drive, and slot numbers if necessary. If you have both .IN and .TF directives in the same assembly, and the files involved are not on the same disk, you will need to specify slot number (and maybe drive number) with every .IN and with every .TF directive.

If your program consists of several pieces with different origins, and you want them all to be put on files, each piece will require a separate .TF directive. The object code is written on a binary file, which may only have one origin.

During assembly, S-C ASSEMBLER II temporarily patches DOS to allow a binary file to be handled with text file commands. It also creates a text file with your specified name and uses text file techniques to write the object code into the file. When assembly is complete, or when the .TF range is ended by encountering another .TF or .TA or .OR, the text file is transformed into a binary file by modifying the DOS directory entry.

If you have typed MON C (a DOS command) before assembly, the DOS commands issued by the assembler for the .TF directive will print on the assembly listing. For each .TF directive, during pass two, you will see the following sequence:

```
OPEN file name
DELETE file name
OPEN file name
WRITE file name
```

User Directive .US

To allow for possible expansion of the assembler by users, the .US directive has been included. When the opcode is processed, it will branch to $100C. That location normally contains a JMP instruction, which treats the .US as a comment.

If you desire to use the .US directive, you change it to jump to your own program. Details of the steps necessary to implementing your own directives are not available at this time. You may disassemble S-C ASSEMBLER II, if you wish, and examine the existing directives.

There are two types of labels used in Version 4.0 of S-C ASSEMBLER II; normal labels, and local labels.

Normal Labels:

In previous versions of S-C ASSEMBLER II, labels could be 1-4 or 1-6 characters in length. In version 4.0, labels may be up to 32 characters long.

The first character of a normal label must be a letter; subsequent characters may be letters, digits, or the period character ("."). The period is useful for making long labels readable. For example, a subroutine to read the next source line might be named "READ.NEXT.SOURCE.LINE".

Tab stops are set up within the editor assuming that most of your labels will be six characters or less. However, since the assembler is relatively free-format, you may type any length label followed by a blank and the opcode, operand, andcomment fields. Or, if you wish, you may type the long label on a line all by itself. In this form, the label is assigned the current value of the location counter, just as if you had appended ".EQ *" to the line.

```
1000 *   SAMPLE PROGRAM WITH LONG LABELS
1010 SOURCE.LINE.POINTER .EQ $13 AND $14
1020 CHAR.POINTER .EQ $12
1030 *
1040 READ.NEXT.CHAR.FROM.LINE
1050        LDY CHAR.POINTER
1060        LDA (SOURCE.LINE.POINTER),Y
1070        INC SOURCE.LINE.POINTER
1080        RTS
```

Local Labels:

Version 4.0 introduces a new kind of label, called "local labels". The main purpose for local labels is to make programs more readable by reducing the number of label names you must invent. As a side effect, local labels save considerable space in the symbol table during assembly; they only require two bytes each. The use of local labels also encourages structured programming habits.

Local labels have a period as the first character, followed by one or two digits. Any label from ".0" through ".99" may be used.

Local labels are defined internally relative to the normal label which comes before it in the source program. The value must be no more than 255 greater than the value of the associated normal label.

Since each set of local labels is associated with a particular normal label, you may re-use the same local label names.

Here is an example of three little routines in the same source program, using normal and local labels:

```
1000 PRINT.MESSAGE
1010        PHA          SAVE A-REGISTER
1020 .1     JSR PRINT.CHARACTER
1030        INY
1040        LDA MESSAGES,Y
1050        BNE .1       =0 FOR END OF MESSAGE
1060        PLA          RESTORE A-REGISTER
1070        RTS
1080 *
1090 GET.NEXT.CHARACTER
1100        LDY CHAR.POINTER
1110        LDA INPUT.BUFFER,Y
1120        CMP #RETURN
1130        BEQ .1       END OF LINE
1140        INC CHAR.POINTER
1150 .1     RTS
1160 *
1170 GET.NEXT.NONBLANK .CHAR
1180 .1     JSR GET.NEXT.CHARACTER
1190        BEQ .2       END OF LINE
1200        CMP #BLANK
1210        BEQ .1
1220 .2     RTS
```

The S-C ASSEMBLER II program is about 5376 bytes long ($1500), and occupies $1000 through $24FF in memory. The symbol table begins at $2500 and extends upward; your source program begins at the bottom of DOS and extends downward.

During source program entry or editing, memory usage is monitored so that the source program does not grow so large as to overlap the symbol table. Overlapping will cause the "MEM FULL ERROR" message to print. During assembly, memory required by the symbol table is monitored, to prevent the symbol table from overlapping the source program. Overlapping will generate the "MEM FULL ERROR" message and abort the assembly.

In addition, memory usage by the object program is monitored, so that it will not destroy the source program, DOS, the S-C ASSEMBLER II, the symbol table, or switch any I/O addresses. Therefore, if the object program bytes are directed at any memory address between $1000 and the top of the symbol table, or any address above the beginning of the source program, the "MEM PROTECT ERROR" will be printed and assembly aborted.

The assembler also uses a many locations in page zero during editing or assembly. However, the only ones which you need to maintain unchanged during execution or testing of your object program are $4A-4D, $CA-CD, $73-74, $D9. Location $D9 is used by DOS as a flag to allow commands to be entered. The other locations are used to point at the beginning and end of your source program and symbol table.

Page one ($100-1FF) is used both as a stack and as storage for the symbol table routines. The high addresses in page one are used for the stack. The low end is used for a symbol buffer and for pointers to the 27 hash chains used in storing the symbol table.

Page two ($200-2FF) is used as an input buffer.

The high end of page three ($3D0-3FF) is used by DOS and by the assembler. You must not change any bytes between $3D0 and $3EF. $300-3CF is free to be used in any way you wish.

Locations $400-7FF are used by the Apple II as the screen buffer. There are 32 bytes which are unused by the screen image, which are instead used by certain peripheral boards such as the disk controller or printer interface boards.

Locations $800-$FFF are free to be used for storing your object program. Also, with care and planning, you can find space for object program storage between the top of the symbol table and the bottom of the source program. If the space there is too difficult to determine, or insufficient size, you need to use the ".TF" directive to store the object program directly on a binary disk file.

Since Version 4.0 will execute regardless of which set of language ROMs is in force, S-C ASSEMBLER II no longer uses any routines from the Integer BASIC ROMS. The following table is believed to be a complete list of the monitor ROM routines used.

F94A	Print (X) blanks
FBF4	Advance cursor
FC10	Backspace cursor
FC1A	Move cursor up one line
FC2B	An RTS instruction
FC42	Clear to end of page
FC66	Move cursor down one line
FC9C	Clear to end of line
FCA8	Delay
FD0C	Read next input character
FD84	Add char to input line
FD99	Print (Y,X) in hex with dash
FDDA	Print (A) in hex
FDED,FDF0	Print (A) as ASCII char
FE00	Display memory in hex
FE2C	Move block of memory
FECD	Write block of memory on tape
FEFD	Read tape into memory
FF2D	Print "ERR", ring bell
FF3A	Ring bell
FFA7	Get hex number
FFBE	Process monitor command
FFC7	Clear monitor mode byte
FFCC	Table of monitor commands

At long last, publishers have begun to release some good books for learning how to program the 6502 microprocessor. Some new ones which have been published since the S-C ASSEMBLER II Version 3.2 Reference Manual was written are listed here.

Apple Assembly Line, a monthly newsletter published by S-C SOFTWARE. See advertisement inside back cover for details.

APPLE-GRAM, newsletter of the Apple Corps of Dallas, Texas. Subscriptions currently $12/year, from Apple Corps, Attn: Membership Chairman, P. 0. Box 5537, Richardson, TX 75080. Contains many useful articles covering every aspect of the Apple II, including assembly language programming.

6502 Software Design, Leo J. Scanlon. One of the Blacksburg Continuing Education Series, published by Howard W. Sams & Co., 1980. 270 pages, paper, $10.50.

6502 Assembly Language Programming, Lance A. Leventhal. Osborne/McGraw-Hill, Inc., 1979. Over 80 programming examples, tested on an Apple II.

Programming & Interfacing the 6502, With Experiments, Marvin L. De Jong. One of the Blacksburg Continuing Education Series, published by Howard W. Sams & Co., 1980. 414 pages, paper, $13.95.

6502 Software Gourmet Guide & Cookbook, Robert Findley. Scelbi Publications, 1979. 204 pages, paper, $10.95. Includes listings of conversion routines, search and sort routines, and floating point routines.

6502 Games, Rodney Zaks, SYBEX. The third in the SYBEX series on programming the 6502. Includes listings of games in assembly language, of the type which are usually programmed in BASIC.

Wozpak II and Other Assorted Goodies. A collection of Apple II documentation published by the publishers of CALL A.P.P.L.E. It contains many useful programs in assembly language which can be used and/or studied.

Practical Microcomputer Programmina: the 6502, W. J. Weller, Northern Technology Books, 1980. 459 pages, $32.95. Includes a listing of a 6502 assembler and of a debugging package.

Apple Assembly Line

A newsletter dedicated to Apple Assembly Language!

* Tutorial articles for beginners!
* Advanced hints and techniques which save memory and time!
* Ways to modify and improve standard products!
* Utility programs, ready to type in and run!
* Super subroutines for use in your programs!
* Commented assembly listings of code found in DOS and the Apple ROMS!
* New features you can add to your copy of the S-C ASSEMBLER II Version 4.0!
* A place to publish your handy routines and articles and advertisements!
* Editor and principal author is Bob Sander-Cederlof, author of the S-C ASSEMBLER II!
* Only $12 per year (12 issued)
* All back issues stil in print!
* A quarterly disk available to subscribers ($15 each), containing all the source code printed in AAL during the quarter!

Partial Table of Contents

80 Columns on your Printer
Applesoft Internal Entry Points
Modify S-C ASSEMBLER II Error Messages
Fast String Input Routine for Applesoft
Hiding Things Under DOS
Commented Listing of DOS 3.2.1 Formatter
Commented Listing of DOS 3.3 Formatter
Substring Search for Applesoft

May, 1981
Hi-Res SCRN Function for Applesoft
Conquering Paddle Jitter
Tutorial About Shifting (ASL, LSR, ROL, and ROR Opcodes)
6502 Programming Model
Commented Listing of DOS 3.2.1 $B800-BCFF

June, 1981
More About 80 Columns on Your Printer
Two Fancy Tone Generators
More About Multiplying on the 6502
Specialized Multiplications
Commented Listing of DOS 3.3 $B800-$BCFF
A Review of "Beneath Apple DOS"

July, 1981
Using Firmware Card in Slot 4
The Lower Case Apple
Screen Printer
Restoring Clobbered Page 3 Pointers
Step-Trace Debugging Utility

Quarterly Disks

All source programs printed in the newsletter are available on quarterly disks for $15 per disk. Source code is formatted for the S-C ASSEMBLER II Version 4.0.

AAL-QD#1	Oct 1980 thru Dec 1980	(Now)
AAL-QD#2	Jan 1981 thru Mar 1981	(Now)
AAL-QD#3	Apr 1981 thru Jun 1981	(Now)
AAL-QD#4	Jul 1981 thru Sep 1981	(Soon)

S-C ASSEMBLER II Version 4.0

Still the easiest Apple assembler for beginners, and powerful enough for experts! Thousands of happy owners!

You can purchase it at many local computer stores for $55. If not available at your store, you can order by mail from S.C SOFTWARE (same price, we pay postage in USA, Mexico, and Canada).

Apple Assembly Line Subscriptions and Back Issues

Only $12 per year in the USA, Mexico, and Canada. Other countries add $12/year for postage. All back issues are available at $1.20 each. (Other countries add $1 postage for each back issue ordered.)

S-C SOFTWARE
P.O. Box 280300
Dallas, TX 75228
(214) 324-2050