

Aids for Hand Assembling Programs

BRAVEC

The program takes a 16 bit number ORigin and adds two to it. The new number then is subtracted from another 16 bit number, DEstination. The difference, which may be positive or negative, in two's complement, is stored in POINTL. The difference is also examined to determine if it is larger than +127 (if positive) or smaller than -127 (if negative). If this is the case, FF is loaded into POINTH; otherwise 00 is loaded. POINTH and POINTL are then displayed by transferring control to the (KIM) operating system.

Listing 1: Program description for BRAVEC. This description should be the first step taken when writing a program.

Erich A Pfeiffer PhD
Wells Fargo Alarm Services
Engineering Center
1533 26th St
Santa Monica CA 90404

Resident assembler programs and interpreters for high level languages are available increasingly for microcomputer systems based on the more popular microprocessors. Nevertheless, many operators of small microcomputer systems are unable to use such programs because their systems are not large enough to support them. Unless they are lucky enough to have access to a timesharing service or to some larger computer which supports a cross assembler, their only way of developing a usable object program is to assemble it by hand.

While the mere idea of such an endeavor might horrify any programmer who is used to working with large machines, the hand assembly of shorter programs for 8 bit microprocessors actually is not very difficult. It has been my experience that the assembly of programs can be greatly simplified and the likelihood of errors can be reduced by using some simple aids in the assembly process.

One of these aids is in the form of hardware and consists of a special program assembly form. The software aids are several short utility routines which run even on the smallest microcomputer systems. Develop-

ment of the assembly method described in this article is based on experience gained from working with programmable calculators of the keyboard language type. Matt Biever of the Pro-Log Corporation has long been advocating some of the techniques that I am using. The article's assembly method is used for program development for a KIM-1 microcomputer. It can be adapted easily for other microcomputer systems as long as they use an 8 bit processor. The assembly method will be demonstrated with a sample program.

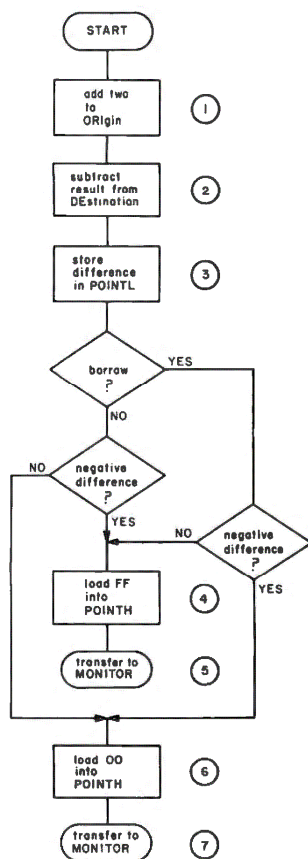
Before writing a program, it is a good idea to put down in writing what the program is supposed to do. Such a program description, as shown in listing 1, might state any limitations on the magnitude of variables used or might indicate what happens if these limitations are exceeded.

The next step is to develop a concept of the program in the form of a flowchart as in figure 1. While the symbols used in such charts are standardized, the chart's degree of detail is a matter of personal preference. From program descriptions and flowcharts, one can determine how many memory locations or registers will be necessary to store data and temporary results. These locations should be written in the program register table as shown in table 1. This table also contains the addresses of sub-routines or registers of the monitoring system that are called by the program, or of PIA registers that will be addressed. The table is similar to the symbol table printed by the computer during the machine assembly of a program.

After a program description is developed the actual writing of the program can begin. The programmer, who writes a symbolic listing for machine assembly, arranges a program in the form of lines. Each line is successively numbered, contains one mnemonic for an operation (unless it is an "all comment" line) and later will be punched into one punch card for computer entry. Because the operation described by the mnemonic can have a length of one, two or three bytes, each line eventually results in

one, two or three machine instructions. Therefore, there exists no simple relation between the line number and the address at which the machine code is stored in the computer memory. For the hand assembly of programs, it is advantageous to use a different format for the program listing in which there is a one to one relationship between program line and memory location. The writing of the symbolic program and the assembly into machine code is greatly simplified by the use of a special program assembly

Figure 1: Flowchart of the program described in listing 1. The circled numbers refer to the comment numbers in listing 2.



Use	Label	Location
ORIGIN	ORLO ORHI	0000 01
DESTINATION	DELO DEHI	02 03
"open cell"	POINTL POINTH	FA from listing of FB KIM monitor
Transfer to KIM monitor	START	1C4F from listing of KIM monitor

Table 1: Program register table for program BRAVEC. This table contains all descriptions of all memory locations used by the program.

Build The World's Most Powerful 8-Bit Computer

Featuring The Famous Intel 8085!

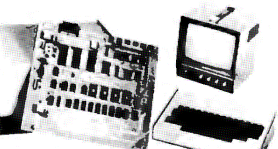
Explorer/85™

Starting for just \$129.95 you can now build yourself a sophisticated, state-of-the-art computer that can be expanded to a level suitable for industrial, business and commercial use. You learn as you go... In small, easy-to-understand, inexpensive levels!

- Features Intel 8085 cpu/100% compatible with 8080A software!
- Onboard S-100 bus (up to 6 slots)!
- Onboard RAM and ROM expansion!
- Built-in deluxe 2K Monitor/Operating ROM!
- Cassette/RS 232 or 20 ma J-17 8-bit parallel I/O and timer all on beginner's Level "A" system!

EXPLORER/85 gives you a big computer features immediately without turning you into an appliance operator. You can run pre-developed software for life. Simply connect EXPLORER to a terminal, video monitor or tv set and 8 volt power supply and start running programs the very first night! Level "A" teaches you machine language and computer fundamentals. It lets you run exercise programs including programs to examine the cpu registers, examine memory, fill memory, move memory and make us games. You can load and play back these programs on an ordinary tape cassette—and display your efforts on any tv screen, video monitor or printer. (\$8.95 RF modulator required for tv use.) The simplified architecture of the Intel 8085 makes EXPLORER far easier to understand than computers using the older, more complex but less powerful 8080A. Then, when you're ready, EXPLORER can be expanded—by you—to rival the power of any 8-bit computer on earth. Or you can customize it to perform a dedicated task. Thanks to onboard preprogramming RAM and ROM expansion capabilities.

LEVEL "A" SPECIFICATIONS
EXPLORER's Level "A" system features an advanced Intel 8085 cpu, which is 50% faster than its 8080A predecessor yet 100% compatible with 8080A software which you'll discover exists by the ton. Big computer features include an 8355 ROM with 2K deluxe monitor/operating system which has two programmable 8-bit bi-directional parallel I/O ports, built-in cassette interface with tape control circuitry to allow labeling cassette files, and commands which include: display contents of memory, run at user location (go to), insert data, move contents of memory, examine registers individually or all, the command (to fill the contents of memory with any variable) automatic baud rate selection, program mode characters are line display output format, and more! An 8155 RAM—I/O chip contains 256 bytes of RAM, two programmable 8-bit bi-directional and one programmable 8-bit bi-directional I/O ports plus programmable 14-bit binary counter/timer, user interrupt and reset switches. Onboard expansion provisions exist for up to six S-100 boards, 4K of RAM and 8K of ROM, PROM or EPROM.



As featured in
POPULAR ELECTRONICS
EXPLORER/85 shown with Video Monitor and Keyboard/Video Terminal

CHOICE OF HEX KEYBOARD OR TERMINAL INPUT
If you plan to customize EXPLORER for dedicated use, we recommend that you order hex keypad input. But, if you are planning to go whole hog and blow EXPLORER up into a full size, state-of-the-art system with 8K or extended basic (coming soon) up to 64K of memory, floppy disks, telephone interface, printers, and all sorts of S-100 plug-ins—you'll be better off with the Keyboard/Video Terminal input. The \$149.95 EXPLORER Keyboard/Video Terminal includes full ASCII decoding with 108 ASCII upper/lower case set, 96 printable characters, onboard register and variable display formats—32x16 for tv set or 64x16 for video monitor (not included).

EXPAND EXPLORER, LEVEL-BY-LEVEL
Includes all parts necessary to generate the signals for S-100 bus connectors. Just add two S-100 bus connectors and you have a complete S-100 computer system with a world of add-ons at your fingertips. Choose from hundreds of products to satisfy your individual needs. Level "B" kit also includes the address decoders for onboard RAM and ROM expansion which are addressable anywhere in the 64K field.
Level "C" expansion, at \$39.95, expands the S-100 bus to allow a total of six S-100 cards to be plugged into EXPLORER's motherboard and contained in EXPLORER's steel cabinet. Includes all hardware mounting brackets, board guides, etc. Just add the number of S-100 bus connectors you need.
Level "D" expansion, at \$69.95, gives you 4K of onboard static RAM utilizing 2114 IC's. Your board will also accept four 2716 EPROM's, which can be purchased separately. You now have an advanced mainframe that can be customized with the peripherals of your choice to fit any (or all) specific requirements. Each level of EXPLORER is separately regulated for the ultimate in stability. Factory service is available from Netronics. Order your EXPLORER today!


ORDER FROM THIS COUPON TODAY!
Netronics R&D Ltd., Dept. EY-5, 333 Litchfield Road, New Milford, CT 06676

<input type="checkbox"/> Level "A" EXPLORER/85 kit (specify <input type="checkbox"/> terminal or <input type="checkbox"/> hex keypad input) \$129.95 plus \$3.00 p&h	<input type="checkbox"/> Deluxe Steel Cabinet for EXPLORER/85 \$39.95 plus \$3.00 p&h
<input type="checkbox"/> Power Supply kit 5 amp ±8 volt, \$34.95 plus \$2.00 p&h	<input type="checkbox"/> Deluxe Steel Cabinet for Keyboard/Video Terminal \$19.95 plus \$2.50 p&h
<input type="checkbox"/> Intel 8085 User's Manual \$7.50 p&h	<input type="checkbox"/> RF Modulator kit \$8.95 p&h
<input type="checkbox"/> ASCII Keyboard/Video Terminal kit \$149.95 plus \$3.00 p&h	<input type="checkbox"/> Total Enclosed Components Add Tax \$
<input type="checkbox"/> Hex Keypad kit for hex version \$69.95 plus \$2.00 p&h	<input type="checkbox"/> VISA <input type="checkbox"/> MasterCard <input type="checkbox"/> Exp. Date
<input type="checkbox"/> Level "B" S-100/Onboard RAM/ROM Decoder kit (needs S-100 connectors) \$49.95 plus \$2.00 p&h	PHONE ORDERS CALL (203) 354-9375
<input type="checkbox"/> Level "C" S-100 S-Card Expander kit (needs connectors) \$39.95 plus \$2.00 p&h	Account # _____
<input type="checkbox"/> S-100 Bus Connectors (gold) \$4.85 each	Address _____
<input type="checkbox"/> Level "D" 4K Onboard RAM kit \$69.95 plus \$2.00 p&h	City _____
	State _____ Zip _____

DEALER INQUIRIES INVITED

MORE BANG PER BUCK

The PERKIN-ELMER BANTAM



ALL
NEW


\$799.00

All the Features of the
Hazeltine 1400 & LSI ADM-3A
Plus

Upper/Lower Case
7 x 10 Char. Matrix
White or Black Char
Transparent Mode

Tab Function
Backspace Key
Shiftlock Key
Print Key
Integrated Numeric
Pad


\$41.61 per month
Lease-Purchase



\$1095.00

TELETYPE
MODEL 43
KSR

with RS232
10 or 30 CHAR/SEC
132 COLUMNS
UPPER/LOWER CASE




USR-310
Originate
Acoustic
Coupler

0-300 Baud
Crystal Controlled

\$149.00


Stand Alone
RS232



USR-330
Originate
Auto-Answer
Modem

FCC Certified for Direct Connection
to Phone Lines

\$324.00



USR-320 Auto-Answer
Modem

Only Modem

\$299.00

All Units include a 120 day warranty.
Optional Maintenance package available

Any Product may be returned
within 10 days for a full refund.

U.S. ROBOTICS, INC.

1035 W. LAKE ST.
CHICAGO, ILL. 60607

Sales (312) 733-0497
General Offices (312) 733-0498
Service (312) 733-0499

form. The form I developed for our KIM-1 system is shown in listing 2. (Similar forms are available from the Pro-Log Corporation; order Nr CF-1.) Each line of the coding form corresponds to one memory location with the least significant hexadecimal digit of the address preprinted in the ADD column. The form can be used with any computer system that uses a hexadecimal machine code. For octal notation, a different layout is advantageous.

The programmer starts writing a program by adding the other digits of the program starting address in the ADD and Page

Listing 2: Program listing of BRAVEC using the author's hand assembly form for the KIM-1. This form can be used with any hexadecimal based micro-processor.

Program: **BRAVEC**

Page 1 of 2 Date:

Programmer:

Page	ADD	OPC	Label	MNE	Mode	Operand	N	Comment
00	00		ORLO					
	1		ORHI					
	2		DELO					
	3		DEHI					
	4	18		CLC				①
	5	A9		LDA	#	2		
	6	02		/				
	7	65		ADC	Z	ORLO		
	8	00		/				
	9	90		BCC		NELO		
	A	02		/				
	B	E6		INC	Z	ORHI		
	C	01		/				
	D	85	NELO	STA	Z	ORLO		
	E	00		/				
	F	38		SBC				②
10	A5			LDA	Z	DELO		
	1	02		/				
	2	E5		SBC	Z	ORLO		
	3	00		/				
	4	85		STA	Z	POINTL		③
	5	FA		/				
	6	A5		LDA	Z	DEHI		②
	7	03		/				
	8	E5		SBC	Z	ORHI		
	9	01		/				
	A	A5		LDA	Z	POINTL		
	B	FA		/				
	C	90		BCC		NELO		
	D	09		/				
	E	10		BPL		OUT		
	F	09		/				

VA-BECC Program Assembly Form


Listing 2 continued:

Page 2 of 2 Date:

Programmer:

Page	ADD	OPC	Label	MNE	Mode	Operand	N	Comment
	2	0	A9	FLAG	LDA	#	4 FF	④
		1	FF					↓
		2	85		STA	Z	POINTH	
		3	FB					
		4	4C		JMP	ABS	START	⑤
		5	4F					↓
		6	1C					
		7	10	NEG	BPL		FLAG	
		8	F7					
		9	A9	OUT	LDA	#	00	⑥
		A	00					↓
		B	85		STA	Z	POINTH	
		C	FB					
		D	4C		JMP	ABS	START	⑦
		E	4F					↓
		F	1C					
		0						
		1						
		2						
		3						
		4						
		5						
		6						
		7						
		8						
		9						
		A						
		B						
		C						
		D						
		E						
		F						

VA-BECC Program Assembly Form



Career Opportunities in Robotics and Computer Vision Systems

Texas Instruments has immediate openings for highly motivated, talented individuals with interest in the areas of robotics and pattern recognition. You will be a member of a team whose function is to develop and apply advanced technologies, design and implement working systems, and develop state-of-the-art tools and procedures for a broad range of industrial automation applications.

Hardware/Software

- Computer Architecture
- Operating Systems
- Systems Programming
- Mini/Micro Assembly Language Programming
- Electro Optics/Video Display Systems

- Robotics
- Computer Vision System
- Computer Speech I/O
- Intelligent Machines
- Servo Systems

If you have an Associate or higher degree, or equivalent experience, and are looking for a challenging opportunity in any of the above areas, send your resume in complete confidence to: Staffing Manager/P. O. Box 225474, M.S. 217/Dallas, TX 75265.

TEXAS INSTRUMENTS
INCORPORATED

An equal opportunity employer M/F

mode addressing is commonly indicated by the symbol #. For other addressing modes, suitable abbreviations of the column headings in the programmer's reference card should be used. For operations which have only one addressing mode, the Mode column is left empty. The addressing mode determines how many address bytes will have to follow the op code byte. After filling in the Mode column, the programmer should cross out the appropriate number of lines in the MNE column. This reserves the corresponding memory locations for the address or operand part of the instruction.

The Label column will carry an entry for two conditions only:

- If the line contains the start of a subroutine.
- If the line is the destination of a conditional or unconditional jump or branch instruction.

While assembly programs sometimes put certain limitations on the choice of labels, any suitable word or letter and number combination can be used as a label for hand assembly. However, it makes sense to pick a word or abbreviation that indicates what

the subroutine or branch destination is doing in the program, (ie: "WAITLOOP," "COUNT," or simply "LOOP 7").

The next column to fill in is the one with the heading Operand. When writing programs for machine assembly, the programmer enters a symbolic label in this field and leaves it up to the assembly program to figure out what to do with it. When writing for hand assembly, the programmer can make the task easier by being a bit more specific. The operand can be one of the following things:

1. In the immediate addressing mode, it is simply the number that is to be entered by the operation. Rather than give this number a symbolic name which is defined somewhere in a symbol table, it is much easier to enter it directly in the Operand column. One has to be careful to remember which number system is being used. A number without a prefix indicates decimal notation. The prefix % indicates binary notation. A bit mask for bit 2 and 0, for example, would have the operand %00000101. If the number is in hexadecimal form, the prefix \$ would normally be used, but in this case it is much simpler to enter the hexadecimal number directly in the OPC column of the following line.

2. With a jump or branch instruction, the operand symbol indicates the destination of the operation. The operand of such an operation must have a counterpart in the label column somewhere in the program. The only exception is when the program calls subroutines that are stored in read only memory (as I do frequently with subroutines of the KIM monitoring system). In this case, the operand symbol has to have a counterpart in the stored program.

3. With any other memory referenced instruction, the operand must symbolize a memory location. I have found it useful to think of these locations as registers even though, unlike the registers of the processor, they are physically located somewhere in memory. As a matter of fact, their location, if possible, is in page zero of the memory to take advantage of the shorter addressing mode. For registers used in stock subroutines, I have assigned locations which begin at the upper end of page zero and work their way downward. They are listed in a master register list and care has been taken that subroutines that are likely to be used in the



Get your PC masters in as little as 2 weeks

At Echo Design your circuit drawings can be converted into finished artwork masters in only 2 to 6 weeks, depending on complexity.

We do board layouts for many of the biggest names in the business.

And we have broad capability. Such as computer boards having 450 ICs.

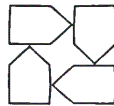
Choose any or all these services:

- Layout (to digitizing standards if desired)
- Schematic drawing
- Tape-up (artwork)
- Bill of material
- Fab drawing
- Printed board

Place a call now to John Offenbacher or Al Chew and get your new board moving at competitive prices.

FREE
Ask for
a copy

Basic
Guidelines
for
Printed
Circuit
Partitioning



echo
DESIGN AND
DEVELOPMENT
CORPORATION

195 EAST GISH ROAD • SAN JOSE, CA 95112

408-292-0918

We also provide contract technical personnel world wide

same program do not occupy the same register addresses. The symbolic names for registers that will be used in the main program are noted in a program register table (table 1) with the addresses to be assigned later. The symbols again should be words or abbreviations which indicate the meaning of the data contained in the register, such as STARLO to mean starting address, low order byte.

The column N of the program assembly form can be used to indicate the number of cycles it takes to execute the instruction. This is necessary, for example, to determine the time of timing loops. In most cases, however, this column will be left empty.

Finally, the Comment column should be used to explain the function of the operation listed in the current line and sometimes some following lines. While this information may not be needed by the programmer, it is tremendous help for any other person trying to understand what the program is doing. If the program has been flowcharted first, which is highly recommended for all but the shortest programs, the comment can simply be a number which refers to an equally numbered symbol on the flowchart.

In this way the programmer works down the lines of the program assembly form. Every time a 0 is encountered in the ADD column, (s) he adds the most significant bit. If that addition makes the ADD column is also advanced. Eventually the program will be completed and the hand assembly can begin. Like the computer, I do this in a number of passes.

The first pass is the easiest one. Using a listing of the instruction set, or the programmer reference chart, the mnemonic and the entry in the Mode column is used to look up the op code of the instruction, which is entered into the OPC column of the line. A frequent error during this operation is to mistake an 8 for a B or vice versa, and I double check op codes with these symbols. The programmer's reference cards supplied by the manufacturers, although they fit nicely into a shirt pocket, were apparently not intended for use by programmers over 40 years of age. The listing of the instruction set in the data sheets or system manuals is usually printed in a more reasonable letter size.

The second step is to assign absolute addresses to the symbols of the program register list. First, all registers and their addresses used in stock subroutines to be called by the program are transferred

from the master register list to the program register list. Then absolute addresses are assigned to all other registers listed, making sure that no duplication occurs. Registers which contain the low and high order bytes of numbers, or registers which contain successive bytes if multiple precision operations are used, have to be arranged in such a way that their absolute addresses are adjacent in increasing order (STARLO = B3, STARHI = B4).

With the completed program register list one can go over the program again. For each memory referenced instruction other than branch and jump instructions, the program register list will contain an absolute address for the symbol in the operand column. This hexadecimal number is now entered into the OPC column of the following line. For registers located outside of page zero (such as the registers in PIAs) the address will be entered in two lines and care has to be taken to enter the low order byte first, followed by the high order byte. During this pass I also check all lines with a # in the Mode column and, if necessary, convert the binary or decimal operand into hexadecimal notation which is entered in the OPC column of the following line.

Parallel Processing Power for the S-100 bus

Discussed and dreamed about by computer scientists for years, Content-Addressable Memory (CAM) is now here at an affordable price. CAMs have been so costly to build that few have actually been produced. Now Semionics has developed a simplified design, lowering the cost by two orders of magnitude. This new memory is called Recognition Memory (REM), since (like the human brain) it can recognize words, patterns, etc.

Adding a REM board to an ordinary microcomputer converts it into a very powerful machine known as a Content-Addressable Parallel Processor (CAPP).

Features:

- 4K bytes per board
- Static — no refresh needed
- Can be used as ordinary RAM or as CAM
- RAM access time: 200 ns
- CAM access time: 4 μ s
- Multiwrite — writing into multiple locations with one instruction
- Masking — for individual bit access
- Multiple REM boards accessed in parallel

Adds 17 associative memory functions to instruction set of Z-80 or 8080.

Applications:

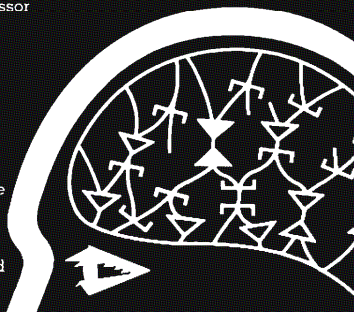
- Pattern Recognition • Information Retrieval • Compiling & Interpreting • Natural Language Processing • Code Compression • Artificial Intelligence

Price: \$345

2K firmware package of REM routines: \$40

SEMIONICS

41 Tunnel Road • Berkeley • CA 94705
(415) 548-2400



Circle 320 on inquiry card.

May 1979 • BYTE Publications Inc. 243

With this step completed, the OPC column should show a hexadecimal number in most lines. The next step is to pass over the program listing another time.

Any line with an open OPC column where the mnemonic indicates a branch instruction will require that the branch vector for the relative addressing mode be calculated. For short forward branches this poses no problem because the offset can easily be counted off (beginning at the second line following the one which contains the branch instruction, and continuing to the line which has the corresponding symbol in the label column). For longer branches and especially backwards branches, if memory pages are crossed it is very easy to make a mistake and miss by one count in either direction. I have found it advantageous to let the microcomputer perform this operation because, after all, it is much better in hexadecimal calculations than any programmer.

The example program BRAVEC receives the origin and destination of a branch and calculates the branch vector in two's complement notation. A flag is set if the relative addressing range is exceeded. The program is loaded from cassette tape beginning at memory location 0000. Loading begins here because this location in the KIM-1 system can be addressed easily by pressing the space bar of the connected terminal. The first four locations are actually data registers into which the low and high order bytes of origin and destination of the branch are entered.

When the program is executed beginning at location 0004, it displays or prints the branch vector in two's complement as the low order byte of the address field. The high order byte of this field normally shows 00, while FF indicates that the reach of the relative addressing mode has been exceeded.

While the program, as listed, is written for the 6502 microprocessor, only instructions that have an equivalent in the instruction set for the 6800 were used. The program, therefore, can be converted easily. However, the registers POINTHI and POINTLO, which are displayed as an address in the LED display of the KIM-1 microcomputer, are specific for this system. For other computers the user will have to find another way of displaying the result of the calculation.

After all branch vectors have been calculated in this fashion and entered in the appropriate lines, the only open spaces in the OPC column should be the address parts

of jump instructions. For jumps within the main program, it is easy to find the line with a matching entry in the label column and to enter the address of this line into the OPC columns of the lines following the one containing the jump instruction. For subroutines called from read only memory, the address has to be looked up in the subroutine listing.

Stock subroutines which have been written on some other occasion and which can be loaded from magnetic or paper tape frequently can be used. Normally such subroutines will be tacked on after the last memory location occupied by the main program. The KIM-1 system has a relocating loading routine for loading from magnetic tape. If this feature is not available, some area in the memory should be set aside into which the subroutines are loaded. A move program then can be executed to pull up the subroutine. For the 6502 processor I use a program called MOVBL0 which requires only 14 program steps due to one very convenient addressing mode of this processor.

Unless one is very pressed for memory space, it is a good idea to have all subroutines start in lines with a 0 as the least significant digit because it is easier to keep track of the starting address after relocation. In order to be relocatable, a subroutine may not contain any absolute jump instructions and only relative addressing within the subroutine is permitted.

After the last addresses for the stock subroutines have been entered in the program assembly form, the hand assembly is completed. I have never clocked the operation, but by following the methods described, it goes much faster than one would expect. With all op codes being listed in a single column it is much easier to enter them into the machine, either from a hexadecimal keyboard or from the keyboard of a terminal. This is another occasion in which operator errors can easily occur and I proofread all programs after entry. This operation is again greatly simplified by the use of the assembly form which shows address and op code in adjacent columns.

The assembly method and the assembly aids described have been in use for several months and have been found to greatly reduce the likelihood of assembly errors. Unfortunately, this method does not protect from programming errors and the debugging of the program still is a time consuming but necessary step to follow the assembly of a program.■