Robert Reiling, editor □ Post Office Box 626 □ Mountain View, CA 94042
Volume Number 2, Issue 4          April 30, 1976

THIS ISSUE - Robert Reiling

MORE 6800 SOFTWARE - Ray Boaz has memory test programs which he describes in this issue under the title BAD BIT GETTERS.....Tom Pittman explains HOW TO MAKE THE 6800 RESIDENT ASSEMBLER AND EDITOR WORK IN YOUR SYSTEM. He has completed a rather detailed analysis and this article reveals many secrets of the software. Tom is the software expert who developed TINY BASIC for the 6800 (See last months Newsletter for details).

BOOTSTRAP THE 8080 - Users of 8080's are not overlooked in this issue. The DATA FILE contains a unique bootstrap written by Lichen Wang. It's a short program that can save plenty of operator time.

BAY AREA CLUBS - Richard Lindberg reveals the meaning of BAMUG by answering the question WHAT'S A BAMUG?

It seems you can take your computer to meet a computer if you visit the LO*OP CENTER. Read NOTES FROM THE NORTH.

Computer clubs are invited to send material for the Newsletter and I will, as space permits, publish your items. This applies particularly if you are in the San Francisco area and don't have a newsletter going yet or just don't want the hassle that goes with producing one.

GREAT NEWS POSTAGE PAID THIS ISSUE - Postage for this issue of the Newsletter has been paid by a contribution of NATIONAL TELETYPewriter CORP., 207 Newtown Road, Plainview, NY 11803. Joe Gibbons, President, explains that his company has new and rebuilt TTY's from Model 28 through 40. They have other equipment available also. Lease/purchase arrangements can be made. Write to Joe at the above address and get a complete list of equipment and prices. In the event you want to telephone call 516-292-0444. Thank you Joe for the contribution.

WHAT IS YOUR SYSTEM? - In the Newsletter I have published software and hardware material of varied content but so far have not had an article on a "HOMEBREW SYSTEM". I am sure there would be plenty of interest in system descriptions. Could some of you work up a description of your system? Plan to use a page, two pages, three pages, or just a portion of a page as dictated by the complexity of your system or the time available to prepare the information. Naturally camera ready copy, if at all possible, would be a big help. Don't forget to answer the question people always ask; how do you use your system?

ERROR - Last months Newsletter attributed principle development of the INTERMETERS PL/M6800 compiler incorrectly. Dan Fylstra is the author of this compiler while Glenn Bingham does the marketing.
WHAT'S A BAMUG?
By Richard Lindberg

BAMUG stands for Bay Area Microprocessor's Users Group. It is one of the many computer hobby clubs which have sprung up like weeds in the past year. We meet in San Leandro on the first Thursday of each month. Meetings are held at the Great Western Savings building at E14th and Davis at 7:00 PM.

BAMUG is a relatively small group. Some of our members are also Homebrewers. Anyone is welcome to join us, but we are mostly interested in the Intel 8080. We are starting a software library which, though still small, contains some very useful programs.

We try to keep the meetings interesting. We have some demonstration at every meeting if possible. At the last meeting we had one of our members, Ken Jackman, tell us some of the concepts involved in Nim-like games. He is writing a book on the subject and presents it well with the aid of his vintage Nova. We expect that he will tell us more about this and other subjects in the future.

At the next meeting we will have George Morrow. He will be hauling odds and ends out of his basement to show us. Of particular interest is his inexpensive tape cassette interface. We also hope to get a preview of his 16K ALTAIR-IMSAI compatible RAM board.

The next meeting is May 7th. If you have a program to contribute to our library, please bring us a copy. We have voluntary dues of .50 per meeting, but you are welcome to attend without paying.

NOTES FROM THE NORTH

The SONOMA COUNTY MICRO COMPUTER CLUB is small but powerful. We are a group of several ALTAIR's, an IMSAI, a JOLT, two PDP-8's, an APPLE and some others on order. We all have people up and running.

We meet the first Tuesday in each month at LO*OP CENTER in Cotati. Meeting time is 7:30 PM. Any interested systems are invited to attend with their operators.

In April the APPLE 6502 system was our special guest. We are grateful to STEVE WOZNIAK for providing transportation.

BYE BYE BIRDIE, LO*OP CENTER CLASSIC PDP-8
LO*OP CENTER, 8099 La Plaza, Cotati, CA 94928
HOMEBREW COMPUTER CLUB MEETING 4/28/76 - An interesting meeting to say the least. A fascinating holographic display was demonstrated by Sharon from MULTIPLEX, 454 Shotwell, San Francisco, CA 94110, telephone 285-9035. Sharon is an artist and she is searching for someone with a video display capable of producing 360 degree graphics. She plans to put computer graphics and holographic art together into what I am sure will be a beautiful combination. Telephone Sharon if you can help with graphics.

Carl Helmers, the editor of BYTE, was on hand and met many of the HCC members. Everyone was delighted that Carl scheduled a visit to the meeting as part of his busy schedule while on the West Coast.

Marty of M&R ENTERPRISES, P.O. Box 1011, Sunnyvale, CA 94088, telephone 408-738-3772 displayed his ASTRO 2000 boards and promised he would soon have one of the best BASIC programs available to the hobbyist.

Preliminary specifications of the VDM-1 STAND ALONE ADAPTER CARD were given out by Lee Felsenstein. He would like your comments on the specs. Send them to LGC ENGINEERING, 1807 Delaware Street, Berkeley, CA 94703, telephone 415-845-4736. For those people who can't attend the HCC meetings, Lee moderates most of the proceedings in a style that keeps member interest and participation at a high level.

Everyone interested in hobbyist computers is invited to the HCC meetings. No dues or fees; however, a donation toward support of the Newsletter to cover printing, postage, etc. will be greatly appreciated.

CLUB MEETING SCHEDULE - HOMEBREW COMPUTER CLUB meetings are held every two weeks; May 12, 1976, May 26, 1976, June 9, 1976, etc. Location is Stanford Linear Accelerator Center, Menlo Park, California. Meetings begin at 7:00 P.M. Ask the guard at the gate for directions to the meeting room.

SOLID STATE MUSIC - Effective May 1, 1976 hours are as follows: Walk-in hours Tuesday, Thursday, and Friday 4:00 P.M. - 7:00 P.M., Saturday 10:00 A.M. - 5:30 P.M. and Will-call hours Tuesday, Thursday, and Saturday 10:00 A.M. - 5:30 P.M. Purchasers of the AMI prototype board will be able to get the additional parts they need from SOLID STATE MUSIC. Contact John Burgoon, 2102A Walsh Ave., Santa Clara, CA 95050, telephone 408-246-2707.

COMPUTER NOTES - Andrea Lewis has taken over the editorship of the ALTAIR Users Group publication COMPUTER NOTES. MITS has sent a number of copies of the current issue for distribution to HCC members. I will have them at the May meetings as long as the supply lasts. Andrea is doing a great job. The publication runs 16 pages and is full of information for ALTAIR users and others interested in microcomputers. For more information contact Andrea Lewis, Editor, c/o MITS, Inc., 2450 Alamo S. E., Albuquerque, NM 87106.

MINI MICRO MART - During the past several weeks the mail has been full of material from MINI MICRO MART, 1618 James Street, Syracuse, NY 13203, telephone 315-422-4467. New product data, samples of documentation, club discount opportunities, etc. Maury Goldberg feels their problems are behind them and that they can properly respond to hobbyists needs. Write to him for a list of current products. Tell me if your orders are handled the way you like. Also tell me if you have problems.
These two programs were designed to co-reside in 9K of RAM with 762 bytes of workspace remaining for edit buffer or symbol table. If you have more, your workspace is thus larger. Also, both the editor and assembler are designed to be able to use the other's program space for additional workspace if desired. Figure 1 is a memory map of the system. The first two pages (locations 100-2B5) in these two programs are the same, and contain all of the I/O routines for both.

If location 300 is non-zero, the Editor will use the assembler program space for additional buffer; if location 1540 is non-zero, the Assembler will use the editor program space for additional buffer. The Editor is designed to use all available contiguous memory, but the Assembler will limit its symbol table to memory below 2000, or below the contents of 301-302.

The key to the I/O routines is a "transfer vector" starting in location 0100. There are 19 or so JMP instructions in this vector which permit access to all of the routines, so that the entry to the routines do not depend of their sizes. By careful analysis of these routines I have made the following determinations.

The I/O routines are designed to work with a Teletype ASR33 or Silent 700 with cassettes, where the same data is used for both printed and tape copy. The paper tape reader on the terminal may be controlled either by X-ON/X-OFF or by a relay connected to an I/O line on the computer. The punch is controlled by TAPE/TAPE or the program is instructed to make separate punch passes. There are two versions of the I/O routines, one for MIKBUG which uses a PIA for the TTY interface, and one for EXBUG, which uses an ACIA. MIKBUG is a crummy operating system (it was designed as an evaluation program), and Motorola is pretty tight about giving out information on EXBUG except with the sale of an Exorciser, so unless you have one of these in your system you are probably better off rewriting the I/O.

The tape I have seen was done for EXBUG, and since it is most likely to be the same as your copy, the following remarks apply to that version.

There are 31 references to upper memory in the I/O routines (none in the main part of the assembler and editor; I checked). Of these, there are four references to FF02
(locations 018B, 01CD, 023F, and 025F); this is apparently a flag to indicate whether the terminal is a Silent 700 or not, with 00= not. All four are TST instructions, which may be altered to test any zero memory location. There are two references to FF62 and one to FF53 (0154, 023C, and 0271), which seem to be related to the EXBUG I/O flags; they may be NOPed out (three NOPs will replace each reference).

It is assumed that location FCFD contains some kind of constant for the paper tape reader control; this is fetched in locations 01DE and 0205. Bit 5 of this byte should be 0, and bit 6 should be 1 if RTS of the ACIA is used for reader control, and zero otherwise; bits 0, 2, and 4 should be ones. It may be convenient to NOP out these two fetches, and replace the opcodes of the respective following instructions with LDA A immediate (instead of AND A). Note that bit 5 enables interrupts, which may not be desired, so #15 should be used instead of #35.

The ACIA is assumed to be at locations FCF4-FCF5. Two references to its control register are used to turn the paper tape reader on and off (locations 01E3 and 020A). Two references to the input data register (locations 0212 and 0215) occur to clear out any remaining input after turning off the reader. Location 0287 checks the status register for input during a timeout loop, and 028D reads the data character when it comes. If your ACIA is in a different location, you may alter the addresses in these instructions.

There are seven subroutines in EXBUG which are called from these I/O routines. Of these, four directly must have correspondence in your system. They input a single character from the ACIA, output a single character, and exit to the monitor or operating system; they are summarized in Table 2.

Table 2 also shows the calls for composite operations. These, however, may be encoded easily to rely only on the single character input and output functions. The discussion to follow describes how these functions may be encoded in an arbitrary system.

Assume that your operating system provides a single call each for one character input "INPUT", and one character output "OUTPUT". We also assume that if your terminal requires nulls between the CR and LF that your operating system will detect the CR and insert the necessary nulls in the output call. If you have separate devices for file input and output which do not go through the terminal I/O, the beginning of

- File/Terminal I/O separator

<table>
<thead>
<tr>
<th>INPUT</th>
<th>TST</th>
<th>RDRON</th>
<th>Check if File Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRA</td>
<td>FILE IN</td>
<td>Yes, file.</td>
<td></td>
</tr>
<tr>
<td>OUTPUT</td>
<td>CMP A #11</td>
<td>Reader-On Control?</td>
<td></td>
</tr>
<tr>
<td>BRA</td>
<td>FILE IN</td>
<td>Yes, file.</td>
<td></td>
</tr>
<tr>
<td>CMP A #12</td>
<td>TAPe-On?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRA</td>
<td>FILE IN</td>
<td>Yes, file.</td>
<td></td>
</tr>
<tr>
<td>CMP A #13</td>
<td>Reader Off?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRA</td>
<td>FILE IN</td>
<td>Yes, file.</td>
<td></td>
</tr>
<tr>
<td>XON</td>
<td>INC</td>
<td>RDRON</td>
<td>Turn on &quot;reader&quot; =file input.</td>
</tr>
<tr>
<td>RTS</td>
<td>XOFF</td>
<td>CLR</td>
<td>RDRON</td>
</tr>
<tr>
<td>RTS</td>
<td>TCPE</td>
<td>CLR</td>
<td>RDRON</td>
</tr>
<tr>
<td>RTS</td>
<td>NTAF E</td>
<td>CLR</td>
<td>RDRON</td>
</tr>
</tbody>
</table>

![Figure 3. Suggestion for Terminal/File separation](image)

<table>
<thead>
<tr>
<th>LOCK</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>Beginning of Assembler</td>
</tr>
<tr>
<td>0103</td>
<td>Beginning of Editor</td>
</tr>
<tr>
<td>0105</td>
<td>Initialize I/O flags in P2, P5, P6</td>
</tr>
<tr>
<td>0109</td>
<td>Output character in A to Punch, adding LF to CR</td>
</tr>
<tr>
<td>010C</td>
<td>Punch character string pointed to by X, ended by 04</td>
</tr>
<tr>
<td>010F</td>
<td>Input (buffered) character from reader to A</td>
</tr>
<tr>
<td>0112</td>
<td>Read a line into input buffer; first char to A</td>
</tr>
<tr>
<td>0115</td>
<td>Type character in A, adding LF to CR, conv Tab to space</td>
</tr>
<tr>
<td>0118</td>
<td>Input character (from keyboard) to A</td>
</tr>
<tr>
<td>011B</td>
<td>Input character to A (with no echo?)</td>
</tr>
<tr>
<td>011C</td>
<td>Type text string on new line</td>
</tr>
<tr>
<td>0121</td>
<td>Type text string pointed to by X, ended by 04</td>
</tr>
<tr>
<td>0124</td>
<td>Type CR &amp; LF</td>
</tr>
<tr>
<td>0127</td>
<td>Punch 32 Nulls for leader</td>
</tr>
<tr>
<td>012A</td>
<td>Turn on Punch (if not already on)</td>
</tr>
<tr>
<td>012D</td>
<td>Turn off Punch</td>
</tr>
<tr>
<td>0130</td>
<td>Jump to monitor (exit from program)</td>
</tr>
<tr>
<td>0133</td>
<td>Jump to 0112</td>
</tr>
<tr>
<td>0136</td>
<td>Jump to 0121</td>
</tr>
</tbody>
</table>

Table 1. I/O Routines Jump Table
INPUT and OUTPUT will be complicated sufficiently to distinguish file I/O from terminal I/O as described a little later.

Figure 2 is a sample program to provide the functions supplied by EXBUG. The various entries are labeled in the listing by the address in EXBUG which provides the same function (see Table 2). Normally you would include these routines in your operating system monitor, and patch jumps to them from the I/O linkages in lower memory. If you do this, you probably also wish to modify the references to upper memory described earlier to point to your own ACIA.

For a completely generalized I/O system, you need to set up separate calls for I/O on the terminal-console and I/O on the data file. Operating system commands can then be used to designate different devices as file or terminal, so that for example, you can assemble from cassette or paper tape, edit from one device to another, etc. Figure 3 shows how two flag bytes in RAM somewhere (labeled PCHON and RDRON) can be used to direct a single stream of output to either the terminal or the output file, and conversely to accept a byte from either the terminal or the input file. Note that this routine traps the Teletype Tape control to select the data stream, and does not pass these controls through to the output. If you are going to do this, you should delete references to the ACIA from the I/O routines in lower memory (see Table 3), and insert the following instruction in the input routine in lower memory:

01EF  JSR  FILEIN

Table 2. Assembler and Editor calls to EXBUG

<table>
<thead>
<tr>
<th>LOCN</th>
<th>INSTRUCTION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0157</td>
<td>F15</td>
<td>Accept input character from terminal</td>
</tr>
<tr>
<td>0177</td>
<td>F18</td>
<td>Type character to terminal (or punch)</td>
</tr>
<tr>
<td>0165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0130</td>
<td>F564</td>
<td>Monitor exit</td>
</tr>
<tr>
<td>01DA*</td>
<td>F9CF</td>
<td>Output character to ACIA</td>
</tr>
<tr>
<td>01D9*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0248*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0256*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>026A*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011E</td>
<td>F24</td>
<td>Type string on new line</td>
</tr>
<tr>
<td>0121</td>
<td>F27</td>
<td>Type string</td>
</tr>
<tr>
<td>0124</td>
<td>F21</td>
<td>Type CR &amp; LF</td>
</tr>
</tbody>
</table>

Note that the monitor subroutine FILEIN (as all good monitor routines) should not alter the Index register or the B accumulator. On return, FILEIN leaves the Carry flag set to one to indicate an end-of-file, otherwise the carry is cleared. In the case of an end-of-file, the A should still contain some appropriate character (I realize this is not the proper way of designing a system--so complain at Motorola) such as hex 1A (Control Z, which is not the ASCII EOF, but is used by these programs as an EOF character).

It should be noted that most of this information was inferred by analysis of the dis-assembled software, and lacks some of the insight afforded by true annotated assembly listings. Therefore, no warranty is made for the accuracy of the interpretations presented here. I have attempted to verify my findings, and it does seem to work. Please let me know if you find any differences or exceptions to this analysis.
BAD BIT GETTERS (OR MEMORY TEST PROGRAMS) - RAY BOAZ

Every computer system needs a memory test program or two to ensure a high level of confidence in the memory system hardware. One bad memory bit can send a program off to Never-Never Land. The memory test programs listed here are for use with 6800 systems operating with MIKBUG as a system monitor. MIKBUG is a simple monitor (as it was meant to be) which has many useful subroutines. Several of them are made use of in these memory test programs.

The terms used herein are consistent with the 6800 nomenclature. A and B are the two accumulators, X is the index register, PC is the program counter, SP is the stack pointer, CCR is the condition code register, and M is the memory location of interest.

Of the two programs, the shortest, MT1, is written to be used in the 6810 RAM used by MIKBUG as a scratch-pad. It is 27 bytes long and fits into address space A060 to A07A. This should work well for the SWTP 6800 Computer Systems. The second, MT2, is a more general test program but takes up 58 bytes total. The start and end addresses in MT1 are direct operands and therefore, shown blank in the listing. MT2 uses locations A002-3 for start and A004-5 for end addresses. So these locations must be loaded before the program is started. Also in MT2 the I/O interrupt pointer locations (A000-1) are used so if applicable it must be set after running MT2. In general both MT1 and MT2 operate the same until an error is found.

In both programs a store is made to an M start address, then the data is fetched back, compared to good data, and if true, it is incremented to test M again with A+1. This continues for all 256 bit combinations - 00 to FF. Then X is incremented and the next location is tested. This continues until the address is encountered.

In MT1, on finding an error, a branch to MIKBUG software interrupt is made. Which results in the status registers being printed out as follows: CCR-B-A-X-PC-SP. B is the good data, A is the error data, and X is the address with the error. If testing 1K bytes, 2000-23FF, and an error was found which resulted in B=00, A=00 and X=007, the chip containing data bit 7 (MSB) at M=2007 is bad. M+1 must be loaded as the new start address to continue the test to end address.

MT2 on finding an error goes to a routine to print A-B-X, the continues to the next M until end address is reached. Here again A=error data, B=good data, and X=M error.

Both of these programs have worked well as a memory tester and as chip tester for 2102 type memory chips.

MT1

```
A060 CE   START
A063 4F   LDX
A064 5F   CLR A
A065 06   CLR B
A066 A7 00  TAP
A068 A6 00  STAA
A06A 11   LDAA
A068 26 00  CBA
```

LOAD X WITH START ADDR
CLEAR A
CLEAR B
CLEAR CCR
STORE A AT X ADDR
LOAD A WITH DATA JUST STORED.
COMPARE B-A
IF NOT SAME BRANCH TO ERROR
A06D 4C  INCA  ELSE INCREMENT A
A06E 5C  INCB  INCREMENT B
A06F 26 F5  BNE  IF B NOT EQUAL TO 00 BRANCH TO NVAL
A071 0B  INX  ELSE INCREMENT X
A072 BC  CPX  COMPARE X TO END ADDR
A075 26 EC  BNE  IF NOT SAME BRANCH TO OMT
A077 7E E0E3  JMP  ELSE JUMP TO MIKBUG CONTROL
A07A 3F  ERROR  SWI  PRINT ERROR STATUS

MT2

0000 FE A002  START  LDX  LOAD X WITH START ADDR
0003 4F  OMT  CLRA  CLEAR A
0004 5F  CLR B  CLEAR B
0005 0B  TAP  CLEAR CCR
0006 A7 00  NVAL  STAA  STORE A AT X ADDR
0008 A6 00  LOA A  LOAD A WITH DATA JUST STORED
000A 11  CBA  COMPARE B-A
000B 26 0D  BNE  IF NOT SAME BRANCH TO ERROR
000D 4C  INCA  ELSE INCREMENT A
000E 5C  INCB  INCREMENT B
000F 26 F5  BNE  IF B NOT EQUAL TO 00 BRANCH TO NVAL
0011 0B  INX  ELSE INCREMENT X
0012 BC A004  CPX  COMPARE X TO END ADDR+4 AT A004
0015 26 EC  BNE  IF NOT SAME BRANCH TO OMT
0017 7E E0E3  JMP  ELSE JUMP TO MIKBUG CONTROL
001A 37 A000  ERROR  STAA  STORE A AT A000
001D F7 A001  STAB  STORE B AT A001
0020 FF A002  STX  STORE X AT A002
0023 CE E19D  LDX  LOAD X WITH ADDR OF CE/LF/X STRING
0026 60 E07E  JSR  JUMP TO PRINT CR/LF/X
0029 CE A000  LDX  LOAD X WITH ADDR OF ERROR DATA
002C BD E0CA  JSR  JUMP TO PRINT ERROR (A)
002F BD E0CA  JSR  JUMP TO PRINT DATA (B)
0032 BD E0CB  JSR  JUMP TO PRINT ERROR ADDR (X)
0035 7C A003  INC  INCREMENT X FOR NEW START ADDR
0038 20 C6  BRA  BRANCH TO START
BOOTSTRAP THE 8080

Lichen Wang

If your 8080 µP system is not equipped with non-volatile memory, you probably have to reload the memory from time to time. To read the Intel hex format paper tape, you need to key in a loader of some eighty odd bytes long. This is rather tedious and often leads to error. Altair Basic has a bootstrap loader of twenty or twenty one bytes long. In principle, you can use this bootstrap to load in your own loader which will then load in your program. However, since Mr. Bill Gates claims that he did not get payed enough and is in the mood of calling people thieves. (See HRCC newsletter Y2-1.) I decided to code one myself. What comes out is a bootstrap of sixteen bytes long. This is still too long, maybe our professional experts can make it shorter. For the time being you are welcome to copy mine and I will not call you a thief (this includes Mr. Gates).

The part that you have to key in look like this:

0000 DB00 READ IN 0 ;READ AND
0002 E520 ANI 20H ;MASK THE STATUS BIT
0004 CA0000 JZ READ ;NOT READY YET
0007 DB01 IN 1 ;READY, READ IN A FRAME
0009 010900 HERE LXI B,HERE ;LATER BECOMES INX B, STAX B, CPI
000C 02 STAX B ;LATER BECOMES FF
000D C30000 JMP READ ;LATER BECOMES JNZ READ

And the paper tape should have the binary equivalent of the hex numbers shown below:

01 01 ... 01 03 02 FE FF C2 00 00 XX XX ...... XX XX FF
<- leader -> <- bootstrapping -> <- your loader -> marker

Where your loader is punched in binary format on the paper tape between the 00 and the FF denoted by XX XX ...... XX XX. Your loader cannot have any byte with the value FF. The marker FF tells the bootstrap to start your loader starting at 10H. After the FF, the paper tape is read by your loader. Use whatever format you want.

If your loader cannot be loaded at 10H, then you will have to write another loader which can be loaded at 10H, use it to load in your first loader to load in your program. This sounds very confusing, but that is how bootstrap works. Have your ever tried to get yourself off the ground by pulling your bootstrap?

Incidentally, the I/O ports at locations 1 and 8, the status bit mask at 3, and the jump condition at 4 may have to be changed for different I/O interface board. Your loader should copy them from the bootstrap rather than setting them up on its own. (Or, you can code your loader to change location 9 to RET and use READ as your input routine.) This way the same paper tape can be used on different machines. To carry this one step further, your program should in turn copy them from your loader, so that it too can work on different machines.
BULLETIN BOARD


TELETYPES - Models 28 through 40, new or rebuilt, RO's, KSR's, ASR's. All available immediately. NATIONAL TELETYPETRITER CORP., 207 Newtown Rd., Plainview, NY 11803. Contact Joe Gibbons at 516-293-0444.

WANTED - People who are interested in use of home computers for text or word processing. I have not found the technical advice to put together a configuration, or the software. I will try to stimulate and coordinate interest in a practical home word processor. Send me your ideas. Robert H. Edmonds, P.O. Box 464, Estudillo Station, San Leandro, CA 94577. (Count me in on this - the Newsletter needs a system - Bob Reiling)

SELECTRICS - IBM 1052-2 (I/O) $850; 1053-2 (R/O) $580; you pay shipping; these are heavy-duty Selectrics; IBM Maintenance eligible; top condition; used by my hospital in research project less than 100 hours; full set of manuals; WM. J. Schenker, M.D., 2086 Essenay, Walnut Creek, CA 94596, 415-939-6295.

NEW NAME AND ADDRESS - Starting May 5, 1976 the Arrowhead Computer Co. has a new name and location: THE COMPUTER STORE, 820 Broadway, Santa Monica, CA 90401, 213-451-0713.