

# DR. WANG'S PALO ALTO TINY BASIC

*By Roger Rauskolb*

Tiny Basic was first proposed in Dr. Dobb's Journal. Li-Chen Wang's version of Palo Alto Tiny Basic originally appeared in Issue No. 5, May 1976 of Dr. Dobb's Journal. A complete listing was printed, but Dr. Wang did the assembly on an IBM computer and he defined different mnemonics. In order to assemble with an Intel Compatible Assembler a translation of most mnemonics had to be performed.

I had developed my own system, which consists of two small p.c. boards, one containing the 8080 CPU, 4k of 2708 type EROM and 1K of RAM. The other PCB contained the RS-232 Interface using the Intel 8251. So I wanted to change the I/O section.

If you want to change I/O, all routines are contained in the OUTC and CHKIO routines. My system uses the following configuration:

2708	EROMS	0000H	TO	07FFH
1k	OF RAM	1000H	TO	13FFH
8251	DATA PORT	0FAH		
8251	STATUS PORT			

Command Instruction      27H = 2 stop bits parity disabled.  
8 bit characters.  
Baud Rate Factor of 04.

Mode Instruction      0CFH = No Hunt Mode. Not (RTS) forced  
to 0. Receive Enabled Data Terminal Ready Transmit Enabled.

Transmitter Ready Status Bit = Bit 0 (01H)

Receiver Buffer Ready Status Bit = Bit 1 (02H)

The program is contained in locations 0000H to 0768H.

In 1K of RAM 847 bytes are left over for program.

Tiny Basic does not offer much in terms of functions and general mathematical capabilities. But it is great to teach programming basics to children (and adults) and for games, since it has the RND function. It takes up little memory space and executes a lot faster than other basics.

Dr. Wang was very helpful and assisted me all the way. Some errors were eliminated. I appreciate his help and he deserves a lot of credit for his implementation of Tiny Basic.

See Microcomputer Software Depository Program Index for Copies of this program.

## THE TINY BASIC LANGUAGE

### Numbers

In Tiny Basic, all numbers are integers and must be less than or equal to 32767.

### Variables

There are 26 variables denoted by letters A through Z. There is also a single array @(1). The dimension of this array (i.e., the range of value of the index 1) is set automatically to make use of all the memory space that is left unused by the program. (i.e., 0 through SIZE/2, see SIZE function below.)

### Functions

For the time being, there are only 3 functions:

ABS(X) gives the absolute value of X.

RND(X) gives a random number between 1 and X (inclusive).

SIZE gives the number of bytes left unused by the program.

### Arithmetic and Compare Operators

/ divide. Note that since we have integers only,  $\frac{2}{3}=0$ .

\* multiply.

- subtract.

+ add.

> compare if greater than.

< compare if less than.

= compare if equal to. Note that to certain versions of Basic "LET A=B=0" means "set both A and B to 0". To this version of Tiny Basic, it means "set A to the result of comparing B with 0".

# compare if not equal to.

>= compare if greater than or equal to.

<= compare if less than or equal to.

+, -, \*, and / operations result in a value of between -32767 and 32767. All compare operators result in a 1 if true and a 0 if not true.

### Expressions

Expressions are formed with numbers, variables, and functions with arithmetic and compare operators between them. + and - signs can also be used at the beginning of an expression. The value of an expression is evaluated from left to right, except that \* and / are always done first, and then + and -, and then compare operators. Parentheses can also be used to alter the order of evaluation.

### Statements

A Tiny Basic statement consists of a statement

number between 1 and 32767 followed by one or more commands. Commands in the same statement are separated by a semi-colon ";". "GOTO", "STOP", and "RETURN" commands must be the last command in any given statement.

### Program

A Tiny Basic program consists of one or more statements. When a direct command "RUN" is issued, the statement with the lowest statement number is executed first, then the one with the next lowest statement number, etc. However, the "GOTO", "GOSUB", "STOP", and "RETURN" commands can alter this normal sequence. Within the statement, execution of the commands is from left to right. The "IF" command can cause the execution of all the commands to its right in the same statement to be skipped over.

### Commands

Tiny Basic commands are listed below with examples. Remember that commands can be concatenated with semi-colons. In order to store the statement, you must also have a statement number in front of the commands. The statement number and the concatenation are not shown in the examples.

#### REM or REMARK Command

REM anything goes  
This line will be ignored by TBI.

#### LET Command

LET A=234-5\*6, A=A/2, X=A-100,  
@(X+9)=A-1

will set the variable A to the value of the expression  $234-5*6$  (i.e., 204), set the variable A (again) to the value of the expression  $A/2$  (i.e., 102), set the variable X to the value of the expression  $A-100$  (i.e., 2), and then set the variable @(11) to 101 (where 11 is the value of the expression  $X+9$  and 101 is the value of the expression  $A-1$ ).

LET U=A#B, V=(A>B)\*X+(A<B)\*Y

will set the variable U to either 1 or 0 depending on whether A is not equal to or is equal to B; and set the variable V to either X, Y or 0 depending on whether A is greater than, less than, or equal to B.

#### PRINT Command

PRINT

will cause a carriage-return (CR) and a line-feed (LF) on the output device.

PRINT A\*3+1, "ABC 123 !@#", 'CBA '

will print the value of the expression  $A*3+1$  (i.e., 307), the string of characters "ABC 123 !@#", and the string "CBA ", and then a CR-LF. Note that either single or double quotes can be used to quote strings, but pairs must be matched.

PRINT A\*3+1, "ABC 123 !@#", 'CBA '

will produce the same output as before, except that there is no CR-LF after the last item is printed. This enables the program to continue printing on the same line with another "PRINT".

PRINT A, B, #3, C, D, E, #10, F, G

will print the values of A and B in 6 spaces, the values of C, D, and E in 3 spaces, and the values of F and G in 10 spaces. If there are not enough spaces specified for a given value to be printed, the value will be printed with enough spaces anyway.

PRINT 'ABC', ←, 'XXX'

will print the string "ABC", a CR without a LF, and then the string "XXX" (over the ABC) followed by a CR-LF.

#### INPUT Command

INPUT A, B

When this command is executed, Tiny Basic will print "A:" and wait to read in an expression from the input device. The variable A will be set to the value of this expression. Then "B:" is printed and variable B is set to the value of the next expression read from the input device. Note that not only numbers, but also expressions can be read as input.

INPUT 'WHAT IS THE WEIGHT', A, "AND SIZE", B

This is the same as the command above, except the prompt "A:" is replaced by "WHAT IS THE WEIGHT:" and the prompt "B:" is replaced by "AND SIZE:". Again, both single and double quotes can be used as long as they are matched.

INPUT A, 'STRING', ←, "ANOTHER STRING", B

The strings and the ← have the same effect as in "PRINT".

#### IF Command

IF A<B LET X=3; PRINT 'THIS STRING'

will test the value of the expression  $A<B$ . If it is not zero (i.e., if it is true), the commands in the rest of this statement will be executed. If the value of the expression is zero (i.e., if it is not true), the rest of this statement will be skipped over and execution continues at next statement. Note that the word "THEN" is not used.

#### GOTO Command

GOTO 120

will cause the execution to jump to statement 120. Note that GOTO command cannot be followed by a semi-colon and other commands. It must be ended with a CR.

GOTO A\*10+B

will cause the execution to jump to a different statement number as computed from the value of the expression.

#### GOSUB and RETURN Commands

GOSUB command is similar to GOTO command

except that: a) the current statement number and position within the statement is remembered; and b) a semi-colon and other commands can follow it in the same statement.

**GOSUB 120**

will cause the execution to jump to statement 120.

**GOSUB A\*10+B**

will cause the execution to jump to different statements as computed from the value of the expression  $A*10+B$ .

**RETURN**

A RETURN command must be the last command in a statement and followed by a CR. When a RETURN command is encountered, it will cause the execution to jump back to the command following the most recent GOSUB command.

GOSUB can be nested. The depth of nesting is limited only by the stack space.

**LIST**

will print out all the statements in numerical order.

**LIST 120**

will print out all the statements in numerical order 120.

**NEW**

will delete all the statements.

**Stopping the Execution**

The execution of program or listing of program can be stopped by the Control-C key on the input device.

**Abbreviation and blanks**

You may use blanks freely, except that numbers, command key words, and function names can not have embedded blanks.

You can truncate all command key words and function names and follow each by a period. "P.", "PR.", "PRI.", and "PRIN." all stand for "PRINT." Also the word LET in LET command can be omitted. The "shortest" abbreviation for all the key words are as follows:

A.=ABS	F.=FOR	GOS.=GOSUB	G.=GOTO
IF=IF	IN.=INPUT	L.=LIST	N.=NEW
N.=NEXT	P.=PRINT	REM=REMARK	R.=RETURN
R.=RND	R.=RUN	S.=SIZE	S.=STEP
S.=STOP	TO=TO		
Implied=LET			

**Control of Output Device**

The Control-O key on the input device can be used to turn the output device ON and OFF. This is useful when you want to read in a program punched on paper tape.

To produce such a paper tape, type "LIST" without CR. Turn on the paper tape punch and type a few Control-Shift-P's and then a CR. When listing is finished, type more Control-Shift-P's and turn off the punch.

To read back such a paper tape, type "NEW," CR, and Control-O, then turn on the paper tape reader.

When the paper tape is finished, turn it off and type a Control-O again.

Control-Shift-P's and turn off the punch.

To read back such a paper tape, type "NEW," CR, and Control-O, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-O. then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control O again.

**Error Report**

There are only three error conditions in TINY BASIC. The statement with the error is printed out with a question mark inserted at the point where the error is detected.

(1) WHAT? means it does not understand you. Example:

WHAT? 210 P?TINT "THIS" where PRINT is mistyped

WHAT? 260 LET A=B+3, C=(3+4), X=4

(2) HOW? means it understands you but does not know how to do it.

HOW? 310LET A=B\*C?+2 where B\*C is larger than 32767

HOW? 380 GOTO 412? where 412 dose not exist

(3) SORRY means it understands you and knows how to do it but there is not enough memory to do it.

**Error Corrections**

If you notice an error in typing before you hit the CR, you can delete the last character by the Rub-Out key or delete the entire line by the Alt-Mode key. Tiny basic will echo a back-slash for each Rub-Out. Echo for Alt-Mode consists of a LF, a CR, and an up-arrow.

To correct a statement, you can retype the statement number and the correct commands. Tiny Basic will replace the old statement with the new one.

To delete a statement, type the statement number and a CR only.

Verify the corrections by "LIST nnnn" and hit the Control-C key while the line is being printed.

**FOR and NEXT Commands**

**FOR X=A+1 TO 3\*B STEP C-1**

The variable X is set to the value of the expression  $A+1$ . The values of the expressions (not the expressions themselves)  $3*B$  and  $C-1$  are remembered. The name of the variable X, the statement number and the position of this command within the statement are also remembered. Execution then continues the normal way until a NEXT command is encountered.

The STEP can be positive, negative or even zero. The word STEP and the expression following it can be omitted if the desired STEP is +1.

**NEXT X**

The name of the variable (X) is checked with that of the most recent FOR command. If they do not agree, that FOR is terminated and the next recent FOR is checked, etc. When a match is found, this variable will be set to its current value plus the value of the STEP expression saved by the FOR command. The updated

value is then compared with the value of the TO expression also saved by the FOR command. If this is within the limit, execution will jump back to the command following the FOR command. If this is outside the limit, execution continues following the NEXT command itself.

FOR can be nested. The depth of nesting is limited only by the stack space. If a new FOR command with the same control variable as that of an old FOR command is encountered, the old FOR will be terminated automatically.

**STOP Command**

**STOP**

This command stops the execution of the program and returns control to direct commands from the input device. It can appear many times in a program but must be the last command in any given statement, i.e., it cannot be followed by semi-colon and other commands.

**Direct Commands**

As defined earlier, a statement consists of a statement number followed by commands. If the statement number is missing, or if it is 0, the commands will be executed after you have typed the CR. All the commands described above can be used as direct commands. There are three more commands that can be used as direct command but not as part of a statement:

**RUN**

will start to execute the program starting at the lowest statement number.

See Microcomputer Software Depository Program Index for copies of this program.

```

*****
;
; TINY BASIC FOR INTEL 8080
; VERSION 2.0
; BY LI-CHEN WANG
; MODIFIED AND TRANSLATED
; TO INTEL MNEMONICS
; BY ROGER RAUSKOLB
; 10 OCTOBER, 1976
; @COPYLEFT
; ALL WRONGS RESERVED
;
*****
;
; *** ZERO PAGE SUBROUTINES ***
;
; THE 8080 INSTRUCTION SET LETS YOU HAVE 8 ROUTINES IN LOW
; MEMORY THAT MAY BE CALLED BY RST N, N BEING 0 THROUGH 7
; THIS IS A ONE BYTE INSTRUCTION AND HAS THE SAME POWER AS
; THE THREE BYTE INSTRUCTION CALL LLHH - TINY BASIC WILL
; USE RST 0 AS START AND RST 1 THROUGH RST 7 FOR
; THE SEVEN MOST FREQUENTLY USED SUBROUTINES.
; TWO OTHER SUBROUTINES (CRLF AND TSTNUM) ARE ALSO IN THIS
; SECTION. THEY CAN BE REACHED ONLY BY 3-BYTE CALLS.
;
DWA MACRO WHERE
DB (WHERE SHR 8) +128
DB (WHERE AND 0FFH)
ENDM
ORG 0H
0000 START:
0000 310014 LXI SP,STACK ;*** COLD START ***
0001 2EFF MVI A,0FFH
0005 C34206 JMP INIT
;
0008 E3 XTHL ;*** TSTC OR RST 1 ***
0009 EF RST 5 ;IGNORE BLANKS AND
000A BE CMP M ;TEST CHARACTER
000E C26800 JMP TC1 ;REST OF THIS IS AT TC1
;
000E 3E00 CRLF: MVI A,CR ;*** CRLF ***
;
0010 F5 PUSH PSW ;*** OUTC OR RST 2 ***
0011 3A0010 LDR OC3W ;PRINT CHARACTER ONLY
0014 B7 ORA A ;IF OC3W SWITCH IS ON
0015 C36C06 JMP OC2 ;REST OF THIS IS AT OC2
;
0018 CD7103 CALL EXPR2 ;*** EXPR OR RST 3 ***
001B E5 PUSH H ;EVALUATE AN EXPRESSION
001C C22D03 JMP EXPR1 ;REST OF IT IS AT EXPR1

```

```

001F 57 DB 'W'
;
0020 7C MOV A,H ;*** COMP OR RST 4 ***
0021 8A CMP D ;COMPARE HL WITH DE
0022 C0 RNZ ;RETURN CORRECT C AND
0023 7D MOV A,L ;Z FLAGS
0024 8B CMP E ;BUT OLD A IS LOST
0025 C9 RET
0026 414E DB 'AN'
;
0028 1A LDAX D ;*** IGBLK/RST 5 ***
0029 FE20 SS1: CPI 20H ;IGNORE BLANKS
002B C0 RNZ ;IN TEXT (WHERE DE->)
002C 13 INX D ;AND RETURN THE FIRST
002D C32800 JMP SS1 ;NON-BLANK CHAR. IN A
;
0030 F1 POP PSW ;*** FINISH/RST 6 ***
0031 C0B204 CALL FIN ;CHECK END OF COMMAND
0034 C3C604 JMP QWHAT ;PRINT "WHAT?" IF WRONG
0037 47 DB 'G'
;
0038 EF RST 5 ;*** TSTV OR RST 7 ***
0039 D640 SUI 40H ;TEST VARIABLES
003B D8 RC ;C NOT A VARIABLE
003C C25800 JNZ TV1 ;NOT "0" ARRAY
003F 13 INX D ;IT IS THE "0" ARRAY
0040 CD1A04 CALL PARN ;0 SHOULD BE FOLLOWED
0042 29 DAD H ;BY (EXPR) AS ITS INDEX
0044 DA9F00 JC OHOW ;IS INDEX TOO BIG?
0047 D5 FUSH D ;WILL IT OVERWRITE
0048 EB XCHG ;TEXT?
0049 CD5904 CALL SIZE ;FIND SIZE OF FREE
004C E7 RST 4 ;AND CHECK THAT
004D DAF404 JC ASORRY ;IF SO, SAY "SORRY"
0050 216613 LXI H,VARBGN ;IF NOT, GET ADDRESS
0053 CD7C04 CALL SUBDE ;OF (EXPR) AND PUT IT
0056 D1 POP D ;IN HL
0057 C9 RET ;C FLAG IS CLEARED
0058 FE1B TV1: CPI 1BH ;NOT 0, IS IT A TO Z?
005A 3F CMC ;IF NOT RETURN C FLAG
005B D8 RC
005C 13 INX D ;IF A THROUGH Z
005D 216613 LXI H,VARBGN ;COMPUTE ADDRESS OF
0059 87 RLC ;THAT VARIABLE
0060 85 SPS ;AND RETURN IT IN HL
0062 6F MVI A,B ;WITH C FLAG CLEARED
0063 3E00 MVI A,0
0065 8C ADC H
0066 67 MOV H,A
0067 C9 RET
;
; TSTC XCH HL (SP) *** TSTC OR RST 1 ***
; ; THIS IS AT LOC. 8
; ; AND THEN JMP HERE
;
TC1: INX H ;COMPARE THE BYTE THAT
; J2 TC2 ;FOLLOWS THE RST INST.
; FUSH B ;WITH THE TEXT (DE->)
; MOV C,M ;IF NOT =, ADD THE 2ND
; MVI B,0 ;BYTE THAT FOLLOWS THE
; DAD B ;RST TO THE OLD PC
; POP B ;I.E., DO A RELATIVE
; DAD B ;JUMP IF NOT =
; TC2: INX D ;IF =, SKIP THOSE BYTES
; INX H ;AND CONTINUE
; XTHL
; RET
;
; TSTNUM:
; LXI H,0 ;*** TSTNUM ***
; MOV B,H ;TEST IF THE TEXT IS
; RST 5 ;A NUMBER
; TC1: CPI 30H ;IF NOT, RETURN 0 IN
; RC ;B AND HL
; CPI 3AH ;IF NUMBERS, CONVERT
; RNC ;TO BINARY IN HL AND
; MVI A,0F0H ;SET A TO # OF DIGITS
; ANA H ;IF H>255, THERE IS NO
; JNZ OHOW ;ROOM FOR NEXT DIGIT
; INR B ;B COUNTS # OF DIGITS
; FUSH B ;
; MOV B,H ;HL=10*HL+(NEW DIGIT)
; MOV C,L ;
; DAD H ;WHERE 10* IS DONE BY
; DAD H ;SHIFT AND ADD
; DAD B ;
; DAD H ;
; LDAX D ;AND (DIGIT) IS FROM
; INX D ;STRIPPING THE ASCII
; ADD L ;CODE
; MOV L,A ;
; MVI A,0 ;
; ADC H ;
; MOV H,A ;
; POP B ;
; LDAX D ;DO THIS DIGIT AFTER
; IF PSH D ;DIGIT 5 SAYS OVERFLOW
; OHOW: LXI D,HOW ;*** ERROR: "HOW?" ***
; AHOW: LXI D,HOW
; JMP ERROR
;
00A6 484F573F HOW: DB 'HOW?'
00A8 8C DB CR
00AB 4F4B OK: DB 'OK?'
00AD 0C DB CR
00AE 57484154 WHAT: DB 'WHAT?'
00B2 3F DB CR
00B3 0C DB CR
00B4 534F5252 SORRY: DB 'SORRY'
00B8 59 DB CR
00B9 0C DB CR
;
; *****
;
; *** MAIN ***
;
; THIS IS THE MAIN LOOP THAT COLLECTS THE TINY BASIC PROGRAM
; AND STORES IT IN THE MEMORY.
;
; AT START, IT PRINTS OUT "<CR><OK><CR>", AND INITIALIZES THE
; STACK AND SOME OTHER INTERNAL VARIABLES. THEN IT PROMPTS
; ">" AND READS A LINE. IF THE LINE STARTS WITH A NON-ZERO
; NUMBER, THIS NUMBER IS THE LINE NUMBER. THE LINE NUMBER
; (IN 16 BIT BINARY) AND THE REST OF THE LINE (INCLUDING CR)
; IS STORED IN THE MEMORY. IF A LINE WITH THE SAME LINE
; NUMBER IS ALREADY THERE, IT IS REPLACED BY THE NEW ONE. IF
; THE REST OF THE LINE CONSISTS OF A CR ONLY, IT IS NOT STORED
; AND ANY EXISTING LINE WITH THE SAME LINE NUMBER IS DELETED.
;
; AFTER A LINE IS INSERTED, REPLACED, OR DELETED, THE PROGRAM
; LOOPS BACK AND ASKS FOR ANOTHER LINE. THIS LOOP WILL BE
; TERMINATED WHEN IT READS A LINE WITH ZERO OR NO LINE
; NUMBER; AND CONTROL IS TRANSFERRED TO "DIRECT".

```



```

; *****
;
; *** FOR *** & NEXT ***
;
; 'FOR' HAS TWO FORMS:
; 'FOR VAR=EXP1 TO EXP2 STEP EXP3' AND 'FOR VAR=EXP1 TO EXP2'
; THE SECOND FORM MEANS THE SAME THING AS THE FIRST FORM WITH
; EXP3=1 (I.E., WITH A STEP OF +1.)
; TBI WILL FIND THE VARIABLE VAR, AND SET ITS VALUE TO THE
; CURRENT VALUE OF EXP1. IT ALSO EVALUATES EXP2 AND EXP3
; AND SAVE ALL THESE TOGETHER WITH THE TEXT POINTER ETC. IN
; THE 'FOR' SAVE AREA, WHICH CONSISTS OF 'LOPVAR', 'LOPINC',
; 'LOPLMT', 'LOPLN', AND 'LOPPT'. IF THERE IS ALREADY SOME-
; THING IN THE SAVE AREA (THIS IS INDICATED BY A NON-ZERO
; 'LOPVAR'), THEN THE OLD SAVE AREA IS SAVED IN THE STACK
; BEFORE THE NEW ONE OVERWRITES IT.
; TBI WILL THEN DIG IN THE STACK AND FIND OUT IF THIS SAME
; VARIABLE WAS USED IN ANOTHER CURRENTLY ACTIVE 'FOR' LOOP.
; IF THAT IS THE CASE, THEN THE OLD 'FOR' LOOP IS DEACTIVATED.
; (PURGED FROM THE STACK.)
;
; 'NEXT VAR' SERVES AS THE LOGICAL (NOT NECESSARILLY PHYSICAL)
; END OF THE 'FOR' LOOP. THE CONTROL VARIABLE VAR, IS CHECKED
; WITH THE 'LOPVAR'. IF THEY ARE NOT THE SAME, TBI DIGS IN
; THE STACK TO FIND THE RIGHT ONE AND PURGES ALL THOSE THAT
; DID NOT MATCH. EITHER WAY, TBI THEN ADDS THE 'STEP' TO
; THAT VARIABLE AND CHECKS THE RESULT WITH THE LIMIT. IF IT
; IS WITHIN THE LIMIT, CONTROL LOOPS BACK TO THE COMMAND
; FOLLOWING THE 'FOR'. IF OUTSIDE THE LIMIT, THE SAVE AREA
; IS PURGED AND EXECUTION CONTINUES.
;
01F8 CD1906 FOR: CALL PUSHA ;SAVE THE OLD SAVE AREA
01FB CDA004 CALL SETVAL ;SET THE CONTROL VAR.
01FE 2E DCX H ;HL IS ITS ADDRESS
01FF 220910 SHLD LOPVAR ;SAVE THAT
0202 211207 LXI H,TAB5-1 ;USE 'EXEC' TO LOOK
0205 C23B07 JMP EXEC ;FOR THE WORD 'TO'
0208 DF FR1: RST 3 ;EVALUATE THE LIMIT
0209 220910 SHLD LOPLMT ;SAVE THAT
020C 211907 LXI H,TAB6-1 ;USE 'EXEC' TO LOOK
020F C23B07 JMP EXEC ;FOR THE WORD 'STEP'
0212 DF FR2: RST 3 ;FOUND IT, GET STEP
0213 C31902 JMP FR4
0216 210100 LXI H,1H ;NOT FOUND, SET TO 1
0219 220E10 FR4: SHLD LOPINC ;SAVE THAT TOO
021C 2A0110 FR5: LHL CURRNT ;SAVE CURRENT LINE #
021F 220F10 SHLD LOPLN
0222 EB XCHG ;AND TEXT POINTER
0223 221110 SHLD LOPPT
0226 010A00 LXI B,0AH ;DIG INTO STACK TO
0229 2A0310 LHL LOPVAR ;FIND 'LOPVAR'
022C EB XCHG
;
022D 60 MOV H,B
022E 68 MOV L,B ;HL=0 NOW
022F 39 DAD SP ;HERE IS THE STACK
0230 0E DB 3EH
0231 09 FR7: DAD B ;EACH LEVEL IS 10 DEEP
0232 7E MOV A,M ;GET THAT OLD 'LOPVAR'
0233 21 INX H
0234 B6 ORA M
0235 CH5202 JZ FR8 ;0 SAYS NO MORE IN IT
0238 7E MOV A,M
0239 2E DCX H
023A BA CMP D ;SAME AS THIS ONE?
023B C23102 JNZ FR7
023E 7E MOV A,M ;THE OTHER HALF?
023F BE CMP E
0240 C23102 JNZ FR7
0243 EB XCHG ;YES, FOUND ONE
0244 210000 LXI H,0H
0247 39 DAD SP ;TRY TO MOVE SP
0248 44 MOV B,H
0249 4D MOV C,L
024A 210A00 LXI H,0AH
024D 19 DAD D
024E CDEE05 CALL MVDOWN ;AND PURGE 10 WORDS
0251 F9 SPHL ;IN THE STACK
0252 2A1110 FR8: LHL LOPPT ;JOB DONE, RESTORE DE
0255 EB XCHG
0256 F7 RST 6 ;AND CONTINUE
;
0257 FF NEXT: RST 7 ;GET ADDRESS OF VAR.
0258 DAC604 JC OMHAT ;NO VARIABLE, "WHAT?"
025B 220510 SHLD VARHXT ;YES, SAVE IT
025E 05 NX0: PUSH D ;SAVE TEXT POINTER
025F EB XCHG
0260 2A0910 LHL LOPVAR ;GET VAR. IN 'FOR'
0263 7C MOV A,H
0264 B5 ORA L ;0 SAYS NEVER HAD ONE
0265 CAC704 JZ AHAT ;SO WE ASK: "WHAT?"
0268 E7 RST 4 ;ELSE WE CHECK THEM
0269 CA7602 JZ NX3 ;OK, THEY AGREE
026C D1 POP D ;NO, LET'S SEE
026D CDFD05 CALL POPA ;PURGE CURRENT LOOP
0270 2A0510 LHL VARHXT ;AND POP ONE LEVEL
0273 C5E02 JMP NX0 ;GO CHECK AGAIN
0276 5E NX3: MOV E,M ;COME HERE WHEN AGREED
0277 23 INX H
0278 56 MOV D,M ;DE=VALUE OF VAR.
0279 2A0B10 LHL LOPINC
027C E5 PUSH H
027D 7C MOV A,H
;
027E AA XRA D
027F 7A MOV A,D
0280 19 DAD D ;ADD ONE STEP
0281 FA8202 JM NX4
0284 AC XRA H
0285 FAAA02 JM NX5
0288 EB XCHG
0289 2A0910 NX4: LHL LOPVAR ;PUT IT BACK
028C 73 MOV M,E
028D 23 INX H
028E 72 MOV M,D
028F 2A0D10 LHL LOPLMT ;HL=LIMIT
0292 F1 POP PSW ;OLD HL
0293 B7 ORA A
0294 F29B02 JP NX1 ;STEP > 0
0297 EB XCHG
0298 CD9B04 NX1: CALL OKHLDE ;COMPARE WITH LIMIT
029B D1 POP D ;RESTORE TEXT POINTER
029C DAC02 JC NX2 ;OUTSIDE LIMIT
029F 2A0F10 LHL LOPLN ;WITHIN LIMIT, GO
02A2 220110 SHLD CURRNT ;BACK TO THE SAVED
02A5 2A1110 LHL LOPPT ;'CURRNT' AND TEXT
02A8 EB XCHG ;POINTER
02A9 F7 RST 6
02AA E1 NX5: POP H
02AB D1 POP D
02AC CDFD05 NX2: CALL POPA ;PURGE THIS LOOP
02AF F7 RST 6
;
; *****
;
; *** REM *** IF *** INPUT *** & LET (& DEFLT) ***
;
; 'REM' CAN BE FOLLOWED BY ANYTHING AND IS IGNORED BY TBI.
; TBI TREATS IT LIKE AN 'IF' WITH A FALSE CONDITION.
;
; 'IF' IS FOLLOWED BY AN EXPR. AS A CONDITION AND ONE OR MORE
; COMMANDS (INCLUDING OTHER 'IF'S) SEPERATED BY SEMI-COLONS.
; NOTE THAT THE WORD 'THEN' IS NOT USED. TBI EVALUATES THE
; EXPR. IF IT IS NON-ZERO, EXECUTION CONTINUES. IF THE
; EXPR. IS ZERO, THE COMMANDS THAT FOLLOWS ARE IGNORED AND
; EXECUTION CONTINUES AT THE NEXT LINE.
;
; 'INPUT' COMMAND IS LIKE THE 'PRINT' COMMAND, AND IS FOLLOWED
; BY A LIST OF ITEMS. IF THE ITEM IS A STRING IN SINGLE OR
; DOUBLE QUOTES, OR IS A BACK-ARROW, IT HAS THE SAME EFFECT AS
; IN 'PRINT'. IF AN ITEM IS A VARIABLE, THIS VARIABLE NAME IS
; PRINTED OUT FOLLOWED BY A COLON. THEN TBI WAITS FOR AN
; EXPR. TO BE TYPED IN. THE VARIABLE IS THEN SET TO THE
; VALUE OF THIS EXPR. IF THE VARIABLE IS PRECEDED BY A STRING
; (AGAIN IN SINGLE OR DOUBLE QUOTES), THE STRING WILL BE
; PRINTED FOLLOWED BY A COLON. TBI THEN WAITS FOR INPUT EXPR.
; AND SET THE VARIABLE TO THE VALUE OF THE EXPR.
;
; IF THE INPUT EXPR. IS INVALID, TBI WILL PRINT "WHAT?",
; "HOW?" OR "SORRY" AND REPRINT THE PROMPT AND REDO THE INPUT.
; THE EXECUTION WILL NOT TERMINATE UNLESS YOU TYPE CONTROL-C.
; THIS IS HANDLED IN 'INPERR'.
;
; 'LET' IS FOLLOWED BY A LIST OF ITEMS SEPERATED BY COMMAS.
; EACH ITEM CONSISTS OF A VARIABLE, AN EQUAL SIGN, AND AN EXPR.
; TBI EVALUATES THE EXPR. AND SET THE VARIABLE TO THAT VALUE.
; TBI WILL ALSO HANDLE 'LET' COMMAND WITHOUT THE WORD 'LET'.
; THIS IS DONE BY 'DEFLT'.
;
; *****
;
02B0 210000 REM: LXI H,0H ;*** REM ***
02B3 3E DB 3EH
;
02B4 DF IFF: RST 3 ;*** IF ***
02B5 7C MOV A,H ;IS THE EXPR. =0?
02B6 B5 ORA L
02B7 C25701 JNZ RUNSHL ;NO, CONTINUE
02BA CD5605 CALL FND5KP ;YES, SKIP REST' OF LINE
02BD D25001 JNC RUNSHL
02C0 C3BA00 JMP RST6
;
02C3 2A0710 INPERR: LHL STKINP ;*** INPERR ***
02C6 F9 SPHL ;RESTORE OLD SP
02C7 E1 POP H ;AND OLD 'CURRNT'
02C8 220110 SHLD CURRNT
02CB D1 POP D ;AND OLD TEXT POINTER
02CC D1 POP D ;REDO INPUT
;
02CD 05 INPUT: ;*** INPUT ***
IP1: PUSH D ;SAVE IN CASE OF ERROR
CALL QTSTG ;CALL QTSTG
JMP IP2 ;NO
RST 7 ;YES, BUT FOLLOWED BY 'A
02D5 DA1502 JC IP4 ;VARIABLE? NO
02D8 C3EB02 JNC IP3 ;YES, INPUT VARIABLE
IP2: PUSH D ;SAVE FOR 'PRSTG'
RST 7 ;MUST BE VARIABLE NOW
JC OMHAT ;"WHAT?" IT IS NOT?
LDAX D ;GET READY FOR 'PRSTG'
MOV C,A
SUB A
STAX D
POP D
CALL PRSTG ;PRINT STRING AS PROMPT
;
02E8 79 MOV A,C ;RESTORE TEXT
02E9 1B DCX D
02EA 12 STAX D
02EB D5 IP3: PUSH D ;SAVE TEXT POINTER
02EC EB XCHG
02ED 2A0110 LHL CURRNT ;ALSO SAVE 'CURRNT'
02F0 E5 PUSH H
02F1 21CD02 LXI H,IP1 ;A NEGATIVE NUMBER
02F4 220110 SHLD CURRNT ;AS A FLAG
02F7 210000 LXI H,0H ;SAVE SP TOO
02FA 39 DAD SP
02FB 220710 SHLD STKINP
02FE D5 PUSH D ;OLD HL
02FF 3E3A MVI A,3AH ;PRINT THIS TOO
CALL GETLN ;AND GET A LINE
LXI D,BUFFER ;POINTS TO BUFFER
RST 3 ;EVALUATE INPUT
NOP ;CAN BE 'CALL ENDCHK'
NOP
NOP
POP D ;OK, GET OLD HL
XCHG
MOV M,E ;SAVE VALUE IN VAR.
INX H
MOV M,D
POP H ;GET OLD 'CURRNT'
0310 E1 POP H
0311 220110 SHLD CURRNT
0314 D1 POP D ;AND OLD TEXT POINTER
IP4: POP PSW ;PURGE JUNK IN STACK
RST 1 ;IS NEXT CH. 'A'
DE ' '
0317 2C DE ' '
0318 03 DE IP5-#-1
0319 C3CD02 JMP IP1 ;YES, MORE ITEMS.
IP5: RST 6
;
031D 1A DEFLT: LDAX D ;*** DEFLT ***
031E FE00 CPI 0 ;EMPTY LINE IS OK
0320 CAC003 JZ LT1 ;ELSE IT IS 'LET'
;
0323 CDA004 LET: CALL SETVAL ;*** LET ***
0326 CF RST 1 ;SET VALUE TO VAR.
0327 2C DE ' '
0328 03 DE LT1-#-1
0329 C23303 JMP LET ;ITEM BY ITEM
032C F7 RST 6 ;UNTIL FINISH
;
; *****
;
; *** EXPR ***
;
; 'EXPR' EVALUATES ARITHMETICAL OR LOGICAL EXPRESSIONS.
; <EXPR> ::= <EXPR2>
;
; <EXPR2> ::= <REL OP > <EXPR2>
;
; WHERE <REL OP > IS ONE OF THE OPERATORS IN TAB8 AND THE
; RESULT OF THESE OPERATIONS IS 1 IF TRUE AND 0 IF FALSE.
; <EXPR2> ::= (<+ OR ->) <EXPR2> (<+ OR ->) <EXPR3> (<+ OR ->)
; WHERE (<+ OR ->) ARE OPTIONAL AND (<+ OR ->) ARE OPTIONAL REPEATS.
; <EXPR3> ::= <EXPR4> (<< OR >>) <EXPR4> (<< OR >>)
; <EXPR4> ::= <VARIABLE>
;

```

Address	Instruction	Comment	Address	Instruction	Comment
		<FUNCTION>	03F7 D1	XP25: POP D	<AND TEXT POINTER
		<EXPR2>	03F8 7C	MOV A,H	<HL MUST BE +
		<EXPR2> IS RECURSIVE SO THAT VARIABLE "0" CAN HAVE AN <EXPR2>	03F9 B7	ORA A	
		<AS INDEX, FUNCTIONS CAN HAVE AN <EXPR2> AS ARGUMENTS, AND	03FA F9F00	JM OH0W	<ELSE IT IS OVERFLOW
		<EXPR2> CAN BE AN <EXPR2> IN PARANTHESES.	03FB 78	MOV A,B	
			03FC B7	ORA A	
032D 212107	EXPR1: LXI H,TAB8-1	<LOOKUP REL OP.	03FF FC800	CM CHG5GN	<CHANGE SIGN IF NEEDED
0330 C32807	JMP EXEC	<GO DO IT	0402 C3A00	JMP XP31	<LOOK FOR MORE TERMS
0332 CD5003	XP11: CALL XP18	<REL OP ">="	0405 210107	EXPR4: LXI H,TAB4-1	<FIND FUNCTION IN TAB4
0336 D8	RC	<NO, RETURN HL=0	0408 C32807	JMP EXEC	<AND GO DO IT
0337 6F	MOV L,A	<YES, RETURN HL=1	040B FF	XP40: RST 7	<NO, NOT A FUNCTION
0338 C9	RET		040C D81404	JC XP41	<NOT A VARIABLE
0339 CD5003	XP12: CALL XP18	<REL OP "<="	040F 7E	MOV A,M	<VARIABLE
033C C8	RZ	<FALSE, RETURN HL=0	0410 23	INX H	
033D 6F	MOV L,A	<TRUE, RETURN HL=1	0411 66	MOV H,M	<VALUE IN HL
033E C9	RET		0412 6F	MOV L,A	
033F CD5003	XP13: CALL XP18	<REL OP "<="	0413 C9	RET	
0342 C8	RZ	<FALSE, RETURN HL=0	0414 CD7700	XP41: CALL TSTNUM	<OR IS IT A NUMBER
0343 D8	RC	<ALSO FALSE, HL=0	0417 78	MOV A,B	<# OF DIGIT
0344 6F	MOV L,A	<TRUE, HL=1	0418 B7	ORA A	
0345 C9	RET		0419 C0	RNZ	<OK
0346 CD5003	XP14: CALL XP18	<REL OP "<="	041A CF	FARN: RST 1	<NO DIGIT, MUST BE
0349 6F	MOV L,A	<SET HL=1	041B 28	DB <<	
034A C8	RZ	<REL, TRUE, RETURN	041C 05	DB XP42-#-1	
034B D8	RC		041D 0F	RST 3	<"<EXPR">
034C 6C	MOV L,H	<ELSE SET HL=0	041E CF	RST 1	
034D C9	RET		041F 29	DB <>	
034E CD5003	XP15: CALL XP18	<REL OP "<="	0420 01	DB XP43-#-1	
0351 C0	RNZ	<FALSE, RETURN HL=0	0421 C9	XP42: RET	
0352 6F	MOV L,A	<ELSE SET HL=1	0422 C3C604	XP43: JMP OHART	<ELSE SAY: "WHAT?"
0353 C9	RET				
0354 CD5003	XP16: CALL XP18	<REL OP "<="	0425 CD1A04	RND: CALL FARN	<*** RND<EXPR> ***
0357 D0	RNC	<FALSE, RETURN HL=0	0428 7C	MOV A,H	<EXPR MUST BE +
0358 6F	MOV L,A	<ELSE SET HL=1	0429 B7	ORA A	
0359 C9	RET		042A F9F00	JM OH0W	
035A E1	XP17: POP H	<NOT REL OP.	042D B5	ORA L	<AND NON-ZERO
035B C9	RET	<RETURN HL=<EXPR2>	042E C9F00	JZ OH0W	
035C 79	XP18: MOV A,C	<SUBROUTINE FOR ALL	0431 05	PUSH D	<SAVE BOTH
035D E1	POP H	<REL OP ">="	0432 E5	PUSH H	
035E C1	POP B		0433 2A1210	LHLD RANFNT	<GET MEMORY AS RANDOM
035F E5	PUSH H	<REVERSE TOP OF STACK	0436 116300	LXI D,LSTROM	<NUMBER
0360 C5	PUSH B		0439 E7	RST 4	
0361 4F	MOV C,A		043A DA4004	JC RA1	<WRAP AROUND IF LAST
0362 CD7103	CALL EXPR2	<GET 2ND <EXPR2>	043D 210000	LXI H,START	
0365 EB	XCHG	<VALUE IN DE NOW	0440 5E	RA1: MOV E,M	
0366 E3	XTHL	<1ST <EXPR2> IN HL	0441 23	INX H	
0367 CD9804	CALL CKHLDE	<COMPARE 1ST WITH 2ND	0442 56	MOV D,M	
036A D1	POP D	<RESTORE TEXT POINTER	0443 221310	SHLD RANFNT	
036B 210000	LXI H,0H	<SET HL=0, A=1	0446 E1	POP H	
036E 3E01	MVI A,1		0447 EE	XCHG	
0370 C9	RET		0448 65	PUSH B	
			0449 CD6604	CALL DIVIDE	<RND<N>=MOD<M,N>+1
0371 CF	EXPR2: SET 1	<NEGATIVE SIGN?	044C C1	POP B	
0372 20	CS		044D D1	POP D	
0373 86	DB XP21-#-1		044E 23	INX H	
0374 210000	LXI H,0H	<YES, MAKE "0"	044F C9	RET	
0377 CD9803	JMP NP26	<TREAT LIKE SUBTRACT			
037A CF	XP21: RST 1	<POSITIVE SIGN? IGNORE	0450 CD1A04	ABS: CALL FARN	<*** ABS<EXPR> ***
037B 3B	DB <>		0453 2E	DCX D	
037C 00	DB XP22-#-1		0454 CD8204	CALL CHKSGN	<CHECK SIGN
037D CD4501	XP22: CALL EXPR3	<1ST <EXPR3>	0457 12	INX D	
0380 CF	XP23: RST 1	<ADD?	0458 C9	RET	
0381 16	DB <>		0459 2A1510	SIZE: LHLD TXTUNF	<*** SIZE ***
0382 15	DB XP25-#-1		045C D5	PUSH D	<GET THE NUMBER OF FREE
0383 E5	PUSH H	<YES, SAVE VALUE	045D EE	XCHG	<BYTES BETWEEN 'TXTUNF'
0384 CD4501	CALL EXPR3	<GET 2ND <EXPR3>	045E 216612	LXI H,VAREGN	<AND 'VAREGN'
0387 EB	XP24: XCHG	<2ND IN DE	0461 CD7C04	CALL SUBDE	
0388 E3	XTHL	<1ST IN HL	0464 D1	POP D	
0389 7C	MOV A,H	<COMPARE SIGN	0465 C9	RET	
038A AA	XRA D				
038B 7A	MVI A,D				
038C 19	DAD D				
038D D1	POP D	<RESTORE TEXT POINTER			
038E F98003	JM NP23	<1ST 2ND SIGN DIFFER			
0391 AC	XRA H	<1ST 2ND SIGN EQUAL			
0392 F28003	JP NP23	<SO IS RESULT			
0395 CD9F00	JMP OH0W	<ELSE WE HAVE OVERFLOW			
0398 CF	XP25: RST 1	<SUBTRACT?			
0399 20	CS				
039A 86	DB XP42-#-1				
039B E5	XP26: PUSH H	<YES, SAVE 1ST <EXPR3>			
039C CD4503	CALL EXPR3	<GET 2ND <EXPR3>			
039F CD8604	CALL CHG5GN	<NEGATE			
03A2 CD8703	JMP NP24	<AND ADD THEM			
03A5 CD0504	EXPR3: CALL EXPR4	<GET 1ST <EXPR4>			
03A8 CF	XP31: RST 1	<MULTIPLY?			
03A9 2A	DB <>				
03AA 20	DB XP34-#-1				
03AB E5	PUSH H	<YES, SAVE 1ST			
03AC CD0504	CALL EXPR4	<AND GET 2ND <EXPR4>			
03AF 0000	MVI B,0H	<CLEAR B FOR SIGN			
03B1 CD8104	CALL CHKSGN	<CHECK SIGN			
03B4 E1	XTHL	<1ST IN HL			
03B5 CD8204	CALL CHKSGN	<CHECK SIGN OF 1ST			
03B8 EB	XCHG				
03B9 E1	XTHL				
03BA 7C	MOV A,H	<IS HL > 255 ?			
03BB E7	ORA A				
03BC CD4503	JZ NP22	<NO			
03BF 7A	MOV A,D	<YES, HOW ABOUT DE			
03C0 B2	ORA D				
03C1 EB	XCHG	<PUT SMALLER IN HL			
03C2 C2A000	JNZ AH0W	<ALSO D, WILL OVERFLOW			
03C5 7C	MOV A,L	<THIS IS DUMB			
03C6 210000	LXI H,0H	<CLEAR RESULT			
03C9 E7	ORA A	<ADD AND COUNT			
03CA C9F793	JZ XP25				
03CD 19	DAD D				
03CE DAA000	JC AH0W	<OVERFLOW			
03D1 3D	DCR A				
03D2 C2C003	JNZ XP23				
03D5 CD7F03	JMP XP25	<FINISHED			
03D8 CF	XP34: RST 1	<DIVIDE?			
03D9 2F	DB <>				
03DA 46	DB XP42-#-1				
03DB E5	PUSH H	<YES, SAVE 1ST <EXPR4>			
03DC CD0504	CALL EXPR4	<AND GET 2ND ONE			
03DF 0600	MVI B,0H	<CLEAR B FOR SIGN			
03E1 CD8204	CALL CHKSGN	<CHECK SIGN OF 2ND			
03E4 E3	XTHL	<GET 1ST IN HL			
03E5 CD8204	CALL CHKSGN	<CHECK SIGN OF 1ST			
03E8 EB	XCHG				
03E9 E3	XTHL				
03EA EB	XCHG				
03EB 7A	MOV A,D	<DIVIDE BY 0?			
03EC B3	ORA E				
03ED CAA000	JZ AH0W	<SAV "HOW?"			
03F0 C5	PUSH B	<ELSE SAVE SIGN			
03F1 CD6604	CALL DIVIDE	<USE SUBROUTINE			
03F4 60	MOV H,B	<RESULT IN HL NOW			
03F5 69	MOV L,C				
03F6 C1	POP B	<GET SIGN BACK			
			0466 E5	DIVIDE: PUSH H	<*** DIVIDE ***
			0467 6C	MOV L,H	<DIVIDE H BY DE
			0468 2E00	MVI H,0	
			0469 CD7104	CALL DV1	
			046D 41	MOV B,C	<SAVE RESULT IN B
			046E 7D	MOV A,L	<<REMAINDER+L>/DE
			046F E1	POP H	
			0470 67	MOV H,A	
			0471 BEFF	MVI C,0FFH	<RESULT IN C
			0473 0C	INR C	<DUMB ROUTINE
			0474 CD7C04	CALL SUBDE	<DIVIDE BY SUBTRACT
			0477 CD7204	JNC DV2	<AND COUNT
			047A 19	DAD D	
			047B C9	RET	
			047C 7D	SUBDE: MOV A,L	<*** SUBDE ***
			047D 93	SUB E	<SUBTRACT DE FROM
			047E 6F	MOV L,A	<HL
			047F 7C	MOV A,H	
			0480 9A	SBB D	
			0481 67	MOV H,A	
			0482 C9	RET	
			0483 7C	CHKSGN: MOV A,H	<*** CHKSGN ***
			0484 E7	ORA A	<CHECK SIGN OF HL
			0485 F0	RP	<IF - , CHANGE SIGN
			0486 7C	CHG5GN: MOV A,H	<*** CHG5GN ***
			0487 F5	PUSH PSN	
			0488 3F	CMA	<CHANGE SIGN OF HL
			0489 67	MOV H,A	
			048A 7D	MOV A,L	
			048B 2F	CMA	
			048C 6F	MOV L,A	
			048D 23	INX H	
			048E F1	POP PSN	
			048F AC	XRA H	
			0490 F29F00	JP OH0W	
			0492 78	MOV A,B	<AND ALSO FLIP B
			0494 EE50	MVI 50H	
			0496 47	MOV B,A	
			0497 C9	RET	

```

0498 7C      CKHLD:  MOV  A,H
0499 AA      XRA  D      ;SAME SIGN?
049A F29E04  JP   CK1     ;YES, COMPARE
049B EB      XCHG      ;NO. XCH AND COMP
049C E7      RST  4
049D C9      RET

;*****
; *** SETVAL *** FIN *** ENDCHK *** & ERROR (& FRIENDS) ***
; "SETVAL" EXPECTS A VARIABLE, FOLLOWED BY AN EQUAL SIGN AND
; THEN AN EXPR. IT EVALUATES THE EXPR. AND SET THE VARIABLE
; TO THAT VALUE.
; "FIN" CHECKS THE END OF A COMMAND. IF IT ENDED WITH ";",
; EXECUTION CONTINUES. IF IT ENDED WITH A CR, IT FINDS THE
; NEXT LINE AND CONTINUE FROM THERE.
; "ENDCHK" CHECKS IF A COMMAND IS ENDED WITH CR. THIS IS
; REQUIRED IN CERTAIN COMMANDS. (GOTO, RETURN, AND STOP ETC.)
; "ERROR" PRINTS THE STRING POINTED BY DE (&ND ENDS WITH CR).
; IT THEN PRINTS THE LINE POINTED BY 'CURRNT' WITH A "?".
; INSERTED AT WHERE THE OLD TEXT POINTER (SHOULD BE ON TOP
; OF THE STACK) POINTS TO. EXECUTION OF TB IS STOPPED
; AND TB1 IS RESTARTED. HOWEVER, IF 'CURRNT' -> ZERO
; (INDICATING A DIRECT COMMAND), THE DIRECT COMMAND IS NOT
; PRINTED. AND IF 'CURRNT' -> NEGATIVE # (INDICATING 'INPUT'
; COMMAND), THE INPUT LINE IS NOT PRINTED AND EXECUTION IS
; NOT TERMINATED BUT CONTINUED AT 'INPERR'.
; RELATED TO 'ERROR' ARE THE FOLLOWING:
; 'OHAT' SAVES TEXT POINTER IN STACK AND GET MESSAGE "WHAT?"
; 'AHAT' JUST GET MESSAGE "WHAT?" AND JUMP TO 'ERROR'.
; 'OSORRY' AND 'ASORRY' DO SAME KIND OF THING.
; 'OHOW' AND 'AHOW' IN THE ZERO PAGE SECTION ALSO DO THIS

04A0 FF      SETVAL:  RST  7      ;*** SETVAL ***
04A1 D9C604  JC   OHAT     ;"WHAT?" NO VARIABLE
04A2 E5      PUSH  H      ;SAVE ADDRESS OF VAR.
04A3 CF      RST  1      ;PASS "=" SIGN
04A4 2D      DB  "="
04A5 08      DB  SV1-#-1
04A6 DF      RST  2      ;EVALUATE EXPR.
04A7 44      MOV  B,H      ;VALUE IN BC NOW
04A8 4D      MOV  C,L
04A9 40      POP  H      ;GET ADDRESS
04AA E1      MOV  M,C      ;SAVE VALUE
04AB 71      INX  H
04AC 23      MOV  M,B
04AD 70      RET
04AE C9      SV1:  JMP  OHAT     ;NO "=" SIGN

04B0 0F      FIN:   RST  1      ;*** FIN ***
04B1 2B      DB  "BH"
04B2 04      DB  F11-#-1
04B3 F1      POP  PSW     ;" ", PURGE RET ADDR.
04B4 C5701  JMP  RUNSML   ;CONTINUE SAME LINE
04B5 CF      F11:  RST  1      ;NOT " ", IS IT CR?
04B6 08      DB  CR
04B7 04      DB  F12-#-1
04B8 F1      POP  PSW     ;YES, PURGE RET ADDR.
04B9 C5701  JMP  RUNNXL   ;RUN NEXT LINE
04BA C9      F12:  RET         ;ELSE RETURN TO CALLER

ENDCHK:
04C2 EF      RST  5      ;*** ENDCHK ***
04C3 FE00   CPI  CR      ;END WITH CR?
04C4 C8      RZ         ;OK, ELSE SAY: "WHAT?"

04C5 05      OHAT:  PUSH  D      ;*** OHAT ***
04C6 11AE00  AHAT:  LXI  D,AHAT ;*** AHAT ***
04C7 97      ERROR:  SUB  A      ;*** ERROR ***
04C8 CD6005  CALL  PRSTG   ;PRINT "WHAT?", "HOW?"
04C9 D1      POP  D      ;OR "SORRY"
04CA 1A      LDRX  D      ;SAVE THE CHARACTER
04CB E5      PUSH  PSW    ;AT WHERE OLD DE ->
04CC 97      SUB  A      ;AND PUT A 0 THERE
04CD 12      STAX  D
04CE 2A0110  LHLD  CURRNT ;GET CURRENT LINE #
04CF E5      PUSH  H
04D0 7E      MOV  A,M     ;CHECK THE VALUE
04D1 23      INX  H
04D2 B6      ORA  H
04D3 C4      POP  D
04D4 CBA900  JZ   RSTART  ;IF ZERO, JUST RESTART
04D5 7E      MOV  A,M     ;IF NEGATIVE,
04D6 B7      ORA  A
04D7 FAC202  JM  INPERR   ;REDO INPUT
04D8 C0C205  CALL  PRTLN  ;ELSE PRINT THE LINE
04D9 1B      DCX  D      ;UP TO WHERE THE 0 IS
04DA F1      POP  PSW    ;RESTORE THE CHARACTER
04DB 12      STAX  D
04DC 2E2F   MVI  A,2FH   ;PRINT A "?"
04DD C7      RST  2
04DE 97      SUB  A      ;AND THE REST OF THE
04DF C06005  CALL  PRSTG
04E0 C3BA00  JMP  RSTART

OSORRY:
04E2 D5      PUSH  D      ;*** OSORRY ***

ASORRY:
04E4 11B400  LXI  D,SORRY ;*** ASORRY ***
04E5 C3CA04  JMP  ERROR

;*****
; *** GETLN *** FNDLN (& FRIENDS) ***
; "GETLN" READS A INPUT LINE INTO 'BUFFER'. IT FIRST PROMPT
; THE CHARACTER IN A (GIVEN BY THE CALLER), THEN IT FILLS THE
; THE BUFFER AND ECHOS. IT IGNORES LF'S AND NULLS, BUT STILL
; ECHOS THEM BACK. RUB-OUT IS USED TO CAUSE IT TO DELETE
; THE LAST CHARACTER (IF THERE IS ONE), AND ALT-MOD IS USED TO
; CAUSE IT TO DELETE THE WHOLE LINE AND START IT ALL OVER.
; CR SIGNALS THE END OF A LINE, AND CAUSE 'GETLN' TO RETURN.
; "FNDLN" FINDS A LINE WITH A GIVEN LINE # (<IN HL) IN THE
; TEXT SAVE AREA. DE IS USED AS THE TEXT POINTER. IF THE
; LINE IS FOUND, DE WILL POINT TO THE BEGINNING OF THAT LINE
; (I.E., THE LOW BYTE OF THE LINE #), AND FLAGS ARE NC & Z.
; IF THAT LINE IS NOT THERE AND A LINE WITH A HIGHER LINE #
; IS FOUND, DE POINTS TO THERE AND FLAGS ARE NC & NZ. IF
; WE REACHED THE END OF TEXT SAVE ARE AND CANNOT FIND THE
; LINE, FLAGS ARE C & NZ.
; "FNDLN" WILL INITIALIZE DE TO THE BEGINNING OF THE TEXT SAVE
; AREA TO START THE SEARCH. SOME OTHER ENTRIES OF THIS
; ROUTINE WILL NOT INITIALIZE DE AND DO THE SEARCH.
; "FNDLNP" WILL START WITH DE AND SEARCH FOR THE LINE #.
; "FNDXMT" WILL BUMP DE BY 2, FIND A CR AND THEN START SEARCH.
; "FNDSKP" USE DE TO FIND A CR, AND THEN START SEARCH.

04FA D7      GETLN:  RST  2      ;*** GETLN ***
04FB 119D12  LXI  D,BUFFER ;PROMPT AND INIT.
04FC C8406  GL1:  CALL  CHKTO ;CHECK KEYBOARD
04FD CAFE04  JZ   GL1     ;NO INPUT, WAIT
04FE F77F   CPI  7FH     ;DELETE LAST CHARACTER?
04FF 07      JZ   GL2     ;YES
0500 07      RST  2      ;INPUT, ECHO BACK
0501 FE0A   CPI  0AH     ;IGNORE LF
0502 CAFE04  JZ   GL1     ;IGNORE NULL
0503 B7      ORA  A
0504 CAFE04  JZ   GL1     ;DELETE THE WHOLE LINE?
0505 FE7D   CPI  70H     ;YES
0506 C03005  JZ   GL4     ;ELSE, SAVE INPUT
0507 12      STAX  D      ;AND BUMP POINTER
0508 13      INX  D
0509 FE00   CPI  00H     ;WAS IT CR?
050A C8      RZ         ;YES, END OF LINE
050B 78      MOV  A,E     ;ELSE, MORE FREE ROOM?
050C FE00   CPI  BUFEND  ;AND OFFH
050D C2FE04  JNZ  GL1     ;YES, GET NEXT INPUT
050E 78      MOV  A,E     ;DELETE LAST CHARACTER
050F FE90   CPI  BUFFER  ;AND OFFH
0510 C23005  JZ   GL4     ;NO, REDO WHOLE LINE
0511 18      DCX  D      ;YES, BACKUP POINTER
0512 3E5C   MVI  A,5CH   ;AND ECHO A BACK-SLASH
0513 07      RST  2
0514 C3FE04  JMP  GL1     ;GO GET NEXT INPUT
0515 C06E00  GL4:  CALL  CLRF   ;REDO ENTIRE LINE
0516 7E5E   MVI  A,05EH  ;CR, LF AND UP-ARROW
0517 C3FA04  JMP  GETLN

0520 7C      FNDLN:  MOV  A,H      ;*** FNDLN ***
0521 B7      ORA  A      ;CHECK SIGN OF HL
0522 0000   JM  0000     ;IT CANNOT BE -
0523 11710  LXI  D,TEXTBN ;INIT. TEXT POINTER

0540 E5      FNDLP:  PUSH  H      ;*** FNDLP ***
0541 2A1510  FL1:  LHLD  TXTUNF ;CHECK IF WE PASSED END
0542 2B      DCX  H
0543 E7      RST  4
0544 E1      POP  H      ;GET LINE # BACK
0545 08      RC         ;C.NZ PASSED END
0546 1A      LDRX  D      ;WE DID NOT; GET BYTE 1
0547 95      SUB  L      ;IS THIS THE LINE?
0548 47      MOV  B,A     ;COMPARE LOW ORDER
0549 13      INX  D
054A 1A      LDRX  D      ;GET BYTE 2
054B 9C      SBB  H      ;COMPARE HIGH ORDER
054C 1A      JC   FL2     ;NO, NOT THERE YET
054D 9C      DCX  D      ;ELSE WE EITHER FOUND
054E D95505  ORA  B      ;IT, OR IT IS NOT THERE
054F 18      RET         ;NO. 2: FOUND; NO. 2: NO

0554 12      FNDXMT:  INX  D      ;*** FNDXMT ***
0555 12      FL2:  INX  D      ;FIND NEXT LINE
; JUST PASSED BYTE 1 & 2

0556 1A      FNDSKP:  LDRX  D      ;*** FNDSKP ***
0557 FE00   CPI  CR      ;TRY TO FIND CR
0558 C25505  JNZ  FL2     ;KEEP LOOKING
0559 13      INX  D      ;FOUND CR, SKIP OVER
055A C34005  JMP  FL1     ;CHECK IF END OF TEXT

;*****
; *** PRSTG *** OTSTG *** PRNUM *** & PRTLN ***
; "PRSTG" PRINTS A STRING POINTED BY DE. IT STOPS PRINTING
; AND RETURNS TO CALLER WHEN EITHER A CR IS PRINTED OR WHEN
; THE NEXT BYTE IS THE SAME AS WHAT WAS IN A (GIVEN BY THE
; CALLER). OLD A IS STORED IN B. OLD B IS LOST.
; "OTSTG" LOOKS FOR A BACK-ARROW, SINGLE QUOTE, OR DOUBLE
; QUOTE. IF NONE OF THESE, RETURN TO CALLER. IF BACK-ARROW,
; OUTPUT A CR WITHOUT A LF. IF SINGLE OR DOUBLE QUOTE, PRINT
; THE STRING IN THE QUOTE AND DEMANDS A MATCHING UNQUOTE.
; AFTER THE PRINTING THE NEXT 2 BYTES OF THE CALLER IS SKIPPED
; OVER (USUALLY A JUMP INSTRUCTION).
; "PRNUM" PRINTS THE NUMBER IN HL. LEADING BLANKS ARE ADDED
; IF NEEDED TO PAD THE NUMBER OF SPACES TO THE NUMBER IN C.
; HOWEVER, IF THE NUMBER OF DIGITS IS LARGER THAN THE # IN
; C, ALL DIGITS ARE PRINTED ANYWAY. NEGATIVE SIGN IS ALSO
; PRINTED AND COUNTED IN. POSITIVE SIGN IS NOT.
; "PRTLN" PRINTS A SAVED TEXT LINE WITH LINE # AND ALL.

0560 47      PRSTG:  MOV  B,A     ;*** PRSTG ***
0561 1A      PS1:  LDRX  D      ;GET A CHARACTER
0562 13      INX  D      ;BUMP POINTER
0563 B8      CMP  B      ;SAME AS OLD A?
0564 C8      RZ         ;YES, RETURN
0565 07      RST  2      ;ELSE PRINT IT
0566 FE00   CPI  CR      ;WAS IT A CR?
0567 C26105  JNZ  PS1     ;NO, NEXT
0568 C9      RET         ;YES, RETURN

056C 0F      OTSTG:  RST  1      ;*** OTSTG ***
056D 22      DB  "/"
056E 0F      DB  Q13-#-1
056F 2E22   MVI  A,22H   ;IT IS A "
0570 C06005  OT1:  CALL  PRSTG ;PRINT UNTIL ANOTHER
0571 FE00   CPI  CR      ;WAS LAST ONE A CR?
0572 E1      POP  H      ;RETURN ADDRESS
0573 C04701  JZ   RUNNXL  ;WAS CR, RUN NEXT LINE
0574 33      INX  H      ;SKIP 3 BYTES ON RETURN
0575 23      INX  H
0576 23      INX  H
0577 E9      PCHL  H      ;RETURN
0578 0F      OT2:  RST  1      ;IS IT A ?
0579 27      DB  27H
057A 05      DB  04-#-1
057B 27      MVI  A,27H  ;YES, DO SAME
057C 0F      JMP  OT1     ;AS IN "
057D 0F      OT3:  RST  1      ;IS IT BACK-ARROW?
057E 5F      DB  5FH
057F 05      DB  05-#-1
0580 27      MVI  A,05DH ;YES, CR WITHOUT LF
0581 0F      RST  2      ;DO IT TWICE TO GIVE
0582 07      RST  2      ;TTY ENOUGH TIME
0583 E1      POP  H      ;RETURN ADDRESS
0584 C27A05  JMP  OT2
0585 C9      OT5:  RET         ;NONE OF ABOVE

0592 0600   PRNUM:  MVI  B,0     ;*** PRNUM ***
0593 C08204  CALL  CHKSGN ;CHECK SIGN
0594 F29E05  JP   PN1     ;NO SIGN
0595 062D   MVI  B,"/"  ;B=SIGN
0596 00      DCX  C      ;"/" TAKES SPACE
0597 05      PUSH  D
0598 110A00  LXI  D,0AH   ;DECIMAL

```



```

05A1 05      PUSH D      ;SAVE AS A FLAG
05A2 06      DCR C      ;C=SPACES
05A3 05      PUSH B      ;SAVE SIGN & SPACE
05A4 0D6604  FN2:  CALL DIVIDE ;DIVIDE HL BY 10
05A7 78      MOV A,B      ;RESULT 0?
05A8 81      ORA C      ;YES, WE GOT ALL
05A9 0AB405  JZ  FN3     ;NO, SAVE REMAINDER
05AC 83      XTHL      ;AND COUNT SPACE
05AD 2D      DCR L      ;HL IS OLD BC
05AE 85      PUSH H,B    ;MOVE RESULT TO BC
05AF 89      MOV L,C      ;AND DIVIDE BY 10
05B0 89      JMP  FN2     ;HE GOT ALL DIGITS IN
05B4 01      FN3:  POP B      ;THE STACK
05B5 00      DCR C      ;LOOK AT SPACE COUNT
05B6 79      MOV A,C      ;NO LEADING BLANKS
05B7 87      ORA A      ;LEADING BLANKS
05B8 0AC105  JM  FN5     ;MORE?
05BB 2E20     MVI A,20H   ;PRINT SIGN
05BD 07      RST 2      ;IF SO, RETURN
05BE 02B505  FN5:  MOV A,B  ;ELSE CONVERT TO ASCII
05C1 78      ORA A      ;AND PRINT THE DIGIT
05C2 87      ORA A      ;GO BACK FOR MORE
05C3 041000  CNZ 10H    ;LAST REMAINDER IN E
05C6 5D      MOV E,L    ;CHECK DIGIT IN E
05C7 7B      MOV A,E    ;10 IS FLAG FOR NO MORE
05C8 0E0A    CPI 0AH    ;IF SO, RETURN
05CA 01      POP D      ;ELSE CONVERT TO ASCII
05CB 08      RZ        ;AND PRINT THE DIGIT
05CC 0E20    RDI 20H   ;AND PRINT THE DIGIT
05CE 07      RST 2     ;GO BACK FOR MORE
05CF 02C705  JMP  FN6   ;GO BACK FOR MORE

05D2 1A      PRTLN: LDAX D   ;*** PRTLN ***
05D3 6F      MOV L,A    ;LOW ORDER LINE #
05D4 13      INX D     ;HIGH ORDER
05D5 1A      LDAX D    ;HIGH ORDER
05D6 67      MOV H,A   ;HIGH ORDER
05D7 13      INX D     ;HIGH ORDER
05D8 0E04    MVI C,4H  ;PRINT 4 DIGIT LINE #
05DA 0D9205  CALL PRTNM ;PRINT 4 DIGIT LINE #
05DD 2E20     MVI A,20H ;FOLLOWED BY A BLANK
05DF 07      RST 2     ;FOLLOWED BY A BLANK
05E0 97      SUB A     ;FOLLOWED BY A BLANK
05E1 0D6005  CALL PRTSTG ;FOLLOWED BY A BLANK
05E4 09      RET

;*****
; *** MVUP *** MVDOWN *** POPA *** & PUSHA ***
; 'MVUP' MOVES A BLOCK UP FROM WHERE DE-> TO WHERE BC-> UNTIL
; DE = HL
; 'MVDOWN' MOVES A BLOCK DOWN FROM WHERE DE-> TO WHERE HL->
; UNTIL DE = BC
; 'POPA' RESTORES THE 'FOR' LOOP VARIABLE SAVE AREA FROM THE
; STACK
; 'PUSHA' STACKS THE 'FOR' LOOP VARIABLE SAVE AREA INTO THE
; STACK

05E5 07      MVUP:  RST 4      ;*** MVUP ***
05E6 08      RZ        ;DE = HL, RETURN
05E7 1A      LDAX D    ;GET ONE BYTE
05E8 02      STAX B    ;MOVE IT
05E9 13      INX D    ;INCREASE BOTH POINTERS
05EA 02      INX B    ;INCREASE BOTH POINTERS
05EB 02E505  JMP  MVUP ;UNTIL DONE

;*****
; *** TABLES *** DIRECT *** & EXEC ***
; THIS SECTION OF THE CODE TESTS A STRING AGAINST A TABLE.
; WHEN A MATCH IS FOUND, CONTROL IS TRANSFERRED TO THE SECTION
; OF CODE ACCORDING TO THE TABLE.
; AT 'EXEC', DE SHOULD POINT TO THE STRING AND HL SHOULD POI
; TO THE TABLE-1. AT 'DIRECT', DE SHOULD POINT TO THE STRI
; HL WILL BE SET UP TO POINT TO TABLE-1, WHICH IS THE TABLE (
; ALL DIRECT AND STATEMENT COMMANDS.
; A 'X' IN THE STRING WILL TERMINATE THE TEST AND THE PARTI
; MATCH WILL BE CONSIDERED AS A MATCH. E.G., 'PR', 'FR',
; 'PRI', 'PRIN', OR 'PRINT' WILL ALL MATCH 'PRINT'.
; THE TABLE CONSISTS OF ANY NUMBER OF ITEMS. EACH ITEM
; IS A STRING OF CHARACTERS WITH BIT 7 SET TO 0 AND
; A JUMP ADDRESS STORED HI-LOW WITH BIT 7 OF THE HIGH
; BYTE SET TO 1.
; END OF TABLE IS AN ITEM WITH A JUMP ADDRESS ONLY. IF THE
; STRING DOES NOT MATCH ANY OF THE OTHER ITEMS, IT WILL
; MATCH THIS NULL ITEM AS DEFAULT.

05EE 78      MVDOWN: MOV A,B    ;*** MVDOWN ***
05EF 92      SUB D    ;TEST IF DE = BC
05F0 02F505  JNZ  MD1   ;NO, GO MOVE
05F1 79      MOV A,C  ;MAYBE, OTHER BYTE?
05F4 92      SUB E    ;YES, RETURN
05F5 08      RZ        ;YES, RETURN
05F6 1B      MD1:  DCR D    ;ELSE MOVE A BYTE
05F7 2B      DCR H    ;BUT FIRST DECREASE
05F8 1A      LDAX D  ;BOTH POINTERS AND
05F9 77      MOV H,A  ;THEN DO IT
05FA 02EE05  JMP  MVDOWN ;LOOP BACK

05FD 01      POPA:  POP B    ;BC = RETURN ADDR.
05FE 01      POP H    ;RESTORE LOPVAR, BUT
05FF 020910  SHLD LOPVAR ;#0 MEANS NO MORE
0600 7C      MOV A,H  ;#0 MEANS NO MORE
0601 85      ORA L    ;#0 MEANS NO MORE
0602 85      ORA L    ;#0 MEANS NO MORE
0603 0A1705  JZ  PP1    ;YEP, GO RETURN
0604 01      POP H    ;NOP, RESTORE OTHERS
0605 08      SHLD LOPINC ;RESTORE OTHERS
0606 01      POP H    ;RESTORE OTHERS
0607 01      SHLD LOPLMT ;RESTORE OTHERS
0608 01      POP H    ;RESTORE OTHERS
0609 01      SHLD LOPFLN ;RESTORE OTHERS
0610 01      POP H    ;RESTORE OTHERS
0611 01      SHLD LOPPT  ;RESTORE OTHERS
0612 05      PP1:  PUSH B   ;BC = RETURN ADDR.
0613 09      RET

0619 210E12  PUSHA: LXI H,STKLT ;*** PUSHA ***
061C 0D8604  CALL CHSSGN

061F 01      POP B    ;BC=RETURN ADDRESS
0620 09      DAD SP   ;IS STACK NEAR THE TOP?
0621 02F204  JNC  OSORRY ;YES, SORRY FOR THAT.
0622 0A0910  LHLD LOPVAR ;ELSE SAVE LOOP VAR.S
0623 7C      MOV A,H  ;BUT IF LOPVAR IS 0
0624 85      ORA L    ;THAT WILL BE ALL
0625 08      RZ        ;THAT WILL BE ALL
0626 0A2F0E  JZ  PU1    ;YEP, GO RETURN
0627 01      LHLD LOPPT ;NOP, RESTORE OTHERS
0628 01      PUSH H   ;RESTORE OTHERS
0629 01      LHLD LOPLN ;RESTORE OTHERS
0630 01      PUSH H   ;RESTORE OTHERS
0631 01      LHLD LOPLMT ;RESTORE OTHERS
0632 01      PUSH H   ;RESTORE OTHERS
0633 01      LHLD LOPINC ;RESTORE OTHERS
0634 01      PUSH H   ;RESTORE OTHERS
0635 01      LHLD LOPVAR ;RESTORE OTHERS
0636 01      PUSH H   ;RESTORE OTHERS
0637 05      PU1:  PUSH B   ;BC = RETURN ADDR.
0638 09      RET

;*****
; *** OUTC *** & CHKIO ***
; THESE ARE THE ONLY I/O ROUTINES IN TBI.
; 'OUTC' IS CONTROLLED BY A SOFTWARE SWITCH 'OCSW'. IF OCSW=0
; 'OUTC' WILL JUST RETURN TO THE CALLER. IF OCSW IS NOT 0,
; IT WILL OUTPUT THE BYTE IN A. IF THAT IS A CR, A LF IS ALSO
; SEND OUT. ONLY THE FLAGS MAY BE CHANGED AT RETURN. ALL REG.
; ARE RESTORED.

0640 05      OUTC:  RST 4      ;*** OUTC ***
0641 09      RET

0642 05      CHKIO: RST 4      ;*** CHKIO ***
0643 09      RET

0644 05      INIT:  STA  OCSW  ;INITIALIZE OCSW
0645 3E0F    MVI A,0CFH ;INITIALIZE OCSW
0646 03FB    OUT 0FBH  ;INITIALIZE OCSW
0647 3E27    MVI A,27H  ;INITIALIZE OCSW
0648 03FB    OUT 0FBH  ;INITIALIZE OCSW
0649 1619    MVI D,19H  ;INITIALIZE OCSW
0650 00E000  CALL CRLF  ;INITIALIZE OCSW
0651 15      DCR D     ;INITIALIZE OCSW
0652 024F0E  JNZ  PATLOP ;INITIALIZE OCSW
0653 97      SUB A     ;INITIALIZE OCSW
0654 11A20E  LXI D,MSG1 ;INITIALIZE OCSW
0655 0D6005  CALL PRTSTG ;INITIALIZE OCSW
0656 11A20E  LXI H,STRT ;INITIALIZE OCSW
0657 021210  SHLD RANPNT ;INITIALIZE OCSW
0658 117110  LXI H,TXTIBGN ;INITIALIZE OCSW
0659 021510  SHLD TXTUNF  ;INITIALIZE OCSW
0660 02BA00  JMP  RSTART  ;INITIALIZE OCSW
0661 02710E  JNZ  OC2    ;IT IS ON
0662 02710E  JNZ  OC2    ;IT IS OFF
0663 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0664 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0665 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0666 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0667 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0668 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0669 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0670 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0671 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0672 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0673 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0674 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0675 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0676 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0677 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0678 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0679 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0680 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0681 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0682 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0683 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0684 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0685 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0686 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0687 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0688 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0689 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0690 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0691 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0692 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0693 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0694 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0695 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0696 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0697 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0698 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0699 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0700 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0701 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0702 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0703 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0704 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0705 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0706 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0707 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0708 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0709 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0710 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0711 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0712 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0713 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0714 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0715 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0716 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0717 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0718 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0719 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0720 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0721 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0722 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0723 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0724 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0725 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0726 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0727 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0728 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0729 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0730 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0731 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0732 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0733 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0734 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0735 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0736 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0737 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0738 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0739 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0740 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0741 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0742 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0743 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0744 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0745 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0746 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0747 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0748 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0749 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0750 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0751 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0752 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0753 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0754 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0755 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0756 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0757 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0758 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0759 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0760 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0761 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0762 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0763 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0764 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0765 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0766 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0767 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0768 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0769 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0770 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0771 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0772 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0773 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0774 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0775 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0776 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0777 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0778 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0779 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0780 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0781 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0782 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0783 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0784 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0785 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0786 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0787 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0788 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0789 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0790 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0791 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0792 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0793 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0794 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0795 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0796 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0797 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0798 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0799 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0800 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0801 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0802 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0803 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0804 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0805 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0806 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0807 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0808 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0809 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0810 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0811 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0812 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0813 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0814 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0815 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0816 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0817 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0818 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0819 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0820 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0821 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0822 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0823 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0824 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0825 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0826 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0827 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0828 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0829 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0830 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0831 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0832 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0833 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0834 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0835 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0836 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0837 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0838 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0839 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0840 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0841 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0842 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0843 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0844 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0845 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0846 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0847 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0848 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0849 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0850 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0851 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0852 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0853 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0854 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0855 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0856 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0857 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0858 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0859 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0860 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0861 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0862 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0863 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0864 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0865 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0866 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0867 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0868 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0869 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0870 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0871 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0872 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0873 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0874 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0875 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0876 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0877 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0878 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0879 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0880 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0881 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0882 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0883 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0884 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0885 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0886 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0887 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0888 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0889 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0890 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0891 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0892 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0893 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0894 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0895 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0896 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0897 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0898 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0899 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0900 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0901 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0902 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0903 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0904 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0905 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0906 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0907 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0908 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0909 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0910 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0911 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0912 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0913 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0914 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0915 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0916 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0917 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0918 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0919 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0920 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0921 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0922 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0923 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0924 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0925 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0926 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0927 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0928 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0929 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0930 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0931 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0932 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0933 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0934 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0935 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0936 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0937 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0938 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0939 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0940 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0941 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0942 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0943 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0944 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0945 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0946 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0947 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0948 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0949 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0950 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0951 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0952 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0953 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0954 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0955 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0956 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0957 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0958 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0959 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0960 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0961 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0962 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0963 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0964 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0965 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0966 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0967 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0968 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0969 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0970 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0971 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0972 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0973 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0974 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0975 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0976 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0977 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0978 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0979 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0980 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0981 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0982 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0983 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0984 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0985 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0986 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0987 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0988 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0989 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0990 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0991 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0992 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0993 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0994 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0995 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0996 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0997 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0998 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
0999 02710E  JNZ  OC2    ;RESTORE AF AND RETURN
1000 02710E  JNZ  OC2    ;RESTORE AF AND RETURN

```

**SOFTWARE SECTION**

**MICROCOMPUTER DEVELOPMENT SOFTWARE**

```

06DA 5245455 DB 'RETURN'
06DE 524E + DWA RETURN
06E0 81 + DB (001DFH SHR 8) +128
06E1 DF + DB 001DFH AND 0FFH
06E2 52454D DB 'REM'
06E5 82 + DWA REM
06E6 B0 + DB (002B0H SHR 8) +128
06E7 464F52 DB 'FOR'
06EA 81 + DWA FOR
06EB F8 + DB (001F8H SHR 8) +128
06EB F8 + DB 001F8H AND 0FFH
06EC 494E5055 DB 'INPUT'
06F0 54 + DWA INPUT
06F1 82 + DB (002CDH SHR 8) +128
06F2 CD + DB 002CDH AND 0FFH
06F3 5052494E DB 'PRINT'
06F7 54 + DWA PRINT
06F8 81 + DB (00187H SHR 8) +128
06F9 87 + DB 00187H AND 0FFH
06FA 53544F50 DB 'STOP'
06FE 81 + DWA STOP
06FF 38 + DB (0013BH SHR 8) +128
06FF 38 + DB 0013BH AND 0FFH
0700 83 + DWA DEFLT
0701 1D + DB (0031DH SHR 8) +128
0701 1D + DB 0031DH AND 0FFH
TAB4: ;FUNCTIONS
0702 524E44 DB 'RND'
0705 84 + DWA RND
0706 25 + DB (00425H SHR 8) +128
0706 25 + DB 00425H AND 0FFH
0707 414253 DB 'ABS'
070A 84 + DWA ABS
070B 50 + DB (00450H SHR 8) +128
070B 50 + DB 00450H AND 0FFH
070C 53495A45 DB 'SIZE'
0710 84 + DWA SIZE
0711 59 + DB (00459H SHR 8) +128
0711 59 + DB 00459H AND 0FFH
0712 84 + DWA XP40
0713 0B + DB (0040BH SHR 8) +128
0713 0B + DB 0040BH AND 0FFH
0714 544F TAB5: ;"TO" IN "FOR"
0716 82 + DWA FR1
0717 05 + DB (00208H SHR 8) +128
0717 05 + DB 00208H AND 0FFH
0718 84 + DWA QWHAT
0719 C6 + DB (004C6H SHR 8) +128
0719 C6 + DB 004C6H AND 0FFH
TAB6: ;"STEP" IN "FOR"
071A 53544550 DB 'STEP'
071E 82 + DWA FR2
071F 12 + DB (00212H SHR 8) +128
071F 12 + DB 00212H AND 0FFH
0720 82 + DWA FR3
0721 16 + DB (00216H SHR 8) +128
0721 16 + DB 00216H AND 0FFH
TAB8: ;RELATION OPERATORS
0722 3E3D DB '>='
0724 83 + DWA XP11
0725 33 + DB (00333H SHR 8) +128
0725 33 + DB 00333H AND 0FFH
0726 23 DB '#='
0727 82 + DWA XP12
0728 39 + DB (00339H SHR 8) +128
0728 39 + DB 00339H AND 0FFH
0729 3E DB '<='
072A 82 + DWA XP13
072B 3F + DB (0033FH SHR 8) +128
072B 3F + DB 0033FH AND 0FFH
072C 3D DB '/='
072D 83 + DWA XP15
072E 4E + DB (0034EH SHR 8) +128
072E 4E + DB 0034EH AND 0FFH
072F 3C3D DB '<<='
0731 83 + DWA XP14
0732 16 + DB (00346H SHR 8) +128
0732 16 + DB 00346H AND 0FFH
0733 3C DB '<'
0734 83 + DWA XP16
0735 54 + DB (00354H SHR 8) +128
0735 54 + DB 00354H AND 0FFH
0736 83 + DWA XP17
0737 5A + DB (0035AH SHR 8) +128
0737 5A + DB 0035AH AND 0FFH
;
;
DIRECT:
0738 21A006 LXI H,TAB1-1 ;*** DIRECT ***
;
EXEC:
0738 EF RST 5 ;*** EXEC ***
073C 05 PUSH D ;IGNORE LEADING BLANKS
073D 1A LDM D ;SAVE POINTER
073E 12 INX D ;IF FOUND ' ' IN STRING
073F FE2E CPI 2EH ;BEFORE ANY MISMATCH
0741 CA5A07 JZ EX3 ;WE DECLARE A MATCH
0741 23 INX H ;HL->TABLE
0745 BE CMP M ;IF MATCH, TEST NEXT
0746 CA2D07 JZ EX1
0749 3E7F MVI A,07FH ;ELSE, SEE IF BIT 7
074B 1B DCX D ;OF TABLE IS SET, WHICH
074C BE CMP M ;IS THE JUMP ADDR. (HI)
074D DA6107 JC EX5 ;C: YES, MATCHED
0750 23 EX2: INX H ;C: NO, FIND JUMP ADDR.
0751 BE CMP M
0752 D25007 JNC EX2

```

*Merry Christmas*

*Happy New Year*