```
;Modified Nov 1 2016 by Donn Stewart for use in CPUville Z80 computer
;Changed UART (ACIA) port numbers to 3 for status, 2 for data in INIT, CHKIO, OUTC
;Status bit for read in CHKIO changed to 0x02
;Status bit for write in OUTC (actually OC3) changed to 0x01
;Changed UART initialization parameters in INIT
;Changed ORG statements at end of file to match system with 2K RAM
;*********************************************************
;
;                 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
START:  LXI  SP,STACK                  ;*** COLD START ***
        MVI  A,0FFH
        JMP  INIT
;
        XTHL                           ;*** TSTC OR RST 1 ***
        RST  5                         ;IGNORE BLANKS AND
        CMP  M                         ;TEST CHARACTER
        JMP  TC1                       ;REST OF THIS IS AT TC1
;
CRLF:   MVI  A,CR                      ;*** CRLF ***
;
        PUSH PSW                       ;*** OUTC OR RST 2 ***
        LDA  OCSW                      ;PRINT CHARACTER ONLY
        ORA  A                         ;IF OCSW SWITCH IS ON
        JMP  OC2                       ;REST OF THIS IS AT OC2
;
        CALL EXPR2                     ;*** EXPR OR RST 3 ***
        PUSH H                         ;EVALUATE AN EXPRESSION
        JMP  EXPR1                     ;REST OF IT AT EXPR1
        DB   'W'
;
        MOV  A,H                       ;*** COMP OR RST 4 ***
        CMP  D                         ;COMPARE HL WITH DE
        RNZ                            ;RETURN CORRECT C AND
        MOV  A,L                       ;Z FLAGS
        CMP  E                         ;BUT OLD A IS LOST
        RET
```

```
        DB   'AN'
;
SS1:    LDAX D                          ;*** IGNBLK/RST 5 ***
        CPI  20H                        ;IGNORE BLANKS
        RNZ                             ;IN TEXT (WHERE DE->)
        INX  D                          ;AND RETURN THE FIRST
        JMP  SS1                        ;NON-BLANK CHAR. IN A
;
        POP  PSW                        ;*** FINISH/RST 6 ***
        CALL FIN                        ;CHECK END OF COMMAND
        JMP  QWHAT                      ;PRINT "WHAT?" IF WRONG
        DB   'G'
;
        RST  5                          ;*** TSTV OR RST 7 ***
        SUI  40H                        ;TEST VARIABLES
        RC                              ;C:NOT A VARIABLE
        JNZ  TV1                        ;NOT "@" ARRAY
        INX  D                          ;IT IS THE "@" ARRAY
        CALL PARN                       ;@ SHOULD BE FOLLOWED
        DAD  H                          ;BY (EXPR) AS ITS INDEX
        JC   QHOW                       ;IS INDEX TOO BIG?
        PUSH D                          ;WILL IT OVERWRITE
        XCHG                            ;TEXT?
        CALL SIZE                       ;FIND SIZE OF FREE
        RST  4                          ;AND CHECK THAT
        JC   ASORRY                     ;IF SO, SAY "SORRY"
        LXI  H,VARBGN                   ;IF NOT GET ADDRESS
        CALL SUBDE                      ;OF @(EXPR) AND PUT IT
        POP  D                          ;IN HL
        RET                             ;C FLAG IS CLEARED
TV1:    CPI  1BH                        ;NOT @, IS IT A TO Z?
        CMC                             ;IF NOT RETURN C FLAG
        RC
        INX  D                          ;IF A THROUGH Z
        LXI  H,VARBGN                   ;COMPUTE ADDRESS OF
        RLC                             ;THAT VARIABLE
        ADD  L                          ;AND RETURN IT IN HL
        MOV  L,A                        ;WITH C FLAG CLEARED
        MVI  A,0
        ADC  H
        MOV  H,A
        RET
;
;TSTC:  XTHL                            ;*** TSTC OR RST 1 ***
;       RST  5                          ;THIS IS AT LOC. 8
;       CMP  M                          ;AND THEN JUMP HERE
TC1:    INX  H                          ;COMPARE THE BYTE THAT
        JZ   TC2                        ;FOLLOWS THE RST INST.
        PUSH 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
        DCX  D                          ;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
TN1:    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 B TO # OF DIGITS
        ANA  H                          ;IF H>255, THERE IS NO
        JNZ  QHOW                       ;ROOM FOR NEXT DIGIT
        INR  B                          ;B COUNTS # OF DIGITS
        PUSH 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
        ANI  0FH                        ;CODE
        ADD  L
        MOV  L,A
        MVI  A,0
        ADC  H
        MOV  H,A
        POP  B
        LDAX D                          ;DO THIS DIGIT AFTER
        JP   TN1                        ;DIGIT. S SAYS OVERFLOW
QHOW:   PUSH D                          ;*** ERROR "HOW?" ***
AHOW:   LXI  D,HOW
        JMP  ERROR
HOW:    DB   'HOW?'
        DB   CR
OK:     DB   'OK'
        DB   CR
WHAT:   DB   'WHAT?'
        DB   CR
SORRY:  DB   'SORRY'
        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 TRANSFERED TO "DIRECT".
;
; TINY BASIC PROGRAM SAVE AREA STARTS AT THE MEMORY LOCATION
; LABELED "TXTBGN" AND ENDS AT "TXTEND".  WE ALWAYS FILL THIS
; AREA STARTING AT "TXTBGN", THE UNFILLED PORTION IS POINTED
; BY THE CONTENT OF A MEMORY LOCATION LABELED "TXTUNF".
;
; THE MEMORY LOCATION "CURRNT" POINTS TO THE LINE NUMBER
; THAT IS CURRENTLY BEING INTERPRETED.  WHILE WE ARE IN
; THIS LOOP OR WHILE WE ARE INTERPRETING A DIRECT COMMAND
```

```
; (SEE NEXT SECTION). "CURRNT" SHOULD POINT TO A 0.
;
RSTART: LXI  SP,STACK
ST1:    CALL CRLF                       ;AND JUMP TO HERE
        LXI  D,OK                       ;DE->STRING
        SUB  A                          ;A=0
        CALL PRTSTG                     ;PRINT STRING UNTIL CR
        LXI  H,ST2+1                    ;LITERAL 0
        SHLD CURRNT                     ;CURRENT->LINE # = 0
ST2:    LXI  H,0
        SHLD LOPVAR
        SHLD STKGOS
ST3:    MVI  A,3EH                      ;PROMPT '>' AND
        CALL GETLN                      ;READ A LINE
        PUSH D                          ;DE->END OF LINE
        LXI  D,BUFFER                   ;DE->BEGINNING OF LINE
        CALL TSTNUM                     ;TEST IF IT IS A NUMBER
        RST  5
        MOV  A,H                        ;HL=VALUE OF THE # OR
        ORA  L                          ;0 IF NO # WAS FOUND
        POP  B                          ;BC->END OF LINE
        JZ   DIRECT
        DCX  D                          ;BACKUP DE AND SAVE
        MOV  A,H                        ;VALUE OF LINE # THERE
        STAX D
        DCX  D
        MOV  A,L
        STAX D
        PUSH B                          ;BC,DE->BEGIN, END
        PUSH D
        MOV  A,C
        SUB  E
        PUSH PSW                        ;A=# OF BYTES IN LINE
        CALL FNDLN                      ;FIND THIS LINE IN SAVE
        PUSH D                          ;AREA, DE->SAVE AREA
        JNZ  ST4                        ;NZ:NOT FOUND, INSERT
        PUSH D                          ;Z:FOUND, DELETE IT
        CALL FNDNXT                     ;FIND NEXT LINE
                                        ;DE->NEXT LINE
        POP  B                          ;BC->LINE TO BE DELETED
        LHLD TXTUNF                     ;HL->UNFILLED SAVE AREA
        CALL MVUP                       ;MOVE UP TO DELETE
        MOV  H,B                        ;TXTUNF->UNFILLED AREA
        MOV  L,C
        SHLD TXTUNF                     ;UPDATE
ST4:    POP  B                          ;GET READY TO INSERT
        LHLD TXTUNF                     ;BUT FIRST CHECK IF
        POP  PSW                        ;THE LENGTH OF NEW LINE
        PUSH H                          ;IS 3 (LINE # AND CR)
        CPI  3                          ;THEN DO NOT INSERT
        JZ   RSTART                     ;MUST CLEAR THE STACK
        ADD  L                          ;COMPUTE NEW TXTUNF
        MOV  L,A
        MVI  A,0
        ADC  H
        MOV  H,A                        ;HL->NEW UNFILLED AREA
        LXI  D,TXTEND                   ;CHECK TO SEE IF THERE
        RST  4                          ;IS ENOUGH SPACE
        JNC  QSORRY                     ;SORRY, NO ROOM FOR IT
        SHLD TXTUNF                     ;OK, UPDATE TXTUNF
        POP  D                          ;DE->OLD UNFILLED AREA
        CALL MVDOWN
        POP  D                          ;DE->BEGIN, HL->END
        POP  H
        CALL MVUP                       ;MOVE NEW LINE TO SAVE
```

```
        JMP  ST3                        ;AREA
;
;*********************************************************
;
; WHAT FOLLOWS IS THE CODE TO EXECUTE DIRECT AND STATEMENT
; COMMANDS.  CONTROL IS TRANSFERED TO THESE POINTS VIA THE
; COMMAND TABLE LOOKUP CODE OF 'DIRECT' AND 'EXEC' IN LAST
; SECTION.  AFTER THE COMMAND IS EXECUTED, CONTROL IS
; TRANSFERED TO OTHERS SECTIONS AS FOLLOWS:
;
; FOR 'LIST', 'NEW', AND 'STOP': GO BACK TO 'RSTART'
; FOR 'RUN': GO EXECUTE THE FIRST STORED LINE IF ANY, ELSE
; GO BACK TO 'RSTART'.
; FOR 'GOTO' AND 'GOSUB': GO EXECUTE THE TARGET LINE.
; FOR 'RETURN' AND 'NEXT': GO BACK TO SAVED RETURN LINE.
; FOR ALL OTHERS: IF 'CURRENT' -> 0, GO TO 'RSTART', ELSE
; GO EXECUTE NEXT COMMAND.  (THIS IS DONE IN 'FINISH'.)
;*********************************************************
;
; *** NEW *** STOP *** RUN (& FRIENDS) *** & GOTO ***
;
; 'NEW(CR)' SETS 'TXTUNF' TO POINT TO 'TXTBGN'
;
; 'STOP(CR)' GOES BACK TO 'RSTART'
;
; 'RUN(CR)' FINDS THE FIRST STORED LINE, STORE ITS ADDRESS (IN
; 'CURRENT'), AND START EXECUTE IT.  NOTE THAT ONLY THOSE
; COMMANDS IN TAB2 ARE LEGAL FOR STORED PROGRAM.
;
; THERE ARE 3 MORE ENTRIES IN 'RUN':
; 'RUNNXL' FINDS NEXT LINE, STORES ITS ADDR. AND EXECUTES IT.
; 'RUNTSL' STORES THE ADDRESS OF THIS LINE AND EXECUTES IT.
; 'RUNSML' CONTINUES THE EXECUTION ON SAME LINE.
;
; 'GOTO EXPR(CR)' EVALUATES THE EXPRESSION, FIND THE TARGET
; LINE, AND JUMP TO 'RUNTSL' TO DO IT.
;
NEW:    CALL ENDCHK                     ;*** NEW(CR) ***
        LXI  H,TXTBGN
        SHLD TXTUNF
;
STOP:   CALL ENDCHK                     ;*** STOP(CR) ***
        JMP  RSTART
;
RUN:    CALL ENDCHK                     ;*** RUN(CR) ***
        LXI  D,TXTBGN                   ;FIRST SAVED LINE
;
RUNNXL: LXI  H,0                        ;*** RUNNXL ***
        CALL FNDLP                      ;FIND WHATEVER LINE #
        JC   RSTART                     ;C:PASSED TXTUNF, QUIT
;
RUNTSL: XCHG                            ;*** RUNTSL ***
        SHLD CURRNT                     ;SET 'CURRENT'->LINE #
        XCHG
        INX  D                          ;BUMP PASS LINE #
        INX  D
;
RUNSML: CALL CHKIO                      ;*** RUNSML ***
        LXI  H,TAB2-1                   ;FIND COMMAND IN TAB2
        JMP  EXEC                       ;AND EXECUTE IT
;
GOTO:   RST  3                          ;*** GOTO EXPR ***
        PUSH D                          ;SAVE FOR ERROR ROUTINE
        CALL ENDCHK                     ;MUST FIND A CR
        CALL FNDLN                      ;FIND THE TARGET LINE
```

```
        JNZ  AHOW                       ;NO SUCH LINE #
        POP  PSW                        ;CLEAR THE PUSH DE
        JMP  RUNTSL                     ;GO DO IT
;
;*********************************************************
;
; *** LIST *** & PRINT ***
;
; LIST HAS TWO FORMS:
; 'LIST(CR)' LISTS ALL SAVED LINES
; 'LIST #(CR)' START LIST AT THIS LINE #
; YOU CAN STOP THE LISTING BY CONTROL C KEY
;
; PRINT COMMAND IS 'PRINT ....;' OR 'PRINT ....(CR)'
; WHERE '....' IS A LIST OF EXPRESIONS, FORMATS, BACK-
; ARROWS, AND STRINGS.  THESE ITEMS ARE SEPERATED BY COMMAS.
;
; A FORMAT IS A POUND SIGN FOLLOWED BY A NUMBER.  IT CONTROLS
; THE NUMBER OF SPACES THE VALUE OF A EXPRESION IS GOING TO
; BE PRINTED.  IT STAYS EFFECTIVE FOR THE REST OF THE PRINT
; COMMAND UNLESS CHANGED BY ANOTHER FORMAT.  IF NO FORMAT IS
; SPECIFIED, 6 POSITIONS WILL BE USED.
;
; A STRING IS QUOTED IN A PAIR OF SINGLE QUOTES OR A PAIR OF
; DOUBLE QUOTES.
;
; A BACK-ARROW MEANS GENERATE A (CR) WITHOUT (LF)
;
; A (CRLF) IS GENERATED AFTER THE ENTIRE LIST HAS BEEN
; PRINTED OR IF THE LIST IS A NULL LIST.  HOWEVER IF THE LIST
; ENDED WITH A COMMA, NO (CRLF) IS GENERATED.
;
LIST:   CALL TSTNUM                     ;TEST IF THERE IS A #
        CALL ENDCHK                     ;IF NO # WE GET A 0
        CALL FNDLN                      ;FIND THIS OR NEXT LINE
LS1:    JC   RSTART                     ;C:PASSED TXTUNF
        CALL PRTLN                      ;PRINT THE LINE
        CALL CHKIO                      ;STOP IF HIT CONTROL-C
        CALL FNDLP                      ;FIND NEXT LINE
        JMP  LS1                        ;AND LOOP BACK
;
PRINT:  MVI  C,6                        ;C = # OF SPACES
        RST  1                          ;IF NULL LIST & ";"
        DB   3BH
        DB   PR2-$-1
        CALL CRLF                       ;GIVE CR-LF AND
        JMP  RUNSML                     ;CONTINUE SAME LINE
PR2:    RST  1                          ;IF NULL LIST (CR)
        DB   CR
        DB   PR0-$-1
        CALL CRLF                       ;ALSO GIVE CR-LF AND
        JMP  RUNNXL                     ;GO TO NEXT LINE
PR0:    RST  1                          ;ELSE IS IT FORMAT?
        DB   '#'
        DB   PR1-$-1
        RST  3                          ;YES, EVALUATE EXPR.
        MOV  C,L                        ;AND SAVE IT IN C
        JMP  PR3                        ;LOOK FOR MORE TO PRINT
PR1:    CALL QTSTG                      ;OR IS IT A STRING?
        JMP  PR8                        ;IF NOT, MUST BE EXPR.
PR3:    RST  1                          ;IF ",", GO FIND NEXT
        DB   ','
        DB   PR6-$-1
        CALL FIN                        ;IN THE LIST.
        JMP  PR0                        ;LIST CONTINUES
```

```
PR6:    CALL CRLF                       ;LIST ENDS
        RST  6
PR8:    RST  3                          ;EVALUATE THE EXPR
        PUSH B
        CALL PRTNUM                     ;PRINT THE VALUE
        POP  B
        JMP  PR3                        ;MORE TO PRINT?
;
;*************************************************************
;
; *** GOSUB *** & RETURN ***
;
; 'GOSUB EXPR;' OR 'GOSUB EXPR (CR)' IS LIKE THE 'GOTO'
; COMMAND, EXCEPT THAT THE CURRENT TEXT POINTER, STACK POINTER
; ETC. ARE SAVE SO THAT EXECUTION CAN BE CONTINUED AFTER THE
; SUBROUTINE 'RETURN'.  IN ORDER THAT 'GOSUB' CAN BE NESTED
; (AND EVEN RECURSIVE), THE SAVE AREA MUST BE STACKED.
; THE STACK POINTER IS SAVED IN 'STKGOS', THE OLD 'STKGOS' IS
; SAVED IN THE STACK.  IF WE ARE IN THE MAIN ROUTINE, 'STKGOS'
; IS ZERO (THIS WAS DONE BY THE "MAIN" SECTION OF THE CODE),
; BUT WE STILL SAVE IT AS A FLAG FOR NO FURTHER 'RETURN'S.
;
; 'RETURN(CR)' UNDOS EVERYTHING THAT 'GOSUB' DID, AND THUS
; RETURN THE EXECUTION TO THE COMMAND AFTER THE MOST RECENT
; 'GOSUB'.  IF 'STKGOS' IS ZERO, IT INDICATES THAT WE
; NEVER HAD A 'GOSUB' AND IS THUS AN ERROR.
;
GOSUB:  CALL PUSHA                      ;SAVE THE CURRENT "FOR"
        RST  3                          ;PARAMETERS
        PUSH D                          ;AND TEXT POINTER
        CALL FNDLN                      ;FIND THE TARGET LINE
        JNZ  AHOW                       ;NOT THERE. SAY "HOW?"
        LHLD CURRNT                     ;FOUND IT, SAVE OLD
        PUSH H                          ;'CURRNT' OLD 'STKGOS'
        LHLD STKGOS
        PUSH H
        LXI  H,0                        ;AND LOAD NEW ONES
        SHLD LOPVAR
        DAD  SP
        SHLD STKGOS
        JMP  RUNTSL                     ;THEN RUN THAT LINE
RETURN: CALL ENDCHK                     ;THERE MUST BE A CR
        LHLD STKGOS                     ;OLD STACK POINTER
        MOV  A,H                        ;0 MEANS NOT EXIST
        ORA  L
        JZ   QWHAT                      ;SO, WE SAY: "WHAT?"
        SPHL                            ;ELSE, RESTORE IT
        POP  H
        SHLD STKGOS                     ;AND THE OLD 'STKGOS'
        POP  H
        SHLD CURRNT                     ;AND THE OLD 'CURRNT'
        POP  D                          ;OLD TEXT POINTER
        CALL POPA                       ;OLD "FOR" PARAMETERS
        RST  6                          ;AND WE ARE BACK HOME
;
;*************************************************************
;
; *** 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 CHECK 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.
 ;
 FOR:    CALL PUSHA                      ;SAVE THE OLD SAVE AREA
         CALL SETVAL                     ;SET THE CONTROL VAR.
         DCX  H                          ;HL IS ITS ADDRESS
         SHLD LOPVAR                     ;SAVE THAT
         LXI  H,TAB5-1                   ;USE 'EXEC' TO LOOK
         JMP  EXEC                       ;FOR THE WORD 'TO'
 FR1:    RST  3                          ;EVALUATE THE LIMIT
         SHLD LOPLMT                     ;SAVE THAT
         LXI  H,TAB6-1                   ;USE 'EXEC' TO LOOK
         JMP EXEC                        ;FOR THE WORD 'STEP'
 FR2:    RST  3                          ;FOUND IT, GET STEP
         JMP  FR4
 FR3:    LXI  H,1H                       ;NOT FOUND, SET TO 1
 FR4:    SHLD LOPINC                     ;SAVE THAT TOO
 FR5:    LHLD CURRNT                     ;SAVE CURRENT LINE #
         SHLD LOPLN
         XCHG                            ;AND TEXT POINTER
         SHLD LOPPT
         LXI  B,0AH                      ;DIG INTO STACK TO
         LHLD LOPVAR                     ;FIND 'LOPVAR'
         XCHG
         MOV  H,B
         MOV  L,B                        ;HL=0 NOW
         DAD  SP                         ;HERE IS THE STACK
         DB   3EH
 FR7:    DAD  B                          ;EACH LEVEL IS 10 DEEP
         MOV  A,M                        ;GET THAT OLD 'LOPVAR'
         INX  H
         ORA  M
         JZ   FR8                        ;0 SAYS NO MORE IN IT
         MOV  A,M
         DCX  H
         CMP  D                          ;SAME AS THIS ONE?
         JNZ  FR7
         MOV  A,M                        ;THE OTHER HALF?
         CMP  E
         JNZ  FR7
         XCHG                            ;YES, FOUND ONE
         LXI  H,0H
         DAD  SP                         ;TRY TO MOVE SP
         MOV  B,H
         MOV  C,L
         LXI  H,0AH
         DAD  D
```

```
        CALL MVDOWN                     ;AND PURGE 10 WORDS
        SPHL                            ;IN THE STACK
FR8:    LHLD LOPPT                      ;JOB DONE, RESTORE DE
        XCHG
        RST  6                          ;AND CONTINUE
;
NEXT:   RST  7                          ;GET ADDRESS OF VAR.
        JC   QWHAT                      ;NO VARIABLE, "WHAT?"
        SHLD VARNXT                     ;YES, SAVE IT
NX0:    PUSH D                          ;SAVE TEXT POINTER
        XCHG
        LHLD LOPVAR                     ;GET VAR. IN 'FOR'
        MOV  A,H
        ORA  L                          ;0 SAYS NEVER HAD ONE
        JZ   AWHAT                      ;SO WE ASK: "WHAT?"
        RST  4                          ;ELSE WE CHECK THEM
        JZ   NX3                        ;OK, THEY AGREE
        POP  D                          ;NO, LET'S SEE
        CALL POPA                       ;PURGE CURRENT LOOP
        LHLD VARNXT                     ;AND POP ONE LEVEL
        JMP  NX0                        ;GO CHECK AGAIN
NX3:    MOV  E,M                        ;COME HERE WHEN AGREED
        INX  H
        MOV  D,M                        ;DE=VALUE OF VAR.
        LHLD LOPINC
        PUSH H
        MOV  A,H
        XRA  D
        MOV  A,D
        DAD  D                          ;ADD ONE STEP
        JM   NX4
        XRA  H
        JM   NX5
NX4:    XCHG
        LHLD LOPVAR                     ;PUT IT BACK
        MOV  M,E
        INX  H
        MOV  M,D
        LHLD LOPLMT                     ;HL->LIMIT
        POP  PSW                        ;OLD HL
        ORA  A
        JP   NX1                        ;STEP > 0
        XCHG                            ;STEP < 0
NX1:    CALL CKHLDE                     ;COMPARE WITH LIMIT
        POP  D                          ;RESTORE TEXT POINTER
        JC   NX2                        ;OUTSIDE LIMIT
        LHLD LOPLN                      ;WITHIN LIMIT, GO
        SHLD CURRNT                     ;BACK TO THE SAVED
        LHLD LOPPT                      ;'CURRNT' AND TEXT
        XCHG                            ;POINTER
        RST  6
NX5:    POP  H
        POP  D
NX2:    CALL POPA                       ;PURGE THIS LOOP
        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 PROCEDED 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'.
;
REM:    LXI  H,0H                       ;*** REM ***
        DB   3EH                        ;THIS IS LIKE 'IF 0'
;
IFF:    RST  3                          ;*** IF ***
        MOV  A,H                        ;IS THE EXPR.=0?
        ORA  L
        JNZ  RUNSML                     ;NO, CONTINUE
        CALL FNDSKP                     ;YES, SKIP REST OF LINE
        JNC  RUNTSL                     ;AND RUN THE NEXT LINE
        JMP  RSTART                     ;IF NO NEXT, RE-START
;
INPERR: LHLD STKINP                     ;*** INPERR ***
        SPHL                            ;RESTORE OLD SP
        POP  H                          ;AND OLD 'CURRNT'
        SHLD CURRNT
        POP  D                          ;AND OLD TEXT POINTER
        POP  D                          ;REDO INPUT
;
INPUT:                                  ;*** INPUT ***
IP1:    PUSH D                          ;SAVE IN CASE OF ERROR
        CALL QTSTG                      ;IS NEXT ITEM A STRING?
        JMP  IP2                        ;NO
        RST  7                          ;YES, BUT FOLLOWED BY A
        JC   IP4                        ;VARIABLE?   NO.
        JMP  IP3                        ;YES.  INPUT VARIABLE
IP2:    PUSH D                          ;SAVE FOR 'PRTSTG'
        RST  7                          ;MUST BE VARIABLE NOW
        JC   QWHAT                      ;"WHAT?" IT IS NOT?
        LDAX D                          ;GET READY FOR 'PRTSTR'
        MOV  C,A
        SUB  A
        STAX D
        POP  D
        CALL PRTSTG                     ;PRINT STRING AS PROMPT
        MOV  A,C                        ;RESTORE TEXT
        DCX  D
        STAX D
IP3:    PUSH D                          ;SAVE TEXT POINTER
```

```
         XCHG
         LHLD CURRNT                    ;ALSO SAVE 'CURRNT'
         PUSH H
         LXI  H,IP1                     ;A NEGATIVE NUMBER
         SHLD CURRNT                    ;AS A FLAG
         LXI  H,0H                      ;SAVE SP TOO
         DAD  SP
         SHLD STKINP
         PUSH D                         ;OLD HL
         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'
         SHLD CURRNT
         POP  D                         ;AND OLD TEXT POINTER
 IP4:    POP  PSW                       ;PURGE JUNK IN STACK
         RST  1                         ;IS NEXT CH. ','?
         DB   ','
         DB   IP5-$-1
         JMP  IP1                       ;YES, MORE ITEMS.
 IP5:    RST  6
 ;
 DEFLT:  LDAX D                         ;***  DEFLT ***
         CPI  CR                        ;EMPTY LINE IS OK
         JZ   LT1                       ;ELSE IT IS 'LET'
 ;
 LET:    CALL SETVAL                    ;*** LET ***
         RST  1                         ;SET VALUE TO VAR.
         DB   ','
         DB   LT1-$-1
         JMP  LET                       ;ITEM BY ITEM
 LT1:    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 -)<EXPR3>(+ OR -<EXPR3>)(....)
 ; WHERE () ARE OPTIONAL AND (....) ARE OPTIONAL REPEATS.
 ; <EXPR3>::=<EXPR4>(* OR /><EXPR4>)(....)
 ; <EXPR4>::=<VARIABLE>
 ;           <FUNCTION>
 ;           (<EXPR>)
 ; <EXPR> IS RECURSIVE SO THAT VARIABLE '@' CAN HAVE AN <EXPR>
 ; AS INDEX, FUNCTIONS CAN HAVE AN <EXPR> AS ARGUMENTS, AND
 ; <EXPR4> CAN BE AN <EXPR> IN PARANTHESE.
 ;
 ;EXPR:  CALL EXPR2                     ;THIS IS AT LOC. 18
 ;       PUSH H                         ;SAVE <EXPR2> VALUE
 EXPR1:  LXI  H,TAB8-1                  ;LOOKUP REL.OP.
         JMP  EXEC                      ;GO DO IT
```

```
XP11:   CALL XP18                       ;REL.OP.">="
        RC                              ;NO, RETURN HL=0
        MOV  L,A                        ;YES, RETURN HL=1
        RET
XP12:   CALL XP18                       ;REL.OP."#"
        RZ                              ;FALSE, RETURN HL=0
        MOV  L,A                        ;TRUE, RETURN HL=1
        RET
XP13:   CALL XP18                       ;REL.OP.">"
        RZ                              ;FALSE
        RC                              ;ALSO FALSE, HL=0
        MOV  L,A                        ;TRUE, HL=1
        RET
XP14:   CALL XP18                       ;REL.OP."<="
        MOV  L,A                        ;SET HL=1
        RZ                              ;REL. TRUE, RETURN
        RC
        MOV  L,H                        ;ELSE SET HL=0
        RET
XP15:   CALL XP18                       ;REL.OP."="
        RNZ                             ;FALSE, RETURN HL=0
        MOV  L,A                        ;ELSE SET HL=1
        RET
XP16:   CALL XP18                       ;REL.OP."<"
        RNC                             ;FALSE, RETURN HL=0
        MOV  L,A                        ;ELSE SET HL=1
        RET
XP17:   POP  H                          ;NOT .REL.OP
        RET                             ;RETURN HL=<EXPR2>
XP18:   MOV  A,C                        ;SUBROUTINE FOR ALL
        POP  H                          ;REL.OP.'S
        POP  B
        PUSH H                          ;REVERSE TOP OF STACK
        PUSH B
        MOV  C,A
        CALL EXPR2                      ;GET 2ND <EXPR2>
        XCHG                            ;VALUE IN DE NOW
        XTHL                            ;1ST <EXPR2> IN HL
        CALL CKHLDE                     ;COMPARE 1ST WITH 2ND
        POP  D                          ;RESTORE TEXT POINTER
        LXI  H,0H                       ;SET HL=0, A=1
        MVI  A,1
        RET
;
EXPR2:  RST  1                          ;NEGATIVE SIGN?
        DB   '-'
        DB   XP21-$-1
        LXI  H,0H                       ;YES, FAKE '0-'
        JMP  XP26                       ;TREAT LIKE SUBTRACT
XP21:   RST  1                          ;POSITIVE SIGN? IGNORE
        DB   '+'
        DB   XP22-$-1
XP22:   CALL EXPR3                      ;1ST <EXPR3>
XP23:   RST  1                          ;ADD?
        DB   '+'
        DB   XP25-$-1
        PUSH H                          ;YES, SAVE VALUE
        CALL EXPR3                      ;GET 2ND <EXPR3>
XP24:   XCHG                            ;2ND IN DE
        XTHL                            ;1ST IN HL
        MOV  A,H                        ;COMPARE SIGN
        XRA  D
        MOV  A,D
        DAD  D
        POP  D                          ;RESTORE TEXT POINTER
```

```
        JM   XP23                       ;1ST AND 2ND SIGN DIFFER
        XRA  H                          ;1ST AND 2ND SIGN EQUAL
        JP   XP23                       ;SO IS RESULT
        JMP  QHOW                       ;ELSE WE HAVE OVERFLOW
XP25:   RST  1                          ;SUBTRACT?
        DB   '-'
        DB   XP42-$-1
XP26:   PUSH H                          ;YES, SAVE 1ST <EXPR3>
        CALL EXPR3                      ;GET 2ND <EXPR3>
        CALL CHGSGN                     ;NEGATE
        JMP  XP24                       ;AND ADD THEM
;
EXPR3:  CALL EXPR4                      ;GET 1ST <EXPR4>
XP31:   RST  1                          ;MULTIPLY?
        DB   '*'
        DB   XP34-$-1
        PUSH H                          ;YES, SAVE 1ST
        CALL EXPR4                      ;AND GET 2ND <EXPR4>
        MVI  B,0H                       ;CLEAR B FOR SIGN
        CALL CHKSGN                     ;CHECK SIGN
        XTHL                            ;1ST IN HL
        CALL CHKSGN                     ;CHECK SIGN OF 1ST
        XCHG
        XTHL
        MOV  A,H                        ;IS HL > 255 ?
        ORA  A
        JZ   XP32                       ;NO
        MOV  A,D                        ;YES, HOW ABOUT DE
        ORA  D
        XCHG                            ;PUT SMALLER IN HL
        JNZ  AHOW                       ;ALSO >, WILL OVERFLOW
XP32:   MOV  A,L                        ;THIS IS DUMB
        LXI  H,0H                       ;CLEAR RESULT
        ORA  A                          ;ADD AND COUNT
        JZ   XP35
XP33:   DAD  D
        JC   AHOW                       ;OVERFLOW
        DCR  A
        JNZ  XP33
        JMP  XP35                       ;FINISHED
XP34:   RST  1                          ;DIVIDE?
        DB   '/'
        DB   XP42-$-1
        PUSH H                          ;YES, SAVE 1ST <EXPR4>
        CALL EXPR4                      ;AND GET THE SECOND ONE
        MVI  B,0H                       ;CLEAR B FOR SIGN
        CALL CHKSGN                     ;CHECK SIGN OF 2ND
        XTHL                            ;GET 1ST IN HL
        CALL CHKSGN                     ;CHECK SIGN OF 1ST
        XCHG
        XTHL
        XCHG
        MOV  A,D                        ;DIVIDE BY 0?
        ORA  E
        JZ   AHOW                       ;SAY "HOW?"
        PUSH B                          ;ELSE SAVE SIGN
        CALL DIVIDE                     ;USE SUBROUTINE
        MOV  H,B                        ;RESULT IN HL NOW
        MOV  L,C
        POP  B                          ;GET SIGN BACK
XP35:   POP  D                          ;AND TEXT POINTER
        MOV  A,H                        ;HL MUST BE +
        ORA  A
        JM   QHOW                       ;ELSE IT IS OVERFLOW
        MOV  A,B
```

```
        ORA  A
        CM   CHGSGN                ;CHANGE SIGN IF NEEDED
        JMP  XP31                  ;LOOK FOR MORE TERMS
;
EXPR4:  LXI  H,TAB4-1              ;FIND FUNCTION IN TAB4
        JMP  EXEC                  ;AND GO DO IT
XP40:   RST  7                     ;NO, NOT A FUNCTION
        JC   XP41                  ;NOR A VARIABLE
        MOV  A,M                   ;VARIABLE
        INX  H
        MOV  H,M                   ;VALUE IN HL
        MOV  L,A
        RET
XP41:   CALL TSTNUM                ;OR IS IT A NUMBER
        MOV  A,B                   ;# OF DIGIT
        ORA  A
        RNZ                        ;OK
PARN:   RST  1
        DB   '('
        DB   XP43-$-1
        RST  3                     ;"(EXPR)"
        RST  1
        DB   ')'
        DB   XP43-$-1
XP42:   RET
XP43:   JMP  QWHAT                 ;ELSE SAY: "WHAT?"
;
RND:    CALL PARN                  ;*** RND(EXPR) ***
        MOV  A,H                   ;EXPR MUST BE +
        ORA  A
        JM   QHOW
        ORA  L                     ;AND NON-ZERO
        JZ   QHOW
        PUSH D                     ;SAVE BOTH
        PUSH H
        LHLD RANPNT                ;GET MEMORY AS RANDOM
        LXI  D,LSTROM              ;NUMBER
        RST  4
        JC   RA1                   ;WRAP AROUND IF LAST
        LXI  H,START
RA1:    MOV  E,M
        INX  H
        MOV  D,M
        SHLD RANPNT
        POP  H
        XCHG
        PUSH B
        CALL DIVIDE                ;RND(N)=MOD(M,N)+1
        POP  B
        POP  D
        INX  H
        RET
;
ABS:    CALL PARN                  ;*** ABS(EXPR) ***
        DCX  D
        CALL CHKSGN                ;CHECK SIGN
        INX  D
        RET
;
SIZE:   LHLD TXTUNF                ;*** SIZE ***
        PUSH D                     ;GET THE NUMBER OF FREE
        XCHG                       ;BYTES BETWEEN 'TXTUNF'
        LXI  H,VARBGN              ;AND 'VARBGN'
        CALL SUBDE
        POP  D
```

```
        RET
;
;*********************************************************
;
; *** DIVIDE *** SUBDE *** CHKSGN *** CHGSGN *** & CKHLDE ***
;
; 'DIVIDE' DIVIDES HL BY DE, RESULT IN BC, REMAINDER IN HL
;
; 'SUBDE' SUBSTRACTS DE FROM HL
;
; 'CHKSGN' CHECKS SIGN OF HL.  IF +, NO CHANGE.  IF -, CHANGE
; SIGN AND FLIP SIGN OF B.
;
; 'CHGSGN' CHECKS SIGN N OF HL AND B UNCONDITIONALLY.
;
; 'CKHLDE' CHECKS SIGN OF HL AND DE.  IF DIFFERENT, HL AND DE
; ARE INTERCHANGED.  IF SAME SIGN, NOT INTERCHANGED.  EITHER
; CASE, HL DE ARE THEN COMPARED TO SET THE FLAGS.
;
DIVIDE: PUSH H                          ;*** DIVIDE ***
        MOV  L,H                        ;DIVIDE H BY DE
        MVI  H,0
        CALL DV1
        MOV  B,C                        ;SAVE RESULT IN B
        MOV  A,L                        ;(REMINDER+L)/DE
        POP  H
        MOV  H,A
DV1:    MVI  C,0FFH                     ;RESULT IN C
DV2:    INR  C                          ;DUMB ROUTINE
        CALL SUBDE                      ;DIVIDE BY SUBTRACT
        JNC  DV2                        ;AND COUNT
        DAD  D
        RET
;
SUBDE:  MOV  A,L                        ;*** SUBDE ***
        SUB  E                          ;SUBSTRACT DE FROM
        MOV  L,A                        ;HL
        MOV  A,H
        SBB  D
        MOV  H,A
        RET
;
CHKSGN: MOV  A,H                        ;*** CHKSGN ***
        ORA  A                          ;CHECK SIGN OF HL
        RP                              ;IF -, CHANGE SIGN
;
CHGSGN: MOV  A,H                        ;*** CHGSGN ***
        PUSH PSW
        CMA                             ;CHANGE SIGN OF HL
        MOV  H,A
        MOV  A,L
        CMA
        MOV  L,A
        INX  H
        POP  PSW
        XRA  H
        JP   QHOW
        MOV  A,B                        ;AND ALSO FLIP B
        XRI  80H
        MOV  B,A
        RET
;
CKHLDE: MOV  A,H
        XRA  D                          ;SAME SIGN?
        JP   CK1                        ;YES, COMPARE
```

```
        XCHG                            ;NO, XCH AND COMP
CK1:    RST  4
        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 (AND 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 TBI 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:
; 'QWHAT' SAVES TEXT POINTER IN STACK AND GET MESSAGE "WHAT?"
; 'AWHAT' JUST GET MESSAGE "WHAT?" AND JUMP TO 'ERROR'.
; 'QSORRY' AND 'ASORRY' DO SAME KIND OF THING.
; 'AHOW' AND 'AHOW' IN THE ZERO PAGE SECTION ALSO DO THIS.
;
SETVAL: RST  7                          ;*** SETVAL ***
        JC   QWHAT                      ;"WHAT?" NO VARIABLE
        PUSH H                          ;SAVE ADDRESS OF VAR.
        RST  1                          ;PASS "=" SIGN
        DB   '='
        DB   SV1-$-1
        RST  3                          ;EVALUATE EXPR.
        MOV  B,H                        ;VALUE IS IN BC NOW
        MOV  C,L
        POP  H                          ;GET ADDRESS
        MOV  M,C                        ;SAVE VALUE
        INX  H
        MOV  M,B
        RET
SV1:    JMP  QWHAT                      ;NO "=" SIGN
;
FIN:    RST  1                          ;*** FIN ***
        DB   3BH
        DB   FI1-$-1
        POP  PSW                        ;";", PURGE RET. ADDR.
        JMP  RUNSML                     ;CONTINUE SAME LINE
FI1:    RST  1                          ;NOT ";", IS IT CR?
        DB   CR
        DB   FI2-$-1
        POP  PSW                        ;YES, PURGE RET. ADDR.
        JMP  RUNNXL                     ;RUN NEXT LINE
FI2:    RET                             ;ELSE RETURN TO CALLER
;
ENDCHK: RST  5                          ;*** ENDCHK ***
        CPI  CR                         ;END WITH CR?
```

```
        RZ                              ;OK, ELSE SAY: "WHAT?"
;
QWHAT:  PUSH D                          ;*** QWHAT ***
AWHAT:  LXI  D,WHAT                     ;*** AWHAT ***
ERROR:  SUB  A                          ;*** ERROR ***
        CALL PRTSTG                     ;PRINT 'WHAT?', 'HOW?'
        POP  D                          ;OR 'SORRY'
        LDAX D                          ;SAVE THE CHARACTER
        PUSH PSW                        ;AT WHERE OLD DE ->
        SUB  A                          ;AND PUT A 0 THERE
        STAX D
        LHLD CURRNT                     ;GET CURRENT LINE #
        PUSH H
        MOV  A,M                        ;CHECK THE VALUE
        INX  H
        ORA  M
        POP  D
        JZ   RSTART                     ;IF ZERO, JUST RESTART
        MOV  A,M                        ;IF NEGATIVE,
        ORA  A
        JM   INPERR                     ;REDO INPUT
        CALL PRTLN                      ;ELSE PRINT THE LINE
        DCX  D                          ;UPTO WHERE THE 0 IS
        POP  PSW                        ;RESTORE THE CHARACTER
        STAX D
        MVI  A,3FH                      ;PRINT A "?"
        RST  2
        SUB  A                          ;AND THE REST OF THE
        CALL PRTSTG                     ;LINE
        JMP  RSTART                     ;THEN RESTART
;
QSORRY: PUSH D                          ;*** QSORRY ***
ASORRY: LXI  D,SORRY                    ;*** ASORRY ***
        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 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 AREA 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 #.
; 'FNDNXT' WILL BUMP DE BY 2, FIND A CR AND THEN START SEARCH.
; 'FNDSKP' USE DE TO FIND A CR, AND THEN START SEARCH.
;
GETLN:  RST  2                          ;*** GETLN ***
        LXI  D,BUFFER                   ;PROMPT AND INIT.
GL1:    CALL CHKIO                      ;CHECK KEYBOARD
```

```
        JZ   GL1                        ;NO INPUT, WAIT
        CPI  7FH                        ;DELETE LAST CHARACTER?
        JZ   GL3                        ;YES
        RST  2                          ;INPUT, ECHO BACK
        CPI  0AH                        ;IGNORE LF
        JZ   GL1
        ORA  A                          ;IGNORE NULL
        JZ   GL1
        CPI  7DH                        ;DELETE THE WHOLE LINE?
        JZ   GL4                        ;YES
        STAX D                          ;ELSE SAVE INPUT
        INX  D                          ;AND BUMP POINTER
        CPI  0DH                        ;WAS IT CR?
        RZ                              ;YES, END OF LINE
        MOV  A,E                        ;ELSE MORE FREE ROOM?
        CPI  BUFEND AND 0FFH
        JNZ  GL1                        ;YES, GET NEXT INPUT
GL3:    MOV  A,E                        ;DELETE LAST CHARACTER
        CPI  BUFFER AND 0FFH            ;BUT DO WE HAVE ANY?
        JZ   GL4                        ;NO, REDO WHOLE LINE
        DCX  D                          ;YES, BACKUP POINTER
        MVI  A,5CH                      ;AND ECHO A BACK-SLASH
        RST  2
        JMP  GL1                        ;GO GET NEXT INPUT
GL4:    CALL CRLF                       ;REDO ENTIRE LINE
        MVI  A,05EH                     ;CR, LF AND UP-ARROW
        JMP  GETLN
;
FNDLN:  MOV  A,H                        ;*** FNDLN ***
        ORA  A                          ;CHECK SIGN OF HL
        JM   QHOW                       ;IT CANNOT BE -
        LXI  D,TXTBGN                   ;INIT TEXT POINTER
;
FNDLP:                                  ;*** FDLNP ***
FL1:    PUSH H                          ;SAVE LINE #
        LHLD TXTUNF                     ;CHECK IF WE PASSED END
        DCX  H
        RST  4
        POP  H                          ;GET LINE # BACK
        RC                              ;C,NZ PASSED END
        LDAX D                          ;WE DID NOT, GET BYTE 1
        SUB  L                          ;IS THIS THE LINE?
        MOV  B,A                        ;COMPARE LOW ORDER
        INX  D
        LDAX D                          ;GET BYTE 2
        SBB  H                          ;COMPARE HIGH ORDER
        JC   FL2                        ;NO, NOT THERE YET
        DCX  D                          ;ELSE WE EITHER FOUND
        ORA  B                          ;IT, OR IT IS NOT THERE
        RET                             ;NC,Z:FOUND, NC,NZ:NO
;
FNDNXT:                                 ;*** FNDNXT ***
        INX  D                          ;FIND NEXT LINE
FL2:    INX  D                          ;JUST PASSED BYTE 1 & 2
;
FNDSKP: LDAX D                          ;*** FNDSKP ***
        CPI  CR                         ;TRY TO FIND CR
        JNZ  FL2                        ;KEEP LOOKING
        INX  D                          ;FOUND CR, SKIP OVER
        JMP  FL1                        ;CHECK IF END OF TEXT
;
;*********************************************************
;
; *** PRTSTG *** QTSTG *** PRTNUM *** & PRTLN ***
;
```

```
; 'PRTSTG' 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.
;
; 'QTSTG' 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 3 BYTES OF THE CALLER IS SKIPPED
; OVER (USUALLY A JUMP INSTRUCTION.
;
; 'PRTNUM' 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.
;
PRTSTG: MOV  B,A                        ;*** PRTSTG ***
PS1:    LDAX D                          ;GET A CHARACTER
        INX  D                          ;BUMP POINTER
        CMP  B                          ;SAME AS OLD A?
        RZ                              ;YES, RETURN
        RST  2                          ;ELSE PRINT IT
        CPI  CR                         ;WAS IT A CR?
        JNZ  PS1                        ;NO, NEXT
        RET                             ;YES, RETURN
;
QTSTG:  RST  1                          ;*** QTSTG ***
        DB   '"'
        DB   QT3-$-1
        MVI  A,22H                      ;IT IS A "
QT1:    CALL PRTSTG                     ;PRINT UNTIL ANOTHER
        CPI  CR                         ;WAS LAST ONE A CR?
        POP  H                          ;RETURN ADDRESS
        JZ   RUNNXL                     ;WAS CR, RUN NEXT LINE
QT2:    INX  H                          ;SKIP 3 BYTES ON RETURN
        INX  H
        INX  H
        PCHL                            ;RETURN
QT3:    RST  1                          ;IS IT A '?
        DB   27H
        DB   QT4-$-1
        MVI  A,27H                      ;YES, DO THE SAME
        JMP  QT1                        ;AS IN "
QT4:    RST  1                          ;IS IT BACK-ARROW?
        DB   5FH
        DB   QT5-$-1
        MVI  A,08DH                     ;YES, CR WITHOUT LF
        RST  2                          ;DO IT TWICE TO GIVE
        RST  2                          ;TTY ENOUGH TIME
        POP  H                          ;RETURN ADDRESS
        JMP  QT2
QT5:    RET                             ;NONE OF ABOVE
;
PRTNUM: MVI  B,0                        ;*** PRTNUM ***
        CALL CHKSGN                     ;CHECK SIGN
        JP   PN1                        ;NO SIGN
        MVI  B,'-'                      ;B=SIGN
        DCR  C                          ;'-' TAKES SPACE
PN1:    PUSH D                          ;SAVE
        LXI  D,0AH                      ;DECIMAL
        PUSH D                          ;SAVE AS A FLAG
```

```
        DCR  C                          ;C=SPACES
        PUSH B                          ;SAVE SIGN & SPACE
PN2:    CALL DIVIDE                     ;DIVIDE HL BY 10
        MOV  A,B                        ;RESULT 0?
        ORA  C
        JZ   PN3                        ;YES, WE GOT ALL
        XTHL                            ;NO, SAVE REMAINDER
        DCR  L                          ;AND COUNT SPACE
        PUSH H                          ;HL IS OLD BC
        MOV  H,B                        ;MOVE RESULT TO BC
        MOV  L,C
        JMP  PN2                        ;AND DIVIDE BY 10
PN3:    POP  B                          ;WE GOT ALL DIGITS IN
PN4:    DCR  C                          ;THE STACK
        MOV  A,C                        ;LOOK AT SPACE COUNT
        ORA  A
        JM   PN5                        ;NO LEADING BLANKS
        MVI  A,20H                      ;LEADING BLANKS
        RST  2
        JMP  PN4                        ;MORE?
PN5:    MOV  A,B                        ;PRINT SIGN
        ORA  A
        CNZ  10H
        MOV  E,L                        ;LAST REMAINDER IN E
PN6:    MOV  A,E                        ;CHECK DIGIT IN E
        CPI  0AH                        ;10 IS FLAG FOR NO MORE
        POP  D
        RZ                              ;IF SO, RETURN
        ADI  30H                        ;ELSE CONVERT TO ASCII
        RST  2                          ;AND PRINT THE DIGIT
        JMP  PN6                        ;GO BACK FOR MORE
;
PRTLN:  LDAX D                          ;*** PRTLN ***
        MOV  L,A                        ;LOW ORDER LINE #
        INX  D
        LDAX D                          ;HIGH ORDER
        MOV  H,A
        INX  D
        MVI  C,4H                       ;PRINT 4 DIGIT LINE #
        CALL PRTNUM
        MVI  A,20H                      ;FOLLOWED BY A BLANK
        RST  2
        SUB  A                          ;AND THEN THE NEXT
        CALL PRTSTG
        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
;
MVUP:   RST  4                          ;*** MVUP ***
        RZ                              ;DE = HL, RETURN
        LDAX D                          ;GET ONE BYTE
```

```
        STAX B                          ;MOVE IT
        INX  D                          ;INCREASE BOTH POINTERS
        INX  B
        JMP  MVUP                       ;UNTIL DONE
;
MVDOWN: MOV  A,B                        ;*** MVDOWN ***
        SUB  D                          ;TEST IF DE = BC
        JNZ  MD1                        ;NO, GO MOVE
        MOV  A,C                        ;MAYBE, OTHER BYTE?
        SUB  E
        RZ                              ;YES, RETURN
MD1:    DCX  D                          ;ELSE MOVE A BYTE
        DCX  H                          ;BUT FIRST DECREASE
        LDAX D                          ;BOTH POINTERS AND
        MOV  M,A                        ;THEN DO IT
        JMP  MVDOWN                     ;LOOP BACK
;
POPA:   POP  B                          ;BC = RETURN ADDR.
        POP  H                          ;RESTORE LOPVAR, BUT
        SHLD LOPVAR                     ;=0 MEANS NO MORE
        MOV  A,H
        ORA  L
        JZ   PP1                        ;YEP, GO RETURN
        POP  H                          ;NOP, RESTORE OTHERS
        SHLD LOPINC
        POP  H
        SHLD LOPLMT
        POP  H
        SHLD LOPLN
        POP  H
        SHLD LOPPT
PP1:    PUSH B                          ;BC = RETURN ADDR.
        RET
;
PUSHA:  LXI  H,STKLMT                   ;*** PUSHA ***
        CALL CHGSGN
        POP  B                          ;BC=RETURN ADDRESS
        DAD  SP                         ;IS STACK NEAR THE TOP?
        JNC  QSORRY                     ;YES, SORRY FOR THAT
        LHLD LOPVAR                     ;ELSE SAVE LOOP VAR'S
        MOV  A,H                        ;BUT IF LOPVAR IS 0
        ORA  L                          ;THAT WILL BE ALL
        JZ   PU1
        LHLD LOPPT                      ;ELSE, MORE TO SAVE
        PUSH H
        LHLD LOPLN
        PUSH H
        LHLD LOPLMT
        PUSH H
        LHLD LOPINC
        PUSH H
        LHLD LOPVAR
PU1:    PUSH H
        PUSH B                          ;BC = RETURN ADDR.
        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.
;
; 'CHKIO' CHECKS THE INPUT.  IF NO INPUT, IT WILL RETURN TO
; THE CALLER WITH THE Z FLAG SET.  IF THERE IS INPUT, Z FLAG
; IS CLEARED AND THE INPUT BYTE IS IN A.  HOWEVER, IF THE
; INPUT IS A CONTROL-O, THE 'OCSW' SWITCH IS COMPLIMENTED, AND
; Z FLAG IS RETURNED.  IF A CONTROL-C IS READ, 'CHKIO' WILL
; RESTART TBI AND DO NOT RETURN TO THE CALLER.
;
;OUTC:  PUSH PSW                      ;THIS IS AT LOC. 10
;       LDA  OCSW                     ;CHECK SOFTWARE SWITCH
;       ORA  A
INIT:   STA  OCSW
        MVI  A,4EH                    ;Initialize 8251A UART -- 3 is status port
        OUT  3                        ;1 stop bit, no parity, 8-bit char, 16x baud
        MVI  A,37H                    ;enable receive and transmit
        OUT  3
        MVI  D,19H
PATLOP:
        CALL CRLF
        DCR  D
        JNZ  PATLOP
        SUB  A
        LXI  D,MSG1
        CALL PRTSTG
        LXI  H,START
        SHLD RANPNT
        LXI  H,TXTBGN
        SHLD TXTUNF
        JMP  RSTART
OC2:    JNZ  OC3                      ;IT IS ON
        POP  PSW                      ;IT IS OFF
        RET                           ;RESTORE AF AND RETURN
OC3:    IN   3                        ;Check status
        ANI  1H                       ;STATUS BIT
        JZ   OC3                      ;NOT READY, WAIT
        POP  PSW                      ;READY, GET OLD A BACK
        OUT  2                        ;Out to data port
        CPI  CR                       ;WAS IT CR?
        RNZ                           ;NO, FINISHED
        MVI  A,LF                     ;YES, WE SEND LF TOO
        RST  2                        ;THIS IS RECURSIVE
        MVI  A,CR                     ;GET CR BACK IN A
        RET
;
CHKIO:  IN   3                       ;*** CHKIO ***
        NOP                           ;STATUS BIT FLIPPED?
        ANI  2H                       ;MASK STATUS BIT
        RZ                            ;NOT READY, RETURN "Z"
        IN   2                       ;READY, READ DATA
        ANI  7FH                      ;MASK BIT 7 OFF
        CPI  0FH                      ;IS IT CONTROL-O?
        JNZ  CI1                      ;NO, MORE CHECKING
        LDA  OCSW                     ;CONTROL-O FLIPS OCSW
        CMA                           ;ON TO OFF, OFF TO ON
        STA  OCSW
        JMP  CHKIO                    ;GET ANOTHER INPUT
CI1:    CPI  3H                       ;IS IT CONTROL-C?
        RNZ                           ;NO, RETURN "NZ"
        JMP  RSTART                   ;YES, RESTART TBI
;
MSG1:   DB   'TINY '
        DB   'BASIC'
        DB   CR
;
```

```
;*************************************************************
;
; *** TABLES *** DIRECT *** & EXEC ***
;
; THIS SECTION OF THE CODE TESTS A STRING AGAINST A TABLE.
; WHEN A MATCH IS FOUND, CONTROL IS TRANSFERED TO THE SECTION
; OF CODE ACCORDING TO THE TABLE.
;
; AT 'EXEC', DE SHOULD POINT TO THE STRING AND HL SHOULD POINT
; TO THE TABLE-1.  AT 'DIRECT', DE SHOULD POINT TO THE STRING.
; HL WILL BE SET UP TO POINT TO TAB1-1, WHICH IS THE TABLE OF
; ALL DIRECT AND STATEMENT COMMANDS.
;
; A '.' IN THE STRING WILL TERMINATE THE TEST AND THE PARTIAL
; MATCH WILL BE CONSIDERED AS A MATCH.  E.G., 'P.', 'PR.',
; '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.
;
TAB1:                                   ;DIRECT COMMANDS
        DB   'LIST'
        DWA  LIST
        DB   'RUN'
        DWA  RUN
        DB   'NEW'
        DWA  NEW
;
TAB2:                                   ;DIRECT/STATEMENT
        DB   'NEXT'
        DWA  NEXT
        DB   'LET'
        DWA  LET
        DB   'IF'
        DWA  IFF
        DB   'GOTO'
        DWA  GOTO
        DB   'GOSUB'
        DWA  GOSUB
        DB   'RETURN'
        DWA  RETURN
        DB   'REM'
        DWA  REM
        DB   'FOR'
        DWA  FOR
        DB   'INPUT'
        DWA  INPUT
        DB   'PRINT'
        DWA  PRINT
        DB   'STOP'
        DWA  STOP
        DWA  DEFLT
;
TAB4:                                   ;FUNCTIONS
        DB   'RND'
        DWA  RND
        DB   'ABS'
        DWA  ABS
        DB   'SIZE'
```

```
        DWA  SIZE
        DWA  XP40
;
TAB5:                                   ;"TO" IN "FOR"
        DB   'TO'
        DWA  FR1
        DWA  QWHAT
;
TAB6:                                   ;"STEP" IN "FOR"
        DB   'STEP'
        DWA  FR2
        DWA  FR3
;
TAB8:                                   ;RELATION OPERATORS
        DB   '>='
        DWA  XP11
        DB   '#'
        DWA  XP12
        DB   '>'
        DWA  XP13
        DB   '='
        DWA  XP15
        DB   '<='
        DWA  XP14
        DB   '<'
        DWA  XP16
        DWA  XP17
;
DIRECT: LXI  H,TAB1-1                   ;*** DIRECT ***
;
EXEC:                                   ;*** EXEC ***
EX0:    RST  5                          ;IGNORE LEADING BLANKS
        PUSH D                          ;SAVE POINTER
EX1:    LDAX D                          ;IF FOUND '.' IN STRING
        INX  D                          ;BEFORE ANY MISMATCH
        CPI  2EH                        ;WE DECLARE A MATCH
        JZ   EX3
        INX  H                          ;HL->TABLE
        CMP  M                          ;IF MATCH, TEST NEXT
        JZ   EX1
        MVI  A,07FH                     ;ELSE SEE IF BIT 7
        DCX  D                          ;OF TABLE IS SET, WHICH
        CMP  M                          ;IS THE JUMP ADDR. (HI)
        JC   EX5                        ;C:YES, MATCHED
EX2:    INX  H                          ;NC:NO, FIND JUMP ADDR.
        CMP  M
        JNC  EX2
        INX  H                          ;BUMP TO NEXT TAB. ITEM
        POP  D                          ;RESTORE STRING POINTER
        JMP  EX0                        ;TEST AGAINST NEXT ITEM
EX3:    MVI  A,07FH                     ;PARTIAL MATCH, FIND
EX4:    INX  H                          ;JUMP ADDR., WHICH IS
        CMP  M                          ;FLAGGED BY BIT 7
        JNC  EX4
EX5:    MOV  A,M                        ;LOAD HL WITH THE JUMP
        INX  H                          ;ADDRESS FROM THE TABLE
        MOV  L,M
        ANI  7FH                        ;MASK OFF BIT 7
        MOV  H,A
        POP  PSW                        ;CLEAN UP THE GABAGE
        PCHL                            ;AND WE GO DO IT
;
LSTROM:                                 ;ALL ABOVE CAN BE ROM
;       ORG  1000H                      ;HERE DOWN MUST BE RAM
        ORG  0800H
```

```
OCSW:   DS   1                          ;SWITCH FOR OUTPUT
CURRNT: DS   2                          ;POINTS TO CURRENT LINE
STKGOS: DS   2                          ;SAVES SP IN 'GOSUB'
VARNXT: DS   2                          ;TEMP STORAGE
STKINP: DS   2                          ;SAVES SP IN 'INPUT'
LOPVAR: DS   2                          ;'FOR' LOOP SAVE AREA
LOPINC: DS   2                          ;INCREMENT
LOPLMT: DS   2                          ;LIMIT
LOPLN:  DS   2                          ;LINE NUMBER
LOPPT:  DS   2                          ;TEXT POINTER
RANPNT: DS   2                          ;RANDOM NUMBER POINTER
TXTUNF: DS   2                          ;->UNFILLED TEXT AREA
TXTBGN: DS   2                          ;TEXT SAVE AREA BEGINS
;       ORG  1366H
;       ORG  1F00H
        ORG  0F00H                      ;for 2K RAM
TXTEND: DS   0                          ;TEXT SAVE AREA ENDS
VARBGN: DS   55                         ;VARIABLE @(0)
BUFFER: DS   64                         ;INPUT BUFFER
BUFEND: DS   1                          ;BUFFER ENDS
STKLMT: DS   1                          ;TOP LIMIT FOR STACK
;       ORG  1400H
;       ORG  2000H
        ORG  1000H                      ;for 4K system -- 2k ROM, 2K RAM
STACK:  DS   0                          ;STACK STARTS HERE
;
CR      EQU  0DH
LF      EQU  0AH

        END
```