```
;PROGRAM TO TEST LS-100
;BY RVA
;LAST CHANGE
;7/11/83
;COPYRIGHT 1983
;DIGITAL RESEARCH COMPUTERS
;ALL RIGHTS RESERVED
;
;
TRUE    EQU     0FFH            ;VALUE FOR TRUE
FALSE   EQU     0H              ;VALUE FOR FALSE
;
;LS-100 BOARD EQUATES
;
PBASE   EQU     0D0H            ;LS-100 BASE I/O PORT ADDRESS
DATA    EQU     PBASE           ;DATA TRANSFER PORT
TRK     EQU     PBASE+1         ;MSB ADDRESS LATCH
SCTR    EQU     PBASE+2         ;LSB ADDRESS LATCH
;
NUMSCT  EQU     0               ;256 SECTORS/TRACK
NUMBYT  EQU     128             ;BYTES/SECTOR
BDOS    EQU     5               ;CP/M ENTRY
;
LIGHTS  EQU     FALSE           ;SET TO TRUE FOR FRONT PANEL
IMSAI   EQU     TRUE            ;SET TO TRUE FOR IMSAI
;
;
ORG     0100H                   ;CP/M TPA STARTS HERE
        LXI     SP,STACK        ;SET UP PROGRAM STACK
MAIN:   LXI     D,MESG1         ;PRINT SIGN ON MESSAGE
        MVI     C,09            ;BDOS FUNCT.
        CALL    BDOS            ;SIGNON
MAIN1:  LXI     D,MESG2         ;GET BOARD TO TEST
        MVI     C,09
        CALL    BDOS            ;PRINT MESSAGE
        MVI     C,01
        CALL    BDOS            ;GET RESPONSE
        MOV     B,A             ;SAVE A
        SUI     31H             ;SEE IF TO SMALL
        JC      ERR1            ;IF SO PRINT ERROR
        MOV     A,B             ;GET VALUE BACK
        SUI     38H             ;SEE IF TO LARGE
        JNC     ERR1            ;IF SO PRINT ERROR
        MOV     A,B             ;RESTORE A
        STA     BDNO            ;SAVE IT
        SUI     31H             ;REMOVE ASCII BIAS/MAKE RELATIVE
        LXI     H,TAB1          ;HL @ START/END VALUE TABLE
        LXI     D,0             ;CLEAR MSB
        MOV     E,A             ;LSB=A
        DAD     D               ;INDEX INTO TABLE
        MOV     A,M             ;GET START VALUE
        STA     TSTRT           ;STORE TEST START VALUE
        INX     H               ;HL@ END VALUE
        MOV     A,M             ;GET VALUE
        STA     TEND            ;STORE END VALUE
        LXI     D,MESG3
        MVI     C,09
```

```
        CALL    BDOS
MAIN2:  LDA     PATT            ;GET INITIAL PATTERN VALUE
        INR     A               ;INCREMENT VALUE
        STA     PATT            ;STORE IT
        IF      LIGHTS          ;PUT PATTERN
        ENDIF
        IF      IMSAI           ;TO PANEL LIGHTS
        CMA                     ;NEGITAVE TRUE
        OUT     0FFH            ;SET LIGHTS
        ENDIF
        CALL    INIT            ;INITIALIZE LS-100 REGS.
        XRA     A
        CALL    CLEAR           ;CLEAR MEMORY
        CALL    INIT            ;START AT GROUND ZERO
        LDA     PATT            ;GET PATTERN BYTE
F1:     CALL    ABORT           ;CHECK FOR ABORT
        MOV     B,A             ;SAVE PATTERN BYTE
        XRA     A               ;CLEAR A
        MOV     C,A             ;CLEAR C
        MVI     E,NUMBYT        ;E=BYTES/SECTOR
CK1:    IN      DATA            ;MAKE SURE THIS SECTOR IS STILL
        ORA     A               ;CLEAR IN CASE OF ADDRESSING ERROR
        CNZ     ERROR           ;CHECK FDR NON ZERO
        DCR     E               ;DECREMENT BYTE COUNT
        JNZ     CK1             ;CHECK REST
        MVI     E,NUMBYT
        MOV     A,B             ;restore pattern byte
F2:     OUT     DATA            ;write data to ls-100
        INR     A               ;bump pattern byte
        DCR     E               ;decrement byte count
        JNZ     F2              ;check for zero
        DCR     D               ;decrement sector count
        JZ      F3              ;check for zero
        CALL    NSCT            ;bump sector count
        JMP     F1              ;write to all sectors
F3:     MOV     B,A             ;save pattern byte in B
        CALL    NTCK            ;bump track count
        LDA     TEND            ;get end value
        CMP     H               ;check for limit
        JZ      CHECK           ;if at limit check for errors
        MOV     A,B             ;restore pattern byte
        JMP     F1              ;keep going
;
;CHECK MEMORY FDR ERRORS
;
CHECK:  CALL    INIT            ;start at ground zero
        LDA     PATT            ;set pattern byte
        MOV     C,A             ;save in C reg
R1:     IN      DATA            ;get data from LS-100
        CMP     C               ;check it
        CNZ     ERROR           ;log any errors
        INR     C               ;bump pattern byte
        DCR     E               ;decrement byte count
        JNZ     R1              ;loop until 0
        DCR     D               ;decrement sector count
        JZ      R2              ;check for zero
        CALL    NSCT            ;bump sector count
```

```
        JMP     R1              ;read em all
R2:     CALL    NTCK            ;bump track count
        LDA     TEND            ;get end value
        CMP     H               ;check for end
        JNZ     R1              ;read rest
        LHLD    PCNT            ;get value of pass count
        INX     H               ;bump counter
        SHLD    PCNT            ;save pass count
        PUSH    H               ;save on stack
        MOV     A,H             ;move into position
        CALL    CNVHX           ;convert to hex ascii
        SHLD    PCNT1           ;put MSB into message
        POP     H               ;restore value
        MOV     A,L             ;move into position
        CALL    CNVHX           ;convert to hex ascii
        SHLD    PCNT2           ;put lsb into message
        CALL    PMAP            ;print board map
        JMP     MAIN2           ;repeat test
;
;INITIALIZE MSB/LSB REGS. TO ZERO
;SET D=SECTOftS/TRACK, E=BYTES/SECTOR
;
INIT:   XRA     A               ;CLEAR A
        STA     SCNT            ;RESET SECTOR COUNT
        OUT     SCTR            ;SET LS-100 REGISTER
        LDA     TSTRT           ;GET STARTING TRACK #
        OUT     TRK             ;SET LS-100 REGISTER
        STA     TCNT            ;STORE CURRENT TRACK COUNT
        MVI     D,NUMSCT        ;D=SECTORS/TRfiCK
        MVI     E,NUMBYT        ;E=BYTES/SECTOR
        RET
;
;SET MEMORY = TO ACCU.
;
CLEAR:  OUT     DATA            ;WRITE DATA TO LS-100
        DCR     E               ;DECREMENT BYTE COUNT
        JNZ     CLEAR           ;CHECK FOR ZERO
        DCR     D               ;DECREMENT SECTOR COUNT
        JZ      CL2             ;CHECK FOR ZERO
        CALL    NSCT            ;BLiMP SECTOR COUNT
        JMP     CLEAR           ;FILL SOME MORE
CL2:    MOV     B,A             ;SAVE DATA BYTE IN B REG
        CALL    NTCK            ;BUMP TRACK COUNT
        LDA     TEND            ;GET END VALUE
        CMP     H               ;CHECK FOR LIMIT
        RZ                      ;RETURN IF AT END
        MOV     A,B             ;RESTORE DATA BYTE
        JMP     CLEAR           ;MOT DONE YET
;
;INCREMENT SECTOR NUMBER
;
NSCT:   MOV     B,A             ;SAVE DATA BYTE IN B REG
        LDA     SCNT            ;GET CURRENT SECTOR COUNT
        INR     A               ;BUMP COUNT
        STA     SCNT            ;SAVE IT
        OUT     SCTR            ;UPDATE LSB REG.
        MVI     E,NUMBYT        ;E=BYTES/SECTOR
```

```
        MOV     A,B             ;RESTORE A
        RET
;
;INCREMENT TRACK NUMBER AND RESET SECTOR NUMBER
;
NTCK:   XRA     A               ;CLEAR A
        OUT     SCTR            ;SET LSB REG.
        STA     SCNT            ;ZERO SECTOR COUNT
        MVI     E,NUMBYT        ;E=BYTES/SECTOR
        LDA     TCNT            ;GET CURRENT COUNT
        INR     A               ;BUMP COUNT
        STA     TCNT            ;STORE TRACK COUNT
        OUT     TRK             ;SET MSB REGISTER
        MOV     H,A             ;H=CURRENT TRACK
        RET
;
;CHECK KEYBOARD , AND ABORT IF ANY KEY IS PRESSED
;
ABORT:  PUSH    A
        PUSH    H
        PUSH    D               ;SAVE REGISTERS
        MVI     C,0BH           ;CON STATUS FUNCTION
        CALL    BDOS            ;CHECK FOR KEY ENTRY
        ORA     A               ;SET Z BIT
        JNZ     EXIT            ;EXIT IF NOT 0
        POP     D
        POP     H
        POP     A               ;RESTORE REGISTERS
        RET
;
;CLEAR INPUT AND RETURN TO CP/M
;
EXIT:   MVI     C,01            ;GET RID OF INPUT CHAR.
        CALL    BDOS
        POP     D
        POP     H
        POP     A                ;FIX UP STACK
        JMP     0                ;RETURN TO CP/M
;
;PROCESS ERROR
;
ERROR:  PUSH    D               ;SAVE DE
        STA     BAD             ;SAVE BAD BYTE
        LDA     TSTRT           ;GET STARTING TRACK #
        MOV     E,A             ;COPY INTO E
        LDA     TCNT            ;GET CURRENT TRACK #
        SUB     E
        LXI     H,BNK0          ;HL=BASE OF MAP
        LXI     D,8             ;DE=NUM. OF BITS/BANK
        RAR                     ;CONVERT INTO BANK
        ORA     A               ;CHECK FOR BANK ZERO
        JZ      SKP1            ;SKIP REST IF SO
MULU:   DAD     D               ;MAKE HL POINT
        DCR     A               ;TO CORRECT BANK
        JNZ     MULU            ;HL=BANK*BITS
SKP1:   LDA     BAD             ;SET FAILED BYTE
        XRA     C               ;EXCLUSIVE OR WITH GOOD BYTE
```

```
CHK1:   RAR                     ;CHECK FOR BAD BIT(S)
        CC      MKBAD           ;UPDATE MAP
        INX     H               ;POINT TO NEXT BIT IN MAP
        DCR     E               ;CHECK ALL BITS
        JNZ     CHK1
        POP     D               ;RESTORE DE
        RET
MKBAD:  MVI     M,'B'           ;MAKE IT BAD
        RET
;
; VALUE OF BOARD TO TEST OUT OF RANGE
;
ERR1:   LXI     D,MESG4         ;DE=ERROR MESSASE
        PUSH    B               ;SAVE B
        MVI     C,09H           ;SETUP FDR CP/M CALL
        CALL    BDOS            ;PRINT MESSAGE
        POP     B               ;RESTORE B
        JMP     MAIN1
;
;PRINT MEMORY MAP
;
PMAP:   LXI     H,BNK0          ;HL=MAP ARRAY
        LXI     D,HDR1          ;DE=BANK-0 HEADER
        CALL    SMESG           ;PRINT IT
        LXI     D,HDR2          ; DE=BANK-1 HEADER
        CALL    SMESG           ;PRINT IT
        LXI     D,HDR3          ;DE=BANK-2 HEADER
        CALL    SMESG           ;PRINT IT
        LXI     D,HDR4          ;DE=BANK-3 HEADER
        CALL    SMESG           ;PRINT IT
        LXI     D,CRLF          ;SEND A CRLF
        MVI     C,09
        CALL    BDOS
        RET
;
;SEND MESSAGE @DE TO CONSOLE
;
SMESG:  PUSH    H               ;SAVE HL
        MVI     C,09H           ;BDOS FUNCT.
        CALL    BDOS            ;send MESSA6E TO CONSOLE
        POP     H               ;restore HL
        MVI     B,8             ;B=number of bytes to send
;
; send B characters to console
;
SMESG1: MOV     E,M             ;get char from memory
        CALL    PCHAR           ;SEND THEM ONE AT A TIME
        INX     H               ;BUMP POINTER
        DCR     B               ;DECREMENT COUNT
        JNZ     SMESG1          ;CHECK FOR ZERO
        RET
;
; send character in E Register to console
;
PCHAR:  PUSH    H               ;SAVE HL
        PUSH    B               ;SAVE BC
        MVI     C,2             ;BDOS FUNCTION
```

```
        CALL    BDOS            ;PRINT CHAR
        LXI     D,SPC           ;PRINT THREE SPACES
        MVI     C,9             ;BDOS FUNCTION
        CALL    BDOS            ;SEND EM
        POP     B               ;RESTORE REGS
        POP     H
        RET
;
;CONVERT HEX BYTE IN ACCU INTO TWO ASCII CHARS. IN HL
;
CNVHX:  PUSH    PSW
        RRC
        RRC
        RRC
        RRC
        CALL    HASCI
        MOV     L,A
        POP     PSW
        CALL    HASCI
        MOV     H,A
        RET
HASCI:  ANI     0FH
        ADI     90H
        DAA
        ACI     40H
        DAA
        RET
;
;MESSAGE AREA
;
SPC:    DB      20H,20H,20H,'$'
CRLF:   DB      0DH,0AH,'$'
MESG1:  DW      0D0AH
        DW      0D0AH
        DB      ' LS-100 MEMORY DIAGNOSTIC PROGRAM'
        DW      0D0AH
        DW      0D0AH
        DB      '    DIGITAL RESEARCH COMPUTERS'
        DW      0D0AH
        DB      '        ALL RIGHTS RESERVED'
        DW      0D0AH
        DB      '          COPYRIGHT 1983'
        DW      0D0AH
        DB      '           Version 1.0'
        DW      0D0AH
        DW      0D0AH
        DB      '$'
MESG2:  DB      'ENTER BOARD NUMBER TO TEST (1-8): '
        DB      '$'
MESG3:  DW      0D0AH
        DW      0D0AH
        DB      'PRESS ANY KEY TO EXIT TO CP/M'
        DW      0D0AH
        DB      '$'
MESG4:  DW      0D0AH
        DW      0D0AH
        DB      'VALUE OF BOARD TO TEST MUST BE IN RANGE OF 1 - 8'
```

```
        DW      0D0AH
        DB      'RE-ENTER VALUE OF BOARD TO TEST'
        DW      0D0AH
        DW      0D0AH
        DB      '$'
HDR1:   DW      0D0AH
        DB      'TEST RESULTS FOR BOARD '
BDNO:   DB      0
        DW      0D0AH
        DW      0D0AH
        DB      'PASS NUMBER '
PCNT1:  DW      0
PCNT2:  DW      0
        DB      'H'
        DW      0D0AH
        DW      0D0AH
        DB      '      DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7'
        DW      0D0AH
        DB      'BANK-0 '
        DB      '$'
HDR2:   DW      0D0AH
        DB      'BANK-1 '
        DB      '$'
HDR3:   DW      0D0AH
        DB      'BANK-2 '
        DB      '$'
HDR4:   DW      0D0AH
        DB      'BANK-3 '
        DB      '$'
;
;PROGRAM STORAGE AREA
;
PCNT:   DW      0       ;CURRENT PASS COUNT
SCNT:   DB      0       ;CURRENT SECTOR #
TCNT:   DB      0       ;CURRENT TRACK #
PATT:   DB      0FFH    ;STARTING TEST BYTE
BAD:    DB      0       ;BAD BYTE FROM LS-100
TSTRT:  DB      0       ;STARTING TRACK # TO TEST
TEND:   DB      0       ;ENDING TRACK # TO TEST
;
;TABLE FOR START END VALUES
;
TAB1:   DB      00H
        DB      08H
        DB      10H
        DB      18H
        DB      20H
        DB      28H
        DB      30H
        DB      38H
        DB      40H
;
;MEMORY MAP ARRAY STORAGE
;
;               DB0 DB1 DBS DB3 DB4 DB5 DBS DB7
BNK0:   DB      'G','G','G','G','G','G','G','G'
BNK1:   DB      'G','G','G','G','G','G','G','G'
```

```
BNK2:   DB      'G','G','G','G','G','G','G','G'
BNK3:   DB      'G','G','G','G','G','G','G','G'
;
;PROGRAM STACK AREA
;
        DS      100
STACK   EQU     $
        END
```